UNIVERSIDAD NACIONAL DE COLOMBIA

Kernel-based Enhancement of General Stochastic Network for Supervised Learning

Diego Fabián Collazos Huertas

Universidad Nacional de Colombia
Faculty of Engineering and Architecture
Department of Electric, Electronic and Computing Engineering
Manizales, Colombia
2016

Kernel-based Enhancement of General Stochastic Network for Supervised Learning

Diego Fabián Collazos Huertas

Dissertation submitted as a partial requirement to receive the grade of:
Master in Engineering – Industrial Automation

Advisor:
Prof. Germán Castellanos-Domínguez, Ph.D.

Co-Advisor:
Andrés Marino Álvarez-Meza, Ph.D.

Academic Research Group:
Signal Processing and Recognition Group - SPRG

Universidad Nacional de Colombia
Faculty of Engineering and Architecture
Department of Electric, Electronic and Computing Engineering
Manizales, Colombia
2016

Mejoramiento de Redes Estocásticas Generales para Aprendizaje Supervisado

Diego Fabián Collazos Huertas

Disertación presentada como requisito parcial para recibir el grado de:
Maestría en Ingeniería – Automatización Industrial

Director:
Prof. Germán Castellanos-Domínguez, Ph.D.

Co-Director:
Andrés Marino Álvarez-Meza, Ph.D.

Grupo de Investigación Académica:
Grupo de Control y Procesamiento Digital de Señales - GCPDS

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departmento de Ingenieriía Eléctrica, Electrónica and Computación
Manizales, Colombia
2016

The Five Never:
Never give up
Never pretend
Never stand still
Never cling to the past
Never stop dreaming
Steve Jobs.

Acknowledgements

I would like to thank God first, for giving me wisdom and guide me always on my way. I would like to thank especially to my parents for their love and their advice in difficult times, to my brother and my family for being an unconditional support.

I would like to thank professor Germán Castellanos for his trust and orientation during this research. Also, I would like to express my deep gratitude to Andrés Meza Alvarez Marino, “the boss” co-director of this thesis, for his guidance and friendship during this research. Thanks to all the Signal Processing and Recognition Group (SPRG) of the National University of Colombia in Manizales, for the moments of leisure and academic discussion. Special gratitude to my friends, Juan David, David, Luisa, Ernesto, Santiago and Jorge Ivan, to my master partners David Insuasty and Fily for their support during this investigation. Thank you, Jessi, for being my life partner and my confidante.

Finally, I recognize that this research would not have been possible without the support and financial support of the Proyecto e-salud regalías: “Plataforma tecnologica para los servicios de teleasistencia, emergencias medicas, seguimiento y monitoreo permanente de pacientes y apoyo a los programas de prevencion”, Eje 3 - ARTICA.

Diego Fabián Collazos Huertas
2016

Abstract

In recent years, significant developments have been taking place in high-dimensional data analysis allowing to support a wide amount of applications in machine learning systems and signal processing among others. However, it is difficult to interpret the available information due to its complexity and a large amount of obtained features. Deep learning algorithms arise as a tool that deals this kind of issue by the extraction of representations (abstractions) from the data. Artificial neural networks based on deep learning algorithms, i.e., Deep Neural Networks (DNN) employ a feature hierarchy approach that increases complexity and abstraction. It makes DNN capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions. Besides, this provides a better representation, allowing faster learning and more accurate classification. However, DNNs as well as most of the artificial neural networks, are still vulnerable to *Over-fitting*. With respect to this, kernel functions appear as an alternative approach that in the first place, helps to high dimensional data analysis allowing enhancing representation and data interpretation, for supporting signal processing and machine learning systems. Moreover, kernel-based methods develop better-performing solutions by adapting the kernel to a given problem, which, allows dealing with over-fitting problem present in deep neural networks learning stage.

In this study, we propose a data representation framework based on kernel functions to enhance the performance of a kind of Deep Neural Networks, called General Stochastic Networks in supervised learning systems. Namely, the proposed framework is divided in two kernel-based methodologies, which aim to enhance network parameter setting according to data relationships. First, we develop an automatic architecture selection criterion based on kernel functions, that allows quantifying the optimal layer size preserving the joint input-output information. Thus, resulting hidden layer size are highlighted aiming to favor learning training stages in classification and object recognition tasks. This approach named Joint spectrum allows exploiting the joint data similarity for a given input sample set. Second, we introduce a supervised network Pre-training approach that highlights the relationship between hidden states and target information. Thus, a supervised pre-training approach based on a CKA-based function is introduced to learn a projection matrix, which encodes discriminate information from data to get a suitably hidden representation. So, we seek a pre-training method that captures main variations from the input distribution to support learning stage. Finally, an enhanced General Stochastic Network was developed to support supervised learning tasks. The proposal considers two strategies network parameter setting based on kernel functions, above mentioned, exploiting its main properties. The resulting network highlights relevant data dependencies and the user prior knowledge (supervised information). Thus, a generalizable network able to capture a lot of information from input data distribution and encode discriminant patterns is built. Along this study, the proposed network improvement using a kernel-based framework is applied to image data as an alternative to support classification systems and image-based object analysis. In fact, the introduced kernel-based framework improve, in most of the cases, supervised learning performances, supporting the analysis of a large amount of data using deep learning architectures.

Keywords: General stochastic network, Kernel-based learning, Supervised learning, Architecture selection, Network pre-training.

Resumen

En los últimos años, han tenido lugar avances significativos en el análisis de datos de alta dimensión que permiten soportar una amplia cantidad de aplicaciones en sistemas de aprendizaje automático y procesamiento de señales entre otros. Sin embargo, es difícil de interpretar la información disponible debido a su complejidad y a la gran cantidad de características obtenidas. Los algoritmos de aprendizaje profundo o *Deep learning* surgen como una herramienta que se ocupa de este tipo de problemas mediante la extracción de representaciones (abstracciones) a partir de los datos. Las redes neuronales artificiales basadas en algoritmos de aprendizaje profundo, es decir, las redes neuronales profundas o *Deep neural networks* (DNN) emplean un enfoque de jerarquía de característica que aumenta la complejidad y la abstracción.

Lo anterior hace a las DNNs capaces de manejar conjuntos de datos muy grandes, de alta dimensión con miles de millones de parámetros que pasan a través de funciones no lineales. Además, esto proporciona una mejor representación de los datos, lo que permite un aprendizaje más rápido y una clasificación más precisa. Sin embargo, las DNNs, así como la mayoría de las redes neuronales artificiales, siguen siendo vulnerables a sufrir un *over-fitting* o exceso de ajuste. Con respecto a esto, las *funciones kernel* aparecen como un enfoque alternativo que, en primer lugar, ayuda al análisis de datos de alta dimensión permitiendo mejorar la representación y la interpretación de los datos para el apoyo a sistemas de procesamiento de señales y de aprendizaje automático. Por otra parte, los métodos basados en kernel desarrollan soluciones que ofrecen mejores resultados mediante la adaptación del kernel a un problema dado, el cual, permite combatir el problema de over-fitting presente en la etapa de aprendizaje de las redes neuronales profundas.

En este estudio, proponemos un marco de representación de datos basado en funciones kernel para mejorar el rendimiento de una especie de redes neuronal profunda, llamada Red Estocástica General o Red Estocástica Generativa de aprendizaje supervisado. En concreto, el marco propuesto se divide en dos metodologías basadas en kernel, que tienen por objeto mejorar el ajuste de parámetros de la red de acuerdo con las relaciones entre los datos.

En primer lugar, se desarrolla un criterio de selección automática de la arquitectura de la red basado en funciones kernel, que permita cuantificar el tamaño óptimo de la capa preservando la información de conjunta entrada-salida. Por lo tanto, el tamaño de la capa oculta resultante se destaca con el objetivo de favorecer el aprendizaje de las etapas de entrenamiento en la clasificación y tareas de reconocimiento de objetos. Este enfoque llamado *Joint Spectrum* permite la explotación de la similitud de datos conjunta para un conjunto de muestras de entrada dado. En segundo lugar, se introduce un enfoque pre-entrenamiento supervisado de la red que resalta la relación entre los estados ocultos y la información objetivo. Por lo tanto, se introduce un enfoque pre-entrenamiento supervisado basado en una función kernel CKA para aprender una matriz de proyección, la cual codifica la información discriminante de los datos para obtener una representación oculta adecuada. Por lo tanto, buscamos un método de pre-entrenamiento que capta las variaciones principales de la distribución de entrada para apoyar la etapa de aprendizaje. Por último, una red estocástico generativa mejorada fue desarrollado para apoyar las tareas de aprendizaje supervisado. La propuesta considera las dos estrategias de sintonización de parámetros de la red basadas en funciones kernel antes mencionadas, explotando sus principales propiedades. La red resultante destaca dependencias de los datos relevantes y el conocimiento previo del usuario (información supervisada). Por lo tanto, se construye una red generalizable capaz de capturar una gran cantidad de información a partir de la distribución de datos de entrada y codificar patrones discriminantes.

A lo largo de este estudio, el mejoramiento de red propuesto usando un marco basado en el kernel se aplica a los datos de imagen como una alternativa para apoyar los sistemas de clasificación y análisis de objetos basado en la imagen. De hecho, el marco basado en funciones kernel introducido mejora, en la mayoría de los casos, el rendimiento en tareas de aprendizaje supervisado, apoyando el análisis de una gran cantidad de datos utilizando arquitecturas de aprendizaje profundo.

Palabras clave: Red estocástica general, Aprendizaje basado en kernel, Selección de la arquitectura, Pre-entrenamiento de la red.

Contents

Acknowledgements vii

Abstract ix

Resumen xi

List of Acronyms xv

List of Figures xvii

List of Tables xviii

I. Preliminaries 1

1. Introduction 2

1.1. Motivation . . . 2
1.2. Problem statement . . . 3
1.3. Literature review . . . 4
1.4. Objectives . . . 7
1.4.1. General objective . . . 7
1.4.2. Specific objectives . . . 7
1.5. Contributions of this work . . . 8

2. Mathematical preliminaries 10

2.1. Reproducing Kernel Hilbert Spaces in Machine Learning . . . 11
2.1.1. Reproducing kernel Hilbert spaces . . . 11
2.1.2. The covariance function . . . 13
2.1.3. Reproducing kernel Hilbert spaces in machine learning . . . 14
2.2. Artificial Neural Networks theory . . . 15
2.2.1. Multi-Layer Neural Networks . . . 15
2.2.2. Auto-Encoders . . . 17
2.3. Generalizing Denoising Auto-Encoders as Generative Models . . . 18
2.3.1. Definition and training . . . 18
2.3.2. Sampling . . . 18

2.3.3. Walkback Training . . . 19
2.3.4. Generalizing the denoising autoencoder to GSNs . . . 20
2.3.5. General Stochastic Networks for Unsupervised Learning . . . 20
2.3.6. General Stochastic Networks for Supervised Learning . . . 23

II. Materials and Methods **24**

3. Supervised Kernel Approach for Automated Learning using General Stochastic Networks **25**
3.1. General stochastic networks for supervised learning . . . 25
3.2. GSN architecture selection . . . 27
3.2.1. Motivation . . . 27
3.2.2. Selecting the hidden layer size using kernel functions . . . 27
3.3. GSN pre-training stage based on center kernel alignment . . . 28
3.3.1. Motivation . . . 28
3.3.2. Pre-training stage using kernel functions . . . 29

4. Experimental set-up **32**
4.1. Testing databases . . . 32
4.2. Setting of GSN training parameters . . . 34

5. Results and Discussion **36**
5.1. Results of GSN layer-size optimization . . . 36
5.2. Results of GSN pre-training . . . 39

6. Summary **45**

III. Final remarks **47**

7. Conclusions and future work **48**
7.1. Conclusions . . . 48
7.2. Future work . . . 49

8. Appendix **51**
8.1. Gradient descend-based optimization of CKA approach . . . 51

List of Acronyms

AE	AutoEncoders
ANN	Artificial Neural Network
CKA	Center Kernel Alignment
DAE	Denoising AutoEncoders
DBN	Deep Belief Network
DNN	Deep Neural Network
GSN	Generative Stochastic Network
ICA	Independent Component Analysis
JSR	Joint-Spectrum Regularization
MLP	Multi-Layer Perceptron
MSE	Minimum Squared Error
PCA	Principal Component Analysis
Rand	Random Weights
RBMs	Restricted Boltzmann Machines
RKHS	reproducing kernel Hilbert space
SAE	Stacked AutoEncoders
SGD	Stochastic Gradient Descent
SI	Sparse Initialization
SPRG	Signal Processing and Recognition Group

List of Figures

1-1. Relevant issues for a suitable topology choice in a DNN scheme. 4
1-2. Enhanced General Stochastic Network representation framework. The red boxes corresponds to the main contributions according to the relevant issues associated with the topology selection of the network. 9

2-1. kernel-based mapping. 14
2-2. Multi-layer neural network 16
2-3. Denoising Auto-Encoders: schematic representation of the process 17
2-4. Denoising AutoEncoders (DAE) Markov chain 21
2-5. GSN Markov chain 21
2-6. GSN Markov chain with multiple layers and backprop-able stochastic units . 22
2-7. dGSN Markov chain for input X_{t+0} and target Y_{t+0} with backprop-able stochastic units 23

3-1. Schematic representation of a Generative Stochastic Network (GSN) Markov chain with back-probable stochastic units. – Upward step. – downward step 26
3-2. L-curve criterian for $\widetilde{m}$ estimation 28
3-3. Diagram of the proposed CKA approach 30

4-1. Kernel-based topology optimization guideline used for GSN enhancement. . . 32
4-2. Images of the databases employed. 33

5-1. Resulting eigenvalues for the joint spectral decomposition calculated within a range of m values for each dataset. The highlighted red dot (●) denotes the $\widetilde{m}$ obtained by the regularization criterion. 36
5-2. GSN accuracy calculated within a range of m values for each dataset. The highlighted red dot (●) denotes the $\widetilde{m}$ obtained by the regularization criterion. 37
5-3. Visualization of the pre-training weights at the first layer for considered databases. In each plot, all weights of the 40 largest hidden units are plotted for AutoEncoders (top), Principal Component Analysis (PCA) (middle), and Center Kernel Alignment (CKA) (bottom) approaches. The gray values represent weight magnitudes. 40

5-4. GSN average accuracy and standard deviation vs iterations for the four pre-training approaches: Random, AutoEncoder, PCA-based projection, and CKA-based projection. Glass (top), Wine (middle), and Wdbc (bottom) datasets. 41

List of Tables

4-1. Summary of details for the UCI and image datasets. 33

5-1. Performed classification accuracy by each considered approach on the tested databases. Boldface stands for the best result of each learning case. Blank fields (–) correspond to not provided values or unfeasible calculations. 38

5-2. GSN classification accuracy for layer size selection criteria and pre-training approaches. The marked blank fields (–) corresponds to infeasible calculations where the layer dimension can not enable a pre-training matrix. 42

Part I.

Preliminaries

1. Introduction

1.1. Motivation

Machine learning can be broadly defined as a set of computational methods that uses experience or information obtained via interaction with the environment (observed data), in order to improve the performance or to make accurate predictions. In this sense, many learning algorithms are successfully deployed in a variety of fields and applications including text or document classification, natural language processing, speech recognition, computational biology, computer vision tasks, games and medical diagnosis [Mohri et al., 2012]. Unlike the conventional machine-learning techniques, representation learning includes a set of methods that allows a machine automatically discover the representations needed for detection or classification, from the raw data. These methods have turned out to be very good at discovering intricate structures in high-dimensional data. Deep learning is a representation learning method with multiple levels of representation, obtained by composing simple but non-linear modules where each of which transforms the representation at one level into a representation at a higher, slightly more abstract levels [LeCun et al., 2015].

In a local context, the Signal Processing and Recognition Group (SPRG) of the Universidad Nacional de Colombia has been working in the analysis of bio-signal data and biomedical image processing, in order to propose machine learning methodologies aimed to support the development of automatic systems for diagnostic assistance by means of representation learning [Orbes-Arteaga et al., 2015, Collazos-Huertas et al., 2015]. Deep learning has turned out to be very good at discovering intricate structures in high-dimensional data and it is, therefore, applicable to many tasks. For this reason, the SPRG is also interested in exploiting this property in high-dimensional problems such as, discover structures in magnetic resonance images or pattern recognition in an image sequence for video analysis [Cardenas-Pena et al., 2014, Molina-Giraldo et al., 2015].

Constructing a pattern recognition or machine learning based system demands accurate engineering. For example, when the processing of input signal that contains a set of specific properties without considering a suitable representation model, could lead to unstable performance results. Thus, is necessary to design a feature extractor system that transforms the data into a suitable internal representation allowing to the learning subsystem, often a classifier, to detect patterns in the input [LeCun et al., 2015].

For the local and global context, it is necessary to continue the development of methodologies that suitably represent the input distribution, with the additional benefit of improving the system performance, in terms of classification accuracy and data interpretation.

1.2. Problem statement

Machine learning is programming computers to optimize a performance criterion based on the data knowledge and environment information (e.g. the training set, or past experience), allowing to learn a representative model for any particular task. Based on the desired outcome or the type of inputs available during training, machine learning algorithms can be organized into different groups, such as supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, among others.

Artificial Neural Network (ANN) is an approach to machine learning, that is inspired by the structure and functional aspects of biological neural networks. The learning in ANN finds the model parameters, that makes the network assume the desired behavior. Depending on the problem and how neurons are connected, such behavior may require long and deep layers of computational stages, where each stage, transforms (often in a non-linear way) the aggregate activation of the network. To deal with the complexity imposed by the learning across many network stages, Deep learning algorithms have been used due to enables to computational models that are composed of multiple processing layers, learn representations of data with multiple levels of abstraction [Schmidhuber, 2015]. In particular, the neural networks based on deep learning or Deep Neural Network (DNN) exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones.

Despite the increasing complexity of the problems, deep learning based models have been well received by the research community due to improvements in the field of unsupervised learning of representations. Among the most relevant models found the Restricted Boltzmann Machines (RBMs) and Deep Belief Network (DBN) for digit classification task presented in [Hinton et al., 2006, Bengio et al., 2007], the feature extraction model based on AutoEncoders (AE) proposed in [Vincent et al., 2008] and the Sparse coding introduced in [Ngiam et al., 2011] evaluated on object and phone classification. In the same way, deep models have been obtained important improvements in supervised learning problems, such as speech recognition as exposed in [Seide et al., 2011], computer vision tasks and object identification using Deep Convolutional Networks by dropout algorithm proposed in [Krizhevsky et al., 2012].

A new DNN is introduced in [Bengio et al., 2013c] as the result of generalizing the generative view of DAE by introducing latent variables in the framework, called GSN. GSN are a deep neural network architecture based on learning the transition distribution of a Markov chain for estimating the data generating distribution. The combination of noise, a multi-layer feed-forward neural network, and walkback training makes GSN simplify the learning problem, be less like density estimation, and resemble more a supervised function approximation, with gradients that can be obtained by backprob-able stochastic units at each layer [Alain et al., 2015].

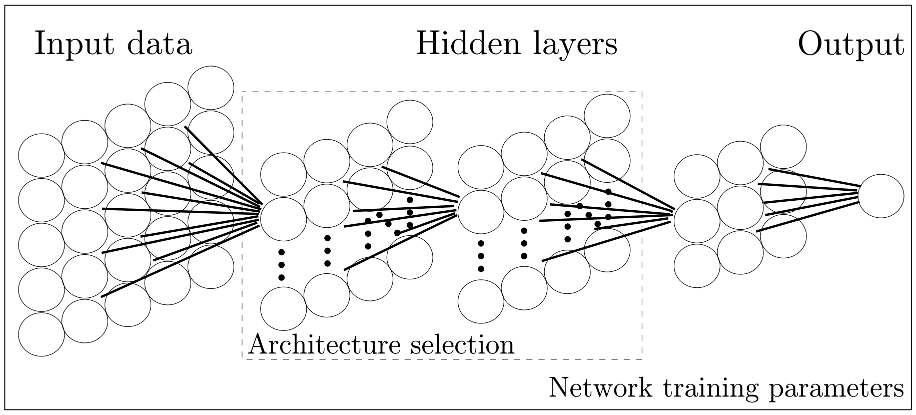


Figure 1-1.: Relevant issues for a suitable topology choice in a DNN scheme.

As usual, this kind of generative representation is obtained through a greedy-layerwise training procedure called contrastive divergence proposed in [Hinton et al., 2006]. In this case, the network layer learns the representation from the layer below by treating the latter as static input by walkback algorithm according to Bengio and coworkers [Bengio et al., 2013c]. Recently, an extension of generative stochastic networks to supervised learning of representations is presented in [Zöhrer and Pernkopf, 2014], in this case, the authors introduce a hybrid training objective considering a generative and discriminative cost function.

However, to bring proper results, deep neural networks require correct data pre-processing, architecture selection, and network training [Panchal et al., 2011]. The first aspect deals with learning data representations better extracting useful information, for which the generative stochastic model provides an alternative to the conventional maximum likelihood estimator, resulting in additional temporal information when building classifiers or other predictors. To open a wider field of new applications for GSN, however, the exploration of the remaining issues should be carried out [Zöhrer and Pernkopf, 2014]. In this sense, exist two relevant aspects associated with the topology selection which influences the network performance and deals with these issues (see fig. **1-1**): *i)* how to choose the number of layers and their size, and *ii)* how to deal the non-convexity of the training criterion for the parameter space searching.

1.3. Literature review

Throughout the literature, architecture selection, and network training problems in conventional neural networks as well as DNN, have been tackled from different perspectives leading to a myriad of methods and algorithms. Overall, most of the state-of-the-art techniques seek among other things, to decrease the network complexity, to improve the performance in terms of highest accuracy and minimal error, and to increase the learning speed.

One of the major challenging issues for researchers of neural learning is the network architecture selection, that is, how to choose the number of layers and their size, aiming at reaching the best system performance. Since the hidden neuron can influence the error on the nodes to which their output is connected, the excessive hidden neurons will cause overfitting; that is, the neural networks have overestimated the complexity of the target problem. By contrast, the system loses its capability to learn complex class boundaries correctly in the case of too small networks, making the training algorithm prone to converge to local minima [Ke and Liu, 2008]. Therefore, either case of a disproportionate number of neurons usually significantly reduces the classification performance. Overall, the network architecture influences the performance of used training algorithm through the following factors [Sheela and Deepa, 2013]: *i)* input and output dimensions, *ii)* number of training samples, and *iii)* noise injection method.

On a first attempt, the architecture selection can be accomplished by an exhaustive heuristic search that thoroughly examines all combinations, following a given rule of change for the layer sizes like linear [Sun, 2012] or exponential [Doukim et al., 2010]. The optimal architecture is chosen based on a predefined performance measure. Nevertheless, exhaustive approaches are very high time-consuming, limiting their use in practice. Less costly searching procedures have been proposed to reduce the computational load of heuristic strategies. Thus, the trial-and-error learning explores several variations of the number of hidden units until the desired performance is achieved [Hunter et al., 2012]. Also, the searching procedure can be speeded up using quantum parallelism and non-linear quantum operators [da Silva et al., 2016]. With either searching approach, a priori knowledge of the problem at hand and certain skills in the network field are required, often resulting suboptimal architectures [Rani B et al., 2012]. With the purpose to achieve a trade-off between the layer size and system performance, the heuristic pruning approaches had been developed that iteratively remove, from a starting architecture with too many neurons, the network parameters in agreement with an established relevance measure. In a simplistic case, the dropout pruning technique randomly omits a unit from a fully connected network with a probability of 0.5 until a lower bound on the performance is accomplished [Hinton et al., 2012]. A more elaborate approach cuts out the tree branches based on the entropy gain of the hidden units [Yuan et al., 2003]. Despite all these enhancements, the heuristic searching algorithms usually demand substantial computational resources due to the required cross-validation of multiple architectures [Dahl et al., 2013].

On the other hand, the analytical-based learning approaches had also been proposed (termed *rule-of-thumb*) to avoid the computational cost of searching algorithms, which derive the layer size as a function associating network architectures with datasets (input and output dimensions, the number of samples, among others). The following methods serve as examples of this strategy. In [Trenn, 2008], an explicit equation resulting from the sensitivity analysis is proposed for the hidden units of a Multi-Layer Perceptron (MLP). Hence, the layer size is calculated grounded on the desired approximation order and the number of inputs. In [Ke and Liu, 2008], the number of hidden layers, input dimensions, and samples are also included. Further, the geometric mean, computed between input and output sizes, is suggested to fix an appropriate number of hidden neurons [Shibata and Ikeda, 2009]. While these solutions are suitable for concrete cases, such analytics require more theoretical evidence to support the discovery of the optimal structure because the functions are hypotheses principally formulated for singular testing conditions and problems [Stathakis, 2009]. Moreover, the associating functions cannot be generalized and must be adjusted for each particular learning task. Nonetheless, the lower and upper bounds of the layer size can be confined to the number of linear functions required to separate the input sample set correctly as discussed in [Jiang et al., 2008]. Therefore, the analytical approaches can be employed for conducting the needed searching so that the number of alternative architectures is bounded and the tuning process can be speeded up.

The second key issue to improve for deep architectures is the parameter space searching due to the training criterion is non-convex and involves many local minima, getting worse for architectures with more than two or three levels [Erhan et al., 2009]. Although a straightforward procedure to cope with this issue is the use of multiple random initializations, it highly increases the time consumption [Bengio, 2012]. A more efficient random initialization is to establish the distribution and range of network parameters either empirically or to assume the characteristics of hidden units. In the latter case, the Glorot-style normalized initialization ensures that each neuron operates in the active region of its saturating function so that the forward and backward propagated variances are layer-wise fixed [Glorot and Bengio, 2010]. Yet, this approach promotes the fast parameter saturation during training of complex problems, decreasing the system performance [Glorot and Bengio, 2010]. With the purpose of avoiding the saturation of dense initializations, the Sparse Initialization (SI) technique randomly connects neurons of consecutive layers, draws the weights from a unit Gaussian, and sets the biases to zero [Martens, 2010]. Despite allowing the use of second order optimizations, SI is highly sensitive to the established activation function and scale constant, slowing down the learning speed [Sutskever et al., 2013].

To improve the initialization random approaches, the results obtained in [Erhan et al., 2010] suggest that unsupervised pretraining guides the learning towards basins of attraction of minima that support better generalization from the training data set. The layer-wise pre-training is the most used approach for finding a suitable initialization, aiming to extract a useful higher-level description of the output of the preceding layer of representation. To this end, a greedy layer-wise stage of unsupervised learning is firstly carried out, followed by a fine, supervised tuning [Hinton et al., 2006]. Particularly, this pre-training has been used for parameter optimization in [Bengio et al., 2007], using the RBMs as building blocks, and in [Vincent et al., 2008] for estimating the parameters of stacked DAE. As a result, either supervised learning machine regularizes the fine tuning, depending on the architecture depth and layer size [Erhan et al., 2010]. Nonetheless, the greedy principle underachieves if the conditional output distribution is not accurately associated with the input structure [Bengio et al., 2007]. On the other hand, some approaches have been discussed to boost the learning speed. Thus, the salient features can be extracted by Independent Component Analysis (ICA) to initialize the first MLP layer though it yields to local minimum solutions [Chen and Lu, 2013]. In contrast to the layer-wise approaches, the parameters learned simultaneously by a multilayer generative network can also be employed to initialize the training of a supervised feed-forward network, increasing the classification accuracy as discussed in [Mohamed et al., 2011]. Thus, the contractive regularization for pre-training two-layered auto-encoders forces the system to have small derivatives on the inputs, outperforming greedy methods in data generalization and classification accuracy [Schulz et al., 2015]. In general terms, the above discussed unsupervised pre-training approaches generate more useful hidden representations than the input space, but many of the resulting features may be irrelevant for discrimination tasks [Weston et al., 2012].

1.4. Objectives

1.4.1. General objective

Develop an improvement framework based on supervised kernel-based learning within a scheme of General Stochastic Networks, that allows setting network parameters disclosing relevant patterns from available input data. The developed framework must process the samples of the data in order to exploit their inherent structure and/or statistical distribution considering supervised information. In addition, this improvement framework must be adapted to cope with both network topology issues above-described, focusing on supervised learning tasks. Thus, the proposed kernel-based learning must summarize and capture the main input patterns to support classification and object recognition tasks, improving the network performance in terms of task accuracy, learning speed and generalization ability.

1.4.2. Specific objectives

- Develop a methodology using kernel-based functions that allows determining the hidden layer size in a supervised GSN, based on the spectral decomposition of the data using a regularization method. The aim is to take advantage of the spectral decomposition property to analyze the joint data similarity (i.e. samples and labels) with a regularization method and find the suitable architecture or an optimal boundary value of layer size that support a better learning stage. This methodology is useful because it reduces the required learning and testing time, in problems with high-dimensional data while avoiding the over-fitting issue.

- Propose a pre-training strategy based on supervised learning using kernel-based functions, that includes the input data information and prior knowledge regarding the studied process, e.g., supervised information. The aim is to extract relevant pair-wise sample relationships and to find a suitable data representation space or projection matrix. Specifically, this approach must be useful as a weight matrix initialization technique within a GSN. This approach improves the network learning process (i.e. more quickly) and favors the system accuracy.

- Integrate the two methodologies for network enhancement based on kernel functions into a GSN to support supervised learning tasks, e.g., object recognition and classification tasks. Thus, the network can be adapted regarding the studied process according to the needs of the problem, discovering relationships among data samples and their labels. In this way, we obtain a generalizable framework able to capture a lot of information from input data. We seek to improve the system performance in terms of recognition accuracy and to increase the learning speed.

1.5. Contributions of this work

The present work is done within kernel-based enhancement of General Stochastic Network for supervised learning. We aim to provide some strategies to set network parameters using kernel functions. We aim to optimize the layer size and develop a supervised pre-training method to support the weight initialization stage in order to learn efficiently data relations in multilayer learning systems. With this in mind, the framework can be adapted according to data statistical distribution and the learning procedure. In the following, the main contributions of the work are described:

- An Automatic estimation of the GSN layer size built on the L-curve regularization of the eigenvalues computed from the joint spectral decomposition of data. Specifically, we propose the Joint-Spectrum Regularization (JSR) that creates a tensor kernel to join the information of input and output samples into a single space.

- A supervised layer-wise pre-training to deal with the non-convex cost function. This stage makes use of the CKA that relies upon the linear projection that best matches the input samples with available labels, maximizing the information encoded by each layer. The resulting projections are used to initialize the weight matrices in a GSN.

- An enhanced General Stochastic Network is proposed for supervised learning tasks: classification and object recognition tasks. The kernel-based enhancement consists of implementing methodologies that: i) select the proper number of hidden neurons in the layers by regularization method and ii) learn a projection matrix based on Centered Kernel Alignment (CKA) strategy that improves the accuracy and increases the convergence speed. The full system seeks to exploit the contained data information using the prior knowledge regarding the studied learning task in order to encode discriminative features facilitating the classification task and avoiding over-fitting issues.

The plan of this thesis is as follow. First, each one of the proposed methodologies is evaluated separately within a GSN for supervised learning tasks. Later, the validation is carried out on datasets from machine learning repositories and the results are compared against baseline approaches in terms of system accuracy. Such results show that JSR provides the best trade-off between network complexity and system performance while the CKA-based pre-training improves both the learning speed and the classification accuracy. Finally, an enhanced GSN is introduced as the result of the combination of above-mentioned approaches. The guideline of the contributions in this work is described in fig. **1-2** according to the supervised kernel-based learning within a framework of General Stochastic Networks. The agenda of this thesis is organized as follows: We describe the mathematical preliminaries in Chapter 2, then our proposed approaches are presented in Chapter 3. Chapters 4 and 5 illustrate the results of carried out evaluations on six well-known datasets. Lastly, the conclusion about this work as well as ideas for a possible future investigation are presented in Chapter 7.

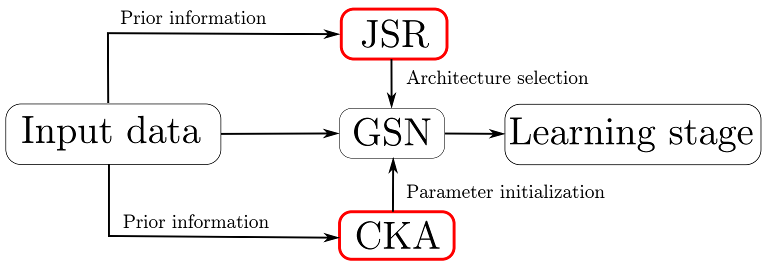


Figure 1-2.: Enhanced General Stochastic Network representation framework. The red boxes corresponds to the main contributions according to the relevant issues associated with the topology selection of the network.

2. Mathematical preliminaries

In this chapter, we provide a brief account of the introductory concepts of the mathematical theory of reproducing kernel Hilbert spaces in machine learning systems, artificial neural networks, as well as the representation learning schemes based on deep learning approaches (supervised and unsupervised). In particular, we define the deep learning from a representation learning point of view and then we describe the feature learning algorithms based on generative learning models. The contents of this chapter are based on the following papers, for reproducing kernel Hilbert spaces [Kreyszig, 1989, Parzen, 1959, Scholkopf and Smola, 2001] and the investigations carried out by Vincent [Vincent et al., 2008], Bengio [Bengio et al., 2013a, Bengio et al., 2013c, Bengio et al., 2013b, Bengio, 2009], Alain [Alain and Bengio, 2014], and Zohrer [Zöhrer and Pernkopf, 2014] within the framework of deep model representation..

Representation Learning: deep learning approach
Representation learning consists in learning representations of the data that make it easier to extract useful information when building classifiers or other predictors. Representation-learning algorithms attempt to characterize the data-generating distribution through the discovery of a set of features or latent variables whose variations capture most of the structure of the data-generating distribution. These feature learning algorithms can be stacked to form deeper and more abstract representations, i.e., deep architectures. *Deep learning* algorithms learn multiple levels of representation, where the number of levels is data-dependent. There are theoretical arguments and much empirical evidence to suggest that when they are well-trained, deep learning algorithms can perform better than their shallow counterparts, both in terms of learning features for the purpose of classification tasks and for generating higher-quality samples.
In recent years, have surged many features learning algorithms (supervised and unsupervised) that can be used to form deep architectures. In particular, it was empirically observed that layer-wise stacking of feature extraction often yielded better representations, e.g., in terms of classification error, quality of the samples generated by a probabilistic model, or in terms of the invariance properties of the learned features. Specifically, we explore the unsupervised feature learning algorithms, that are based on minimizing some form of reconstruction error, such as auto-encoder. Besides, we analyze the auto-encoder properties based on generative learning models for unsupervised and supervised learning tasks.

2.1. Reproducing Kernel Hilbert Spaces in Machine Learning

2.1.1. Reproducing kernel Hilbert spaces

Let $\mathscr{X}$ be a set and $\mathscr{F}$ be a vector space of functions from $\mathscr{X}$ to the field $\mathbb{F}$; in particular, let $\mathbb{F}{=}\mathbb{R}$. Then, there exits a reproducing kernel Hilbert space (RKHS) $\mathscr{H}$ on $\mathscr{X}$ over $\mathbb{R}$, if:

- $\mathscr{H}$ is a vector subspace of $\mathscr{F}$.
- $\mathscr{H}$ is endowed with an inner product, $\langle\cdot,\cdot\rangle_{\mathscr{H}}$, and is complete in the metric induced by it.
- For every $x{\in}\mathscr{X}$ and $f{\in}\mathscr{H}$, the linear evaluation functional $F_x : \mathscr{H} \to \mathbb{R}$, defined as $F_x(f) = f(x)$, is bounded.

From the Riez theorem [Kreyszig, 1989], it is known that for any bounded functional H on a Hilbert space $\mathscr{H}$, there exists a unique vector $h{\in}\mathscr{H}$ such that: $H(f){=}\langle h, f\rangle_{\mathscr{H}}$ for all $f{\in}\mathscr{H}$. In turn, for each evaluation functionals F_x there exist a corresponding vector $\kappa_x{\in}\mathscr{H}$. The bivariate function defined by:

$$\kappa(x, x') = \kappa_x(x') \tag{2-1}$$

is called a reproducing kernel for $\mathscr{H}$, with $x'{\in}\mathscr{X}$. So, it can be verified that

$$\kappa(x, x') = \langle\kappa_x, \kappa_{x'}\rangle_{\mathscr{H}} \tag{2-2}$$

and $\|F_x\|^2_{\mathscr{H}}{=}\|\kappa_x\|^2_{\mathscr{H}} = \langle\kappa_x, \kappa_{x'}\rangle_{\mathscr{H}}{=}\kappa(x, x)$, where $\|\cdot\|$ stands for the norm operator.

Let $\mathscr{H}$ be a RKHS on the set $\mathscr{X}$ with kernel κ. The linear span of $\{\kappa(x,\cdot) : x{\in}\mathscr{X}\}$ is dense in $\mathscr{H}$. This results from the fact that any function f orthogonal to the span of $\{\kappa(x,\cdot) : x{\in}\mathscr{X}\}$ must satisfy $\langle f, \kappa_x\rangle_{\mathscr{H}}$, and thus $f(x){=}0$.

Lemma 2.1.1. *Let $\{f_n\} \subset \mathscr{H}$, being $n{\in}\mathbb{N}$ an index counter. If $\lim_{n\to+\infty}\|f_n - f\|_{\mathscr{H}} = 0$, then $f(x){=}\lim_{n\to+\infty} f_n(x)$ for every $x{\in}\mathscr{X}$.*

Proof 2.1.1. *This is a simple consequence of the reproducing property and Cauchy-Schwarz inequality:*

$$|f_n(x) - f(x)| = |\langle f_n - f, \kappa_x\rangle_{\mathscr{H}}| \leq \|f_n - f\|_{\mathscr{H}}\,\|\kappa_x\|_{\mathscr{H}} \to 0$$

□

Proposition 2.1.1. *Let $\mathscr{H}_1$ and $\mathscr{H}_2$ be RKHS on $\mathscr{X}$ with kernels κ_1 and κ_2, respectively. If $\kappa_1(x,x'){=}\kappa_2(x,x')$ for all $x,x'{\in}\mathscr{X}$, then $\mathscr{H}_1{=}\mathscr{H}_2$ and $\|f\|_{\mathscr{H}_1}{=}\|f\|_{\mathscr{H}_2}$ for every f.*

Proof 2.1.2. *we can take $\kappa(x,x'){=}\kappa_1(x,x'){=}\kappa_2(x,x')$ and thus the $M_l{=}\text{span}\{\kappa_x{\in}M_l : x{\in}\mathscr{X}\}$ is dense in $\mathscr{H}_l$, and for any $f(x){=}\sum_n \alpha_n\kappa_{x_n}(x)$ there is no regard about whether f belongs to either M_1 or M_2. Note that $\|f\|^2_{\mathscr{H}_1}{=}\sum_{n,n'}\alpha_n\alpha_{n'}\kappa(x_n,x_{n'}){=}\|f\|^2_{\mathscr{H}_2}$, and thus $\|f\|_{\mathscr{H}_1}{=}\|f\|_{\mathscr{H}_2}$ for every $f{\in}M_1{=}M_2$. If $f{\in}\mathscr{H}_1$, then there is a sequence of functions $\{f_n\}\subset M_1$ that converge to f in norm. Since $\{f_n\}$ is Cauchy in M_1 is also Cauchy in M_2, so by completeness of $\mathscr{H}_2$ there exist $g{\in}\mathscr{H}_2$ such that $f_n\to g$. Then, by Lemma 2.1.1 we have that $f(x){=}\lim_{n\to+\infty} f_n(x){=}g(x)$ for every $x{\in}\mathscr{X}$, thus every $f{\in}\mathscr{H}_1$ is also in $\mathscr{H}_2$ and vice versa, and $\mathscr{H}_1{=}\mathscr{H}_2$. Finally, we can extend $\|f\|_{\mathscr{H}_1}{=}\|f\|_{\mathscr{H}_2}$ to all $\mathscr{H}_1$ and $\mathscr{H}_2$.*

□

Thus, two different RKHSs do not have the same reproducing kernel. The following theorem shows an alternative way to express the reproducing kernel of a RKHS $\mathscr{H}$.

Theorem 2.1.1. *Let $\mathscr{H}$ have reproducing kernel κ. if $\{e_\lambda : \lambda{\in}\Lambda\}$ is an orthonormal basis of $\mathscr{H}$, then:*

$$\kappa(x,x') = \sum_{\lambda\in\Lambda} e_\lambda(x)e_\lambda(x'), \tag{2-3}$$

where the series converges point-wise.

Proof 2.1.3. *For a fixed $\{x_n\}\subseteq\mathscr{X}$, we have:*

$$\sum_{n,n'=1}^{N}\alpha_n\alpha_{n'}\kappa(x_n,x_{n'}) = \left\langle \sum_{n=1}^{N}\alpha_n\kappa_{x_n}, \sum_{n'=1}^{N}\alpha_{n'}\kappa_{x_{n'}} \right\rangle_{\mathscr{H}} = \left\| \sum_{n=1}^{N}\alpha_n\kappa_{x_n} \right\|_{\mathscr{H}} \geq 0$$

□

Added to that, the Moore's Theorem is introduced, which is the converse to the above result and provides us a characterization of a positive definite function to be a sufficient condition for the function to be the reproducing kernel of some RKHS.

Theorem 2.1.2. *Let $\mathscr{X}$ be a set and $\kappa : \mathscr{X}\times\mathscr{X}\to\mathbb{R}$ be a positive definite function. Then, there exits a RKHS $\mathscr{H}$ of functions on $\mathscr{X}$, such that, κ is the reproducing kernel of $\mathscr{H}$.*

Proof 2.1.4. *Consider the functions $\kappa_x(x'){=}\kappa(x,x')$ and the space W spanned by the set $\{\kappa_x : x{\in}\mathscr{X}\}$. The following bilinear map $B : W\times W\to\mathbb{R}$:*

$$B\left(\sum_i \alpha_n\kappa_{x_n}, \sum_{n'}\beta_{n'}\kappa_{x_{n'}}\right) = \sum_{n,n'}\alpha_n\beta_{n'}\kappa(x_n,x_{n'}),$$

where $\alpha_n\beta_{n'}\in\mathbb{R}$, is well defined on W. To support the above claim, notice that if $f(x)=\sum_n \alpha_n\kappa_{x_n}(x)$ is zero for all $x\in\mathscr{X}$, then by definition $B(f,\kappa_x)=0$ for all x. Conversely, if $B(f,w)=0$ for all $w\in W$, then by taking $w=\kappa_x$ it can be seen that $f(x)=0$. Then, B is well defined.

Since κ is positive definite $B(f,f)\geq 0$ and we see that $B(f,f)=0$ if and only if $B(w,f)=0$ for all $w\in W$, therefore $f(x)=0$ for all $\mathscr{X}$. Now we have shown that W is a pre-Hilbert space with inner product B. Let $\mathscr{H}$ denote the completion of W, we need to show that every element of $\mathscr{H}$ is function on $\mathscr{X}$. Let $h\in\mathscr{H}$ be the limit point of a Cauchy sequence $\{f_n\}\subseteq W$. By Cauchy-Schwarz inequality:

$$|f_n(x)-f_{n'}(x)| = |B(f_n-f_{n'},\kappa_x)| \leq \|f_n-f_{n'}\|\kappa(x,x).$$

Therefore, the point-wise limit $h(x)=\lim_{n\to+\infty} f_n(x)$ is well defined. Concluding, let $\langle\cdot,\cdot\rangle_{\mathscr{H}}$ be the inner product on $\mathscr{H}$. Then, we have $\langle h,\kappa_x\rangle_{\mathscr{H}}=\lim_{n\to+\infty}\langle f_n,\kappa_x\rangle_{\mathscr{H}}=\lim_{n\to+\infty} B(f_n,\kappa_x)=h(x)$. Thus $\mathscr{H}$ is a RHKS with reproducing kernel κ.

□

Combining Proposition 2.1.1 with the Moore's Theorem (Theorem 2.1.2) shows the correspondence between RKHS's on the set $\mathscr{X}$ and positive definite functions on this set.

2.1.2. The covariance function

Consider a stochastic process $\{X(t):t\in\tau\}$, where $X(t)$ are real random variables defined on a probability space $(\Omega,\mathscr{B},\mathscr{P})$ with bounded second order moments, that is:

$$\mathbb{E}_t\left\{|X(t)|^2\right\} = \int\limits_{\Omega} |X(t)|^2 d\mathscr{P} < \infty, \tag{2-4}$$

where $\mathbb{E}\left\{\cdot\right\}$ stands for the expectation operator. Without loss of generality, we can consider random variables with zero mean, $\mathbb{E}_t\left\{X(t)\right\}=0$ for all $t\in\tau$. The covariance function is defined as:

$$R(t,t') = \mathbb{E}_{t,t'}\left\{X(t)X(t')\right\} = \int\limits_{\Omega} X(t)X(t')d\mathscr{P}, \tag{2-5}$$

where $t,t'\in\tau$. It is easy to verify that $R:\tau\times\tau\to\mathbb{R}$ is a positive definite function and therefore defines a RKHS of functions on τ. A result originally due to Loeve and presented by Parzen in [Parzen, 1959] showed a congruence map between the RKHS induced by the function R on L_2 space that corresponds to the completion of the span of the set $\{X(t):t\in\tau\}$ denoted by $L_2(X(t):t\in\tau)$.

Theorem 2.1.3. *Let $\{X(t) : t \in \tau\}$ be a random process with covariance kernel R. Then $L_2(X(t) : t \in \tau)$ is congruent with the RKHS $\mathscr{H}$ with reproducing kernel R. Furthermore, any linear map $\phi_R : \mathscr{H} \to L_2(X(t))$ which has the property that for any $f \in \mathscr{H}$ and any $t \in \tau$*

$$\mathbb{E}_t \{\phi_R(f) X(t)\} = f(t) \tag{2-6}$$

is the congruence from $\mathscr{H}$ onto $L_2(X(t))$, which maps $R(t, \cdot)$ into $X(t)$.

2.1.3. Reproducing kernel Hilbert spaces in machine learning

It is universally acknowledged that the study of positive definite kernels is a topic of interest for the machine learning community as a generalization of a well body of theory that has been developed for linear models. In this way, a positive definite kernel κ is an implicit way to represent the samples of the input space $\mathscr{X}$. Owing to there is a correspondence between κ and a RKHS of functions $\mathscr{H}$, the kernel can be understood as an indirect way to compute inner products between elements of a Hilbert space that are the result of mapping the elements of $\mathscr{X}$ to $\mathscr{H}$. So, there is a mapping function $\varphi : \mathscr{X} \to \mathscr{H}$ such that:

$$\kappa(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathscr{H}}. \tag{2-7}$$

Regarding this, the space $\mathscr{H}$ can be viewed as a feature space and φ is called the feature map. Consequently, by performing linear operations in $\mathscr{H}$ it is possible to perform nonlinear manipulations in the input space $\mathscr{X}$, however, there is no need to perform any explicit computations in $\mathscr{H}$ (see Figure 2-1).

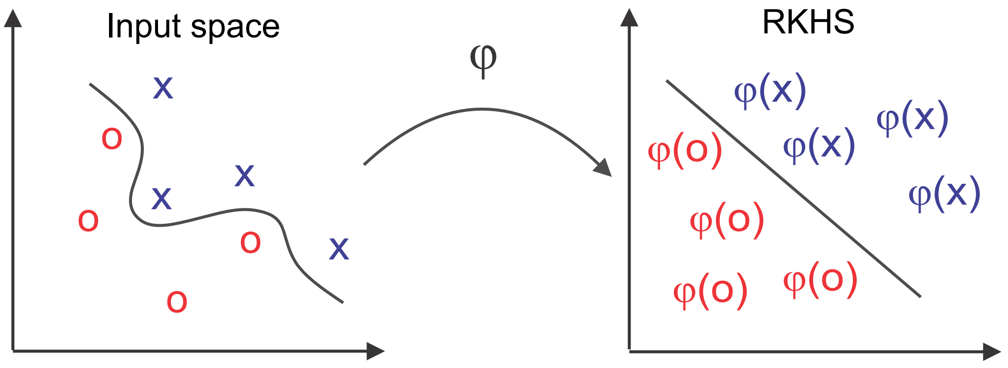


Figure 2-1.: kernel-based mapping.

Note that this idea is completely different to the congruence map introduced in Theorem 2.1.3. Then, an important property associated with the use of positive definite kernels in machine learning is the so-called representer theorem[Scholkopf and Smola, 2001]:

Theorem 2.1.4. *Let $\Omega : [0, +\infty) \rightarrow \mathbb{R}$ be a strictly monotonic increasing function, $\mathscr{X}$ be a set, and $\epsilon : (\mathscr{X} \times \mathbb{R}^2)^N \rightarrow \mathbb{R} \cup \infty$ be an arbitrary loss function. Then, each minimizer $f \in \mathscr{H}$ of the regularized risk functional:*

$$\epsilon\left((x_1, y_1, f(x_1)), \ldots, (x_N, y_N, f(x_N))\right) + \Omega\left(\|f\|^2_{\mathscr{H}}\right), \tag{2-8}$$

admits a representation of the form:

$$f(x) = \sum_{n=1}^{N} \alpha_n \kappa(x_n, x), \tag{2-9}$$

where each $y_n \in \mathbb{R}$ is a given output associated with the input $x_n \in \mathscr{X}$.

Proof 2.1.5. *Let $S = \text{span}\{\kappa(x_n, \cdot) : x_n \in \mathscr{X}, n \in [1, N]\}$ denotes the subspace of $\mathscr{H}$ spanned by the N training samples. Consider the solution $f \in \mathscr{H}$, this solution can be written as: $f = f_S + f_{S^\perp}$, where $f_S \in S$, $f_{S^\perp} \in S^\perp$, and $\perp$ stands for the orthogonal symbol. Consequently, $f(x_n) = f_S(x_n) + f_{S^\perp}(x_n) = f_S(x_n) + 0$. Now, for the second term of the regularized risk funtional:*

$$\Omega\left(\|f\|^2_{\mathscr{H}}\right) = \Omega\left(\|f_S\|^2_{\mathscr{H}} + \|f_{S^\perp}\|^2_{\mathscr{H}}\right),$$

since Ω is strictly monotonic increasing it is possible to see that the minimum will be achieved for $\|f_{S^\perp}\| = 0$, which implies that $f_{S^\perp} = 0$.

□

With this in mind, it is possible to conclude that the representer theorem basically states that the solution of the minimization of the regularized risk functional can be expressed in term of the so-called training sample $\{(x_n, y_n) : n \in [1, N]\}$. Therefore, it allows us to deal with problems that a first glance appear to be infinite dimensional. Nonetheless, the regularization does not prevent of having local multiple minima, such a property requires some extra conditions, namely, convexity.

2.2. Artificial Neural Networks theory

2.2.1. Multi-Layer Neural Networks

A typical set of equations for multi-layer neural networks is the following. As illustrated in figure 2-2, layer k computes an output vector $\boldsymbol{h}^k$ using the output $\boldsymbol{h}^{k-1}$ of the previous layer, starting with the input $\boldsymbol{x} = \boldsymbol{h}^0$,

$$\boldsymbol{h}^k = \tanh(\boldsymbol{b}^k + \boldsymbol{W}^k \boldsymbol{h}^{k-1}) \tag{2-10}$$

with parameters $\boldsymbol{b}^k$(a vector of offsets) and $\boldsymbol{W}^k$ (a matrix of weights). The tanh is applied element-wise and can be replaced by $sigm(u) = 1/(1+e^{-u}) = \frac{1}{2}(\tanh(u)+1)$ or other saturating non-linearities. The top layer output $\boldsymbol{h}^l$ is used for making a prediction and is combined with a supervised target y into a loss function $L(\boldsymbol{h}^l, y)$, typically convex in $\boldsymbol{b}^l + \boldsymbol{W}^l\boldsymbol{h}^{l-1}$. The output layer might have a non-linearity different from the one used in others layers, e.g., the softmax $\boldsymbol{x} = \boldsymbol{h}^0$,

$$\boldsymbol{h}_i^l = \frac{e^{\boldsymbol{b}_i^l + \boldsymbol{W}_i^l \boldsymbol{h}^{l-1}}}{\sum_j e^{\boldsymbol{b}_j^l + \boldsymbol{W}_j^l \boldsymbol{h}^{l-1}}} \tag{2-11}$$

where $\boldsymbol{W}_i^l$ is the ith row of $\boldsymbol{W}^l$, $\boldsymbol{h}_i^l$ is positive and $\sum_i \boldsymbol{h}_i^l = 1$. The softmax output $\boldsymbol{h}_i^l$ can be used as estimator of $P(Y = i|\boldsymbol{x})$, with the interpretation that Y is the class associated with input pattern $\boldsymbol{x}$. In this case one often uses the negative conditional log-likelihood $L(\boldsymbol{h}^l, y) = -\log P(Y = y|\boldsymbol{x}) = -\log \boldsymbol{h}_y^l$ as a loss, whose expected value over $(\boldsymbol{x}, y)$ pairs is to be monimized.

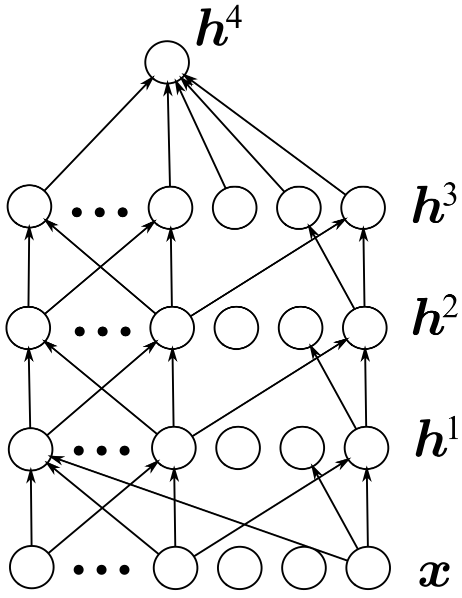


Figure 2-2.: Multi-layer neural network

Multi-layer neural network, typically used in supervised learning to make a prediction or classification, through a series of layers, each of which combines an affine operation and a non-linearity. Deterministic transformations are computed in a feedforward way from the input $\boldsymbol{x}$, through the hidden layers $\boldsymbol{h}^k$, to the network output $\boldsymbol{h}^l$, which gets compared with a label y to obtain the loss $L(\boldsymbol{h}^l, y)$ to be minimized as shown in 2-2.

2.2.2. Auto-Encoders

An autoencoder takes an input vector $\boldsymbol{x} \in [0,1]^d$, and first maps it to a hidden representation $\boldsymbol{y} \in [0,1]^{d'}$ through a deterministic mapping $\boldsymbol{y} = f_\theta(\boldsymbol{x}) = s(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, parameterized by $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$. $\boldsymbol{W}^{d' \times d}$ is a weight matrix, $\boldsymbol{b}$ is a bias vector and $s(\cdot)$ is an activation function e.g. sigmoid function $s(x) = \frac{1}{1+e^{-x}}$. The resulting latent representation $\boldsymbol{y}$ is then mapped back to a "reconstructed" vector $\boldsymbol{z} \in [0,1]^d$ in input space $\boldsymbol{z} = g_{\theta'}(\boldsymbol{y}) = s(\boldsymbol{W}'\boldsymbol{y} + \boldsymbol{y})$ with $\theta' = \{\boldsymbol{W}', \boldsymbol{b}'\}$. The weight matrix $\boldsymbol{W}'$ of the reverse mapping may optionally be constrained by $\boldsymbol{W}' = \boldsymbol{W}^T$, in which case the autoencoder is said to have *tied weights*. Each training $\boldsymbol{x}^{(i)}$ is thus mapped to a corresponding $\boldsymbol{y}^{(i)}$ and a reconstruction $\boldsymbol{z}^{(i)}$ for i training steps. The parameters of this model are optimized to minimize the *average reconstruction error*:

$$\begin{aligned} \theta^\star, \theta'^\star &= \arg\min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}) \\ &= \arg\min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(\boldsymbol{x}^{(i)}, g_{\theta'}(f_\theta(\boldsymbol{x}^i))) \end{aligned} \tag{2-12}$$

where L is a loss function such as the traditional squared error $L(\boldsymbol{x}, \boldsymbol{z}) = \| \boldsymbol{x} - \boldsymbol{z} \|^2$.

Denoising Auto-Encoders DAE

We will now to train it to reconstruct a clean "repaired" input from a *corrupted*, partially destroyed one. This is done by first corrupting the initial input $\boldsymbol{x}$ to get a partially destroyed version $\tilde{\boldsymbol{x}}$ by means of a stochastic mapping $\tilde{\boldsymbol{x}} \sim q_{\mathcal{D}}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$. The corrupted input $\tilde{\boldsymbol{x}}$ is the mapped, as with the basic auto-encoder, to a hidden representation $\boldsymbol{y} = f_\theta(\tilde{\boldsymbol{x}}) = s(\boldsymbol{W}\tilde{\boldsymbol{x}} + \boldsymbol{b})$ from which we reconstruct a $\boldsymbol{z} = g_{\theta'}(\boldsymbol{y}) = s(\boldsymbol{W}'\boldsymbol{y} + \boldsymbol{b}')$ (in figure 2-3 an example is presented for a input $\boldsymbol{x}$ such is corrupted to $\tilde{\boldsymbol{x}}$. In this way the auto-encoder maps it to $\boldsymbol{y}$ and attempts to reconstruct $\boldsymbol{x}$).

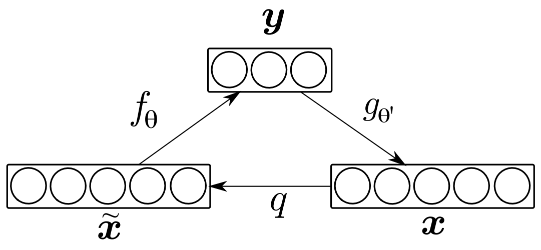


Figure 2-3.: Denoising Auto-Encoders: schematic representation of the process

2.3. Generalizing Denoising Auto-Encoders as Generative Models

We start of a probabilistic interpretation of DAEs, which is valid for any data type, any corruption process (so long as it has broad enough support), and any reconstruction loss (so long as we can view it as a log-likelihood). The basic idea is that if we corrupt observed random variable X into $\tilde{X}$ using conditional distribution $\mathcal{C}(\tilde{X}|X)$, we are really training the DAE to estimate the reverse conditional $P(X|\tilde{X})$. Combining this estimator with the known $\mathcal{C}(\tilde{X}|X)$, we show that we can recover a consistent estimator of $P(X)$ through a Markov chain that alternates between sampling from $P(X|\tilde{X})$ and sampling from $\mathcal{C}(\tilde{X}|X)$, i.e., encode/decode, sample from the reconstruction distribution model $P(X|\tilde{X})$, apply the stochastic corruption procedure $\mathcal{C}(\tilde{X}|X)$, and iterate.

2.3.1. Definition and training

Let $\mathcal{P}(X)$ be the data-generating distribution over observed random variable X. Let $\mathcal{C}$ be a given corruption process that stochastically maps an X to a $\tilde{X}$ through conditional distribution $C(\tilde{X}|X)$. The training data for the generalized denoising auto-encoder is a set of pairs $(X, \tilde{X})$ with $X \sim \mathcal{P}(X)$ and $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$. The DAE is trained to predict X given $\tilde{X}$ through a learned conditional distribution $P_\theta(X|\tilde{X})$, by choosing this conditional distribution within some family of distributions indexed by θ. The training procedure for the DAE can generally be formulated as learning to predict X given $\tilde{X}$ by possibly regularized maximum likelihood, i.e., the generalization performance that this training criterion attempts to minimize is

$$\mathcal{L}(\theta) = -E[\log P_\theta(X|\tilde{X})] \tag{2-13}$$

where the expectation is taken over the joint data-generating distribution

$$\mathcal{P}(X, \tilde{X}) = \mathcal{P}(X)\mathcal{C}(\tilde{X}|X). \tag{2-14}$$

2.3.2. Sampling

We define the following pseudo-Gibbs Markov chain associated with P_θ:

$$\begin{aligned} X_t &\sim P_\theta(X|\tilde{X}_{t-1}) \\ \tilde{X}_t &\sim \mathcal{C}(\tilde{X}|X_t) \end{aligned} \tag{2-15}$$

which can be initialized from an arbitrary choice X_0. This is the process by which we are going to generate samples X_t according to the model implicitly learned by choosing θ. We define $T(X_t|X_{t-1})$ the transition operator that defines a conditional distribution for X_t given X_{t-1}, independently of t, so that the sequence of X_t's forms a homogeneous Markov chain.

Algorithm 1 The generalized denoising Auto-Encoder training algorithm requires a training set or training distribution $\mathcal{D}$ of examples X, a given process $\mathcal{C}(\tilde{X}|X)$ from which one can sample, and with which one trains a conditional distribution $P_\theta(X|\tilde{X})$ from which one can sample.

```
repeat
• sample training example $X \sim \mathcal{D}$
• sample corrupted input $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$
• use $(X, \tilde{X})$ as an additional training example towards minimizing the expected value of
$-\log P_\theta(X|\tilde{X})$,e.g., by a gradient step with respect to $\theta$.
until convergence of trining (e.g., as measured by early stopping on out-of-sample negative
log-likelihood)
```

2.3.3. Walkback Training

Sampling in high-dimensional spaces (like in experiments below) using a simple local corruption process (such as Gaussian or salt-and-pepper noise) suggests that if the corruption is too local, the DAE's behavior far from the training examples can create spurious modes in the regions insufficiently visited during training. To alleviate this problem, We exploit knowledge of the currently learned model $P(X|\tilde{X})$ to define the corruption, so as to pick values of $\tilde{X}$ that would be obtained by following the generative chain: wherever the model would go if we sampled using the generative Markov chain starting at a training example X, we consider to be a kind of "negative example" $\tilde{X}$ from which the auto-encoder should move away (and towards X). This approach is very similar to the CD-k (Contrastive Divergence with k MCMC steps) procedure proposed to train RBMs [Hinton et al., 2006].
More precisely, the modified corruption process $\tilde{\mathcal{C}}$ we propose is the following, based on the original corruption process $\mathcal{C}$. We use it in a version of the training algorithm called *walkback*, where we replace the corruption process $\mathcal{C}$ of Algorithm 1 by the walkback process $\tilde{\mathcal{C}}$ of Algorithm 2. This also provides extra training examples (taking advantage of the $\tilde{X}$ samples generated along the walk away from X). It is called *walkback* because it forces the DAE to learn to walk back from the random walk it generates, towards the X's in the training set.

Algorithm 2 THE WALKBACK ALGORITHM is based on the walkback corruption process $\tilde{\mathcal{C}}(\tilde{X}|X)$, defined below in terms of a generic original corruption process $\mathcal{C}(\tilde{X}|X)$ and the current model's reconstruction conditional distribution $P(X|\tilde{X})$. For each training example X, it provides a sequence of additional training examples $(X, \tilde{X}^*)$ for the DAE. It has a hyper-parameter that is a geometric distribution parameter $0 < p < 1$ controlling the length of these walks away from X, with $p = 0.5$ by default. Training by Algorithm 1 is the same, but using all $\tilde{X}^*$ in the returned list L to form the pairs $(X, \tilde{X}^*)$ as training examples instead of just $(X, \tilde{X})$.

```
1: $X* \leftarrow X, L \leftarrow []$
2: Sample $\tilde{X}^* \sim \mathcal{C}(\tilde{X}|X^*)$
3: Sample $u \sim Uniform(0,1)$
4: **if** $u > p$ **then**
5:       Append $\tilde{X}^*$ to $L$ and **return** $L$
6: If during training, append $\tilde{X}^*$ to $L$, so $(X, \tilde{X}^*)$ will be an additional training example.
7: Sample $X^* \sim P(X|\tilde{X}^*)$
8: **goto** 2.
```

2.3.4. Generalizing the denoising autoencoder to GSNs

The denoising auto-encoder Markov chain is defined by $\tilde{X}_t \sim C(\tilde{X}|X_t)$ and $X_{t+1} \sim P_\theta(X|\tilde{X}_t)$, where X_t alone can serve as the state of the chain. The Generative Stochastic Network (GSN) framework generalizes this by defining a Markov chain with both a visible X_t and a latent variable H_t as state variables, of the form

$$\begin{aligned} H_{t+1} &\sim P_{\theta_1}(H|H_t, X_t) \\ \tilde{X}_{t+1} &\sim P_{\theta_2}(X|H_{t+1}) \end{aligned} \tag{2-16}$$

Denoising auto-encoders are thus a special case of GSNs. Note that, given that the distribution of H_{t+1} depends on a previous value of H_t, we find ourselves with an extra H_0 variable added at the beginning of the chain. This H_0 complicates things when it comes to training, but when we are in a sampling regime we can simply wait a sufficient number of steps to burn in.

2.3.5. General Stochastic Networks for Unsupervised Learning

The input distribution $P(X)$ is sampled to convergence in a Markov chain. In the case of the DAE, the transition operator first samples the hidden state H_t from a corruption distribution $C(H|X)$, and generates a reconstruction from the parameterized model, i.e the density $P_{\theta_2}(X|H)$.

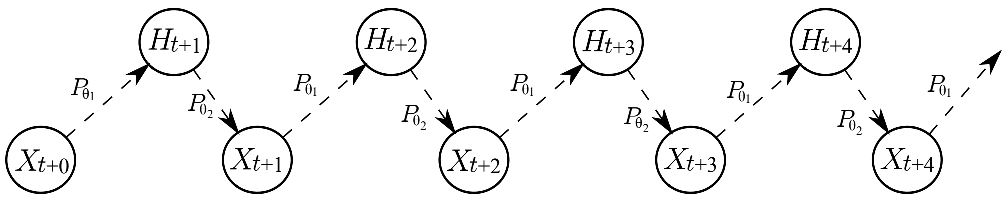


Figure 2-4.: DAE Markov chain

The resulting DAE Markov chain, shown in figure **2-4**, is defined as

$$H_{t+1} \sim P_{\theta_1}(H|X_{t+0}), \qquad X_{t+1} \sim P_{\theta_2}(X|H_{t+1}), \tag{2-17}$$

where X_{t+0} is the input sample X, fed into the chain at time step 0 and X_{t+1} is the reconstruction of X at time step 1. In the case of a GSN, an additional dependency between the latent variables H_t over time is introduced to the network graph. The GSN Markov chain is defined as follows:

$$H_{t+1} \sim P_{\theta_1}(H|H_{t+0}, X_{t+0}), \qquad X_{t+1} \sim P_{\theta_2}(X|H_{t+1}), \tag{2-18}$$

Figure **2-5** shows the corresponding network graph. This chain can be expressed with deterministic functions of random variables $f_\theta \supseteq \{\hat{f}_\theta, \check{f}_\theta\}$. In particular, the density f_θ is used to model $H_{t+1} = f_\theta(X_{t+0}, Z_{t+0}, H_{t+0})$, specified for some independent noise source Z_{t+0}, with the condition that X_{t+0} cannot be recovered exactly from H_{t+1}.

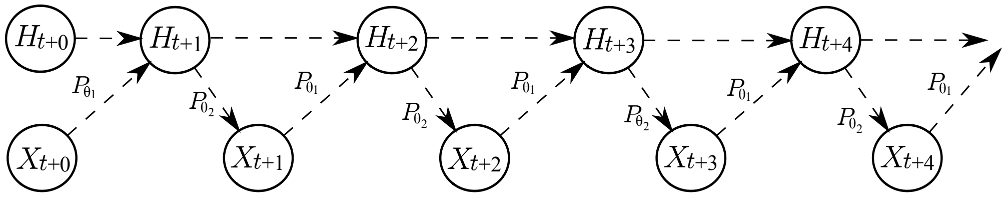


Figure 2-5.: GSN Markov chain

We introduce $\hat{f}^i_\theta$ as a back-probable stochastic non-linearity of the form $\hat{f}^i_\theta = \eta_{out} + g(\eta_{in} + \hat{a}_i)$ with noise processes $Z_t \supseteq \{\eta_{in}, \eta_{out}\}$ for layer i. The variable $\hat{a}^i$ is the activation for unit i, where $\hat{a}^i = \boldsymbol{W}^i I^i_t + \boldsymbol{b}^i$ with a weight matrix $\boldsymbol{W}^i$ and bias $\boldsymbol{b}^i$, representing the parametric distribution. It is embedded in a non-linear activation function g. The input I^i_t is either the realization x^i_t of observed sample X^i_t or the hidden realization h^i_t of H^i_t. In general, $\hat{f}^i_\theta(I^i_t)$ specifies an upward path in a GSN for a specific layer i. In the case of $X^i_{t+1} = \check{f}^i_{theta}(Z_{t+0}, H_{t+1})$ we define $\check{f}^i_\theta(H^i_t) = \eta_{out} + g(\eta_{in} + \check{a}_i)$ as a downward path in the network i.e. $\check{a}^i = (\boldsymbol{W}^i)^T H^i_t + \boldsymbol{b}^i$, using the transpose of the weight matrix $\boldsymbol{W}^i$ and the bias $\boldsymbol{b}^i$. This formulation allows to directly back-propagate the reconstruction log-likelihood $P(X|H)$ for all parameters $\theta \supseteq \{\boldsymbol{W}^0, \ldots, \boldsymbol{W}^d, \boldsymbol{b}^0, \ldots, \boldsymbol{b}^d\}$ where d is the number of hidden layers. In figure **2-5** the GSN includes a simple hidden layer. This can be extended to multiple hidden layers requiring multiple deterministic functions of random variables $f_\theta \in \{\hat{f}_\theta^0, \ldots, \hat{f}_\theta^d, \check{f}_\theta^0, \ldots, \check{f}_\theta^d\}$.

Figure 2-6 visualizes the Markov chain for a multi-layer GSN, inspired by the unfolded computational graph of a deep Boltzmann machine Gibbs sampling process.

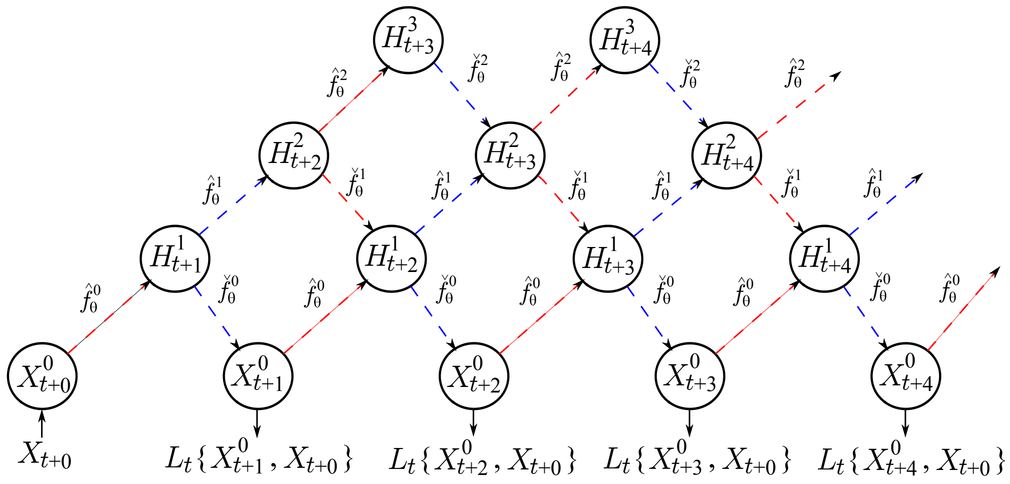


Figure 2-6.: GSN Markov chain with multiple layers and backprop-able stochastic units

In the training case, alternatively even or odd layers are updated at the same time. The information is propagated both upwards and downwards for K steps allowing the network to build higher order representations. An example for this update process is given in figure 2-6. In the even update (marked in red) $H^1_{t+1} = \hat{f}^0_\theta(X^0_{t+0})$. In the odd update (marked in blue) $X^0_{t+1} = \check{f}^0_\theta(H^1_{t+1})$ and $H^2_{t+2} = \hat{f}^1_\theta(H^1_{t+1})$ for $k = 0$. In the case of $k = 1$, $H^1_{t+2} = \hat{f}'_\theta(X^0_{t+1}) + \check{f}^1_\theta(H^2_{t+2})$ and $H^3_{t+3} = \hat{f}^2_\theta(H^2_{t+2})$ in the even update and $X^0_{t+2} = \check{f}^0_\theta(H^1_{t+2})$ and $H^2_{t+3} = \hat{f}^1_\theta(H^1_{t+2}) + \check{f}^2_\theta(H^3_{t+3})$ in the odd update. In case of $k = 2$, $H^1_{t+3} = \hat{f}^1_\theta(X^0_{t+2}) + \check{f}^1_\theta(H^2_{t+3})$ and $H^3_{t+4} = \hat{f}^2_\theta(H^2_{t+3})$ in the even update and $X^0_{t+3} = \check{f}^0_\theta(H^1_{t+3})$ and $H^2_{t+4} = \hat{f}^1_\theta(H^1_{t+3}) + \check{f}^2_\theta(H^3_{t+4})$ in the odd update. The cost function of a generative GSN can be written as:

$$C = \sum_{k=1}^{K} L_t\{X^0_{t+k}, X_{t+0}\}, \tag{2-19}$$

L_t is a specific loss-function such as the mean squared error (MSE) at time step t. In general any arbitrary loss function could be used (as long as they can be seen as a log-likelihood). X^0_{t+k} is the reconstruction of the input X^0_{t+0} at layer 0 after k steps. Optimizing the loss function by building the sum over the costs of multiple corrupted reconstructions is called *walkback training*. This form of network training leads to a significant performance boost when used for input reconstruction. The network is able to handle multi-modal input representations and is therefore considerably more favorable than standard generative models.

2.3.6. General Stochastic Networks for Supervised Learning

In order to make a GSN suitable for a supervised learning task we introduce the output Y to the network graph. In this case $L = \log P(X) + \log P(Y|X)$. Although the target Y is not fed into the network, it is introduced as an additional cost term. The layer update-process stays the same.

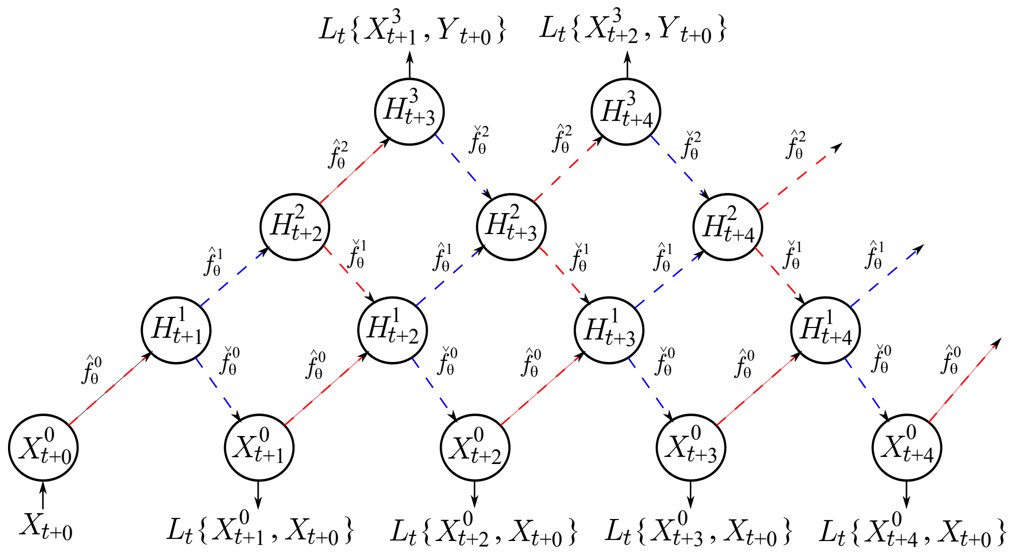


Figure 2-7.: dGSN Markov chain for input X_{t+0} and target Y_{t+0} with backprop-able stochastic units

We define the following cost function for a 3-layer dGSN:

$$C = \frac{\lambda}{K} \sum_{k=1}^{K} L_t\{X^0_{t+k}, X_{t+0}\} + \frac{1-\lambda}{K-d+1} \sum_{k=d}^{K} L_t\{H^3_{t+k}, Y_{t+0}\} \tag{2-20}$$

This is a non-convex multi-objective optimization problem, where λ weights the generative and discriminative part of C. The parameter d specifies the number of network layers i.e. depth of the network. Scaling the mean loss in 2-20 is not mandatory, but allows to equally balance both loss terms with $\lambda = 0.5$ for input X_{t+0} and target Y_{t+0} scaled to the same range. Again figure **2-7** shows the corresponding network graph for supervised learning with red and blue edges denoting the even and odd network updates. In general the hybrid objective optimization criterion is not restricted to $\langle X, Y \rangle$, as additional input and output terms could be introduced to the network.

Part II.

Materials and Methods

3. Supervised Kernel Approach for Automated Learning using General Stochastic Networks

3.1. General stochastic networks for supervised learning

Provided the input distribution $P(\mathcal{X})$ for which we only have empirical samples $\boldsymbol{X}\subset\mathcal{X}$, General stochastic networks (GSN) combines a multi-layer feed-forward neural network, noisy propagations, and *walkback* training for estimating the corresponding transition operator of a Markov chain. Figure 3-1 illustrates a GSN Markov chain for an L-layered deep network, where $\boldsymbol{X}_t\in\mathbb{R}^{N\times P}$ is the input matrix sampled at time instants $t\in T$ time, $\boldsymbol{S}_t^l\in\mathbb{R}^{N\times m_l}$ is the l-th hidden state matrix at time step t and $m_l\in\mathbb{N}$ is the size of the l-th layer so that $l\in L$ and $T>L$. Matrix $\boldsymbol{S}_t^l$ holds N vectors $\boldsymbol{s}_{t,i}^l\in\mathbb{R}^{m_l}$, mapping the input samples to the layer l.

Further, the dependency among available latent variables (hidden states) $\boldsymbol{S}_t^l$ are encoded in the GSN graph through the following set of upward/downward iterations [Zöhrer and Pernkopf, 2014].

$$\begin{cases}\boldsymbol{S}_t^l & = \varsigma_{out} + \phi^l(\boldsymbol{b}^l + \boldsymbol{S}_{t-1}^{l-1}\boldsymbol{W}^l + \varsigma_{in})\\ \boldsymbol{S}_t^{l-1} & = \varsigma_{out} + \varphi^l(\boldsymbol{a}^l + \boldsymbol{S}_t^l\left(\boldsymbol{W}^l\right)^\top + \varsigma_{in})\\ \boldsymbol{S}_t^0 & = \boldsymbol{X}_t\end{cases} \tag{3-1}$$

where $\boldsymbol{b}^l\in\mathbb{R}^{m_l}$ and $\boldsymbol{a}^l\in\mathbb{R}^{m_{l-1}}$ are the offset vectors, $\boldsymbol{W}^l\in\mathbb{R}^{m_{l-1}\times m_l}$ is the l-th linear projection, $\varsigma_{out}\in\mathbb{R}^{N\times m_l}, \varsigma_{in}\in\mathbb{R}^{N\times m_{l-1}}$ are independent noise sources, and the functions $\phi^l(\cdot)\in\mathbb{R}$ and $\varphi^l(\cdot)\in\mathbb{R}$ apply saturating, non-linear, element-wise operations.

We will define an L-layered GSN for classification through the following cost function:

$$v(\mathcal{X},\mathcal{Y})=\log\left(P(\mathcal{X})\right)+\log\left(P(\mathcal{Y}|\mathcal{X})\right) \tag{3-2}$$

where $P(\mathcal{Y}|\mathcal{X})$ is the conditional probability distribution between the input set and the output $\mathcal{Y}$ that is introduced to make a GSN suitable for a supervised learning task. Note that both distributions are sampled to guarantee convergence, namely, $\boldsymbol{X}\subset\mathcal{X}$ with $\boldsymbol{X}\in\mathbb{R}^{N\times P}$, holds

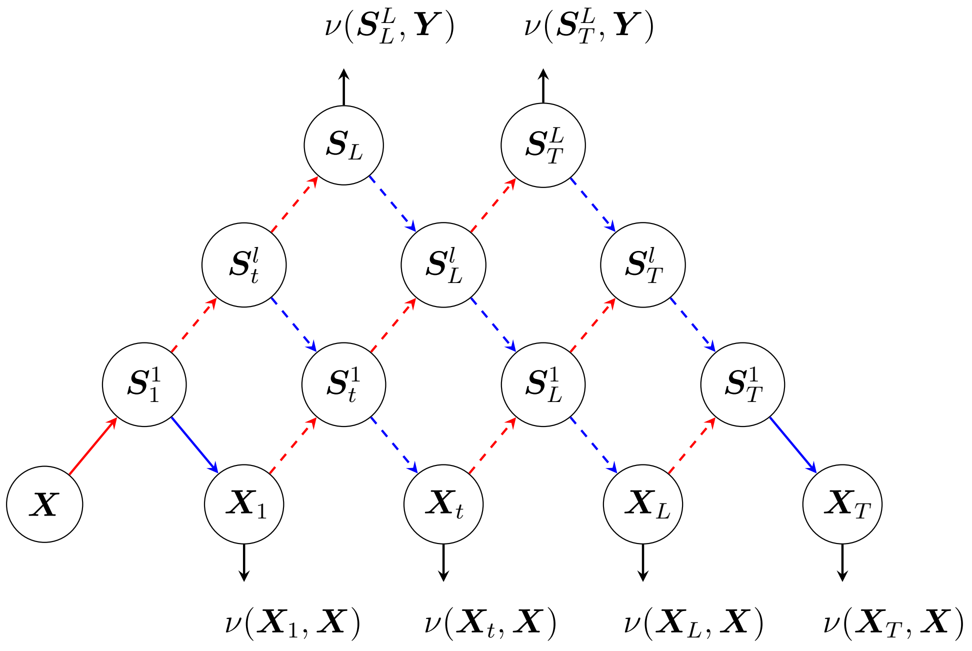


Figure 3-1.: Schematic representation of a GSN Markov chain with back-probable stochastic units. – Upward step. – downward step

N input vectors $\boldsymbol{x}_i{\in}\mathbb{R}^P$ $(i{\in}N)$ and $\boldsymbol{Y}{\subset}\mathcal{Y}$ with $\boldsymbol{Y}{\in}[0,1]^{N\times C}$ that contains N output vectors $\boldsymbol{y}_i{\in}[0,1]^C$ representing C mutually exclusive classes. Hence, the last layer is fixed to the output dimension, respectively, i.e. $m_L{=}C$.

Due to the target $\boldsymbol{Y}$ cannot be forward propagated through the graph network, it is instead introduced in the cost function so that the Markov chain is updated using the so-termed back-probable units [Alain et al., 2015]. Grounded on the *walkback* training approach, we compute for an L-layered GSN in eq. (3-2) the following hybrid multiobjective training criterion, dividing the cost function into a generative and discriminative:

$$\widetilde{v} = \zeta\mathbb{E}\left\{\nu\left(\boldsymbol{X}_t, \boldsymbol{X}\right) : \forall t{\in}[1\ldots T]\right\} + (1-\zeta)\mathbb{E}\left\{\nu\left(\boldsymbol{S}_t^L, \boldsymbol{Y}\right) : \forall t{\in}[L\ldots T]\right\} \tag{3-3}$$

where the real-valued parameter $\zeta{\in}[0,1]$ is the trade-off that searchers for a compromise between the generative (first term) and discriminative (second term) parts of eq. (3-3), notation $\mathbb{E}\left\{\cdot\right\}$ stands for the expectation operator, and $\nu(\cdot,\cdot)$ can be assumed as a loss-function, approximating the marginal and conditional log-likelihoods in eq. (3-2).

3.2. GSN architecture selection

3.2.1. Motivation

One of biggest challenges facing researchers is the selection of hidden neurons in Multi-layer neural networks. An exceeding number of hidden neurons made on the network deepens the local minima problem [Sun, 2012]. Besides, the system would have a large generalization error due to over-fitting and high variance. In contrast, if the number of hidden neurons becomes too small, the hidden units becomes unstable. For this reason, various criteria have been proposed to fix the number of hidden neurons by researchers during the last couple of decades. The techniques most popular are: trial rule, heuristic and exhaustive search, analytical approaches and pruning algorithms. Each one has specific properties in order to enhance the problems above mentioned. Nevertheless, these techniques can not be generalized because they are not always valid for all training cases.
Considering the importance of the choice of network architecture, we introduce in this section a tuning criterion for choose the proper number of the hidden nodes on a layer, based on the spectral decomposition of the data. We use the L-curve regularization of the eigenvalues of the tensor kernel matrix to obtain an optimum layer size. The main objective is to minimize error an to improve classification accuracy.

3.2.2. Selecting the hidden layer size using kernel functions

Here, we rely on a kernel-based similarity to quantify the hidden layer size value upon the joint sample set $\xi{=}\{(\boldsymbol{x}_i,\boldsymbol{y}_i){:}i,j{\in}N\}$ through the following eigen-spectrum analysis:

$$\boldsymbol{K}^{\xi} = \boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^{\top} \tag{3-4}$$

where the columns of $\boldsymbol{V}{\in}\mathbb{R}^{N\times N}$ and diagonal of $\boldsymbol{\Delta}{\in}\mathbb{R}^{N\times N}$ hold the eigenvectors and eigenvalues $\{\lambda_n{\in}\mathbb{R}^{+}{:}n{\in}N\}$ of the introduced kernel matrix $\boldsymbol{K}^{\xi}{\in}\mathbb{R}^{N\times N}$, respectively.

Generally speaking, we aim to quantify the optimal layer size that allows preserving the joint input-output information, encoded on the elements of $\boldsymbol{K}^{\xi}$, and computed as the following tensor product kernel:

$$k_{ij}^{\xi} = \kappa^{x}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)\kappa^{y}\left(\boldsymbol{y}_i,\boldsymbol{y}_j\right); \forall i,j{\in}N, \tag{3-5}$$

where $\kappa^{x}{:}\mathbb{R}^{N}{\times}\mathbb{R}^{N}{\mapsto}\mathbb{R}$ and $\kappa^{y}{:}[0,1]^{C}{\times}[0,1]^{C}{\mapsto}\mathbb{R}$ are the corresponding positive definite kernel functions defined in $\mathbb{R}^{N}$ and $\mathbb{R}^{C}$, respectively.

Therefore, to face a trade-off between system complexity and data information preserved in kernel matrix $\boldsymbol{K}^{\xi}$, we employ the widely-known L-curve regularization framework as below (see Figure **3-2**) [Hansen et al., 2007]:

$$\widetilde{m} = \underset{\forall n \in N}{\arg\min} \left\| \frac{\lambda_n}{\|\boldsymbol{\Delta}\|_\infty} - \frac{n}{N} \right\|_2, \tag{3-6}$$

where $\widetilde{m} \in \mathbb{N}$ is the estimated value for the optimal layer size and notation $\|\cdot\|_q$ stands for the l_q-norm.

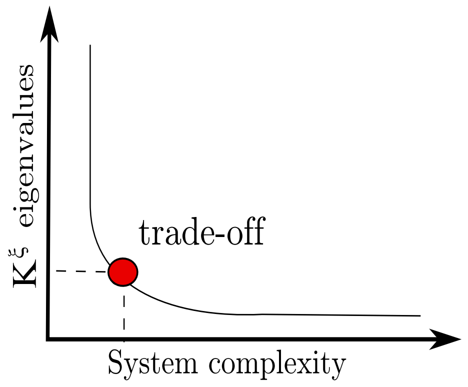


Figure 3-2.: L-curve criterian for $\widetilde{m}$ estimation

As a result, the supervised kernel-based analysis described above allows properly estimating the network architecture. Thus, the regularization criterion determines the optimal size $\widetilde{m}$ of the projected space from the spectral decomposition of the sample set ξ so that the joint information is maximally preserved. To this criterion, we have called Joint Spectrum Regularization (JSR).

3.3. GSN pre-training stage based on center kernel alignment

3.3.1. Motivation

In recent researches, DNN have shown significantly to outperform comparable but shallow competitors and often match or beat the state of the art in challenging artificial intelligence or AI related tasks such as computer vision, natural language processing and information retrieval. However, in virtually all instances of deep learning, the objective function is a highly non-convex function of the parameters, with the potential for many distinct local minimal in the model parameter space [Erhan et al., 2010].
To deal with this issue, an essential procedure for deep learning methods implementation is the initializing deep architecture (termed pre-training) that can be carried out by training a network to optimize directly only the supervised objective of interest, starting from

a set of randomly-initialized parameters. This procedure arose as a phase of the training strategies for deep architectures with the algorithms for training DBN and Stacked AutoEncoders (SAE), which are all based on a similar approach: greedy layer-wise unsupervised pre-training followed by supervised fine-tuning. However, this strategy performs poorly in practice [Vincent et al., 2010]. Particular examples that use unsupervised representation learning are the following: RBMs, AE, sparse AE [Ranzato et al., 2007], and the greedy layer-wise that is the most common approach that learns one layer of a deep architecture at a time [Bengio, 2012]. Although the unsupervised pre-training generates hidden representations that are more useful than the input space, many of the resulting features may be irrelevant for the discrimination task [Weston et al., 2012, Mohamed et al., 2011].
In this section we develop a pre-training stage in order to learn a projection matrix using the data distribution and prior data information. To this end, we compute the weight matrix $\widetilde{\boldsymbol{W}}^l$ maximizing the alignment between both centered kernels, label and projected data.

3.3.2. Pre-training stage using kernel functions

Kernel functions are bivariate measures of similarity based on the inner product between samples embedded in a Hilbert space, such as described above in Section 2.1.1. For a given domain $\mathscr{S}^l$ containing the input feature estimation of a given machine learning task, a kernel $\kappa_{S^l}{:}\mathscr{S}^l{\times}\mathscr{S}^l{\to}\mathbb{R}$ is assumed to be a positive-definite function, which defines an implicit mapping $\phi_{S^l}{:}\mathscr{S}^l{\to}\mathscr{H}_{S^l}$ that embeds any element $s^l{\in}\mathscr{S}^l$ into the element $\phi_{S^l}(s^l){\in}\mathscr{H}_{S^l}$ of some Reproducing Kernel Hilbert Space RKHS noted as $\mathscr{H}_{S^l}$.
Also, we set a positive definite kernel $\kappa^Y{:}\mathscr{Y}{\times}\mathscr{Y}{\mapsto}$over a target space $\mathscr{Y}$ related to the user prior knowledge, e.g., the label space. Then, the RKHS $\mathscr{H}_Y$ defines the implicit mapping $\phi_Y{:}\mathscr{Y}{\mapsto}\mathscr{H}_Y$, which maps any element $y{\in}\mathscr{Y}$ into the element $\phi_Y(l){\in}\mathscr{H}_Y$.

After estimation of $\widetilde{m}$ in Section 3.2.2, we obtain the set of kernel matrices $\{\boldsymbol{K}^l{:}l{\in}L\}$, accounting for similarity between a given pair of latent samples $\left\{\boldsymbol{s}^l_{t,i}, \boldsymbol{s}^l_{t,j}\right\}$ as follows:

$$k^l_{ij} = \kappa^l\left(\mathrm{d}_l\left(\boldsymbol{s}^l_{t,i}, \boldsymbol{s}^l_{t,j}\right)\right) \tag{3-7}$$

where $\mathrm{d}_l{:}\mathbb{R}^{m_l}{\times}\mathbb{R}^{m_l}{\to}\mathbb{R}^+$ is a certain distance operator introduced to implement the positive definite kernel function $\kappa^l(\cdot)$. Upon the assumption of linearity between the GSN layer transitions, we apply the Mahalanobis distance that is defined for P-dimensional spaces by the following inverse covariance matrix $\boldsymbol{W}^l\boldsymbol{W}^{l^\top}$:

$$\mathrm{d}_l\left(\boldsymbol{s}^l_{t,i}, \boldsymbol{s}^l_{t,j}\right) = \left(\boldsymbol{s}^l_{t,i} - \boldsymbol{s}^l_{t,j}\right)\boldsymbol{W}^l\boldsymbol{W}^{l^\top}\left(\boldsymbol{s}^l_{t,i} - \boldsymbol{s}^l_{t,j}\right)^\top \tag{3-8}$$

where $\boldsymbol{W}^l{\in}\mathbb{R}^{m_{l-l}\times m_l}$ holds the linear projection $\boldsymbol{z}^l_{t,i}{=}\boldsymbol{s}^l_{t,i}\boldsymbol{W}^l$, with $\boldsymbol{z}^l_{t,i}{\in}\mathbb{R}^{m_l}$, under assumption that $m_l{\leq}P$.

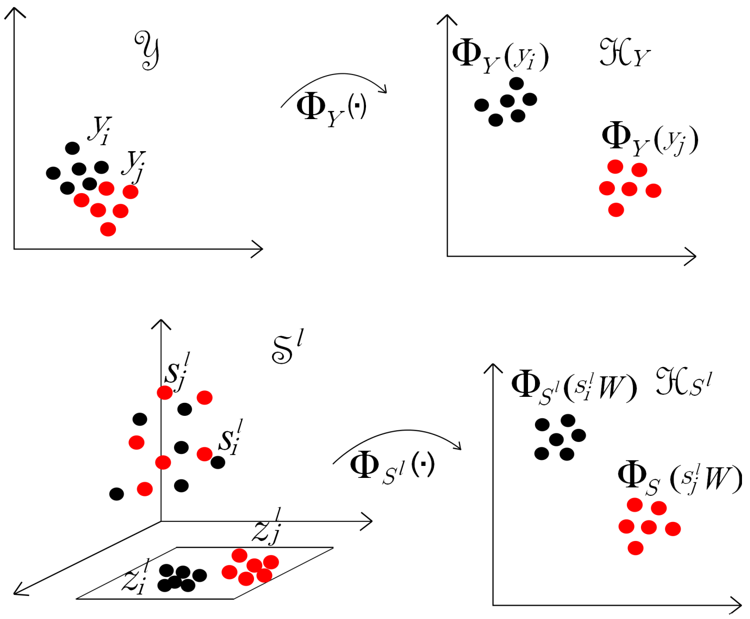


Figure 3-3.: Diagram of the proposed CKA approach

With the purpose of improving the system performance regarding the learning speed and classification accuracy, we introduce the available supervised knowledge into the pre-training stage. Consequently, we further enclose the output similarities in a matrix $\boldsymbol{K}^y$ with elements:

$$k_{ij}^y = \kappa^y\left(\boldsymbol{y}_i, \boldsymbol{y}_j\right). \tag{3-9}$$

However, we must encode the discriminative information to get a suitable $\boldsymbol{W}^l$, favoring the system performance. To this end, we also propose each matrix $\boldsymbol{W}^l$ to be learned by maximizing the similarity between $\boldsymbol{K}^l$ and $\boldsymbol{K}^y$ through the following real-valued function, $\rho\left(\cdot,\cdot\right)\in[0,1]$, that is termed as *centered kernel alignment* (CKA) [Brockmeier et al., 2014]:

$$\rho\left(\boldsymbol{K}^l, \boldsymbol{K}^y\right) = \frac{\left\langle \boldsymbol{H}\boldsymbol{K}^l\boldsymbol{H}, \boldsymbol{H}\boldsymbol{K}^y\boldsymbol{H}\right\rangle_F}{\left\|\boldsymbol{H}\boldsymbol{K}^l\boldsymbol{H}\right\|_F\left\|\boldsymbol{H}\boldsymbol{K}^y\boldsymbol{H}\right\|_F}, \tag{3-10}$$

where $\boldsymbol{H}=\boldsymbol{I}-N^{-1}\mathbf{1}\mathbf{1}^\top$ is a centering matrix ($\boldsymbol{H}\in\mathbb{R}^{N\times N}$), $\mathbf{1}\in\mathbb{R}^N$ is an all-ones vector, and notations $\langle\cdot,\cdot\rangle_F$ and $\|\cdot,\cdot\|_F$ stand for the Frobenius inner product and norm, respectively. In Figure **3-3** describes the introduced CKA approach.

Due to the CKA cost function in eq. (3-10) builds a set of projected features that match the best all provided target classes from each hidden state $\boldsymbol{S}_t^l$, we devise the following optimization problem to compute, at the end, the projection matrix:

$$\widetilde{\boldsymbol{W}}^l = \underset{\boldsymbol{W}^l}{\arg\max}\, \rho\left(\boldsymbol{K}^l, \boldsymbol{K}^y\right), \tag{3-11}$$

where the pre-trained $\widetilde{\boldsymbol{W}}^l$ initializes each l-th network layer. Section 8.1 describes the optimization details for Equation (3-11).

As a result, the supervised kernel-based analysis described above allows properly estimating an initial training setup. Thus, the maximization of the centered kernel alignment score, ρ, provides an assembly of discriminative linear projections $\{\boldsymbol{W}^l\}$, better matching the relations between hidden states $\boldsymbol{S}^l$ and target information $\boldsymbol{Y}$.

4. Experimental set-up

We validate the introduced kernel-based approach for GSN enhancement within a classification framework. In this regard, we follow the scheme shown in Figure 4-1 that comprises the following stages: *i)* Layer size optimization based on the eigenspectrum analysis of joint input-output kernel similarities, *ii)* Layer-wise pre-training of the linear projections through the alignment maximization between the latent sample kernel and supervised kernel (i.e., labels), *iii)* Fine parameter tuning by minimizing the GSN cost function. Figure 4-1 outlines a red box that includes both stages of topology configuration that are under investigation: Joint Spectrum Regularization (JSR) in the layer size optimization and Centered Kernel Alignment (CKA) in the layer-wise pre-training.

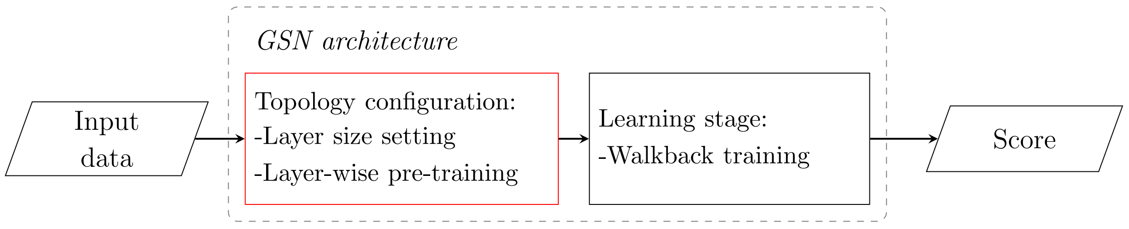


Figure 4-1.: Kernel-based topology optimization guideline used for GSN enhancement.

4.1. Testing databases

The proposed GSN approach is validated on six widely studied classification datasets: Three data collections from the well-known `UCI` machine learning repository[1], and three image collections used for object recognition:

i) `Glass` [Evett and Spiehler, 1987]. This collection is the results of a study of classification of types of glass (building windows and vehicle windows) that was motivated by the criminological investigation. Thus, a set of 6 types of glass were defined in terms of their oxide content (i.e. Na, Fe, K, etc).

ii) `Wine` [Lichman, 2013]. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determines the quantities of 13 constituents found in each of the three types of wines.

[1]https://archive.ics.uci.edu/ml/datasets.html

iii) `Wdbc` [Street et al., 1993]. This database describes the features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. The collection describes the characteristics of the cell nuclei present in the image. A set of 569 images were processed yielding a database of 30 dimensional points.

iv) `MNIST` [LeCun et al., 1995]. This collection holds 3000 handwritten digit images of size 28×28 pixels and 256 level grayscale of digits between 0 and 9. Data are evenly distributed with 300 representative images for every one of the ten digits, for which some examples are shown in Figure 4.2a. For testing, we extract 784 pixels from each non-processed image to feed the GSN inputs together with the provided label set that holds the same cardinal for the tested image data set.

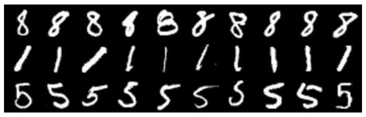
v) `Columbia COIL-20` [Nene et al., 1996]. This database includes 20 common household objects. Each object was placed on a turntable and photographed at 32×32 pixels every five degrees of rotation for a total of 72 views as shown in Figure 4.2b. The full set contains 1440 gray-scale images picked up for 20 objects.

vi) `Olivetti` [Samaria and Harter, 1994]. It holds 400 intensity-value pictures picked up for 40 individuals (males and females) with small variations in viewpoint, significant variation in expression, and the occasional addition of glasses. There are ten images sizing 112×92 per person, some examples are shown in Figure 4.2c.

Table **4-1** summarizes the dataset characteristics regarding the number of samples, features, and classes:

Description	*Dataset*					
	`Glass`	`Wine`	`Wdbc`	`MNIST`	`COIL-20`	`Olivetti`
#samples (N)	214	178	569	3000	1440	400
#features (P)	8	13	30	784	1024	10304
#classes (C)	6	3	2	10	20	40

Table **4-1**.: Summary of details for the UCI and image datasets.



a `MNIST`

b `Columbia COIL-20`

c `Olivetti`

Figure 4-2.: Images of the databases employed.

4.2. Setting of GSN training parameters

Particularly for image classification, we train the generative stochastic model by feeding P pixels in gray-scale from each non-processed image and estimating the provided label set that holds the same cardinal for the testing image. Namely, $P{=}\{784, 1024, 10304\}$ for `MNIST`, `COIL-20`, and `Olivetti` databases, respectively. Also, we carry out validation under the 3-fold cross-validation scheme for the `MNIST` and `Columbia COIL-20` databases, while a 5-fold cross-validation scheme is applied for the `Olivetti` collection. All GSN simulations are then executed on a GPU with the help of the mathematical expression compiler Theano[2] for the following parameter setting:

i) Number of layers: Two hidden layers ($L{=}2$) with $m_1{=}m_2{=}\widetilde{m}$ for all experimental settings.

ii) Loss-function $\nu(\cdot,\cdot)$ for approximating the learned marginal and conditional likelihoods in Equation (3-2). We employ the Minimum Squared Error (MSE) since its properties have been widely studied:

$$\nu(\boldsymbol{X}_t, \boldsymbol{X}) = \mathbb{E}\left\{\|\boldsymbol{x}_{t,i} - \boldsymbol{x}_{0,i}\|_2^2 : \forall i \in [1, \ldots, N]\right\} \tag{4-1}$$

iii) GSNs (see Equation (3-3)) and CKA (Equation (3-11)) optimization. Due to its efficiency on large scale problems and ease of implementation, we use the Stochastic Gradient Descent (SGD) [Bottou, 2010], for which the following working parameters are fixed: Learning rate is 0.1 for `MNIST` dataset as suggested in [Zöhrer and Pernkopf, 2014], and the value 1 is empirically set for `COIL-20` and `Olivetti`. The batch size is 1, the momentum term is 0.9, and the multiplicative annealing factor is 0.99.

iv) Activation function of the hidden states. `tanh` is chosen that is described as follows:

$$\phi^l(u) = \frac{\exp(2u) - 1}{\exp(2u) + 1}, \quad \forall l \in [1, \ldots, L-1]. \tag{4-2}$$

For the output layer ($l{=}L$), we choose function `softmax` that in our case operates like a classifier so that:

$$\phi^L(u_c) = \frac{\exp(u_c)}{\sum_{c'=1}^{C} \exp(u_{c'})}, \quad \forall c \in [1, \ldots, C] \tag{4-3}$$

v) Fine tuning of parameters. This optimization procedure is accomplished using walkback training approach with the rule $T{=}2L{=}4$ that is sufficient for convergence [Zöhrer and Pernkopf, 2014]. Note that we do not apply pre- nor post-activation noise in the network training setup since our goal is to test the performance of pre-training approaches.

[2]http://deeplearning.net/software/theano/

In the case of kernel functions, the following setup is considered:

i) For the layer size estimation (see Equation (3-6)), we encode the nonlinear dependencies among input-output variables computed as a tensor product kernel for the Gaussian and delta function-based kernels, respectively described as follows:

$$\kappa^x(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/2\sigma_x^2\right) \tag{4-4a}$$

$$\kappa^y(\boldsymbol{y}_i, \boldsymbol{y}_j) = \delta(\boldsymbol{y}_i - \boldsymbol{y}_j) \tag{4-4b}$$

where $\delta(\boldsymbol{y}_i - \boldsymbol{y}_j)$=1 if $\boldsymbol{y}_i$=$\boldsymbol{y}_j$, otherwise, $\delta(\boldsymbol{y}_i - \boldsymbol{y}_j)$=0, and the bandwidth $\sigma_x \in \mathbb{R}^+$ is computed as the median value of the input set L_2 distances.

ii) Layer-wise pre-training of the weighting matrices (see Equation (3-11)). The Gaussian kernel is also considered in Equation (3-7) as follows:

$$\kappa^l\left(\mathrm{d}_l\left(\boldsymbol{s}_{t,i}^l, \boldsymbol{s}_{t,j}^l\right)\right) = \exp\left(-\mathrm{d}_l^2\left(\boldsymbol{s}_{t,i}^l, \boldsymbol{s}_{t,j}^l\right)/2\sigma_l^2\right) \tag{4-5}$$

where $\sigma_l \in \mathbb{R}^+$ is adjusted to the median value of the Mahalanobis distances $\mathrm{d}_l\left(\boldsymbol{s}_{t,i}^l, \boldsymbol{s}_{t,j}^l\right)$.

5. Results and Discussion

5.1. Results of GSN layer-size optimization

In this case, we analyze the GSN performance, in terms of classification accuracy and network complexity, achieved by the proposed JSR as architecture selection. To this end, we firstly estimate the spectral decomposition eigenvalues of Equation (3-4). On the basis of Equation (3-6), we then calculate the hidden layer size, yielding $\widetilde{m}=\{6, 11, 11, 41, 50, 45\}$ for `Wine`, `Glass`, `Wdbc`, `MNIST`, `COIL-20`, and `Olivetti`, respectively. Note that JSR selects the largest eigenvalues in all cases as shown in Figure **5-1**, meaning that it holds most of information using the lowest number of components.

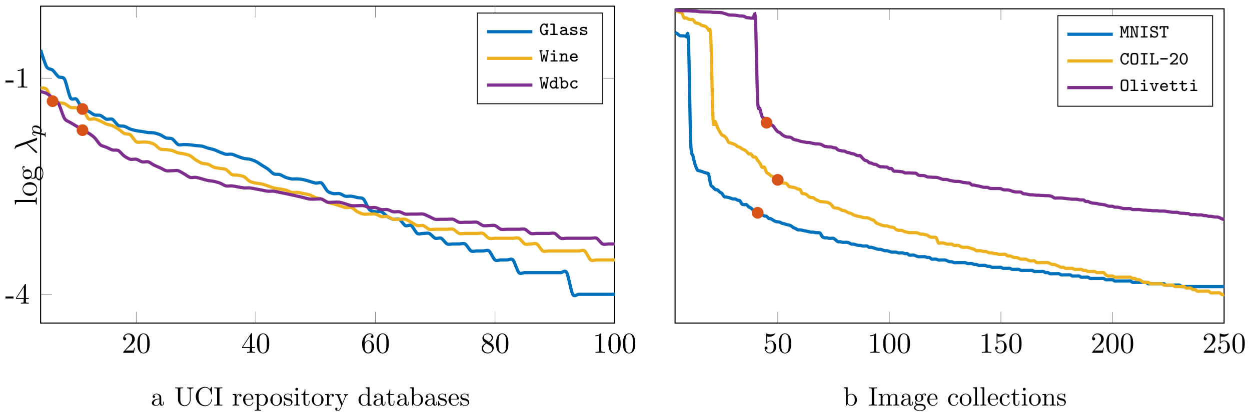


Figure 5-1.: Resulting eigenvalues for the joint spectral decomposition calculated within a range of m values for each dataset. The highlighted red dot (●) denotes the $\widetilde{m}$ obtained by the regularization criterion.

As a means to evaluate the use of JSR for estimating the GSN layer size, we contrast its performed classification accuracy with the conventional exhaustive search along with three widely used analytic selection strategies: Linear dependency estimation [Li et al., 1995], Geometric mean [Shibata and Ikeda, 2009], and Theoretical stability analysis [Ke and Liu, 2008]. Also, the comparison is carried out with two semi-supervised classification approaches that have network architecture similar to the GSN learning approach, and their performance has been reported on the same data tested in this paper. Namely, a deep-learning baseline method that feeds a transductive semi-supervised maximum margin clustering with a non-linear data representation (TSN) [Chen, 2015], and the reference network-based classification, or TAGnet, that emulates the sparse coding-based clustering pipeline using a feed-forward network structure (just for testing of image databases) [Wang et al., 2015].

Regarding the exhaustive searching procedure, validation is carried out within a framed range of layer sizes, namely, m=[4, . . . , N] for the `UCI` repository and m=[4, . . . , 250] for the image data sets. As seen in Figure **5-2**, the JSR criterion provides rather a small number of neurons m (marked with a red dot), allowing to reach a performance that is comparable to the highest accuracy obtained by the exhaustive search in almost all databases. Note that the computed layer sizes are bigger for image collections than UCI databases due to the larger variability present in the image classes. However, the estimated small m is not enough to reach the best performance of the exhaustive search for `glass` collection that has the most complicated data structure. Therefore, the optimal number of neurons that is the highest accuracy for which the least network complexity is achieved depends on the class distribution.

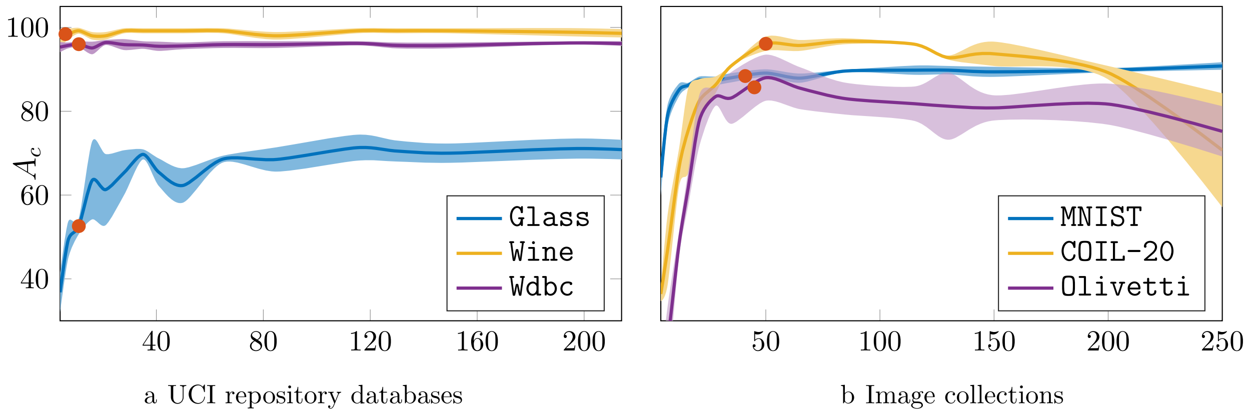


Figure 5-2.: GSN accuracy calculated within a range of m values for each dataset. The highlighted red dot (●) denotes the $\widetilde{m}$ obtained by the regularization criterion.

Table **5-1** summarizes the performance, regarding the estimated number of hidden neurons $\widetilde{m}$ and achieved classifier accuracy A_c, accomplished by the other compared learning approaches. Note that we keep the same setup of the TSN testing reported in the literature. Specifically, we consider a single-layered network and m=64 for UCI datasets, a single-layered network and m=100 for `COIL-20`, and a three-layered network ($\boldsymbol{m}$=[400, 200, 100]) for `MNIST`. Due to the label information is not employed, both semi-supervised classification approaches produce the worst accuracy for all databases (besides TSN in `Wine` data) despite they demand more complex architectures. As expected, the incorporation of label information leads the analytical approaches to improve the reached GSN accuracy. However, the classifier performance depends on the added number of neurons. Thus, while the theoretical stability analysis demands a larger $\widetilde{m}$ to reach a better accuracy, the linear dependency estimation demands the lowest $\widetilde{m}$ but yields the worst accuracy. Note that very high layer sizes overtrain the network, yielding a performance dropping as it is the case for the theoretical stability analysis approach in `COIL-20`. By contrast, the proposed JSR procedure outperforms above approaches attaining the highest accuracy at the smallest network complexity with the benefit of avoiding the network overtraining and time-consuming exhaustive search.

Approach	*Layer size*	m	A_c	m	A_c	m	A_c
UCI database		`Glass`		`Wine`		`Wdbc`	
[Li et al., 1995]	$\sqrt{1+8P}-1)/2$	4	36.8±4.7	4	97.5±2.1	7	95.7±0.7
[Ke and Liu, 2008]	$(P+\sqrt{N}/L)$	**11**	52.6±1.9	13	**98.4±0.9**	26	**95.9±1.3**
[Shibata and Ikeda, 2009]	$\sqrt{PC}$	7	49.2±4.4	6	98.4±1.3	7	95.7±0.7
[Chen, 2015]	Empirically set	64	50.9	64	98.8	64	91.5
JSR	Equation (3-6)	**11**	**52.6±1.9**	**6**	98.4±1.3	**11**	**96.0±0.7**
Image collection		`MNIST`		`COIL-20`		`Olivetti`	
[Li et al., 1995]	$\sqrt{1+8P}-1)/2$	**39**	87.3±1.2	**44**	94.7±1.1	143	85.0±1.7
[Ke and Liu, 2008]	$(P+\sqrt{N}/L)$	413	**91.0±1.1**	527	37.3±1.2	5160	–
[Shibata and Ikeda, 2009]	$\sqrt{PC}$	88	89.8±1.0	143	**96.2±0.7**	641	**95.0±2.6**
[Chen, 2015]	Empirically set	100	78.0	–	68.0	–	–
[Wang et al., 2015]	Convolutional Net	–	69.22	–	89.91	–	–
Exhaustive search	–	372	91.0±1.1	93	96.2±0.7	596	95.0±2.6
JSR	Equation (3-6)	41	88.4±0.9	50	**96.1±1.7**	**45**	83.2±6.2

Table **5-1**.: Performed classification accuracy by each considered approach on the tested databases. Boldface stands for the best result of each learning case. Blank fields (–) correspond to not provided values or unfeasible calculations.

Discussion

In this work, we propose an approach to enhance General Stochastic Networks using supervised kernel-based learning for classification tasks. In this regard, we introduce an automatic tuning criterion for the layer size, aiming at enhancing network configuration and performance. From the obtained results upon datasets from machine learning repositories, some important aspects of implementing are to be considered, which we comment in more detail below.

The layer size setting is the first stage of GSN topology configuration under study. For achieving the best trade-off between the network complexity and system performance, we propose the L-curve regularization that is intended to provide the smallest number of representative components. Also, we propose the eigenvalue extraction from the tensor kernel matrix that relies upon the spectral decomposition of the joint input-output sample set, maximally preserving the mutual information and making more discriminating the obtained decomposition. Validation on the testing data shows that the introduced Joint Spectrum Regularization (JSR) allows improving the network performance regarding classification accuracy and network complexity as compared with other state-of-the-art architecture selection

approaches. Another plus is that the proposed JSR has been developed as an automatic architecture selection strategy, taking advantage of the free-parameter procedures for the used regularization and kernel alignment.
Although JSR provides the best accuracy for the `Glass` collection, its value remains poor as compared with other datasets. Two main reasons may account for this performance. Firstly, the L-curve approach must be calculated for a large number of eigenvalues to find an appropriate level of regularization, and this dataset holds the lowest number of samples per class (just a few dozen). Secondly, some data points clearly yield different clusters having low similarity within the same class, making unreliable the estimation of kernel bandwidth that assumes just one homogeneous cluster per class. Hence, the median of data distances increases the kernel bandwidth parameter in Equation (4-4a) and reduces the eigenvalues obtained for the decomposition in Equation (3-4).
The above-mentioned aspects indicate that the proposed criterion sets a component trade-off as an optimal boundary of architecture selection, in which the network has a stable behavior and improve the classification accuracy. Besides, the hidden layer size $\widetilde{m}$ attained by JSR, prevents the hidden units from becoming unstable due to problems of under-over fitting.

5.2. Results of GSN pre-training

The purpose of this section is to investigate the influence on GSN classification accuracy performed by the proposed CKA-based pre-training that is compared against other baseline pre-training methods: Random Weights (Rand), PCA, and AutoEncoders (AE) [Vincent et al., 2010]. During the performance evaluation, the above-examined selection strategies are also reflected that determine the size of projection matrices to be computed: $\boldsymbol{W^1}\in\mathbb{R}^{P\times\widetilde{m}}$, $\boldsymbol{W^2}\in\mathbb{R}^{\widetilde{m}\times\widetilde{m}}$, and $\boldsymbol{W^3}\in\mathbb{R}^{\widetilde{m}\times C}$, where $\widetilde{m}$ is chosen from Table **5-1**.

Figure **5-3** visually illustrates the influence produced by each pre-training approach on the data projection, that is, the weights estimated for the first hidden layer. Thus, the first 40 neurons, ranked in descending order of their norm values, are compared for AutoEncoders (top), PCA (middle), and CKA (bottom) for each image dataset. Evidently, while resulting AE weights are highly scattered, PCA and CKA approaches identify dynamic structures of the input distribution. Specifically, the computed PCA-based units tend to hold the main variations so that the first PCA eigenvectors encode the turning table movement of `COIL-20` database (see Figure 5.3k) and the principal face shapes (also known as eigenfaces) of `Olivetti` collection. By contrast, the CKA-based units present more discriminating features. As shown in Figure 5.3l for `Olivetti` dataset, each neuron emphasizes the areas that discriminate the most every face like eyes and mouth.

Further, we evaluate the pre-training influence on the learning speed by calculating the number of training epochs that must be carried out until the algorithm converges to the maximum accuracy value. As seen in Figure **5-4** that displays the network performance reached within the range of 500 successive epochs, CKA performs the highest accuracy for

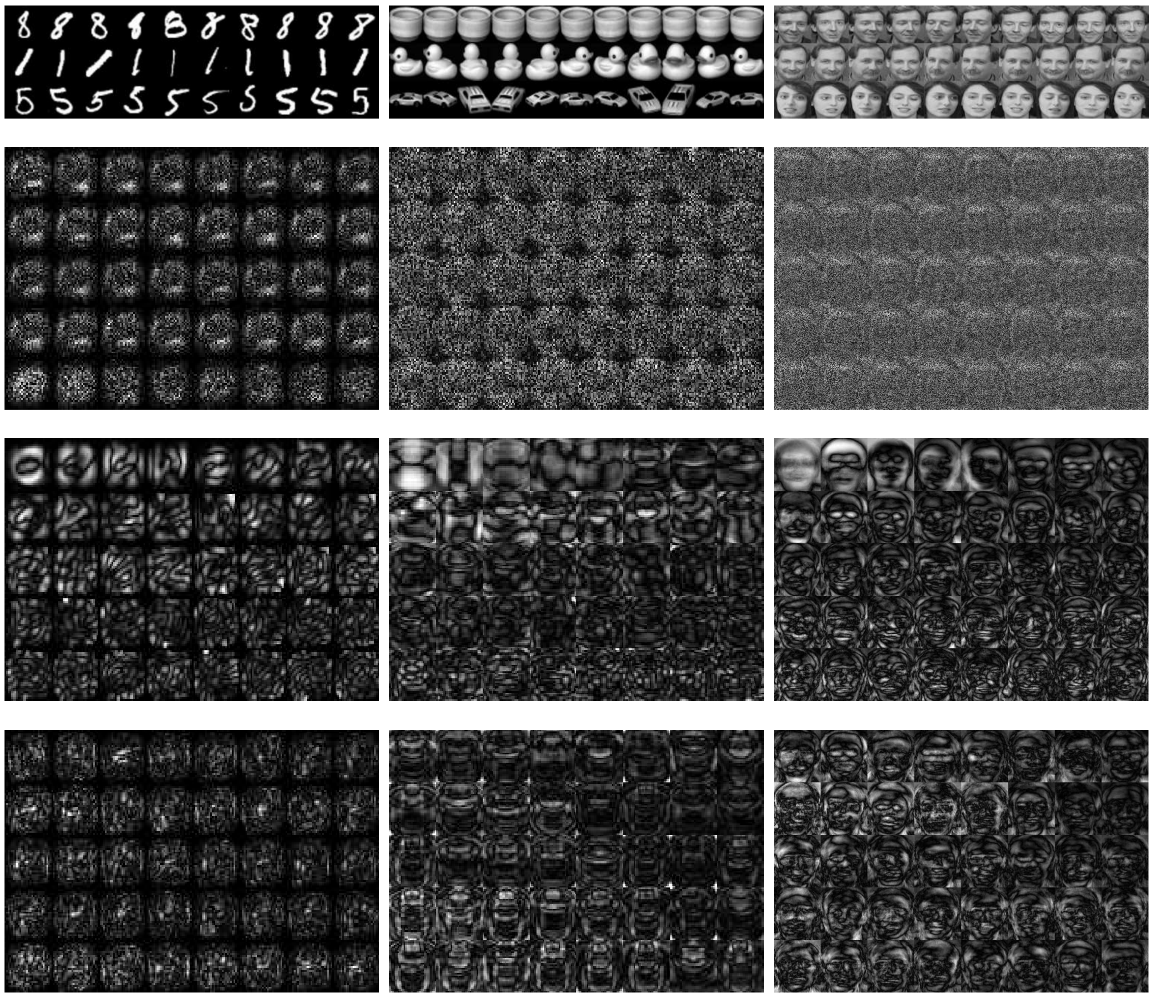
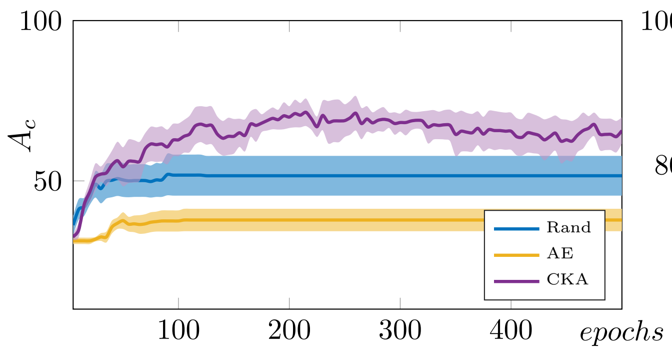


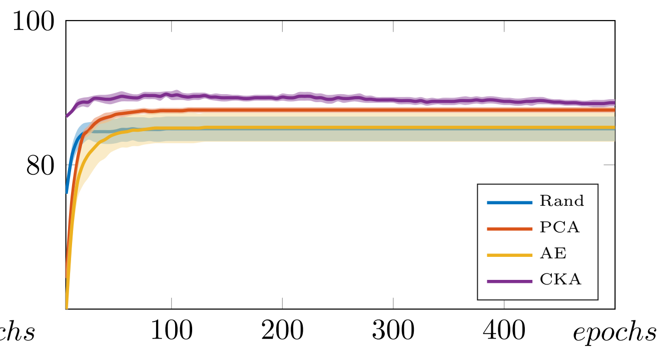
Figure 5-3.: Visualization of the pre-training weights at the first layer for considered databases. In each plot, all weights of the 40 largest hidden units are plotted for AutoEncoders (top), PCA (middle), and CKA (bottom) approaches. The gray values represent weight magnitudes.

all databases except in the case of `Wdbc`. Thus, our proposal increases the performance in ∼19.6% points for `Glass`. Note that the accuracy deviation achieved by CKA is much lower almost in all cases. At the same time, the CKA learning speed is surely the fastest besides in `Wine` and `Wdbc` data, for which all considered pre-training approaches converge quite rapidly.

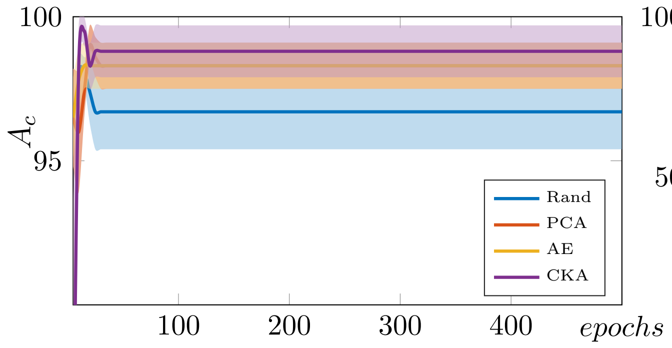
Table 5-2 presents the GSN classification accuracy reached by each one of the evaluated pre-training methods (i.e., PCA, AE, and CKA) where the best performance estimated is denoted in bold for each database. For easier interpretation, the studied architecture selection strategies are also ranked in ascending order of the estimated hidden layer size. Thus, in the case of `Wine`, the architecture selection in Li [Li et al., 1995] provides the



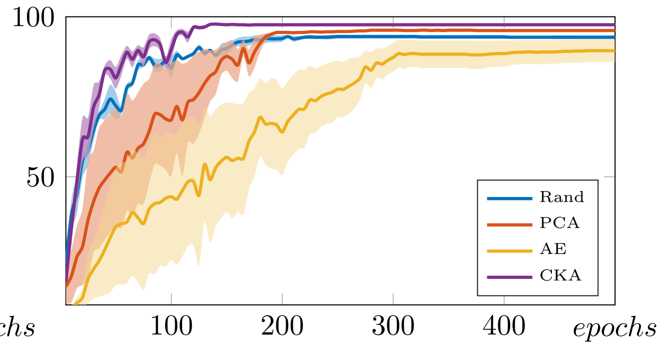
a `Glass`



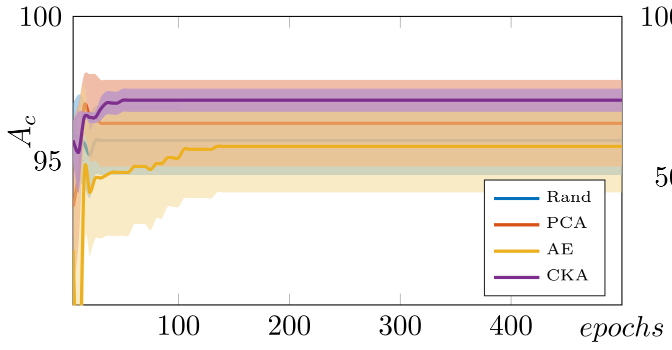
b `MNIST`



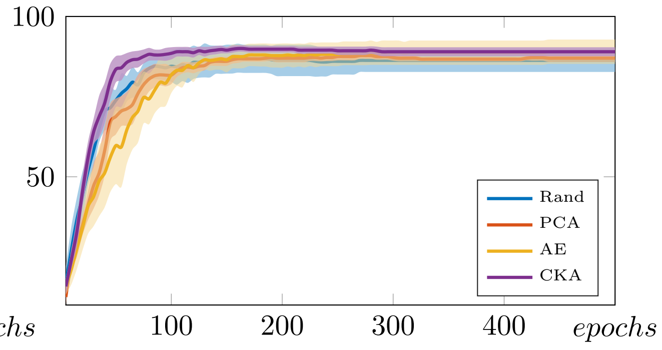
c `Wine`



d `COIL-20`



e `Wdbc`



f `Olivetti`

Figure 5-4.: GSN average accuracy and standard deviation vs iterations for the four pre-training approaches: Random, AutoEncoder, PCA-based projection, and CKA-based projection. Glass (top), Wine (middle), and Wdbc (bottom) datasets.

lowest network complexity while approaches in JSR and Shibata [Shibata and Ikeda, 2009] perform the best accuracy. Nonetheless, the linear dependency estimation is not carried out as architecture selection in the case of `Glass` dataset due to the computed hidden layer size

	Method	$\widetilde{m}$	*Pre-training Approach*		
			PCA	AE	CKA
Wine	[Li et al., 1995]	4	97.6±1.3	97.6±1.6	99.0±0.7
	[Shibata and Ikeda, 2009]	6	**99.0±0.7**	**98.3±0.8**	**99.5±0.5**
	[Ke and Liu, 2008]	13	94.3±1.3	96.9±1.6	98.3±0.8
	[Chen, 2015]	64	–	97.4±0.9	98.3±0.8
Glass	[Shibata and Ikeda, 2009]	7	35.1±1.8	35.8±3.4	41.3±4.9
	JSR/[Ke and Liu, 2008]	11	–	**37.8±3.5**	**71.4±4.3**
	[Chen, 2015]	64	–	40.0±1.9	65.0±6.7
Wdbc	[Shibata and Ikeda, 2009][Li et al., 1995]	7	95.5±1.5	94.5±1.6	96.5±0.2
	JSR	11	**96.9±1.1**	**95.5±1.6**	**97.1±0.4**
	[Ke and Liu, 2008]	26	96.7±0.6	96.1±1.0	97.7±0.8
	[Chen, 2015]	64	–	96.3±1.8	97.2±1.2
MNIST	[Li et al., 1995]	39	86.1±0.4	83.9±2.2	88.1±0.4
	JSR	41	**87.6±0.4**	**85.2±2.0**	**89.9±0.4**
	[Shibata and Ikeda, 2009]	88	86.3±0.5	87.1±1.2	90.0±1.0
	[Chen, 2015]	100	85.8±0.5	87.6±0.6	89.9±0.6
	[Ke and Liu, 2008]	413	87.0±0.7	88.8±1.6	91.2±0.7
COIL-20	[Li et al., 1995]	44	89.0±2.3	88.9±2.3	95.8±2.8
	JSR	50	**95.7±0.2**	**89.4±3.5**	**97.7±0.3**
	[Chen, 2015]	100	95.1±0.3	87.3±4.1	97.4±0.7
	[Shibata and Ikeda, 2009]	143	91.5±0.7	83.6±6.2	97.6±0.5
	[Ke and Liu, 2008]	527	96.7±0.3	62.2±17.0	97.2±0.7
Olivetti	JSR	45	**87.8±1.0**	**89.0±3.8**	**90.0±1.0**
	[Li et al., 1995]	143	87.6±3.9	81.0±2.7	88.0±5.3
	[Shibata and Ikeda, 2009]	641	–	92.5±2.9	89.6±2.1
Average			84.2±1.02	83.0±2.90	90.1±1.57

Table **5-2**.: GSN classification accuracy for layer size selection criteria and pre-training approaches. The marked blank fields (–) corresponds to infeasible calculations where the layer dimension can not enable a pre-training matrix.

is smaller than the number of classes. Such cases are not contemplated for classification tasks based on neural networks.

As seen in the bottom row of Table **5-2** that shows the mean value of performed accuracy for all testing data, the AE approach performs the worst (83.0±2.9) though it can be employed for all testing data. Then, PCA improves a little (84.2±1.02) may be limited by the lack of label information in the matrix computations. However, PCA has the largest amount of cases with unfeasible calculations (marked with blank fields –), meaning that the computed hidden dimensions are bigger than the inputs so that PCA can not enable a pre-training matrix. Lastly, CKA clearly performs the best (90.1±1.57), enabling training in all cases too. Moreover, CKA presents the most significant classification improvement on `Glass`, increasing twice the accuracy as compared with another pre-training methods.

Noteworthy that all pre-training approaches, regardless of the tested database, reach their best accuracy concurrently for the same architecture selection strategy, that is, the proposed JSR approach. Therefore, the proposed CKA approach allows producing the best accuracy for almost the smallest network complexity. Furthermore, some achieved layer sizes larger than JSR leads to a small accuracy improvement (as in `Wdbc` and `MNIST`) or even decreases the performance because of the emerged overfitting (see `Wine`, `Glass`, `COIL-20`, and `Olivetti`).

Discussion

The second stage of GSN topology configuration regards the layer-wise pre-training, for which we propose to carry out using the Centered Kernel Alignment (CKA) in order to enhance further the system classification accuracy as well as its learning speed. In this sense, the set of projection matrices is computed to maximize the alignment between both, the label and projected data, centered kernels. The projection matrices obtained shown that pre-trained networks learn qualitatively different features (see Figure 5-3). However, supervised CKA-based pre-training focuses attention on identifying localized features in a certain region of weight space. For this reason, CKA-based projection detects more interesting structures present in data highlighting the representation of discriminant features.
Later, we study the effect generated by the pre-training stage over optimization problem and testing accuracy. Based on the results described in Figure 5-4, we conclude that the starting point in the non-convex optimization problem is indeed quite important. In this sense, the CKA-based matrices capture most of the complex dependencies between parameters, increase the convergence speed in the learning stage, and enhance the class discrimination. Resulting training curves prove that our proposal converges faster than the baseline approaches for a given GSN architecture.
In addition, CKA improves classification accuracy of all the studied architecture selection methods, attaining the best result at the JSR-based layer size. In the particular case of `Glass` data, the combination of CKA and JSR significantly increases the achieved classification accuracy (as much as twice) due to the following reasons: Firstly, the estimated layer-sizes are larger than the number of classes and features, increasing the separability among data points. Secondly, learning of the projection matrices within discriminative scheme forces each coordinate of the hidden space to represent a different cluster of the dataset.
On the other hand, though the introduction of a pre-training stage speeds the walkback training procedure, there is a need for an additional computing of the initial projection matrices. In this sense, the iterative optimization adopted used by CKA makes the GSN pre-training more expensive than the known cost for PCA. The observations in this experiment confirm that starting the supervised optimization from pre-trained weights that use the data

distribution and prior knowledge regarding the studied process, e.g., supervised information, achieves good performance in classification and object recognition tasks , provides a better optimization, increases convergence speed and improve the network stability rather than from randomly and unsupervised pre-training methods.

6. Summary

In this work, we propose an automated topology estimation of Generative Stochastic Networks (GSN) for classification tasks through the introduction of kernel theory in order to improve the network performance and to increase the learning speed.

First, we analyze the influence of the network architecture selection procedure in the system performance. To this, we introduce a tuning criterion named Joint-Spectrum Regularization (JSR) to estimate automatically the GSN layer size built on the regularization of the eigenvalues computed from the joint spectral decomposition of data. JSR creates a tensor kernel to join the information of input and output samples into a single space and obtains the least number of hidden units as a trade-off component between network performance and architecture complexity. For the sake of comparison, our proposal is validated against analytical architecture setup approaches in six widely use data sets of the `UCI` repository databases and image-based collections. The evaluation and the learning process is the same for all considered datasets using Walkback training and SGD for the optimization problem. Obtained results show that the estimated $\widetilde{m}$ provides a proper number of hidden neurons in comparison with the baselines methods. Therefore, proposed approach is a suitable alternative to support automatic parameter selection related to fixing the suitable number of hidden neurons in a DNN, achieving enhance the classification accuracy while avoiding both the over-fitting produced by a poor parameter selection and the costly heuristic exhaustive searches.

On the other hand, since the GSN training is carried out using iterative algorithms, the network accuracy and convergence speed, depend on the initial set of parameters due to the non-convexity of the cost function. In this sense, we introduce supervised step-wise pre-training stage, to learn projection matrices using the data distribution and prior information making use of the Center Kernel Alignment (CKA). To this end, we compute a weight matrices set $\{\boldsymbol{W}^l\}$ maximizing the alignment between both centered kernels, label and projected data. Thus, the information encoded by each network layer is maximized. The validation of the proposed approach is carried out for supervised learning tasks using three pre-training strategies (AE, PCA, and CKA). From the obtained results we noted that supervised CKA-based pre-training stage is an important procedure due to its ability to capture main variations from the input data. Besides, CKA influences the system behavior by tuning the starting point in the non-convex optimization problem and capturing the dependencies between the parameters of the model. Thus, the proposed approach achieves more discriminative representation spaces which improve the learning speed and the network

performance in terms of accuracy.
Finally, we combine the above-mentioned kernel-based methodologies to enhance the GSN setup and examine the effect on system performance. According to hidden layer size, networks pre-trained tend to improve their performance in terms of classification although in some cases the high complexity of architecture leads to network over-fitting reducing its accuracy. So, set layer size using JSR together with CKA-based pre-training method proves to be the best network topology. This configuration allows learning hidden representation more discriminant due to highlights different localized features focus interest in areas that contains the major data information, i.e, major variation. Therefore, the enhanced GSN outperforms state-of-the-art strategies in terms of system accuracy and increases the convergence speed of the fine tuning stage.

Part III.

Final remarks

7. Conclusions and future work

7.1. Conclusions

This work highlighted an automated topology estimation of Generative Stochastic Networks for classification tasks through the introduction of kernel theory in the architecture selection and pre-training stages. In this sense, two kernel strategies were proposed to learn automatically relevant data relations that are then used to set the parameters in GSNs. The introduced approaches naturally lead to data-dependent processing tuned to the particular samples restrictions and to the considered learning scenario, focusing on supervised tasks. Besides, the introduced kernel-based enhancement framework is tested in some classification and object recognition tasks related to data processing and image analysis applications. Overall, attained results demonstrated that proposed approaches allow summarizing and capturing the main input patterns, favoring the learning performance, increasing the convergence speed in comparison to state-of-the-art methods. Following, the main concluding remarks regarding each provided representation strategy are described:

- An automatic architecture selection criterion based on kernel functions that allows to quantify the optimal layer size holding the most mutual information shared by the input and output spaces was presented. The considered joint spectral decomposition of the data tensor kernel allows gathering both, input and output samples, into a single space, and the L-curve regularization reduces the number of hidden units required for discriminating the classes. The proposed Joint Spectrum Regularization criterion is tested as hidden layer size selection strategy to support classification and object recognition tasks. Attained results at this stage prove that our proposal provides the best trade-off between network complexity and system performance, in comparison with analytical architecture setup approaches, with the additional benefit of avoiding the costly heuristic exhaustive searches.

- Aiming to cope with the non-convex cost function, we capture the complex label dependencies at each layer using supervised step-wise pre-training stage that maximizes the centered kernel alignment between sample and label kernels. Our approach, learns a discriminative projection matrix based on a CKA-based function that encode the discriminative information to get a suitable hidden representation. Thereby, CKA-based pre-training builds a set of projected features that match the best all provided target classes from each hidden state. The introduced pre-training method was tested on classification and object recognition based on image analysis. As a result, linear projection matrices better matching samples with their corresponding class improve the learning speed of the fine tuning and reduce the classification errors for high-dimensional pattern recognition problems.

- An enhanced General Stochastic Network was developed to support supervised learning tasks. Regarding this, the proposed framework considers the two kernel-based strategies above mentioned (JSR and CKA-based pre-training approach). Our proposal highlights the properties of joint information by the samples and labels to find a suitable layer dimension preserving the maximal information. Then, we use the resulting layer size to learn the projection matrix computed by the CKA-based function that initializes the weight matrices in a GSN. As a result, an enhance network able to capture a lot of information from input data distribution and encode discriminant patterns is built. According to our experiments, the proposed supervised kernel approach for automated learning outperforms, in most of the cases, state-of-the-art strategies in terms of system accuracy and learning speed.

7.2. Future work

We have illustrated a kernel-based representation framework aiming to enhance General Stochastic Networks setup in supervised machine learning tasks. However, from the attained theoretical and experimental results, there are still many issues that can be led to improve the network performance in terms of system accuracy. In particular, the following considerations could be of interest for future work approaches:

- We plan to improve the proposed layer size estimation using more complex kernel functions, like multiple kernel ensemble and multi-dimensional Gaussian, to build the spectral representation, aiming to accurately capture all point similarities within the same class. Furthermore, JSR can be tested especially as a relevant data representation approach to support dimensionality reduction. Besides, the proposed criterion can be used as a tool of feature ranking to support feature selection algorithms.

- Two important remarks for further research is related to evaluated the quality of CKA-based pre-training method. Firstly, we will extend the pre-training scheme to various supervised tasks (like regression and forecasting) by adapting the target kernel function to the output relations. Secondly, the proposed supervised pre-training methodology can be used to learn initial projection matrices in others DNN as RBMs and SAE.

- An interesting line of future work involves the evaluation of the introduced GSN by employing a loss function based on information measures, e.g., cross-entropy. Besides, we will introduce kernel-based cost functions, as the centered kernel alignment, into the fine tuning stage for improving the network performance. Furthermore, it would be interesting to investigate the possibility of employing others non-convex multi-objective optimization strategies.

8. Appendix

8.1. Gradient descend-based optimization of CKA approach

The explicit objective function of the empirical CKA in Equation (3-10) yields [Brockmeier et al., 2014]:

$$\rho\left(\boldsymbol{K}^l, \boldsymbol{K}^y\right) = \log\left(\operatorname{tr}\left(\boldsymbol{K}^l(\boldsymbol{W}^l)\boldsymbol{H}\boldsymbol{K}^y\boldsymbol{H}\right)\right) - \tfrac{1}{2}\log\left(\operatorname{tr}\left(\boldsymbol{K}^l(\boldsymbol{W}^l)\boldsymbol{H}\boldsymbol{K}^l(\boldsymbol{W}^l)\boldsymbol{H}\right)\right) + \rho_0, \quad (8\text{-}1)$$

where $\rho_0 \in \mathbb{R}$ is a constant that we assume independent on $\boldsymbol{W}^l$. Consequently, a gradient descent algorithm iteratively solves the optimization at hand, for the gradient of the objective function in Equation (8-1) results in the form:

$$\nabla_{\boldsymbol{W}^l}\left(\rho\left(\boldsymbol{K}^l, \boldsymbol{K}^y\right)\right) = -4\left(\boldsymbol{W}^l\right)^\top\left(\left(\boldsymbol{G}\circ\boldsymbol{K}^l(\boldsymbol{W}^l)\right) - \operatorname{diag}\left(\boldsymbol{1}^\top\left(\boldsymbol{G}\circ\boldsymbol{K}^l(\boldsymbol{W}^l)\right)\right)\right)\boldsymbol{W}^l, \quad (8\text{-}2)$$

where notations diag$(\cdot)$ and $\circ$ denote the diagonal operator and the Hadamard product, respectively. $\boldsymbol{G} \in \mathbb{R}^{N\times N}$ is the gradient of the objective function with respect to $\boldsymbol{K}^l$, calculated as follows:

$$\boldsymbol{G} = \nabla_{\boldsymbol{K}^l}\left(\rho\left(\boldsymbol{K}^l, \boldsymbol{K}^y\right)\right) = \frac{\boldsymbol{H}\boldsymbol{K}^y\boldsymbol{H}}{\operatorname{tr}\left(\boldsymbol{K}^l\boldsymbol{H}\boldsymbol{K}^y\boldsymbol{H}\right)} - \frac{\boldsymbol{H}\boldsymbol{K}^l\boldsymbol{H}}{\operatorname{tr}\left(\boldsymbol{K}^l\boldsymbol{H}\boldsymbol{K}^l\boldsymbol{H}\right)}. \quad (8\text{-}3)$$

Then, the updating rule for estimating $\boldsymbol{W}^l$, provided the initial guess $\boldsymbol{W}^l_o$, becomes:

$$\boldsymbol{W}^l_{t+1} = \boldsymbol{W}^l_t - \mu_t \nabla_{\boldsymbol{W}^l_t}\left(\rho\left(\boldsymbol{K}^l, \boldsymbol{K}^y\right)\right), \quad (8\text{-}4)$$

where $\mu_t \in \mathbb{R}^+$ is the step size of the learning rule at optimization epoch t.

Bibliography

[Alain and Bengio, 2014] Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593. (Cited on page 10)

[Alain et al., 2015] Alain, G., Bengio, Y., Yao, L., Yosinski, J., Thibodeau-Laufer, E., Zhang, S., and Vincent, P. (2015). Gsns: Generative stochastic networks. *arXiv preprint arXiv:1503.05571*. (Cited on pages 4 and 26)

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127. (Cited on page 10)

[Bengio, 2012] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer. (Cited on pages 6 and 29)

[Bengio et al., 2013a] Bengio, Y., Courville, A., and Vincent, P. (2013a). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828. (Cited on page 10)

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153. (Cited on pages 3, 6, and 7)

[Bengio et al., 2013b] Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2013b). Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*. (Cited on page 10)

[Bengio et al., 2013c] Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013c). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907. (Cited on pages 3, 4, and 10)

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer. (Cited on page 34)

[Brockmeier et al., 2014] Brockmeier, A., Choi, J., Kriminger, E., Francis, J., and Principe, J. (2014). Neural decoding with kernel-based metric learning. *Neural Computation*, 26:–. (Cited on pages 30 and 51)

[Cardenas-Pena et al., 2014] Cardenas-Pena, D., Orbes-Arteaga, M., Castro-Ospina, A., Alvarez-Meza, A., and Castellanos-Dominguez, G. (2014). A kernel-based representation to support 3d mri unsupervised clustering. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 3203–3208. IEEE. (Cited on page 2)

[Chen, 2015] Chen, G. (2015). Deep transductive semi-supervised maximum margin clustering. *arXiv preprint arXiv:1501.06237*. (Cited on pages 37, 38, and 42)

[Chen and Lu, 2013] Chen, L.-Y. and Lu, C.-J. (2013). An improved independent component analysis algorithm based on artificial immune system. *International Journal of Machine Learning and Computing*, 3(1):93–97. (Cited on page 7)

[Collazos-Huertas et al., 2015] Collazos-Huertas, D., Álvarez-Meza, A., Gaviria-Gómez, N., and Castellanos-Dominguez, G. (2015). Kernel-based feature relevance analysis for ecg beat classification. In *Pattern Recognition and Image Analysis*, pages 291–299. Springer. (Cited on page 2)

[da Silva et al., 2016] da Silva, A. J., Ludermir, T. B., and de Oliveira, W. R. (2016). Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks*, 76:55–64. (Cited on page 5)

[Dahl et al., 2013] Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE. (Cited on page 5)

[Doukim et al., 2010] Doukim, C. A., Dargham, J. A., and Chekima, A. (2010). Finding the number of hidden neurons for an mlp neural network using coarse to fine search technique. In *Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on*, pages 606–609. IEEE. (Cited on page 5)

[Erhan et al., 2010] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660. (Cited on pages 6 and 28)

[Erhan et al., 2009] Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on artificial intelligence and statistics*, pages 153–160. (Cited on page 6)

[Evett and Spiehler, 1987] Evett, I. W. and Spiehler, E. (1987). Rule induction in forensic science. *KBS in Goverment, Online Publications*, pages 107–118. (Cited on page 32)

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256. (Cited on page 6)

[Hansen et al., 2007] Hansen, P. C., Jensen, T. K., and Rodriguez, G. (2007). An adaptive pruning algorithm for the discrete l-curve criterion. *Journal of computational and applied mathematics*, 198(2):483–492. (Cited on page 28)

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554. (Cited on pages 3, 4, 6, and 19)

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. (Cited on page 5)

[Hunter et al., 2012] Hunter, D., Yu, H., Pukish III, M. S., Kolbusz, J., and Wilamowski, B. M. (2012). Selection of proper neural network sizes and architectures—a comparative study. *Industrial Informatics, IEEE Transactions on*, 8(2):228–240. (Cited on page 5)

[Jiang et al., 2008] Jiang, N., Zhang, Z., Ma, X., and Wang, J. (2008). The lower bound on the number of hidden neurons in multi-valued multi-threshold neural networks. In *Intelligent Information Technology Application, 2008. IITA'08. Second International Symposium on*, volume 1, pages 103–107. IEEE. (Cited on page 6)

[Ke and Liu, 2008] Ke, J. and Liu, X. (2008). Empirical analysis of optimal hidden neurons in neural network modeling for stock prediction. In *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, volume 2, pages 828–832. IEEE. (Cited on pages 5, 6, 36, 38, and 42)

[Kreyszig, 1989] Kreyszig, E. (1989). *Introductory functional analysis with applications*, volume 81. wiley New York. (Cited on pages 10 and 11)

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. (Cited on page 3)

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. (Cited on page 2)

[LeCun et al., 1995] LeCun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U., Sackinger, E., Simard, P., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276. (Cited on page 33)

[Li et al., 1995] Li, J.-Y., Chow, T. W., and Yu, Y.-L. (1995). The estimation theory and optimization algorithm for the number of hidden units in the higher-order feedforward neural network. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 3, pages 1229–1233. IEEE. (Cited on pages 36, 38, 40, and 42)

[Lichman, 2013] Lichman, M. (2013). UCI machine learning repository. (Cited on page 32)

[Martens, 2010] Martens, J. (2010). Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742. (Cited on page 6)

[Mohamed et al., 2011] Mohamed, A.-r., Sainath, T. N., Dahl, G., Ramabhadran, B., Hinton, G. E., Picheny, M., et al. (2011). Deep belief networks using discriminative features for phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5060–5063. IEEE. (Cited on pages 7 and 29)

[Mohri et al., 2012] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press. (Cited on page 2)

[Molina-Giraldo et al., 2015] Molina-Giraldo, S., Carvajal-González, J., Álvarez-Meza, A., and Castellanos-Domínguez, G. (2015). Video segmentation framework based on multi-kernel representations and feature relevance analysis for object classification. In *Pattern Recognition Applications and Methods*, pages 273–283. Springer. (Cited on page 2)

[Nene et al., 1996] Nene, S. A., Nayar, S. K., Murase, H., et al. (1996). Columbia object image library (coil-20). Technical report. (Cited on page 33)

[Ngiam et al., 2011] Ngiam, J., Chen, Z., Bhaskar, S. A., Koh, P. W., and Ng, A. Y. (2011). Sparse filtering. In *Advances in Neural Information Processing Systems*, pages 1125–1133. (Cited on page 3)

[Orbes-Arteaga et al., 2015] Orbes-Arteaga, M., Cárdenas-Peña, D., Álvarez, M. A., Orozco, A. A., and Castellanos-Dominguez, G. (2015). Kernel centered alignment supervised metric for multi-atlas segmentation. In *Image Analysis and Processing—ICIAP 2015*, pages 658–667. Springer International Publishing. (Cited on page 2)

[Panchal et al., 2011] Panchal, G., Ganatra, A., Kosta, Y., and Panchal, D. (2011). Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, 3(2):332–337. (Cited on page 4)

[Parzen, 1959] Parzen, E. (1959). *Statistical inference on time series by Hilbert space methods*. Stanford Univ. (Cited on pages 10 and 13)

[Rani B et al., 2012] Rani B, S. et al. (2012). Role of hidden neurons in an elman recurrent neural network in classification of cavitation signals. *International Journal of Computer Applications*, 37(7):9–13. (Cited on page 5)

[Ranzato et al., 2007] Ranzato, M. A., Poultney, C., Chopra, S., and LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In *Proceedings of NIPS*. (Cited on page 29)

[Samaria and Harter, 1994] Samaria, F. S. and Harter, A. C. (1994). Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE. (Cited on page 33)

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117. (Cited on page 3)

[Scholkopf and Smola, 2001] Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press. (Cited on pages 10 and 14)

[Schulz et al., 2015] Schulz, H., Cho, K., Raiko, T., and Behnke, S. (2015). Two-layer contractive encodings for learning stable nonlinear features. *Neural Networks*, 64:4–11. (Cited on page 7)

[Seide et al., 2011] Seide, F., Li, G., and Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pages 437–440. (Cited on page 3)

[Sheela and Deepa, 2013] Sheela, K. G. and Deepa, S. (2013). Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, 2013. (Cited on page 5)

[Shibata and Ikeda, 2009] Shibata, K. and Ikeda, Y. (2009). Effect of number of hidden neurons on learning in large-scale layered neural networks. In *ICCAS-SICE, 2009*, pages 5008–5013. IEEE. (Cited on pages 6, 36, 38, 41, and 42)

[Stathakis, 2009] Stathakis, D. (2009). How many hidden layers and nodes? *International Journal of Remote Sensing*, 30(8):2133–2147. (Cited on page 6)

[Street et al., 1993] Street, W. N., Wolberg, W. H., and Mangasarian, O. L. (1993). Nuclear feature extraction for breast tumor diagnosis. In Acharya, R. S. and Goldgof, D. B., editors, *Biomedical Image Processing and Biomedical Visualization*. SPIE-Intl Soc Optical Eng. (Cited on page 33)

[Sun, 2012] Sun, J. (2012). Learning algorithm and hidden node selection scheme for local coupled feedforward neural network classifier. *Neurocomputing*, 79:158–163. (Cited on pages 5 and 27)

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147. (Cited on page 6)

[Trenn, 2008] Trenn, S. (2008). Multilayer perceptrons: approximation order and necessary number of hidden units. *Neural Networks, IEEE Transactions on*, 19(5):836–844. (Cited on page 5)

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM. (Cited on pages 3, 6, and 10)

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408. (Cited on pages 29 and 39)

[Wang et al., 2015] Wang, Z., Chang, S., Zhou, J., and Huang, T. S. (2015). Learning a task-specific deep architecture for clustering. *arXiv preprint arXiv:1509.00151*. (Cited on pages 37 and 38)

[Weston et al., 2012] Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer. (Cited on pages 7 and 29)

[Yuan et al., 2003] Yuan, H., Xiong, F., and Huai, X. (2003). A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy. *Computers and Electronics in Agriculture*, 40(1):57–64. (Cited on page 5)

[Zöhrer and Pernkopf, 2014] Zöhrer, M. and Pernkopf, F. (2014). General stochastic networks for classification. In *Advances in Neural Information Processing Systems*, pages 2015–2023. (Cited on pages 4, 10, 25, and 34)