Selección de perceptrones multicapa usando aprendizaje bayesiano

Multilayer perceptron selection using bayesian learning

Fernando Ceballos[1], Luís Eduardo Muñoz[2], Julián Moreno Cadavid[3]
Universidad de Antioquia, Medellín, Colombia
jmoreno@unal.edu.co
lemunozg@utp.edu.co
fceball@udea.edu.co

***Resumen*— La Regularización Bayesiana de perceptrones multicapa pretende resolver el problema de optimización de los pesos de la red neuronal simultáneamente con el problema de generalización. En este trabajo se realiza un análisis de la regularización Bayesiana, que parece ser una de las más poderosas técnicas de entrenamiento de perceptrones multicapa, para luego hacer un comparativo con los resultados obtenidos usando Regla Delta Generalizada. Finalmente se discute alguna implicación de los resultados obtenidos respecto a la técnica basada en algoritmos constructivos para la selección final de neuronas en la capa oculta.**

***Palabras clave*— multicapa, perceptrones, regularización bayesiana, regla delta generalizada, redes neuronales.**

***Abstract*— The Bayesian regularization of perceptrones multilayer seeks to solve the problem of optimization of the weights in a neural net simultaneously with the generalization problem. In this work is carried out an analysis of the Bayesian regularization, that it seems to be one of those but powerful techniques of multilayer perceptron training, stops then to make a comparative one with the obtained results using Generalized Delta's rule. Finally you some implication of the obtained results against constructive algorithms for the selection of neurons in the hidden layer.**

***Key Word* — bayesian regularization, nonlinear regression, neural networks. multilayer, perceptrons.**

I. INTRODUCCIÓN

La aproximación de la forma estructural de una función, a partir de un conjunto de datos, ha sido un problema que ha estado presente en diversas disciplinas. Esta problemática ha sido ampliamente estudiada y discutida en el caso lineal y se han empleado diferentes metodologías para su aproximación [1-3]. Estos métodos se quedan cortos en los casos complejos. Es así, como los métodos de aproximación no paramétrica han sido comúnmente usados en la construcción de modelos de regresión (lineal y no lineal) [4], [5]; y dentro de ellos, los modelos de redes neuronales artificiales, entendidos como modelos no paramétricos de regresión no lineal [6], [7], los cuales han ganado terreno en su aplicación a este tipo de problemas, y han sido aplicados exitosamente en áreas como la ingeniería, economía y afines [8-10]. Los problemas solucionados pueden ser catalogados de forma general en las categorías de regresión, outliers, supresión de ruido y pronóstico [11], [12].

Esta problemática se puede resolver aplicando un modelo de red neuronal del tipo perceptrón multicapa que requiere realizar el proceso de selección del modelo, el cual abarca tres dificultades fundamentales: la selección de las entradas relevantes, la determinación del número de capas ocultas y la determinación del número de neuronas en cada capa oculta, y finalmente, la estimación de los pesos de las conexiones. Sin embargo, aspectos como la preparación de los datos o la técnica de entrenamiento usada juegan un papel fundamental en la calidad del modelo final obtenido haciendo del problema de la construcción del modelo una tarea que no es trivial[13]. Aunque el teorema de Kolmogorov garantiza que es posible aproximar con una precisión arbitraria la variable dependiente en el conjunto de datos de entrenamiento, usando un perceptrón multicapa con una sola capa oculta, con solo aumentar el número de neuronas en la capa oculta a un número suficientemente grande, no indica cómo obtener los pesos de la red neuronal, ni da cuenta del fenómeno de sobre ajuste[14]. Por otra parte, el problema de la determinación de los pesos óptimos de las conexiones de la red neuronal está asociado directamente la dificultad de obtener el óptimo global del error, debido a la existencia de múltiples puntos de mínima local [15], que hacen que el proceso de entrenamiento sea difícil. Con la

[1] Ingeniero de sistemas, PhD (c).

Fecha de Recepción: 26 de Agosto de 2011
Fecha de Aceptación: 12 de Noviembre de 2011

[2] Ingeniero de sistemas, MsC.

[3] Ingeniero de sistemas, PhD (c).

regularización bayesiana se pretenden resolver los problemas de generalización y optimización de los pesos, ya que esta presenta una convergencia a un conjunto de pesos óptimos, los cuales toman los datos iniciales como una distribución de probabilidad y de éstos extrae información necesaria para generar uno conjunto de parámetros denominados hiperparámetros que representen la distribución de los datos de entrada.

En consecuencia, el objetivo de este trabajo es presentar un análisis comparativo entre los métodos de regularización bayesiana y regla delta, mostrando las ventajas en cuanto a calidad de estimación de una metodología sobre la otra, utilizando los resultados del entrenamiento de redes neuronales artificiales, con una arquitectura similar, para el mismo conjunto de datos, además de un aporte respecto a la sensibilidad de estos modelos al punto de arranque aleatorio asociado a los pesos de la red. Para alcanzar este objetivo, el resto de este artículo esta divido en las siguientes partes: en la sección 2 se presenta de una manera formal la estructura de una red neuronal, además de problemas más comúnmente encontrados en su construcción. En las secciones 3 y 4 se presenta la estructura formal del entrenamiento basado en regla delta y regulación bayesiana. En la sección 5 se describen los problemas analizados, se presentan los resultados prácticos obtenidos y, por último, se presentan las conclusiones de este trabajo.

II. PROBLEMATICA ALREDEDOR DE PERCEPTRONES MULTICAPA

A. Representación matemática de un perceptrón multicapa.

Las redes neuronales aproximan cualquier función arbitrariamente, de una manera directa respecto a su número de neuronas. Los perceptrones multicapa poseen una alta simplicidad de elaboración y alto nivel de aproximación. Las redes neuronales de este tipo pueden ser representadas de la siguiente manera:

$$\vec{y} = \vec{f}_t(\vec{W}, \vec{X}) = \vec{W}_{ho}^T s(\vec{W}_{ih}\vec{X} + \vec{W}_{bh}) + \vec{W}_{bo} \quad (1)$$

X es la entrada y *y* es la salida del perceptrón multicapa, *Wih* es la matriz de pesos de la capa de entrada a la capa oculta, *Wbh* es el vector de pesos de la neurona adaptativa a las neuronas de la capa oculta, *Who* es la matriz de pesos de las neuronas de la capa de oculta a la capa de salida y por último, *Wbo* es el vector de pesos de la neurona adaptativa a las neuronas de la capa de salida. La función *s* utilizada como función de activación está dada por:

$$s(\mu) = \frac{1}{1 + \exp(-\mu)} \quad (2)$$

B. Selección de la arquitectura del modelo.

La selección de neuronas en la capa oculta y la estimación de los pesos óptimos para el aprendizaje de la red, son los problemas que inicialmente se presentan al modelador en redes neuronales y este proceso debe tener unas bases estadísticas al momento de la selección del modelo[16], [17]. Un posible criterio de parada es determinar cuando la agregación de una nueva neurona, no aporta al desempeño de la red.

1. Selección de neuronas en la capa oculta

La determinación del número de neuronas en la capa oculta, ha sido realizada por diferentes técnicas que pueden ser agrupadas en las siguientes categorías:

Métodos de construcción de modelos

- Regulación: Se asume que la función generadora de los datos es derivable:

$$H[f] = \sum_{i=1}^{N} [f(x_i) - y_i]^2 + \lambda\phi[f] \quad (3)$$

Donde el primer término mide la distancia entre los datos y la función deseada *f* y el *Φ* se conoce como parámetro de regulación o penalización. Una de las principales dificultades de este método es la complejidad.

- Algoritmos destructivos (prunning): Después de calcular una medida de relevancia de las neuronas es necesario definir un algoritmo que puede recortar los nodos automáticamente[16].
- Algoritmos constructivos: Se selecciona un pequeño conjunto de datos de entrenamiento y el resto de datos se toman como candidatos[18].
- métodos evolutivos: Son procesos en los cuales se utilizan algoritmos genéticos, para descubrir cuál es el tamaño de la red y el valor de los pesos, simultáneamente [17], [19].

Criterios de parada

Los criterios de parada están enfocados a determinar cuándo se finaliza el proceso de agregación o eliminación de neuronas en la capa oculta.

- Validación cruzada: La principal desventaja de este método es su lentitud, y por lo tanto no es muy recomendable[20].
- Pruebas estadísticas de significancia: La estimación de los valores de los pesos están directamente relacionadas al algoritmo de entrenamiento[21].

Como se puede observar, los problemas de cada uno de los métodos radican en los supuestos que debe cumplir el modelo, tanto como su complejidad de cálculo como exigencia computacional.

2. Estimación de los valores de los pesos.

Se ha concluido que los criterios en algunos casos son heurísticos, tales como valores de los parámetros del algoritmo de entrenamiento. Es aconsejable reducir el espacio de búsqueda utilizando mínimos cuadrados al estimar los pesos óptimos utilizando la expresión:

$$L_n(\vec{w}) = \frac{1}{2n}\sum_{i=1}^{n}[t_i - y_i]^2 \quad (4)$$

Donde $L_n(\vec{w})$ es una función de costo que se minimiza iterativamente [14], [22], [23]. La determinación del modelo final que permita realizar un entrenamiento adecuado no tiene por lo tanto bases teóricas que permitan decidir el correcto funcionamiento de los métodos, por lo tanto hace falta en la literatura construir un análisis comparativo entre los métodos de entrenamiento, para así hacer más sencillo el trabajo del experto.

III. APRENDIZAJE EN REDES NEURONALES USANDO REGULARIZACION BAYESIANA

Este método de aprendizaje se basa en la optimización de la función de densidad de probabilidad de los hiperparámetros α y β, los cuales explican la varianza del error, que se expresa como la diferencia entre la salida esperada y la salida estimada de la red neuronal.

Mediante la aplicación del teorema de Bayes, Se observa que se extrae información de los datos, para determinar a θ. Esta expresión lleva a la siguiente forma de error:

$$S(\theta) = s(\theta^*) + \frac{1}{2!}(\theta - \theta^*)^T H(\theta - \theta^*) \quad (5)$$

Nótese que esta es una expansión aproximada del verdadero término de error, la cual se halla mediante una aproximación gaussiana [21]. Para encontrar el θ_{MP} (más probable) se procede a la minimización de la función de error S(θ) por medio de un proceso de optimización no lineal, en este caso Levenbert-Marquardt [7], [11]. Para el cálculo de los hiperparámetros α y β, se fijan como constantes conocidas en el primer paso del proceso iterativo y después mediante la aproximación Bayesiana se infieren los valores más apropiados para estos hiperparámetros a partir de los datos de entrenamiento. Tales valores deben maximizar la función de distribución a posteriori $p(\theta \mid D)$.

Llegamos a obtener la varianza de la red que se expresa de la siguiente manera:

$$\sigma_t^2 = \frac{1}{\beta} + (\nabla_\theta y|_{\theta_{MP}})^T H^{-1}(\nabla_\theta y|_{\theta_{MP}}) \quad (6)$$

Básicamente, los hiperparámetros explican la varianza de la red neuronal, la cual se desea minimizar.

Estas cantidades nos permiten estimar los valores correspondientes a los parámetros α y β más probables, así:

$$\alpha_{MP} = \frac{\left(\sum_{i=1}^{N_M}\left(\frac{\lambda_i}{\lambda_i+\alpha}\right)\right)}{2E_\theta^{MP}} \quad (7)$$

$$\beta_{MP} = \frac{\left(N_y - \sum_{i=1}^{N_M}\left(\frac{\lambda_i}{\lambda_i+\alpha}\right)\right)}{2E_D^{MP}} \quad (8)$$

Siendo λi cada uno de los valores propios de la matriz G, la cual se obtiene como $\nabla\nabla E_D$. Mediante la estimación recursiva de estas cantidades se obtienen los valores presentados en las secciones siguientes.

IV. APRENDIZAJE EN REDES NEURONALES USANDO REGLA DELTA

V.

En esta sección desarrollaremos el método de entrenamiento de perceptrones multicapa, llamado regla delta. Por tal fin se mostrara inicialmente de manera formal la regla de entrenamiento, se prosigue con la especificación de algoritmo y por ultimo una apreciaciones respecto al modelo. Se utiliza un mecanismo de aprendizaje sobre las respuestas de acuerdo a la variación de los pesos *wi* según la expresión $w_i \leftarrow w_i + \Delta w_i$, donde $\Delta w_i = \eta(t - \hat{y})x_i$.

Es el aporte de corrección al nuevo valor de los pesos w_i, dada la diferencia entre t que es la salida esperada de la red neuronal y $\hat{y}$ es la salida obtenida a una velocidad η de aprendizaje.

Dado el conjunto de datos de entrenamiento $(\vec{x}, \vec{t})$,tasa de aprendizaje η, 2 neuronas en la capa oculta y 1 en la capa de salida, función de activación sigmoidea en la capa oculta y función de activación lineal a la salida, tenemos:

Algoritmo de Regla Delta

- $(\vec{x}, \vec{t})$ Datos de entrenamiento.
- η tasa de aprendizaje.
- W_{ih} Vector de pesos de la capa de entrada a la capa oculta.
- W_{bh} Vector de pesos de bias a la capa oculta.
- W_{ho} Vector de pesos de la capa oculta a la capa de salida.
- W_{bh} Vector de pesos de bias a la capa de salida.
- ε Error máximo permitido.

Como se puede observar, este algoritmo funciona mediante una corrección del error de los pesos de las conexiones internas de la red, y cuando se haya obtenido una condición de parada, se termina el entrenamiento.

V. SIMULACIONES Y ANALISIS DE RESULTADOS

En la presente sección se muestran los resultados obtenidos de varias redes neuronales usando regularización Bayesiana, en las cuales se modificaron los valores de los datos aleatorios iniciales normalizados correspondientes a los pesos para las neuronas de la capa oculta y de salida. En la primera parte de este apartado se presenta una comparación práctica de las dos metodologías antes

descritas, junto con la especificación del proceso de aproximación de las funciones de Hwang (ver Figura 1). Se pretende demostrar las bondades del método bayesiano, frente el método de regla delta, a la luz de los resultados obtenidos en el proceso de entrenamiento.

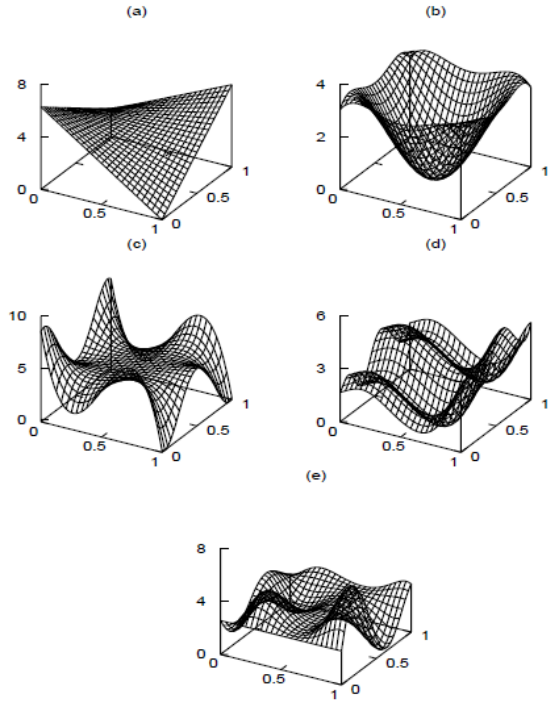

Figura 1: Funciones de Hwang. (a)Función de interacción simple $g^{(1)}$ (b)Función radial $g^{(2)}$ (c)Función armónica $g^{(3)}$ (d)Función aditiva $g^{(4)}$ (e)Función de interacción complicada $g^{(5)}$[24]

A. Algoritmo de entrenamiento

A continuación se detalla el proceso de entrenamiento mediante una herramienta de Matlab[25]. Se inicializan los valores correspondientes al número de neuronas ocultas, el número de ciclos iterativos externos e internos, el número de las variables de entrada, y el número de salidas de la red. Se inicializa también un vector de opciones, que permite definir características especiales de visualización de resultados, tal como pueden ser mostrar o no los valores de error, ver los resultados del chequeo del gradiente, definir los ciclos de entrenamiento, definir un punto de exactitud de las aproximaciones, eliminar los mensajes de error.

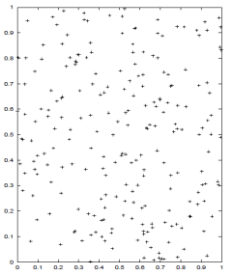

Figura 2: Puntos de entrenamiento (elaboración propia)

El protocolo de la simulación utilizó 10000 puntos generados de una distribución uniforme (0,1) en un campo cuadrado 1 x 1 entrenar la red. Para entrenar la red neuronal se utilizaron 225 puntos (figura 2) seleccionados de manera aleatoria en cada una de las funciones de Hwang en cada una de las 20 diferentes estructuras de red. Se realizan 100 ciclos externos con 400 ciclos internos y en cada uno de los ciclos externos se recalculan los pesos e hiperparámetros, mediante los datos de entrenamiento.

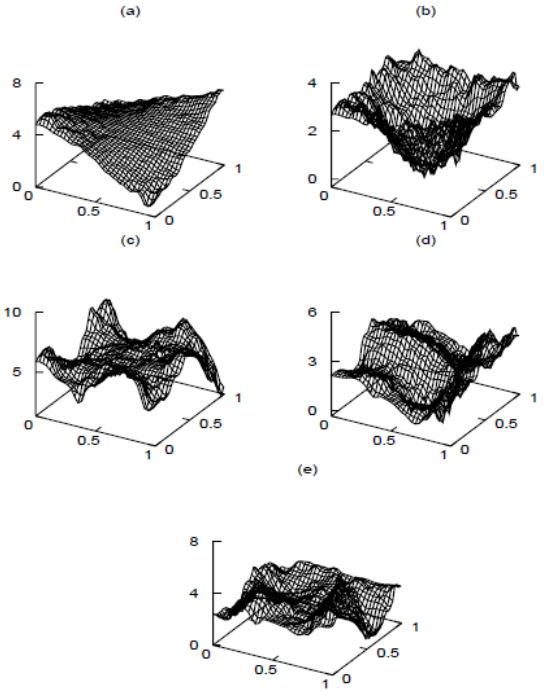

Figura 3: Datos de entrenamiento con ruido (a)Función de interacción simple $y^{(1)}$ (b)Función radial $y^{(2)}$ (c)Función armónica $y^{(3)}$ (d)Función aditiva $y^{(4)}$ (e)Función de interacción complicada $y^{(5)}$ (elaboración propia)

Las funciones de Hwang son evaluadas con redes neuronales con una arquitectura clásica contra una arquitectura Bayesiana, obteniendo los siguientes resultados en una función clásica de varianza, tomada de [22], y permite evaluar si los datos obtenidos mediante la red neuronal, se aproximan de una manera clara a los valores de entrenamiento, además es proporcional al error cuadrático medio.

Para los modelos sin ruido se obtuvieron los siguientes resultados (figura 4): Para la función de interacción simple (G1) se observa un alto grado de ajuste, lo cual nos asegura el funcionamiento de manera correcta en funciones que no involucren alta complejidad, además la probabilidad del error del ajuste de la neurona nos indica que es explicada con sólo 3 neuronas, de una manera óptima.

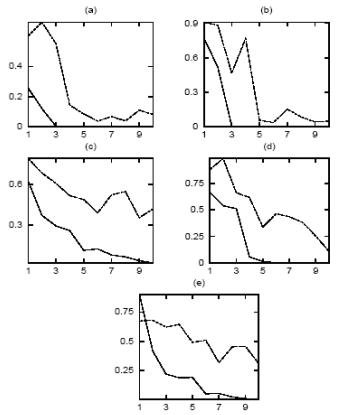

Figura 4: Error cuadrático medio (ECM) obtenido en entrenamiento de las funciones de Hwang por cantidad de neuronas, para modelos sin ruido. La línea punteada denota el error de entrenamiento mediante regla delta y la continua entrenamiento mediante método bayesiano. (a)ECM Función de interacción simple. (b)ECM Función radial. (c)ECM Función armónica. (d)ECM Función aditiva. (e)ECM Función de interacción complicada (elaboración propia)

Para la función radial, se observa un ajuste de 70% para una arquitectura de 5 neuronas, lo cual nos lleva a pensar que el nivel explicativo de la red ha bajado considerablemente, y para una estructura de 10 neuronas se logra poca ganancia en el ajuste.

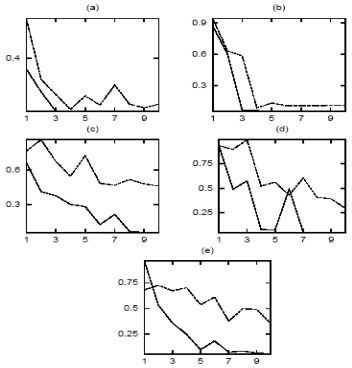

Figura 5: Error cuadrático medio (ECM) obtenido en entrenamiento de las funciones de Hwang por cantidad de neuronas, para modelos con ruido. La línea punteada denota el error de entrenamiento mediante regla delta y la continua entrenamiento mediante método bayesiano. (a)ECM Función de interacción simple. (b)ECM Función radial. (c)ECM Función armónica. (d)ECM Función aditiva. (e)ECM Función de interacción complicada (elaboración propia)

La selección de la arquitectura del modelo depende directamente del algoritmo de entrenamiento, ya que como se puede apreciar, para modelos diferentes de regularización bayesiana y tradicional, los resultados son muy diferentes, lo cual se ve reflejado en complejidad de los mismos.

B. Aportes del trabajo

Como se indicó anteriormente, un problema fundamental en la metodología fue determinar el efecto del número de neuronas en el algoritmo de entrenamiento; respecto a este punto, se encontraron evidencias numéricas, que confirman esta hipótesis, ya que los resultados fluctúan de manera extraña, observándose incrementos en el error para modelos con un número superior de neuronas en la capa oculta. Para las funciones de Hwang se observa un buen ajuste, con un error bajo, inclusive menor que el obtenido utilizando regla delta, ya que el método de regularización Bayesiana extrae el máximo conocimiento posible del conjunto de datos de entrada. No hay tampoco un método que especifique los pasos a seguir a la hora de comparar metodologías de entrenamiento de redes neuronales y como se ha mostrado en secciones anteriores, todos los modelos que tienen su base en el teorema de Kolmogorov, y además, como objetivo de dichos análisis era demostrar que los valores que toma el error en procesos como estos, debe disminuir o en el peor de los casos[26]. Continuar constante, pero se ha mostrado que tiene fluctuaciones de crecimiento.

VI. CONCLUSIONES

Los resultados del análisis comparativo entre metodologías de entrenamiento para redes neuronales permiten concluir que los métodos de regularización aquí presentados tienen |problemas respecto a la forma de construcción ya que ninguno de los dos tiene un decremento sostenido respecto al error de ajuste, lo cual va en contra del teorema de Kolgomorov, para este proceso, se sugiere emplear un método diferente de construcción de redes neuronales, que involucre la experiencia de la construcción de una red con n - 1 neuronas en la capa oculta, es decir, que el entrenamiento de las redes sea incremental, ya que se pierde el conocimiento adquirido al cambiar el número de neuronas.

En cuanto a los pesos, se observa que bajo la generación de pesos aleatorios, para cada una de las arquitecturas de red presentadas.

La regularización Bayesiana es evidentemente más robusta que el método de regla delta, ya que los resultados obtenidos en todo el proceso presentaron un mayor ajuste en estas arquitecturas.

Se observa que no hay nada que indique la carencia de neuronas en la capa oculta, lo cual no auspicia que se sigan agregando, además, dado el caso que hallan más neuronas de las necesarias, el proceso tampoco da información de ello. Hay que hacer una claridad respecto a los métodos de entrenamiento, ya que presentan comportamientos diferentes en cuanto al número de neuronas en la capa oculta, pero estos no son métodos de selección de arquitectura para redes.

Se sugiere para trabajos futuros explorar el problema mediante técnicas evolutivas híbridas, en las cuales se defina automáticamente el número de neuronas en la capa oculta, ya que según su metodología de construcción, podría tener mejor desempeño en este tipo de modelos.

REFERENCIAS

[1] D. C. Montgomery, E. A. Peck, G. G. Vining, and V. G. Pozo, *Introducción al análisis de regresión lineal*. Editorial Continental, 2004.

[2] R. Walpole and R. Myers, "Probabilidad y estadística para ingenieros," 1999.

[3] W. Mendenhall, "Probabilidad y estadística para ingenieria y ciencias," 1997.

[4] J. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, 1991.

[5] M. S. Al-Batah, N. A. Mat Isa, K. Z. Zamli, and K. A. Azizli, "Modified Recursive Least Squares algorithm to train the Hybrid Multilayered Perceptron (HMLP) network," *Applied Soft Computing*, vol. 10, no. 1, pp. 236-244, Jan. 2010.

[6] W. Sarle, "Neural networks and statistical models," 1994.

[7] D. Charalampidis and B. Muldrey, "Clustering using multilayer perceptrons," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 71, no. 12, p. e2807-e2813, Dec. 2009.

[8] P. Mehra, "Artificial neural networks: concepts and theory," *Society Press Tutorial, Los Alamitos, CA:*, 1992.

[9] R. Herbrich and M. Keilback, *Neural networks in economy*. 1999.

[10] O. Westheider, "Predicting stock index returns by means of genetically engineered neural networks," 1997.

[11] E.-L. Silva-Ramírez, R. Pino-Mejías, M. López-Coello, and M.-D. Cubiles-de-la-Vega, "Missing value imputation on missing completely at random data using multilayer perceptrons.," *Neural networks : the official journal of the International Neural Network Society*, vol. 24, no. 1, pp. 121-9, Jan. 2011.

[12] T. Koskela and M. Lehtokangas, "Time series prediction with multilayer perceptron, FIR and Elman neural networks," *Proceedings of the World Congress on Neural Networks*, 1996.

[13] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.

[14] T. Masters, *Practical neural network recipes in C++*. Morgan Kaufmann, 1993.

[15] W. Duch and J. Korczak, "Optimization and global minimization methods suitable for neural networks," *Neural computing surveys*, vol. 2, pp. 163–212, 1998.

[16] U. Anders and O. Korn, "Model selection in neural networks," *Neural Networks*, vol. 12, no. 2, pp. 309–323, 1999.

[17] J. F. Unger and C. Könke, "An inverse parameter identification procedure assessing the quality of the estimates using Bayesian neural networks," *Applied Soft Computing*, vol. 11, no. 4, pp. 3357-3367, Jun. 2011.

[18] H. Braun and J. Weisbrod, "Evolving neural feedforward networks," in *Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms*, 1993, pp. 25–32.

[19] J. Branke, "Evolutionary algorithms for neural network design and training," in *In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications*, 1995.

[20] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761–767, 1998.

[21] B. T. Zhang, "Accelerated learning by active example selection," *International Journal of Neural Systems*, vol. 5, no. 1, pp. 67–76, 1994.

[22] H. Allende, C. Moraga, and R. Salas, "Artificial neural networks in time series forecasting: A comparative analysis," *KYBERNETIKA-PRAHA-*, vol. 38, no. 6, pp. 685–708, 2002.

[23] M. Correa, C. Bielza, and J. Pamies-Teixeira, "Comparison of Bayesian networks and artificial neural networks for quality detection in a machining process," *Expert Systems with Applications*, vol. 36, no. 3, pp. 7270-7279, Apr. 2009.

[24] J. N. Hwang, S. R. Lay, M. Maechler, R. D. Martin, and J. Schimert, "Regression modeling in back-propagation and projection pursuit learning," *Neural Networks, IEEE Transactions on*, vol. 5, no. 3, pp. 342–353, 1994.

[25] I. Nabney, *NETLAB: algorithms for pattern recognition*. Springer Verlag, 2002.

[26] W. Duch, K. Grabczewski, and others, "Searching for optimal MLP," in *in Proc. 4th Conf. Neural Networks and Their Applications*, 1999.