Chapter 1

Collaborative Reconstruction Network

The Collaborative Clustering paradigm is based on the prerequisite that each view has to contain a set of common individuals (described by different sets of features for the horizontal case) as big as possible to allow to exchange information. However, the individuals which are not common to all the views are not used in this paradigm. We introduce in this section a method which use these remaining data to construct an approximation of the individual in the views where it is missing.

1.1 Context

This method may be used to solve problem such as cold start in some cases. For example, if the scores of a player are known for a certain amount of games, its predicted scores could be approximated for a game he hasn't played yet. This approximation could then be used to improve match-making (the action to group people supposedly balanced teams before entering a game) or even to recommend a new game to this player. This would typically be impossible without prior information of the player, but the system presented here takes the information from external views and use it to build an approximation of the seeked information.

This problem of predicting missing values based on existing ones is not new, and has already been extensively studied in fields such as recommendation (in [1], [3], [10] and [2] for example) or such as collaborative filtering [5]. However, what we present here is a reconsutruction system with an additionnal constraint: a view should not have access to original data from its external peers.

This constraint has naturally appeared because of the growing concern of people regarding their private data and the use that is made of them. The fast growth of social networks such as Facebook and Twitter or even the targeted advertising performed by Google are examples of generation and use that can be made of people everydays data. Based on this concern, our method shares the information without communication of any original data between views. This is based on the anonymization of each original database using an autoencoder (see Section 1.2 for further details). Using this method, the information which is transfered between views do not permit to recover the original data, while being usable for further analysis (see Section 1.3 for a detailed architecture of the system). In the next is presented a quick summary of what is a Neural Network, the machine learning algorithm on which is based the Collaborative Reconstruction System presented here.

1.2 Neural Networks

1.2.1 Introduction

Neural Networks are a specific kind of Machine Learning algorithm based on an analogy of the interaction of the neurons in a human brain. Their history has known many steps the most known being the presentation of the perceptron (a.k.a. a neuron) by Rosenblatt in 1958 [8], the use of the backpropagation algorithm by Werbos in 1975 [11] and the presentation of the deep beliefs networks by Hinton in 2006 [4]. The original version has been modified to produce several types of neural networks, depending on the aim to achieve. The two most famous being Convolutional Neural Network [6] to perform image analysis and the Recurrent Neural Network [7] which are used to analyze temporal data. In this thesis, we are only interested in the Multi Layer Perceptron. The following sections briefly sum up the principal components of a Multi Layer Perceptron (MLP).

1.2.2 A neuron

A MLP is made of several layers of several neurons (see Figure 1.1), each having a set a parameters which are trained during the MLP learning. To get the ouput of a neuron, each feature of the input vector is weighted by a parameter of the neuron, before being summed and put in an activation function to get the final output. Regarding the activation function, the first one which has been used is the sigmoid function, which definition can be found at Equation 1.1. There are many different activation functions which can be used, however the one which tends to be the most commonly met in recent research work is the Rectified Linear Unit (ReLU), which definition is simply $ReLU(x) = \max(0, x)$. The activation function being known. The backpropagation algorithm is applied to train the weighting coefficients of the neuron.

$$sigmoid(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)} \tag{1.1}$$

1.2.3 The backpropagation method

The backpropagation method consists in the the propagation of the gradient of the error between the ouput of the MLP and what is expected from the output to the input of the system, hence the term backpropagation. The main equation of the gradient descent is the one presented in Equation 1.2, with w being the parameter to optimize, and E the error function depending on w. The minus

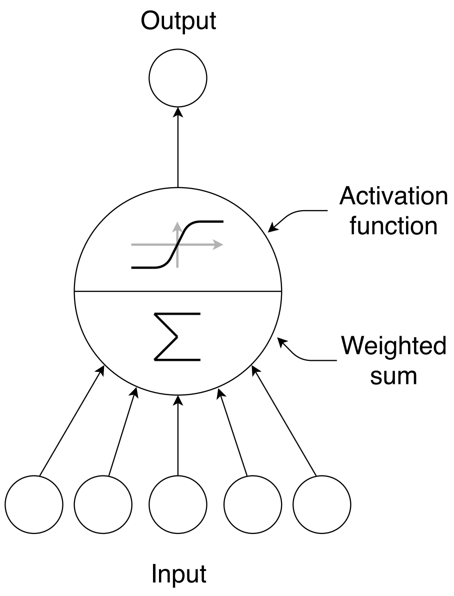

Figure 1.1: A single neuron. Each input x_i is weighted by a parameter w_i before being summed and put in an activation function. The output from this funtion corresponds to the output of the neuron.

symbol represents the idea that, when using this method, one tries to achieve the lowest point of the error function, as graphically represented on Figure 1.2.

$$w_{new} = w_{old} - \varepsilon \times \frac{\partial E}{\partial w} \tag{1.2}$$

The main difficulty of the update of the parameter using Equation 1.2 is to compute the value of the partial derivative of the error function E. This is achieved using the partial derivative composition property considering that the E function can be written as follows:

$$E\left(x_{target}, x_{output}, W\right) = l\left(x_{target}, f\left(\sum_{i=1}^{I} w_i x_i\right)\right) \tag{1.3}$$

With l a loss function such as the l_2-norm, f the activation function of the neuron, x_i the i-th value of the input (with a total of I input) and w_i the corresponding weight of the neuron. For clarity of the equation, the following notation will be used:

$$a_i = \sum_{i=1}^{I} w_i x_i \tag{1.4}$$

This allows to express the partial derivative of E in the following way:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial f\left(a_i\right)} \frac{\partial f(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_i} = \frac{\partial E}{\partial f\left(a_i\right)} \frac{\partial f(a_i)}{\partial a_i} x_i \tag{1.5}$$

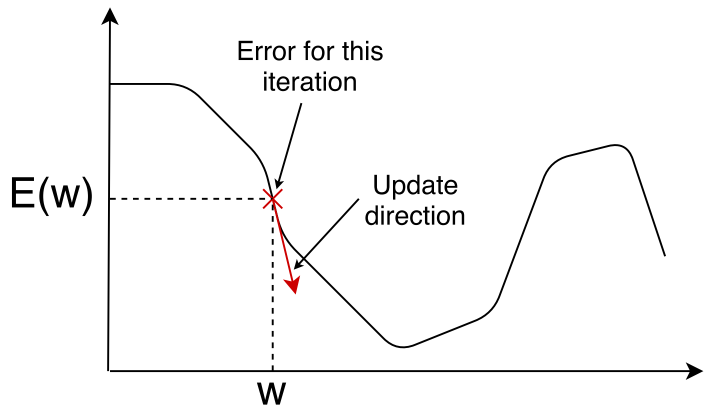

Figure 1.2: One step of gradient descent. The red cross is the current point, while the red arrow represents the direction in which the parameter has to be updated to lower the error.

Equation 1.5 being generic, it can be used with any combination of loss and activation functions. The same composition rule is also used when dealing with several layers of neurons, but in this case the partial derivate of a_i by w_i has to be composed again in order to "reach" the parameter w_i in the following layer.

Knowing this method, the learning of a MLP is performed by iteratively applying Equation 1.2 to all the parameters of the network until the norm of the gradient is small enough to be considered negligeable. In the following Section are presented the two kinds of Neural Networks which are trained by this method and which are used in the CRS.

1.2.4 MLP and Autoencoder

The term MLP designates the supervised Neural Network algorithm which makes possible to learn a regression between the input and the output. This definition implies that the input and the output have to be different. A special kind of Neural Networks, presented in [9], uses the input of the system as its output. They are called Autoencoders, because the intermediate layers of the networks, and more specifically their activations, can be used as codes to represent the input individuals. A graphical representation of both the MLP and the Autoencoder are presented on Figure 1.3.

This kind of network is composed of two subnetworks trained together: the one between the input and the hidden layer used as code called the encoder, and the one between the code layer and the output called the decoder. While it has not been strictly demonstrate, one of the advantage of this kind of network is to make the coded version of a sample undecryptable without the access to the decoder network.

The use of these two kinds of network in the Collaborative Reconstruction system is detailed in Section 1.3.

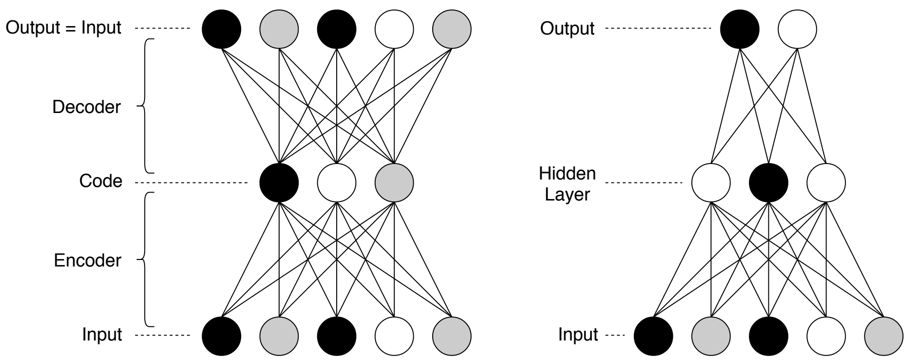

Figure 1.3: An Autoencoder (left) and a Multi-Layer Perceptron (right). The intensity of the grey in each neuron symbolizes its activation.

1.3 Architecture

To recall Section 1.1, the Collaborative Reconstruction System is defined through 3 different axes:

- The reconstruction of missing data…
- …in a collaborative context…
- …with a privacy constraint.

The design of the method has been defined following these three points. Thus, the global architecture of a CRS is made of 3 main components:

- A set of Autoencoders to encode data available in each view. This ensures that no original data will be shared among the views.
- A set of Multi-Layer Perceptrons to decode the data got from each external view.
- A method which combines the representations received from each external view, here called the Masked Weighting Method.

This Section presents the use of each component and how they are linked together. A graphical representation of the whole system can be found on Figure 1.4.

This system is based on the horizontal Collaborative Clustering (see Section 1.4.3)

1.3.1 One Autoencoder per view

To ensure that no view can access the original data from an external view, one autoencoder per view is trained in order to cypher the original data. The advantage of this method is that the data is encoded under the form of a scalar vector which can be used for further applications. The architecture of the autoencoders does not have to be the same for all the views. The important point is to ensure that the representation makes possible to get a good reconstruction

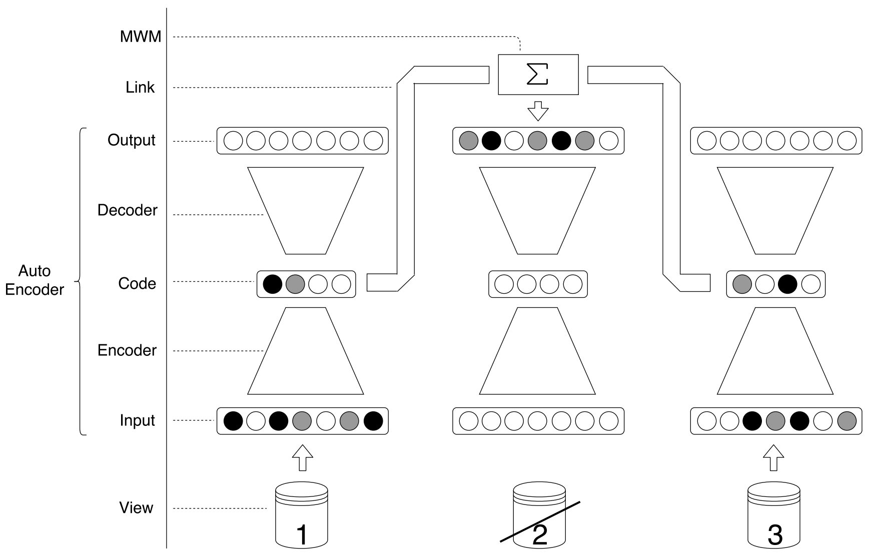

Figure 1.4: Architecture of a Collaborative Reconstruction System with 3 views. In this example, an individual is missing in the second view.

of the individual at the output. In other words, one wants to ensure that the code is defined in such a way that it allows to retrieve the original data: the difference of information between the individual and its coded version should be as small as possible. This latter point is important because if it is not verified, the external reconstruction based on the defective coded version may be greatly harmed.

However, in order to make it possible for a Link (see Section 1.3.5) to reconstruct an individual, the size of the code should also be as small as possible. The smaller the size of the code, the easier the training of the associated Link.

The first step of the training of the CRS is to train one Autoencoder per view and to encode each original dataset using these latter. At the end of this step, N encoded datasets are available for the next step. From now, the original dataset of the i-th view is called V_i and the subset of V_i which only contains individuals which also have a representation in the j-th view is called $V_{i|j}$. This clarification of notation is important because during in the following Section, the training of the Link relies entirely on the subset of each dataset with common individuals.

1.3.2 Links to get the external reconstructions

When each original dataset has been encoded, it is possible to start the training of the MLP which are responsible of the reconstruction of the missing individual. In the context of this system, the MLP are called the Links to represent the fact that their role is to draw a link between each pair of views. The training process is the same as the one described in Section 1.2. However, the datasets used as input and output have to be precised: because of its role, a Link needs to have each time two representations of the same individual, namely the one coded by the autoencoder in the external view, and the original one from the output view. This is the context previously defind in 1.4.3 about the Horizontal Collaborative Clustering. A graphical representation of the learning process of a Link is displayed on Figure 1.5.

It has to be noted that, because of the privacy constraint, the learning is supposed to be launched in the view with the original output data. For example, in a problem with 3 different views, the Link trained with the encoded database of the first view and the original database of the second view is supposed to be accessible only by actors from the second views, because only them are supposed to have access to the original data from the second view. Moreover, during the training, when looking at the error produced by the network, the reconstruction based on the coded version of an individiual x_i has to be compared with the version of x_i available in the output view.

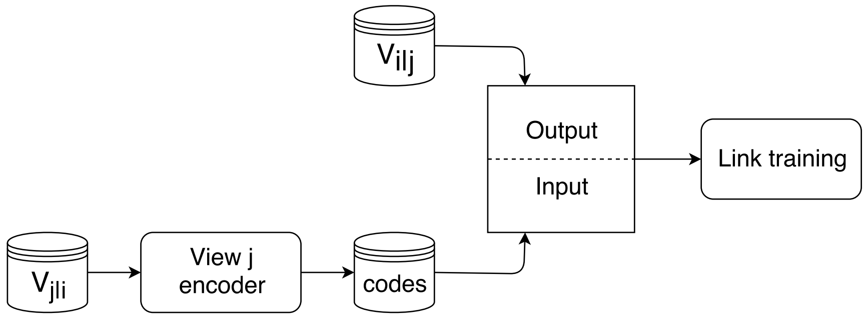

Figure 1.5: Training process of a Link. Each external dataset is encoded and is feeded as input to the Link, while the original dataset is used as the output.

Here again, the architectures of the Links are independent one from another, the only obvious constraints being that the input layer has the same number of neurons than the code it has to decode and that the output layer has the same number of neurons than the dimension of the output space.

When a view i has access to the $N-1$ Links trained with the $N-1$ respective external datasets, it has access to $N-1$ representations of a potentially missing individual. In the next Section is presented a weighting method which aim is to find the best combination of these $N-1$ individuals in order to get the best final reconstruction.

1.3.3 The Masked Weighting Method

1.3.4 The Reconstruction Process

1.3.5 The system Modularity

1.4 Experiments

1.5 Limitations

1.6 Perspectives

Bibliography

[1] Wei Chen, Zhendong Niu, Xiangyu Zhao, and Yi Li. A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web*, 17(2):271–284, 2014.

[2] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

[3] Blake Hallinan and Ted Striphas. Recommended for you: The netflix prize and the production of algorithmic culture. *New Media & Society*, 18(1):117–137, 2016.

[4] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[5] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[8] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[9] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[10] Zhibo Wang, Jilong Liao, Qing Cao, Hairong Qi, and Zhi Wang. Friendbook: a semantic-based friend recommendation system for social networks. *IEEE transactions on mobile computing*, 14(3):538–551, 2015.

[11] Paul Werbos. Beyond regression: new fools for prediction and analysis in the behavioral sciences. *PhD thesis, Harvard University*, 1974.