Analyzing the Relationships between Tags on Movies using Kohonen Self-Organizing Feature Maps

Palle Klewitz
Technische Universität München
palle.klewitz@tum.de

ABSTRACT

Artificial Neural Networks have become a popular approach to solve a large set of supervised and unsupervised learning tasks. Common neural network architectures for different learning paradigms with a focus on Kohonen Self-Organizing Maps are explained. The use of Self-Organizing Maps to perform data analysis on a large, high dimensional data set and to efficiently generate suggestions in a recommendation system for movies based on user provided search queries is explored.

KEYWORDS

SOM, Kohonen Self-Organizing Feature Maps, Competitive Learning, Neural Networks

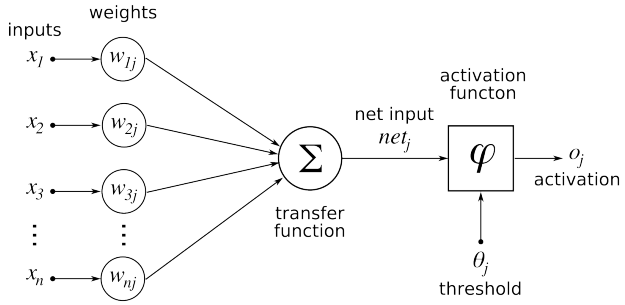


Figure 1: Computations performed by a single neuron: The sum of weighted inputs and a threshold are passed through a nonlinear activation function to compute its output.
[7]

1 NEURAL NETWORKS

Artificial Neural Networks are computing models modelled after nervous systems found in biological organisms such as the human brain, which are utilized in many areas of machine learning. An artificial neural network consists of nodes, that are comparable to neurons in a brain, and edges, which are equivalent to connections between neurons used to propagate signals [1]. Many different computational problems such as classification problems in computer vision, dimensionality reduction, clustering or speech recognition and synthesis can be modelled using different architectures of neural networks [2, 3, 4]. Neural networks are typically trained before operation. Training refers to the process of finding the right parameters, such as connection weights, for the network to perform well on a task it has been designed for. A neural network is trained either unsupervised or supervised depending on its type and application.

Supervised learning refers to learning where a set of inputs and corresponding outputs is known and the network is used to generalize the mapping between input and output. A network may for example learn how a given object looks and will be able to identify that object in previously unknown lighting conditions and camera angles or to recognize a spoken word from a voice it has not been trained with.

A neural network used in an unsupervised training scenario can perform statistical analysis on a data set without requiring a training data set where samples are assigned a desired result. Instead, the network is presented with the data without additional knowledge about data points. An unsupervised network can for example find a more efficient representation of a data set by dividing it into clusters or compressing a data point into a lower dimensional representation [5].

1.1 Feed Forward Neural Networks

Feed forward neural networks can approximate linear and nonlinear functions mapping values of an input vector like an image, to an output vector or scalar value.

1.1.1 Structure of Feed Forward Neural Networks. A feed forward neural network consists of layers of neurons, which propagate signals to the next layer. The first layer, also referred to as the input layer, takes an input vector, where every element of the vector is mapped to a neuron. The last layer has no outgoing connections. The output of the layer is the output of the whole network. A Feed Forward Neural Network may have multiple hidden layers between the input and output layer [1]. More hidden neurons improve the ability of the network to approximate highly nonlinear functions but also increase the computation cost associated with the training of the network. A network with many hidden layers is commonly referred to as a deep neural network [6].

As shown in Figure 1 each neuron of a hidden or output layer has incoming connections from the anterior layer, where each connection has an associated weight value. A positive weight corresponds to a stimulating signal and a negative weight corresponds to a inhibitory signal in a biological neural network. The magnitude of the weight determines the strength of the influence that the output of a neuron has on the input of another neuron. Neurons perform nonlinear transformations such as the hyperbolic tangent, sigmoid or rectified linear unit function (Figure 2) on its input and propagate the resulting output to allow the network to approximate nonlinear functions [1].

1.1.2 Backpropagation. Feed forward neural networks are typically trained using backpropagation with gradient descent. The

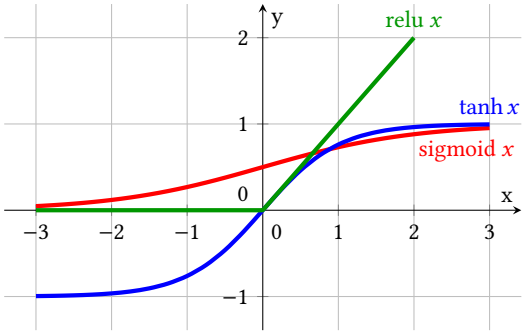


Figure 2: Commonly used activation functions: Hyperbolic tangent, Sigmoid and Rectified Linear Unit

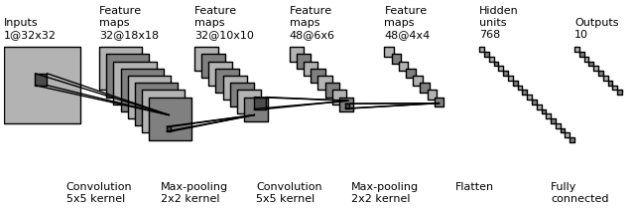


Figure 3: Structure of a convolutional neural network: The first layers are alternating convolution and pooling layers. Multiple convolution filters per layer are used to generate more multiple feature maps. The last layers are usually fully connected.

difference between expected and actual output for a training sample are used to compute the error of the network. The gradient of the error can then be computed for each layer of neurons beginning from the output layer. Connection weights can then be adjusted in the opposite direction of the gradient to reduce the error of the network thereby improving its accuracy [1].

1.1.3 Convolutional Neural Networks. Classical feed forward neural networks use a large number of weights between layers as layers are generally fully connected, so each neuron is connected to every neuron of the posterior layer. This leads to problems such as computational and space complexity and overfitting. Overfitting occurs when the network fails to generalize and instead learns individual samples. Networks for example for image processing have a great number of inputs and thereby require many weights. To avoid these disadvantages of classical feed forward neural networks convolutional neural networks can be used. Convolutional neural networks use convolution layers, which share weights across neurons of a layer and limit the number of inputs to a neuron to a small receptive field, and pooling layers, which reduce the amount of data by dropping small inputs, to address these issues [8]. Each convolution layer provides further abstraction away from the input space into the feature space. The first layer of a network may detect edges in an image whereas the later layers may detect higher level features which are composed of lower level features [9].

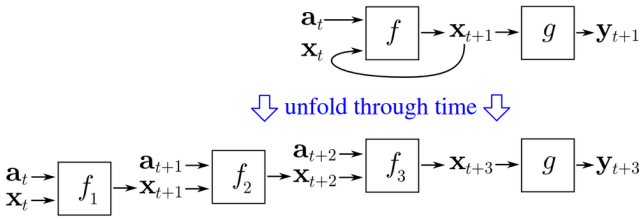


Figure 4: A recurrent neural neural network is unfolded into a feed forward neural network for training [12]

1.1.4 Use cases of Feed Forward Neural Networks. Feed forward neural networks are typically used in regression and classification tasks. Networks used in classification assign a label to a data point, e.g. a recognized letter to an image of a letter. Regression networks compute a mapping function between two or more variables. The output could for example be the location of an object in an image. While both classification and regression are supervised learning tasks, a feed forward neural network can also be applied to certain unsupervised learning problems such as auto encoding. An auto encoding neural network has a bottleneck layer which has fewer units than the output and input layer. The network is then trained to output its input and thereby to find a lower dimensional representation of its input [2, 10].

1.2 Recurrent Neural Networks

In a recurrent neural network, directed edges between neurons may form a cycle so that the output of a neuron becomes part of its input one or a few iterations later. This property enables the network to store temporal states [3]. Recurrent neural networks can be used to process and generate temporal sequences of data. Different architectures of recurrent neural network have been developed.

1.2.1 Long Short-Term Memory. A popular architecture of recurrent neural networks is the Long Short-Term Memory network (LSTM). Neurons of these networks use internal memory and have multiple gates controlling the content of its memory and how the memory affects the output of the neuron. LSTMs perform well on operations on human language such as speech recognition and synthesis [3, 4].

1.2.2 Training a Recurrent Neural Network. Recurrent neural networks working on sequences are typically trained using backpropagation through time. For this, recurrent connections of the network are unfolded into a feed forward neural network as shown in Figure 4, which is then trained using backpropagation [11].

2 KOHONEN SELF-ORGANIZING FEATURE MAPS

A Kohonen Self-Organizing (Feature) Map is a type of unsupervised neural network that is used to achieve a lower dimensional, discrete representation of a high dimensional data set [13]. Self-Organizing Maps were introduced in 1982 by Teuvo Kohonen [14].

2.1 Structure of a Self-Organizing Map

A Self-Organizing map M is a neural network architecture in which neurons are arranged in a lattice where each neuron (w, p) has a weight vector w pointing to a location in a k dimensional input space $\mathbb{R}^k$ and a position p describing its location in the lattice of the map. For a two dimensional lattice, which is most commonly used, a Self-Organizing Map can be defined as

$$M \coloneqq \{(w, p) \mid w \in W \subset \mathbb{R}^k, p \in [n] \times [m], |W| = n \cdot m\}.$$

2.2 Features of a Self-Organizing Map

A trained Self-Organizing Map preserves the topology of a data set. Samples of the data set which are close to each other in the input space will likely be projected onto the same node or neighbouring nodes of the Self-Organizing Map.

The density of nodes of the self organizing map approximates the density of data points in the data set with which the map was trained. Sparse regions will contain fewer nodes than dense regions [13].

2.3 Training

A Self-Organizing Map is adapted to a finite data set $S \subset \mathbb{R}^k$ in a training phase where the quality of the lower dimensional representation of the data set is improved through multiple iterations. Each training iteration consists of a competitive and a cooperative stage.

At the beginning of an iteration, a random sample $s \in S$ is chosen. In the competition phase the best matching unit $m_{BMU} \in M$ which is closest to the sample s is then selected

$$m_{BMU} = \underset{(w,p)\in M}{\arg\min} \|w - s\|_2 .$$

In the following cooperative stage, the best matching unit and neurons which are topologically close to it will be updated to be closer to the input sample, as shown in Figure 5. This process will make topologically close nodes respond to similar inputs. A neuron (w_t, p) in the iteration t is updated by

$$\Delta w_t = \eta(t) \cdot \phi(p_{BMU}, p, t) \cdot (s - w_t)$$

where $\eta(t)$ is the learning rate and $\phi(p_{BMU}, p, t)$ is a neighbourhood function. The updated weight is then given by

$$w_{t+1} = w_t + \Delta w_t$$

The neighbourhood function determines the magnitude of the update of any given neuron depending on the topological distance towards the best matching unit. Both the learning rate and the neighbourhood function are dependent on the current iteration. The learning rate gradually decreases over time. The size of the neighbourhood also shrinks over time. The most commonly used neighbourhood function is a Gaussian function

$$\phi(p_0, p_1, t) \coloneqq \exp\left(\frac{\|p_0 - p_1\|}{2 \cdot \sigma(t)^2}\right).$$

The size of the neighbourhood function is given by the standard deviation function

$$\sigma(t) \coloneqq \sigma_0 \cdot \exp\left(-2 \cdot \frac{t}{T}\right)$$

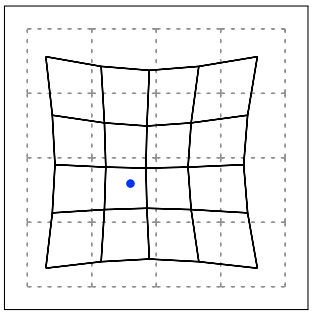
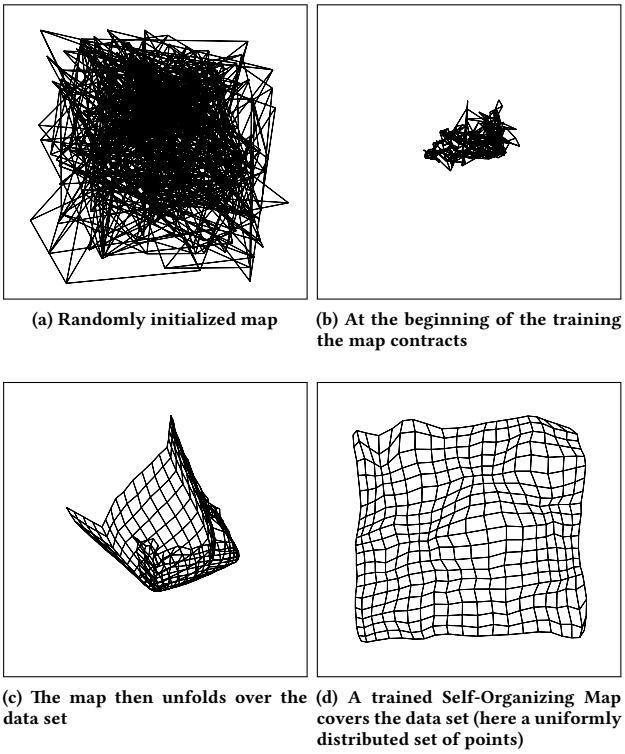


Figure 5: A single training epoch on a Self-Organizing Map: The map before the update is shown dashed in grey. The map is then updated towards the blue sample point.

where T is the total number of training iterations to perform and σ_0 is the initial neighbourhood size. As shown in Figure 6, in the beginning the Self-Organizing Map is tangled, as weights are initialized randomly. During training, the map first contracts and then unfolds over the data set [15].



(a) Randomly initialized map

(b) At the beginning of the training the map contracts

(c) The map then unfolds over the data set

(d) A trained Self-Organizing Map covers the data set (here a uniformly distributed set of points)

Figure 6: Training progress of a Self-Organizing Map

As training phase is a optimization process in which the representation of the map is optimized to fit the data set, the Self-Organizing Map may reach a faulty state. While unfolding, knots or similar defects may occur. The neighbourhood shrinks over time, so defects may not disappear later in training [15]. Figure 7 shows a knot as an example of a faulty state.

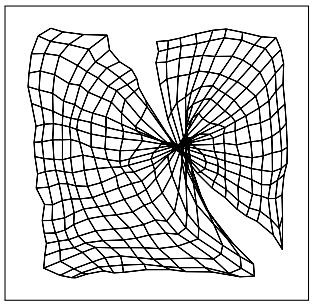


Figure 7: Faulty convergence: The network converged to a state containing a knot which will not disappear after further training.

2.4 Biological Motivation

Self-Organizing Maps are modelled after structures of the human brain in which for example visual, tactile and acoustic stimuli are mapped onto different regions of the cerebral cortex. This process is achieved through computational maps consisting of arrays of neurons which are selectively tuned to become activated by different stimuli. Incoming signals are encoded by the location of maximum activity within the map. This information can then be used by higher order units of the brain. In these maps, neurons which respond to similar stimuli are spatially close to each other. The spatial location derived by the computational map corresponds to features in its input [16].

2.5 Visualizing a Self-Organizing Map

2.5.1 Unified Distance Matrix. A unified distance matrix (U-matrix) shows the average distance towards direct neighbours of a node. Regions with low U-values have a high density of nodes and thereby a high density of sample points. Regions with high U-values have a low density of nodes. Such a visualization can be used to identify clusters and cluster boundaries. Statistical outliers can be identified by finding samples which map to low density regions of the map [17].

2.5.2 Component Planes. To visualize the relation between individual components of the input space and to find correlated features, component planes can be rendered. A component plane shows the distribution of values of a single dimension of the weight vectors. Similar to a unified distance matrix, nodes are equally spaced and sized. The color of a node corresponds to the value of the examined component from the weight vector of a node [18].

2.5.3 Visualization in Input Space. For low dimensional data sets it is possible to render the structure of the Self-Organizing Map directly into the input space. For example if the map is used to generate a one dimensional representation of a two dimensional data set, it would be rendered as a strip of points.

2.6 Applications of Self-Organizing Maps

2.6.1 Data Visualization. Self-Organizing Maps can be used to visualize high dimensional data sets. As data sets may have a high number of dimensions, it may not be possible to generate direct visualizations which can be interpreted by a human. Self-Organizing Maps can be used to perform multidimensional downscaling to reduce the data set to a low dimensional representation. It is then possible to generate visualizations of the Self-Organizing Map which reflect features of the data set such as correlations between dimensions [1].

2.6.2 Finding Related Data Points. Another application of Self-Organizing Maps is to perform search queries on a data set. It is possible to find samples which are related to other samples by selecting nodes which are close to the best matching unit of the input samples and finding data points which project onto these nodes. Also, the map can be used to find samples based on individual components, which describe properties of data points and find the best matching unit to the components of the query [19].

3 ANALYZING THE MOVIELENS DATA SET USING A SELF-ORGANIZING MAP

A Self-Organizing Map was used to visualize the relation between genres of movies. The tag genome data set which is part of the MovieLens data set was analyzed for this purpose.

3.1 The MovieLens Data Set

The genome tag of the MovieLens data set contains a set of 1128 tags describing features of a movie such as genre, oscar nominations, moods etc. For every of the approximately 10,000 movies contained in the data set each tag has a relevance score between 0 (not relevant) and 1 (very relevant). The data set was generated from user contributed ratings on the MovieLens website. Tags can be genres such as Action or Science Fiction or other properties of a movie like Oscar Nominations, the mood and more. Each movie is assigned a vector consisting of scores for every tag. This vector is referred to as the genome of the movie as it describes the movie in the tag space just similar to how the genome of a biological organism describe features of it [20].

3.2 Using a Self-Organizing Map to Analyze the Data

To train a self organizing map on the tag genome, a vector of tag scores for each movie was generated. The component n of a movie vector for the movie k stores the relevance of the tag n for the movie k. To visualize the data set, multiple two dimensional hexagonal grid maps consisting from 10 times 10 to 50 times 50 nodes were generated, trained and evaluated. The map was trained over 100,000 to 5,000,000 iterations with different neighbourhood sizes.

3.3 Visualizing the Self-Organizing Map

3.3.1 Component Planes. For every of the 1128 tags contained in the data set a component plane was rendered. Relatively low values for components are shown as blue while relatively high values are shown as red.

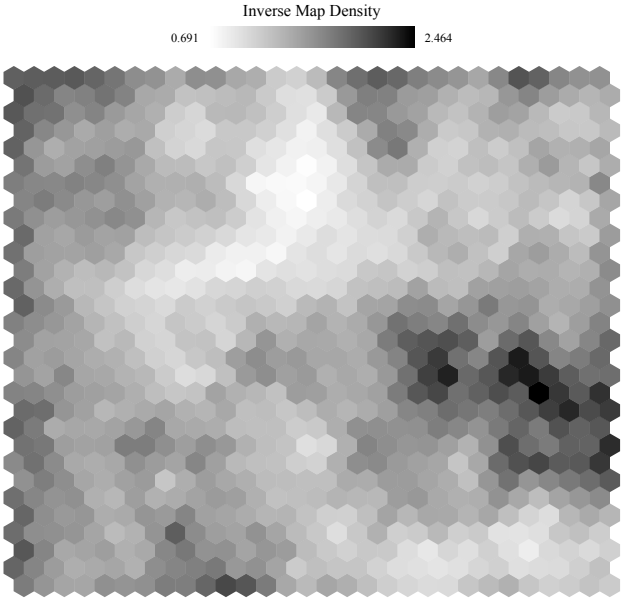


Figure 8: U-matrix of a Self-Organizing Map trained on the Tag Genome of the MovieLens data set. Bright regions (e.g. top middle) indicate clusters of closely related nodes. Dark regions (e.g. bottom right) show cluster boundaries.

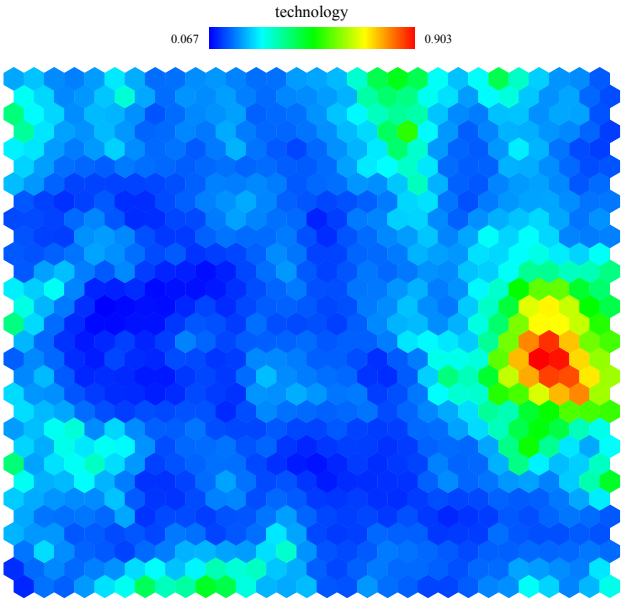


Figure 9: Component Plane for the Technology tag

3.4 Results

The component planes can be used to find similar tags or tags which are subsets of other tags in the data set.

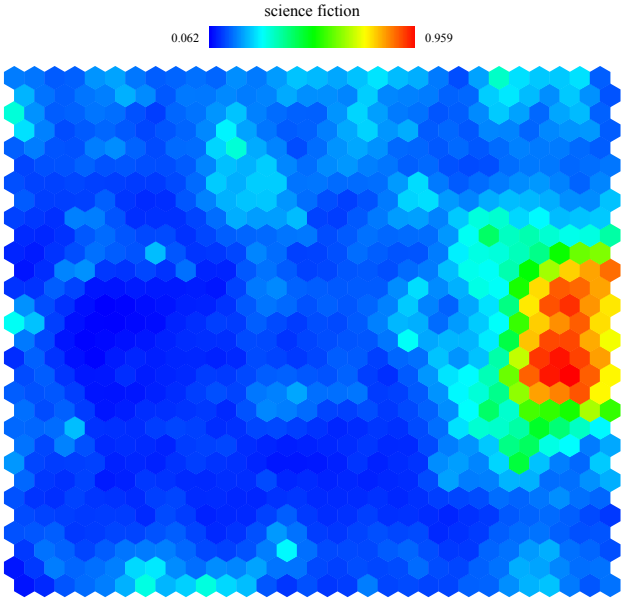


Figure 10: Component plane for the science fiction tag. Only small differences can be observed when comparing the component planes for science fiction and technology, indicating that a strong correlation between these tags exist.

The tags ”Technology” and ”Science Fiction” have similar component planes indicating a strong correlation between these tags. A movie with a high ”Technology” component value will likely have a high ”Science Fiction” component value and vice versa.

The tag ”Aliens” occupies a part of the region which occupied by the tag ”Science Fiction”. As the ”Science Fiction” tag has high component values other regions of the map as well, a movie about aliens will likely be a science fiction movie whereas a science fiction movie may not necessarily be a movie about aliens.

3.5 Building a Recommendation System

Additionally to generating visualizations to show the relationship between tags, a search engine for movies was built with the goal of finding movies based on a query of preferred and disfavoured tags, which is shown in Figure 12. The search engine has been realized in form of a web site which sends queries made by the user to a server which responds with search results generated using a self-organizing map.

3.5.1 Finding similar tags. One goal of the search engine is to offer the user a list of suggested tags which they can choose to refine their query. This was achieved by computing the euclidean distance between component planes of all tags to the component planes of the requested tags and choosing the tags with the lowest distance.

3.5.2 Finding movies based on tags. Based on a set of preferred and disfavoured tags it is possible to assign each movie a score by multiplying the weight of the requested tag by the value of its component in the tag genome of a movie. A high score indicates

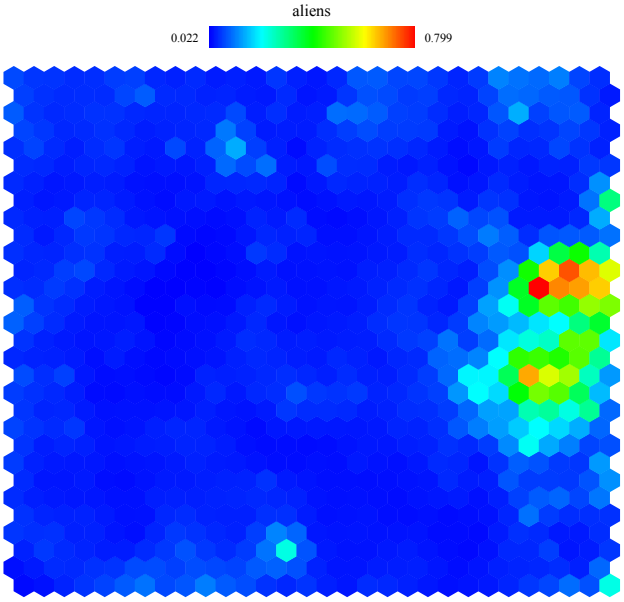


Figure 11: Component plane for the aliens tag. The aliens tag occupies similar regions compared to the science fiction tag.

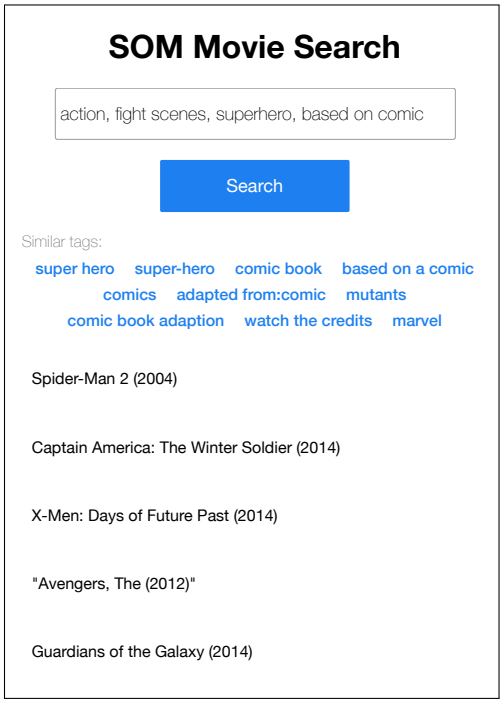


Figure 12: The user interface of the Recommendation System: The user enters a set of tags. The system responds with a set of similar tags to make the search more precise and a set of movies corresponding to the current search query.

that the tag genome correlates well with a given request while a low score indicates a low correlation. As the data set of movies is very large and can grow even further in a real world application, it may be very computationally expensive to compute a score for each movie. To work around this, the computation of scores was shifted from individual data points to nodes of a Self-Organizing Map. Instead of assigning each of the approximately 10,000 movies a score, a score was only computed for each node of the map, which results in a speed up, as the map only consists of 400 to 2,500 nodes. For the node with the highest score all movies for which the node is the best matching unit are returned as the search result. The mapping from nodes to movies can be computed in advance so responses for queries can be generated very fast.

4 CONCLUSION

Artificial Neural Networks can achieve promising results in many fields of machine learning. As shown, Self-Organizing Maps can be used to explore a data set such as the MovieLens data set and to find correlations by visual inspection. A search engine which can find related genres and movies based on a set of desired genres was developed in the form of a web site. A server responding to user queries uses a Self-Organizing Map in the background to improve response time over search algorithms which would inspect every movie to generate a search result. As the Self-Organizing Map preserves features of the data set such as topology and density, the quality of search results approaches an algorithm, which would generate scores for individual movies instead of map nodes.

REFERENCES

[1] Ajith Abraham. "Artificial neural networks". In: *Handbook of measuring system design* (2005).

[2] IA Basheer and M Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of microbiological methods* 43.1 (2000), pp. 3–31.

[3] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pp. 6645–6649.

[4] Heiga Zen and Haşim Sak. "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis". In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4470–4474.

[5] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. "Artificial neural networks: A tutorial". In: *Computer* 29.3 (1996), pp. 31–44.

[6] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*. Tech. rep. May 2014. URL: https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/.

[7] Chrislb. *File:ArtificialNeuronModel english.png*. July 2005. URL: https://commons.wikimedia.org/w/index.php?title=File:ArtificialNeuronModel%5C_english.png&oldid=233886112.

[8] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[9] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[11] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. Vol. 5. GMD-Forschungszentrum Informationstechnik, 2002.

[12] Headlessplatter. *File:Unfold through time.png*. June 2010. URL: https://commons.wikimedia.org/wiki/File:Unfold_through_time.png.

[13] Dubravko Miljković. "Brief Review of Self-Organizing Maps". In: *MIPRO 2017*. 2017.

[14] Teuvo Kohonen. "Self-organized formation of topologically correct feature maps". In: *Biological cybernetics* 43.1 (1982), pp. 59–69.

[15] Marc M Van Hulle. "Self-organizing maps". In: *Handbook of Natural Computing*. Springer, 2012, pp. 585–622.

[16] Simon Haykin. *Neural networks : a comprehensive foundation*. Upper Saddle River, N.J: Prentice Hall, 1999. ISBN: 0139083855.

[17] Alfred Ultsch. *U*-matrix: a tool to visualize clusters in high dimensional data*. Fachbereich Mathematik und Informatik Marburg, 2003.

[18] Juha Vesanto et al. "Enhancing SOM based data visualization". In: *Proceedings of the 5th International Conference on Soft Computing and Information/Intelligent Systems. Methodologies for the Conception, Design and Application of Soft Computing*. Vol. 1. 1998, pp. 64–67.

[19] Teuvo Kohonen. "The self-organizing map". In: *Neurocomputing* 21.1 (1998), pp. 1–6.

[20] F Maxwell Harper and Joseph A Konstan. "The movielens datasets: History and context". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5.4 (2016), p. 19.