Università
Ca'Foscari
Venezia

Corso di Dottorato di ricerca
in Computer Science
ciclo XXXII

Tesi di Ricerca

Semi-Supervised Learning and Applications: a Game-Theoretic Perspective

SSD: INF/01

Coordinatore del Dottorato
ch. prof. Agotino Cortesi

Supervisore
ch. prof. Marcello Pelillo

Dottorando
Sebastiano Vascon, PhD
Matricola 788442

Abstract

The fields of pattern recognition and machine learning are concerned with the automatic discovery of regularities in data and with the use of these regularities to take actions such as perform a data classification into different categories (supervised learning) or separates data into meaningful groups (unsupervised learning). This thesis is focused on semi-supervised learning (SSL) algorithms, a family of methods lying in between supervised and unsupervised learning. The main characteristic of SSL algorithms is that they exploit at the same time the structure of the data (their features) and the available labeling information, to estimates the boundaries of the classes/clusters. For this reason, they are particularly suitable in a regime of scarcity of labeled data or in the cases whether the data annotation is expensive or time-consuming. Here, in particular, we will exploit a recent algorithm rooted in the evolutionary game-theory, named "Graph Transduction Games" (GTG). The GTG algorithm explicitly models a semi-supervised learning problem as a non-cooperative game where players represent the data and the strategies the possible labels. A player chooses a strategy and receives a payoff which is proportional to the choice of the other players and to their similarities. The game is iterated until all the players have chosen their best strategy, and no one has any incentive to change his/her choice. Such a game is modeled through a dynamical system, shifting the established paradigm from local optima of a cost function to an equilibrium condition in a non-cooperative game. During the labeling process, the similarities between all the data are taken into account, creating a context in which similar points affect each other in deciding the final labeling assignment. The neighboring players (data), hence the context, help in situations in which intrinsic ambiguities in the data may lead to inconsistent class assignments. Within this thesis, the GTG algorithm and the context in which players are playing will be explored into applications like bioinformatics, natural language processing, computer vision, and pure machine learning problems.

Abstract

Il riconoscimento di pattern e l'apprendimento automatico riguardano la scoperta automatica delle regolarità nei dati e dell'uso di queste regolarità per intraprendere azioni quali la classificazione degli stessi in diverse categorie (apprendimento supervisionato) o la separazione dei dati in gruppi significativi (apprendimento non supervisionato). Questa tesi è incentrata sugli algoritmi di apprendimento semi-supervisionato, una famiglia di metodi che si colloca tra l'apprendimento supervisionato e l'apprendimento non supervisionato. La caratteristica principale degli algoritmi semi-supervisionati è che sfruttano allo stesso tempo la struttura dei dati (le loro features) e le informazioni di etichettatura disponibili, per stimare le zone di separazione delle classi o dei clusters. Per questo motivo, sono particolarmente adatti in un regime di scarsità di dati etichettati o nei casi in cui l'annotazione sia particolarmente dispendiosa in termini economici o di tempo. Nello specifico, in questa tesi, sfrutteremo un recente algoritmo basato sulla teoria dei giochi evoluzionistica, denominato "graph transduction games". Tale algoritmo modella esplicitamente problemi di apprendimento semi-supervisionato come un gioco non-cooperativo dove i gicatori sono i dati e le strategie le possibili etichette. I giocatori ricevono un guadagno proporzionale alle similarità tra i giocatori stessi e alla strategia scelta da loro e dai loro avversari. Il gioco prosegue finche tutti i giocatori non avranno scelto la loro miglior strategia (etichetta) e non cambieranno piu idea. Tale gioco viene modellato attraverso un sistema dinamico, spostando quindi il paradigma dall'ottimizzazione di una funzione di costo alle condizioni di equilibrio in un gioco non-cooperativo. Durante il processo di etichettatura, le similarità tra tutti i dati vengono prese in considerazione, creando un contesto in cui punti simili si influenzano a vicenda nel decidere l'assegnazione finale dell'etichettatura. I giocatori (dati) vicini tra loro, e quindi il contesto, sono di aiuto in situazioni in cui le ambiguità intrinseche nei dati possono portare a risultati incoerenti. All'interno di questa tesi, l'algoritmo GTG e il contesto in cui i giocatori stanno giocando saranno esplorati in applicazioni come la bioinformatica, l'elaborazione del linguaggio naturale, la visione artificiale, e problemi di puro apprendimento automatico.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is centered on semi-supervised learning algorithms and their usage in different fields of computer science. The semi-supervised learning (SSL) is a branch of machine learning which lies in between *unsupervised* and *supervised* learning [19]. The goal of SSL is to achieve better separation boundaries for a classifier/clustering algorithm by exploiting, at the same time, the structure of both labeled and unlabeled points. Thus, SSL is particularly useful when the amount of labeled data is smaller, compared to the unlabeled one. Different SSL algorithms are explored within this thesis (see Chapter 2), with a particular emphasis on the graph-based ones (see Section 2.2.1) since are the direct competitors of the chosen method for this work. The graph-based SSL methods represent both labeled and unlabeled points in terms of an un/weighted graph, while the labeling information is spread into the graph exploiting its connectivity. Graph-based methods have also the advantage of modeling the interaction between pairs of data points through the graph edges, creating an explicit context where data interact or not. In SSL we made a distinctions between *transductive* and *inductive* algorithms (see Section 2.1.2): the first refers to methods that directly label the unlabeled points using both labeled and unlabeled data jointly, while the latter requires the learning of mapping (a function) between data representation (features) and labels to perform the inference. The graph-based methods mainly belong to the family of the transductive algorithm. In particular, the emphasis of this thesis is pointed to a recent graph-based SSL algorithm, called called "*Graph Transduction Game*" [35] (GTG). The GTG proved to be competitive or even outperforming traditional graph-based semi-supervised learning algorithms (see Part I, Part II and in particular the Chapter 4). The GTG is a semi-supervised algorithm that casts a transductive process into a non-cooperative game (see Chapter 3). In such a game, the players are divided into labeled and unlabeled ones representing the points of a dataset. The strategies that each player can choose are the possible labels, while the payoffs (the gain received by a player when playing a particular strategy) are proportional to the similarity between the opponents. Indeed, the goal of each player is to maximize his/her reward considering the choices made by the opponents. The more two players are similar, the more they will support each other in picking the same strategy (class). The final outcome of the game, which is the assignment between players and strategies, is obtained through a dynamical system named *replicator dynamic* (RD). The RD simulates the interactions between players and their preferences toward specific strategies. The game is played until all the players are satisfied with their chosen strategy (label), and no one wants to change the choice. The condition above is known as *Nash Equilibrium* (NE). The final labeling is then the results of the interaction between all the players, explicitly considering the context (neighbors) in which each player is playing.

1.1 Thesis Contributions and Structure

This thesis contributes the field in different directions, both *methodological* and *applicative*. From a methodological point of view, we fruitfully embed game-theory in known methods based on SSL algorithms, obtaining remarkable results. In particular, we show the power and simplicity of modeling SSL problems in terms of non-cooperative games and that the paradigm shift from traditional function minimization/maximization to equilibrium condition in a dynamical system is solid and leads to superior performances. We outperform previous state-of-the-art methodologies in tasks like Unsupervised Domain Adaptation (Chapter 4), Non-Negative matrix factorization (Chapter 5) and we propose a new model to train deep neural network in the absence of huge amount of labeled data (Chapter 6). From an applicative perspective, we showed that transductive methods based on games could be applied successfully in the domains of bioinformatics, natural language processing, and computer vision. In particular, we faced the highly complex tasks of protein function prediction (Chapter 7), the disambiguation of verbs using textual and visual features (Chapter 8) and, in the context of computer vision, offering a newer perspective in the task of ancient coin recognition (Chapter 9).
The organization of the thesis is the following: there are two introductory chapters on semi supervised learning, and on the graph transduction game method. In the Chapter 2, we briefly review what is semi-supervised learning, its properties, and some known models and algorithm in the state of the art. The Chapter 3 deepen the algorithm used among this thesis, the GTG. Then the contributions are divided into two parts, the *methodological* (see Part I) and the *applicative* (see Part II), respectively.

1.2 Publications

The results of this "*PhD journey*" are resumed here.

International Journals (peer reviewed)

J4 M. Denitto, M. Bicego, A. Farinelli, ***S. Vascon***, and M. Pelillo. Biclustering with dominant sets. *Pattern Recognition*, 2019. ***under revision***

J3 S. Aslan, ***S. Vascon***, and M. Pelillo. Two Sides of the Same Coin: Improved Ancient Coin Classification Using Graph Transduction Games. *Pattern Recognition Letters*, 2019. ***accepted for publication***

J2 G. Sandi, ***S. Vascon***, and M. Pelillo. Hypergraph Isomorphism Using Association Hypergraphs. *Pattern Recognition Letters*, 2019. ***accepted for publication***

J1 ***S. Vascon***, M. Frasca, R. Tripodi, G. Valentini, and M. Pelillo. Protein function prediction as a graph-transduction game. *Pattern Recognition Letters*, 2018. DOI:10.1016/j.patrec.2018.04.002. *in press*

Book Chapters (peer reviewed)

B1 ***S. Vascon*** and M. Pelillo. Detecting conversational groups in images using clustering games. In X. Alameda-Pineda, E. Ricci and N. Sebe (Eds.), *Multi-modal Behavior Analysis in the Wild: Advances and Challenges*, pp. 247–267, Academic Press - Elsevier, 2019.
DOI:10.1016/B978-0-12-814601-9.00024-9

International Conferences (peer reviewed)

C9 ***S. Vascon****, G. Bigaglia*, L. Giudice, and M. Pelillo. Multimodal Verb Sense Disambiguation using Graph Transduction Games. ***under submission***. *= equal contribution

C8 I. Elezi, ***S. Vascon***, A. Torcinovich, M. Pelillo and L. Leal-Taixé. The Group Loss for Deep Metric Embedding. ***under revision***

C7 ***S. Vascon***, S. Aslan, A. Torcinovich, T. van Laarhoven, E. Marchiori, and M. Pelillo. Unsupervised Domain Adaptation using Graph Transduction Games. IEEE International Joint Conference on Neural Networks, IEEE, 2019. **Oral presentation** .

C6 S. Aslan, ***S. Vascon***, and M. Pelillo. Ancient coin classification using graph transduction games. *In Proc. of the IEEE 4th International Conference on Metrology for Archaeology and Cultural Heritage*, IEEE, 2018. **Oral presentation** .

C5 ***S. Vascon****, Y. Parin*, E. Annavini*, M. D'Andola, D. Zoccolan, and M. Pelillo. Characterization of visual object representations in rat primary visual cortex. *In Proc. of European Conference on Computer Vision Workshop (ECCVw)*, Lecture Notes in Computer Science 11131, Springer, 2018. *= equal contribution.

C4 G. Sandi, ***S. Vascon***, and M. Pelillo. On association graph techniques for hypergraph matching. *In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 481-490. Springer, 2018. **Oral presentation** .

C3 I. Elezi*, A. Torcinovich*, ***S. Vascon****, and M. Pelillo. Transductive label augmentation for improved deep network learning. *In Proc. of the 24rd International Conference on Pattern Recognition (ICPR)*, pp. 1432-1437, IEEE, 2018.*= equal contribution.

C2 F. Hibraj*, ***S. Vascon****, T. Stadelmann, and M. Pelillo. Speaker clustering using dominant sets. *In Proc. of the 24rd International Conference on Pattern Recognition (ICPR)*, pp. 3549-3554, IEEE, 2018. *= equal contribution.

C1 R. Tripodi*, ***S. Vascon****, and M. Pelillo. Context aware nonnegative matrix factorization clustering. *In Proc. of the 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1719–1724. IEEE, 2016. *= equal contribution.

Chapter 2

Semi-supervised learning

2.1 Introduction

Semi-supervised learning (SSL) is a class of machine learning algorithms which lies in between *unsupervised* and *supervised* methods. Unsupervised learning regards techniques in which data have no labels associated, and a typical goal is to inspect the hidden structure of the data. For example, perform partitioning on dataset such that similar data are into clusters [37, 122] or find a meaningful low-dimensional representation in which the core features emerge [73]. On the other side, the aim of *supervised* learning methods aim at learning a mapping between data represented in terms of multidimensional features and their labels. Once trained, the algorithms in this class are able to infer classes to objects for which a label is not provided. Examples of such methods are the support vector machines (SVM) [156], k nearest neighbors [31], artificial neural networks, including recent deep-learning techniques [55].
The goal of SSL is to exploits, at the same time, the labeled and the unlabeled portions of a dataset to accomplish a particular task, for example, clustering or classification.
The chapter is organized as follows: in Section 2.1.1 we review the core assumptions of SSL, in Section 2.2 we report the more interesting method in the literature, with particular emphasis on the graph-based ones.

2.1.1 Assumptions

The semi-supervised learning algorithm lies on particular assumptions that must hold to make them possible to work.

Smoothness assumption

In the presence of a *smoothness assumption*, we assume that given two samples x_1 and x_2, their classes y_1 and y_2 are tightly related to the closeness of the samples itself. We can say that the output of a classifier changes *smoothly* with the distance of the samples. Indeed, this assumption holds particularly in the case of *supervised* learning, allowing us to generalize to possibly infinite test samples. In the case of SSL, the

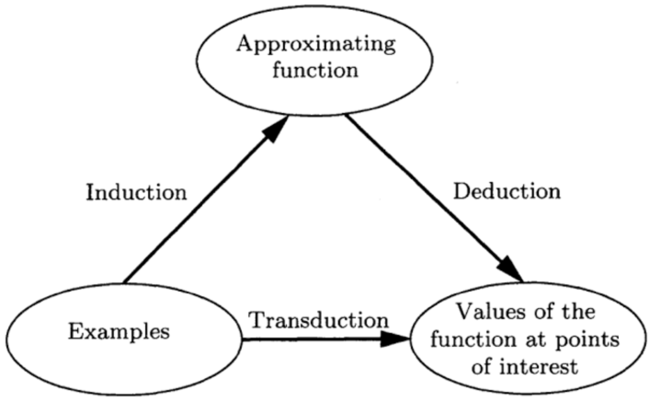


Figure 2.1: Difference between induction and transduction. Image from [155]

assumption mentioned above can be rephrased in this way: ”If two points x_1, x_2 in a **high-density region** are close, then so should be the corresponding outputs y_1, y_2” [19]. This assumption implies that points belonging to the same cluster (a high-density region in the feature space) are likely to belong to the same class. If points are separated by a region with low-density, the output of an SSL algorithm (label or regression) should be different.

Cluster assumption

The cluster assumption states the following: ”*If points are in the same cluster, they are likely to be of the same class.*” [19]. Even if this assumption could sound similar to the *smoothness* ones, here we are pointing to the notion of continuity rather than smoothness. In fact, it is unlikely that a densely populated continuous set of objects can be divided into different labels. In other words, objects having different labels cannot be part of the same cluster. The cluster assumption can be seen as a special case of the smoothness assumption [19].

Manifold assumption

Here we assume that data intrinsically lies on a manifold of lower dimension compared to its original input representation. This assumption is important to avoid the *course of dimensionality*[1] problem, for which most of the statistical methods suffer.

2.1.2 Transductive vs Inductive learning

The *transductive learning* was introduced by Vapnik in [48] with the idea of predicting the labels directly for a test set, using an annotated training set. On the other hand, an inductive process learns a model/function from the available training data, and later perform the classification on the unlabeled data (see Figure 2.1). Although the two approaches might look similar, they are indeed radically different because in the inductive approach a general rule is learned and the test data is seeing only during inference. Indeed, an *inductive* process allows generalizing to test dataset of potentially infinite size since a general rule is learned, while a *transductive* one requires the entire labeled and unlabeled data to perform inference on the unlabeled part.

2.2 Models

In the following sections, we present different semi-supervised learning algorithms models with a particular emphasis on ones based on graph since more related to the aim of this thesis.

2.2.1 Graph-based models

Modeling an SSL task in terms of graph is quite natural, in particular if a transductive paradigm holds. In the literature methods that use such structure are abundant, here we report the well-known ones, the *label propagation*, the *label spreading* and the *harmonic function* while the core method of this thesis, the "*graph transduction game*" [35], is deepened in the dedicated Chapter 3. In Chapter 4 the aforementioned algorithms are empirically compared, showing the superiority of [35].
The common idea of graph-based methods is to build a graph in which nodes are both labeled and unlabeled data points, and edges represent pairwise similarities. Labeling information, associated with labeled nodes, are then propagated through the graph structure to assign a label to the unlabeled vertices. In general, graph-based methods hold the cluster assumption for obvious reasons[2] and, if the similarity function between nodes represents the manifold where they lie (for example a path-based similarity), then we can even state that, the particular method under exams, holds the manifold assumption.

Label Propagation

The label propagation algorithm has been proposed by Zhu et al [181]. The underlying idea is to propagate labeling information from a set of (small) labeled points to the unlabeled ones. The propagation is performed considering a graph-based structure in which the edges weight between pairs reflects their similarity. Like other graph-based methods, the main assumption is that closer data points tend to have similar class labels.

[1]The space grows exponentially with the number of dimensions, hence we need an exponentially number of samples to have a statistical coverage for the entire space. Furthermore, in high-dimensional space the distances become meaningless.

[2]similar nodes tend to lies in close proximity thus forming clusters

Being more formal, given a set of labeled points $L = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$ and a set of unlabeled ones $U = \{(x_{l+1}, y_{l+1}), \dots, (x_{l+u}, y_{l+u})\}$ the goal is to assign a class to each $x_i \in U$. Here, x_i and y_i are the d-dimensional descriptor (feature) and the label of the i-th point, respectively. The labeling information is represented as a probability vector over the possible classes. Here is assumed that the number of classes C is known in advance, hence $y_i \in \Delta^C$ the C-dimensional simplex. Then, a weighted graph $G = (V, E, \omega)$ is build on top of the data points. Here, $V = L \cup U$, $E \subseteq V \times V$ is the set of edges and $\omega : (i, j) \in E \to \mathbb{R}_{\geq 0}$. The weighting function ω quantifies the similarity between pair of nodes:

$$\omega_{i,j} = e^{-\frac{\|x_i - x_j\|_2^2}{\sigma^2}} \tag{2.1}$$

The labels are propagated to all nodes through the edges, considering their weight. The more a pair of node is similar the more the label will flows. A transition matrix T of size $((l + u) \times (l + u))$ accounts for the likelihood of going from a node to another

$$T_{i,j} = P(j \to i) = \frac{\omega_{i,j}}{\sum_{k=1}^{l+u} \omega_{k,j}} \tag{2.2}$$

Thus $T_{i,j}$ is the probability of jumping from node j to i. The union of all labeling assignment y_i forms a stochastic matrix Y of size $(l + u) \times C$. The propagation takes into account the similarities between pair of nodes and their labeling preferences.
The algorithm is then composed by the following three steps

1 Compute $Y \leftarrow T \times Y$

2 Normalize Y such that each row adds-up to 1

3 Reset the rows of Y belonging to labeled samples to one-hot vector in the correct class.

which are repeated until convergence of matrix Y.

The final classification is then performed by *argmax* the rows of matrix Y.

Label Spreading

The label spreading algorithm [180] is similar in spirit to *label propagation*. A similarity graph $G = (V, E, \omega)$ is constructed on top of both labeled and unlabeled data points, following the same schema of [180]. Then the normalized Laplacian of the similarity graph is computed:

$$S = D^{-1/2} W D^{-1/2}$$

where W is the pairwise similarity matrix of the nodes and $D_{ii} = \sum_{j=1}^{n} \omega_{i,j}$.
The iterative procedure is slightly different from [180] and is composed as follow:

$$F(t + 1) = \alpha S F(t) + (1 - \alpha)Y$$

where S is the normalized Laplacian, Y is the stochastic matrix containing the labeling assignment and F maintain the current labeling assignment, $F(t = 0) = Y$.

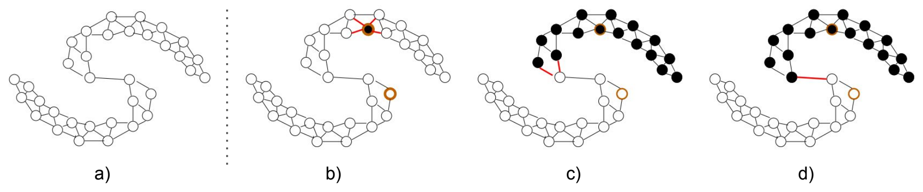


Figure 2.2: From left to right: a) the case in which no labeled data points are used $E(\mathbf{y}) = 0$, b) then a pair of labeled data are injected. The energy in presence of labeled data points decreases (from b-d) until the minimum, $E(\mathbf{y}) = 4$ in b) to $E(\mathbf{y}) = 1$ in d) [3]

During each update of F, every vertex $i \in V$ receives the information from its direct neighbors (first term of F: $\alpha SF(t)$), while keeping the initial labeling information (second term of F: $(1 - \alpha)Y$). The parameter α has the role of balancing how much the initial labeling information weighs compared to the one gained from the graph.
At convergence of F the final labeling is obtained by its *argmax*.

Gaussian Fields and Harmonic Functions

The *Gaussian Fields and Harmonic Functions* method [183], like in the previous graph-based methods, relies on a similarity graph constructed on labeled and unlabeled data points. The rationale here is that ”...we want unlabeled points that are nearby in the graph to have similar labels.” The resulting method is similar to a nearest neighbor classification approach, where the nearest labeled examples are computed in terms of a random walk on the weighted graph. Differently from the previous algorithms, a quadratic energy function representing nearby nodes, is constructed and further minimized.

$$E(\mathbf{y}) = \frac{1}{2}\sum_i \sum_j \omega_{ij}(y_i - y_j)^2$$

here ω_{ij} is the weight of the edge connecting node i and j. The variable $\mathbf{y} \in \{0, 1\}^{|V|}$ represents the assignment of all nodes to a class.
Indeed, if we consider the weights $\omega \in \{0, 1\}$ and no labeled data, the minimum of the energy is obtained when all the nodes have the same class. If a certain amount of labeled information is injected, the energy at the beginning is high while decreasing until a point of separation between classes is reached. See figure 2.2 for an example. The authors further relax the binary constraint on $\mathbf{y}$ and proposes a closed form solution.

2.2.2 Low Density Region-based models

The underlying idea of models within this class is that the *decision boundary lies in regions with low density*. The motivation is the following, a low-density region is an area in which fewer samples fall, which means that it is an area that separates different classes/clusters. To give an example, consider the task of recognizing ”cats” between ”chairs”. A sample which falls in between the two classes should be similar to both a ”cat” and a ”chair”, which is indeed something pretty unlikely to happen, that is why

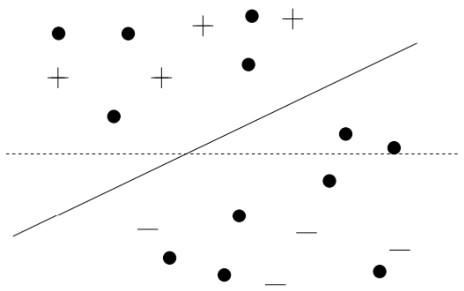


Figure 2.3: The dashed line represents the hyperplane considering a standard inductive SVM trained on the labeled instances. The solid line is the separator corresponding to the transductive SVM, in which the unlabeled data are also considered during the parameter optimization. Figure from [72].

the region between the two classes will be sparsely populated.
The most common approach falling in this category is the usage of the notion of *maximum margin*, like in the *Support Vector Machines* (SVM). A well-known algorithm within this family is the *Transductive Support Vector Machine* (TSVM) [71].

Transductive SVM

A Transductive support vector machine (TSVM) implements the idea of transductive learning including test points in the computation of the margin. See figure 2.3 for an example. The algorithm starts by training an (inductive) SVM on the training set which is used to assign labels to an unlabeled set. The unlabeled points with the new labels are then used in conjunction with the training set to retrain the margin. This is performed multiple times while slowly increasing the weights of the unlabeled set.

2.2.3 SSL and Deep Neural Network

Recently, semi-supervised learning has seen a revival within the deep learning (DL) field [119]. The motivations are mainly due to the fact that DL requires a lot of labeled data, while SSL roots his assumption in a strong imbalance between labeled information and unlabeled ones. For this reason, scientists start asking whether SSL can be applied to DL architectures to learn models in scarcity of labeled data. In this section, we report some recent approaches that exploit semi-supervised techniques to train deep-neural network classifiers or that used directly unlabeled data during training:

Pseudo-labeling

Pseudo labels [94] is one of the simpler, and widely used techniques to use unlabeled data to train a deep neural network model or, more in general, any inductive model. The idea is to use the small amount of labeled information to train a classifier (DNN, SVM etc.), then use the trained model to predict the classes of the unlabeled samples.

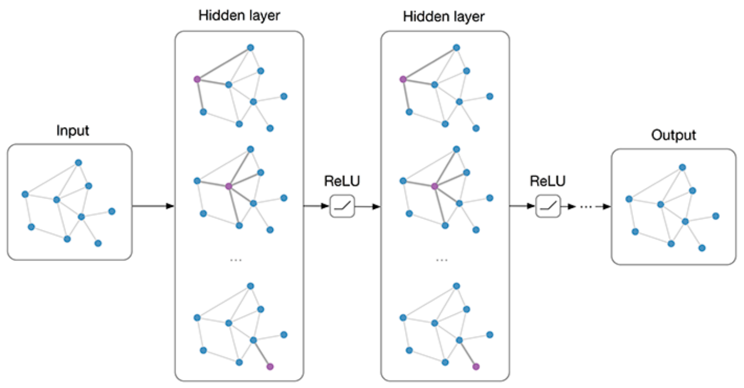


Figure 2.4: An example of a graph convolutional neural network. Image from [82].

The newly labeled set is added to the original (small) training set, and the model is further retrained.

Label Propagation and DNN Learning

In [69], the authors propose to use the label propagation algorithm to generate pseudo-labels and further label the unlabeled part of a dataset. Then, the newly labeled dataset is used to train a deep neural network.
This approach is similar to ours (see Chapter 6) with two main differences: *i)* the usage of label propagation algorithm instead of the graph transduction game and *ii)* a weighting mechanism for each sample which considers at the same time the confidence towards a particular label and the classes numerosity accounting for class unbalances. Then the weighing mechanism is used during training in the loss function

SSL and Graph Neural Networks

Another exciting area of research in which SSL is growing is the Graph Neural Network [167, 179], and in particular, the Graph Convolutional Neural Network (GCNN) [82]. The GCNNs are a very powerful neural network architecture to deal with data structured in terms of a graph and to produce useful feature representations of its nodes. Here we briefly report the work proposed in [82] as a seminal work combining graph neural network and semi-supervised learning. Given a graph $G = (V, E)$, where V is the set of vertices of size $n = |V|$ and E the edges connecting pair of vertices, a GCNN takes as input a feature matrix X of size $n \times d^0$ (d^0 is the dimension of the features) and an adjacency matrix A of size $n \times n$ representing the node connectivity. Each hidden layer produces as output a new feature representation for all nodes Z (an $n \times d^1$ where d^1 is the number of output features per node). Then this representation is aggregated to generate the features for the next layer. By stacking multiple hidden layers, the generated features become increasingly more abstract capturing the structure of the graph.

Ladder Network

The Ladder Network [127] (LN) combines both unsupervised and supervised learning to jointly train a deep neural network (mainly autoencoders) using labeled and unlabeled data. The underlying principle of the LN is to add an additional decoder to a supervised neural network classifier in order to exploits the unlabeled data. A LN is mainly composed by an encoder-decoder structure in which the encoder is a feed-forward network that we want to train in a supervised way. Then a LN is made of two encoder paths, the first *clean* and the latter *corrupted* by a Gaussian noise at each encoder layer. Each layer of the two branches are then connected to a decoder, with the role of reconstructing the input signal, this is the unsupervised part of the LN. The sum of the differences between the output of the *clean* reconstruction and the *corrupted* ones corresponds to the cost of the unsupervised branch. The cost of the supervised branch is calculated from the output of the corrupted encoder and the output target. The sum of the two costs represents the loss of the network which is used for the training with the backpropagation algorithm.

Chapter 3

Graph Transduction Games

3.1 Introduction

In this chapter, the *Graph Transduction Game* (GTG) algorithm is deepened and dissected in its components. In Section 3.2 we provide a brief introduction to game-theory, being the foundation of the aforementioned algorithm, while in Section 3.3 the method is explained in detail.

3.2 Game Theory

Game theory (GT) was introduced by [162] in order to develop a mathematical framework able to model the essentials of decision making in interactive situations. In its *normal-form* representation, it consists of a finite set of players $I = \{1, .., n\}$, a set of pure strategies for each player $S = \{s_1, ..., s_m\}$, and a utility function $u : S_1 \times S_2 ... \times S_n \rightarrow \mathbb{R}$, which associates strategies to payoffs. Here we assume that all the players have the same set of strategies S, but in the more general formulation this is not mandatory. Each player can adopt a strategy in order to play a game and the utility function depends on the combination of strategies played at the same time by the players involved in the game, not just on the strategy chosen by a single player. An important assumption in game theory is that the players try to maximize their utility u. Furthermore, in *non-cooperative games*, the players choose their strategies independently, considering what other players can play in order to find the best strategy profile to employ in a game. Nash Equilibria (NE) [117] represent the key concept of game theory and can be defined as those strategy profiles in which each strategy is the best response to the strategy of the co-player and in which no player has the incentive to unilaterally deviate from his decision (the players are in equilibrium). The NE of a game exists in two forms: *i)* pure-strategy and *ii)* mixed-strategy. In a pure-strategy NE, each player adopts only one strategy while in the latter case is a probability distribution among the possible strategies. A mixed strategy for a player is defined as a stochastic column vector $\mathbf{x} = (x^1, \dots, x^m) \in \Delta^m$, where m is the number of pure strategies and each component x^h denotes the probability that a particular player chooses its h-th pure strategy.

Each mixed strategy corresponds to a point in the m-dimensional simplex Δ^m defined as,

$$\Delta^m = \Big\{x \in \mathbb{R} : \sum_{h=1}^{m} x^h = 1, x^h \geq 0, \forall h\Big\}, \tag{3.1}$$

whose corners correspond to pure strategies (pure strategy NE can be seen as an extremal case of mixed-strategies).
In a *two-player game*, a strategy profile can be defined as a pair $(\mathbf{x}_i, \mathbf{x}_j)$ where $\mathbf{x}_i \in \Delta^m$ and $\mathbf{x}_j \in \Delta^m$. The expected payoff for this strategy profile is computed as:

$$\begin{aligned} u(\mathbf{x}_i, \mathbf{x}_j) &= \mathbf{x}_i^T A_{ij} \mathbf{x}_j \\ u(\mathbf{x}_j, \mathbf{x}_i) &= \mathbf{x}_j^T A_{ji} \mathbf{x}_i \end{aligned} \tag{3.2}$$

where A_{ij} (conversely A_{ji}) is the $m \times m$ payoff matrix of the game between player i and j. Each entry (h, k) of the payoff matrix A_{ij} corresponds to the gain received by player i when he plays strategy h against strategy k.

The strategy space of each player i is defined as a mixed strategy $\mathbf{x}_i$, as defined above. The payoff corresponding to the h-th pure strategy can be computed as:

$$u(x_i^h) = \sum_{j=1}^{n} (A_{ij}\mathbf{x}_j)^h \tag{3.3}$$

while the expected payoff of the entire mixed-strategy for player i is:

$$u(\mathbf{x}_i) = \sum_{j=1}^{n} \mathbf{x}_i^T A_{ij} \mathbf{x}_j \tag{3.4}$$

where n is the number of players with whom i plays and $A_{i,\_}$ is their payoff matrix of the game. Given these two functions we can find the NE of the game, and to this end we will use a result in the domain of Evolutionary Game Theory (EGT). The EGT, introduced by [108], is a branch of game theory which aims to use the notions of GT to model the evolution of behavior in animal conflicts. In EGT we have a set of agents which play games repeatedly with their neighbors and update their beliefs on the state of the system choosing their strategy according to what has been effective and what has not in previous games. This loop is repeated until the system converges, which means that no player needs to update its strategies because there is no way to do better. To find those states, which correspond to the NE of the game, we use the *replicator dynamics* [165]:

$$x_i^h(t+1) = x_i^h(t) \frac{u(x_i^h)}{u(\mathbf{x}_i)} \;\forall h \in S \tag{3.5}$$

The replicator equation allows better than average strategies to grow at each iteration, hence each iteration can be considered as an *inductive learning process*, in which the players learn from the others how to play their best strategy in a determined context (see bottom part of Fig.7.1). The complexity of each step of the replicator dynamics

(Eq.3.5) is quadratic but there are different dynamics that can be used with our framework to solve the problem more efficiently, such as the recently introduced *infection and immunization* dynamics [131] that has a linear-time/space complexity per step and it is known to be much faster than, and as accurate as, the replicator dynamics.

3.3 Graph Transduction and Game Theory

3.3.1 Graph Transduction

Graph transduction is a semisupervised learning technique that aims at estimating a classification function defined over a graph of labeled and unlabeled data points. Models based on this technique use a graph to represent the data, with nodes corresponding to labeled and unlabeled points and edges encoding the pairwise similarity among each pair of nodes. This technique works propagating the label information from labeled nodes to unlabeled, exploiting the graph structure.

It was introduced by [156] and motivated by the fact that it is easier than inductive learning, because inductive learning tries to learn a general function to solve a specific problem, while transductive learning tries to learn a specific function for the problem at hand.

Graph transduction consists of a set of labeled objects (x_i, y_i) $(i = 1, 2, ..., l)$, where $x_i \in \mathbb{R}^n$ the real-valued vector describing the object i, and $y_i \in (1, ..., m)$ its label, for $i \in \{1, 2, \dots, n\}$, and a set of k unlabeled objects $(x_{l+1}, ..., x_{l+k})$. Rather than finding a general rule for classifying future examples, transductive learning aims at classifying only (the k) unlabeled objects exploiting the information derived from labeled ones.

Within this framework, it is common to represent the geometry of the data as a weighted graph. For a detailed description of algorithms and applications on this field of research, named graph transduction, we refer to [181]. Formally we have a graph $G = (V, E, w)$ in which V is the set of nodes representing both labeled and unlabeled points, E is the set of edges connecting the nodes of the graph and $w : E \to \mathbb{R}_{\geq 0}$ is a weight function assigning a non-negative similarity value to each edge $\epsilon \in E$. The task of transduction learning is to estimate the labels of the unlabeled points given the pairwise similarity among the data points and a set of possible labels.

3.3.2 Graph Transduction Game

The graph transduction game (GTG) algorithm [35] interprets the graph transduction task as a non-cooperative multiplayer game with a solid mathematical foundation rooted in game theory. Two attractive aspects of GTG drive our choice to this algorithm: *i)*it does not impose any constraint on the pairwise similarity function used to weight the graph and *ii)* it allows the injection of prior knowledge on the data labeling. Classical graph transduction algorithms are based on the homophily principle [72, 177, 183], that simply states that *similar data points* are expected to have the *same class*. We found this assumption too strong for the more general transductive task and for this reason we extended it using the approaches proposed in [149, 150] that is reminiscent of the Hume association principle [66], that states that *similar objects* are expected to have similar properties and hence to belong to *similar classes*. With this approach we are able to exploit two sources of information: the *similarity among the data points*, as in classical graph transduction approaches and the *similarity among their classes*. With the latter source of information it is possible to build a structural classifier that produce consistent labeling of the data according to information provided by an ontology that encodes both information about the classes and their reciprocal relations. In the Chapter 7 we will show that these information can be easily embedded in a game-theoretical framework as part of the payoff function.

In graph transduction game, objects of a dataset are represented as players and their labels as strategies. In synthesis, a non-cooperative multiplayer game is played among the objects, until an equilibrium condition is reached, the *Nash Equilibria* [117]. Given a set of players $I = \{1, \dots, n\}$ and a set of possible pure strategies $S = \{1, \dots, m\}$:

1 *mixed strategy*: a mixed strategy x_i is a probability distribution over the possible strategies (labels) for player (object) i. Then $x_i \in \Delta^m$, where

$$\Delta^m = \left\{ \sum_{h=1}^{m} x_i(h) = 1, x_i(h) \geq 0,\ h = \{1, \dots, m\} \right\}$$

is the standard m-dimensional simplex and $x_i(h)$ is the probability of player i choosing the pure strategy h.

2 *mixed strategy space*: it corresponds to the set of all mixed strategies of the players $x = \{x_1, \dots, x_n\}$

3 *utility function*: it represents the gain obtained by a player when it chooses a certain mixed strategy, in particular $u : \Delta^m \to \mathbb{R}_{\geq 0}$.

Here, it is assumed that the payoffs associated to each player are additively separable, thus the algorithm is a member of polymatrix games [62]. In GTG, the aforementioned definitions turns into the following:

Strategy space The strategy space x is the starting point of the game and contains all the mixed strategies. The space x can be initialized in different ways based on the fact that some prior knowledge exists or not. Here, we distinguish the initialization based on the type of object, *labeled* or *unlabeled*. For the labeled object, since their class is known, a one-hot vector is assigned:

$$x_i(h) = \begin{cases} 1, & \text{if } i \text{ has label } h \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

. For the unlabeled objects all the labels have the same probability of being associated to an object, thus:

$$x_i(h) = \frac{1}{m} \qquad h = \{1, \dots, m\} \tag{3.7}$$

Payoff function The utility function reflects the likelihood of choosing a particular label and considers the similarity between labeled and unlabeled players. Similar players influence each other more in picking one of the possible strategies (labels). Once the game reaches an equilibrium, every player play their best strategies which correspond to a consistent labeling [111] not only for the player itself but also for the others. Under equilibrium conditions the label of player i is given by the strategy played with the highest probability. Formally, given a player i and a strategy h:

$$u_i(h) = \sum_{j \in U} (A_{ij} x_j)_h + \sum_{k=1}^{m} \sum_{j \in L_k} A_{ij}(h,k) \tag{3.8}$$

$$u_i(x) = \sum_{j \in U} x_i^T A_{ij} x_j + \sum_{k=1}^{m} \sum_{j \in L_k} x_i^T (A_{ij})_k \tag{3.9}$$

where L_k is the set of labeled points with class k, $u_i(x)$ is the utility received by player i when it plays the mixed strategy x_i and $A_{ij} \in \mathbb{R}^{m \times m}$ is the *partial payoff matrix* between players i and j. As in [35], $A_{ij} = I_m \times \omega_{ij}$ where ω_{ij} is the similarity between player i and j and I_m is the identity matrix of size $m \times m$. The similarity function between players (objects) can be given or computed starting from the features. Given two objects i, j and their features f_i, f_j, their similarity is computed following the method proposed by [174]:

$$\omega(i,j) = exp\left\{-\frac{\|f_i - f_j\|_2}{\sigma_i\,\sigma_j}\right\} \tag{3.10}$$

where σ_i corresponds to the distance between i and its 7-nearest- neighbors. Similarity values are stored in matrix W.

Similarity sparisification In the cases that the data are particularly noisy, the above similarities can be sparsified using, for example, a k-NN graph. The payoffs become:

$$\mathcal{N}_i = k\text{-nearest neighbors of player } i \tag{3.11}$$

$$u_i(h) = \sum_{j \in \mathcal{U}_i} (A_{ij} x_j)_h + \sum_{k=1}^{m} \sum_{j \in \mathcal{L}_k^i} A_{ij}(h,k) \tag{3.12}$$

$$u_i(x) = \sum_{j \in \mathcal{U}_i} x_i^T A_{ij} x_j + \sum_{k=1}^{m} \sum_{j \in \mathcal{L}_k^i} x_i^T (A_{ij})_k \tag{3.13}$$

where $\mathcal{U}_i \subseteq \mathcal{N}_i$ and $\mathcal{L}_k^i \subseteq \mathcal{N}_i$ are the unlabeled and labeled (of class k) nearest neighbors of i, respectively.

Finding Nash Equilibria The last component of our method is an algorithm for finding equilibrium conditions in this game. Here the Replicator Dynamics are used (see Eq 3.5):

$$x_i(h)^{t+1} = x_i(h)^t \frac{u_i(h)^t}{u_i(x^t)} \tag{3.14}$$

where $x_i(h)^t$ is the probability of strategy h at time t for player i.
The RD are iterated until convergence, this means either the distance between two successive steps is zero (formally $||x^{t+1} - x^t||_2 \leq \varepsilon$) or a certain amount of iterations is reached (See [123] for a detailed analysis). In practical applications one could set the ε to a small number but typically 10-20 iterations are sufficient.

The GTG algorithm has commonalities with harmonic labeling of graphs [9] that should be further deepened. In fact, [9] considers the label of a node in a graph as the average labeling of its neighbors. Indeed, this is similar in the spirit of GTG since the final classification is the result of direct-neighbors labeling agreement. This relation deserves further deepening. Harmonic analysis has been extensively studied in the past also in other fields, for example in ranking methods [25].

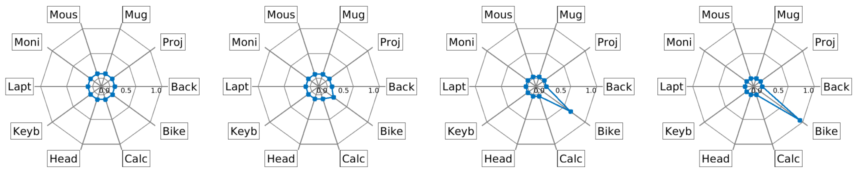


Figure 3.1: From left to right the starting point of the dynamical system and the point of convergence. Evolution of the mixed strategy associated to a player during the GTG process. As the dynamic is iterated, the entropy progressively decreases and the distribution peaks toward the correct class.

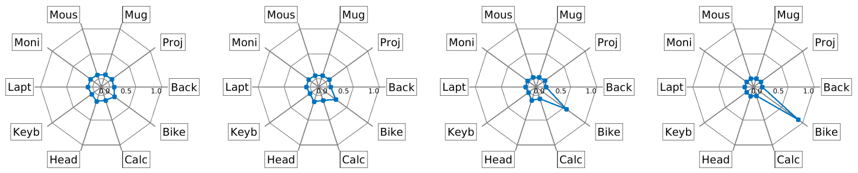


Figure 3.2: From left to right the starting point of the dynamical system and the point of convergence. In this example the dynamics start from three different classes, while in the end, thanks to the refinement of the neighboring mixed strategies the correct class is chosen.

Part I

Methodological Contributions

Chapter 4

Unsupervised Domain Adaptation

Reference: ***S. Vascon***, S. Aslan, A. Torcinovich, T. van Laarhoven, E. Marchiori, and M.Pelillo. Unsupervised Domain Adaptation using Graph Transduction Games. IEEE International Joint Conference on Neural Networks, IEEE, 2019. **Oral presentation**

The content of this chapter is taken from the above reference.

Unsupervised domain adaptation (UDA) amounts to assigning class labels to the unlabeled instances of a dataset from a target domain, using labeled instances of a dataset from a related source domain. In this chapter we propose to cast this problem in a game-theoretic setting as a non-cooperative game and introduce a fully automatized iterative algorithm for UDA based on graph transduction games (GTG). The main advantages of this approach are its principled foundation, guaranteed termination of the iterative algorithms to a Nash equilibrium (which corresponds to a consistent labeling condition) and soft labels quantifying uncertainty of the label assignment process. We also investigate the beneficial effect of using pseudo-labels from linear classifiers to initialize the iterative process. The performance of the resulting methods is assessed on publicly available object recognition benchmark datasets involving both shallow and deep features. Results of experiments demonstrate the suitability of the proposed game-theoretic approach for solving UDA tasks.

4.1 Introduction

The success of deep learning in computer vision classification tasks relies on the availability of a large amount of images annotated with their ground truths. However, manual label annotation is typically an expensive process and may contain wrong annotations. In order to overcome these limitations, Semi-Supervised Learning (SSL) approaches have been developed, usually involving the training of a classifier from a large dataset with plenty of unlabeled data and substantially less annotated data. In

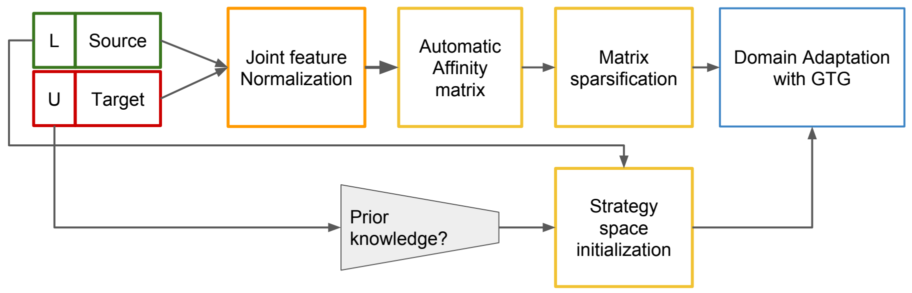

L
Source
U
Target
Joint feature Normalization
Automatic Affinity matrix
Matrix sparsification
Domain Adaptation with GTG
Prior knowledge?
Strategy space initialization

Figure 4.1: Pipeline of the proposed method.

some cases however, it is expensive to obtain unlabeled data too, resorting instead on data coming from a different source. This problem is best formulated in the Unsupervised Domain Adaptation setting, where unlabeled data comes from a different related distribution than that of the labeled data. Specifically, an annotated source dataset is exploited to infer the labels of an unlabeled target dataset from a different, related domain. Due to the tight relation between SSL and UDA problems, it is not uncommon to approach them with similar techniques (cf. [59] and [58] for example).

In this chapter we investigate the use of a game-theoretic graph-transductive approach, known as Graph Transduction Games, for domain adaptation (GTDA), which has been successfully applied in SSL tasks such as in [34], [157], [151] and [5], and we show that this approach, paired with a preprocessing step, provides overall improvements in three standard domain adaptation cases. Specifically we propose a fully automatized pipeline to perform UDA with GTG, comparing our results with those of recent methods, trying also to include prior information provided by a simple classifier, i.e. Logistic Regression. We perform also a comparison of GTG with other three standard graph-transductive algorithms. The picture that arises from the experimental results is promising and suggests considering graph transduction as a key-module when addressing UDA problems. The choice of using GTG as a transduction algorithm for DA has been motivated by its theoretical properties which guarantee: *i)* a consistent labeling of unknown samples at convergence, *ii)* the output of soft-labelings (probability distribution over the classes) for further refinements, *iii)* the possibility of injecting prior knowledge at the beginning of the transductive process.

The main contributions of this work are the following:

- We adopt the theory of label consistency of graph transduction games to propose a principled technique for UDA. This will offer a novel perspective on the UDA problem.
- We propose a parameter-free method for UDA based on game theory which bypasses intensive training phase.
- We reach state-of-the-art performance results on publicly available object recognition domain adaptation tasks.

The chapter is organized as follows. To make the chapter self-contained, in Sec. 8.3.2 we introduce the GTG algorithm. In Sec. 4.2 we describe in detail the proposed method. In Sec. 4.3 we discuss the experimental setting and the competing methods, while in Sec. 4.4 we report and analyze our experimental results. Finally, Sec.4.5 concludes the chapter.

4.1.1 Related work

Many methods for UDA have been introduced. Here we focus on the recent approaches used in our comparative assessment.

A number of DA models align the distributions of features from source and target domains by reducing their discrepancy. For instance, CORrelation ALignment (CORAL) [144] finds a linear transformation that minimizes the distance between the covariance of source and target. Subspace Alignment (SA) [38] computes a linear map that minimizes the Frobenius norm of the difference between the source and target domains, which are represented by subspaces described by eigenvectors. Feature Level Domain Adaptation (FLDA) [85] models the dependence between the two domains by means of a feature-level transfer model that is trained to describe the transfer from source to target domain. FLDA assigns a data-dependent weight to each feature representing how informative it is in the target domain. To to do it uses information on differences in feature presence between the source and the target domain.

Recently, end-to-end UDA methods based on deep neural networks have been shown to perform better than the aforementioned approaches. However they need large train data [139], use target labels to tune parameters [101] and are sensitive to (hyper-)parameters of the learning procedure [50]. Therefore, current state of the art based on this approach start from pre-trained network architectures. Various state-of-the-art methods considered in our comparative analysis use the ResNet pre-trained network, like Deep Domain Confusion (DDC) [152], Deep Adaptation Network (DAN) [99], Residual Transfer Networks (RTN) [101], Reverse Gradient (RevGrad) [49, 50], and Joint Adaptation Networks (JAN) [100].

4.2 Domain Adaptation with GTG

In this section we present our method, GTDA. We will explain how to cast the unsupervised domain adaptation problem in graph-transduction game setting. We consider given labeled (L) and unlabeled (U) datasets from the source and target domain, respectively. Then, labels from source data are propagated to the target instances by playing a non-cooperative multiplayer game in which the players are the objects (instances) and the labels the possible strategies.

The interaction between the players are represented in terms of a weighted undirected graph in which the edges are weighted by the similarity of player pairs, hence how much they will affect each others. In particular, the process is illustrated in Fig. 8.1 and explained in the following steps:

Joint feature standardization Given a dataset of features D and two domains $d_s, d_t \in D$ (source and target respectively), we normalize their features jointly as a pre-processing step. We perform two types of normalization on the union of the features: *std* features are scaled by their standard deviation or *z-score* subtract the mean and scaled by their standard deviation. Depending whether the sparsity of the features should be preserved or not, we pick the *std* or *z-score*, respectively.

Initialization of the mixed strategy profile The initial mixed strategy profile of the players, denoted as $x^{(0)}$ represents the starting point of the game. If prior knowledge is available, we can leverage it for its initialization. In our experimental settings, we explore two different initializations: *i)* in which no prior information is exploited (*no-prior* in the following) and *ii)* where an output from a logistic regression classifier is used (*+LR*). In the latter case, the logistic regression classifier has been trained for each pair of jointly normalized domains in a dataset. The training has been performed considering only the features belonging to the source, in a 2-fold cross validation setting, with an hyperparameter search for the C variable in the following log-scale range $C = \left[10^{-3}, 10^{4}\right]$. We end up with a LR model $M_{i,j}$ for each pair of domains d_i and d_j in a dataset. Given an unlabeled observation, the LR model outputs a probability distribution over the classes which is later used as prior knowledge in the strategy space. The choice of having as prior a probability distribution for each unlabeled object, instead of a one-hot vector, is mandatory since the one-hot vector cannot be updated by the GTG algorithm hence the performances would have been the same as the LR itself.

Algorithm 1 GTDA algorithm

Input: source feature matrix $F_S \in \mathbb{R}^{|S|\times d}$, target feature matrix $F_T \in \mathbb{R}^{|T|\times d}$, one-hot source labels $Y_S \in \Delta^{|S|\times m}$, minimum tolerance ε, maximum number of iterations K.
Output: target soft predictions $Y_T \in \Delta^{|T|\times m}$

```
1:  $N = |S| + |T|$
2:  $\hat{F}_S$ = normalize($F_S$)                                   ▷ Sec. III.a
3:  $\hat{F}_T$ = normalize($F_T$)                                   ▷ Sec. III.a
4:  $P_T$ = LR($\hat{F}_S, Y_S, \hat{F}_T$)                          ▷ Get log. reg. priors for $F_T$
5:  $x(0) = \begin{bmatrix} Y_S \\ \hat{P}_T \end{bmatrix}$          ▷ Init. Mixed Strategy Prof. Sec. III.b
6:  $W = [\omega(i,j)]_{ij}$                                         ▷ Eq. 7.6
7:  $\hat{W}$ = sparsify($W$)                                        ▷ Sec. III.d
8:  $tol = +\infty, t = 0$
9:  while $tol \geq \varepsilon$ and $t < K$ do
10:     for $i = 1, \ldots, N$ do
11:         $x_i(t+1) = \frac{x_i(t) \odot (\hat{W}x(t))_i}{x_i(t)(\hat{W}x(t))_i^T}$      ▷ Eq. 3.5
12:     $tol = ||x(t+1) - x(t)||_2$
13:     $t = t + 1$
14: $Y_T = x(t-1)_{|S|:N,1:m}$                                       ▷ Get the target predictions
```

Computation of the affinity matrix The core of the GTG is stored in the affinity matrix W (payoff of the players), so its computation requires particular care. In our experimental setting, we decided to rely on the following standard similarity kernel:

$$\omega(i,j) = \begin{cases} \exp\left\{-\frac{d(f_i,f_j)^2}{\sigma_i\sigma_j}\right\} & \text{if } i \neq j \\ 0 & \text{else} \end{cases} \tag{4.1}$$

where f_i, f_j are the features of observations i and j respectively, $d(f_i, f_j)$ is the cosine distance between features f_i and f_j. Here, motivated by [174] and [151], we set the scaling parameter σ_i automatically, considering the local statistics of the neighborhood of each point. Accordingly to [174], the value of σ_i is set to the distance of the 7-th nearest neighbour of observation i.

Affinity sparsification The sparsification of the graph plays an important role in the performances of the algorithm. Indeed, filtering out the small noisy similarities which may bias the utilities in Eq. 7.6, would prevent incorrect class labelings.

Here, we follow a statistical connectivity principle in random graph, which states that a graph is connected if each node has at least $k = \lfloor log_2(n) \rfloor + 1$ nearest neighbours [161]. The rationale of this choice is that the labels in GTG are propagated from the labeled elements to the unlabeled ones. If the graph is not connected the propagation might get stuck at a certain point. This sparsification ensures that the graph is connected, hence all the nodes are reached at the equilibrium condition of the dynamical system (cf. Eq3.5). The sparsification is performed for each node i independently considering the distance value of the k-NN as a threshold for the other nodes in the graph. In order to obtain a symmetric neighborhood, we include the node i in the neighbourhood of j (and viceversa) if one of the two is in the neighborhood of the other. Indeed, alternative sparsification methods can be considered [36, 47, 142, 143]. For example, in [36] the authors proposed a principle sparsification method based on the notion of *edge resistance*, a measure accounting for edges importance. The underlying idea is, given a weighted graph, obtain a graph with a reduced set of edges such that it maintains the same properties of the original one. This sparsification method can be effectively applied to our algorithm and its effect will be deepened in our future works.

Execution of GTG Once the affinity matrix is computed and the mixed strategy profile is initialized, GTG can be finally played up to convergence. The final probabilities, which determine then the labels for the unlabeled observations, correspond to the adaptation from sources to targets.

In algorithm 1 we present the pseudo-code of the entire method.

4.3 Experimental Setting

To validate our approach, we perform experiments on two publicly available popular datasets for object recognition domain adaptation: the Office-Caltech 10 [54] and the

Office 31 [133].

4.3.1 Datasets

In the following we present a short description of each datasets used for our experiments.

Office-31 Office-31 [133] is a dataset containing 31 classes divided in 3 domains: *Amazon* (A), *DSLR* (D) and *Webcam* (W). Office-31 has a total of 4110 images, with a maximum of 2478 images per domain. In this dataset we use deep features extracted from the ResNet-50 architecture [60] pretrained on ImageNet.

Office-Caltech Office-Caltech [54] consists in observations taken from the common classes of Office 31 and Caltech256 (10 in total) and are divided in 4 image domains, namely the ones of Office-Caltech and the additional *Caltech* (C). The features we consider are of two kinds: 800 SURF features [8], which we preprocess by z-score standardization, and deep features in the same fashion as the previous dataset.

4.3.2 Evaluation Criteria

We evaluate and report the accuracy on the target domain for each adaptation. Accuracy is computed as the fraction of the correctly labeled target instances. Furthermore, we report the average accuracy per methods and the top-3 performing by different coloring (**best performing**, second and third). Along the analysis of the results we highlight also the number of hit time that a method perform better than the competitors.

4.3.3 Comparing Methods

In order to assess our method in a broad context, the performances of GTDA has been compared with both standard domain adaptation methods, recent deep-learning based algorithm and baselines classifiers (SVM and LR). Furthermore, we assess the effectiveness of GTG, replacing it with other GT methods (Label Propagation, Label Spreading and Gaussian Fields and Harmonic Functions).

In our experiments, since we are dealing with more than two classes, we used one-vs-all linear SVM and multi-class logistic regression. More details on the methods we experimented for comparative analysis are given below.

Shallow Domain Adaptation Methods

The most prevalent domain adaptation methods accomplish domain adaptation task by reducing the discrepancy between source and target distributions via computing a feature transformation. We chose CORrelation ALignment (CORAL) [144] and Subspace Alignment (SA) [38] which are two popular methods following this approach. Reported performances for both methods are appealing whereas their application to high dimensional data might be problematic since they are not scalable to high number of features. Another approach for domain adaptation is modeling the dependence of source and target domain in feature level. We experimented by a recent work, namely Feature Level Domain Adaptation (FLDA) [85], that follows this approach. We use published source codes of the shallow DA methods for all datasets.

DNN-based Domain Adaptation Methods

Motivated by their reported stunning performances in recent years, we compared performance of GTDA with the performances of a number of Deep Neural Networks-based domain adaptation methods reported on the Office 31 dataset based on ResNet50 features. Specifically, we make comparison with Deep Domain Confusion (DDC) [152] where an objective function including an additional domain confusion term is proposed for learning domain-invariant representations for classification, Deep Adaptation Network (DAN) [99] where more transferable features are learned by adapting source and target distributions in multiple task-specific layers, Residual Transfer Networks (RTN) [101] that achieves feature adaptation and classifier adaptation simultaneously by deep residual learning [60], Reverse Gradient (RevGrad) [49, 50] that improves domain adaptation by employing adversarial training paradigm, and Joint Adaptation Networks (JAN) [100] that uses an adversarial learning strategy to maximize a joint maximum mean discrepancy such that distributions of source and target domains be more distinguishable. Despite of high performance accuracies, some disadvantages of DNN-based methods to be taken into account are that they require abundant training data for improvement in performance, they use target labels for parameter tuning [101] and their sensitivity to hyperparameters of the learning procedure is high [50]. We refer to the performance results reported in [100] for the aforementioned DNN-based techniques to make comparison with our technique, hence we will add results for the Office31 dataset only.

Graph Transduction Techniques for Domain Adaptation

We compare our game-theoretic graph transduction technique against three other transductive techniques, namely Label Propagation (LP) [180], Label Spreading (LS) [178] and Harmonic Function (HF) [183] for the domain adaptation problem. Similar to our method, these techniques exploit the so-called smoothness principle which states that closer instances tend to belong to the same class. LP [180] performs hard clamping of input labels which yields to avoiding change on the original label distribution at every iteration, while LS [178] adopts soft clamping where initial assignments are changed by a fraction α at each iteration. Moreover, employing regularization, the cost employed in LS differentiates from LP, which provides better robustness to noise. *Gaussian Fields and Harmonic Functions (HF)* [183] tries instead to compute a function f by minimizing a corresponding energy function $E(f)$. The solution is *harmonic* and this property can be exploited to propagate information according to the aforementioned smoothness principle.

To make a fair comparison with our approach, we provide to these algorithms the same affinity matrix as our, i.e. W, which is computed using the same scheme of normalization, σ selection and sparsification. For the LS technique, we experimented with a variety of values in the range of $(0, 1)$ for the parameter α. Since we got the best

Table 4.1: Comparative analysis on Office-31 dataset (ResNet-50 features)

	A→D	A→W	D→A	D→W	W→A	W→D	avg
Baselines							
Source SVM	76.9	73.8	60.3	97.5	59.4	**100.0**	78.0
Source LR	74.7	70.8	60.6	97.5	60.2	**100.0**	77.3
Shallow models							
SA	76.7	75.5	62.2	97.9	60.3	**100.0**	78.8
FLDA-Q	76.3	75.5	59.9	97.5	58.6	99.8	77.9
CORAL	78.9	76.9	59.7	98.2	59.9	**100.0**	78.9
Graph-transductive methods							
Lab Prop	2.4	3.6	3.3	3.6	3.3	99.8	19.3
Lab Spread	77.3	79.2	63.1	98.6	60.8	99.8	79.8
Harmonic Function	73.7	80.3	62.3	98.1	46.8	99.8	76.8
Proposed method, GTDA							
GTDA	80.5	78.0	66.2	**98.9**	62.9	99.8	81.1
GTDA + LR	**82.5**	**84.2**	**67.1**	97.9	**69.1**	99.8	**83.4**

Table 4.2: Comparative analysis on Office-31 dataset (ResNet-50 features)

	A→D	A→W	D→A	D→W	W→A	W→D	avg
Deep Neural Networks (results taken from [100])							
DDC	76.5	75.6	62.2	96.0	61.5	98.2	78.3
DAN	78.6	80.5	63.6	97.1	62.8	99.6	80.4
RTN	77.5	84.5	63.6	96.8	64.8	99.4	81.6
RevGrad	79.7	82.0	68.2	96.9	67.4	99.1	82.2
JAN-A	**85.1**	**86.0**	**69.2**	96.7	**70.7**	99.7	**84.6**
Proposed method, GTDA							
GTDA	80.5	78.0	66.2	**98.9**	62.9	**99.8**	81.1
GTDA + LR	82.5	84.2	67.1	97.9	69.1	**99.8**	83.4

results when $\alpha = 0.2$ which is also the suggested default value we present the results of LS with $\alpha = 0.2$.

4.4 Results

We use the notation of $A \rightarrow B$ to indicate the adaptation with A as source and B as target dataset. While we discuss the performances of the techniques, (i) we consider the averaged accuracy over all adaptation tasks and (ii) the number of adaptation tasks that a method outperforms.

4.4.1 Office 31 dataset

We present performances on Office 31 for the shallow and graph transductive methods in Table 4.1 while additional comparisons with deep-learning models is outlined in Table 4.2 reporting results from [100].

Non DL methods The results on non-DL methods are reported in Table 4.1. The GTDA outperforms CORAL, i.e. the best performed shallow DA method on this dataset, by around 2% and 4% in averaged accuracy with and without prior, respectively. When we compare GTDA with other GT methods, i.e. Label Prop., Label Spread and Harmonic Function, we see that GTDA without prior outperforms all of them with a performance of 81.1%, while Label Spread follows GTDA with the performance of 79.8 %. When the prior knowledge is used in GTDA the performances are far better, being the top performing ones. Another point to highlight from this experiment is that, in general, the transductive methods outperform the shallow models. Without considering the average results, the GTDA+LR outperforms 5 over 6 times the shallow models. When prior is not used the GTDA is the second best performing algorithm (still considering the GTDA+LR).

DL-based methods The results on DL-based methods are reported in Table 4.2. The GTDA outperforms DDC, DAN, RTN and RevGrad which are end-to-end learned system for DA. This is surprising, considering that GTDA does not require an extensive training phase and neither a parameter optimization like in DNN. JAN-A achieves the best averaged accuracy, i.e. 84.6%, on this dataset being for 4 times the best performing and just once as a second. Among the other DL models no one is able to clearly outperform JAN-A. Our GTDA without prior outperform JAN-A on 2 cases (D-W, W-D) while in the remaining five it becomes as third best only once. When GTDA benefits from prior (GTDA +LR) the performances approaches to JAN-A (83.4 % vs. 84.6 %) becoming the second best algorithm, even outperforming the other DL approaches.

Table 4.3: Comparative analysis on Office-Caltech dataset (SURF features)

	A→C	A→D	A→W	C→A	C→D	C→W	D→A	D→C	D→W	W→A	W→C	W→D	avg
Baselines													
Source SVM	41.0	40.1	42.0	52.7	45.9	**47.5**	33.0	32.1	75.9	38.4	34.6	75.2	46.5
Source LR	42.8	36.3	35.3	**54.1**	42.7	40.7	33.9	31.2	83.1	37.3	32.9	71.3	45.1
Shallow Models													
SA	37.4	36.3	39.0	44.9	39.5	41.0	32.9	34.3	65.1	34.4	31.0	62.4	41.5
FLDA-L	41.5	**45.9**	42.0	49.5	**48.4**	44.1	31.7	34.1	75.6	35.3	33.8	72.6	46.2
FLDA-Q	43.5	43.3	40.7	53.5	44.6	45.1	30.8	31.2	73.2	35.2	32.1	75.8	45.7
CORAL	**45.1**	39.5	**44.4**	52.1	45.9	46.4	37.7	33.8	84.7	35.9	33.7	86.6	48.8
Graph-transductive methods													
Lab Prop	13.4	7.6	9.8	9.6	45.9	9.8	9.6	13.4	9.8	9.6	13.4	**89.8**	20.2
Lab Spread	41.3	36.3	32.5	53.3	47.8	41.4	36.1	34.2	90.2	36.0	34.2	88.5	47.7
Harmonic Function	41.1	38.9	35.9	52.2	47.1	37.6	30.8	29.3	89.2	32.2	32.7	88.5	46.3
Proposed method, GTDA													
GTDA	40.2	37.6	32.9	53.8	46.5	35.9	**41.3**	**39.9**	**92.2**	34.6	**38.5**	89.2	48.5
GTDA + LR	40.2	37.6	38.3	52.6	45.9	45.1	39.2	35.4	**92.2**	**41.0**	37.1	89.2	**49.5**

4.4.2 Office-Caltech 10 dataset

We present performances on this dataset in Tables 4.3 and 4.4 when SURF and ResNet50 features are used, respectively. In Table 4.3 we see that CORAL outperforms other shallow DA methods, i.e. FLDA and Source SVM, by achieving the second best averaged accuracy over all the methods. While we achieve almost same performance as CORAL on average (48.8% for CORAL and 48.5% for GTDA) when we do not use

Table 4.4: Comparative analysis on Office-Caltech dataset (ResNet-50 features)

	A→C	A→D	A→W	C→A	C→D	C→W	D→A	D→C	D→W	W→A	W→C	W→D	avg
Baselines													
Source SVM	91.0	88.5	87.5	94.1	94.9	87.8	90.0	86.1	98.6	89.1	85.9	**100.0**	91.1
Source LR	89.9	91.7	88.5	94.5	93.6	85.1	90.1	85.8	98.0	89.7	85.5	**100.0**	91.0
Shallow models													
SA	89.7	93.0	90.8	94.6	91.1	**93.2**	89.8	84.1	99.0	88.9	84.3	**100.0**	91.5
FLDA-Q	91.1	93.6	92.2	94.5	94.3	89.5	90.3	86.3	97.6	90.3	83.7	**100.0**	91.9
CORAL	85.9	91.1	89.8	94.3	93.0	**93.2**	92.8	86.8	98.6	90.9	85.5	**100.0**	91.8
Graph-transductive methods													
Lab Prop	13.4	7.6	9.8	9.6	7.6	9.8	9.6	13.4	9.8	9.6	13.4	**100.0**	17.8
Lab Spread	87.1	88.5	95.9	93.4	91.1	84.1	88.0	88.6	**99.7**	90.1	85.6	**100.0**	91.0
Harmonic Function	88.6	80.9	85.4	93.5	94.9	89.2	88.1	83.2	**99.7**	57.4	69.2	**100.0**	85.8
Proposed method, GTDA													
GTDA	90.0	87.9	**98.0**	93.5	91.7	79.7	89.4	**89.4**	99.3	93.2	88.8	**100.0**	91.7
GTDA + LR	**91.5**	**98.7**	94.2	**95.4**	**98.7**	89.8	**95.2**	89.0	99.3	**95.2**	**90.4**	**100.0**	**94.8**

priors. When we get benefit of the priors (GTDA+LR) we outperform CORAL by 1% in average accuracy becoming the best one over all methods. We outperform other GT methods both in averaged accuracy and at majority of the adaptation tasks. In particular, our best competitor is CORAL reaches 5 top results among the shallow models while GTDA with prior outperform 6 time the shallow ones becoming the more stable algorithm in this setting.

We see in Table 4.4 that when ResNet50 features are used the performances of all methods are improved significantly over the ones obtained when SURF features were used, except Label Prop. All shallow DA methods and GTDA (when not using prior) achieve very similar performances. In particular, while FLDA-Q and CORAL outperforms GTDA in averaged accuracy when prior was not used by 0.2% and 0.1%, we see that GTDA reaches to 3 top results while FLDA-Q and CORAL stay at 1 and 2, respectively. Following them, baseline methods and Lab Spread achieves similar performances in averaged accuracy. When we get benefit from prior we outperform shallow DA methods at every adaptation task (except $C \rightarrow W$), i.e. we reach to the best result at 8 adaptation tasks, second best results at 3 adaptation tasks and third best result at 1 adaptation task, and we are the best among both shallow DA and other GT methods with 94.8% in averaged accuracy with GTDA +LR.

4.4.3 Overall analysis

The method yields competitive results in different adaptation setting. This is surprising since the method is quite simple and the choice made by the dynamical system are greedy. The other thing that is worth noting is the use of the prior, which significantly improves the performance. Prior knowledge can be easily injected in the model through the strategy space and it is of legitimate use even under unsupervised DA providing that the training for whichever model is performed only on the source data. The proposed GTDA is the most stable in terms of number of best DA, thereby making it an attractive alternative.

4.5 Conclusions

In this work we have proposed a new algorithm to tackle unsupervised domain adaptation tasks. The methodology is based on graph-transduction and game-theory, offering a principled perspective to the problem. The GTDA proposed here has two main advantages: *i)* it is completely parameter-free and *ii)* it allows the direct embedding of prior knowledge on the target labels to be predicted. The results achieved on publicly available benchmark datasets demonstrate the validity of the proposed approach, whose performance is competitive with respect to state-of-the-art DA techniques with shallow and deep features as well as to other standard graph-based transductive methods. Furthermore, GTDA reaches comparable performances as that of known deep-learning UDA methods. As a future work we plan to extend the comparison with other recent DA techniques using more real-life datasets. As for the methodology we are interested in investigating other aspects, like semi-supervised domain adaptation.

Acknowledgment

This work has been partially funded by the Netherlands Organization for Scientific Research (NWO) within the EW TOP Compartment 1 project 612.001.352.

Chapter 5

Context-Aware Non-Negative Matrix Factorization

Reference: R. Tripodi*, ***S. Vascon****, and M. Pelillo. Context aware non-negative matrix factorization clustering. *In Proc. of the 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1719–1724. IEEE, 2016. *= equal contribution.

The content of this chapter is taken from the above reference.

In this chapter we propose a method to refine the clustering results obtained with the nonnegative matrix factorization (NMF) technique, imposing consistency constraints on the final labeling of the data. The research community focused its effort on the initialization and on the optimization part of this method, without paying attention to the final cluster assignments. We propose a game theoretic framework in which each object to be clustered is represented as a player, which has to choose its cluster membership. The information obtained with NMF is used to initialize the strategy space of the players and a weighted graph is used to model the interactions among the players. These interactions allow the players to choose a cluster which is coherent with the clusters chosen by similar players, a property which is not guaranteed by NMF, since it produces a soft clustering of the data. The results on common benchmarks show that our model is able to improve the performances of many NMF formulations.

5.1 Introduction

Nonnegative matrix factorization (NMF) is a particular kind of matrix decomposition in which an input matrix X is factorized into two non-negative matrices W and H of rank k, such that WH^T approximates X. The significance of this technique is to find those vectors that are linearly independent in a determined vector space. In this way, they can be considered as the essential representation of the problem described by the vector space and can be considered as the latent structure of the data in a reduced space. The advantage of this technique, compared to other dimension reduction techniques such as Single Value Decomposition (SVD), is that the values taken by each vector are positive. In fact, this representation gives an immediate and intuitive glance of the importance of the dimensions of each vector, a characteristic that makes NMF particularly suitable for soft and hard clustering [168].

The dimensions of W and H^T are $n \times k$ and $k \times m$, respectively, where n is the number of objects, m is the number of features, and k is the number of dimensions of the new vector space. NMF uses different methods to initialize these matrices [15, 92, 166] and then optimization techniques are employed to minimize the differences between X and WH^T [97].

The initialization of the matrices W and H [15], is crucial and can lead to different matrix decompositions, since it is performed randomly in many algorithms [176]. To the contrary, the step involving the final clustering assignment received less attention by the research community. In fact, once W and H are computed, soft clustering approaches interpret each value in W as the strength of association among objects and clusters and hard clustering approaches assign each object j to the cluster C_k, where:

$$k = arg\ max(W_{j1}, W_{j2}, ..., W_{jk}). \tag{5.1}$$

This step is also crucial since in hard clustering it could be the case that the assignments have to be made choosing among very similar (possibly equal) values and Equation 5.1 in this case can results inaccurate or even arbitrary. Furthermore this approach does not guarantee that the final clustering is consistent, with the drawback that very similar objects can result in different clusters. In fact, the clusters are assigned independently with this approach and two different runs of the algorithm can result in different partitioning of the data, due to the random initializations [176].

These limitations can be overcome exploiting the relational information of the data and performing a consistent labeling. For this reason in this chapter we use a powerful tool derived from evolutionary game theory, which allows to re-organize the clustering obtained with NMF, making it consistent with the structure of the data. With our approach we impose that the cluster membership has to be re-negotiated for all the objects. To this end, we employ a dynamical system perspective, in which it is imposed that similar objects have to belong to similar clusters, so that the final clustering will be consistent with the structure of the data. This perspective has demonstrated its efficacy in different semantic categorization scenarios [147, 148], which involve a high number of interrelated categories and require the use of contextual and similarity information.

5.2 NMF Clustering

NMF is employed as clustering algorithm in different applications. It has been successfully applied in “parts-of-whole” decomposition [92], object clustering [30], face recognition [164], multimedia analysis [16], and DNA gene expression grouping [175]. It is an appealing method because it can be used to perform together objects and feature clustering. The generation of the factorized matrices starts from the assumption that the objects of a given dataset belong to k clusters and that these clusters can be represented by the features of the matrix W, which denotes the relevance that each cluster has for each object. This description is very useful in soft clustering applications because an object can contain information about different clusters in different measure. For example a text about a the launch of a new car model into the marked can contain information about economy, automotive or life-style, in different proportions. Hard clustering applications require to choose just one of these topics to partition the data and this can be done considering not only the information about the single text, but also the information of the other texts in the texts collection, in order to divide the data in coherent groups.

In many algorithms the initialization of the matrices W and H is done randomly [92] and have the drawback to always lead to different clustering results. In fact, NMF converges to local minima and for this reason has to be run several times in order to select the solution that approximates better the initial matrix. To overcome this limitation there were proposed different approaches to find the best initializations based on feature clustering [166] and SVD techniques [15]. These initializations allow NMF to converge always to the same solution. [166] uses spherical k-means to partition the columns of X into k clusters and selects the centroid of each cluster to initialize the corresponding column of W. Nonnegative Double Singular Value Decomposition (NNDSVD) [15] computes the k singular triplets of X, forms the unit rank matrices using the singular vector pairs, extracts from them their positive section and singular triplets and with this information initializes W and H. This approach has been shown to be almost as good as that obtained with random initialization [15].

A different formulation of NMF as clustering algorithm was proposed by [87] (SymNMF). The main difference with classical NMF approaches is that SymNMF takes a square nonnegative similarity matrix as input instead of a $n \times m$ data matrix. It starts from the assumption that NMF was conceived as a dimension reduction technique and that this task is different from clustering. In fact, dimension reduction aims at finding a few basis vectors that approximate the data matrix and clustering aims at partitioning the data points where similarity is high among the elements of a cluster and low among the elements of different clusters. In this formulation a basis vector strictly represents a cluster.

Common approaches obtain an approximation of X minimizing the Frobenius norm of the difference $||X - WH^T||$ or the generalized Kullback-Leibler divergence $D_{KL}(X||WH^T)$ [10] , using multiplicative update rules [93] or gradient methods [97].

5.3 Our Approach

In this section we present the Game Theoretic Nonnegative Matrix Factorization (GTNMF), our approach to NMF clustering refinement. The pipeline of this method is depicted in Fig. 8.1. We extract the feature vectors of each object in a dataset then, depending on the NMF algorithm used, we give as input to NMF the feature vectors or a similarity matrix. GTNMF takes as input the matrix W obtained with NMF and the similarity graph A (see Section 5.4.3) of the dataset to produce a consistent clustering of the data.

Each data point, in our formulation, is represented as a player that has to choose its cluster membership. The weighted graph A measures the influence that each player

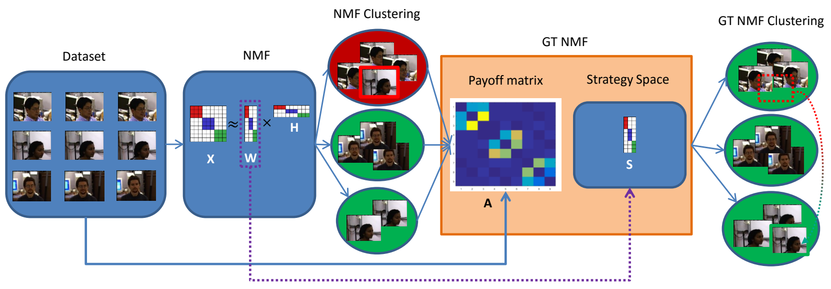


Figure 5.1: The pipeline of the proposed game-theoretic refiner method: a dataset is clustered using NMF obtaining a partition of the original data into k clusters. A pairwise similarity matrix A is constructed on the original set of data and the clustering assignments obtained with NMF. The output of NMF (W) and the matrix A are used to refine the assignments. The matrix W is also used to initialize the strategy space of the games. In red the wrong assignment that is corrected after the refinement. Best viewed in color.

has on the others. The matrix W is used to initialize the strategy space S of the players. We use the following equation $s_{ij} = \frac{w_{ij}}{\sum_{j=1}^{K} w_{ij}}$ to constrain the strategy space of each player to lie on the standard simplex, as required in a game theoretic framework (see Section 3.2). The dynamics are not started on the center of the K-dimensional simplex, as it is commonly done in unsupervised learning tasks, but on a different interior point, which corresponds to the solution point of NMF and do not compromise the dynamics to converge to Nash equilibria [165].

Now that we have the topology of the data A and the strategy space of the game S we can compute the Nash equilibria of the games according to equation (3.5). In each iteration of the system each player plays a game with its neighbors N_i according to the similarity graph A and the payoffs are calculated as follows:

$$u_i(e^h, s) = \sum_{j \in N_i} (a_{ij} s_j)_h \tag{5.2}$$

and

$$u_i(s) = \sum_{j \in N_i} x_i^T (a_{ij} s_j) \tag{5.3}$$

We assume that the payoff of player i depends on the similarity that it has with player j, a_{ij}, and its preferences, (s_j). During each phase of the dynamics a process of selection allows strategies with higher payoff to emerge and at the end of the process each player chooses its cluster according to these constraints. Since Equation 3.5 models a dynamical system it requires some criteria to stop. In the experimental part of this work we used as stopping criteria the maximum number of iterations = 100 and $\delta < 10^4$, where δ is the Euclidean norm between the strategy space at time t and at time $t + 1$.

Table 5.1: Datasets description. ★ = the dataset has been pruned as in [87].

Dataset	Type	Points	Features	Clusters	Balance	Mean Clust	Min Cl Size	Max Cl Size
NIPS★	text	424	17522	9	0.105	47.1	15	143
NIPS	text	451	17522	13	0.028	34.7	4	143
Reuters★	text	8095	14143	20	0.011	404.8	42	3735
Reuters	text	8654	14333	65	0	133.1	1	3735
RCV1★	text	13254	20478	40	0.028	331.4	45	1587
RCV1	text	13732	20478	75	0.001	183.1	1	1587
PIE-Expr	image	232	4096	68	0.75	3.4	3	4
ORL	image	400	5796	40	1	10	10	10
COIL-20	image	1440	4096	20	1	72	72	72
ExtYaleB	image	2447	3584	38	0.678	64.4	59	87

5.4 Experimental Setup and Results

In this section, we show the performances of GTNMF on different text and image datasets, and compare it with standard NMF[1] [79], NMF-S [79] (same as NMF but with the similarity matrix as input instead of the features), SymNMF [87][2] and NNDSVD[3] [15], which use the standard maximization technique to obtain an hard clustering of the data. In Table 5.2 we refer to our approach as *NMF-algorithm+GT* which means that the GTNMF has been initializied with the particular NMF-algorithm.

5.4.1 Datasets description

The evaluation of GTNMF has been conducted on datasets with different characteristics (see Table 5.1). We used textual (Reuters, RCV1, NIPS) and image (COIL-20, ORL, Extended YaleB and PIE-Expr) datasets. Authors in [87] discarded the objects belonging to small clusters in order to make the dataset more balanced, simplifying the task. We tested our method using this approach and also keeping the datasets as they are (without reduction), which lead to situations in which it is possible to have in the same dataset clusters with thousands of objects and clusters with just one object (e.g. RCV1).

5.4.2 Data preparation

The datasets have been processed as suggested in [87]. Given an $n \times m$ data matrix X, the similarity matrix A is constructed according to the type of dataset (textual and image). With textual dataset each feature vector is normalized to have unit 2-norm and the cosine distance is computed, $A = {x_i}^T x_j$. For image datasets each feature (column) is first normalized to lie in the range $[0, 1]$ and then it is applied the following kernel: $A_{i,j} = \exp\{-\frac{\|x_i - x_j\|}{\sigma_i \sigma_j}\}$, where σ_i is the Euclidean distance of the 7-th nearest neighbor [173]. In all cases $A_{ii} = 0$.

[1]Code: `https://github.com/kimjingu/nonnegfac-matlab`

[2]Code: `https://github.com/andybaoxv/symnmf`

[3]Code: `http://www.boutsidis.org/NNDSVD_matlab_implementation.rar`

The matrix is thus sparsified keeping only the q nearest neighbors for each point. The parameter q is set accordingly to [161] and represents a theoretical bound that guarantees the connectedness of a graph:

$$q = \lfloor log_2(n) \rfloor + 1 \tag{5.4}$$

Let $N(i) = \{j \text{ s.t. } x_j \in q\text{-NN of } i\}$ then $A_{ij} = A_{ij}$ if $i \in N(j)$ *or* $j \in N(i)$ and 0 otherwise.

The matrix A is thus normalized in a normalized-cut fashion obtaining the final matrix $A_{ij} = A_{ij} \sqrt{d_i} \sqrt{d_j}$ where $d_i = \sum_{s=1}^{n} A_{is}$. The matrix A is given as input to all the compared methods, expect from NMF to which the data matrix X is given. See [87] for further details on this phase.

5.4.3 Games graph

In Sec.5.4.2 has been explained how to create the similarity matrix for NMF, the same methodology has been used to create the payoff matrix A for the GTNMF, with the only difference that, in this case, we exploit the partitioning obtained with NMF in order to identify what could be the expected size of the clusters. The assumption here is that the clustering obtained via NMF provides a good insight on the size of the final clusters and accordingly with this information a proper number q (see Equation 5.4) can be selected. A cluster C can be considered as a fully connected subgraph and thus the number of neighbors of each element in the cluster C should be at least $q_C = \lfloor log_2(|C|) \rfloor + 1$ to guarantee the connectedness of the cluster itself. The variable q is thus chosen based on the same principle of [161] but instead of taking into account the entire set of points (as in Sec.5.4.2) we focused only on the subsets induced by the NMF clustering. This results in having a different q for each point in the dataset based on the following rule:

$$q_i = \lfloor log_2(|C|) \rfloor + 1 \tag{5.5}$$

where $|C|$ is the cardinality of cluster C to which the i-th element belongs to. For obvious reason $q_i \leq q$, $\forall i = 1, \ldots, n$ and thus concentrating only on the potential number of neighbors that belong to the cluster and not in the entire graph because we are doing a refinement. From a game-theoretic perspective this means to focus the games only among a set of similar players which are likely to belong to the same cluster.

5.4.4 Evaluation measures

The evaluation of oOur approach has been validated using two different measures, accuracy (AC) and normalized mutual information (NMI). AC is calculated as $\frac{\sum_{i=1}^{n} \delta(\alpha_i, map(l_i))}{n}$, where n denotes the total number of documents in the dataset, $\delta(x, y)$ equals to 1 if x and y are clustered in the same class; $map(L_i)$ maps each cluster label l_i to the equivalent label in the benchmark. The best mapping is computed using the Kuhn-Munkres algorithm [103]. The AC counts the number of correct clusters assignments. NMI indicates the level of agreement between the clustering C provided by the ground truth and the clustering C' produced by a clustering algorithm. The mutual information (MI)

Table 5.2: Performance of GTNMF compared to several NMF approaches. The mean and std deviation of 20 runs are reported.

Normalized Mutual Information

Dataset	SymNMF	SymNMF+GT	NMF	NMF+GT	NMFS	NMFS+GT	NNDSVD	NNDSVD+GT
NIPS ★	0.385 (±0.011)	**0.405** (±0.016)	0.375 (±0.022)	**0.386** (±0.011)	0.388 (±0.006)	**0.403** (±0.007)	0.388	**0.399**
NIPS	0.387 (±0.007)	**0.418** (±0.017)	0.401 (±0.016)	**0.406** (±0.016)	0.393 (±0.008)	**0.412** (±0.018)	0.388	**0.421**
Reuters ★	0.502 (±0.014)	**0.51** (±0.016)	0.451 (±0.026)	**0.49** (±0.02)	0.505 (±0.014)	**0.511** (±0.015)	**0.427**	0.425
Reuters	0.517 (±0.007)	**0.525** (±0.006)	0.442 (±0.006)	**0.497** (±0.003)	0.518 (±0.006)	**0.527** (±0.005)	0.488	**0.493**
RCV1 ★	0.406 (±0.007)	**0.422** (±0.007)	0.51 (±0.007)	**0.516** (±0.005)	0.404 (±0.009)	**0.42** (±0.01)	**0.403**	0.402
RCV1	0.411 (±0.006)	**0.422** (±0.006)	0.462 (±0.009)	**0.483** (±0.007)	0.413 (±0.006)	**0.424** (±0.006)	0.398	**0.407**
PIE-Expr	0.95 (±0.004)	**0.968** (±0.004)	0.939 (±0.008)	**0.959** (±0.006)	0.89 (±0.005)	**0.931** (±0.006)	0.86	**0.889**
ORL	0.888 (±0.006)	**0.921** (±0.006)	0.691 (±0.015)	**0.844** (±0.014)	0.889 (±0.006)	**0.918** (±0.004)	0.808	**0.892**
COIL-20	0.871 (±0.009)	**0.875** (±0.012)	0.619 (±0.017)	**0.669** (±0.016)	0.877 (±0.013)	**0.883** (±0.013)	0.824	**0.836**
ExtYaleB	0.308 (±0.005)	**0.313** (±0.005)	**0.356** (±0.006)	0.355(±0.007)	0.309 (±0.007)	**0.314** (±0.005)	0.288	**0.315**

Accuracy

Dataset	SymNMF	SymNMF+GT	NMF	NMF+GT	NMFS	NMFS+GT	NNDSVD	NNDSVD+GT
NIPS ★	0.462 (±0.013)	**0.483** (±0.016)	**0.426** (±0.02)	0.425(±0.014)	0.465 (±0.011)	**0.485** (±0.017)	0.474	**0.509**
NIPS	0.379 (±0.01)	**0.415** (±0.037)	**0.396** (±0.022)	0.39(±0.018)	0.384 (±0.014)	**0.407** (±0.031)	0.466	**0.503**
Reuters ★	0.517 (±0.044)	**0.528** (±0.043)	0.322 (±0.024)	**0.401** (±0.026)	0.516 (±0.037)	**0.525** (±0.037)	0.403	**0.427**
Reuters	0.324 (±0.029)	**0.363** (±0.032)	0.222 (±0.011)	**0.282** (±0.02)	0.339 (±0.023)	**0.378** (±0.024)	0.277	**0.339**
RCV1 ★	**0.292** (±0.015)	0.289(±0.014)	0.383 (±0.009)	**0.387** (±0.01)	**0.298** (±0.017)	0.297(±0.017)	**0.285**	0.276
RCV1	0.243 (±0.008)	**0.247** (±0.008)	0.279 (±0.01)	**0.295** (±0.011)	0.242 (±0.01)	**0.245** (±0.011)	0.239	**0.24**
PIE-Expr	0.81 (±0.021)	**0.85** (±0.019)	0.783 (±0.023)	**0.809** (±0.024)	0.617 (±0.019)	**0.7** (±0.02)	**0.536**	0.513
ORL	0.776 (±0.017)	**0.811** (±0.018)	0.465 (±0.019)	**0.608** (±0.026)	0.77 (±0.013)	**0.804** (±0.015)	0.653	**0.71**
COIL-20	0.727 (±0.036)	**0.729** (±0.037)	0.478 (±0.023)	**0.507** (±0.025)	0.739 (±0.046)	**0.741** (±0.046)	**0.674**	0.672
ExtYaleB	**0.235** (±0.008)	0.228(±0.007)	0.194 (±0.007)	**0.197** (±0.009)	**0.237** (±0.012)	0.23(±0.01)	0.229	**0.242**

between the two clusterings is computed as,

$$\sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)} \tag{5.6}$$

where $p(c_i)$ and $p(c'_i)$ are the probabilities that a document of the corpus belongs to cluster c_i and c'_i, respectively, and $p(c_i, c'_i)$ is the probability that the selected document belongs to c_i as well as c'_i at the same time. The MI information is then normalized with the following equation,

$$NMI(C, C') = \frac{MI(C, C')}{max(H(C), H(C'))} \tag{5.7}$$

where $H(C)$ and $H(C')$ are the entropies of C and C', respectively.

5.4.5 Evaluation

The results of our evaluation are shown in Table 5.2, where we reported the mean and standard deviation of 20 independent runs. For NNDSVD the experiments are run only one time, since it converges always to the same solution. The performances of GTNMF in most of the cases are higher those of the different NMF algorithms. In particular, we can notice that despite the different settings (textual/image datasets) our algorithm is able improve the NMI performance in 33/36 cases with a maximum gain of $\simeq 15.3\%$ (which is quite impressive) and a maximum loss of 0.2%. constant gain in the NMI means, in practice, that the algorithm is able to partition better the dataset, making the final clustering closer to the ground truth. In terms of AC, on 27/36 cases the method improve on the compared methods, with a maximum gain of 14.3% and maximum loss of 2.3%. It worth noting that the negative results are very low and in most of the cases

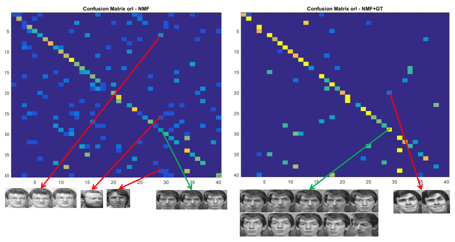
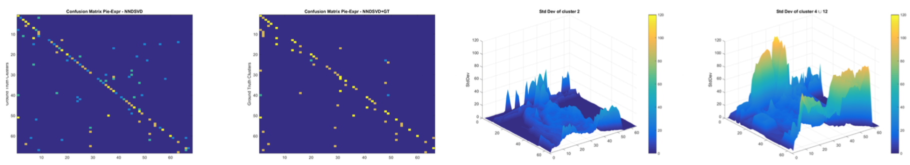


Figure 5.2: On the left side a confusion matrix produced by NMF on the ORL dataset and on the right side the ones produced by our method.

the corresponding number of incorrect reallocations is low, in fact, −0.6% in the NIPS dataset means 2.7 elements or −0.2% in COIL-20 corresponds to 2.8 elements.

The mean gain for NMI and AC are 2.68% and 2.30%, respectively, while the mean loss are 0.16% and 0.01%. In some cases we can see that we obtain a loss in NMI and a gain in AC, for example on ExtYaleB with NMF. In this case the similarity matrix given as input to GTNMF tends to concentrate more objects in the same cluster, because the dataset is not balanced and it could be the case that, in these situations, a big cluster tends to attract many objects, increasing the probability of good reallocations, which results in an increase in AC and in a potentially wrong partitioning of the data. To the contrary in some experiments we have a loss in AC and a gain in NMI. For example on PIE-Expr we noticed that we are able to put together many objects that the other approaches tend to keep separated, but in this particular case GTNMF collected in the same cluster all the objects belonging to four similar clusters and for this reason there was a loss in accuracy (see Fig. 5.3).

We can see that the results of our method on well balanced datasets (ORL, COIL-20) are almost always good. Also on very unbalanced datasets, such as Reuters and Reuters★ we have always good performances, whatever is the method used. These datasets depict better real life situations and the improvements over them are due to the fact that in these cases it is necessary to exploit the geometry of the data in order to obtain a good partitioning.



(a) (b) (c) (d)

Figure 5.3: On a and b the confusion matrices produced by NMF and GTNMF on Pie-Expr. On c the std dev of the objects merged together by GTNMF and on d the std dev of two random clusters combined together.

A positive and a negative case study are shown in Fig. 5.2 and 5.3, respectively. In Fig. 5.2 the confusion matrix obtained with GTNMF is less sparse and more concentrated on the main diagonal. Given the same cluster Id, the NMF method agglomerates different clusters (red arrows) while, after the refinement, the number of elements corresponding to the correct cluster are moved. In Fig. b the algorithm tends to agglomerates the elements on a single cluster (second column of the matrix). This can be explained on how the similarity matrix is composed and on the nature of the data: in Fig. c the std dev of the images in the agglomerated cluster is reported, as one can notice the std is very low meaning that all the faces in that cluster are very similar to each other. To give a counterexample we report on Fig. d the std dev of two random cluster joined together, is straightforward to notice that the std dev is higher than in the previous example meaning that the elements within those two clusters are highly dissimilar in nature and thus easily separable.

5.5 Conclusion

In this work we presented GTNMF, a game theoretic model to improve the clustering results obtained with NMF going beyond the classical technique used to make the final clustering assignments. The W matrix obtained with NMF can have an high entropy which make the choice of a cluster very difficult in many cases. With our approach we try to reduce the uncertainty in the matrix W using evolutionary dynamics and taking into account contextual information to perform a consistent labeling of the data. In fact, with our method similar objects are assigned to similar clusters, taking into account the initial solution obtained with NMF.

We conducted an extensive analysis of the performances of our method and compared it with different NMF formulations and on datasets with different features and of different kind. The results of the evaluation demonstrated that our approach is almost always able to improve the results of NMF and that when it have negative results those results are practically non significant. The algorithm is quite general thanks to the adaptive auto-tuning of the payoff matrix and can deal with balanced and completely unbalanced datasets.

As future work we are planning to use different initialization of the strategy space, to use new similarity functions to construct the games graph, to apply this method to different problems and to different clustering algorithms.

Acknowledgment

This work was supported by Samsung Global Research Outreach Program.

Chapter 6

Transductive Label Augmentation for Improved Deep Network Learning

Reference: I. Elezi*, A. Torcinovich*, ***S. Vascon****, and M. Pelillo. Transductive label augmentation for improved deep network learning. *In Proc. of the 24rd International Conference on Pattern Recognition (ICPR)*, pp. 1432-1437, IEEE, 2018.*= equal contribution.

The content of this chapter is taken from the above reference.

A major impediment to the application of deep learning to real-world problems is the scarcity of labeled data. Small training sets are in fact of no use to deep networks as, due to the large number of trainable parameters, they will very likely be subject to overfitting phenomena. On the other hand, the increment of the training set size through further manual or semi-automatic labellings can be costly, if not possible at times. Thus, the standard techniques to address this issue are transfer learning and data augmentation, which consists of applying some sort of "transformation" to existing labeled instances to let the training set grow in size. Although this approach works well in applications such as image classification, where it is relatively simple to design suitable transformation operators, it is not obvious how to apply it in more structured scenarios. Motivated by the observation that in virtually all application domains it is easy to obtain unlabeled data, in this chapter we take a different perspective and propose a label augmentation *approach. We start from a small, curated labeled dataset and let the labels propagate through a larger set of unlabeled data using graph transduction techniques. This allows us to naturally use (second-order) similarity information which resides in the data, a source of information which is typically neglected by standard augmentation techniques. In particular, we show that by using known game theoretic transductive processes we can create larger and accurate enough labeled datasets which use results in better trained neural networks. Preliminary experiments*

are reported which demonstrate a consistent improvement over standard image classification datasets.

6.1 Introduction

Deep neural networks (DNNs) have met with success multiple tasks, and testified a constantly increasing popularity, being able to deal with the vast heterogeneity of data and to provide state-of-the-art results across many fields and domains [90, 137]. Convolutional Neural Networks (CNNs) [46, 91] are one of the protagonists of this success. Starting from AlexNet [86], until the most recent convolutional-based architectures [60, 63, 145] CNNs have proved to be especially useful in the field of computer vision, improving the classification accuracy in many datasets [1, 27].

However, a common caveat of large CNNs is that they require a lot of training data in order to work well. In the presence of classification tasks on small datasets, typically those networks are *pre-trained* in a very large dataset like ImageNet [27], and then *finetuned* on the dataset the problem is set on. The idea is that the pre-trained network has stored a decent amount of information regarding features which are common to the majority of images, and in many cases this knowledge can be transferred to different datasets or to solve different problems (image segmentation, localization, detection, etc.). This technique is referred as *transfer learning* [169] and has been an important ingredient in the success and popularization of CNNs. Another important technique – very often paired with the previous one – is *data augmentation*, through which small transformations are directly applied on the images. A nice characteristic of data augmentation is its agnosticism toward algorithms and datasets. [21] used this technique to achieve state-of-the-art results in MNIST dataset [89], while [86] used the method almost without any changes to improve the accuracy of their CNN in the ImageNet dataset [27]. Since then, data augmentation has been used in virtually every implementation of CNNs in the field of computer vision.

Despite the practicality of the above-mentioned techniques, when the number of images per class is extremely small, the performances of CNNs rapidly degrade and leave much to be desired. The high availability of unlabeled data only solves half of the problem, since the manual labeling process is usually costly, tedious and prone to human error. Under these assumptions, we propose a new method to perform an automatic labeling, called *transductive label augmentation*. Starting from a very small labeled dataset, we set an automatic label propagation procedure, that relies on graph transduction techniques, to label a large unlabeled set of data. This method takes advantage of second-order similarity information among the data objects, a source of information which is not directly exploited by traditional techniques. To assess our statements, we perform a series of experiments with different CNN architectures and datasets, comparing the results with a first-order "label propagator".

In summary, our contributions in this article are as follows: a) by using graph transductive approaches, we propose and develop the aforementioned label augmentation method and use it to improve the accuracy of state-of-the-art CNNs in datasets where the number of labels is limited; b) by gradually increasing the number of labeled objects, we give detailed results in three standard computer vision datasets and compare

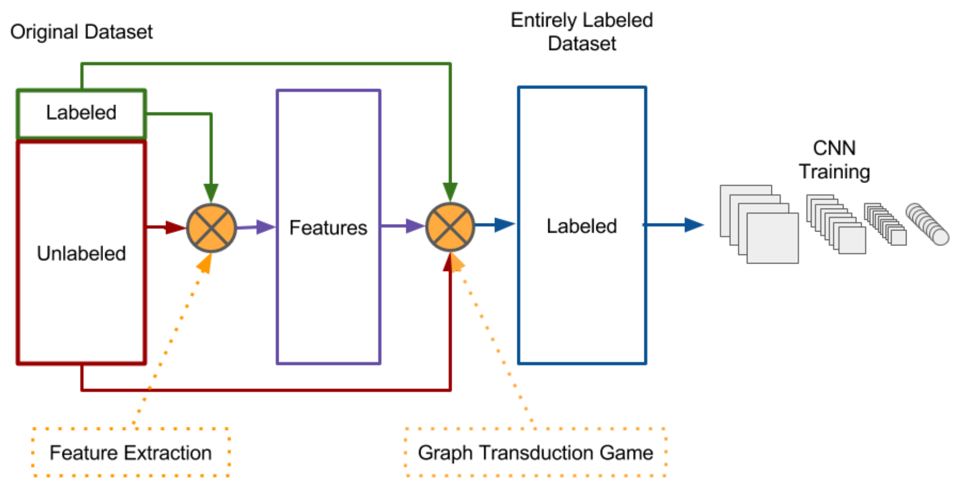


Figure 6.1: The pipeline of our method. The dataset consists of labeled and unlabeled images. First, we extract features from the images, and then we feed the features (and the labels of the labeled images) to graph transduction games. For the unlabeled images, we use a uniform probability distribution as 'soft-labeling'. The final result is that the unlabeled points get labeled, thus the entire dataset can be used to train a convolutional neural network.

the results with the results of CNNs; c) we replace our transductive algorithm with linear support vector machines (SVM) [23] to perform label augmentation and compare the results; d) we give directions for future work and how the method can be used on other domains.

6.1.1 Related Work

Semi-supervised label propagation has a long history of usage in the field of machine learning [156]. Starting from an initial large dataset, with a small portion of labeled observations the traditional way of using semi-supervised learning is to train a classifier only in the labeled part, and then use the classifier to predict labels for the unlabeled part. The labels predicted in this way are called *pseudo-labels*. The classifier is then trained in the entire dataset, considering the pseudo-labels as if they were real labels.

Different methods with the same intent have been previously proposed. In deep learning in particular, there have been devised algorithms to use data with a small number of labeled observations. [68] trained the network jointly in both the labeled and unlabeled points. The final loss function is a weighted loss of both labeled and unlabeled points, where in the case of the unlabeled points, the pseudo-label is determined by the highest score proposed by the model. [59] optimized a CNN on such a way as to produce embeddings that have high similarities for the observations that belong to the same class. [81] used a totally different approach, developing a generative model that allows for effective generalization from small labeled datasets to large unlabeled ones.

In all the mentioned methods, the way how the unlabeled data has been used can be considered as an intrinsic property of their engineered neural networks. Our choice of CNNs as the algorithm used for the experiments was motivated because CNNs are state-of-the-art models in computer vision, but the approach is more general than that. The method presented in this article does not even require a neural network and in principle, non-feature based observations (i.e graphs) can be considered, as long as a similarity measure can be derived for them. At the same time, the method shows good results in relatively complex image datasets, improving over the results of state-of-the-art CNNs.

6.2 Label Generation

The previously explained framework can be applied to a dataset with many unlabeled objects to perform an automatic labeling and thus increase the availability of training objects. In this article we deal with datasets for image classification, but our approach can be applied in other domains too.

Preliminary step: both the labeled and unlabeled sets can be refined to obtain more informative feature vectors. In this article, we used fc7 features of CNNs trained on ImageNet, but in principle, any type of features can be considered. Our particular choice was motivated because fc7 features work significantly better than traditional computer vision features (SIFT [104] and its variations). While this might seem counter-intuitive (using pre-trained CNNs on ImageNet, while we are solving the problem of limited labeled data), we need to consider that our datasets are different from ImageNet (they come from different distributions), and by using some other dataset to pre-train our networks, we are not going against the spirit of the idea of the chapter.

Step 1: the objects are assigned to initial probability distributions, needed to start the GTG. The labeled ones use their respective one-hot label representations, while the unlabeled ones can be set to a uniform distribution among all the labels. In presence of previous possessed information, some labels can be directly excluded in order to start from a multi-peaked distribution, which if chosen wisely, can improve the final results.

Step 2: the extracted features are used to compute the similarity matrix W. The literature [174] presents multiple methods to obtain a W matrix and extra care should be taken when performing this step, since an incorrect choice in its computation can determine a failure in the transductive labeling.

Step 3: once W is computed, graph transduction game can be played (up to convergence) among the objects to obtain the final probabilities which determine the label for the unlabeled objects.

The resulting labeled dataset can then be used to train a classification model. This is very convenient for several reasons: 1) CNNs are fully parametric models, so we do not need to store the training set in memory like in the case of graph transduction. In some aspect, the CNN is approximating in a parametric way the GTG algorithm; 2) the inference stage on CNNs is extremely fast (real-time); 3) CNN features can be used for other problems, like image segmentation, detection and classification, something that we cannot do with graph-transduction or with classical machine learning methods

accuracy	caltech		indoors		scenenet	
2% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.532**	**0.620**	**0.486**	**0.538**	**0.430**	**0.495**
SVM + CNN	0.473	0.539	0.434	0.468	0.370	0.417
CNN	0.266	0.235	0.341	0.323	0.205	0.178

F score	caltech		indoors		scenenet	
2% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.468**	**0.559**	**0.357**	**0.396**	**0.399**	**0.457**
SVM + CNN	0.388	0.455	0.319	0.327	0.352	0.377
CNN	0.181	0.151	0.187	0.172	0.191	0.167

Table 6.1: The results of our algorithm, compared with the results of linear SVM and CNN, when only 2% of the dataset is labeled. We see that in all three datasets and two different neural networks, our approach gives significantly better results than SVM or CNN.

accuracy	caltech		indoors		scenenet	
5% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.625**	**0.698**	**0.568**	**0.613**	**0.563**	**0.621**
SVM + CNN	0.605	0.675	0.516	0.580	0.511	0.601
CNN	0.457	0.444	0.456	0.466	0.408	0.438

F score	caltech		indoors		scenenet	
5% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.571**	**0.653**	**0.454**	**0.508**	**0.536**	**0.608**
SVM + CNN	0.542	0.626	0.426	0.505	0.501	0.590
CNN	0.372	0.358	0.345	0.306	0.394	0.419

Table 6.2: The results of our algorithm, compared with the results of linear SVM and CNN, when 5% of the dataset is labeled.

(like SVM). In the next section we will report the results obtained from state-of-the-art CNNs, and compare those results with the same CNNs trained only on the labeled part of the dataset.

6.3 Experiments

In order to assess the quality of the algorithm, we used it to automatically label three known realistic datasets, namely *Caltech-256* [57], *Indoor Scene Recognition* [125] and *SceneNet-100* [74]. *Caltech-256* contains 30607 images belonging to 256 different categories and it is used for object recognition tasks. *Indoor Scene Recognition* is a dataset containing 15620 images of different common places (restaurants, bedrooms, etc.), divided in 67 categories and, as the name says, it is used for scene recognition. *SceneNet-100* database is a publicly available online ontology for scene understanding that organizes scene categories according to their perceptual relationships. The dataset contains 10000 real-world images, separated into 100 different classes.

Each dataset was split in a training (70%) and a testing (30%) set. In addition, we further randomly split the training set in a small labeled part and a large unlabeled one,

accuracy	caltech		indoors		scenenet	
10% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.667**	**0.727**	**0.598**	**0.645**	**0.624**	**0.686**
SVM + CNN	0.658	0.724	0.576	0.635	0.622	0.660
CNN	0.577	0.598	0.553	0.567	0.571	0.584

F score	caltech		indoors		scenenet	
10% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.622**	**0.694**	0.509	0.574	0.609	**0.700**
SVM + CNN	0.612	0.686	**0.515**	**0.579**	**0.612**	0.650
CNN	0.519	0.533	0.478	0.471	0.565	0.570

Table 6.3: The results of our algorithm, compared with the results of linear SVM and CNN, when 10% of the dataset is labeled.

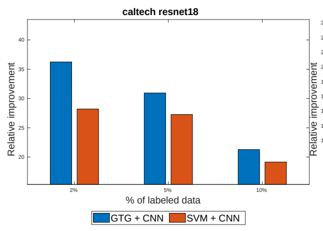
according to three different percentages for labeled objects (2%, 5%, 10%). For feature representation, we used two models belonging to state-of-the-art CNN families of architectures, ResNet and DenseNet. In particular we used the smallest models offered in PyTorch library, the choice motivated by the fact that our datasets are relatively small, and so models with smaller number of parameters are expected to work better. The features were combined to generate the similarity matrix W, as described in Eq. 7.6. The matrix for GTG model was initialized as described in the previous section. We ran the GTG algorithm up to convergence, with the pseudo-labels being computed by doing an *argmax* over the final probability vectors.

We then trained *ResNet18* (RN18) and *DenseNet121* (DN121) in the entire dataset, by not having a distinction between labels and pseudo-labels, using Adam optimizer [80] with $3 * 10^{-4}$ learning rate. We think that the results reported in this section are conservative, and can be improved with a more careful training of the networks, and by doing an exhaustive search over the space of hyper-parameters.

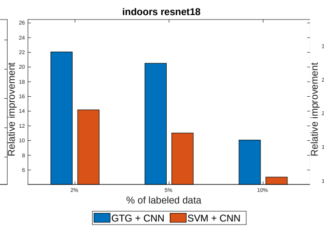
For comparison, we performed an alternative approach, by replacing GTG with a first-order information algorithm, namely linear SVM. While we experimented also with kernel SVM, we saw that its results are significantly worse than those of linear SVM, most likely because the features were generated from a CNN and so they are already quite good, having transformed the feature space in order to solve the classification problem linearly. No other transductive methods have been taken into consideration, since GTG has already been compared with them in [35, 157], showing that it performs better.

On Table I we give the results of the accuracy and F score on the testing set, in all three datasets, while the number of labels is only 2% for each of the datasets (400 observations for Caltech-256, 200 observations for Indoor, and 140 observations for Scenenet). In all three datasets, and both CNNs, our results are significantly better than those of CNNs trained only in the labeled data, or the results of the alternative approach when a linear SVM is used instead of GTG. Table II and Table III give the results of the accuracy and F score while the number of labeled images is 5%, respectively 10%. It can be seen that with the number of labeled points increasing, the performance boost of our model becomes smaller, but our performance still gives better (or equal) results to the alternative approach in all bar three cases, and it gives significantly better results than CNN in all cases.

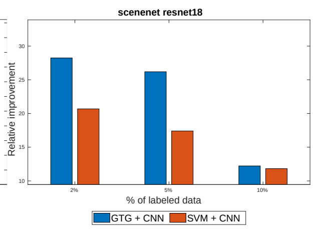
Figure 6.2 shows the results of our approach compared with the other approach and



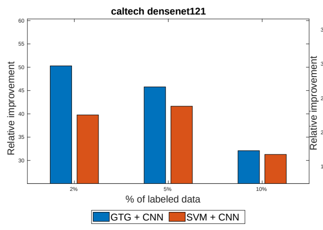
(a)



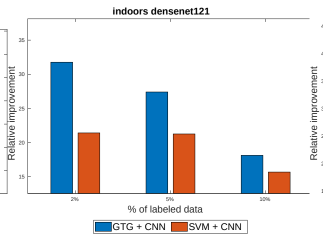
(b)



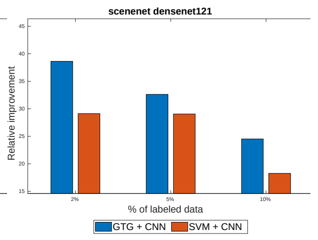
(c)



(d)



(e)



(f)

Figure 6.2: Results obtained on different datasets and CNNs. Here the relative improvements with respect to the CNN accuracy is reported. As can be seen, the biggest advantage of our method compared to the other two approaches, is when the number of labeled points is extremely small (2%). When the number of labeled points increases, the difference on accuracy becomes smaller, but nevertheless our approach continues being significantly better than CNN, and in most cases, it gives better results than the alternative approach.

with the results of CNN. We plotted the relative improvement of our model and the alternative approach over CNN. When the number of labels is very small (2%), in all three datasets we have significantly better improvements compared with the alternative approach. Increasing the number of labels to 5% and 10%, this trend persists. In all cases, our method gives significant improvements compared to CNN trained on only the labeled part of the dataset, with the most interesting case (only 2% of labeled observations), our model gives 36.24% relative improvement over CNN for *ResNet18* and 50.29% relative improvement for *DenseNet121*.

6.4 Conclusions and Future Work

In this chapter, we proposed and developed a game-theoretic model which can be used as a semi-supervised learning algorithm in order to label the unlabeled observations and so augment datasets. Different types of algorithms (including state-of-the-art CNNs) can then be trained on the extended dataset, where the "pseudo-labels" can be treated as normal labels.

Our method is not the only semi-supervised learning model used to train deep learning methods, and at this stage, we do not claim that our method is the best one. However, to the best of our knowledge, the other methods are directed towards deep learning and incorporated within the learning algorithm itself. On the contrary, we offer a different perspective, developing a model which is algorithm-agnostic, and which doesn't even need the data to be on feature-based format.

Part of the future work will consist on tailoring our model specifically towards convolutional neural networks and to make comparisons with other semi-supervised learning algorithms. In addition to this, we believe that the true potential of the model can be unleashed when the data is in some non-traditional format. In particular, we plan to use our model in the fields of bio-informatics and natural language processing, where non-conventional learning algorithms need to be developed. A direct extension of this work is to embed into the model the similarity between classes which has been proven to significantly boost the performances of learning algorithms.

Acknowledgements

This work was supported by Samsung Global Research Outreach Program. We thank the anonymous reviewers for their suggestions to improve the chapter.

Part II

Applicative Contributions

Chapter 7

Protein Function Prediction

Reference: ***S. Vascon***, M. Frasca, R. Tripodi, G. Valentini, and M. Pelillo. Protein function prediction as a graph-transduction game. *Pattern Recognition Letters*, 2018. DOI:10.1016/j.patrec.2018.04.002.

The content of this chapter is taken from the above reference.

Motivated by the observation that network-based methods for the automatic prediction of protein functions can greatly benefit from exploiting both the similarity between proteins and the similarity between functional classes (as encoded, e.g., in the Gene Ontology), in this chapter we propose a novel approach to the problem, based on the notion of a "graph transduction game." We envisage a (non-cooperative) game, played over a graph, where the players (graph vertices) represent proteins, the functional classes correspond to the (pure) strategies, and protein- and function-level similarities are combined into a suitable payoff function. Within this formulation, Nash equilibria turn out to provide consistent functional labelings of proteins, and we use classical replicator dynamics from evolutionary game theory to find them. To test the effectiveness of our approach we conducted experiments on five different organisms and three ontologies, and the results obtained show that our method compares favorably with state-of-the-art algorithms.

7.1 Introduction

The Automatic Function Prediction of proteins (AFP) consists in the computational assignment of the biological functions to the proteins of an organism [45]. It can be modeled as a multi-label classification task, since each protein may be associated with multiple functions, and represents one of the most challenging problems in the context of computational biology [18, 70, 126]. The increasing availability of large-scale networks constructed from high-throughput biotechnologies, representing functional similarities between proteins, such as co-expression networks, protein domain similarities, and protein-protein interactions just to mention a few, opened the avenue of a large class of graph-based algorithms, able to learn from the functional similarities between proteins [140].

These methods can transfer annotations from previously annotated (labeled) nodes to unannotated (unlabeled) ones through a learning process inherently transductive in nature. These learning processes exploit the so-called guilt-by-association principle [120], also known as homophily principle, by which proteins topologically close in the graph are likely to share their functions.

Starting from simple approaches based on local learning strategies [106], several other methods have been proposed in the literature, able to exploit in different ways the overall topology of the functional network. Some examples are represented by label propagation algorithms based on Markov [28] and Gaussian Random Fields [114, 177, 183], methods that integrate local learning strategies with simple weighted combination of diverse information [20], approaches based on the evaluation of the functional flow in graphs [159], algorithms based on Hopfield networks [41, 77], methods that exploit relationships between homologous proteins to connect networks of different species [113], while other approaches applied random walk based methods [84, 102] and their kernelized version by exploiting both local and global learning strategies [128, 153].

Despite their large diversity, network-based methods share the common property of using some notion of similarity between proteins to learn protein functions. The underlying assumption is that *similar* proteins tend to share *the same* functional class, an idea which is reminiscent of the homophily principle widely used in social network analysis [32] and which lies at the heart of virtually all classification algorithms.

This general approach has well-founded biological motivation [140], but also the similarity between functional classes (i.e. the Gene Ontology – GO terms to be predicted) plays a key role in the prediction of protein functions, as outlined by the recent CAFA2 (Critical Assessment of Functional Annotation) challenge for the AFP problem [70], since GO terms are not independent, but hierarchically related according to a directed acyclic graph [53]. To our knowledge, no network-based method has been proposed in the context of AFP to jointly consider the similarities between proteins and the similarities between functional classes. We hypothesize that network-based methods could significantly enhance their performance if they were able to contextually learn from both similarity between the examples (the proteins) and the similarity between the GO terms associated with the proteins their selves. This corresponds to the well-known biological principle for which a protein is fully characterized by the entire spectrum of its structural and functional properties, coded as a set of GO terms [53].

Motivated by this observation, in this chapter we present an application to AFP of a graph transduction model based on game-theoretic principles that conforms to a general classification principle which prescribes that *similar objects* should be assigned to *similar categories*, assuming the existence of a notion of similarity not only at the object but also at the category level. This is, in fact, a generalization of the standard homophily principle which suggests instead that similar objects should be placed *in the same* category.

Along the lines set forth in [35] within a standard homophily-based transductive setting, which ignored potential category-level similarities altogether, the AFP problem can be abstracted in terms of a multi-player non-cooperative game where the players represent proteins, the functional classes correspond to the (pure) strategies, and protein- and function-level similarities are combined in a suitable payoff function. Within this formulation, the Nash equilibrium concept for non-cooperative games turns out to offer a principled solution to the problem of finding a "consistent" labeling assignment [67, 111, 123].[1] In order to find Nash equilibria of our AFP games we use (multi-population) replicator dynamics, a well-known class of dynamical systems developed and studied in evolutionary game theory [165].

Our approach gives us the possibility not only to exploit the contextual information of a protein but also to find the most appropriate functions for the proteins in a determined context. In other words, the proposed model exploits two different kinds of information: structural and semantic. The structural information identifies how the proteins are organized in an organism, semantic information identifies how the functions of the proteins are structured. The integration of these two sources of information in a game theoretic model gives us the possibility to predict the combination of functions that are best suited for the proteins of a given organism. The above result is the most important methodological contribution of our work, which distinguishes it from existing AFP network-based algorithms.

To assess the effectiveness of the proposed game-theoretic approach, we conducted extensive experiments over different model organisms and using the ontologies of the GO, including thousands of functional classes and predictions for tens of thousands of proteins. We found that our proposed algorithms systematically obtain prediction results that are competitive with respect to the state-of-the-art network-based methods for protein function prediction.

7.2 Automatic Function Prediction Game

In this section, the specific model for the AFP problem is explained in detail.

We represent the proteins of an organism as players and their functions as strategies. The games are played between similar players, imposing only pairwise interactions. The payoff matrix is computed using a similarity function among GO terms and is weighted by the structural similarity between the proteins. The payoff function for each player is additively separable and is computed as described in Section 3.2.

Formulating the problem in this way we can apply equation (3.5) to compute the equilibrium state of the system, which corresponds to a consistent labeling of the data [111]. In fact, once stability is reached, all players play the strategy with the highest payoff. Each player arrives into this state not only considering its own strategies but also the strategies that other players are playing.

Our framework (see Fig.7.1) requires: *a)* the network that describes the interactions among the players, *b)* the similarity between the functions, *c)* the strategy space of the game and *d)* the payoff function.

[1]See [83] for a different approach based on MRF's.

7.2.1 Network of interactions

The network of interactions models the interactions among the players and is represented as a weighted graph $G = (V, E, \omega)$ where the set of nodes $V = \{1, \ldots, n\}$ are the players/proteins and $E \subseteq V \times V$ the affinity between them weighted by the function ω. The edges E of G represents the affinity of the players, highest the value of an edge the more likely the two connected players will play together. The graph G is thus represented with an affinity matrix $\boldsymbol{W} = n \times n$, and its role is to encapsulate the similarities (structural, functional, etc.) between pairs of proteins motivated by the fact that similar or interacting proteins should share common functional annotations, such as the participation to the same biological process, the catalysis of similar biochemical reactions or the location inside the same cellular organelle. The crucial point here is having a good similarity measure $sim(\cdot, \cdot) \rightarrow \mathbb{R}_{\geq 0}$ that represent the closeness of pairs i and j:

$$w_{i,j} = sim_W(i, j)\ \forall i, j \in V \tag{7.1}$$

In our experiments the networks of interactions have been constructed combining together eight different protein networks or directly using networks that natively combine different sources of data (Section 7.3.1).

On top of this network, a neighboring function $\mathcal{N}$ is applied to each player in order to sparsify the net and keeping only the more similar players for each one. The game-theoretic rationale that guided this choice is to select the subset of best matching co-players, while from a labeling perspective task this means to select the set of k neighboring of a point that weighs more in the labeling. Deciding the number of neighboring is often a tedious and stressful task which also appears in other methods, i.e. in k-NN classifier or in k-means clustering. To deal with this problem we decided to use two principled heuristics to select the k neighboring of a graph having n nodes/proteins:

GC which stands for Graph Connectivity. The rationale is that by fixing $k = \lfloor log_2(n) + 1 \rfloor$ we guarantee that the underlying graph is statistically connected [161]. Being connected, from a game-theoretic perspective, means that all the players, directly or indirectly through a common neighbor, have the chances to influence the others choices.

***k*-NN** with this heuristics we set $k = \lfloor \sqrt{n} \rfloor$. This rule of thumb is used in k-NN classifier to automatically tune the parameter k [31]. The rationale is that the graph-transduction game and the k-NN classifier are based on the same homophily principles where the labels are propagated from k labeled nodes to the unlabeled ones. If this heuristic holds for k-NN, it should also hold for our method.

Given a value for k, found with the two methods above, the neighbors $\mathcal{N}_i$ of protein i is the set of $j \in \{1...n\}$ s.t. $w_{i,j} \geq \alpha_i$ where α_i is the weight of the k-th most similar element to i [2].

The resulting neighboring set is obviously asymmetric. In order to make it symmetric we use the following policy: given two protein i, j if $j \in \mathcal{N}_i$ while $i \notin \mathcal{N}_j$ then $\mathcal{N}_j = \mathcal{N}_j \cup \{i\}$.

[2] $w_{i,:}$ are sorted in descendent order and α_i correspond to the value at position k

7.2.2 Function similarity graph

The function similarity graph models the similarity between pairs of GO terms from the used ontology. It is a weighted graph $G = (V, E, \omega)$ with self loop in which $\omega(i, j) \to \mathbb{R}_{\geq 0}$ weighs the similarity of the GO terms i and j. The graph G is represented as an $m \times m$ matrix $\mathbf{Z}$:

$$Z_{h,k} = sim_Z(h, k) \tag{7.2}$$

For the details of our implementation see Section 7.3.1.

7.2.3 Strategy space

The role of the strategy space $\mathbf{X}$ is to define all the possible associations between the n proteins and the m functions retrieved from an ontology. The space $\mathbf{X}$ is thus modeled as a $n \times m$ matrix in which each row corresponds to a mixed strategy $\mathbf{x}_i$ and each component x_i^h represents the strength of the association between the player (protein) i and the strategy (function) h. The strategy space $\mathbf{X}$ is the starting point of the game and can be initialized in different ways based on the fact that some prior knowledge about their labeling does exist or not. For the labeled proteins, since their functions are known, we use the following method:

$$x_i^h = \begin{cases} \frac{1}{f_i}, & \text{if } i \text{ has function } h. \\ 0, & \text{if protein } i \text{ does not have function } h. \end{cases} \tag{7.3}$$

where f_i is the number of terms associated with protein i.

For the testing proteins (the ones with no labels) we propose and evaluate two different initialization methods:

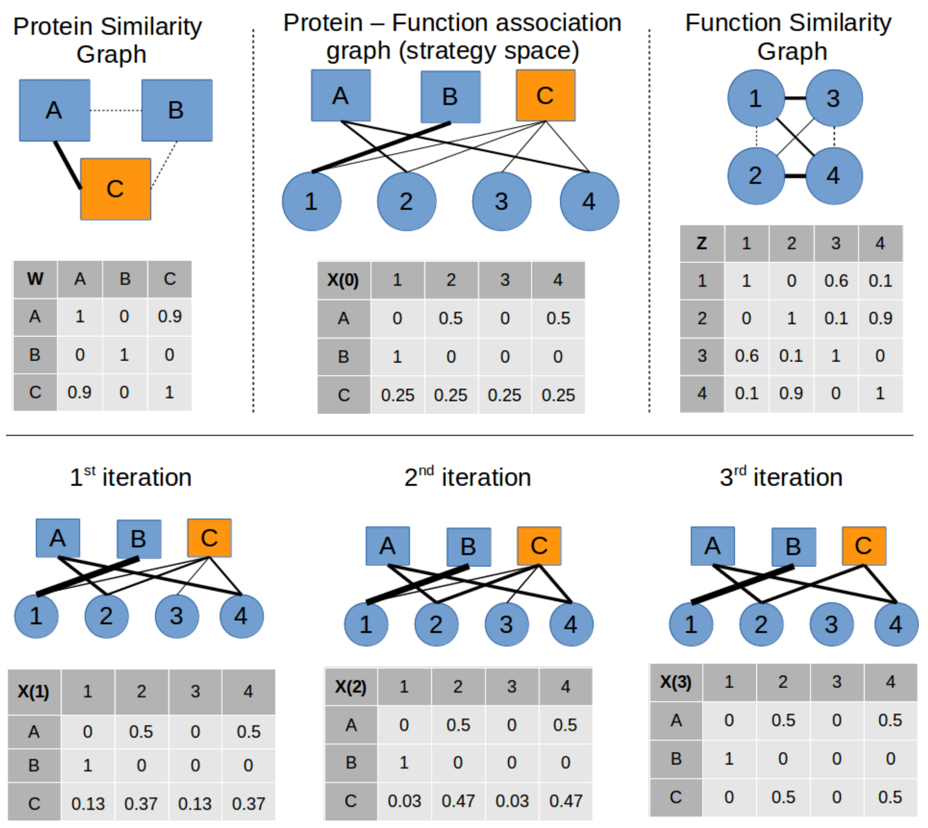
Without priors with this initialization, all the GO terms have the same probability of being associated to a protein:

$$x_i^h = \frac{1}{m} \quad \forall h = \{1 \dots m\} \tag{7.4}$$

With *k*-priors the rationale of this prior is to emphasize the labels assigned to the neighboring set of a certain protein with the idea that similar protein should be assigned to similar classes. Given a protein i and its set of neighboring proteins $\mathcal{N}_i$ (with labels), the prior is composed as follow:

$$x_i^h = \frac{1}{m} + \sum_{j \in \mathcal{N}_i} x_j^h \quad \forall h = \{1 \dots m\} \tag{7.5}$$

then $\mathbf{x}_i$ is normalized such that it add up to 1 $\left(x_i^h = \frac{x_i^h}{\sum_{h=1}^m x_i^h}\right)$ and remains in the m-dimensional simplex. The first term $\left(\frac{1}{m}\right)$ gives the chances also to other functionalities to emerge. If it was set to 0, this possibility would have been lost and the method will focus only on the functions that are assigned in the neighborhood.


Protein Similarity Graph
A
B
C
Protein – Function association graph (strategy space)
A
B
C
1
2
3
4
Function Similarity Graph
1
3
2
4
1st iteration
2nd iteration
3rd iteration

W	A	B	C
A	1	0	0.9
B	0	1	0
C	0.9	0	1

X(0)	1	2	3	4
A	0	0.5	0	0.5
B	1	0	0	0
C	0.25	0.25	0.25	0.25

Z	1	2	3	4
1	1	0	0.6	0.1
2	0	1	0.1	0.9
3	0.6	0.1	1	0
4	0.1	0.9	0	1

X(1)	1	2	3	4
A	0	0.5	0	0.5
B	1	0	0	0
C	0.13	0.37	0.13	0.37

X(2)	1	2	3	4
A	0	0.5	0	0.5
B	1	0	0	0
C	0.03	0.47	0.03	0.47

X(3)	1	2	3	4
A	0	0.5	0	0.5
B	1	0	0	0
C	0	0.5	0	0.5

Figure 7.1: The picture dissects the payoff function in order to understand what are the single components (three graphs on top) and what is happening to the assignment during the iteration of the dynamical system (eq 3.5). Consider the following situation: two similar proteins A and C ($A \in \mathcal{N}_C$) in which C has no prior on the functions (eq. 7.4) while A has the functions $2, 4$ assigned to it (eq. 7.3). In the first iteration we can already note that the labeling for C changes and becomes more similar to A.

7.2.4 Payoff Function

The payoff function has the role of assigning the gain that a certain player i receives when plays a strategy h (in graph-theoretic terms is the compatibility of assigning the function h to the protein i). The rationale is that we want to boost the association between similar players and similar GO terms. What we want for i, when plays with j, is that their labels are mutually affected, including the choice of i and j and also the set of similar labels to the ones associated to both the proteins. The set of similar functions is included with the idea that the correct labels could also be received from similar functions. This turns out to be:

$$u(x_i^h) = \sum_{j \in \mathcal{N}_i} \left(\left(w_{ij} \mathbf{Z} \right) \mathbf{x}_j \right)^h \tag{7.6}$$

and the expected payoff as,

$$u(\mathbf{x}_i) = \sum_{j \in \mathcal{N}_i} \mathbf{x}_i^T \left(w_{ij} \mathbf{Z} \right) \mathbf{x}_j \tag{7.7}$$

In this way, we weight the influence that each protein receive from its neighbors. According to eq. 7.7, we assumed that the payoff of protein i depends on: w_{ij}, i.e. the similarity with its neighborhood proteins $j \in \mathcal{N}_i$; $\mathbf{Z}$, the similarities among the functional terms; $\mathbf{x}_j$, the preferences of neighborhood protein $j \in \mathcal{N}_i$ and the preferences $\mathbf{x}_i$ of the protein i itself. With $u(x_i^h)$ and $u(\mathbf{x}_i)$ we can start the dynamics of the game according to equation (3.5). During each phase of the dynamics, a process of selection allows strategies with higher payoff to emerge and at the end of the process each player chooses its functionalities according to these constraints, which make the labeling consistent (for an example see Fig.7.1).

Table 7.1: Data base and type of data used to construct the integrated protein similarity network for *DanXen*, *SacPomDic* and em Dros

Database	Type of data
PRINTS [6]	Motif fingerprints
PROSITE [65]	Protein domains and families
Pfam [39]	Protein domain
SMART [96]	Simple Modular Architecture Research Tool (database annotations)
InterPro [115]	Integrated resource of protein families, domains and functional sites
Protein Superfamilies [56]	Structural and functional annotations
EggNOG [116]	Evolutionary genealogy of genes: Non-supervised Orthologous Groups
Swissprot [22]	Manually curated keywords describing the function of the proteins at different degrees of abstraction

7.3 Experiments

We applied different variants of our *graph transduction game* method (GTG) (see Section 7.3.4 for more details) to the prediction of the Cellular Component (CC), Molecular Function (MF) and Biological Processes (BP) ontologies of the GO considering different model organisms, ranging from the human to the fruit fly and the zebrafish, involving thousands of functional classes (see Table 7.2).

7.3.1 Data

We constructed five networks representing the functional similarity between proteins. Two networks include phylogenetically related organisms: a) the *DanXen* network encompasses *Danio rerio* (zebrafish) and *Xenopus laevis* (a small austral frog); b) the *SacPomDic* network includes *Saccharomyces cerevisiae*, *Schizosaccharomyces pombe* and *Dictyostelium discoideum* (unicellular eukaryotes). The third network (*Dros*) is reserved to *Drosophila melanogaster* (fruit-fly), the model organism for insects.

Such networks are constructed by integrating 8 different sources of information from public databases (Table 7.1), as briefly described in the following.

At first, we obtained different profiles for each protein by associating for each source of data a binary feature vector, whose elements are 1 or 0 according to the protein annotation for a specific feature (e.g. whether or not a protein includes a specific domain, or a specific motif). Then the protein profiles have been used to construct a set of similarity networks (one for each data type) with edge scores based on the computation of the classical Jaccard similarity coefficient between each possible pair of protein profiles, thus obtaining 8 different protein networks. Finally the networks have been combined by unweighted mean integration [154].

The remaining two networks contain proteins belonging to *Mus musculus* (*Mouse*) and *Homo sapiens* (*Human*) organisms, and have been retrieved from the STRING database, version 10.0 [146]. The STRING networks are highly informative networks merging several sources of information about proteins, coming from databases collecting experimental data like BIND, DIP, GRID, HPRD, IntAct, MINT or from databases collecting curated data such as Biocarta, BioCyc, KEGG, and Reactome.

Each of these networks are then used in Sec.7.2.1 to define the interactions between the players (proteins).

Table 7.2: Number of proteins and GO terms with at least 2 annotations in each protein network.

Network	Proteins	CC terms	MF terms	BP terms
DanXen	6250	125	198	1502
SacPomDic	15836	858	1331	3934
Dros	3195	414	485	2985
Mouse	20648	701	1313	7309
Human	19247	860	1688	6298

As class labels (groundtruth) for the proteins included in our networks, we used the Gene Ontology CC, MF and BP experimental annotations extracted from the Swissprot database[3].

In order to enlarge the number of GO terms to be predicted, while preserving the minimum information needed for the functional predictions, we removed only GO terms having less than two annotations, thus resulting in a number of classes ranging from 125 (CC ontology in *DanXen*) to 7309 (BP ontology in *Mouse* – Table 7.2).

The similarity between the GO terms for each integrated network and each ontology could be in principle computed using semantic similarity measures based e.g. on the Resnick or Lin measures or other recently proposed variants [17], but to show the applicability of our proposed method we adopted a simple Jaccard similarity measure between the annotations of each GO term. These similarities correspond to the entries Z_{ij} in Eq.7.2.

[3]`http://www.expasy.org/` checked 19th May 2016

7.3.2 State-of-the-art methods compared with GTG

We compared GTG with several classical and state-of-the-art graph-based algorithms just applied to the the AFP problem: *Random Walk* (RW) and *Random Walk with Restart* (RWR), the *guilt-by-association* method (GBA), the *label propagation* algorithm (*LP*), three methods based on Hopfield nets, the *Gene Annotation using Integrated Networks* (GAIN), the *Cost-Sensitive Neural Network* (COSNet) and the *COSNet Multi-functionality-based ranking* (COSNetM), the *Multi-Source k-Nearest Neighbors* (MS-kNN), and the *RAnking of Nodes with Kernelized Score Functions* (RANKS). The compared algorithms are briefly described below.

RW A t-step random walk algorithm [102] associates a protein $i \in V$ with a score corresponding to the probability that a random walk in G starting from positive nodes ends at node i after t random steps. The iterative procedure to update the probabilities uses at each step a transition matrix $\boldsymbol{T}$ obtained from $\boldsymbol{W}$ by row normalization, i.e. $\boldsymbol{T} = \boldsymbol{D}^{-1}\boldsymbol{W}$, where $\boldsymbol{D}$ is a diagonal matrix $D_{ii}|_{i=1}^{n}$, with $D_{ii} = \sum_j w_{ij}$.

RWR After many steps the random walker in the RW algorithm may forget the prior information coded in the initial probability vector (0 for nodes in $V \setminus V_+$ and $1/|V_+|$ for nodes in V_+, where V_+ is the set of positive proteins for the current GO term). Thus, the RWR algorithm at each step allows the walker to move another random step with probability $1 - \theta$, or to restart from its initial condition with probability θ.

GBA Family of algorithms relying upon the *guilt-by-association* principle, asserting that similar proteins are more likely to share similar functions [138]. Usually, the GBA discriminant score of a protein i for a given GO term is obtained as the maximum of the weights connecting i to neighboring proteins associated with that term (that is the positive proteins).

LP The label propagation algorithm, based on Gaussian kernels, iteratively propagates labels from labeled proteins to the unlabeled ones until convergence [183]. During the label propagation the initial known labels are preserved.

GAIN An algorithm assigning labels to unlabeled proteins by minimizing the energy function of a Hopfield net [61] associated to the protein network [77]. The net dynamics involves solely the unlabeled proteins, whose activation thresholds are set to 0, and whose initial state is set according to the labeling provided by the current GO term. The equilibrium point reached by the dynamics provides the binary labeling of unlabeled proteins. To provide even a ranking of proteins, in the present work the neuron energy at equilibrium is adopted as ranking score, following the approach presented in [43].

COSNet Suitable for unbalanced data like the GO term annotations, this algorithm extends GAIN by substituting the classical Hopfield net with a parametric Hopfield net [11]. The parameters, namely the neuron activation values and thresholds, are automatically learned in order to cope with the labeling imbalance [40].

COSNetM An extension of COSNet exploiting the multifunctional properties of genes [42].

MS-kNN One of the top-ranked methods in the recent CAFA2 international challenge. MS-kNN integrates several proteins sources/networks by applying the *k-Nearest Neighbours* algorithm [2] to each network independently, and then averages the obtained individual scores [88].

RANKS A ranking method adopting a suitable kernel matrix so as to extend the similarity between two proteins also to non neighboring proteins [128]. The score of each protein i for a given GO term is defined through a local function that takes into account the neighborhood of each protein in the projected Hilbert space, according to the global topology of the underlying network.

For COSNet and RANKS we used the source code publicly available as **R** package [44, 153], and for the other methods we used the code provided by the authors or our in-house software implementations. The parameters required by our GTG approach and the other considered methods in this work have been learned through internal tuning on a small subset of training data.

7.3.3 Experimental setup

To evaluate the generalization performance of the compared methods we applied a 5-fold cross-validation experimental setting. According to the recent CAFA2 international challenge, to compare the results we considered both the "per class" Area Under the Precision Recall Curve (AUPRC), and the "per-example" multiple-label F-score. More precisely if we indicate as $TP_j(t)$, $TN_j(t)$ and $FP_j(t)$ respectively the number of true positives, true negatives and false positives for the protein j at threshold t, we can define the "per-example" multiple-label precision $Prec(t)$ and recall $Rec(t)$ at a given threshold t as:

$$Prec(t) = \frac{1}{n}\sum_{j=1}^{n}\frac{TP_j(t)}{TP_j(t)+FP_j(t)} \qquad Rec(t) = \frac{1}{n}\sum_{j=1}^{n}\frac{TP_j(t)}{TP_j(t)+FN_j(t)} \tag{7.8}$$

where n is the number of examples (proteins). In other words $Prec(t)$ (resp. $Rec(t)$) is the average multi-label precision (resp. recall) across the examples. The F-score multi-label depends on t and according to CAFA2 experimental setting, the maximum achievable F-score ($Fmax$) is adopted as the main multi-label "per-example" metric:

$$Fmax = max_t \frac{2Prec(t)Rec(t)}{Prec(t)+Rec(t)} \tag{7.9}$$

To pursue a fair comparison, the cross-validation has been performed by adopting a non-stratified partition of proteins in folds unique for all methods. The AUPRC results have been averaged across folds having at least one annotated protein (otherwise the AUPRC by definition is meaningless).

7.3.4 GTG variants and settings

In our experiments we applied different variants of the GTG method, depending on the choice of the neighboring function (Section 7.2.1) and of the priors used to initialize the strategy space (Section 7.2.3) – see Table 7.3 for more details.

Table 7.3: Variants of GTG . The column *name* contains the name used for the particular setting in the chapter; *neighbour size* refers to the sec. 7.2.1; *symmetric* if yes the neighbourhood is symmetrized; *prior* if yes the *k*-prior defined in sec. 7.2.3 to initialize the strategy space is used, otherwise no informative prior (uniform distribution) is used.

Name	Neighbour size	Symmetric	Prior
GTG α	GC	No	No
GTG β	GC	Yes	No
GTG γ	GC	Yes	Yes
GTG δ	*k*-NN	Yes	Yes

7.4 Results

In this section we performed an extended cross-validated experimental comparison between GTG and nine state-of-the-art network-based algorithms using 5 different networks, and successively an ”ablation study” to analyze the impact of different ratios of labeled examples on the learning behavior of GTG .

7.4.1 Cross-validated experimental results

We performed an extended experimental comparison between GTG α , GTG β , GTG γ and GTG δ methods and nine state-of-the-art network-based algorithms using 5 different networks (*DanXen*, *SacPomDic*, *Dros*, *Mouse* and *Human*) labeled with terms of the three GO ontologies (BP, MF and CC). In this section we present and discuss the average results across classes (using the AUPRC metric) and across proteins (using the *Fmax* metric) for each network, considering separately the BP, MF and CC ontologies, thus resulting in 15 sets of average results involving thousands of functional classes and tens of thousands of proteins of different model organisms.

Multi-label *Fmax* results are summarized in Table 7.4. Independently of the model organism and the biological ontology considered, our proposed game theory-based transductive methods largely outperform the other methods (Table 7.4). In particular GTG γ and GTG δ achieve better results than the other methods (see the last two rows in Table 7.4). In several cases the relative improvement with respect to the best competing state-of-the-art method is close or larger than 50%: for instance with the MF ontology in *DanXen*, *Dros*, *SacPomDic* and *Mouse* networks, or with the BP ontology in *DanXen*, *Human* and *Mouse*. Also with the other ontologies and the other model organisms considered in this work the improvement with respect to the other network-based methods is impressive.

The only method that attains comparable results (but only limited to the CC ontology in *Human*) is the *MS-kNN* algorithm, one of the top ranked methods in the recent CAFA2 challenge for protein function prediction [70] (Table 7.4). Note that GTG α and GTG β , which uses a uniform distribution to initialize the strategy space **X**, usually obtain worse results than the other proposed variants GTG γ and GTG δ that adopt “neighborhood-aware” priors to initialize **X** (Section 7.2.3). Nevertheless, in

Table 7.4: Fmax results across the terms of the CC, MF and BP ontology for *DanXen*, *Dros*, *SacPomDic*, *Human* and *Mouse* integrated protein networks. For each ontology and network the best results are highlighted in bold.

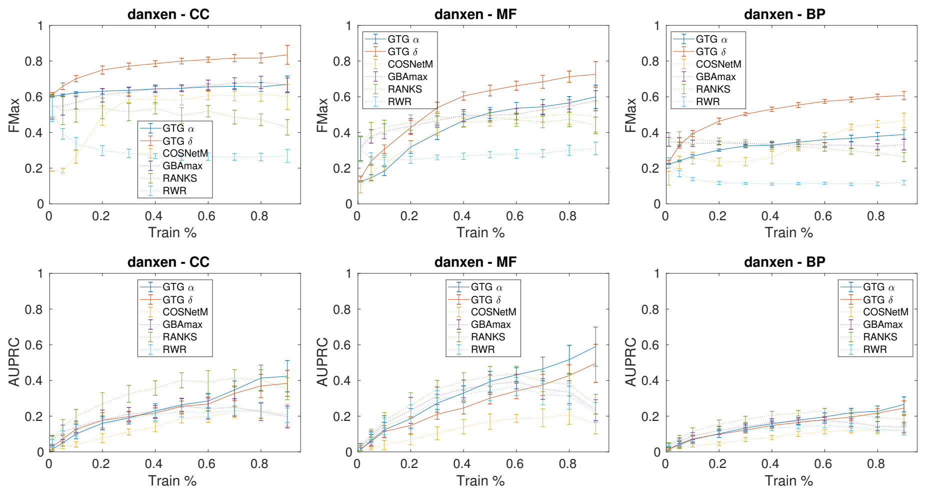
Fmax	Danxen			Dros			SacPomDis			Human			Mouse		
	CC	MF	BP	CC	MF	BP	CC	MF	BP	CC	MF	BP	CC	MF	BP
RANKS	0.5418	0.5075	0.4402	0.5483	0.3522	0.3201	0.6893	0.2951	0.4021	0.2804	0.1157	0.1467	0.2970	0.1354	0.1197
RWR	0.2588	0.2860	0.1156	0.1235	0.2237	0.0744	0.0718	0.1263	0.0662	0.0604	0.0374	0.0493	0.0545	0.0453	0.0367
COSNet	0.6055	0.4849	0.4542	0.5698	0.3811	0.2946	0.7128	0.4735	0.3364	0.1089	0.0847	0.0494	0.5694	0.3819	0.2292
COSNetM	0.6031	0.4831	0.4547	0.4405	0.3262	0.1820	0.5857	0.3953	0.2356	0.1953	0.1369	0.1572	0.4006	0.1958	0.1601
GAIN	0.3346	0.1796	0.2603	0.6215	0.1782	0.3642	0.7093	0.1054	0.1930	0.6015	0.5517	0.0828	0.5934	0.1118	0.0808
GBA	0.6572	0.5336	0.3314	0.5152	0.4532	0.2509	0.5002	0.5138	0.3746	0.3072	0.2365	0.1914	0.3285	0.2327	0.1544
LP	0.6678	0.5513	0.4328	0.6473	0.3687	0.4005	0.7244	0.2411	0.3006	0.6225	0.5361	0.263	0.6114	0.2535	0.2475
MS-kNN	0.3517	0.3574	0.2769	0.7120	0.5361	0.5138	0.8173	0.5386	0.5332	**0.6419**	0.5498	0.2276	0.6325	0.4055	0.2123
RW	0.2322	0.2767	0.0943	0.0962	0.1220	0.0562	0.0436	0.0573	0.0261	0.0481	0.0271	0.0374	0.0420	0.0335	0.0282
GTG α	0.6589	**0.5516**	0.3698	0.6315	**0.5762**	0.4037	0.7254	**0.6650**	0.4622	0.5856	**0.5916**	**0.3248**	0.5959	**0.5730**	**0.3108**
GTG β	0.6670	**0.5602**	0.3814	0.6403	**0.5966**	0.4119	0.7313	**0.6126**	0.4427	0.5852	**0.5966**	**0.3278**	0.5939	**0.5832**	**0.3127**
GTG γ	**0.8107**	**0.7188**	**0.6316**	**0.8283**	**0.7627**	**0.5881**	**0.8956**	**0.7953**	**0.6830**	0.6389	**0.6382**	**0.3902**	**0.6531**	**0.6301**	**0.3643**
GTG δ	**0.8138**	**0.7088**	**0.5973**	**0.8184**	**0.7489**	**0.5848**	**0.8989**	**0.7728**	**0.6694**	0.6397	**0.6346**	**0.3804**	**0.6568**	**0.6119**	**0.3521**

most cases, GTG α and GTG β too achieve comparable or significantly better results than all the other competing methods (Table 7.4).

Table 7.5: Mean AUPRC results averaged across the terms of the CC, MF and BP ontology for *DanXen*, *Dros*, *SacPomDic Human* and *Mouse* integrated protein networks. For each ontology and network the best results are highlighted in bold.

AUPRC	Danxen			Dros			SacPomDis			Human			Mouse		
	CC	MF	BP	CC	MF	BP	CC	MF	BP	CC	MF	BP	CC	MF	BP
RANKS	0.3014	0.266	0.1672	0.2972	0.3038	0.1879	0.2808	0.2183	0.1666	0.3061	0.0988	0.1109	0.2376	0.0933	0.0848
RWR	0.2318	0.2977	0.1399	0.1060	0.2400	0.0979	0.0920	0.2260	0.0880	0.219	0.0630	0.0650	0.157	0.0630	0.0530
COSNet	0.2556	0.2409	0.1469	0.2347	0.2389	0.1398	0.2526	0.1890	0.1240	0.1894	0.0319	0.0452	0.1726	0.072	0.0575
COSNetM	0.2473	0.2400	0.1475	0.2225	0.2363	0.1373	0.2558	0.1860	0.1220	0.1870	0.0317	0.0446	0.1713	0.0716	0.0577
GAIN	0.0216	0.0271	0.0099	0.0332	0.012	0.0145	0.0186	0.0017	0.0044	0.0199	0.0027	0.0022	0.0179	0.0017	0.0024
GBA	0.3213	0.4951	0.2203	0.2746	0.4577	0.1899	0.3074	0.5036	0.2115	0.3314	**0.1129**	**0.1293**	0.2573	**0.1161**	**0.1024**
LP	0.0308	0.0302	0.0187	0.0532	0.0279	0.0359	0.0256	0.0054	0.0112	0.2228	0.0692	0.065	0.1528	0.0563	0.0447
MS-kNN	0.1550	0.1297	0.0833	0.1475	0.1724	0.083	0.2009	0.1496	0.0987	0.1837	0.0109	0.0244	0.1337	0.0105	0.0136
RW	0.1998	0.3903	0.1347	0.0744	0.1476	0.0724	0.0312	0.0903	0.0318	0.1248	0.0402	0.0382	0.0938	0.0383	0.0336
GTG α	**0.4325**	**0.5462**	**0.2698**	**0.3904**	**0.5448**	**0.2379**	**0.5030**	**0.5735**	**0.3131**	**0.3805**	0.1025	0.1194	**0.2739**	0.1076	0.0884
GTG β	**0.4614**	**0.5565**	**0.2747**	**0.4151**	**0.5760**	**0.2448**	**0.5326**	0.5002	**0.2916**	0.3289	0.0933	0.1046	0.2311	0.1017	0.0733
GTG γ	0.3169	0.3534	**0.2357**	**0.2988**	**0.4626**	**0.2283**	**0.3632**	0.3933	**0.2684**	0.2593	0.0692	0.0761	0.1508	0.0632	0.0427
GTG δ	**0.3364**	0.4068	**0.2300**	**0.3213**	0.4554	**0.2349**	**0.4439**	0.4238	**0.2746**	0.2878	0.0721	0.0819	0.1855	0.0674	0.0490

Considering the AUPRC per-class metric, our proposed methods and in particular GTG α and GTG β achieve competitive results with respect to the other state-of-the-art network based algorithms, even if the results are not so compelling as with the per-example metric. Indeed GTG α obtains better AUPRC results with respect to all the other competing methods (boldfaced in Table 7.5) in 11 out of the 15 pairs of network/ontology considered in this experimental comparison, while *GBA*, the second best method, is equal or better than all the other algorithms in 4 out of the 15 network/ontology pairs. Nevertheless, we outline that our methods behave largely better with the *Fmax* per-example metric, since both GTG γ and GTG δ achieve better average results in 14 network/ontology pairs (Table 7.4). This is not surprising, since the proposed graph-based transductive approach is conceived for a per-example multi-label learning: for each protein the labels (GO terms) are learned together in the same learning process taking into account the relationships between GO terms coded in the payoff function (eq. 7.7) used to compute the payoff u_i for each protein $i \in I$ (Section 3.2). Hence it is quite natural that our approach shows better results with the hierarchical *Fmax* score, by which we take into account the multi-labels (i.e. the en-



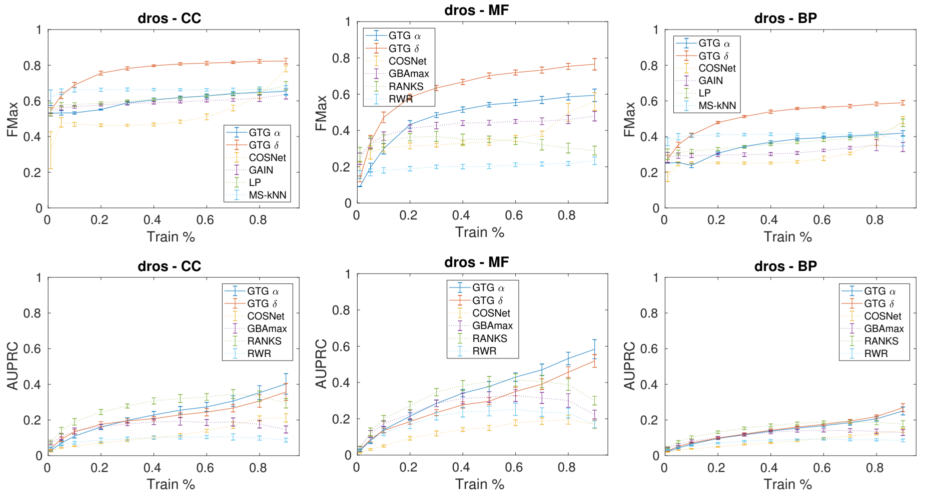
(a) DANXEN - CC (b) DANXEN - MF (c) DANXEN - BP

Figure 7.2: Performance of GTG α and GTG δ and top-four competing methods, on *Danxen* by varying the ratio of labeled examples in the training set. First row: average Fmax results with CC, MF and BP GO ontologies. Second row: average AUPRC results with CC, MF and BP GO ontologies. Error bars refer to the standard deviation across 20 hold-out repetitions.

tire set of GO terms) correctly predicted for each protein, while reasonable but not so compelling results are obtained with the AUPRC metric computed on a per-class basis. Moreover, from a biological standpoint, in most cases biologists are more interested in the set of GO terms associated with a specific protein or a set proteins, than in the predictions for a specific term, since the functional and structural characteristics of a given protein are captured by the entire set of functions (GO terms) associated with the protein under study. We note that for the per-class metric we did not report the classical AUROC (Area Under the Receiver Operating Characteristic curve), but the AUPRC instead. Indeed in the context of the protein function prediction, most of the GO terms are imbalanced, since positive examples usually represent a small subset of the overall number of proteins. In this imbalanced setting, from both a machine learning [26] and a bioinformatics standpoint [134] it is well-known the AUPRC provides a more reliable metric to assess the overall performances of the prediction methods.

7.4.2 Ablation study

Since performances of graph-transduction methods are typically affected by the number of labeled elements, we run a series of experiments to assess the behavior of the proposed model and the ones of the competitors when the number of annotated proteins (labeled elements) increases with respect to the overall available examples. To this end we performed an *hold-out* evaluation with different percentage of the split



(a) DROS - CC (b) DROS - MF (c) DROS - BP

Figure 7.3: Performance of GTG α and GTG δ and top-four competing methods, on *Dros* by varying the ratio of labeled examples in the training set. First row: average Fmax results with CC, MF and BP GO ontologies. Second row: average AUPRC results with CC, MF and BP GO ontologies. Error bars refer to the standard deviation across 20 hold-out repetitions.

$s = \{0.01, 0.05, 0.1, 0.2, ..., 0.9\}$ between the training and test set, where s represents the ratio of training examples. We realized the split in a stratified way in order to maintain in both training and test set the same ratio between labeled and unlabeled examples.

We performed 20 repetitions for each value of s in order to assess the stability of the analyzed methods. The performances (Fmax and AUPRC) of the two more valuable representative variants of GTG (GTG α and GTG δ) have been evaluated in comparison with the other competing methods introduced in Section 7.3.2, using *DanXen* and *Dros* networks and all the three ontologies. Figs. 7.2 and 7.3 show GTG results in comparison with the best four performing methods for each organism, ontology and metric considered.

Fmax results show that GTG and in particular GTG δ significantly outperforms all the other methods across the overall range of s values (Fig. 7.2 and 7.3, first row). With AUPRC we do not obtain so compelling results, since other methods (e.g. *RANKS*) achieve better results, especially for low percentages of labeled examples in the training set (Fig. 7.2 and 7.3, second row). These results are not surprising since GTG learns the overall set of GO labels associated with each protein, while other methods, such as *RANKS*, are designed to learn the GO labels using a per-class strategy, without taking into account the relationships between GO terms. It is worth noting that the differences between GTG α and GTG δ are stronger in terms of Fmax while in AUPRC are close to each other. This is explained by the fact that GTG δ uses a smart initialization with priors which limits the possible ranges of labeling solutions. Analyzing the learning behavior of the different models with respect to the percentage of available training examples, as expected, for GTG the more the number of labeled points the better the performances, while this in not always true for the competing methods. Hence GTG has a more stable and predictable behavior concerning performances and percentage of the labeled samples with respect to the other methods. For example, in the MF ontology with the AUPRC metric, while we can note a constant increment of the performance of GTG methods when more labeled training examples are available, for all the other methods we can observe an opposite trend for large values of s (Fig. 7.2 and 7.3). Moreover, GTG ranks as one of the top-3 methods (the lower curve of GTG is above the worst-best competitor) independently from the organism, ontology or metric taken into account.

Summarizing, results with different ratios of labeled examples confirm the effectiveness of the proposed approach and also show that GTG provides consistent and robust results, improving its performance when more labeled examples are available.

7.4.3 Discussion

The GTG α results in terms of AUPRC, and the ones of GTG γ and GTG δ in terms of the multi-label Fmax score, show that our game-theoretic-based approach can introduce significant improvements in network-based algorithms for AFP problems. The motivation of the success of the proposed approach is likely due to the fact that the game-theoretic model mimics, in a mathematical framework, the driving principle of the "guilt-by-association", and extends it by embedding in the learning process not only the similarities between proteins, but also the similarities between the functional terms of the GO. From a graph-learning standpoint this translates into a network-based semi-supervised approach by which the transductive process contextually learns all the labels (GO terms) associated with a specific protein, thus exploiting at the same time the relationships between both GO terms and proteins. Furthermore, the experimental evidence suggests us the following *rule-of-thumb*: if one is interested in optimizing a per-example metric (like Fmax) prior knowledge should be added to the strategy space (see Sec.7.2.3) and the neighborhood should be symmetric 7.2.1. To optimize a per-class metric (like the AUPRC) using a uniform distribution in the strategy space and an asymmetric neighboring system improve the results. In the first case, this is explained by the fact that each testing sample is treated independently focusing more on the set of possible functions assigned to the neighboring proteins. In the latter case we are interested in a (more) global metric, so assuming no prior knowledge for each sample let the protein-function assignment to naturally emerge from the data, thus capturing phenomena that span across the samples. Moreover, the results of Sec. 7.4.2 confirms the stability, consistency and predictability of the learning behavior of GTG across different experimental settings, showing that our approach is able to predict GO terms even when we may dispose of relatively few annotated proteins, and that the performances nicely improve when more labeled examples are available.

7.5 Conclusions

In this chapter we have introduced a new game-theoretic perspective to the protein function prediction problem, which is motivated by the observation that network-based methods should take advantage not only of similarity information at the level of proteins, as they usually do, but also of similarities between functional classes, which are available in the Gene Ontology. Accordingly, we set up an abstract game whereby proteins (the players) have to choose a strategy (a functional class), in a non-cooperative manner, to get a payoff which is related to *both* protein-level and function-level similarities. It turns out that the Nash equilibria of this AFP game are related to a well-known notion of "consistency" in a contextual labeling problem [67, 111, 123]. The results of extensive experiments confirm our original intuition that it does pay to incorporate functional-class similarities into network-based prediction algorithms, and demonstrate the power of simple game-theoretic dynamics to address the Automatic protein Function Prediction problem.

Chapter 8

Multimodal Verbe Sense Disambiguation

Reference: ***S. Vascon****, G. Bigaglia*, L. Giudice, and M. Pelillo. Multimodal Verb Sense Disambiguation using Graph Transduction Games. *= equal contribution. ***under submission***

The content of this chapter is taken from the above reference.

Verb Sense Disambiguation is a well-known task in NLP, the aim is to find the correct sense of a verb in a sentence. Recently, this problem has been extended in a multimodal scenario, by exploiting both textual and visual features of ambiguous verbs. The sense of a verb is assigned by the actual content of the image paired with it. In this work, such task will be performed in a transductive semi-supervised learning (SSL) setting in which a small amount of labeled information is used to perform the verb-sense classification. The SSL is performed through a game-theoretic framework, in which each multimodal representation of a pair image-verb is a player and the possible strategies correspond to the set of senses that the verb belongs to. A Nash Equilibria in this non-cooperative game corresponds to a consistent labeling between verbs and their possible senses. Experiments have been carried out on the recently published dataset VerSe. The results achieved outperform the current state-of-the-art by a large margin.

8.1 Introduction

Word Sense Disambiguation is a common task in NLP, its goal is to assign a sense to each word in a sentence. When restricting to solely verb nouns, the task is known as Verb Sense Disambiguation (VSD). For instance, considering the verb *run*, the most common sense is the one related to moving quickly, but there are other common senses, like the one related to machine operations (*the washing machine is running*) or to cover

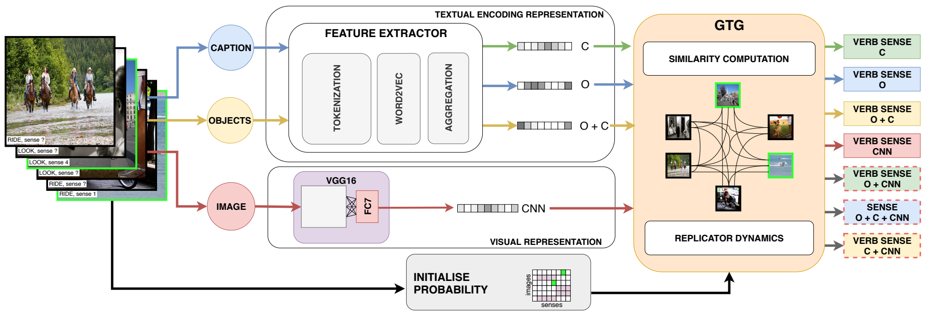


Figure 8.1: Pipeline of the algorithm considering both labeled (green border) and unlabeled images (black border).

a distance (*this train runs hundreds of miles every day*); all these senses share the same verb, but they have quite different meanings.

The VSD is thus an utmost important task, which affect different domains, for example in an image retrieval scenario, we want that the search engine being able to find the correct results in ambiguous queries and to group images by sense [29]. In addition to the typical NLP task, VSD can be brought to a Computer Vision scenario, taking into account problems like Action Recognition (AC) and Human Object Interaction (HOI); however, such tasks typically do not consider the ambiguity of verb senses. Typical learning paradigm in WSD/VSD context are supervised [109] and unsupervised [52, 112, 118, 124].

In this chapter we tackle a specific problem: performing the multimodal VSD in a semi-supervised (SSL) scenario. The rationale is that, by moving in a SSL framework, the required amount of labeled information is much lower than supervised approaches. Moreover, with a little supervision the performances increase significantly [170].

The transductive process that we choose is based on a game-theoretic model called *Graph Transduction Games* (GTG) [35]. The motivations that drive our choice toward the GTG algorithm were that it has been successfully applied in a WSD context [150] and proved to perform better than other alternatives (like Label Propagation [182]) in SSL tasks [158]. The VSD will be accomplished in both a unimodal and multimodal approach, taking into account both the image, its description and tags. The experiments are performed on the recently published VerSe dataset [51]. Our contributions are thus four-fold:

1 we apply GTG for the first time in a Multimodal VSD setting.

2 we exploit the principle of label consistency in the verb-sense disambiguation task, which means that we consider the similarities among all the verbs during inference.

3 we outperform the previous state-of-the-art by considering just 2 labeled pairs $<$ image, verb $>$ per sense. We further provide experimental evidence when the labeled part increases in size.

4 by considering senses as labels (rather than encoded entities as in [51]) we avoid the creation of time consuming hand-crafted queries, and the consequent data crawling to generate the multimodal represention of the senses.

8.2 Related works

Visual Sense Disambiguation for nouns has a very limited literature. One of the first approaches is in [7] which used a statistical model based on joint probabilities between images and words. [132] used LDA to discover a latent space by exploiting dictionaries definitions to learn distributions that represent senses. A similar task was accomplished in [12] that tried to solve linguistic ambiguities using multimodal data. In [14] they used multimodal data for semantic frame labeling . However such techniques are noun-oriented. The first attempt to perform a fully verb-oriented VSD was introduced in [52], which designed a variation of Lesk algorithm [95] that uses the multimodal sense encoding and the multimodal input encoding respectively as the definition and context for the algorithm. Recently, the SSL has been used for WSD: in [150] a similar game-theoretic model as our is used, while in [170] an LSTM and label propagation are combined to perform inference.

8.3 Verb Sense Disambiguation with GTG

In this section, we dissect the different components of our method.

8.3.1 Feature descriptors

Image features are extracted following [51] setup. There are three possible encodings: textual, visual and multimodal (see Fig. 8.1).

Visual features: Input images are fed into a pre-trained VGG16[1] model [141], and the output of the last FC layer is used as feature representation, resulting in a vector of 4096 elements for each image. Such vector is then unit-normalised.

Textual features: As in [51], experiments on text have been run on two possible setups: using VerSe textual data annotations (GOLD) or by generating them through state-of-art object detectors and image descriptors (PRED). In the latter scenario, object labels have been extracted using the VGG16 model described previously. Since the VGG16 net classifies images without performing object detection, in [51] they thresholded the output of the SoftMax layer taking only classes that had a score greater than 0.2 (or the highest in case of empty result). This allows to obtain multiple classes/objects per image. Captions have been generated with NeuralTalk2[2] [78, 160].
For what concerns the encoding, either captions, objects labels or a concatenation of the two can be used. The encoding is performed through word2vec [110] embedding in

[1]We used the PyTorch implementation of VGG
[2]https://github.com/karpathy/neuraltalk2

a 300-dimensional space. It is based on a Gensim model [129] pre-trained on Google News dataset. For each word composing the textual input, a word2vec is extracted. After that, they are aggregated by mean and unit-normalised, resulting in a vector for each image.

Multimodal features: To perform multimodal VSD, textual and visual features can be combined. In [51], beyond the vector concatenation, Canonical Correlation Analysis and Deep Canonical Correlation Analysis are explored. Nevertheless their performances were poorer than concatenation ones, hence we explored only this last option.

8.3.2 Graph Transduction Games for VSD

The Graph-Transduction Game (GTG) [35] is a semi-supervised learning algorithm in which the inference is performed in a transductive manner, hence considering both labeled and unlabeled information. The classification task is casted as a non-cooperative game in which the players (observation of a dataset) play a game in which each of them has to make a choice among a set of possible strategies (labels) and it receives a reward proportional to the compatibility of the choices made by the opponents. The more the players are similar the more likely they will affect each others, hence share the same label. In this specific VSD setting the set of players and the strategies correspond respectively to the pair <image, verb> and the possible senses of the verb. We refer the reader to the original paper [35] for the technical and theoretical details. Briefly, the GTG requires as input three components: *i)* the set of labeled players L and the unlabeled ones U, *ii)* the similarity graph between the players and *iii)* an initial assignment between players and labels.

Similarity between players We define the similarity between players i and j as the cosine[3] of their features embedding f_i and f_j respectively. The features are defined in section 8.3.1.

Probability Initialization The initial assignment between players (<image,verb>) and strategies (senses) is initialized distinguishing between *labeled* and *unlabeled* players. The set L represents the data for which we know the actual label hence a one-hot vector corresponding to the actual label is assigned to them, while for U a uniform distribution among the feasible labels is generated:

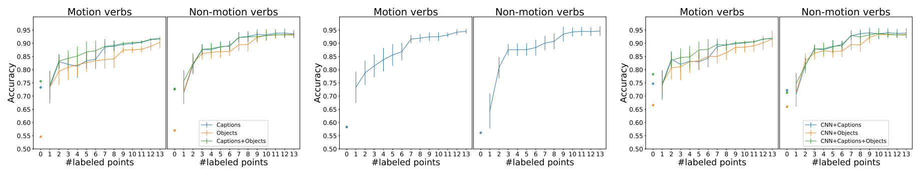
$$x_{i,h} = \begin{cases} 1 & \text{if } i \text{ is labeled and have sense } h \\ \frac{1}{|S_i|} & \text{if } i \text{ is unlabeled and } h \in S_i \\ 0 & \text{otherwise} \end{cases}$$

where $x_{i,h}$ correspond to the probability that the i-th player chooses strategy h while S_i is the set of possible senses associated to the verb in i. As one can note, the matrix x is stochastic by construction (each row adds up to one).

Then GTG iteratively refines the assignment x considering the similarities between players. It uses a dynamical system from Evolutionary Game Theory, named Replicator Dynamics (RD) [107]. The RD iterates until the initial assignment stabilizes or a

[3]Since the features are all non-negative with unit norm, the cosine can be computed using the dot product.

certain amount of iterations is reached[4]. This stable condition is known as Nash Equilibria [117] and it corresponds to a consistent labeling of the data [111]. At equilibrium every player has chosen a strategy representing their best option, considering also the choices of the others, hence they get labeled.



(a) text (b) cnn (c) cnn+text

Figure 8.2: GOLD results for text data, cnn and cnn+text varying the number of labeled points in comparison with Gella et al. approach (circles).

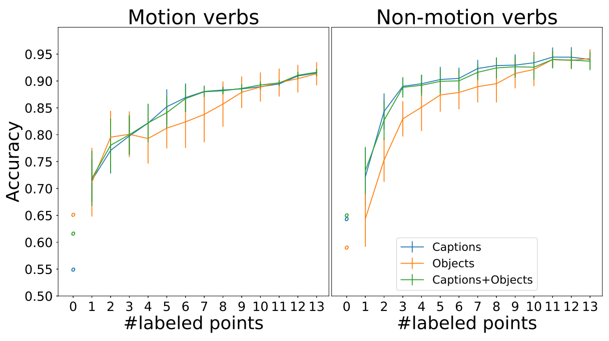
8.4 Experiments

Experiments have been performed on VerSe dataset. VerSe is composed of 3488 images selected from COCO and TUHOI, 90 verbs and 163 possible senses, resulting in 3510 (image, verb) pairs. Each pair represents a player in our non-cooperative game (see Sec. 8.3.2). Accordingly to [51] the performances have been quantified using the accuracy of the predicted sense. Considering the different features and their combinations there are 7 possible setups for the experiments: captions (C), object labels (O), captions with object labels (C+O), CNN features (VIS), CNN features concatenated to captions (CNN+C), CNN features concatenated to object labels (CNN+O) and CNN features concatenated to captions with object labels (CNN+O+C). The experiments with our method have been carried out considering an increasing number of labeled elements per sense (from 1 to 13) and accounting for the accuracy on the unlabeled elements. The GTG process is carried out by randomly sampling elements considered as labeled for each possible class (verb sense). In order to have results that are consistent and not influenced by the stochasticity of initial sampling, each setup has been executed on a pool of 15 different seeds.

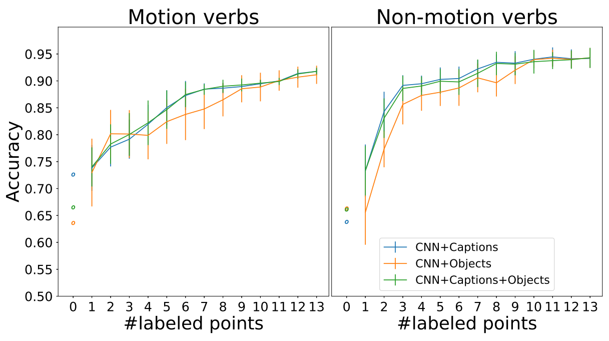
8.5 Results

The results are shown in Figures 8.2 and 8.3 in which means and standard deviations for all of the 15 runs are reported alongside the results of [51]. Verb types have been split using Levin classes (motion and non-motion verbs). The performances of GTG are plotted varying the amount of labeled information from 1 to 13. As can be seen, the performances with 1 labeled point per sense are comparable or better than [51] while with 2 or more labeled points we outperform dramatically the state-of-the-art.

[4]We set the maximum number of iterations to 10 as suggested in [33]



(a) text



(b) cnn+text

Figure 8.3: PRED results for text data, cnn and cnn+text varying the number of labeled points in comparison with Gella et al. approach (circles).

Verb type	Textual		
	O	C	O+C
Motion	73.63	76.19	76.62
Non Motion	83.94	80.58	81.98

Verb type	CNN	CNN+		
		O	C	O+C
Motion	78.83	74.68	77.44	77.82
Non-Motion	80.80	83.10	80.32	81.77

Table 8.1: Results considering modern DNN

In GOLD setting (Fig.8.2), for object labels (O) and visual features (CNN), the GTG approach outperforms Gella et al. results with just 1 labeled point. Whereas, for caption data they are quite aligned; especially if we take into account standard deviations. This has an impact on all derived experiments that are based on image captions. In PRED annotations setting (Fig.8.3), all GTG experiments outperform Gella et al. results when two labeled points are considered. When only 1 labeled point is used, the performances of Gella et. al. are in the range of variances of GTG. In general, we note that the higher the number of labeled point per sense, the higher the overall accuracy and smaller the standard deviation. The accuracy follows a logarithmic growth i.e. the variation of the number of labels has a relevant role when they are few, whereas, with more than 6-7 labeled points per class, the accuracy starts converging.

Performances with modern DNN In this experiment we substitute the image classifier used in the PRED setting with a state-of-the-art object detector [130] which provides us the actual content of the image rather than an estimate of the possible classes of an image. Regarding the captioning, we adopt a more recent automatic captioning method [105].

Faster R-CNN might generate non-unique object labels, for this reason we dropped duplicates. The results of this setting (see Table 8.1) are aligned to the ones obtained with standard PRED.

8.6 Conclusions & Future Work

In this chapter, we showed the suitability of a game-theoretic model in the task of multimodal verb sense disambiguation. The power of the proposed approach relies on the transductive nature of the GTG which performs the inference for the sense of a verb based not only on the similarity with the labeled samples but considering also the unlabeled ones, leading to a more precise boundaries definition thus outperforming the state-of-the-art in the VSD task. As for the future works, we will investigate the use of prior knowledge on the possible verb senses in the system.

Chapter 9

Ancient Coin Classification

Reference[1]: S. Aslan, ***S. Vascon***, and M. Pelillo. Ancient coin classification using graph transduction games. *In Proc. of the IEEE 4th International Conference on Metrology for Archaeology and Cultural Heritage*, IEEE, 2018. **Oral presentation** . Pre-print available at `https://arxiv.org/abs/1810.01091`

The content of this chapter is taken from the above reference.

Recognizing the type of an ancient coin requires theoretical expertise and years of experience in the field of numismatics. Our goal in this work is automatizing this time-consuming and demanding task by a visual classification framework. Specifically, we propose to model ancient coin image classification using Graph Transduction Games (GTG). GTG casts the classification problem as a non-cooperative game where the players (the coin images) decide their strategies (class labels) according to the choices made by the others, which results with a global consensus at the final labeling. Experiments are conducted on the only publicly available dataset which is composed of 180 images of 60 types of Roman coins. We demonstrate that our approach outperforms the literature work on the same dataset with the classification accuracy of 73.6% and 87.3% when there are one and two images per class in the training set, respectively.

9.1 Introduction

Ancient coins, that depict cultural, political and military events, natural phenomena, ideologies and portraits of god and emperors are important source of information for historians and archaeologists. Recognizing the type of an ancient coin requires theoretical expertise and years of experience in the field of numismatics. A common way to detect the period of a discovered coin is searching through the manual books where ancient coins are indexed [24] which requires a highly time consuming labor. Our goal in this chapter is automatizing recognition of Roman coins by employing computer vision

[1]Presented by S. Aslan on October, 23rd 2018 in oral session of "Artificial Intelligence for Measurements in Cultural Heritage" helds in conjunction with MetroArcheo 2018

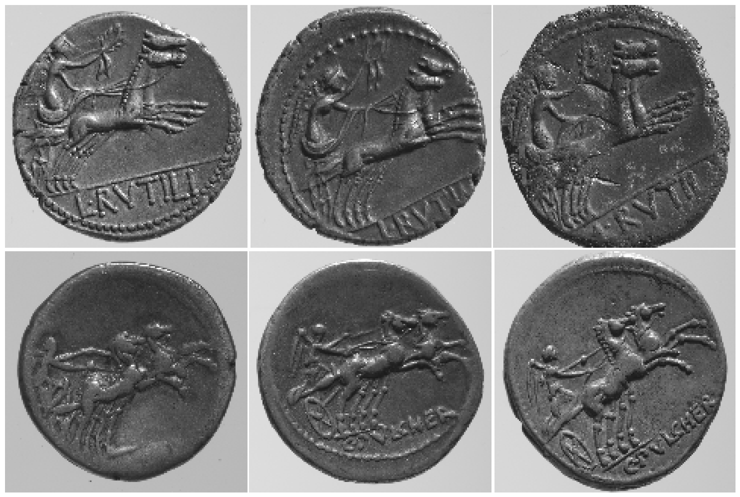


Figure 9.1: Example images of two classes from the Roman coin dataset [163] that is used in this work. *First row:* Images of class 387/1; *Second row:* Images of class 300/1 (listed with Crawford [24] reference number).

and pattern recognition techniques. Automatizing such a manual procedure not only provides faster processing time but also can support historians and archaeologists for a more accurate decision. A visual classification framework for ancient coin recognition can also be used at museums or by individual collectors to organize large collections of coins. From the computer vision point of view, classification of ancient coin images is a highly challenging task. One of the difficulties arises from existence of high number of types (i.e. classes) in ancient coins (e.g. Portuguese coins from medieval period and coins from Roman Republic compose over 1500 [135] and 550 [24] different classes, respectively), while most of the classes include few known specimens as mentioned in [135, 172]. Moreover, intra-class variation is large due to local spatial variations arising from missing parts and degradations on the coins, and manual manufacturing of coins by different engravers. Another reason of large intra-class variation is the metallic structure of these coins yields to strong reflection and shading variations so the appearance of the same coin changes significantly under different lighting conditions. Another challenge in ancient coin classification is the typical low inter-class variations due to high global similarity between classes [171]. Images from two coin classes are presented in Fig. 9.1 to demonstrate the challenges of large intra-class and low inter-class variations.

Ancient coin classification can be accomplished by adopting one of the following approaches for classifiers [13]: (i) *learning-based classifiers*, where the parameters of the classifier (e.g. Deep Neural Networks, SVM, Random Forests, etc.) are learned from data in an intensive training phase. (ii) *non-parametric classifiers*, where the classification decision is directly based on data without pursuing any training phase (e.g. Nearest Neighbor based classifier). Although the first group proved to be superior to the second one, they require extraction of highly discriminative features (possibly from abundant training data) for robust classification. Moreover, pursuing such a time consuming training phase can be impractical for handling dynamic databases where new classes are included continuously.

In this chapter, we adopt a non-parametric classifier for ancient coin classification, which is preferable under existence of aforementioned challenges, i.e. large intra-class and low inter-class variation and lack of abundant training data. We have followed the same approach in [163, 171], i.e. our non-parametric classifier uses a dissimilarity measurement derived from costs of dense matching of SIFT features. Similar to [163, 171], for dense feature matching we use SIFT flow [98], a flow estimation technique developed for image alignment. SIFT flow preserves discontinuity so allows matching objects that locate at different parts of image. This property of SIFT flow makes it well suited for coin images [171], i.e. it helps to deal with large intra-class variation since images from the same class has similar spatial arrangement of features. Additionally, defining similarity between two coin images based on local matches between them helps to deal with low inter-class variation, since two classes mostly differ from each other in variations at local regions.

Differently from [163, 171], in this work we do not use a greedy Nearest Neighbor (*NN*) based classifier where a query image is labeled with the class of its nearest (most similar) image in the dataset. Instead, we use a semi-supervised learning approach, namely *Graph Transduction Game* (*GTG*) [35], for ancient coin classification. The GTG casts the classification problem as a non-cooperative game where the players (the coin images) decide their strategies (class labels) according to the choices made by the others, which results with a global consensus at the final labeling. Experimenting on a small-scale ancient coin dataset having the aforementioned classification challenges, we show that the notion of label consistency [67] provided by GTG brings significant performance gain over the conventional NN-based classifier for this challenging problem.

9.2 Previous works

One of the main problems of ancient coin image analysis that is addressed in the literature is *coin identification* where the goal is recognizing a specific coin instance instead of a coin type [64, 75]. This type of application finds usage at identification of stolen coins. Most of the other works have focused on *coin type recognition* (or *coin classification*) which has found a wider range of practical usage. A number of works [76,163,171,172] employed NN-based classifier where the class of a query coin image is assigned with the label of its most similar one in the training set. Among these, [76] defined coin similarity by number of matched SIFT features that were detected sparsely on the images, while [163,171] employed dense matching costs of SIFT flow as dissimilarity metric. In [172], the authors used densely computed illumination-invariant LIDRIC features and fusing several similarity scores that point out the matching quality they employed an overall similarity score. High performance results are reported in these works although the employed datasets were quite small-scale, i.e. classification accuracies of 90% [76] and 82% [163] are obtained for the datasets with 390 images of 3 classes and 180 images of 60 classes, respectively.

Other works employed learning-based classifiers. Earlier attempts [3, 4] relied on Bag of Visual Words based representation of local image features where a visual dictionary is learned from a training set and classification is achieved with SVM in [3] and GMM in [4]. Recently, Schlag and Arandjelovic proposed to use a deep convolutional neural network for Roman coin classification in [136]. They accomplish training with a large set of images, i.e. around 20K images of 83 classes, and reported around 83% accuracy on 10k images.

A significant obstacle at employing learning-based classifiers for this particular research problem is deficiency of publicly available datasets. A number of works employed datasets of Sassanian dynasty coins [121], some others focused on medieval coins [135], and most of them have worked on coins of the Roman Republic [3, 4, 136, 163, 171, 172]. However, the only publicly available ancient coin dataset is published by Zambanini and Kampel which is composed of 180 images of 60 Roman coin classes [163] which we experimented on in this work.

9.3 Ancient Coin Classification using Graph Transduction Game

By considering the training set of coin images as the labelled players, GTG can be applied for ancient coin classification problem to estimate the labels of the test set images, i.e. unlabelled players. We list the steps that we have employed for the application of GTG for ancient coin classification as follows:

Feature extraction We compute two type of features on the images: (i) In order to analyze local similarities, we compute 128-dimensional SIFT features in the local neighborhood of every image pixel that results with a tensor named as *SIFT-image* [98]; (ii) In order to analyze global similarities between images we compute CNN features. Specifically, since our dataset is quite small, which makes a CNN training unfeasible, we apply transfer learning by using a CNN architecture pre-trained on ImageNet. Finally, for each input image we get its feature from the output of the last fully-connected layer of the CNN.

Initialization of the strategy space Since no other knowledge on the problem exists, but only the distinction between labeled and unlabeled sets, the strategy space is initialized using Eq.7.3 and Eq. 7.4.

Computation of similarity between objects A correct choice of computation for the similarity between images is important to avoid a failure at label estimation. We employ different schemes of similarity computation regarding to the extracted feature types:

i. Similarity between local features of images: It is demonstrated in [171] that matching scores of SIFT flow technique are powerful dissimilarity metric for ancient coin classification. In SIFT flow, SIFT-images are matched along the flow vectors and optimal correspondences are found by minimizing an energy function ($E(w)$ in

[98]) using dual-layer belief propagation [98]. Since runtime of such optimization scales up with the image size, authors of [98] proposed to employ coarse-to-fine search which results with faster computation and better performance of matching. Similar to [163], in this work we used the minimum energy value, say $E^*_{i,j}$, (to which SIFT Flow algorithm converges at the finest level of the coarse-to-fine search) as a dissimilarity metric between image i and j, i.e. we used $d(f_i, f_j) = E^*_{i,j}$ in Eq. 2.1.

ii. Similarity between global descriptions of images: Following the general trend [34, 35], we used Euclidean distance, i.e. $d(f_i, f_j) = \|f_i - f_j\|_2$ in Eq. 2.1, to compute similarity between the CNN features.

Execution of transduction game Giving the similarities to the GTG, it starts to play the game between players, i.e. images, until convergence. We get the final probabilities of strategies, i.e. labels, for the unlabeled objects at the output and we assign the object with the strategy that could get the highest maximum probability.

9.4 Experiments

Dataset We experimented on the only published[2] ancient coin dataset [163] which is acquired at Coin Cabinet of the Museum of Fine Arts in Vienna, Austria. The dataset is composed of 180 images (reverse sides of the coins that includes motifs and legends) of 60 classes with 3 images in each class. Images are resized to 150×150 pixels as in [163].

Experimental setup Since we have experimented on the same dataset, we followed the same experimental setting with [163] to make a fair comparison of techniques. In [163], accepting one of the coins as a query image (or test image), the remaining one or two images per class are used to create the training set. At each classification run, nearest neighbor of the query image is searched in the training set. This procedure leads to 180 and 360 classification runs when two and one training images per class is used. When the training set is created by two images per class, the nearest neighbor search is performed through accumulated dissimilarity values of each training set image over classes.

Adopting the same experimental setting in our approach, we create a dissimilarity matrix with the entries computed as in [163], i.e. as mentioned in Section IV.c. Then we symmetrize it (by getting maximum of entries around diagonal) before giving input to the GTG algorithm. Additionally, at each iteration we substitute the test image and training images as unlabeled object and labeled objects, respectively to be used in GTG and we get the class label of the unlabeled object in the output. In all experiments, the parameter k of the neighboring set $\mathcal{N}_i$ in Eq. 3.11 is set to 2.

Performance evaluation We performed GTG by employing two feature types and with the corresponding dissimilarity metrics as explained in Section IV. In the first experiment, we compute off-the-shelf CNN features by DenseNet-201 which is one of the

[2]`http://cvl.tuwien.ac.at/research/cvl-databases/coin-image-dataset/`

Table 9.1: Classification results

Technique	Training set: 1 image per class		Training set: 2 images per class	
	Correct classifications	Classification accuracy	Correct classifications	Classification accuracy
CNN features + Euclidean distance + GTG	188 / 360	52.2%	113 / 180	62.8%
Dense SIFT + Matching cost + NN [163]	257 / 360	71.4%	150 / 180	83.3%
Dense SIFT + Matching cost + GTG	**265 / 360**	**73.6%**	**157 / 180**	**87.2%**

state-of-the-art CNN architectures where we use the Euclidean distance metric to measure the dissimilarity between the features. In the second experiment, by employing densely computed local SIFT features we use matching costs of SIFT flow as dissimilarity measure. The performance results of these experiments and comparison with the state-of-the-art work on the same dataset [163] are given in Table 9.1.

It can be seen in Table 9.1 that the lowest performance results for both training set sizes are obtained when we use the CNN features. This is an expected outcome, because CNN features provide a global description of images and a high global similarity exists between different classes in this coin dataset. We could outperform [163] that employs a NN-based classifier, by using the GTG for ancient coin classification by 73.6% and 87.2% classification accuracy when the training set is constructed from one and two images per class, respectively. We additionally checked the performance of conventional NN-based classifier which does not adopt the accumulation of class-wise dissimilarities (that were adopted at [163]), when there are two images per class in the training set. In that case, we got 81.67% accuracy which was slightly lower than the reported performance (83.3%) in [163].

In Fig. 9.2, we present two misclassifications of the proposed approach. It can be seen that the misclassifications are mostly due to low variability between different classes.

9.5 Conclusion

In this chapter, we studied the ancient coin classification problem using Graph Transduction Games (GTG) which adopts the approach of non-parametric classifier. The GTG is a game-theoretic semi-supervised learning algorithm, grounded on the notion of label consistency, in which the final labeling of the objects is achieved by reaching an equilibrium condition between all labeling hypothesis. Our experimental results show that GTG works better for the problem of ancient coin classification, which is a highly complex problem due to large intra-class and low inter-class variations, compared to conventional nearest neighbor based non-parametric classifiers that does not consider global agreement at labeling choices of all dataset images.

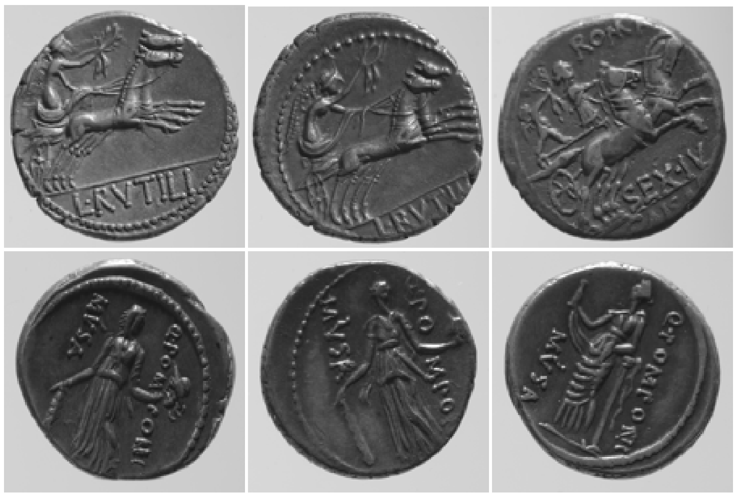


Figure 9.2: Two selected misclassifications of the proposed approach based on GTG. *First column:* test image; *Second column:* another image from the same class; *Third column:* image of selected class by the proposed scheme.

Acknowledgment

The authors would like to thank Sebastian Zambanini, Ismail Elezi, Leulseged Tesfaye Alemu and Alessandro Torcinovich for their invaluable advices, sharing and helps at various technical issues.

Conclusions

In this thesis, we showed how non-cooperative game theory can be used effectively in different fields of computer science. In particular, we have explored an algorithm, rooted in the evolutionary game-theory, named *Graph Transduction Games*. The GTG models a semi-supervised learning problem as a non-cooperative game where the classification hypothesis emerges as the results of the interaction between data points. In such a game, the players are the objects of a dataset, and the labels correspond to the strategies. The aim of the game is letting players play together and pick their best strategy, considering the choices made by others. The interactions between players are modeled through a weighted graph, which defines who and how plays together. The graph explicitly creates a *context* in which the players are playing. In this thesis, the GTG algorithm and the notion of context have been used in different fields, outperforming traditional SSL algorithm and performing on par to deep-learning approaches. Summarizing, from a *methodological* point of view we contributed in different directions:

Unsupervised Domain Adaptation In this work, the context played a central role defining the relations between the data from the source and the target domain. Once the context is created, the labels from the source domain are propagated to the target domain in a consistent way, considering both prior knowledge and similarities. This contributes to offering a newer perspective to UDA, which outperforms more complex and tailored UDA methods while achieving comparable performances with deep-learning approaches.

Clustering using Factorization of Non-Negative Matrices Here we exploit the context and the GTG algorithm to refine the clustering obtained by non-negative matrix factorization algorithms. The idea is to use the *clustering assumption*[3] creating a context composed of similar data points (a sparsely connected graph), letting the assignment between points and clusters to adjust automatically, considering the contextual relations. We initialize a non-cooperative game using both the prior partitioning from a factorization method and the similarity between points, then we let the final assignment to emerge by competition. This is a substantial contribution since NMF clustering methods are widespread and can be easily ameliorated with this game-theoretic approach.

[3]Similar objects are assigned to similar cluster.

Deep Neural Network Training under Data Scarcity In this work we have proposed a new schema to make feasible the train of a neural network under scarcity of labeled data but with abundant unlabeled ones. Here, the GTG algorithm was able to propagate information to an unlabeled pool of data in an effective way using a few annotated points. The ”pseudo-labels” created with GTG are more precise than the one obtained using standard inductive methods, showing better performance during network training.

From an *applicative* point of view, we applied GTG and its notion of the context within different areas, offering a newer perspective and remarkable performances.

Protein Function Prediction Predicting protein functionalities is a really complex task due to the nature of the data and to the fact that multiple functions can be associated with a particular protein. Following the Hume assumption, which states that *similar points belong to similar classes* we were able to encode within the GTG framework not only pairwise similarities between proteins but also similarities between functionalities which mean exploiting the semantics of the classes. The classification task as an emerging property of the data, and the similarities between classes lead us to achieve outstanding performance in such a complex task.

Multimodal Verb Sense Disambiguation and Ancient Coin Classification The prior works in these two fields considered the classification like an action performed on each object independently and in isolation. Due to the complexity of the task, which can lead to an ambiguous assignment, the usage of contextual classification and the idea of labeling as an emergent property shown to be effective.

Summarizing what emerged from this research thesis is the following: *i)* classification as an emergent properties from data proved to be (methodologically) a way to deepen in the next year *ii)* The role of context defined in terms of interaction between data points helps in solving ambiguous conditions *iii)* GTG is a very powerful and flexible algorithm being able to accommodate in a principle way:

1 prior knowledge on the data in terms of classes assignment

2 no assumption on the functions used to compute the similarities

3 allows the usage of similarities between classes

4 the dynamical system used to reach a Nash Equilibria is highly parallelizable

The direction for the next works will be toward the usage of GTG directly into the network training as an additional end-to-end *disambiguation* layer and the usage of alternative sparsification methods.

Bibliography

[1] Geoffrey Hinton Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[2] N. S. Altman. An introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, 1992.

[3] Hafeez Anwar, Sebastian Zambanini, and Martin Kampel. Coarse-grained ancient coin classification using image-based reverse side motif recognition. *Machine Vision and Applications*, 26(2-3):295–304, 2015.

[4] Ognjen Arandjelovic. Automatic attribution of ancient roman imperial coins. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1728–1734, 2010.

[5] Sinem Aslan, Sebastiano Vascon, and Marcello Pelillo. Ancient coin classification using graph transduction games. In *The 4th IEEE International Conference on Metrology and Archeology*, 2018.

[6] Terri K. Attwood, Paul Bradley, Darren R. Flower, Anna Gaulton, Neil Maudling, AL Mitchell, G Moulton, A Nordle, K Paine, P Taylor, et al. Prints and its automatic supplement, preprints. *Nucleic acids research*, 31(1):400–402, 2003.

[7] Kobus Barnard and Matthew Johnson. Word sense disambiguation with pictures. *Artificial Intelligence*, 167(1-2):13–30, 2005.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[9] Itai Benjamini, Van Cyr, Eviatar B. Procaccia, and Ran J. Tessler. Harmonic labeling of graphs. *Discrete Mathematics*, 313(17):1726 – 1745, 2013.

[10] Abraham Berman and Robert J Plemmons. Nonnegative matrices. *The Mathematical Sciences, Classics in Applied Mathematics*, 9, 1979.

[11] A. Bertoni, M. Frasca, and G. Valentini. COSNet: a cost sensitive neural network for semi-supervised learning in graphs. In *ECML*, pages 219–234. Springer, 2011.

[12] Yevgeni Berzak, Andrei Barbu, Daniel Harari, Boris Katz, and Shimon Ullman. Do you see what i mean? visual resolution of linguistic ambiguities. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1477–1487, Lisbon, Portugal, 2015. Association for Computational Linguistics.

[13] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[14] Teresa Botschen, Iryna Gurevych, Jan-Christoph Klie, Hatem Mousselly Sergieh, and Stefan Roth. Multimodal frame identification with multilingual evaluation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1481–1491, 2018.

[15] Christos Boutsidis and Efstratios Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[16] Juan C Caicedo, Jaafar BenAbdallah, Fabio A González, and Olfa Nasraoui. Multimodal representation, indexing, automated annotation and retrieval of image collections via non-negative matrix factorization. *Neurocomputing*, 76(1):50–60, 2012.

[17] H. Caniza, A. Romero, S. Heron, H. Yang, A. Devoto, M. Frasca, M. Mesiti, G. Valentini, and A. Paccanaro. GOssTo: a user-friendly stand-alone and web tool for calculating semantic similarities on the Gene Ontology. *Bioinformatics*, 30(15), 2014.

[18] N. Cesa-Bianchi, M. Re, and G. Valentini. Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning*, 88(1):209–241, 2012.

[19] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, Cambridge, MA, 1st edition, 2006.

[20] H.N. Chua, W. Sung, and L. Wong. An efficient strategy for extensive integration of diverse biological data for protein function prediction. *Bioinformatics*, 23(24):3364–3373, 2007.

[21] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649, 2012.

[22] The UniProt Consortium. Uniprot: a hub for protein information. *Nucleic Acids Research*, 43(D1):D204–D212, 2015.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[24] Michael H Crawford. *Roman republican coinage*, volume 2. Cambridge University Press, 1974.

[25] Manuel Curado, Francisco Escolano, Miguel Angel Lozano, and Edwin Robert Hancock. net4lap: Neural laplacian regularization for ranking and re-ranking. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1366–1371. IEEE, 2018.

[26] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[28] M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins. *J. Comput. Biol.*, 11:463–475, 2004.

[29] Antonio Di Marco and Roberto Navigli. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, 39(3):709–754, 2013.

[30] Chris Ding, Tao Li, and Wei Peng. Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 342. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[31] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2001.

[32] D. A. Easley and J. M. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press, 2010.

[33] I. Elezi, A. Torcinovich, S. Vascon, and M. Pelillo. Transductive label augmentation for improved deep network learning. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1432–1437, Aug 2018.

[34] Ismail Elezi, Alessandro Torcinovich, Sebastiano Vascon, and Marcello Pelillo. Transductive label augmentation for improved deep network learning. In *24th International Conference on Pattern Recognition, ICPR 2018, Beijing, China, August 20-24, 2018*, pages 1432–1437, 2018.

[35] Aykut Erdem and Marcello Pelillo. Graph transduction as a noncooperative game. *Neural Computation*, 24(3):700–723, 2012.

[36] Francisco Escolano, Manuel Curado, Silvia Biasotti, and Edwin R Hancock. Shape simplification through graph sparsification. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 13–22. Springer, 2017.

[37] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[38] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 2960–2967, Washington, DC, USA, 2013. IEEE Computer Society.

[39] Robert D Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, et al. Pfam: clans, web tools and services. *Nucleic acids research*, 34(suppl 1):D247–D251, 2006.

[40] M. Frasca, A. Bertoni, M. Re, and G. Valentini. A neural network algorithm for semi-supervised node label learning from unbalanced data. *Neural Networks*, 43:84–98, 2013.

[41] M. Frasca, A. Bertoni, and G. Valentini. Unipred: Unbalance-aware network integration and prediction of protein functions. *J. Comput. Biol.*, 22(12):1057–1074, 2015.

[42] Marco Frasca. Automated gene function prediction through gene multifunctionality in biological networks. *Neurocomputing*, 162:48 – 56, 2015.

[43] Marco Frasca and Giulio Pavesi. A neural network based algorithm for gene expression prediction from chromatin structure. In *IJCNN*, pages 1–8. IEEE, 2013.

[44] Marco Frasca and Giorgio Valentini. Cosnet: An r package for label prediction in unbalanced biological networks. *Neurocomputing*, 237:397 – 400, 2017.

[45] I. Friedberg. Automated protein function prediction-the genomic challenge. *Brief. Bioinformatics*, 7:225–242, 2006.

[46] Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982.

[47] Wai Shing Fung, Ramesh Hariharan, Nicholas J.A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 71–80, New York, NY, USA, 2011. ACM.

[48] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 148–155. Morgan Kaufmann Publishers Inc., 1998.

[49] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[50] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[51] Spandana Gella, Frank Keller, and Mirella Lapata. Disambiguating visual verbs. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):311–322, 2019.

[52] Spandana Gella, Mirella Lapata, and Frank Keller. Unsupervised visual sense disambiguation for verbs using multimodal embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–192. Association for Computational Linguistics, 2016.

[53] Gene Ontology Consortium. Gene Ontology annotations and resources. *Nucleic Acids Research*, 41:D530–535, 2013.

[54] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE, 2012.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[56] Julian Gough, Kevin Karplus, Richard Hughey, and Cyrus Chothia. Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *Journal of molecular biology*, 313(4):903–919, 2001.

[57] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[58] Philip Häusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2784–2792, 2017.

[59] Philip Häusser, Alexander Mordvintsev, and Daniel Cremers. Learning by association - A versatile semi-supervised training method for neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 626–635, 2017.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[61] J.J. Hopfield. Neural networks and physical systems with emergent collective compatational abilities. *Proc. Natl Acad. Sci. USA*, 79:2554–2558, 1982.

[62] Joseph T Howson Jr. Equilibria of polymatrix games. *Management Science*, 18(5-part-1):312–318, 1972.

[63] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[64] Reinhold Huber-Mörk, Sebastian Zambanini, Maia Zaharieva, and Martin Kampel. Identification of ancient coins based on fusion of shape and local features. *Machine vision and applications*, 22(6):983–994, 2011.

[65] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Marco Pagni, and Christian JA Sigrist. The prosite database. *Nucleic acids research*, 34(suppl 1):D227–D230, 2006.

[66] David Hume. *An enquiry concerning human understanding: A critical edition*, volume 3. Oxford University Press, 2000.

[67] Robert A Hummel and Steven W Zucker. On the foundations of relaxation labeling processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (3):267–287, 1983.

[68] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning (ICML)*, volume 2, page 3, 2013.

[69] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5070–5079, 2019.

[70] Y. Jiang et al. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(184), 2016.

[71] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*.

[72] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 290–297, 2003.

[73] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[74] Ilan Kadar and Ohad Ben-Shahar. Scenenet: A perceptual ontology for scene understanding. In *European Conference on Computer Vision (ECCV)*, pages 385–400. Springer, 2014.

[75] Martin Kampel, Reinhold Huber-Mörk, and Maia Zaharieva. Image-based retrieval and identification of ancient coins. *IEEE Intelligent Systems*, 24(2):26–34, 2009.

[76] Martin Kampel and Maia Zaharieva. Recognizing ancient coins based on local features. In *International Symposium on Visual Computing*, pages 11–22, 2008.

[77] U. Karaoz et al. Whole-genome annotation by using evidence integration in functional-linkage networks. *Proc. Natl Acad. Sci. USA*, 101:2888–2893, 2004.

[78] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[79] Jingu Kim, Yunlong He, and Haesun Park. Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. *Journal of Global Optimization*, 58(2):285–319, 2014.

[80] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.

[81] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3581–3589, 2014.

[82] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[83] Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

[84] S. Kohler, S. Bauer, D. Horn, and P.N. Robinson. Walking the interactome for prioritization of candidate disease genes. *Am. J. Human Genetics*, 82(4):948–958, 2008.

[85] Wouter M Kouw, Laurens JP van der Maaten, Jesse H Krijthe, and Marco Loog. Feature-level domain adaptation. *Journal of Machine Learning Research*, 17(171):1–32, 2016.

[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[87] Da Kuang, Sangwoon Yun, and Haesun Park. Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization*, 62(3):545–574, 2015.

[88] L. Lan, N. Djuric, Y. Guo, and Vucetic S. MS-kNN: protein function prediction by integrating multiple data sources. *BMC Bioinformatics*, 14(Suppl 3:S8), 2013.

[89] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[90] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[91] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[92] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[93] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[94] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, 2013.

[95] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, 1986.

[96] Ivica Letunic, Richard R Copley, Birgit Pils, Stefan Pinkert, Jörg Schultz, and Peer Bork. Smart 5: domains in the context of genomes and networks. *Nucleic acids research*, 34(suppl 1):D257–D260, 2006.

[97] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[98] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011.

[99] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 97–105, 2015.

[100] Mingsheng Long, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*, 2016.

[101] Mingsheng Long, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. *arXiv preprint arXiv:1602.04433*, 2016.

[102] L. Lovász. Random walks on graphs: A survey. In D. Miklós, V. T. Sós, and T. Szőnyi, editors, *Combinatorics, Paul Erdős is Eighty*, volume 2, pages 353–398. János Bolyai Mathematical Society, Budapest, 1996.

[103] L Lovasz and MD Plummer. Matching theory north-holland mathematics studies, 121. *Annals of Discrete Mathematics*, 29, 1986.

[104] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[105] Ruotian Luo, Brian Price, Scott Cohen, and Gregory Shakhnarovich. Discriminability objective for training descriptive captions. pages 6964–6974, 2018.

[106] M.L. Mayer and P. Hieter. Protein networks - guilt by association. *Nature Biotechnology*, 18(12):1242–1243, 2000.

[107] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[108] J Maynard Smith and George R Price. The logic of animal conflict. *Nature*, 246(5427):15–18, 1973.

[109] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, 2016.

[110] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Conference Track Proceedings*, 2013.

[111] Douglas A Miller and Steven W Zucker. Copositive-plus lemke algorithm solves polymatrix games. *Operations Research Letters*, 10(5):285–290, 1991.

[112] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[113] A. Mitrofanova, V. Pavlovic, and B. Mishra. Prediction of protein functions with gene ontology and interspecies protein homology data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):775–784, 2011.

[114] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, 9(S4), 2008.

[115] Nicola J Mulder, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Virginie Buillard, Lorenzo Cerutti, Richard Copley, et al. New developments in the interpro database. *Nucleic acids research*, 35(suppl 1):D224–D228, 2007.

[116] Jean Muller, Damian Szklarczyk, Philippe Julien, Ivica Letunic, Alexander Roth, Michael Kuhn, Sean Powell, Christian von Mering, Tobias Doerks, Lars Juhl Jensen, et al. eggnog v2. 0: extending the evolutionary genealogy of genes with enhanced non-supervised orthologous groups, species and functional annotations. *Nucleic acids research*, 38(suppl 1):D190–D195, 2010.

[117] John Nash. Non-cooperative games. *Annals of Mathematics*, pages 286–295, 1951.

[118] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics, 2010.

[119] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

[120] Stephen Oliver. Proteomics: guilt-by-association goes global. *Nature*, 403(6770):601, 2000.

[121] Seyyedeh-Sahar Parsa, Mohamad Sourizaei, Mohammad Mahdi Dehshibi, Reza Esmaeilzadeh Shateri, and Mohammad Reza Parsaei. Coarse-grained correspondence-based ancient sasanian coin classification by fusion of local features and sparse representation-based classifier. *Multimedia Tools and Applications*, 76(14):15535–15560, 2017.

[122] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):167–172, 2007.

[123] Marcello Pelillo. The dynamics of nonlinear relaxation labeling processes. *Journal of Mathematical Imaging and Vision*, 7(4):309–323, 1997.

[124] Sameer S Pradhan and Nianwen Xue. Ontonotes: The 90% solution. In *HLT-NAACL (Tutorial Abstracts)*, pages 11–12, 2009.

[125] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 413–420, 2009.

[126] P. Radivojac et al. A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10(3):221–227, 2013.

[127] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015.

[128] M. Re, M. Mesiti, and G. Valentini. A Fast Ranking Algorithm for Predicting Gene Functions in Biomolecular Networks. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 9(6):1812–1818, 2012.

[129] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[130] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[131] S. Rota Buló, M. Pelillo, and I. M. Bomze. Graph-based quadratic optimization: A fast evolutionary approach. *Computer Vision and Image Understanding*, 115(7):984–995, 2011.

[132] Kate Saenko and Trevor Darrell. Unsupervised learning of visual sense models for polysemous words. In *Advances in Neural Information Processing Systems*, pages 1393–1400, 2009.

[133] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[134] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10:e0118432, 2015.

[135] Luís Salgado. *Medieval coin automatic recognition by computer vision*. PhD thesis, 2016.

[136] Imanol Schlag and Ognjen Arandjelovic. Ancient roman coin recognition in the wild using deep learning based recognition of artistically depicted face profiles. In *IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 2898–2906, 2017.

[137] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[138] B. Schwikowski, P. Uetz, and S. Fields. A network of protein-protein interactions in yeast. *Nature biotechnology*, 18(12):1257–1261, December 2000.

[139] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016.

[140] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Mol. Sys. Biol.*, 8(88), 2007.

[141] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[142] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[143] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[144] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[145] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[146] Damian Szklarczyk et al. String v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic Acids Research*, 43(D1):D447–D452, 2015.

[147] R. Tripodi and M. Pelillo. Document Clustering Games in Static and Dynamic Scenarios. *ArXiv e-prints*, July 2016.

[148] R. Tripodi and M. Pelillo. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, in press.

[149] R. Tripodi, S. Vascon, and M. Pelillo. Context aware nonnegative matrix factorization clustering. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1719–1724, Dec 2016.

[150] Rocco Tripodi and Marcello Pelillo. A game-theoretic approach to word sense disambiguation. *Comput. Linguist.*, 43(1):31–70, April 2017.

[151] Rocco Tripodi, Sebastiano Vascon, and Marcello Pelillo. Context aware nonnegative matrix factorization clustering. In *International Conference on Pattern Recognition, (ICPR)*, pages 1719–1724, 2016.

[152] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[153] G. Valentini, G. Armano, M. Frasca, J. Lin, M. Mesiti, and M. Re. RANKS: a flexible tool for node label ranking and classification in biological networks. *Bioinformatics*, 32:2872–2874, 2016.

[154] G. Valentini, A. Paccanaro, H. Caniza, A. Romero, and M. Re. An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods. *Artificial Intelligence in Medicine*, 61(2):63–78, 2014.

[155] Vladimir N Vapnik. *The nature of statistical learning theory*. Springer, 1995.

[156] V.N. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[157] S. Vascon, M. Frasca, R. Tripodi, G. Valentini, and M. Pelillo. Protein function prediction as a graph-transduction game. *Pattern Recognition Letters*, 2018 (in press).

[158] Sebastiano Vascon, Sinem Aslan, Alessandro Torcinovich, Twan van Laarhoven, Elena Marchiori, and Marcello Pelillo. Unsupervised domain adaptation using graph transduction games, 2019.

[159] A. Vazquez, A. Flammini, A Maritan, and A Vespignani. Global protein function prediction from protein-protein interaction networks. *Nature Biotechnology*, 21:697–700, 2003.

[160] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[161] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007.

[162] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton University Press, 1944.

[163] Sebastian Zambanini and Martin Kampel. Coarse-to-fine correspondence search for classifying ancient coins. In *Asian Conference on Computer Vision*, pages 25–36, 2012.

[164] Yuan Wang, Yunde Jia, Changbo Hu, and Matthew Turk. Non-negative matrix factorization framework for face recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(04):495–511, 2005.

[165] J. W. Weibull. *Evolutionary Game Theory*. Cambridge University Press, Cambridge, UK, 1995.

[166] Stefan Wild, James Curry, and Anne Dougherty. Improving non-negative matrix factorizations through structured initialization. *Pattern Recognition*, 37(11):2217–2232, 2004.

[167] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[168] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[169] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

[170] Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. Semi-supervised word sense disambiguation with neural models. In *COLING 2016*, 2016.

[171] Sebastian Zambanini and Martin Kampel. Automatic coin classification by image matching. In *12th International conference on Virtual Reality, Archaeology and Cultural Heritage*, pages 65–72, 2011.

[172] Sebastian Zambanini, Albert Kavelar, and Martin Kampel. Classifying ancient coins by local feature matching and pairwise geometric consistency evaluation. In *22nd International IEEE Conference on Pattern Recognition (ICPR)*, pages 3032–3037, 2014.

[173] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2004.

[174] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1601–1608, 2005.

[175] Zhong-Yuan Zhang, Tao Li, Chris Ding, Xian-Wen Ren, and Xiang-Sun Zhang. Binary matrix factorization for analyzing gene expression data. *Data Mining and Knowledge Discovery*, 20(1):28–52, 2010.

[176] Zhong-Yuan Zhang, Tao Li, Chris Ding, and Jie Tang. An nmf-framework for unifying posterior probabilistic clustering and probabilistic latent semantic indexing. *Communications in Statistics-Theory and Methods*, 43(19):4011–4024, 2014.

[177] D. Zhou et al. Learning with local and global consistency. In *Adv. Neural Inf. Process. Syst.*, volume 16, pages 321–328. 2004.

[178] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Scholkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[179] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2018.

[180] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[181] Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of computer science, 2005.

[182] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, 2002.

[183] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.