# A NeuroEvolutionary Neural Network-Based Collaborative Filtering Recommendation System

by

Seyedamin Monemian

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (MSc) in Computational Science

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

# Abstract

The success of a neural network-based recommendation system depends on finding an architecture to fit the task. The Fixed Topology NeuroEvolution approach is considered outdated when compared with an automated method for optimizing neural network structures. A NeuroEvolutionary algorithm has been developed in order to enhance the structure of a neural network to yield a more accurate recommendation system based on some of the existing benchmarks for measuring recommendation capability. The genetic algorithm ensures us to gain a more efficient topology produced by different generations through each step of the evolution process. Results show that this method performs better than many other algorithms and is close to the Singular Value Decomposition based algorithm developed by Funk as a benchmark standard.



## Keywords

NeuroEvolution, Neural Network, NeuroEvolution of Augmenting Topologies (NEAT), Recommendation Systems

# Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Julia Johnson for the continuous support of my study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

I would like to extend my thanks to the thesis committee members: Prof. Kalpdrum Passi and Prof. Abdellatif Serghini.

Finally, I must express my very profound gratitude to my family, especially my beloved and supportive wife, Samin, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

# Contents

# List of Tables

# List of Figures

# Appendices

# Chapter 1

# Introduction

It seems likely that development of the human brain and the process of biological evolution worked in conjunction with each other to bring about the transformation from single celled organisms to intelligent humans. The biological neural network comprising soma, dendrites, axons and synapses are simulated in artificial neural networks (ANNs) by neurons, inputs, outputs and weights, respectively. Genetic algorithms (GAs) use a process similar to the Theory of Evolution by Natural Selection proposed by Charles Darwin to find near optimal solutions. Picking fittest individuals from each generation, breeding them together and adding random mutations simulates the process of biological evolution.

A subfield known as NeuroEvolution has emerged that is concerned with the benefits of combining ANNs and GA. ANNs can learn while GAs cannot. But ANNs are prone to missing the best solution by getting stuck in local minimums

or maximums. Randomness inherent in the GA approach tends to break these stuck points when GA and ANN are combined. Henceforth, the only neural networks that will be discussed are ANNs, abbreviated NN.

Recommendation systems provide suggestions for users of items to help them make decisions such as what movies to watch, what friends to follow or what items to buy. The combination of NN and GA was explored for its application on recommendation systems.

## 1.1 Statement of the Problem

An approach is explored to finding an optimal neural network topology for a collaborative recommender using an evolutionary/genetic algorithm. Neural networks and deep learning have been previously used in recommender systems. Here we use an established NeuroEvolution of Augmenting Topologies (NEAT) algorithm. The novel concept demonstrated is to use the NEAT algorithm to find an optimal network topology for application in recommender systems. NEAT provides the tools for use of GA to find an optimal network topology. The application to recommender systems is the novelty. To that end, we also developed a fitness function that uses recommendation error as a score to evaluate proximity to an optimal solution.

## 1.2 Motivation

Among the implementations of neural networks for recommendation systems, the fixed topology (structure) neural networks suffer from the following challenges. Because artificial neural networks may trap in a more or less local optima, some way is needed to overcome this issue to find the best solution in

the global search space. In addition, as regular neural networks are supervised methods, they are going to fit their model to a predefined outcome leading to challenges when applying them on a real world case. This thesis proposes an unsupervised algorithm by employing an evolutionary process to better traverse the search space to provide better recommendations.

## 1.3 Objectives

An evolutionary neural network-based collaborative filtering recommendation method was developed that enhances the recommendations of the current techniques . A python application is proposed that implements NEAT elements to provide a neural network that enables training and suggests a model to predict upcoming real case events. Quality of the recommendation method will be judged by a comparison with one of the state-of-the-art methods named FuncSVD. The first objective is to apply evolutionary processes on neural networks. The second is to train and test each neural network to yield a better recommendation result.

## 1.4 Outline of the thesis

This thesis consists of the following chapters. In Chapter 2, research in recommendation systems and personalization areas is reviewed. Different aspects of recommendation systems will be introduced there and a wide variety of artificial intelligence and more specifically, machine learning methods applied to recommendation systems will be discussed. The methodology employed to implement and evaluate a neuroevolutionary recommendation system will be described in Chapter 3. In Chapter 4, implementation details are provided and it demon-

strates the results of running the system on a well known recommendation dataset and compares those results with those of a renowned recommendation system on the same dataset. We see in that chapter how the neuroevolutionary method improves the results of the recommendation process. In Chapter 5, we summarize the outcomes from our experiments and suggest future research directions.

# Chapter 2

# Background and Related Work

A recommendation system is a software program that narrows down selections for users based on their expressed preferences, past behavior, or data that can be mined about the user or other users with similar interests (paraphrased from Wikipedia).

## 2.1 Taxonomy of Recommender Systems

A recommendation system assesses a user by means of user profiles. In [1], recommender systems are considered in terms of their profile generation and maintenance techniques, and in terms of their profile exploitation techniques. Those authors state five design decisions that must be made concerning how the system assesses its users: the profile representation technique, the technique used to generate the initial profile, the source from which user tastes, interests

and preferences will be derived, the profile learning technique that gives meaning to user information and puts it in the profile, and the profile adaptation technique to account for changing user interests.

Given the availability of information in user profiles, the system exploits it to make recommendations. The problem of selecting the most appropriate information from a vast array requires an information filtering method. Content-based, demographic and collaborative are possible filtering methods as shown in the Figure 2.1.

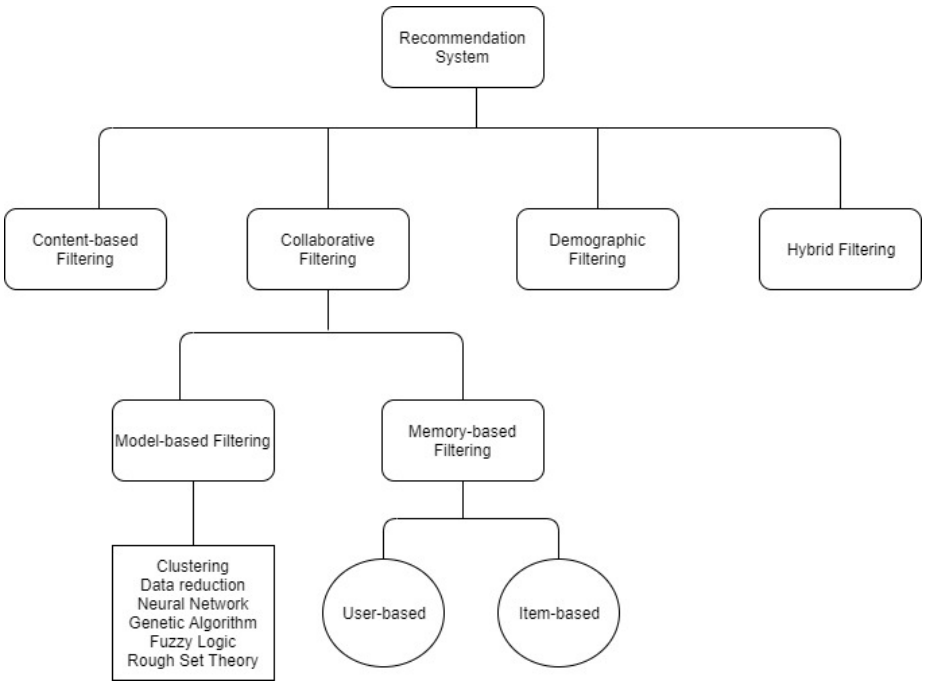


Figure 2.1: Taxonomy of Recommender Systems [2]

### 2.1.1 Content-based Filtering

Content-based Filtering (CBF) uses metadata such as the item's attributes (in movie example, genre, actor, etc.) to recommend items. In other words, it depends on the similarity between the items, rather than the behavior of users. The idea behind the scene is that if you like a certain item, you are most likely to like something that is similar to it. In brief, such algorithms suggest the items which are more likely to be liked than the previously chosen items. For instance, text recommendation systems like the newsgroup filtering system use the words of their texts as features. The learning process of a content-based recommender is based on the items' characteristics of what a user has regarded by ratings. In other words, an "item to item correlation" is used to create a user profile according to applied learning method. Some of the well-recognized methods include decision trees and neural networks. Mostly, CBF uses an item profile (i.e. a collection of unique characteristics) defines an item in the whole system. According to a weighted vector derived from item attributes, the system is responsible to create the content-based user's profile. Based on a weighted vector of item features the system creates a content-based profile of users. The importance of each feature has a corresponding vector weights and is calculated via the rated item vectors through different methods [3].

### 2.1.2 Demographic Filtering

Recommendation Systems based on Demographic filtering (DF) classify users according to their demographic information and recommend services accordingly. User classification in conventional explanations is done by DF in order to represent the attributes of users' classes. Demographic data detects the people who are interested in related services, and that information is interesting for

some trusted third parties to suggest individual services to the users. DF creates categories of users which have similar demographic characteristics and then the cumulative buying behavior or preferences of users within these categories are being tracked. For a new user, recommendations are made by first finding the category into which he falls and then the cumulative buying preferences of previous users are applied to that category. The same as collaborative filtering, DF methods also consider "user-to-user" correlation using the dissimilarity information. The big difference between collaborative and content-based techniques and DF approach is that the former, require a history of user ratings which is not of the kind required by the latter [3].

### 2.1.3 Collaborative Filtering

Collaborative filtering is based on the behavior of a group of users to suggest a recommendation to others. For example, if a user $A$ likes products $X$ and $Y$, and if a user $B$ likes product $X$, it would be a good chance that user $B$ likes $Y$ as well. Let me describe that in a real movie recommender system. Suppose that a group of people have the same rating to movies $X$ and $Y$. If another user rates movie $X$ the same as the group, the movie $Y$ would be recommended to the new user as a relevant suggestion.

This filtering is probably the most widely implemented and most mature of all recommender systems. Collaborative systems are based on Collaborative filtering methods that require collecting and analyzing a large amount of information on user ratings. They generate new recommendations based on inter-user comparison activities and predict what users will like based on similarity found by means of those comparisons. A key benefit of the DF approach is that there is no need to rely on machine analyzable content and resulting in the

ability to precisely recommend complex items such as movies, songs, clothes, etc. without requiring an understanding of the item itself. Many algorithms have been used in measuring item similarity in recommender systems. CF is based on the hypothesis that people who agree in the past will agree in the future and that the relationship between the kinds of items people liked in the past will continue in the future. These systems can be memory or model based, comparing users alongside each other using relationship or other procedures in which a model is derived from the historical rating data and used to make predictions [3]. The technique of collaborative filtering can be divided into two categories: memory-based and model-based [4].

#### 2.1.3.1 Memory-based Collaborative Filtering

Memory-based processes approach the collaborative filtering method considering the whole database. A key role in finding neighboring users with someone is the items that already have been previously rated by him. As a neighbor of a user is found, a variety methods can be applied to combine the favorites of neighbors to make suggestions. There exist two main ways to achieve Memory-based CF, user-based and item-based techniques.

#### 2.1.3.2 User-based Collaborative Filtering (UBCF)

This kind of recommendation system suggests items to a user that are already the interesting items of another user who is similar to the given user. Figure 2.2 clarifies this system more accurately. User $A$ has a desire on item 1, and suppose that user $B$ has very similar attitudes to user $A$. So the system makes a suggestion of item 1 to user $B$, as well. As is obvious, the rating scores of items by users are needed for prediction [5]. Several algorithms are used to calculate

the similarities between users. K-Nearest Neighbor is one of the most popular algorithms to find the nearest neighbors according to similarities between users (We will discuss these kinds of algorithms in the next section). The next step would be the generation of a prediction in terms of items. This will happen by the combination of rating scores of neighboring users based on similarity weighted averaging [6].

#### 2.1.3.3 Item-based Collaborative Filtering (IBCF)

Focusing on the items, this approach recommends items by considering similarities among the item and items that are already liked by the user. As shown in figure 2.3, if two items 1 and 3 are similar, and user $A$ likes item 1, the system will recommend item 3 to user $A$ as well.

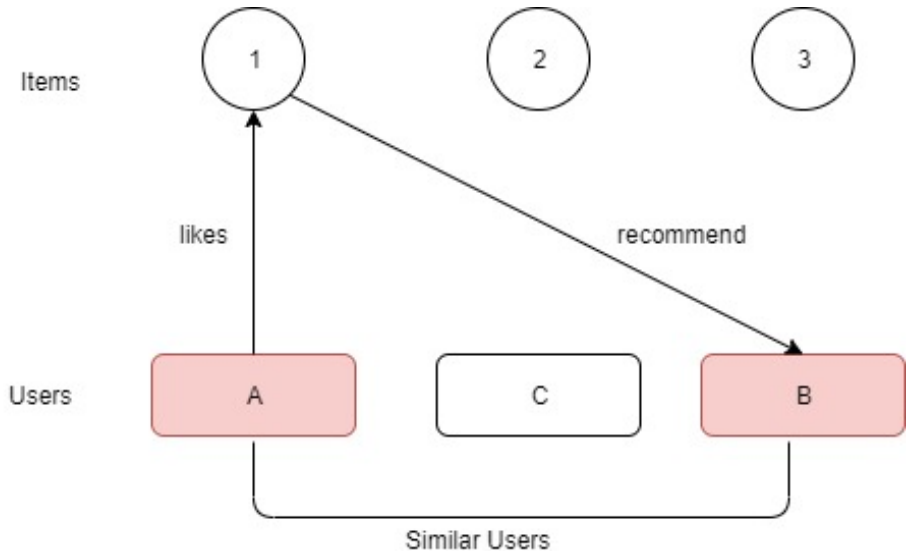


Figure 2.2: User-based Collaborative filtering

#### 2.1.3.4 Model-based Collaborative Filtering

This technique employs the previous ratings to learn a model in order to improve the performance of the Collaborative Filtering Technique. A variety of machine learning and data mining techniques can be used to build the model. These techniques can be used to quickly recommend a set of items because they use a pre-computed model. They have shown producing suggestion outcomes which are comparable to neighborhood-based recommendation systems. These techniques include data reduction and especially dimensionality reduction algorithms such as latent semantic analysis, singular value decomposition, clustering and regression. Model-based methods examine the user-item matrix to find the relationships among items; those relations are used to rank the top-N suggestions.

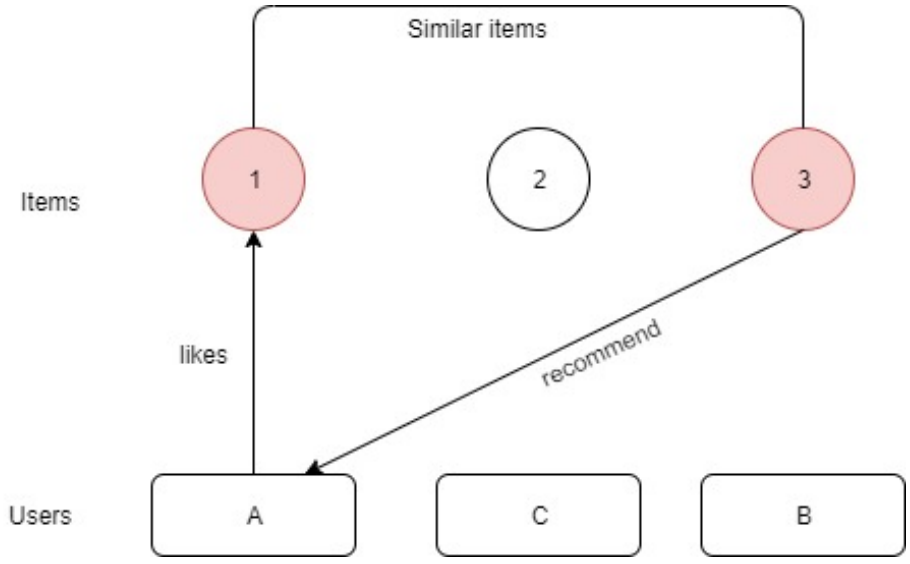


Figure 2.3: Item-based Collaborative filtering

## 2.2 Artificial Intelligence in Collaborative Filtering Recommenders

Artificial intelligence techniques are increasingly being used in many different aspects of model-based recommenders. The use of learning algorithms has changed the manner of recommendations from recommending what to consume by users, to recommending when to actually consume a product. It is therefore important to examine learning algorithms used in model-based recommender systems.

### 2.2.1 Classification and Clustering

Classification and clustering are data science concepts to solve real world problems. Classification algorithms are generally called "supervised learning algorithms" and clustering algorithms are called "unsupervised learning algorithms". Supervised learning is a process of learning from examples. The learning algorithm is provided with a training set which is a set of labeled examples and a test set of unlabeled examples. The aim is to learn from the training set, so as to identify the labels (classes) of examples in the test set with an acceptable level of accuracy.

The authors [7] indicate that unsupervised learning is based on correlations among the input data, and can be used to find patterns in the input data without the help of a teacher. In unsupervised learning, the aim is to model distribution in the data in order to learn more about it. The main goal of a clustering algorithm is to group similar objects in a dataset as a collection of data. This type of learning supports an unlabeled data type. Some readers may see a similarity between clustering and collaborative filtering. Clustering and collaborative filtering both operate in a continuous space but the similarity

ends there. Clustering is done in dense data of relatively low dimension, while in collaborative filtering the data tends to be sparse and high-dimensional. Clustering is a structure-finding exercise, while collaborative filtering is a ranking problem. Clustering is defined by a distance function among data points while collaborative filtering requires inferring which missing dimension is ranked highest. Best performance for K-means clustering (discussed shortly) is achieved by including a cluster initialization algorithm while collaborative filtering typically proceeds via a matrix completion algorithm.

#### 2.2.1.1 K-Nearest Neighbourhood (K-NN)

The K-nearest neighbor (K-NN) classifier searches for $k$ objects from the training set that are closest to the object in the test set. One of the issues of using K-NN is how to select the right number of $k$. Choosing the number of $k$ to be too small results in sensitivity to noise points. Setting the number of $k$ to be too large, will end in the inclusion of too many points in the result from other classes [8]. In [9], the K-NN classification method has been used on-line and in Real-Time to identify clients/visitors click stream data, matching it to a particular user group and recommending a tailored browsing option to a specific user at a particular time.

#### 2.2.1.2 K-means

The K-means clustering algorithm splits a dataset into $K$ groups (clusters) based on similarity. For each cluster, the center is represented by the mean value (centroid) of the objects in that cluster. This k-means clustering algorithm [10] uses the Euclidean distance as a measure to assign the objects to their closest cluster center. Clearly, there is a problem, because how do we find the centroids

in the first place. The standard algorithm assigns n observable points to $K$ clusters as follows:

1. Randomly generate $K$ initial means.
2. Create clusters around said means.
3. The centroids of the clusters become the new means.
4. Repeat Steps 2 and 3 until the convergence of the algorithm.

The given algorithm can have poor performance. The k-means++ algorithm specifies a procedure to initialize the cluster centers (step 1) before proceeding with the remaining steps to alleviate the worst case performance. Note that K-means++ includes a cluster initialization algorithm, whereas K-means is simply a clustering algorithm.

Many additional strategies have been developed to initialize k-means and determine the number of clusters. For example, given one million rows one could start by drawing a random sample and running k-means on the sample to compute initial cluster centroids. Then the sample centroids could be used as a starting point for the full k-means run.The sample centroids are called initial seeds. It is argued in [11] that the initialization algorithm should ascertain that the initial centers are close to the final centers for faster convergence. In that same work the authors argue that there should be no random component in K-means because a random choice of cluster centers may yield different clustering results on different runs of the algorithm. [12] proposes two varieties of algorithms for collaborative filtering recommendation systems. The first one uses the improved k-means clustering technique (K-means++) while the second one uses the improved k-means clustering technique coupled with Principal

Component Analysis as a dimensionality reduction method to improve the recommendation accuracy for large data. The research in [7] applied K-means clustering whose initial seeds are optimized by using genetic algorithms (GA), which is called GA K-means, to a real-world online shopping market segmentation case. In this study, they compared the results of GA K-means to those of a simple K-means algorithm and self-organizing maps (SOM). SOM is the clustering algorithm based on the unsupervised neural network model and has been applied to many studies because of its good performance [7].

#### 2.2.1.3 Inductive Learning and Decision Trees

Inductive learning involves predicting the output for a new instance, hence, generalization. Decision trees are an inductive learning approach in which decisions are modeled in a tree-like structure. Decision tree based recommenders search the tree to compile a recommendation list for a user. The items to classify consist of attributes and their target value. The tree nodes contain either a) *decision nodes* in which a single attribute-value is tested to determine to which branch of the subtree applies, or b) *leaf nodes* which specify the value of the target [13]. The authors [14] seek to improve the quality of recommendations for an Internet shopping mall. The methodology is based on a variety of techniques that include decision tree induction and association rule mining (to be discussed in section 2.1.3). In [15] an improvement over previous decision trees used in recommendation systems was realized by, rather than having only single items at their leaf nodes, lists of recommended items appeared there. This led to a reduced amount of search when using the tree.

#### 2.2.1.4 Naïve Bayes

Employing the Bayes Theorem, Naive Bayes (NB) classifiers are an easy to use model assuming that the predictors are independent. Because NB models usually do not contain complex iterative variables, it is very simple to be applied on huge datasets and interestingly it usually yields good results, so it is also to be preferred to other complicated classification algorithms. One of the most beneficial usage of NB is that it needs only a small portion of data on which to be trained, and according to the assumption of variable independence, it needs just the variances of variables of each class not the whole covariance data. The drawback of NB is it supposes the variables to be independent which in some cases leads to reduced accuracy of method. Paper [16] showed a hybrid recommender system which combines a NB classifier with a collaborative filtering method, and demonstrated its higher accuracy and performance as compared to the recommenders in the same area. [17] applies a NB in order to construct a recommender methodology that can be considered to be conscious about time. It combines time-user rating, item attributes and time data to improve the accuracy of recommendation to satisfy users' desires at a specific time. In paper [18], the authors proposed a content-based approach that is primarily based on NB to find similarity between the items in order to make better recommendations. [19] presented a recommendation method which is constructed on a social network and a NB classifier to suggest best ranking books to customers. Extraction of opinions, review summarization and categorization have been done in addition to consideration of the books attributes such as price. [20] combined NB with singular value decomposition (SVD) extensions to deal with the sparsity problem. The result showed that inclusion of NB decreased the Mean Absolute Error (MAE) in comparison with the absence of NB.

#### 2.2.1.5 Fuzzy C-means

The main focus of fuzzy logic is to model some actual world notions that cannot be described precisely. Specifically, the definition of a fuzzy set over a universe of discourse, extends the notion of a set through the introduction of the degree of membership of the elements. The Fuzzy C-Means Clustering algorithm, like K-means, assigns $n$ observable points to $C$ clusters. However, in fuzzy clustering, data points are allowed to belong to multiple clusters as defined by their degree of membership. The algorithm proceeds as follows:

1. Choose a number of clusters $C$.
2. Assign coefficients, representing membership grades, to all data points randomly.
3. Compute the centroid for each cluster.
4. Compute the coefficients of all data points for each cluster.
5. Repeat Steps 2 and 3 until the convergence of the algorithm.

In paper [21], the authors propose a clustering CF framework and two clustering CF algorithms: Item-based Fuzzy Clustering Collaborative Filtering (IFCCF) and Trust-aware Clustering Collaborative Filtering (TRACCF). While clustering involves dividing a set of objects into a specified number $k$ of clusters, so that objects are similar to other objects in the same cluster, Fuzzy C-Means (FCM) is a particular clustering method, whose adoption allows that an object belongs to two or more clusters with different membership degree.

[22] proposes a Fuzzy C-means approach for user-based Collaborative Filtering, and its performance against different clustering approaches was assessed. [23] reviewed the use of fuzzy tools in recommender systems, for detecting the

more common research topics and also the research gaps, in order to suggest future research lines for boosting the current developments in fuzzy-based recommender systems. The research also presents the contributions focused on incorporating fuzzy approaches in both content-based and memory-based collaborative filtering recommendations.

#### 2.2.1.6 Rough Set Theory

Rough set theory was developed by Zdzislaw Pawlak in the early 1980's. Representative publications include [24], [25]. Basic concepts of the theory include those of lower and upper approximations, indiscernibility class and boundary region. Lower and upper approximations are sets that are used to approximate a concept (also a set). The members of an indiscernibility class are indistinguishable from each other with respect to a subset of considered attributes, and the boundary region expresses vagueness or imprecision of imperfect knowledge. The boundary region is the difference between the lower and upper approximations. The lower approximation contains elements that are certain to be members of the concept to be approximated and the upper approximation contains elements that could possibly be members of the concept to be approximated. Lower approximation is a subset of upper approximation.

In a supervised learning situation the formal framework of rough set theory simplifies the search for dominant attributes in an inconsistent information table leading to derivation of shorter if-then rules. Given a set of attributes S containing attribute $A$ and a set $S' = S - A$, if the indiscernibility classes with and without A are identical then attribute A is redundant. This rule forms the basis for an algorithm that finds the dominant attributes.

A rough set and fuzzy clustering based collaborative filtering recommenda-

tion algorithm is proposed in [26]. This algorithm addresses the issue by automatically filling vacant ratings based on rough set theory, and uses the fuzzy clustering technology to compute user similarity and form nearest neighborhood, and then generates recommendations. In the paper [27], a novel ontology-based recommendation model based on a fuzzy-rough hybrid mechanism is proposed. This model combines the main ideas of both content-based and collaborative filtering recommender methods. Paper [28] applies rough set theory to simplify dimensionality of attributes. Attribution reduction is one of the main processes in the data preprocessing phase to complete the task of making recommendations.

### 2.2.2 Association Rule Mining

An association rule is an implication of the form $A \Rightarrow B$, where $A$ and $B$ are item sets that frequently occur together and $A \cap B = \emptyset$. The rule $A \Rightarrow B$ can be taken as "if item-set $A$ occurs in a transaction, then item-set $B$ is likely to occur in the same transaction" [29]. By such information, [30] introduces some pattern discovery methods for adaptive personalization in a recommender system, by efficiently using association rule mining among both frequent and infrequent items. That is because the pattern extraction component analyses users' sessions and utilizes a large bank of rules.

[31] presents a model-based recommendation algorithm that uses multilevel association rules to alleviate the problems of data sparseness and scalability. In this algorithm, they build a model for preference prediction by using association rule mining.

Multi-layer association rules are widely employed to discover preferences among items. [32] proposes an efficient method of producing associations rules

with higher performances based on a genetic algorithm. Association rules are discovered with genetic algorithms and the recommender system produces recommendations using these rules. Association rules are produced for target objects which can be items or users.

### 2.2.3 Artificial Neural Networks

The brain is composed of an interconnected network of neurons. Artificial neural networks (ANN) simulate the brain. Weights are the result of an iterative learning algorithm. Weights are changed iteratively as long as the system continues to respond incorrectly and form the inputs for the next iteration converging towards the desired weights. The learning is complete once the system produces only correct outputs.

Neural networks have some specific characteristics. First, they are more like a real nervous system. Second, since there exist parallel organizations, it allows solutions to problems which should satisfy multiple constraints simultaneously. Moreover, the employed rules are implicitly applied that helps simplicity and clarity. There also exist drawbacks which is the difficulty with infinite recursion and structured representations [46]. Many researchers using ANN in their works such as [47] that proposed a neural network hybrid recommender system which is used in customer services. The system associated with XML-based personal agents within a multi-agent system with suggestions about flights purchases, whose agents continuously monitor customers' interests and preferences in their commercial Web activities. It performs by constructing and automatically maintaining their profiles, in order to emphasize the advantages which the flight recommender system provides.

In [48] a combination of content-based, model and memory-based collabo-

rative filtering techniques is used in order to remove the drawbacks of Content based recommender systems, and to present predicted ratings more accurately. Feedforward back-propagation neural network is employed in order to train the data. Then, the performance of the system is examined regarding different conditions and variables such as the count of users, the ratings they performed and system model.

[49] Studied Artificial Neural Network and optimized it based on classification system. The performance of the ANN-based classification system was assessed by changing amount of generations and computing the correctness of diverse features. [50] studied a recommendation system for movie lovers using neural networks in collaborative filtering systems for consumers' experiential decisions. The experimental results reveal that it not only improves the accuracy of predicting movie ratings but also increases data transfer rates and provides richer user experiences.

#### 2.2.3.1 Deep Neural Networks

Deep Neural Networks have been widely applied in the field of artificial intelligence, from speech recognition to natural language processing and image processing. Collaborative filtering recommendation systems are another branch in which deep learning is flourishing. In the remainder of this section, a review of the application of deep learning in recommendation systems is provided based on a recent review of the subject [51].

In [52] an Autoencoder-based Collaborative Filtering method (ACF) is proposed that takes a user-based rating (or item-based rating) matrix as an input, and by applying an encoding and decoding process and optimizations, yields a model with optimized parameter and minimized reconstruction error.

In [53] another ACF method was suggested in order to predict the rankings. The input is a preference vector that reflects the users' interests in items, and the parameters are updated using Stochastic Gradient Descent regarding all feedback received. Restricted Boltzmann Machine-Based (RBM) collaborative filtering method is one of the oldest Neural Networks to be used to learning problems they use just a visible layer and a hidden layer. An efficient inference procedure to model tabular data was proposed, such as user ratings of movies, and RBM was observed to outperform SVD models [54]. Another major type of Deep Learning algorithms are Recurrent Neural Networks which are widely used to model sequential data. As it is described in [55], by learning from a back propagation algorithm, better performance was achieved for recommendation systems also alleviating the previous time heterogeneous feedback recommendation problems.

The other category of deep learning algorithms is the Generative Adversarial Network, that includes a discriminator and a generator. The idea is to train two different Neural Networks simultaneously by contesting in the sense of game theory. Wang et al [56] propose a game theoretical minimax algorithm to unify generative and discriminative information retrieval.

### 2.2.4 Genetic Algorithms

A genetic algorithm is a search technique based on Darwin's theory of evolution that incorporates the concept of "survival of the fittest". The algorithm begins by creating an initial population of problem solutions and computing the fitness score of each solution. With each iteration of the algorithm new organisms are created using the fittest organisms and less fit organisms are discarded. New organisms (children) are created using crossover and mutation applied to the

parents' chromosomes. The fitness score of the new organisms is computed and the fittest organisms are combined with the newly created ones to serve as the population for the next iteration of the algorithm.

While the traditional collaborative filtering methods try to discover similarities among users, paper [57] exposes a genetic algorithm approach for refining those similarities before the prediction process. They yield improved outputs according to the statistical analysis achieved on a variety numbers of neighbors for various similarity metrics. They showed that GA reduces the prediction error as compared to the previous traditional methods.

With the increasing growth of social networks, [58] introduces a recommendation system to suggest friends in social media networks using the idea of nature of the graphs. The graph that connects a person to his contacts is considered and the algorithm is developed to examine the user 's friend networks in three layers of depth. The second-layer friends are candidates of suggestions and the behavioral similarity is the main goal of the algorithm to be defined. The concept of user's friends and also friends of those friends are the key mechanism of analyses and development of this recommendation.

[59] focused on the learning materials and tried to propose a hybrid recommender based on a genetic algorithm. The main goal of the system is to deal with the huge existing information in the learning area and suggest the proper studying materials to the students. The system uses the user's ratings on some certain characteristics of items to enhance the quality of the suggestions in order to decrease the complexity of having several attributes. Moreover, the genetic algorithm is applied to reduce the scale of the environment which includes a large number of items and users.

In [60], the outputs of a traditional recommendation algorithm is used as the

input base of a genetic algorithm in order to overcome the large scale of users and items. It defines the weight vectors as different prediction performances of recommenders, so the problem was changed to optimize the weight vectors according to the accuracy of prediction which presented as the fitness function.

## 2.3 NeuroEvolution

Getting stuck in a local optima is likely when using a regular neural network. Due to much randomness in evolutionary algorithms, they are less likely to get stuck in such a way. Moreover, an evolutionary algorithm is recognized as an unsupervised method which means it performs in a natural evolution over time whereas regular neural networks are supervised, trying to model a known output. They are totally aware of what they want to achieve. In other words, evolutionary models are more suitable for dynamic problems which do not have a known outcome as compared to neural networks that know what they want to attain.

Combining genetic algorithms and neural networks involves encoding information about the neural network in the genome of the genetic algorithm. See Figure 2.4 that shows the flow of the combined algorithm. The genetic algorithm operates on a population of network topologies to find the fittest NN for the problem at hand. The neural networks that produce the highest fitness are selected to be bred creating the next generation. NeuroEvolution addresses the problem of how to use GA to build up NN without presupposing the best structure for the neurons and their connections.

One can make own simulated person and teach him how to walk using NN and GA. A criticism of NeuroEvolution with a Fixed Topology approach is the bizarre solutions that often arise. The first successful solution is taken

regardless of efficiency or function. This can lead to some truly phenomenal walking animations.

### 2.3.1 NeuroEvolution of Augmenting Topologies

The NEAT technique attempts to establish a balance between the fitness and the diversity of evolved solutions by alteration of both the weighting parameters and structures of networks. The paper behind the NEAT algorithm is by Kenith Stanley[61]. NEAT includes ideas for separating genomes into species which most GA do not. The fittest NNs in each generation make modifications to each other thereby improving performance. Sometimes an entirely new species

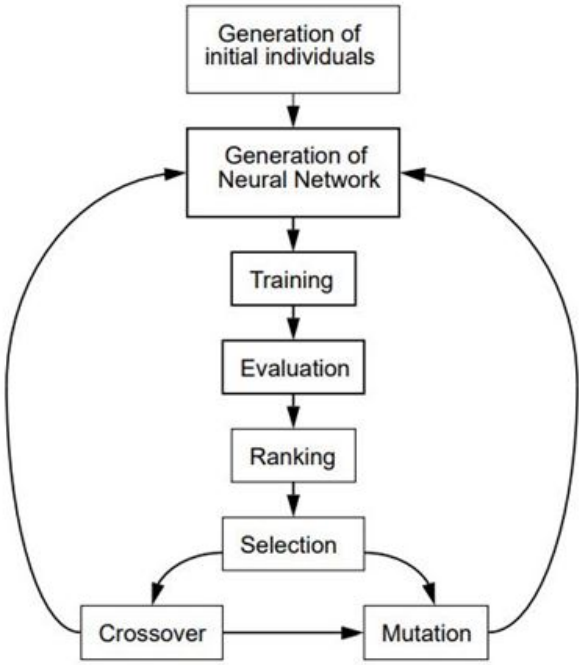


Figure 2.4: Combination of neural network and genetic algorithm

becomes prominent and dominates the others.

NEAT presents three main processes like any other GA algorithms: Selection, Crossover (recombination) and Mutation. NEAT uses an encoding method to represent the genome. There exist two kinds of genes: *node genes* and *connection genes*[61]. Node genes contain information such as the number and the type of node (input, hidden or output). Connection genes specify the node from which that connection comes, and the node to which that connection points, an assigned weight and a number named *innovation number*. Innovation number is a global incremental number that determines the historical origin of genes (when the gene has been created), which is incredibly important, because it is the base that leads to NEAT's efficient crossover. Figure 2.5 which is derived from [61] demonstrates a sample of genomes described by NEAT.

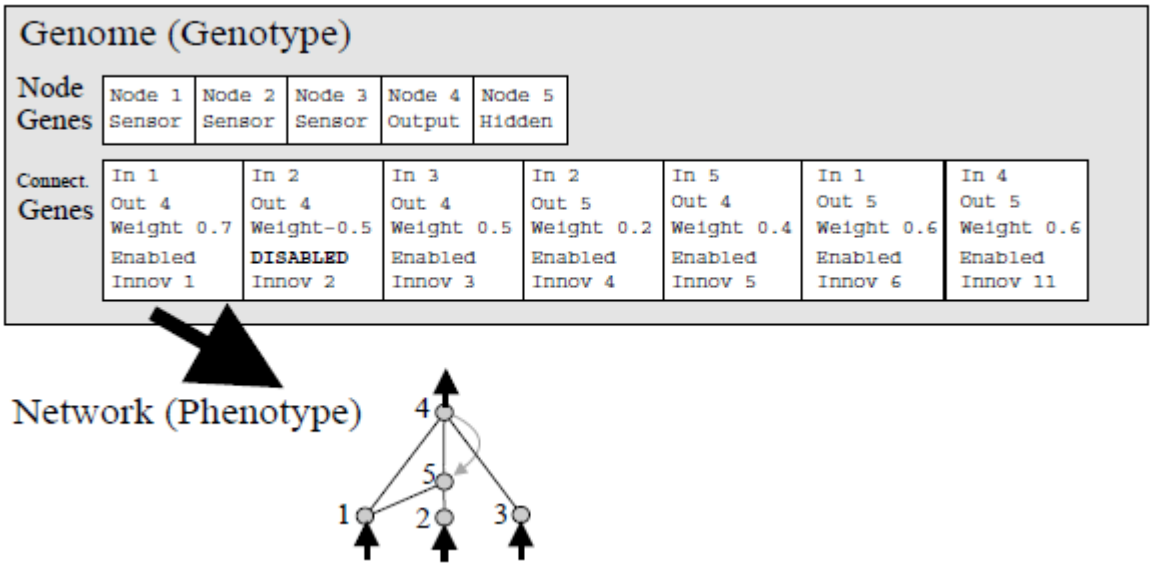


Figure 2.5: Mapping from genotype to phenotype. Three input nodes, one hidden and one output node is depicted [61]

#### 2.3.1.1 Mutation

In the mutation process, there are two types of structural mutation that may happen. Figure 2.6 illustrates the schematic of both adding a node and adding a connection. When adding a new connection, a new gene is added to the end of the genome with the next available innovation number assigned. There is some difference when adding a new node. The previous connection is made disabled, and two new connection genes and a new node gene are added to the end of genome.

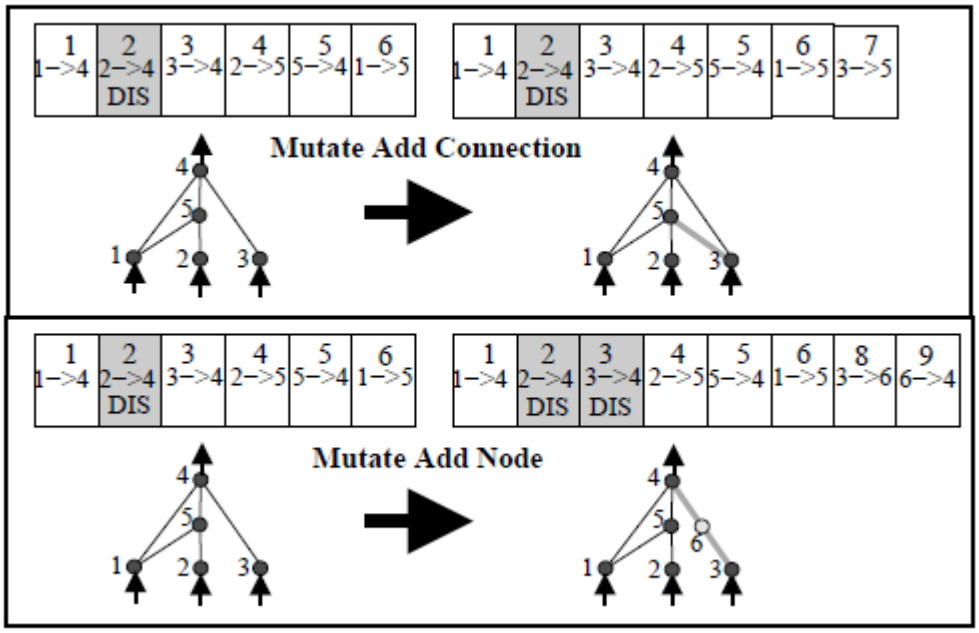


Figure 2.6: Structural mutation in NEAT. Both adding a new connection and a new node are shown [61]

#### 2.3.1.2 Crossover

Because we can derive node genes from connection genes, we only need to crossover the connection genes. The historical marking empowers NEAT with the capability of knowing exactly which genes are more likely to match up with which. The matching genes are randomly selected from a parent, so are the disjoint and excess genes, which are randomly inherited from the parent with higher fitness value. Disjoint genes are those whose innovation numbers exist in other genome, while the excess genes are those whose innovation number do not exist in the other parent (i.e. parent with less fitness value). There also exists a randomness to inherit either the disabled or enabled connections with a preset chance.

#### 2.3.1.3 Speciation

One of the challenges the previous NE methods encountered was that the newly generated genomes did not have enough time to get matured and show their effect on the algorithm evolution. Some important genomes have been eliminated from the population before they can have a chance to compete with others. The NEAT solution is *speciation*, which allows the organisms to compete within their own species not within the whole large population. In other words, putting similar topologies in the same species guaranties the individuals to have time to optimize their structure through competition within species. In addition, NEAT defines an explicit fitness sharing, which illustrates how well the individual is doing among the others in the same species, which again is possible by means of the innovation number.

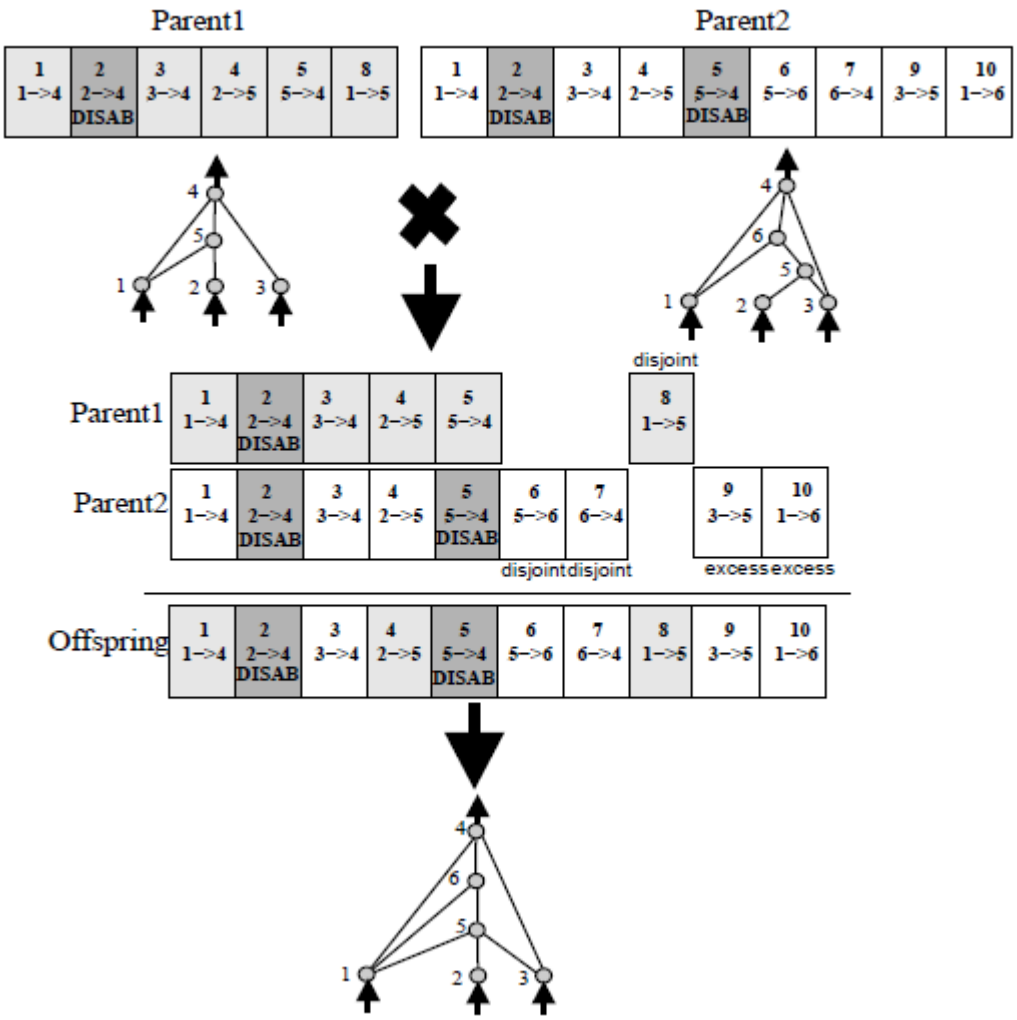


Figure 2.7: Crossover in NEAT. Matching up genomes for different network topologies using innovation numbers [61]

#### 2.3.1.4 Minimal Dimensionality

In contrast to other NE solutions which start their evolutionary process with a random topology, NEAT begins with a minimal-dimensional space (with zero

hidden nodes). The structure gets larger whenever mutation occurs, and only the useful structure remains alive from generation to generation. The search to minimal-dimensional spaces is preferred by initializing a uniform population of networks with no hidden nodes, which means all inputs attach directly to outputs. New topology is presented incrementally as structural mutations happen, and only those topologies continue living that are discovered to be useful through fitness evaluations. Therefore, NEAT guarantees that the structures are always justified by the fitness function and minimal (with no extra nodes or connections) [61].

### 2.3.2 Previous Applications

Since 2002 when Stanley and Miikkulainen published their evolutionary method [61], there emerged a variety of applications of the NEAT algorithm in different research studies. [62] combined NEAT and Convolutional Neural Networks to propose an Image Recognition subject. [63] employs NEAT to find the most appropriate fitness function for the learning agents that can improve their behaviors in a simulation environment. [64] suggests different methods of applying neuroevolution on existing deep neural networks and investigates the importance of this idea to the future of Artificial Intelligence. Moreover, [65] applies NEAT on a biological system to optimize the gene regulatory network for a specific task. [66] employs NEAT to train a computer playing video games.

## 2.4 Data Reduction

Nowadays, with the rapid growth of data in all areas of science and business, processing that large amount of data has become a complex challenge. There-

fore, data reduction strategies are considered to be of great necessity in order to solve many real world problems. The main goals of data reduction are reducing the storage size of data, efficient analysis of big data and decreasing the process time and complexity of dealing with a large amount of data. Preprocessing is needed before being able to efficiently extract useful information from the datasets. Applying a suitable algorithm on the processed data may result in a better solution to the given problem.

Data cleaning, data transformation, and data reduction are some of the main methods of preprocessing data. Data reduction techniques are performed so as to obtain a reduced representation of the main dataset. It is crucial for the reduced dataset to be much smaller in size, and also that it contains the main analytical outcomes of the original one. Data reduction has been widely used in different data mining algorithms such as principal component analysis (PCA) and factor analysis (FA). These two methods are used to reduce the number of variables to prevent the curse of dimensionality which is the difficulty in analyzing high-dimensional data [33]. In general, there exist three main types of data reduction: Dimensionality reduction, numerosity reduction and data compression.

### 2.4.1 Dimensionality Reduction

The following sentences define the problem of dimensionality reduction. Assume we have a dataset represented in a $n \times D$ matrix $X$ consisting of $n$ data vectors $x_i (i \in 1, 2, ..., n)$ with dimensionality $D$. Suppose that this dataset has dimensionality $d$ (where $d < D$, and often $d << D$). Here, in mathematical terms, intrinsic dimensionality means that the points in dataset $X$ are lying on or near a manifold with dimensionality $d$ that is embedded in the D-dimensional

space. Note that we make no assumptions on the structure of this manifold: the manifold may be non-Riemannian because of discontinuities (i.e., the manifold may consist of a number of disconnected submanifolds).

Dimensionality reduction techniques transform dataset $X$ with dimensionality D into a new dataset $Y$ with dimensionality d, while retaining the geometry of the data as much as possible [34]. Two general approaches for dimensionality reduction are feature selection and feature extraction. The main objective of feature selection is to select the most informative subset of the existing features without transformation of the matrix entries themselves. It is the process of finding the most relevant variables for a predictive model. The goal of feature extraction is to transform the existing features into lower dimensional space which may result in modification of the matrix entries themselves. It will create a subset of new features by combining existing features. Example of feature extraction method is the famous principal component analysis (PCA). Also, constructions based on the JL lemma [35] are considered to be of the feature extraction type, and classification and clustering have been used to show [36] that such constructions preserve high dimensional information in the reduced dataset.

With the aim of improving recommendation performance, [37] proposes a new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques. In [38], authors construct a feature profile of a user based on both collaborative and content features. Latent Semantic Indexing (LSI) is employed to extract the main attributes of a user and deliver suggestions based on this reduced dimensional profile. The authors [39] state that unsupervised learning "auto associates information from the input with an intrinsic reduction of data dimensionality or total amount of input data". A

clustering technique, the SVD, is one of the effective methods for dimensionality reduction.

Dimensionality reduction is usually employed as a means to overcome the so called curse of dimensionality – the technical and practical problems arising in high dimensional data, i.e. data in which items have a large number of features. DR techniques can be broadly classified to feature selection and feature extraction. The former reduces the number of features by selecting a small subset of features that are the most significance. By contrast feature extraction does not select from the original features but, rather, aims at fabricating a much smaller set of new features that represent the data within an acceptable range of accuracy. Both DR techniques could be represented by a transformation from the original feature space to a lower dimensional one.

If we denote the dimensionality of the original space by $N$ and the reduced space by $M$, and if the features are all real-valued numbers, a general DR transformation is a function $f : R^N \to R^M$ in which $M < N$. If the transformation is linear, $f$ could be represented by a matrix $A_{N\times M}$ that maps any data point $x_{1\times N}$ to another data point with $y_{1\times M} = xA$ with lower dimensionality. Correspondingly the design matrices are related via the relation $Y_{p\times M} = X_{P\times N}A$. A classic example of $A$ is one which finds a few directions of data which has most variance on and is called principle component analysis (PCA) and could be found by eigendecomposition of $X^tX$ or singular value decomposition (SVD) of $X$.

Both eigendecomposition and SVD are instances of matrix factorization (henceforth MF), which as the name suggests is to factorize a matrix as a product of several other matrices. It can be seen that there is a close relationship between DR and MF. Indeed any factorization of $X_{p\times N}$ in the form of $X =$

$A_{P\times M}B_{M\times M}C_{M\times N}$ could be thought of as an "original" lower-dimensional $A_{P\times M}$ being transformed to a higher-dimensional $X$. DR techniques are usually applied as a preprocessing step before classification, clustering or regression. Those being the classical applications, DR also finds a novel application in the collaborative information filtering problem.

Collaborative filtering (CF) is based on the behavior of a group of users to suggest a recommendation to others. In other words, CF is a method of identifying the similar clients and recommending what the common clients prefer [40]. The problem is essentially the estimation of missing feature values in the data which could be taken as an extension to the regression problems where in the latter case the values of predictor features are known and target values need to be estimated. Indeed the MF method solves the estimation of missing values in one step: factorize the design matrix according to an optimization criteria, multiply the resulting matrices and that is all! The missing values are right there in the multiplication results.

### 2.4.2 SVD

SVD is a matrix factorization method used to generate low rank approximations [41]. Given a $a \times b$ matrix $A$, with $a \geq b$ and rank $k$, the singular value decomposition, SVD (A), is defined as $SVD(A) = U \times \Sigma \times V^T$ where $U$, $\Sigma$ and $V$ are of dimensions $a \times a$, $a \times b$ and $b \times b$, respectively. $U$ and $V$ are two orthogonal matrices. The first $k$ columns of $U$ and $V$ represent the orthogonal eigenvectors associated with the $k$ nonzero eigenvalues of $AA^T$ and $A^TA$, respectively. In other words, the $k$ columns of U corresponding to the nonzero singular values cover the column vector, and the $k$ columns of $V$ cover the row vector of the matrix $A$. $U$ and $V$ are known as the left and the right singular vectors, respec-

tively. Matrix $\Sigma$ is called a diagonal matrix (also known as singular matrix), which has only $k$ nonzero entries. The diagonal entries ( $\sigma_1$, $\sigma_2$, $\sigma_3$, ..., $\sigma_k$ ) of $\Sigma$ have the property that $\sigma_k > 0$ and $\sigma_1 > \sigma_2 > \sigma_3 > \ldots > \sigma_k$ .

SVD gives the best low-rank linear approximation of the original matrix which is an essential property that makes it particularly attractive for our application. It is likely to retain only $m << k$ singular values by removal of other entries. We denote this reduced matrix $\Sigma_m$. Since the entries in $\Sigma$ are sorted i.e. $\sigma_1 > \sigma_2 > \sigma_3 > \ldots > \sigma_m$, the reduction process is performed by retaining the first m singular values. The $U$ and $V$ matrices are reduced to produce matrices $U_m$ and $V_m$, respectively. The matrix $U_m$ is produced by removing $(k - m)$ columns from the matrix $U$, and matrix $V_m$ is produced by removing $(k - m)$ rows from the matrix $V$. When we multiply these three reduced matrices, we obtain a matrix $A_m$. The reconstructed matrix $A_m = U_m \times \Sigma_m \times {V^T}_m$ is the closest approximation to the original matrix A of rank m. So, $A_m$ minimizes the Frobenius norm $||A - A_m||_F$ over all rank $m$ matrices [42].

Researchers have determined that the low-rank approximation of the original vector is better than the original vector itself due to the filtering out of the small singular values that introduce "noise" in the user-item relationship. The dimensionality reduction approach in SVD can be very valuable for the collaborative filtering process. SVD generates a set of uncorrelated eigenvectors [43]. Each user and item is represented by its corresponding eigenvector. The process of dimensionality reduction may help users who rated similar products (but not exactly the same items) to be mapped into the space spanned by the same eigenvectors [44].

### 2.4.3 Numerosity Reduction

Reducing the volume of data by choosing a smaller form of that data in order to represent the original huge data is called numerosity reduction. This method has two different strategies: parametric and non-parametric. Parametric method includes algorithms such as Log-linear models and Regression.

Log-linear models try to approximate a given dataset by estimating the likelihood of each entry in a multi-dimensional environment by a reduced subset of entries. The regression methods find a fit line between attributes and try to anticipate the other ones.

Non-parametric consists of Histograms, Sampling and Clustering. Histograms is a method that approximates the dataset in one or more attributes of a relation by grouping attribute values into buckets and approximating true attribute values and their frequencies in the dataset based on summary statistics (can include mean, mode, median, standard deviation, maximum value, minimum value, etc.) maintained in each bucket [45].

Having a random sample from the dataset which has almost the same features as compared to the original dataset is the goal of a sampling method. Clustering is the method of grouping similar elements of data into a cluster which is then a representative of that group of data.

### 2.4.4 Data Compression

Data compression is the method of encoding data in order to reduce its volume. Applying transformation is a common process in order to compress data and reduce representation of the original dataset. This method can be done in both lossy compression or lossless compression, and can be applied on both dimensionality reduction and numerosity reduction strategies. Lossless compression

guarantees that the transformed data would be the same as the original one after the decoding process, which tends to rebuild the dataset, whereas lossy compression influences the data and leads to partial loss of information. The amount of loss can be defined by a preset threshold. This method is widely used in video and image comprehension and the compressed data has only a slight difference with the original dataset after decompression.

In this chapter we have tied together the ideas of recommender system and NeuroEvolution of Augmented Topologies (NEAT). NEAT expands upon previous NeuroEvolution techniques by generating one neural network topology after another each represented as an individual upon which genetic operations can be performed to select the fittest network topology to solve a given recommendation problem. A variety of artificial intelligence solutions other than neural network based ones for recommendation systems and a variety of recommendation systems were surveyed. Finally, the most popular recommendation system, that based on collaborative filtering, and the NEAT approach, chosen from among many AI techniques, were selected for experimentation with the goal of providing a recommendation system that performs according to benchmark standards. According to the evolutionary nature of NEAT, which has the capability to traverse the search space to find a better solution, even by spending a relatively longer execution time.

# Chapter 3

# Methodology

Refer to Diagram 3.1 to an overview of all the processes involved in our proposed recommendation algorithm.

The concept behind NeuroEvolution (NE) is to evolve, with a Genetic Algorithm (GA), the topological structure used in a Neural Network, instead of using a fixed structure. NEAT is an instance of NE and a powerful technique is its ability to overcome the problems of the previous NE based systems. See Chapter 2 for a list of such problems. Our inquiry includes examination of the different parts of NEAT to show how such problems are solved by a NEAT implementation.

NEAT is adapted for recommendation, where it is used to learn a network topology that will be later used to produce recommendations. A small dataset (ML100K) was used to test and debug the implementation and then neural network approaches were used as baselines to allow a comparison to see if this approach is actually better than the previous ones.

## 3.1 Training

A common way to train a model in machine learning and statistics is to divide data into two subsets of training set and test set, in order to fit the model in training set and make prediction in test data. So it is likely for both *underfitting* or *overfitting* to happen.

*overfitting* happens when the trained model is very well trained and fits closely to the training set. That means that when applying the model on new data (say the test data), the model loses its accuracy and cannot be generalized. This event usually happens in complex systems which have many features and variables as compared to the size of the dataset. In fact, the noise will have been considered as features and the main relation between data and features would have been eliminated.

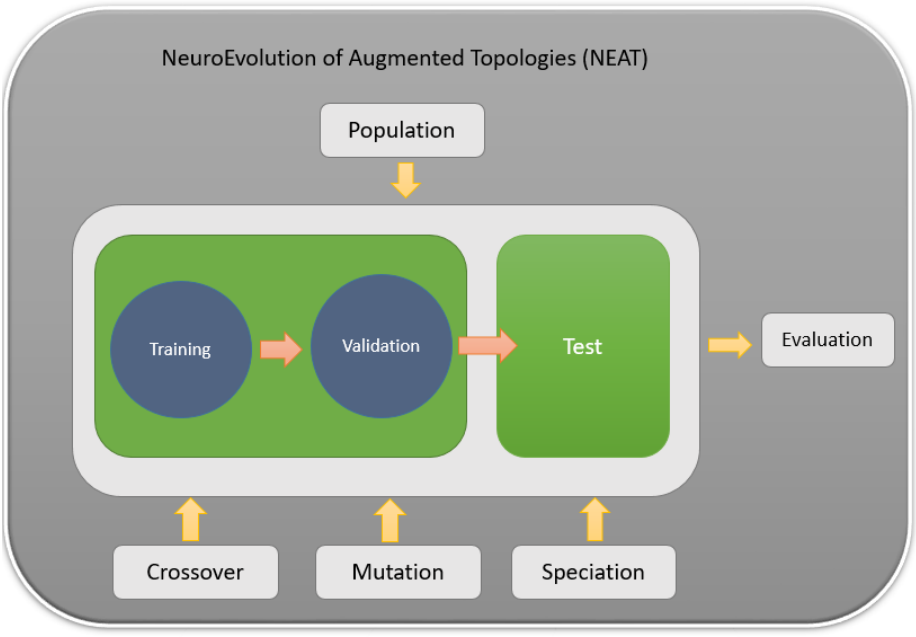


Figure 3.1: Application of NEAT on Recommendation Systems

On the other hand, *underfitting* happens when the trained data could not discover the trends among the dataset and usually occurs when the amount of features in the dataset are quite low in comparison with the size of the dataset, in other words, the model is very simple. The other case that leads to underfitting occurs when we model the training set in a lower dimension than the real model. For example, we fit a linear model (such as linear regression) to data that is not linear. In the real world, overfitting occurs more frequently than underfitting and nevertheless, we aim to prevent both of them.

## 3.2 Cross Validation

Cross validation was applied to attain a better trained model. Here we explain the cross validation concept that was used to train and test the model in order to avoid underfitting and overfitting.

In some cases, when splitting a dataset to train and test, if the dataset is not random, the splitting would result in overfitting. In order to avoid that situation, we split our data into k subsets, and train on k-1 one of those subsets (also called folds). What we do is to hold the last subset for the test. We are able to do that for each of the subsets. This process is called K-fold cross validation method 3.2. Then we calculate the average of the model against each of the folds and finalize the model. Now the system is ready to be applied on the test set.

The dataset was divided into five (in a portion of 80 percent train set and 20 percent test set) subsets which do not have any dependency or intersection with each other. The split of train/test and K-fold cross validation operation was performed using the *Sklearn* python module. *Pandas* module was also employed to load and analyze the data.

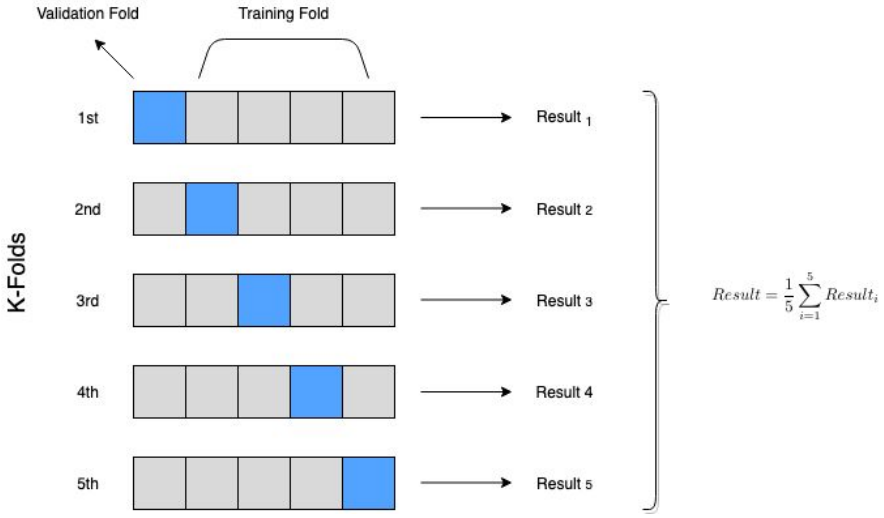


Figure 3.2: 5-Fold Cross Validation

# Chapter 4

# Implementation

A NeuroEvolution approach called NEAT was used in order to reach a better structure for a neural network based Collaborative Filtering system. By describing the different parts of NEAT, we will show how a neuroevolutionary recommender system was created

## 4.1 Choices of Dataset

The following detail about the structure of data was extracted by studying the content of ReadMe file of MovieLens database:

- *Ratings:* includes the ID of movies and users (MovieID and UserID) and also the rating that users gave to the movie using 5 star scale with 0.5 star increments. For more accuracy, the time information of ratings has been inserted in Epoch time.
- *Movies:* the detailed information about the movies including ID (MovieID), title and genre.

- *Tags:* which consist of MovieID, UserID and the tags that user has mentioned for each movie and the Epoch time of it.

- *Links:*contains MovieID and the links to IMDB site (IMDBID) and also The Movie DB (TMDBID).

Data in this data set are all in comma-separated value (CSV) format and the first row of each file contains the header of columns. If there exist column entries that include the character "," then the comma is enclosed by a double quote (") notion. All files are encoded as UTF-8 encoding.

A variety of datasets in different sizes were developed by MovieLens group. Here are the datasets dedicated to development and education purpose of which the smaller size was chosen for our experimentation.

#### 4.1.0.1 Small

A small-size dataset (ML100K) which contains 100K (100836) ratings and 3683 tag applications across 9742 movies. These data were created by 610 users. This dataset is recommended for development and education purposes and, hence, suitable for what is done in this research. A restriction of at least 20 movies rated by a single user was applied. No demographic information was included.

#### 4.1.0.2 Full

This dataset has about 27 million (27753444) ratings and 1108997 tag applications for 58098 movies. These data were created by 283228 users.

## 4.2 Tools

To implement this project, *python* version 3.0 was used and to install the python packages, *python package installer 3.0 (pip3)* was employed. The main Integrated development environment (IDE) was *PyCharm* version 2018.3.5 . The most important packages that were used are listed below and described:

*NumPy:* the basic package for performing scientific computing in Python platform. A N-dimensional array object and math functions, useful linear algebra, such as Fourier Transform, and random number capabilities are some of the characteristics of NumPy package. Besides, NumPy can also be used as an efficient multi-dimensional container of generic data and is compatible with different database connections.

*torch:* a machine learning library that was used specifically for technical computing.

*Sklearn:* a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means. This tool is applied to calculate mean absolute error and sum of squared error in this research.

*Multiprocessing:* this package allows the processes to be run simultaneously. Consequently, the multiprocessing module allows the programmer to employ multiple processors. It helps the implementation to take place at a higher speed.

*Numba:* Numba renders Python functions to optimized machine code at runtime. The optimized numerical methods in Python can approach the speeds of the C programming language which is known for its fast operation. Numba was used in implementation of FunkSVD method which will be discussed in detail in the next section.

## 4.3 Simon Funk's algorithm for singular value decomposition

FunkSVD [67] is a Python version 3 library implementing a version of the famous SVD algorithm popularized by Simon Funk during the Netflix Prize [68] contest. The repository available on GitHub [69] was used in order to implement FunkSVD. The package can be installed using the pip install command. A quick snapshot of the code from FunkSVD implementation to run an experiment and calculate mean absolute error has been shown in listing 4.1.

```
import pandas as pd
import numpy as np

from funk_svd.dataset import fetch_ml_ratings
from funk_svd import SVD

from sklearn.metrics import mean_absolute_error

df = fetch_ml_ratings(variant='100k')

train = df.sample(frac=0.8, random_state=7)
val = df.drop(train.index.tolist()).sample(frac=0.5, random_state
    =8)
test = df.drop(train.index.tolist()).drop(val.index.tolist())

svd = SVD(learning_rate=0.001, regularization=0.005, n_epochs=100,
n_factors=15, min_rating=1, max_rating=5)

svd.fit(X=train, X_val=val, early_stopping=True, shuffle=False)

pred = svd.predict(test)
mae = mean_absolute_error(test["rating"], pred)
```

```
22
23 print(f'Test MAE: {mae:.2f}')
```

Listing 4.1: A code to execute FunkSVD to calculate mean absolute error

## 4.4 NEAT

The module NEAT consists of the basic elements as follows:

*activations:* which denotes the activation function used in FeedForward function to calculate the fitness of a genome.

*crossover:* performs the NEAT crossover function. For each gene in the best parent (the parent with higher fitness value, or if equal, the longer parent in size of gene), and matching up executed regarding innovation number.

*mutation:* based on a randomly generated number in comparison with a predefined `CONNECTION_MUTATION_RATE` in config file (which is preset as 0.8), and based on a preset perturbation (`CONNECTION_PERTURBATION_RATE` with the default value of 0.9), a random value between -1 and 1 is added to the connection weight. The code related to mutation can be found in appendix B

*species:* responsible for the operations like calculating stagnation of species within a generation or measurement of distances between two genomes based on number of excess genes, number of disjoint genes and also the average weight difference.

*population:* first initiates the population according to `POPULATION_SIZE` with default value of 10000 and speciation occurs. Then depending on `NUMBER_OF_GENERATIONS` in config file (which is 1500 by default), the fitness of all genomes of the given generation is calculated, the best genome of each generation is marked and the related network will be drawn. Meanwhile, the stagnation of species of each

generation is calculated and if it meets a threshold, it will either be removed or maintained in the population.

*visualize:* to draw the network using python graph drawing capability (graphviz). The settings for graphviz that were used in drawing the network diagrams in this thesis are described as follows:

node attributes:

- shape: circle
- font size: 9
- height: 0.2
- width: 0.2

connection (edge) attributes:

- style: 'solid' if the connection is enabled else 'dotted'
- color: 'green' if the edge weight is greater than 0, else 'red'
- width: 0.1 + abs(float(`connection_weight` / 5.0)))

Below is a sample of a neural network drawn with the visualize module, which is related to the best genome generated in 9th generation of results (Figure 4.1). More samples can be found in the Appendix A.

The green weights denote the edges with positive weights as compared to red edges which have negative weight. The width (thickness) of edges depends on the absolute value of the correspondent weight. Finally, the solid edge belongs to an enabled connection whereas a dotted edge says that the connection is disabled. The explanation of the different kinds of connections were reviewed in Section 2.3.1

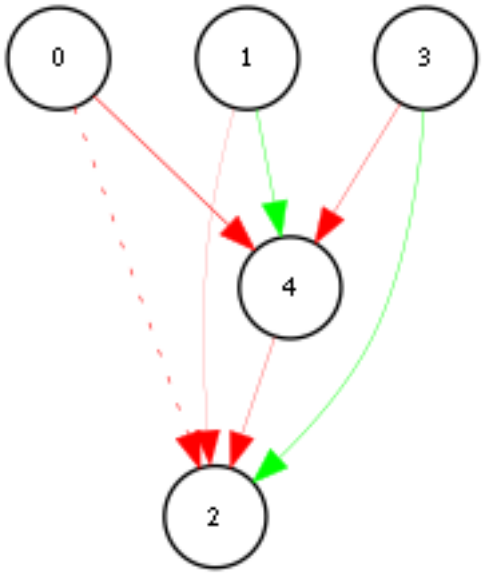


Figure 4.1: The best solution of generation No. 9

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.3891871929168701 style=dotted] |
| $1 \rightarrow 2$ | [color=red penwidth=0.10447230320423842 style=solid] |
| $3 \rightarrow 2$ | [color=green penwidth=0.2884440928697586 style=solid] |
| $0 \rightarrow 4$ | [color=red penwidth=0.4012172281742096 style=solid] |
| $4 \rightarrow 2$ | [color=red penwidth=0.19657768905162812 style=solid] |
| $3 \rightarrow 4$ | [color=red penwidth=0.22929360568523408 style=solid] |
| $1 \rightarrow 4$ | [color=green penwidth=0.2786946356296539 style=solid] |

Two main sub modules were selected for use in NEAT module as follows: genotype and phenotype. *Genotype:* contains the genome class, the node gene and the connection gene. Genome class contains several functions related to the genes such as adding a node or a connection when mutation occurs, counting the number of excess genes and disjoint genes, and other methods that perform adding nodes or connections or returning a specific connection by innovation

number. Node and connection classes were used to define the attributes and methods of the nodes and connections, respectively.

## 4.5 Main module

As the program starts, settings and config files are fetched and saved in local variables, configurations such as the number of generations, minimum and maximum of hidden nodes in neural networks, and a global variable which holds the best solution of each generation and a total best solution. Loading data from dataset (see section 4.1) is the next part of the whole process. Then the maximum possible fitness of the test data is calculated using the Mean Square Error function.

Setting the number of workers (processes) which are considered to perform multiprocessing and initiating the pool of processes is one of the other important parts of program. After all, the software starts to act on the population initialized. In a loop on size of `NUMBER_OF_GENERATIONS` or even the fitness decreases less than a threshold, the fitness of every genome (neural network) are calculated, reproduction happens and based on the remaining species, the new fitness of genomes will be recalculated and mutation occurs based on a preset probability as we discussed in Section 4.4. The statistics of each generation will be printed on the output console. Figure 4.2 presented the flowchart diagram denoting the main tasks. Also, some more code implementation of the main module can be find in appendix B.

## 4.6 SHARCNET

The first few runs of the program occurred very slowly, due to the high load of computation. We used Shared Hierarchical Academic Research Computing Network (SHARCNET[70]) platform, a high performance computing consortium in Canada, in order to run the huge amount of calculations needed. Supported by governmental foundations, SHARCNET has installed $\sim$ 40000 processors and 30Pb of storage in a variety of equipment. The proposed algorithm computations are performed with 30 parallel processes using a compute node with 30 CPUs and 64 GB of RAM for 20 hours.

In this chapter, the dataset was introduced and different tools and the main methodology is discussed; also, the development modules to implement the proposed method reviewed. The next chapter presents the result of our method and compares it with FunkSVD as main benchmark and some other algorithms.

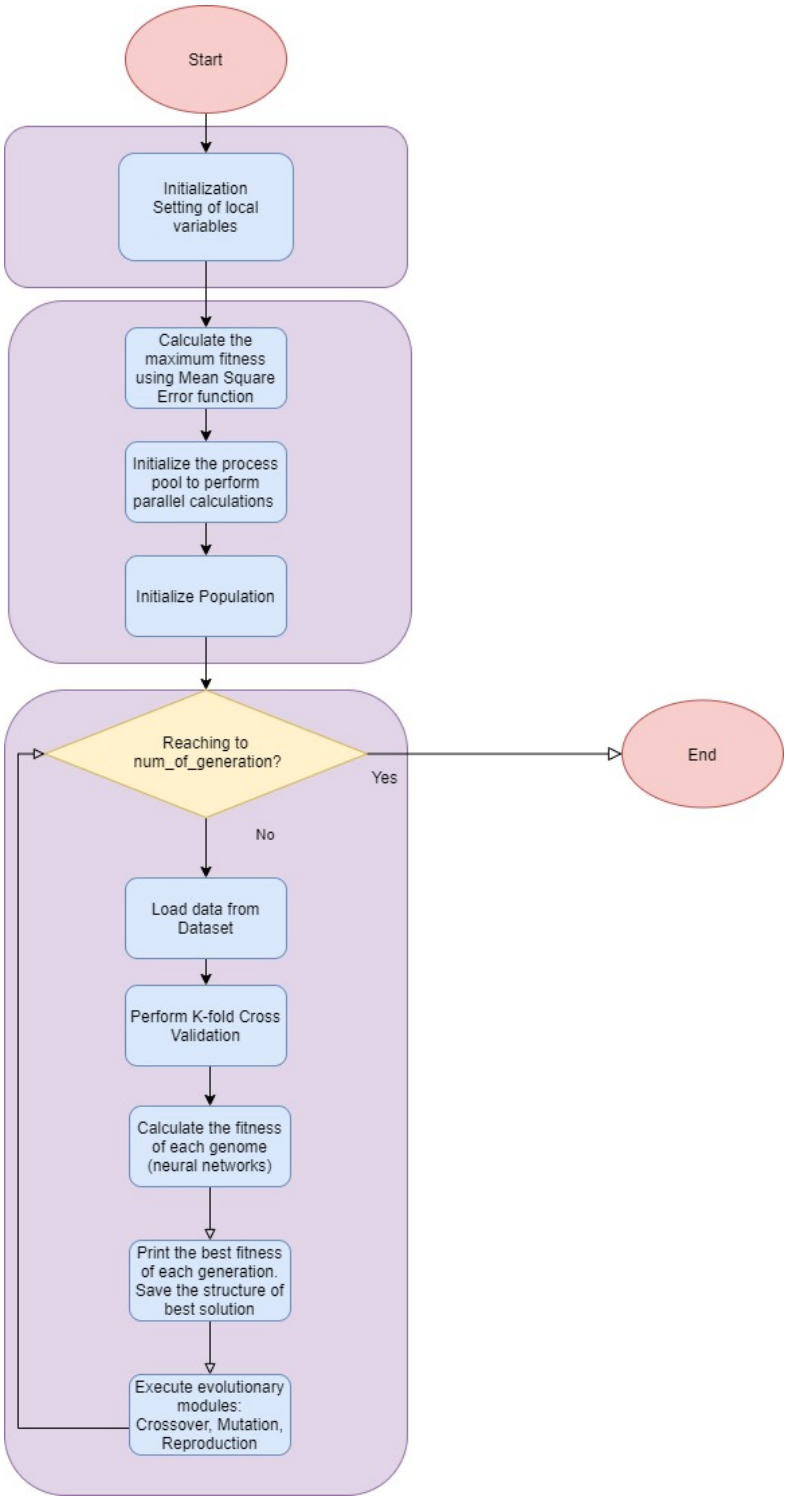


Figure 4.2: Flowchart diagram of main module

# Chapter 5

# Results and Discussion

We have employed NeuroEvolution, a subfield within artificial intelligence (AI) and machine learning (ML) that seeks to evolve neural networks through evolutionary algorithms, to help make the decisions necessary for collaborative based recommendations sytems.

## 5.1 The Fitness Function for Finding the Best Neural Network

We have evolved a neural network-based recommendation system by employing NEAT, the details of which were discussed in the introductory sections. Evolutionary algorithms need a fitness function, and it is customary to use recommendation error as fitness [71]. Indeed fitness has an inverse relation with system's error, the negation of this error being useful for natural selection. As our algorithm requires fitness to be a non-negative number, we add the error with a constant number $A$ to ensure that any reasonable recommendation sys-

tem has a positive fitness. The error function we used is sum of squared error (SSE) measured on test data:

$$SSE = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

$$fitness = max(0,\, A - SSE)$$

in which $y_i$ is the observed value for the $i$th observation and $\hat{y}_i$ is the predicted value. We defined $A$ as the maximum error of a recommendation system in which its outputs fall within the range of valid rating values, namely between zero and five (inclusive):

$$A = \sum_{t \in Test\, Set} max(t,\, 5 - t)^2$$

in which $t$ is the predicted rating of each data in the test set. The intuition behind the formula is that as ratings are between 0 and 5, the maximum error (i.e. SSE) of a recommendation system for any target rating would be $max(t,\, 5-t)^2$. This leads to the finding that the fitness of all recommendation systems would fall between 0 and $A$. For example, for 100K MovieLens dataset, $A$ would be 146324.

We compared the performance of our recommendation system with that of a recommendation system based on FunkSVD algorithm which is a standard matrix factorization method in the field. First, we explain FunkSVD in more details and then compare our method's results with the outputs of FunkSVD as a famous method in this area.

The training phase took 3 sec and the mean absolute error (MAE) of the test set is 0.75. As we discussed above, our new fitness for FunkSVD yields the

value of 146324 which is the baseline of our comparison. Now we have a quick look to see how the algorithm performs. Suppose a huge sparse matrix:

$$R = \begin{pmatrix} ? & 2 & \cdots & ? & ? \\ ? & ? & \cdots & ? & 4.5 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 3 & ? & \cdots & ? & ? \\ ? & ? & \cdots & 5 & ? \end{pmatrix}$$

Storing known ratings for a set of users and items:

$u = 1, ..., U$

$i = 1, ..., I$

Predicting the unknown ratings by factorizing the rating matrix into two smaller matrices to represent user and item characteristics as follows:

$$P = \begin{pmatrix} 0.37 & \cdots & 0.69 \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 1.08 & \cdots & 0.24 \end{pmatrix}, Q = \begin{pmatrix} 0.09 & \cdots & \cdots & \cdots & 0.46 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0.51 & \cdots & \cdots & \cdots & 0.72 \end{pmatrix}$$

We call these two matrices users and items latent factors. Then, by applying the dot product between both matrices we can reconstruct our rating matrix. The trick is that the empty values will now contain estimated ratings.

In order to get more accurate results, the global average rating as well as the user and item biases are used in addition:

$\bar{r} = \frac{1}{N} \sum_{i=1}^{N} K_i$

where $K$ stands for known ratings.

$$bu = \begin{pmatrix} 0.35 & \cdots & 0.07 \end{pmatrix}$$

$$bi = \begin{pmatrix} 0.16 & \cdots & 0.40 \end{pmatrix}$$

Then, we estimate any rating by applying:

$\hat{r}_{u,i} = \bar{r} + bu_u + bi_i + \sum_{f=1}^{F} P_{u,f} * Q_{i,f}$

The learning step consists of performing the stochastic gradient descent (SGD) algorithm where for each known rating the biases and latent factors are updated as follows:

$err = r - \hat{r}$

$bu_u = bu_u + \alpha * (err - \lambda * bu_u)$

$bi_i = bi_i + \alpha * (err - \lambda * bi_i)$

$P_{u,f} = P_{u,f} + \alpha * (err * Q_{i,f} - \lambda * P_{u,f})$

$Q_{i,f} = Q_{i,f} + \alpha * (err * P_{u,f} - \lambda * Q_{i,f})$

where $\alpha$ is the learning rate and lambda is the regularization term.

The FunkSVD's performance against various values of latent space dimensionality (k), is shown in Figure 5.1 (each point in the diagram corresponds to one training session in 100 epochs and is completed in approximately 1 second). It can be seen that the maximum fitness is achieved at $k = 8$ which is roughly equal to 137308.

Applying NEAT we came close to this result. Figure 5.2 illustrates the fitness of the best genome per each generation number.

Figure 5.3 shows the topology of the best neural network generated by NEAT, which corresponds to the output solution for the 140th generation which is roughly the same as FunkSVD result. To find out the best solutions in each generation during the time see appendix A.

The evaluation of the NEAT-based results against a matrix factorization recommender algorithm (FunkSVD) showed that our approach is sound by achieving a similar fitness function score and a neural network topology generated for its fitness on test data is approximately the same as FunkSVD.

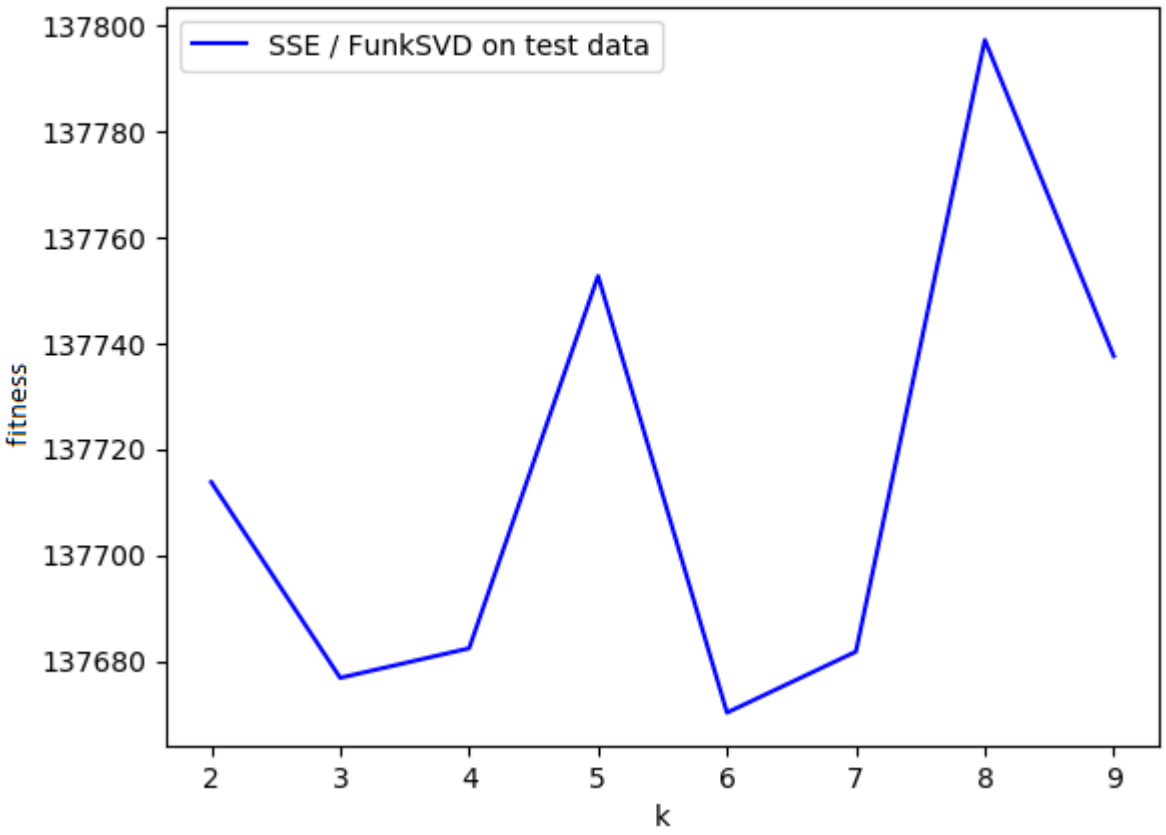


Figure 5.1: Fitness per latent space dimensionality in FunkSVD

## 5.2 More Comparisons: Recommendation Systems Ordered by Average Root Mean Square Error

Here are the average root mean square error (RMSE) and total execution time of various algorithms (with their default parameters) on a 5-fold cross-validation

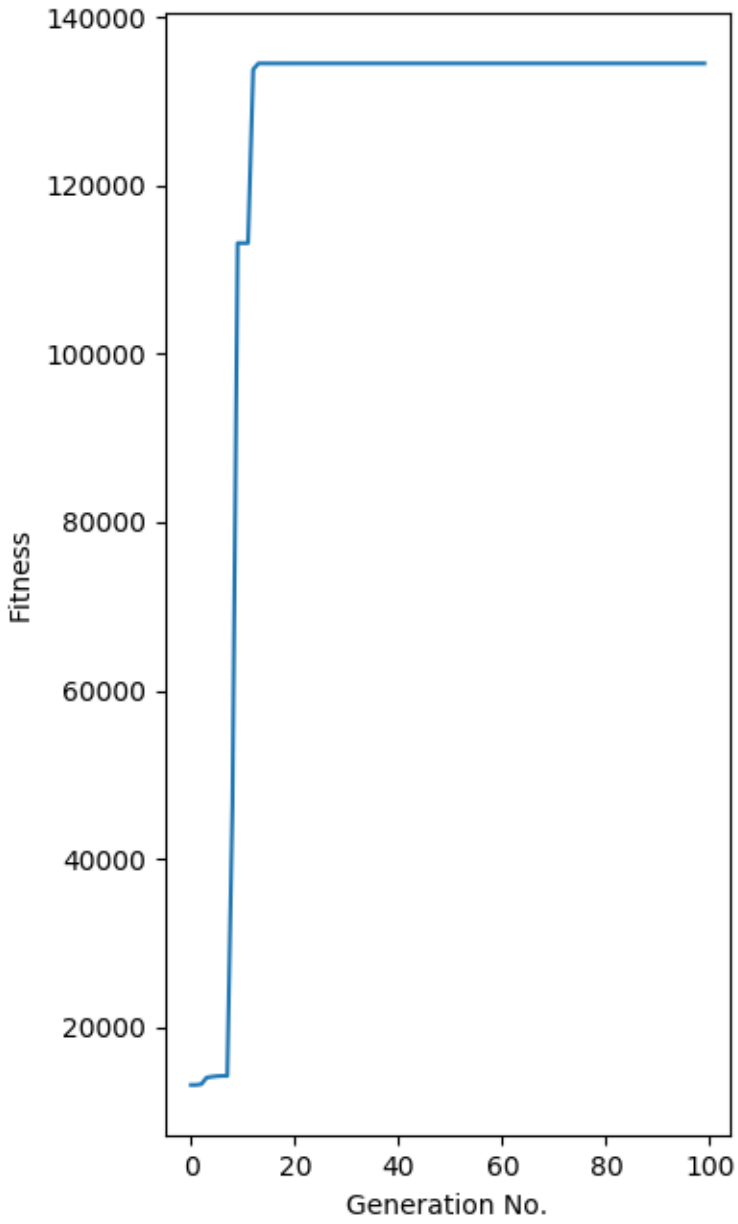


Figure 5.2: Fitness of each generation number

procedure. RMSE is a frequently used measure of the differences between values (sample or population values) predicted by an estimator (predictor) and the values observed. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

The datasets are the Movielens 100k and the folds are the same for all the algorithms. The code for generating these tables can be found in the benchmark example [72]. As $SSE$ (which we used in our fitness formula in 5.1) has the following relation with RMSE:

$$RMSE = \sqrt{\frac{SSE}{N}}$$

in which $N$ is the `POPULATION_SIZE` with value of 10000. So the RMSE for our best NN result is 0.9495. Table 5.1 compares our result with the other methods.

The SVD++ algorithm, which is an extension of FunkSVD taking into account implicit ratings, has the best RMSE. The k-NN Baseline and FunkSVD have almost the same RMSE in a relatively less execution time as compared to SVD++. Even though our method takes a huge time to be executed, corresponding to its nature of evolutionary algorithm and examining each individual in generation, the RMSE of 0.949 is a promising value which can even be better during the time. Some ideas to enhance the results are discussed in Section 6. The comparison shows us that the current proposed methodology is in the middle, not better than many but better than some. Proposed NeuroEvolutionary algorithm outperforms k-nn method, which is a basic collaborative filtering algorithm and takes into account the mean ratings of each user. Co-Clustering algorithm presented by [73], is a collaborative filtering algorithm based on co-clustering in which users and items are assigned some clusters and average rating

| **Movielens 100k** | **RMSE** | **Time** |
|---|---|---|
| SVD++ | **0.92** | 0:09:03 |
| k-NN Baseline | 0.931 | 0:00:12 |
| FunkSVD | 0.934 | 0:00:11 |
| Baseline | 0.944 | 0:00:01 |
| Slope One | 0.946 | 0:00:08 |
| Proposed Method | 0.949 | 21:23:06 |
| Centered k-NN | 0.951 | 0:00:10 |
| NMF | 0.963 | 0:00:15 |
| Co-Clustering | 0.963 | 0:00:03 |
| k-NN | 0.98 | 0:00:10 |
| Random | 1.514 | 0:00:01 |

Table 5.1: Comparison with an approach based on Root Mean Square Error-**Movielen 100K**

of a co-cluster is the average ratings of its relating users. Clusters are assigned using a straightforward optimization method, much like k-means. Proposed method yields a better performance than the mentioned clustering approach which can be interpreted as the evolutionary approach capability to traverse the search space to find a better solution, even by spending a longer execution time.

The concept behind NeuroEvolution (NE) is to evolve the topological structure used in the Neural Network with a Genetic Algorithm (GA), instead of using a fixed structure. But what makes NEAT (NeuroEvolution of Augmenting Topologies) a powerful algorithm is its ability to overcome certain problems of previous NE systems. For instance, traditional NE systems had a fixed net-

work topology or structure. In other words, the network designer chose a priori a network topology for the evolution process, and then during training the weight of connections between neurons was determined. NEAT tries to evolve both structure and weights that would lead to saving time [58]. The other concern was about the performance of an evolutionary neural network which was suspected to be slow. For example a fixed-structure algorithm named Enforced Subpopulations (ESP) had the ability of solving a problem five times faster than when it starts from a random structure. However, NEAT was shown to perform 5 times faster than ESP[61].

In our system, NEAT works on the already existing (not reduced) data to construct the structure of the neural network, though the data are sparse. Commonly, a method should be applied to firstly guess the missing data, and then to build an organized structure of the neural network for the upcoming predictions made by the recommendation system. This suggestion, would be useful for "Cold Start Problem" which means that we cannot make a recommendation for new users or new movies e.g. new rows or columns added to the basic matrix. That is because we have a neural network which is currently doing its best for the current dataset (users and items).

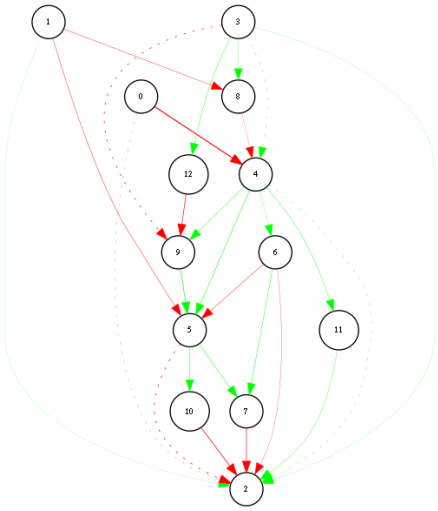


Figure 5.3: The best solution of generation No. 140

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=green penwidth=0.11510254107415677 style=solid] |
| 0 → 4 | [color=red penwidth=0.8551668882369995 style=solid] |
| 4 → 2 | [color=green penwidth=0.24948569238185883 style=dotted] |
| 3 → 4 | [color=green penwidth=0.27409485578536985 style=dotted] |
| 1 → 5 | [color=red penwidth=0.32156639397144315 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=dotted] |
| 4 → 5 | [color=green penwidth=0.39555067420005796 style=solid] |
| 4 → 6 | [color=green penwidth=0.20035351663827897 style=solid] |
| 6 → 2 | [color=red penwidth=0.19354103356599808 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3574644088745117 style=solid] |
| 3 → 9 | [color=red penwidth=0.494989013671875 style=dotted] |
| 1 → 8 | [color=red penwidth=0.2873188316822052 style=solid] |
| 5 → 10 | [color=green penwidth=0.3000000029802322 style=solid] |
| 10 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |
| 4 → 11 | [color=green penwidth=0.3000000029802322 style=solid] |
| 11 → 2 | [color=green penwidth=0.24948569238185883 style=solid] |
| 3 → 12 | [color=green penwidth=0.3000000029802322 style=solid] |
| 12 → 9 | [color=red penwidth=0.494989013671875 style=solid] |
| 6 → 5 | [color=red penwidth=0.3114542722702026 style=solid] |

# Chapter 6

# Conclusion and Future Works

The success of a neural network-based recommendation system depends on finding a network architecture to fit the task. The idea behind neuroevolution is to trigger an evolutionary process similar to the one that produced the human brain. In other words, neuroevolution seeks to develop the means of evolving neural networks through evolutionary algorithms. The Fixed Topology NeuroEvolution approach with its unchanging architecture is considered outdated when compared with an automated method for optimizing neural network structures.

A neuro-evolutionary algorithm has been developed in order to enhance the structure of a neural network to yield a more accurate recommendation system based on some of the existing benchmarks for measuring recommendation capability. The genetic algorithm ensures us to gain a more efficient topology

produced by different generations through each step of the evolution process. Results show that this method performs as well as the benchmark algorithm.

In near future, we pursue the current experiment with the following idea: After meeting a benchmark fitness by plain NEAT, a bunch of artificially-build new genomes (which can be produced by the existing neural network topologies proposed in Section 2.2), may be imported to the population. We expect to gain an even better fitness after a mixture of these populations.

NEAT utilizes genetic algorithms for alteration of both the weighting parameters and the network structures. By extending the proposed method to additionally evolve parameters, it is expected to achieve even better results. Given the available computing power such as that of Canada's SHARCNET, evolution of network topologies is a promising approach to constructing recommendation systems in the future.

One of the main problems we had in the current work was that the evolutionary process was quite time-consuming, even by the powerful platform of SHARCNET. One of the upcoming efforts would be to lower the processing time by applying a data reduction method before feeding the data to the Neural Network-based NEAT process. Johnson–Lindenstrauss lemma (JL lemma) and singular value decomposition (SVD) are some serious candidates by means of which to apply data reduction during the data preparation phase.

# Bibliography

[1] Miquel Montaner, Beatriz López, and Josep Lluís de la Rosa. "A Taxonomy of Recommender Agents on the Internet". en. In: *Artificial Intelligence Review* 19.4 (June 2003), pp. 285–330. ISSN: 1573-7462. DOI: 10.1023/A:1022850703159. URL: https://doi.org/10.1023/A:1022850703159.

[2] *AI Recommender Systems*. en-US. Mar. 2018. URL: https://silo.ai/ai-recommender-systems/.

[3] Iateilang Ryngksai and L Chameikho. "Recommender Systems: Types of Filtering Techniques". en. In: *International Journal of Engineering Research* 3.11 (), p. 4.

[4] J. Bobadilla et al. "Recommender systems survey". en. In: *Knowledge-Based Systems* 46 (July 2013), pp. 109–132. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2013.03.012. URL: http://www.sciencedirect.com/science/article/pii/S0950705113001044.

[5] Badrul Munir Sarwar et al. "Item-based collaborative filtering recommendation algorithms." In: *Www* 1 (2001), pp. 285–295.

[6] Songjie Gong. "A collaborative filtering recommendation algorithm based on user clustering and item clustering." In: *JSW* 5.7 (2010), pp. 745–752.

[7] Kyoung-jae Kim and Hyunchul Ahn. “A recommender system using GA K-means clustering in an online shopping market”. en. In: *Expert Systems with Applications* 34.2 (Feb. 2008), pp. 1200–1209. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2006.12.025. URL: http://www.sciencedirect.com/science/article/pii/S0957417406004076.

[8] Xindong Wu et al. “Top 10 algorithms in data mining”. en. In: *Knowledge and Information Systems* 14.1 (Jan. 2008), pp. 1–37. ISSN: 0219-3116. DOI: 10.1007/s10115-007-0114-2. URL: https://doi.org/10.1007/s10115-007-0114-2.

[9] D. A. Adeniyi, Z. Wei, and Y. Yongquan. “Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method”. en. In: *Applied Computing and Informatics* 12.1 (Jan. 2016), pp. 90–108. ISSN: 2210-8327. DOI: 10.1016/j.aci.2014.10.001. URL: http://www.sciencedirect.com/science/article/pii/S221083271400026X.

[10] Rupali Patil, Shyam Deshmukh, and K Rajeswari. “Analysis of SimpleK-Means with Multiple Dimensions using WEKA”. In: *International Journal of Computer Applications* 110.1 (2015).

[11] Shehroz S. Khan and Amir Ahmad. “Cluster center initialization algorithm for K-means clustering”. en. In: *Pattern Recognition Letters* 25.11 (Aug. 2004), pp. 1293–1302. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2004.04.007. URL: http://www.sciencedirect.com/science/article/pii/S0167865504000996.

[12] Hafed Zarzour et al. “An Improved Collaborative Filtering Recommendation Algorithm for Big Data”. en. In: IFIP Advances in Information

and Communication Technology (2018). Ed. by Abdelmalek Amine et al., pp. 660–668. DOI: 10.1007/978-3-319-89743-1_56.

[13] Francesco Ricci et al., eds. *Recommender Systems Handbook*. en. Springer US, 2011. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3. URL: https://www.springer.com/gp/book/9780387858203.

[14] Yoon Ho Cho, Jae Kyeong Kim, and Soung Hie Kim. "A personalized recommender system based on web usage mining and decision tree induction". en. In: *Expert Systems with Applications* 23.3 (Oct. 2002), pp. 329–342. ISSN: 0957-4174. DOI: 10.1016/S0957-4174(02)00052-0. URL: http://www.sciencedirect.com/science/article/pii/S0957417402000520.

[15] Amir Gershman et al. "A decision tree based recommender system". In: *10th International Conferenceon Innovative Internet Community Systems (I2CS)–Jubilee Edition 2010–* (2010).

[16] Mustansar Ghazanfar and Adam Prugel-Bennett. "An improved switching hybrid recommender system using Naive Bayes classifier and collaborative filtering". In: (2010).

[17] Sutheera Puntheeranurak and Pongpan Pitakpaisarnsin. "Time-aware recommender system using Naïve Bayes classifier weighting technique". In: (2013).

[18] Meghna Khatri. "A survey of naïve bayesian algorithms for similarity in recommendation systems". In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2.5 (2012).

[19] Anand Shanker Tewari, Tasif Sultan Ansari, and Asim Gopal Barman. "Opinion based book recommendation using Naive Bayes classifier". In: (2014), pp. 139–144.

[20] Sutheera Puntheeranurak and Supitchaya Sanprasert. “Hybrid naive bayes classifier weighting and singular value decomposition technique for recommender system”. In: (2011), pp. 473–476.

[21] Cosimo Birtolo and Davide Ronca. “Advances in Clustering Collaborative Filtering by means of Fuzzy C-means and trust”. en. In: *Expert Systems with Applications* 40.17 (Dec. 2013), pp. 6997–7009. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2013.06.022. URL: http://www.sciencedirect.com/science/article/pii/S0957417413004028.

[22] Hamidreza Koohi and Kourosh Kiani. “User based Collaborative Filtering using fuzzy C-means”. In: *Measurement* 91 (Sept. 2016), pp. 134–139. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2016.05.058. URL: http://www.sciencedirect.com/science/article/pii/S0263224116302159.

[23] Raciel Yera and Luis Martínez. “Fuzzy Tools in Recommender Systems: A Survey”. en. In: *International Journal of Computational Intelligence Systems* 10.1 (Jan. 2017), pp. 776–803. ISSN: 1875-6883. DOI: 10.2991/ijcis.2017.10.1.52. URL: https://www.atlantis-press.com/journals/ijcis/25872436.

[24] Zdzisław Pawlak. “Rough sets”. en. In: *International Journal of Computer & Information Sciences* 11.5 (Oct. 1982), pp. 341–356. ISSN: 1573-7640. DOI: 10.1007/BF01001956. URL: https://doi.org/10.1007/BF01001956.

[25] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. en. Theory and Decision Library D: Springer Netherlands, 1991. ISBN: 978-0-7923-1472-1. DOI: 10.1007/978-94-011-3534-4. URL: https://www.springer.com/gp/book/9780792314721.

[26] DanEr Chen, YuLong Ying, and SongJie Gong. "A collaborative filtering algorithm based on rough set and fuzzy clustering". In: *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 1. IEEE. 2008, pp. 17–20.

[27] H. Huang, H. Yang, and E. H. Lu. "A fuzzy-rough set based ontology for hybrid recommendation". In: *2015 IEEE International Conference on Consumer Electronics - Taiwan*. June 2015, pp. 358–359. DOI: 10.1109/ICCE-TW.2015.7216942.

[28] Liming Yao et al. "Synergies Between Association Rules and Collaborative Filtering in Recommender System: An Application to Auto Industry". en. In: *Data Science and Digital Business*. Ed. by Fausto Pedro García Márquez and Benjamin Lev. Cham: Springer International Publishing, 2019, pp. 65–80. ISBN: 978-3-319-95651-0. DOI: 10.1007/978-3-319-95651-0_5. URL: https://doi.org/10.1007/978-3-319-95651-0_5.

[29] Ling Zhou and Stephen Yau. "Efficient association rule mining among both frequent and infrequent items". en. In: *Computers & Mathematics with Applications* 54.6 (Sept. 2007), pp. 737–749. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2007.02.010. URL: http://www.sciencedirect.com/science/article/pii/S0898122107002519 (visited on 05/12/2020).

[30] Daniel Mican and Nicolae Tomai. "Association-Rules-Based Recommender System for Personalization in Adaptive Web-Based Applications". en. In: Lecture Notes in Computer Science (2010). Ed. by Florian Daniel and Federico Michele Facca, pp. 85–90. DOI: 10.1007/978-3-642-16985-4_8.

[31] Choonho Kim and Juntae Kim. "A recommendation algorithm using multi-level association rules". In: *Proceedings IEEE/WIC International Confer-*

*ence on Web Intelligence (WI 2003)*. Oct. 2003, pp. 524–527. DOI: 10.1109/WI.2003.1241257.

[32] Behzad Soleimani Neysiani et al. "Improve Performance of Association Rule-Based Collaborative Filtering Recommendation Systems using Genetic Algorithm". en. In: *International Journal of Information Technology and Computer Science* 11.2 (Feb. 2019), pp. 48–55. ISSN: 20749007, 20749015. DOI: 10.5815/ijitcs.2019.02.06. URL: http://www.mecs-press.org/ijitcs/ijitcs-v11-n2/v11n2-6.html (visited on 01/26/2020).

[33] Jianqing Fan, Fang Han, and Han Liu. "Challenges of big data analysis". In: *National science review* 1.2 (2014), pp. 293–314.

[34] Laurens van der Maaten, Eric O. Postma, and Jaap van den Herik. "Dimensionality Reduction: A Comparative Review". In: 2009.

[35] William B Johnson and Joram Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space". In: *Contemporary mathematics* 26.189-206 (1984), p. 1.

[36] John Fedoruk et al. "Dimensionality reduction via the Johnson–Lindenstrauss Lemma: theoretical and empirical bounds on embedding dimension". In: *The Journal of Supercomputing* 74.8 (2018), pp. 3933–3949.

[37] Hafed Zarzour et al. "A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques". In: Apr. 2018, pp. 102–106. DOI: 10.1109/IACS.2018.8355449.

[38] Panagiotis Symeonidis. "Content-based Dimensionality Reduction for Recommender Systems". en. In: *Data Analysis, Machine Learning and Applications*. Ed. by Christine Preisach et al. Studies in Classification, Data

Analysis, and Knowledge Organization. Springer Berlin Heidelberg, 2008, pp. 619–626. ISBN: 978-3-540-78246-9.

[39] Ke-Lin Du and Madisetti NS Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013.

[40] Nitika Kadam and Shraddha Kumar. "A review of Content and Collaborative filtering approaches on Movielens Data". en. In: 03.03 (), p. 6.

[41] Badrul M Sarwar et al. "Applying knowledge from KDD to recommender systems". In: *University of Minnesota, Minneapolis* 1.612 (1999), pp. 625–4002.

[42] Robert M Bell and Yehuda Koren. "Lessons from the Netflix prize challenge". In: *Acm Sigkdd Explorations Newsletter* 9.2 (2007), pp. 75–79.

[43] Yehuda Koren. "Factorization meets the neighborhood: a multifaceted collaborative filtering model". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 426–434.

[44] Rajeev Kumar, BK Verma, and Shyam Sunder Rastogi. "Social popularity based SVD++ recommender system". In: *International Journal of Computer Applications* 87.14 (2014).

[45] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. "Data cube approximation and histograms via wavelets". In: *Proceedings of the seventh international conference on Information and knowledge management*. CIKM '98. Bethesda, Maryland, USA: Association for Computing Machinery, Nov. 1998, pp. 96–104. ISBN: 978-1-58113-061-4. DOI: 10.1145/288627.288645. URL: https://doi.org/10.1145/288627.288645.

[46] Dalia Sami Jasim. "Data mining approach and its application to dresses sales recommendation". In: *Research Gate* (2015).

[47] Maria Nadia Postorino and Giuseppe ML Sarne. "A neural network hybrid recommender system". In: *Proceedings of the 2011 conference on neural Nets WIRN10*. 2011, pp. 180–187.

[48] Anant Gupta and B. K. Tripathy. "A generic hybrid recommender system based on neural networks". In: *2014 IEEE International Advance Computing Conference (IACC)*. ISSN: null. Feb. 2014, pp. 1248–1252. DOI: 10.1109/IAdCC.2014.6779506.

[49] M. K. Kavitha Devi et al. "Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems". In: *2010 IEEE International Conference on Computational Intelligence and Computing Research*. ISSN: null. Dec. 2010, pp. 1–4. DOI: 10.1109/ICCIC.2010.5705777.

[50] Arthur J. Lin, Chien-Lung Hsu, and Eldon Y. Li. "Improving the effectiveness of experiential decisions by recommendation systems". en. In: *Expert Systems with Applications* 41.10 (Aug. 2014), pp. 4904–4914. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2014.01.035. URL: http://www.sciencedirect.com/science/article/pii/S0957417414000591.

[51] R. Mu. "A Survey of Recommender Systems Based on Deep Learning". In: *IEEE Access* 6 (2018), pp. 69009–69022. DOI: 10.1109/ACCESS.2018.2880197.

[52] Suvash Sedhain et al. "AutoRec: Autoencoders Meet Collaborative Filtering". en. In: *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*. Florence, Italy: ACM Press, 2015,

pp. 111–112. ISBN: 978-1-4503-3473-0. DOI: 10.1145/2740908.2742726. URL: http://dl.acm.org/citation.cfm?doid=2740908.2742726.

[53] Florian Strub and Jérémie Mary. "Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs". en. In: (), p. 9.

[54] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering". en. In: *Proceedings of the 24th international conference on Machine learning - ICML '07*. Corvalis, Oregon: ACM Press, 2007, pp. 791–798. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273596. URL: http://portal.acm.org/citation.cfm?doid=1273496.1273596.

[55] Caihua Wu et al. "Recurrent neural network based recommendation for time heterogeneous feedback". en. In: *Knowledge-Based Systems* 109 (Oct. 2016), pp. 90–103. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2016.06.028. URL: http://www.sciencedirect.com/science/article/pii/S095070511630199X.

[56] Jun Wang et al. "IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. event-place: Shinjuku, Tokyo, Japan. New York, NY, USA: ACM, 2017, pp. 515–524. ISBN: 978-1-4503-5022-8. DOI: 10.1145/3077136.3080786. URL: http://doi.acm.org/10.1145/3077136.3080786.

[57] Yilmaz Ar and Erkan Bostanci. "A genetic algorithm solution to the collaborative filtering problem". en. In: *Expert Systems with Applications* 61 (Nov. 2016), pp. 122–128. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2016.

05.021. URL: http://www.sciencedirect.com/science/article/pii/S0957417416302469.

[58] Nitai B. Silva et al. "A graph-based friend recommendation system using Genetic Algorithm". In: *IEEE Congress on Evolutionary Computation*. ISSN: 1941-0026. July 2010, pp. 1–7. DOI: 10.1109/CEC.2010.5586144.

[59] Mojtaba Salehi, Mohammad Pourzaferani, and Seyed Amir Razavi. "Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model". en. In: *Egyptian Informatics Journal* 14.1 (Mar. 2013), pp. 67–78. ISSN: 1110-8665. DOI: 10.1016/j.eij.2012.12.001. URL: http://www.sciencedirect.com/science/article/pii/S1110866512000527.

[60] Linqi Gao and Congdong Li. "Hybrid Personalized Recommended Model Based on Genetic Algorithm". In: *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*. ISSN: 2161-9654. Oct. 2008, pp. 1–4. DOI: 10.1109/WiCom.2008.2152.

[61] Kenneth O. Stanley and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies". In: *Evolutionary Computation* 10.2 (2002), pp. 99–127. URL: https://doi.org/10.1162/106365602320169811.

[62] Jan Nils Ferner et al. "Combining Neuro-Evolution of Augmenting Topologies with Convolutional Neural Networks". en. In: (), p. 56.

[63] Mehmet Erkan Yuksel. "Agent-based evacuation modeling with multiple exits using NeuroEvolution of Augmenting Topologies". en. In: *Advanced Engineering Informatics* 35 (Jan. 2018), pp. 30–55. ISSN: 1474-0346. DOI: 10.1016/j.aei.2017.11.003. URL: http://www.sciencedirect.com/science/article/pii/S1474034616304281.

[64] Kenneth O. Stanley et al. "Designing neural networks through neuroevolution". en. In: *Nature Machine Intelligence* 1.1 (Jan. 2019), pp. 24–35. ISSN: 2522-5839. DOI: 10.1038/s42256-018-0006-z. URL: https://www.nature.com/articles/s42256-018-0006-z.

[65] Sylvain Cussat-Blanc, Kyle Harrington, and Jordan Pollack. "Gene Regulatory Network Evolution Through Augmenting Topologies". In: *IEEE Transactions on Evolutionary Computation* 19.6 (Dec. 2015), pp. 823–837. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2015.2396199.

[66] Son Pham et al. "Playing SNES Games With NeuroEvolution of Augmenting Topologies". In: *AAAI*. 2018.

[67] Simon Funk. *Netflix Update: Try This At Home*. 2006. URL: https://sifter.org/~simon/journal/20061211.html.

[68] *Netflix Prize*. https://www.netflixprize.com/.

[69] Geoffrey Bolmier. *A python fast implementation of the famous SVD algorithm popularized by Simon Funk during Netflix Prize*. https://github.com/gbolmier/funk-svd.

[70] *SHARCNET*. https://www.sharcnet.ca/.

[71] Jesus Bobadilla et al. "Improving collaborative filtering recommender system results and performance using genetic algorithms". en. In: *Knowledge-Based Systems* 24.8 (Dec. 2011), pp. 1310–1316. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2011.06.005. URL: http://www.sciencedirect.com/science/article/pii/S0950705111001146 (visited on 11/16/2019).

[72] Nicolas Hug. *Benchmark examples in GitHub*. https://github.com/NicolasHug/Surprisey.

[73] Thomas George. “A scalable collaborative filtering framework based on co-clustering”. In: *Fifth IEEE International Conference on Data Mining*. 2005, pp. 625–628.

# Appendix A: List of the best neural network structures of each generation in NEAT

Note: The skipped generation numbers has the same structure as their previous one.

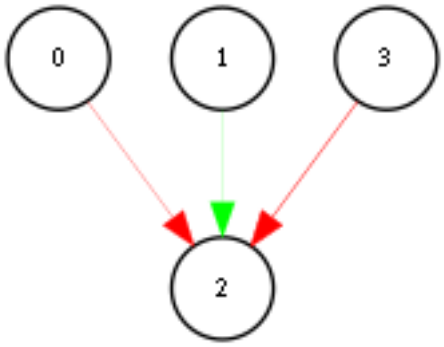


Figure 6.1: The best solution of generation No. 1

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.18394048362970353 style=solid] |
| $1 \rightarrow 2$ | [color=green penwidth=0.10121861859224737 style=solid] |
| $3 \rightarrow 2$ | [color=red penwidth=0.3513897120952606 style=solid] |

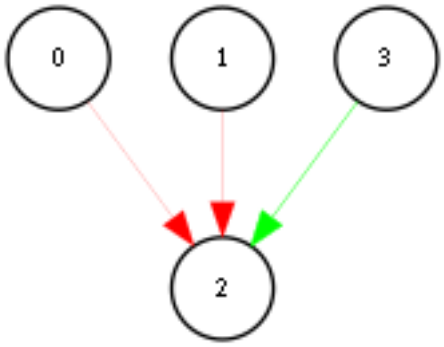


Figure 6.2: The best solution of generation No. 2

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.10914446078240872 style=solid] |
| $1 \rightarrow 2$ | [color=red penwidth=0.11519362907856703 style=solid] |
| $3 \rightarrow 2$ | [color=green penwidth=0.3102854698896408 style=solid] |

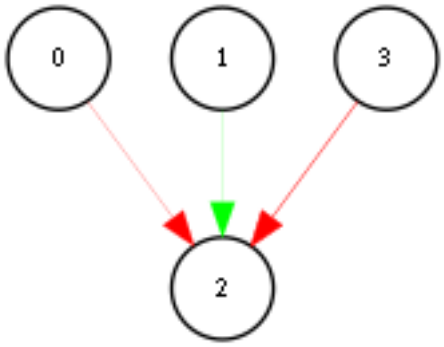


Figure 6.3: The best solution of generation No. 3

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.1434977412223816 style=solid] |
| $1 \rightarrow 2$ | [color=green penwidth=0.10121861859224737 style=solid] |
| $3 \rightarrow 2$ | [color=red penwidth=0.3513897120952606 style=solid] |

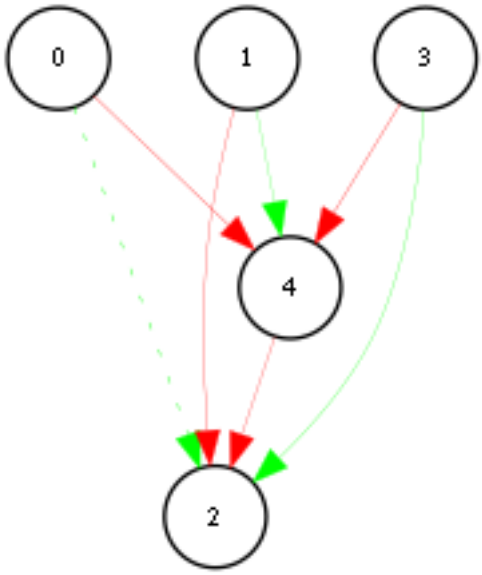


Figure 6.4: The best solution of generation No. 4

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=green penwidth=0.272405332326889 style=dotted] |
| $1 \rightarrow 2$ | [color=red penwidth=0.17917489856481553 style=solid] |
| $3 \rightarrow 2$ | [color=green penwidth=0.17829216867685319 style=solid] |
| $0 \rightarrow 4$ | [color=red penwidth=0.20905877500772477 style=solid] |
| $4 \rightarrow 2$ | [color=red penwidth=0.17074624300003052 style=solid] |
| $3 \rightarrow 4$ | [color=red penwidth=0.21371834576129914 style=solid] |
| $1 \rightarrow 4$ | [color=green penwidth=0.14386123269796372 style=solid] |

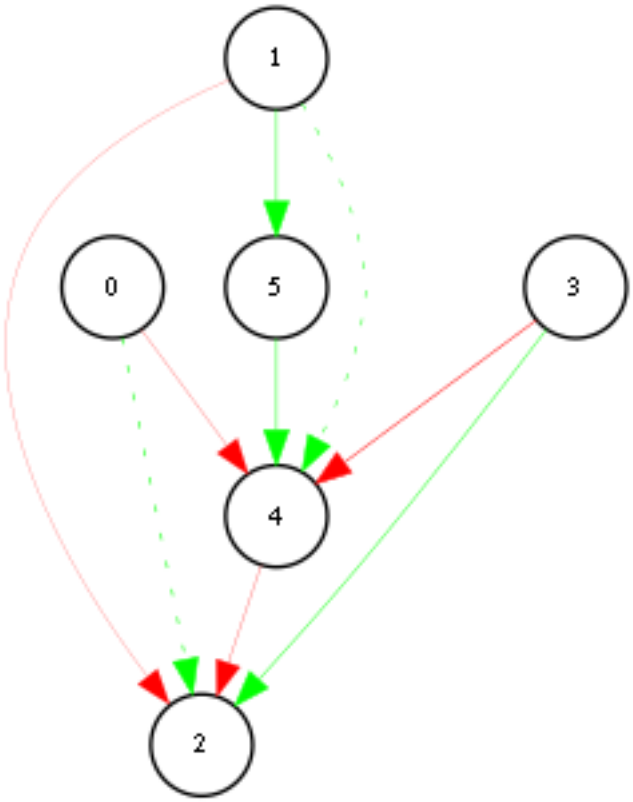


Figure 6.5: The best solution of generation No. 5

| node | attributes |
|---|---|
| 0 → 2 | [color=green penwidth=0.39665035009384153 style=dotted] |
| 1 → 2 | [color=red penwidth=0.12320034429430962 style=solid] |
| 3 → 2 | [color=green penwidth=0.3021953612565994 style=solid] |
| 0 → 4 | [color=red penwidth=0.14885845258831978 style=solid] |
| 4 → 2 | [color=red penwidth=0.17056937366724015 style=solid] |
| 3 → 4 | [color=red penwidth=0.3478629648685455 style=solid] |
| 1 → 4 | [color=green penwidth=0.3161144763231277 style=dotted] |
| 1 → 5 | [color=green penwidth=0.3000000029802322 style=solid] |
| 5 → 4 | [color=green penwidth=0.3161144763231277 style=solid] |

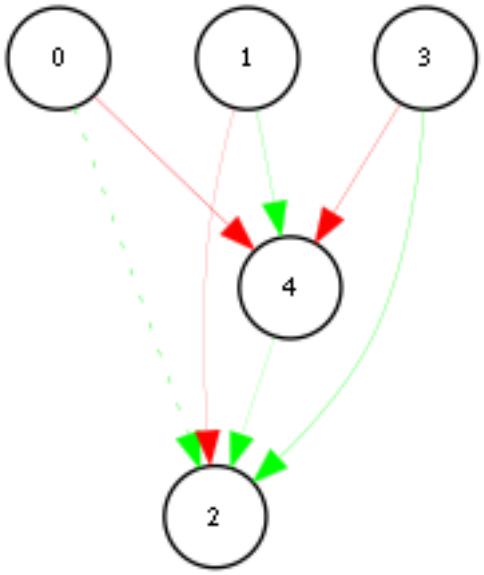


Figure 6.6: The best solution of generation No. 6

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=green penwidth=0.27899996042251585 style=dotted] |
| $1 \rightarrow 2$ | [color=red penwidth=0.10220212945714593 style=solid] |
| $3 \rightarrow 2$ | [color=green penwidth=0.19314291179180146 style=solid] |
| $0 \rightarrow 4$ | [color=red penwidth=0.21453025043010712 style=solid] |
| $4 \rightarrow 2$ | [color=green penwidth=0.10157375331036747 style=solid] |
| $3 \rightarrow 4$ | [color=red penwidth=0.16323540955781937 style=solid] |
| $1 \rightarrow 4$ | [color=green penwidth=0.13968576341867447 style=solid] |

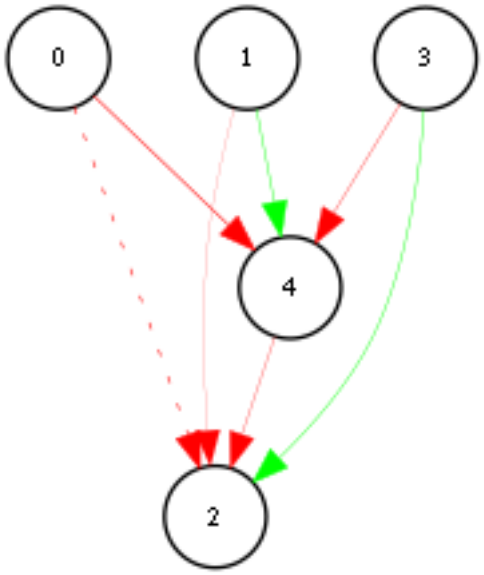


Figure 6.7: The best solution of generation No. 9

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.3891871929168701 style=dotted] |
| $1 \rightarrow 2$ | [color=red penwidth=0.10447230320423842 style=solid] |
| $3 \rightarrow 2$ | [color=green penwidth=0.2884440928697586 style=solid] |
| $0 \rightarrow 4$ | [color=red penwidth=0.4012172281742096 style=solid] |
| $4 \rightarrow 2$ | [color=red penwidth=0.19657768905162812 style=solid] |
| $3 \rightarrow 4$ | [color=red penwidth=0.22929360568523408 style=solid] |
| $1 \rightarrow 4$ | [color=green penwidth=0.2786946356296539 style=solid] |

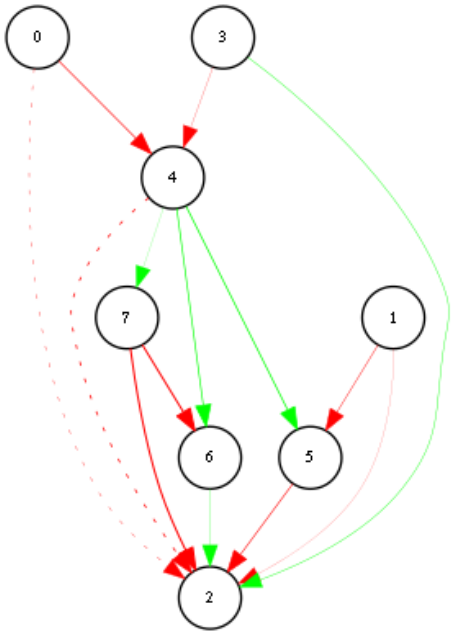


Figure 6.8: The best solution of generation No. 10

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.2384231746196747 style=dotted] |
| $1 \rightarrow 2$ | [color=red penwidth=0.10222489843145013 style=solid] |
| $3 \rightarrow 2$ | [color=green penwidth=0.2740603446960449 style=solid] |
| $0 \rightarrow 4$ | [color=red penwidth=0.43850884437561033 style=solid] |
| $4 \rightarrow 2$ | [color=red penwidth=0.5562108516693115 style=dotted] |
| $3 \rightarrow 4$ | [color=red penwidth=0.15342475324869156 style=solid] |
| $1 \rightarrow 5$ | [color=red penwidth=0.3081859558820724 style=solid] |
| $5 \rightarrow 2$ | [color=red penwidth=0.4128013014793396 style=solid] |
| $4 \rightarrow 5$ | [color=green penwidth=0.5362373948097229 style=solid] |
| $4 \rightarrow 6$ | [color=green penwidth=0.4724526762962341 style=solid] |
| $6 \rightarrow 2$ | [color=green penwidth=0.175114943087101 style=solid] |
| $4 \rightarrow 7$ | [color=green penwidth=0.11537624560296536 style=solid] |
| $7 \rightarrow 2$ | [color=red penwidth=0.7785329580307007 style=solid] |
| $7 \rightarrow 6$ | [color=red penwidth=0.7385876536369324 style=solid] |

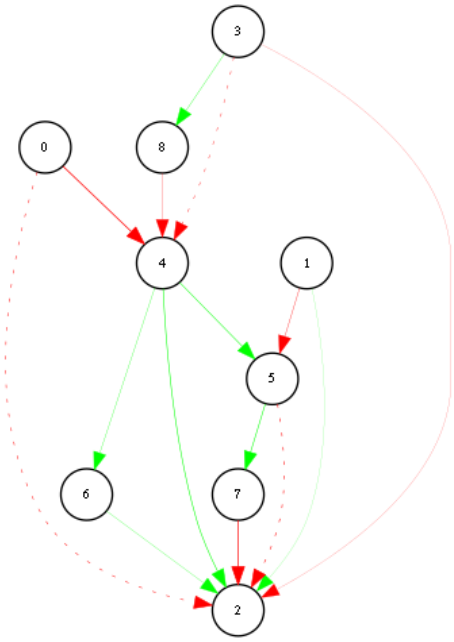


Figure 6.9: The best solution of generation No. 19

| node | attributes |
|---|---|
| $0 \rightarrow 2$ | [color=red penwidth=0.34734596908092497 style=dotted] |
| $1 \rightarrow 2$ | [color=green penwidth=0.1003281951067038 style=solid] |
| $3 \rightarrow 2$ | [color=red penwidth=0.10508393067866564 style=solid] |
| $0 \rightarrow 4$ | [color=red penwidth=0.5357576370239258 style=solid] |
| $4 \rightarrow 2$ | [color=green penwidth=0.3929439544677734 style=solid] |
| $3 \rightarrow 4$ | [color=red penwidth=0.2771649122238159 style=dotted] |
| $1 \rightarrow 5$ | [color=red penwidth=0.23397754728794098 style=solid] |
| $5 \rightarrow 2$ | [color=red penwidth=0.398834890127182 style=dotted] |
| $4 \rightarrow 5$ | [color=green penwidth=0.35414925813674925 style=solid] |
| $4 \rightarrow 6$ | [color=green penwidth=0.20132730007171631 style=solid] |
| $6 \rightarrow 2$ | [color=green penwidth=0.16707025319337845 style=solid] |
| $5 \rightarrow 7$ | [color=green penwidth=0.3782035768032074 style=solid] |
| $7 \rightarrow 2$ | [color=red penwidth=0.505322253704071 style=solid] |
| $3 \rightarrow 8$ | [color=green penwidth=0.21914844810962678 style=solid] |
| $8 \rightarrow 4$ | [color=red penwidth=0.16383900344371796 style=solid] |

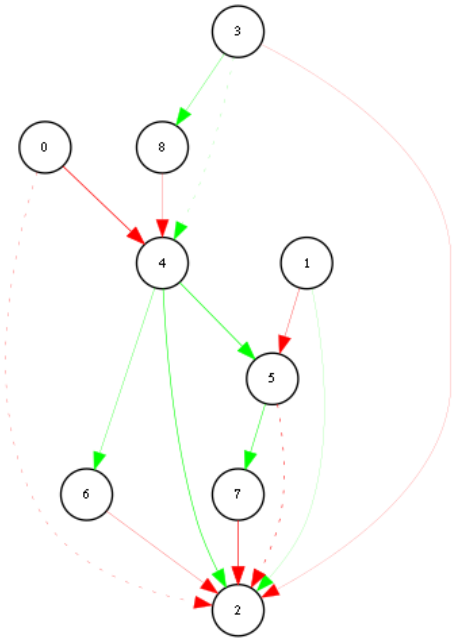


Figure 6.10: The best solution of generation No. 30

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=red penwidth=0.10508393067866564 style=solid] |
| 0 → 4 | [color=red penwidth=0.5357576370239258 style=solid] |
| 4 → 2 | [color=green penwidth=0.4374311029911041 style=solid] |
| 3 → 4 | [color=green penwidth=0.12034918032586575 style=dotted] |
| 1 → 5 | [color=red penwidth=0.23397754728794098 style=solid] |
| 5 → 2 | [color=red penwidth=0.398834890127182 style=dotted] |
| 4 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 4 → 6 | [color=green penwidth=0.23329939544200898 style=solid] |
| 6 → 2 | [color=red penwidth=0.18149497956037522 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.505322253704071 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |

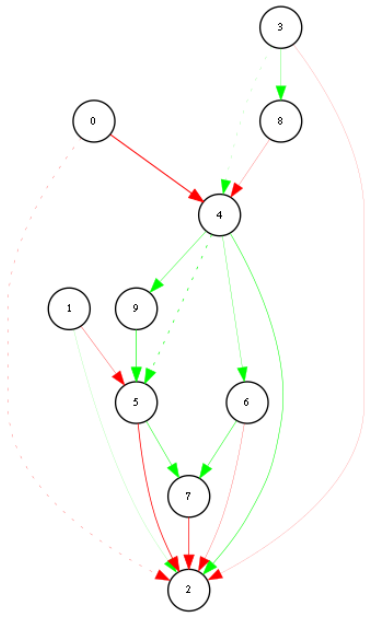


Figure 6.11: The best solution of generation No. 38

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=red penwidth=0.10508393067866564 style=solid] |
| 0 → 4 | [color=red penwidth=0.7631097793579101 style=solid] |
| 4 → 2 | [color=green penwidth=0.3929439544677734 style=solid] |
| 3 → 4 | [color=green penwidth=0.12034918032586575 style=dotted] |
| 1 → 5 | [color=red penwidth=0.23397754728794098 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |
| 4 → 5 | [color=green penwidth=0.505379444360733 style=dotted] |
| 4 → 6 | [color=green penwidth=0.23329939544200898 style=solid] |
| 6 → 2 | [color=red penwidth=0.18149497956037522 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3764008343219757 style=solid] |

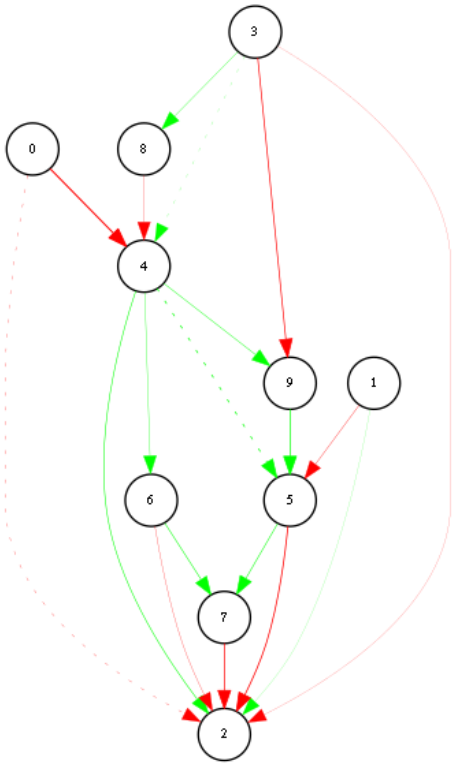


Figure 6.12: The best solution of generation No. 47

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=red penwidth=0.10508393067866564 style=solid] |
| 0 → 4 | [color=red penwidth=0.7631097793579101 style=solid] |
| 4 → 2 | [color=green penwidth=0.3929439544677734 style=solid] |
| 3 → 4 | [color=green penwidth=0.12034918032586575 style=dotted] |
| 1 → 5 | [color=red penwidth=0.23397754728794098 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |
| 4 → 5 | [color=green penwidth=0.505379444360733 style=dotted] |
| 4 → 6 | [color=green penwidth=0.23329939544200898 style=solid] |
| 6 → 2 | [color=red penwidth=0.18149497956037522 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3764008343219757 style=solid] |
| 3 → 9 | [color=red penwidth=0.494989013671875 style=solid] |

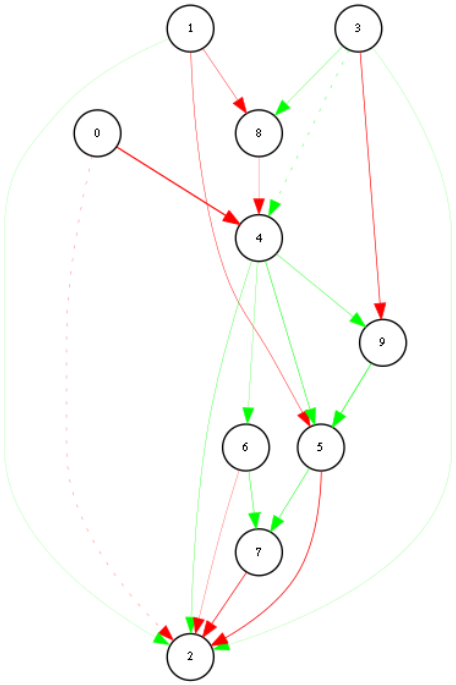


Figure 6.13: The best solution of generation No. 81

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=green penwidth=0.11510254107415677 style=solid] |
| 0 → 4 | [color=red penwidth=0.8551668882369995 style=solid] |
| 4 → 2 | [color=green penwidth=0.24948569238185883 style=solid] |
| 3 → 4 | [color=green penwidth=0.27409485578536985 style=dotted] |
| 1 → 5 | [color=red penwidth=0.32156639397144315 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |
| 4 → 5 | [color=green penwidth=0.39555067420005796 style=solid] |
| 4 → 6 | [color=green penwidth=0.20035351663827897 style=solid] |
| 6 → 2 | [color=red penwidth=0.19354103356599808 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3574644088745117 style=solid] |
| 3 → 9 | [color=red penwidth=0.494989013671875 style=solid] |
| 1 → 8 | [color=red penwidth=0.2873188316822052 style=solid] |

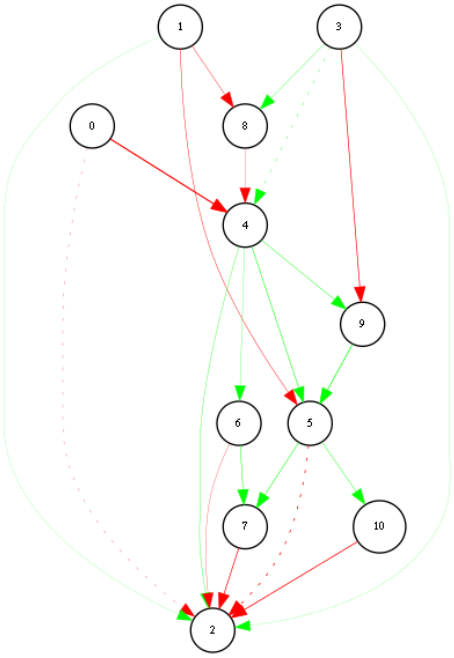


Figure 6.14: The best solution of generation No. 92

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=green penwidth=0.11510254107415677 style=solid] |
| 0 → 4 | [color=red penwidth=0.8551668882369995 style=solid] |
| 4 → 2 | [color=green penwidth=0.24948569238185883 style=solid] |
| 3 → 4 | [color=green penwidth=0.27409485578536985 style=dotted] |
| 1 → 5 | [color=red penwidth=0.32156639397144315 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=dotted] |
| 4 → 5 | [color=green penwidth=0.39555067420005796 style=solid] |
| 4 → 6 | [color=green penwidth=0.20035351663827897 style=solid] |
| 6 → 2 | [color=red penwidth=0.19354103356599808 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3574644088745117 style=solid] |
| 3 → 9 | [color=red penwidth=0.494989013671875 style=solid] |
| 1 → 8 | [color=red penwidth=0.2873188316822052 style=solid] |
| 5 → 10 | [color=green penwidth=0.3000000029802322 style=solid] |
| 10 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |

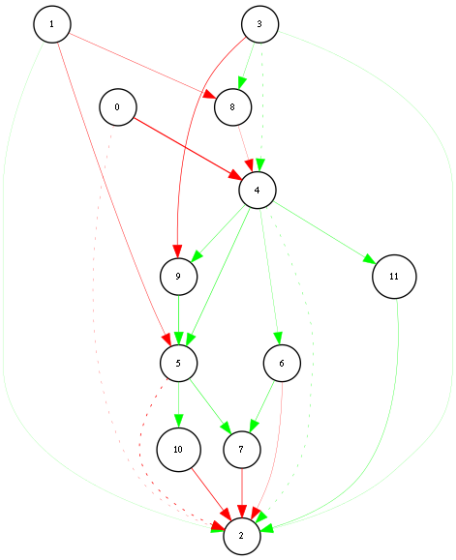


Figure 6.15: The best solution of generation No. 123

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=green penwidth=0.11510254107415677 style=solid] |
| 0 → 4 | [color=red penwidth=0.8551668882369995 style=solid] |
| 4 → 2 | [color=green penwidth=0.24948569238185883 style=dotted] |
| 3 → 4 | [color=green penwidth=0.27409485578536985 style=dotted] |
| 1 → 5 | [color=red penwidth=0.32156639397144315 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=dotted] |
| 4 → 5 | [color=green penwidth=0.39555067420005796 style=solid] |
| 4 → 6 | [color=green penwidth=0.20035351663827897 style=solid] |
| 6 → 2 | [color=red penwidth=0.19354103356599808 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3574644088745117 style=solid] |
| 3 → 9 | [color=red penwidth=0.494989013671875 style=solid] |
| 1 → 8 | [color=red penwidth=0.2873188316822052 style=solid] |
| 5 → 10 | [color=green penwidth=0.3000000029802322 style=solid] |
| 10 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |
| 4 → 11 | [color=green penwidth=0.3000000029802322 style=solid] |
| 11 → 2 | [color=green penwidth=0.24948569238185883 style=solid] |

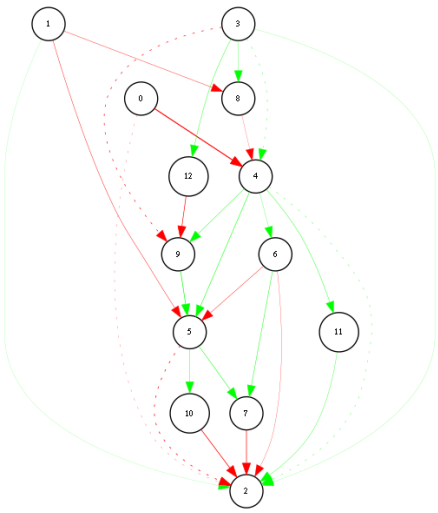


Figure 6.16: The best solution of generation No. 140

| node | attributes |
|---|---|
| 0 → 2 | [color=red penwidth=0.18192986249923707 style=dotted] |
| 1 → 2 | [color=green penwidth=0.1003281951067038 style=solid] |
| 3 → 2 | [color=green penwidth=0.11510254107415677 style=solid] |
| 0 → 4 | [color=red penwidth=0.8551668882369995 style=solid] |
| 4 → 2 | [color=green penwidth=0.24948569238185883 style=dotted] |
| 3 → 4 | [color=green penwidth=0.27409485578536985 style=dotted] |
| 1 → 5 | [color=red penwidth=0.32156639397144315 style=solid] |
| 5 → 2 | [color=red penwidth=0.5664057791233063 style=dotted] |
| 4 → 5 | [color=green penwidth=0.39555067420005796 style=solid] |
| 4 → 6 | [color=green penwidth=0.20035351663827897 style=solid] |
| 6 → 2 | [color=red penwidth=0.19354103356599808 style=solid] |
| 5 → 7 | [color=green penwidth=0.3782035768032074 style=solid] |
| 7 → 2 | [color=red penwidth=0.5177108705043793 style=solid] |
| 3 → 8 | [color=green penwidth=0.21914844810962678 style=solid] |
| 8 → 4 | [color=red penwidth=0.16383900344371796 style=solid] |
| 4 → 9 | [color=green penwidth=0.3000000029802322 style=solid] |
| 9 → 5 | [color=green penwidth=0.505379444360733 style=solid] |
| 6 → 7 | [color=green penwidth=0.3574644088745117 style=solid] |
| 3 → 9 | [color=red penwidth=0.494989013671875 style=dotted] |
| 1 → 8 | [color=red penwidth=0.2873188316822052 style=solid] |
| 5 → 10 | [color=green penwidth=0.3000000029802322 style=solid] |
| 10 → 2 | [color=red penwidth=0.5664057791233063 style=solid] |
| 4 → 11 | [color=green penwidth=0.3000000029802322 style=solid] |
| 11 → 2 | [color=green penwidth=0.24948569238185883 style=solid] |
| 3 → 12 | [color=green penwidth=0.3000000029802322 style=solid] |
| 12 → 9 | [color=red penwidth=0.494989013671875 style=solid] |
| 6 → 5 | [color=red penwidth=0.3114542722702026 style=solid] |

# Appendix B: Implementation of some modules mentioned in thesis

```
if __name__ == "__main__":
NUM_WORKERS = 15
pool = Pool(NUM_WORKERS)
for i in range(1):

neat = pop.Population(c.RecsysConfig)
solution, generation = neat.run(pool, shared_data)

if solution is not None:
avg_num_generations = ((avg_num_generations * num_of_solutions) +
    generation) / (num_of_solutions + 1)
min_num_generations = min(generation, min_num_generations)

num_hidden_nodes = len([n for n in solution.node_genes if n.type ==
     'hidden'])
avg_num_hidden_nodes = ((avg_num_hidden_nodes * num_of_solutions) +
     num_hidden_nodes) / (num_of_solutions + 1)
min_hidden_nodes = min(num_hidden_nodes, min_hidden_nodes)
max_hidden_nodes = max(num_hidden_nodes, max_hidden_nodes)
if num_hidden_nodes == 1:
found_minimal_solution += 1

num_of_solutions += 1
draw_net(solution, view=True, filename='./images/solution-' + str(
    num_of_solutions), show_disabled=True)

print('Total Number of Solutions: ', num_of_solutions)
print('Average Number of Hidden Nodes in a Solution',
    avg_num_hidden_nodes)
```

```
25 print('Solution found on average in:', avg_num_generations, '
       generations')
26 print('Minimum number of hidden nodes:', min_hidden_nodes)
27 print('Maximum number of hidden nodes:', max_hidden_nodes)
28 print('Minimum number of generations:', min_num_generations)
29 print('Found minimal solution:', found_minimal_solution, 'times')
```

Listing 6.1: Main module

```
import random
import neat.utils as utils


def mutate(genome, config):
"""
Applies connection and structural mutations at proper rate.
Connection Mutations: Uniform Weight Perturbation or Replace Weight
     Value with Random Value
Structural Mutations: Add Connection and Add Node
:param genome: Genome to be mutated
:param config: Experiments' configuration file
:return: None
"""

if utils.rand_uni_val() < config.CONNECTION_MUTATION_RATE:
for c_gene in genome.connection_genes:
if utils.rand_uni_val() < config.CONNECTION_PERTURBATION_RATE:
perturb = utils.rand_uni_val() * random.choice([1, -1])
c_gene.weight += perturb
else:
c_gene.set_rand_weight()

if utils.rand_uni_val() < config.ADD_NODE_MUTATION_RATE:
genome.add_node_mutation()

if utils.rand_uni_val() < config.ADD_CONNECTION_MUTATION_RATE:
genome.add_connection_mutation()
```

Listing 6.2: Mutation

```
from copy import deepcopy
import neat.utils as utils
from neat.genotype.genome import Genome

def crossover(genome_1, genome_2, config):
"""
Crossovers two Genome instances as described in the original NEAT
    implementation
:param genome_1: First Genome Instance
:param genome_2: Second Genome Instance
:param config: Experiment's configuration class
:return: A child Genome Instance
"""

child = Genome()
best_parent, other_parent = order_parents(genome_1, genome_2)

# Crossover connections
# Randomly add matching genes from both parents
for c_gene in best_parent.connection_genes:
matching_gene = other_parent.get_connect_gene(c_gene.innov_num)

if matching_gene is not None:
# Randomly choose where to inherit gene from
if utils.rand_bool():
child_gene = deepcopy(c_gene)
else:
child_gene = deepcopy(matching_gene)

# No matching gene - is disjoint or excess
# Inherit disjoint and excess genes from best parent
else:
child_gene = deepcopy(c_gene)
```

```
33
34 # Apply rate of disabled gene being re-enabled
35 if not child_gene.is_enabled:
36 is_reenabeled = utils.rand_uni_val() <= config.
       CROSSOVER_REENABLE_CONNECTION_GENE_RATE
37 enabled_in_best_parent = best_parent.get_connect_gene(child_gene.
       innov_num).is_enabled
38
39 if is_reenabeled or enabled_in_best_parent:
40 child_gene.is_enabled = True
41
42 child.add_connection_copy(child_gene)
43
44 # Crossover Nodes
45 # Randomly add matching genes from both parents
46 for n_gene in best_parent.node_genes:
47 matching_gene = other_parent.get_node_gene(n_gene.id)
48
49 if matching_gene is not None:
50 # Randomly choose where to inherit gene from
51 if utils.rand_bool():
52 child_gene = deepcopy(n_gene)
53 else:
54 child_gene = deepcopy(matching_gene)
55
56 # No matching gene - is disjoint or excess
57 # Inherit disjoint and excess genes from best parent
58 else:
59 child_gene = deepcopy(n_gene)
60
61 child.add_node_copy(child_gene)
62
```

```
63 return child
```

Listing 6.3: Crossover

# Curriculum Vitae

## Personal Data


ADDRESS: 194 Maki Ave., Sudbury, Ontario, P3E 2P2, Canada

EMAIL: smonemian@laurentian.ca

PHONE: +1 (249) 879-9979


## Education

| | |
|---|---|
| JAN 2019-NOW | Master of Science in COMPUTATIONAL SCIENCE, **Laurentian University**, Sudbury, ON, Canada. GPA: 9.5/10 |
| SEP 2003-JUL 2008 | Bachelor of Science in SOFTWARE ENGINEERING, School of Electrical and Computer Engineering, College of Engineering, **University of Tehran**, (Ranked 1$^{st}$ in the country), Tehran, Iran. |

## Teaching Experience

Summer 2012 | Mofid Securities Co.

Introduction to Computer Programming and Code Smells

Responsibilities:

- Teaching the new programmers how to program clean and reusable codes regarding to design patterns and code smells
- Step by step software development by designing small projects to apply the educated skills in a real situation

Fall 2007 | University of Tehran, Tehran

Database Lab

Responsibilities:

- Teaching students of the lab how to query SQL database
- Design weekly homework computer assignments and their solutions
- Consulting students to do their final project

Fall 2005 | University of Tehran, Tehran

Data Structure

Responsibilities:

- Preparing assignments
- Grading assignments and exams
- Consulting students for their final projects

# Publication

- Ghalib, Abdulaziz, Tyler D. Jessup, Julia Johnson, and ***Seyedamin Monemian***. "Clustering and Classification to Evaluate Data Reduction via Johnson-Lindenstrauss Transform." In Future of Information and Communication Conference, pp. 190-209. Springer, Cham, 2020.

- Rahat, Mahmoud, Alireza Talebpour, and ***Seyedamin Monemian***. "A recursive algorithm for open information extraction from Persian texts." International Journal of Computer Applications in Technology 57, no. 3 (2018): 193-206.

# Certificates

- Machine Learning, Taught by Andrew NG, Adjunct Professor, Stanford University; Coursera Online Course, October 2017