UNIVERSIDAD AUSTRAL DE CHILE

FACULTAD DE CIENCIAS DE LA INGENIERÍA

ESCUELA DE INGENIERÍA CIVIL EN INFORMÁTICA

Aplicación de Data Mining sobre un GIS de caracterización forestal.

TESIS DE GRADO PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO CIVIL EN INFORMÁTICA

PATROCINANTE:
Sr. Martín Solar M.
CO-PATROCINANTE:
Sra. Eliana Scheihing G.
INFORMANTE:
Sr. Carlos Bahamondez V.

Camilo José Ruiz-Tagle Molina

VALDIVIA – CHILE

2003

Agradecimientos.

Quiero agradecer a todos aquellos que hicieron posible, de una u otra manera, la realización de este trabajo de tesis.

A Mauricio Acuña por el material facilitado en los comienzos de este trabajo; a Carlos Bahamondez por la posibilidad de aplicar esta técnica en un problema real y por su valiosa ayuda; A los profesores Eliana Scheihing y Martín Solar por su apoyo permanente y cooperación; Al profesor Luis Alvarez y Christian Lazo por facilitarme un lugar de trabajo y un computador donde escribir mi tesis; A vero por todas las molestias; A mi hermano Chicho por su ayuda, su visión y confianza, y a todo el instituto por su apoyo.

Quiero agradecer de manera especial a mis padres y a mi familia por todos estos años de apoyo; sin su ayuda, amor y cariño, nada de esto sería posible.
Finalmente dedico esta tesis a todo mis colegas que creen en lo que hacen.

Valdivia, 14 de Octubre de 2003

De : Martín Gonzalo Solar Monsalves

A : Directora Escuela Ingeniería Civil en Informática

Ref. : Informe Calificación Trabajo de Titulación

Nombre Trabajo de Titulación:

"APLICACION DE DATA MINING SOBRE UN GIS DE CARACTERIZACION FORESTAL"

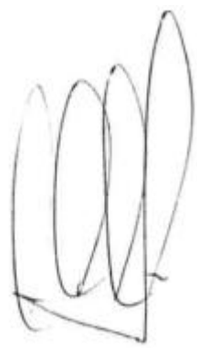
Nombre Alumnos:

Camilo José Ruiz-Tagle Molina

Evaluación:

Cumplimiento del objetivo propuesto	7,0
Satisfacción de alguna necesidad	6.5
Aplicación del método científico	7,0
Interpretación de los datos y obtención de conclusiones	7,0
Originalidad	6.5
Aplicación de criterios de análisis y diseño	7,0
Perspectivas del trabajo	7,0
Coherencia y rigurosidad lógica	7,0
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración	7,0
Nota Final	7,0

Sin otro particular, atte



Martín Solar Monsalves

Universidad Austral de Chile
Instituto de Informática

Valdivia, 7 ele Noviembre 2003
Comunicación Interna N" 18/2003

DE : Eliana Scheihing, Profesora del Instituto de Informática

A : DIRECTORA ESCUELA INGENIERIA CIVIL EN INFORMATICA

MOTIVO:

INFORME TRABAJO DE TITULACION

Nombre Trabajo de Titulaeión: **"Aplicación de Data Mining sobre un GIS de una caracterización forestal".**

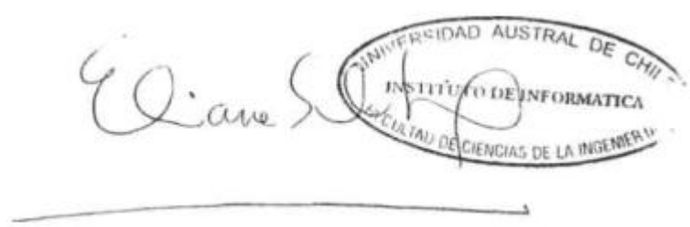
Nombres del Alumno: **Camilo José Ruiz-Tagle Molina**

Nota: ..... 7,0 .... **siete coma cero** .....
(en números) (en letras)

FUNDAMENTO DE LA NOTA:

Trabajo muy interesante y riguroso. Valiosa contribución en la práctica del uso de los algoritmos de Data Mining. Buena presentación.

Atentamente,



INFOR

Valdivia, 21 de Octubre 2003

De : Carlos Bahamóndez
A : Directora Escuela de Ingeniería Civil en Informática,
Universidad Austral de Chile.

Estimada Sra. Directora,

Tengo el agrado de comunicar a Ud. mi evaluación como profesor informante respecto del trabajo de titulación del Sr. Camilo Ruiz-Tagle Molina "APLICACION DE DATA MINING SOBRE UN GIS DE CARACTERIZACION FORESTAL".

El trabajo realizado por el Sr. Ruiz-Tagle ha cumplido con creces el objetivo propuesto describiendo en forma bien documentada y profunda la importancia de herramientas de Data Mining en el contexto de los Sistemas Geográficos de Información, los Sensores Remotos y su uso en el sector forestal.

Adicionalmente, el Sr. Ruiz-Tagle hace un uso pionero, de un caso real, de análisis de datos forestales por medio de redes neuronales y su comparación con redes bayesianas para interpolación de datos espaciales basados en verdad de terreno.

Dado lo anteriormente expuesto, califico el trabajo de titulación del Sr. Camilo Ruiz-Tagle Molina con nota 6,5 (seis coma cinco).

Sin otro particular, se despide atentamente,

Carlos Bahamondez
Instituto Forestal
Sede Los Lagos
Valdivia

INDICE

Resumen

Las características arquitectónicas de un Sistema de Información Geográfica (SIG) permiten realizar diversos análisis sobre los datos que contiene, haciendo uso de herramientas como ARC_INFO y ARC_VIEW, entre otras. Se puede potenciar aún más el análisis sobre los datos de un SIG, haciendo uso de técnicas avanzadas de extracción de conocimiento no trivial. El KDD (Knowledge Discovery Database) se define como el descubrimiento de conocimiento en base de datos que permite extraer modelos novedosos para describir realidades contenidas en los datos. Una de las fases más importantes del KDD es conocida como Data Mining o Minería de datos, que corresponde a la aplicación de algoritmos para encontrar patrones de comportamiento ocultos en los datos. La técnica será aplicada sobre un conjunto de bases de datos de un SIG que caracteriza el nivel de vulnerabilidad ambiental del suelo de la comuna de Lanco, provincia de Valdivia, Chile, utilizando información obtenida por el proyecto "Desarrollo y aplicación de alternativas de manejo para el abastecimiento continuo de bienes y servicios con bosques nativos", desarrollado por el Instituto Forestal (INFOR), sede Los Lagos. Con ella se pretenderá asignar el nivel de vulnerabilidad ambiental a porciones de terreno o pixeles que carecen de mediciones directas de terreno.

Palabras claves: Análisis avanzado de datos, Data Mining, GIS, Redes Neuronales.

Summary

The architectural characteristics of a geographic information system (GIS) allow to carry out diverse analysis on the contained data, making use of tools like ARC_INFO, ARC_VIEW, among others. One can even more increase the analysis on the GIS data making use of advanced techniques of extraction of non trivial knowledge. The KDD is defined as Knowledge Discovery in Database that allows to extract novel models to describe realities contained in the data. One of the most important phases of this is known as data mining that corresponds to the application of algorithms to find hidden behavior patterns in the data. The technique will be applied on a set of data bases of a SIG that characterizes the level of environmental vulnerability of the ground of the commune of Lanco, province of Valdivia, Chile, using data obtained by the project "Development and application of alternatives of handling for the continuous supply of goods and services with native forests", developed by Instituto Forestal (INFOR), seat Los Lagos. With this technique it will be tried to assign the level of environmental vulnerability to portions of land or pixels that lack direct land measurements.

Key words: Advanced analysis of data, data Mining. GIS, Neural Networks.

1. Capítulo1: Introducción.

1.1. **Aplicación de Data Mining sobre un GIS de caracterización forestal.**

Un Sistema de Información Geográfico GIS se define como *"una Herramienta computacional compuesta por equipos, programas, DATOS georreferenciados y USUARIOS que requieren organizar, analizar, automatizar procesos y producir información"* [Sol,2002].

Las características arquitectónicas de este sistema permiten realizar diversos análisis sobre los datos contenidos en éste, haciendo uso de herramientas distintas como es el caso de ARC_INFO, ARC_VIEW, y otras herramientas GIS. Cada uno de estos análisis responden a una necesidad específica de información lo que provoca que estos sean superficiales y fuertemente guiados por los resultados.

Se puede potenciar aún más el análisis sobre los datos de un GIS haciendo uso de técnicas avanzadas de extracción de conocimiento no trivial.

El KDD (Knoledge Discovery Database) se define como el proceso de descubrimiento en Bases de datos, que permite extraer modelos novedosos, que describan realidades contenidas en los datos. También se puede utilizar como un procedimiento científico de validación de conocimiento obtenido empíricamente.

Una de las fases más importantes de este proceso KDD es conocida como Data Mining o Minería de datos, que corresponde a la aplicación de algoritmos basados en redes neuronales, clustering difuso y discreto, series de tiempo, etc., para encontrar patrones de comportamiento ocultos en los datos.

La técnica será aplicada, haciendo uso de redes neuronales, sobre un conjunto de datos pertenecientes a un raster de pixeles de 120 x 120 metros, para resolver el problema de asignación de niveles de vulnerabilidad ambiental del suelo de la comuna de Lanco, representada por el área de estudio.

Estos estudios serán apoyado por el uso del software Data Engine, una potente suite que permite hacer análisis diversos sobre datos y aplicación de algoritmos de data Mining.

Esta suite ha sido utilizada con éxito en investigaciones científicas, de marketing, de procesos industriales de control de calidad y optimización.

1.2. Antecedentes existentes.

Este trabajo de tesis se enmarca dentro de la necesidad de la asignación de niveles de vulnerabilidad ambiental del suelo a porciones de terreno o pixeles que carecen de mediciones directas de terreno, en el contexto del proyecto "Desarrollo y aplicación de alternativas de manejo para el abastecimiento continuo de bienes y servicios con bosques nativos", desarrollado por el Instituto Forestal (INFOR), sede Los Lagos [INFOR,2002].

En el **Capítulo2, "Planteamiento del Problema",** se hace una detallada descripción al problema real, exponiéndose la solución inicial desarrollada por los ejecutores del proyecto haciendo uso de redes bayesianas, y se plantean las bases para el desarrollo completo de este trabajo.

1.3. Objetivos Generales.

Realizar estudios de Data Mining sobre los datos contenidos en un sistema GIS de caracterización forestal perteneciente a la comuna de Lanco, con el fin de asignar niveles de vulnerabilidad ambiental del suelo a posiciones geográficas que carecen de datos de terreno.

1.4. **Objetivos Específicos**.

- Conocer el conjunto de técnicas utilizadas en un proceso de extracción de conocimientos en base de datos (KDD).
- Caracterizar en profundidad los fundamentos que rigen la construcción de redes neuronales Multilayer Perceptrón, y su uso en Data Engine.
- Modelar el problema de asignación de niveles de vulnerabilidad del suelo a píxeles de un GIS asociado a la comuna de Lanco, haciendo un estudio comparativo con el uso de redes Neuronales Multilayer Perceptrón y un modelamiento usando redes Bayesianas Multinet.
- Caracterizar diversas configuraciones de redes Multilayer Perceptrón y redes Multinet Bayes frente al problema de estudio.
- Caracterizar el comportamiento del aprendizaje y la calidad de la clasificación cuando se utilizan poblaciones iguales o distintas en las fases de construcción y clasificación en dichos modelos. Validar KDD y Data Mining como una metodología robusta y seria para el análisis avanzado de datos.
- Conocer y utilizar algoritmos de Data Mining.
- Conocer y utilizar en profundidad el software Data Engine.
- Conocer y utilizar la suite BNSoft: Bn power Constructor y Bn power predictor para la construcción e inferencia de redes bayesianas Multinet, con los datos de estudio.

1.5. Introducción al Data Mining: Conceptos básicos.

Se entiende por data mining a la etapa de extracción de patrones de comportamiento no trivial de los datos, que forma parte del proceso global conocido como "descubrimiento de conocimiento en base de datos" (kdd: knowledge discovery in databases). [Fay,1996], [Web,2000]. Data mining entrega diversas técnicas para encontrar patrones en grandes conjuntos de datos. Este enfoque multidisciplinario combina los resultados e intuiciones provenientes de varias ramas científicas tales como la estadística, el aprendizaje de máquina, tecnologías difusas (fuzzy technologies), y redes neuronales.

En la literatura existen muchas definiciones para caracterizar expresiones como "descubrimiento de conocimiento en base de datos" (kdd: knowledge discovery in databases) y "data mining", donde se hacen las siguientes aseveraciones:

"el descubrimiento de conocimientos en base de datos es el proceso no trivial de identificar patrones en datos que sean válidos, novedosos, potencialmente útiles y, por último, comprensibles" [Fay,1996].

"..data mining se refiere al acto de extraer patrones o modelos a partir de los datos.." [Fay,1996]

una representación frecuente de un proceso típico de KDD, contempla los siguientes nueve pasos [Web,2000]:

1.Desarrollar una comprensión del dominio de la aplicación: Determinar, de manera global, los beneficios de un estudio de Data Mining sobre los datos y las posibles alternativas de análisis, para delimitar el dominio del problema. Conocer el alcance, utilidad y significado de los datos.

2.Crear un conjunto de datos objetivo: Una vez, escogida la tarea global de Data Mining sobre los datos, se creará un conjunto de datos objetivo.

3.Limpieza y pre - procesamiento de datos: Consiste en entender la estructura y formato de los datos de estudio, para poder determinar duplicidad de representación de datos, atributos que son relevantes para el estudio, y comportamiento de éstos.

Se hacen en esta etapa estudios preliminares de los datos, pruebas de correlación, análisis de dependencia de variables, y pruebas estadísticas que den una visión general del comportamiento de los datos.

4.Reducción y transformación de los datos: Se eliminan atributos irrelevantes para el estudio, campos nulos y atributos que se encuentren representados más de una vez.

Finalizando esta etapa, se preparan los datos para la aplicación de los algoritmos de Data Mining. Esta preparación puede incluir adaptación del formato de los datos, normalización, etc.

5.Elegir la tarea de data mining: Se elige la tarea especifica de Data Mining para un análisis más detallado. Aquí se debe decidir si el propósito es, por ejemplo, la agrupación de objetos, la regresión o el modelaje de dependencia. Sobre la base de esta decisión, los más importantes algoritmos de data mining deben ser seleccionados, (paso 6), los que se usan en la búsqueda real de patrones de datos (paso 7).

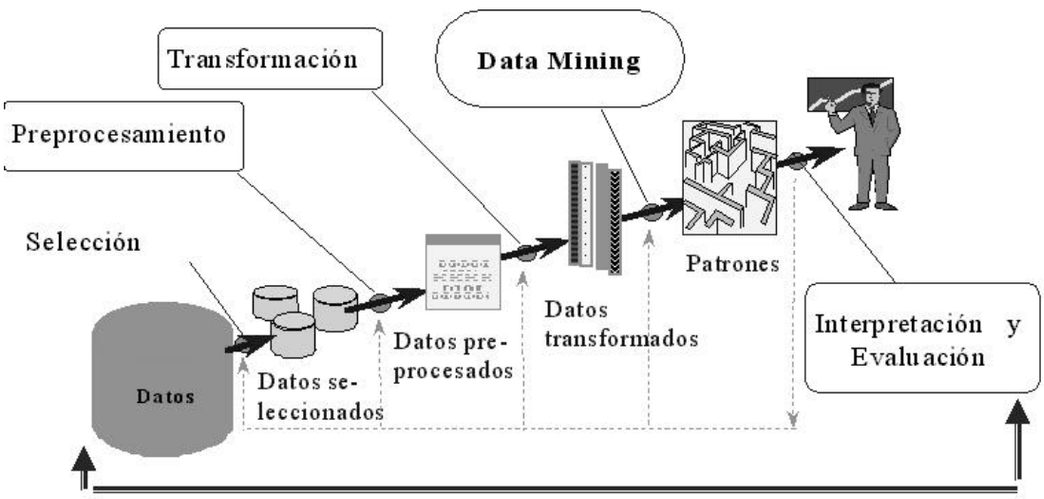
6.Data mining: Ejecución de los algoritmos de Data Mining. Esto puede incluir un contraste de uno o varios algoritmos, evaluando su desempeño, salidas obtenidas, etc. Se utilizarán algoritmos de redes neuronales, agrupamiento difuso, clustering, redes de Kohonen, según sea el caso.

7.Evaluar el resultado de Data mining: Testear, en base al conocimiento experto de los datos de estudio de Data Mining, la coherencia de los resultados arrojados por los algoritmos de Data Mining. Esta etapa corresponde a la validación del conocimiento

descubierto por el proceso, interpretación de las correlaciones obtenidas, o validación de resultados del algoritmo con conocimiento experto.

8. Consolidar el conocimiento experto: Sacar un aprovechamiento real de los resultados obtenidos en el estudio de Data Mining, y su aprovechamiento en el contexto de la problemática de estudio.

A continuación, un esquema global del proceso de Descubrimiento del Conocimiento en las Base de Datos (KDD).



Proceso global: KDD (Descubrimiento de conocimiento en BD)

Capítulo2: Planteamiento del Problema.

1.6. Introducción.

El instituto Forestal, INFOR, dentro del contexto del proyecto "Desarrollo y aplicación de alternativas de manejo para el abastecimiento continuo de bienes y servicios con bosques nativos", se encuentra con la problemática del **desarrollo del índice de Vulnerabilidad ambiental.** La importancia de este último es que permite evaluar la propensión al deterioro de los ecosistemas. La información obtenida con este permite tomar decisiones para realizar actividades de manejo sustentable[INFOR,2002].

1.7. EL Índice de Vulnerabilidad Ambiental: Breve definición.

El termino de **vulnerabilidad** se refiere a la propensión a dañarse debido a la pérdida de protección o al riesgo de ser afectado por un impacto negativo [Kal,1999]. Cuando la vulnerabilidad es considerada alta, la resilencia será considerada baja y viceversa. Los dos términos son utilizados opuestamente en el campo de los índices de vulnerabilidad. Es por ello que la vulnerabilidad y la resilencia son dos términos muy relacionados.

1.8. Objetivo de la modelación.

La modelación de estos procesos tiene por objeto determinar qué variables influyen en la ocurrencia y su condición de suceptibilidad. Es así, que en modelos de cambio de uso de la tierra, es posible tomar un enfoque espacial o no espacial. Los modelos espaciales son apropiados si se tienen datos asociados al territorio y son especialmente útiles en explicar patrones espaciales de deforestación.

Por ello el objetivo final del desarrollo del índice de vulnerabilidad ambiental es la modelación espacial de variables y factores disponibles de mayor importancia que determinan la degradación ecosistémica frente a las actividades forestales de manejo.

1.9. Proceso de desarrollo.

El desarrollo del índice de vulnerabilidad se llevó a cabo sobre el estudio de 78 conglomerados- cada uno de ellos con tres parcelas – ubicados **aleatoriamente** en la comuna de Lanco, provincia de Valdivia. Para el cálculo del índice intervienen 24 variables. Este grupo de variables se pueden dividir en tres componentes ecológicos: Suelo-agua, paisaje y fauna. Cada uno de estos componentes define el cálculo de un índice. La problemática de esta tesis se encuentra asociada al índice de vulnerabilidad del Suelo-agua.

1.10. Cálculo del índice de vulnerabilidad de Suelo - agua.

El **índice de vulnerabilidad de Suelo-agua** dice relación con las variables que afectan al suelo y que son causantes del aporte de sedimentos a los cursos de agua, procesos erosivos y aquellos factores que condicionan las actividades de manejo y cosecha en el sitio[INFOR,2002].

1.10.1. Variables que intervienen en su construcción.

Es importante destacar que en el cálculo de este índice intervienen variables cuya procedencia son mediciones directas en terreno, o bien se construyen a partir de éstas.

Estas variables son enumeradas, con la única intención de ilustrar el problema, en la siguiente tabla:

Variables que intervienen en el cálculo del índice de vulnerabilidad ambiental SUELO – AGUA	
Profundidad del suelo (PROF)	Humedad del suelo (HUM)
Profundidad de hojarasca (HOJAR)	Pendiente (PEND)
Profundidad de humus (PHUMUS)	Exposición (EXPO)
Cobertura de copas (COBERT)	Forma de la pendiente(FPEND)
Textura (TEXT)	Relieve (REL)
Estructura del suelo (ESTRUCT)	Caminos de acceso (CAM)
Moteado (MOT)	Drenaje (DREN)
Grava (GRAVA)	Tipo de erosión (EROSTIP)
Pedregosidad (PEDRE)	Grado de erosión (EROSGRAD)

1.10.2. Proceso de desarrollo

Una vez definidas las variables que intervienen en el cálculo de este índice, se procedió a realizar un tratamiento estadístico de las variables utilizando el método de análisis multivariado de componentes principales (ACP) mediante el software Statgraphics Plus. Una de las características principales de este método es la reducción del número original de variables cuando sea posible y la generación de nuevas variables que puedan expresar la información contenida en el conjunto original de datos [San,1982].

Con los resultados de este método y la reducción a un menor número de factores de información proporcionados por estas variables, se construyó el índice de vulnerabilidad ambiental, como es indicado en la siguiente ecuación[INFOR,2002]:

$$IV = \frac{1}{\sum_{i=1..n} x_i} \left[\sum_{i=1..n} x_i C_i \right]$$

Donde:

IV = Índice de vulnerabilidad

X_i = i-ésimo valor propio

C_i = i-ésima coordenada

De este modo, se consideraron 7 componentes, los cuales explican el **62% de la varianza contenida en los datos originales**. Esta cantidad fue considerada satisfactoria, dada la naturaleza del problema. El siguiente cuadro muestra los valores propios y los coeficientes utilizados para estimar el Índice de Suelos-Agua.

Componente	Valor propio	Coeficiente (V. Propio/sumatoria)
1	2.44318	0.220067448
2	1.93116	0.173947663
3	1.88603	0.169882615
4	1.32036	0.118930351
5	1.31174	0.118153912
6	1.12472	0.101308237
7	1.08477	0.097709774
Sumatoria	11.10196	

Ya se puede definir con precisión la fórmula del cálculo del índice de vulnerabilidad:

$$IV_{Suelo-agua} = \frac{1}{11.10196}\left[2.44318x_i + 1.93116x_i + 1.88603x_i + 1.32036x_i + 1.31174x_i + 1.12472x_i + 1.08477x_i\right]$$

Finalmente, mediante análisis cluster se diferenciaron los valores de índice por parcela, agrupándolos en familias. El presente análisis se llevó a cabo considerando 5 familias o clusters, cuyo ordenamiento concuerda con la escala de valores del modelo que indican valores de vulnerabilidad de acuerdo a la fragilidad de algunos de sus elementos. El

método de clustering utilizado fue el Centroide y la distancia fue calculada mediante el método City-Block.

Los valores del índice para las 220 parcelas pueden ser consultados en [INFOR,2002]. Estos valores varían entre 2. 2644 y –1.1238, los que, cuando se hacen más positivos representan zonas más vulnerables o frágiles para el suelo y, genera más peligro de erosión y aporte de sedimentos hacia los cursos de agua. Esto va de acuerdo a cómo se rankearon las variables y de su entrada al índice, ya que en ellas, valores más altos indican una mayor vulnerabilidad a su degradación.

Para evitar clasificar los valores del índice en límites que pudieran ser arbitrarios, se procedió a realizar un análisis de clusters, para que de acuerdo a su distancia pudieran ser separados en niveles de vulnerabilidad. En este caso, se utilizaron cinco niveles en donde el más bajo índica niveles reducidos de vulnerabilidad y aquellos con niveles altos (1) muestran el caso extremo. Sectores donde se deben aumentar al máximo las prácticas que mitiguen el impacto sobre el medioambiente. El cuadro siguiente muestra en detalle las características de cada uno de estos niveles.

Niveles de vulnerabilidad para Suelo-Agua

Cluster	Nivel	Descripción
5	Bajo	Sectores planos o de baja pendiente, sin problemas evidentes de erosión. Pueden presentar problemas de drenaje a través de anegamiento en ciertos períodos del año, evidenciados por moteado en el perfil del suelo, buen contenido de materia orgánica.
4	Moderado	Sectores de pendientes bajas o moderadas, sin presencia aparente de erosión, buen contenido de materia orgánica, problemas de anegamiento en sectores planos, profundidad media a baja del suelo.
3	Medio	Se evidencian problemas de erosión laminar y de canalículos, agravados por un incremento en el nivel de las pendientes y baja profundidad del suelo en algunas áreas.
2	Alto	Erosión del suelo evidenciada a través de deslizamientos y/o canalículos. Áreas de altas pendientes expuestas a precipitaciones dominantes, bajo contenido de materia orgánica en el suelo.
1	Extremo	Erosión evidente del suelo, presencia de cárcavas y/o zanjas, altas pendientes y caracterizado por una fuerte exposición a las lluvias dominantes.

Como se ha visto, el cálculo del índice de vulnerabilidad ambiental para 220 parcelas (que provienen del estudio de 78 conglomerados- cada uno de ellos con tres parcelas),

esta fuertemente determinado por la información de datos provenientes directa o indirectamente de terreno.

Estas parcelas tienen una ubicación espacial determinada por un pixel, que se encuentra dentro de una imagen satelital perteneciente a un GIS de caracterización forestal. Existe la necesidad de asignar finalmente niveles de vulnerabilidad ambiental sobre pixeles, (o parcelas), donde **no existe información proveniente de variables explicatorias**. Estos pixeles, dada su naturaleza GIS, son caracterizados por otro grupo de variables, las **variables auxiliares**. Estas últimas se encuentran conformadas por:

- C1: banda 1 Landsat 7 ETM+
- C2: banda 2 Landsat 7 ETM+
- C3: banda 3 Landsat 7 ETM+
- C4: banda 4 Landsat 7 ETM+
- C5: banda 5 Landsat 7 ETM+
- m: pendiente del terreno en grados asociado a cada pixel
- expo: exposición en grados por pixel.
- Altitud: altitud sobre el nivel del mar en metros.
- Cluster: cluster de grado de vulnerabilidad. Este valor se puede obtener para los pixeles que cuentan con información proporcionada por variables explicatorias. Este atributo será imprescindible en el entrenamiento de las redes neuronales y en la construcción de las redes bayesianas, porque permite entregar ejemplos de pixeles que pertenecen a cada uno de los niveles de vulnerabilidad. Este punto será desarrollado en los capítulos 6 (“Modificación de la solución inicial usando redes bayesianas”) y 8 (“Solución utilizando Redes neuronales Multilayer Perceptrón”).

La definición y el significado de las variables C1,C2,C3,C4 y C5 son profundizados en el capítulo 2: Fundamentos sobre GIS.

Por último, la posición geográfica del pixel viene dada por la variable **idd:**

idd: identificador de registro que representa al conglomerado y la parcela de un punto dado. Ejemplo: idd=112, conglomerado=11, parcela=2.

La estimación de niveles de vulnerabilidad basados en datos recolectados en terreno y su relación con variables auxiliares explicatorias que concurren al área de estudio, generan la posibilidad de asignar valores de vulnerabilidad a otras posiciones geográficas del área de estudio que carecen de mediciones directas de terreno.

El problema de asignar valores de vulnerabilidad en posiciones sin información más que aquella proveniente de las variables explicatorias, no es trivial, ya que normalmente estos valores de vulnerabilidad no son magnitudes físicas que obedecen a un proceso definido, sino más bien son efecto de un resultado combinado de variables, que no necesariamente pesan de igual forma frente a cada situación. Modelos de regresión pueden ser aplicados pero sus resultados suelen ser pobres y contradictorios, en especial porque los modelos de regresión representan situaciones medias de los datos, y de esta forma los errores de clasificación suelen ser muy altos [INFOR,2002].

Soluciones alternativas y atractivas en este contexto son aquellas de aplicación de redes neuronales (data mining) o de árbol de decisiones o de redes de bayes.

Para este proyecto, inicialmente se hizo un modelamiento del problema con redes Bayesianas al problema de asignación de niveles de vulnerabilidad a posiciones geográficas en el área de estudio de la comuna de Lanco, representada por un raster de pixeles de 120x120 metros.

1.11. Solución inicial usando Redes Bayesianas.

1.12. ¿Que es una Red Bayesiana?

Una red Bayesiana corresponde a un grafo acíclico dirigido con una distribución condicional de probabilidad en forma de tabla para cada nodo del gráfico. Si consideramos un nodo $n \in N$ donde N representa el dominio de los atributos o las variables, y a los arcos $a \in A$, entre los nodos representando a las variables condicionales [INFOR,2002].

En general, en redes bayesianas se hace uso de la regla de la cadena estadística, la cual define que una probabilidad conjunta. Se puede calcular mediante una cadena de probabilidades condicionales. Esta es:

$$P(a,b,c,d,f,g,h) = P(a|b,c,d,f,g,h) * P(b|c,d,f,g,h) * .... * P(h)$$

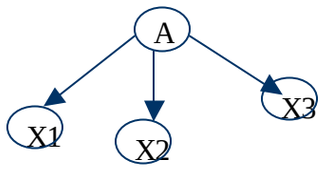
Las redes Bayesianas aprovechan este proceso a través del uso del concepto de independencia condicional, dado que ciertos eventos no se correlacionan una vez que algún atributo ocurre o es conocido. En la ecuación anterior se puede dar que

$$P(a|b,c,d,f,g,h) = P(a|b,c,d,f,g)$$

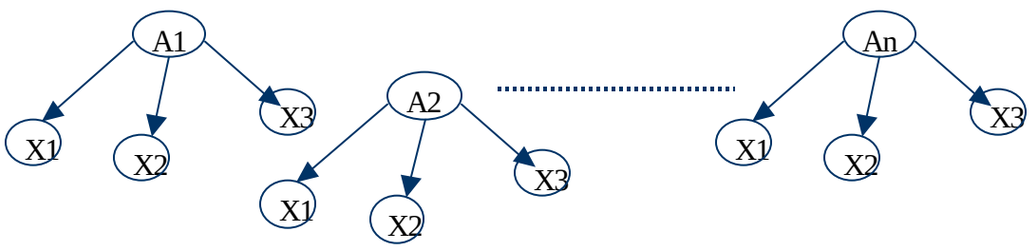
Lo cual implica que “a” puede ser determinado por (b,c,d,g) independientemente de (h). La potencia de esto es que la probabilidad conjunta no requiere ser completamente definida para ser calculada, sino que se pueden usar probabilidades condicionales más pequeñas para esta tarea, entonces se tienen los grafos acíclicos dirigidos donde cada arco $a \in A$ representa una condicional de probabilidad

Las redes bayesianas constan de dos procesos o fases, el aprendizaje y la inferencia para clasificar instancias. Entre los clasificadores los más relevantes son: Naï ve-Bayes (BN), Naï ve-Bayes aumentado en árbol (TAN), Naï ve-Bayes de redes aumentadas (BAN), Multinet Bayes y Redes bayesianas generales (GBN). En general todos estos

clasificadores se diferencian en la forma en que se organizan las estructuras. La aproximación más simple de estructura es la BN, la cual no permite mayores conexiones, como se puede visualizar en la figura a continuación:



Por otra parte, la Multi-net Bayes comprende una aproximación más compleja, aunque eficiente de clasificación, trabajando estructuras específicas a nivel de subconjuntos de los atributos a inferir, según el siguiente diagrama:



Con $A1 \subset A, A2 \subset A, ...An \subset A$.

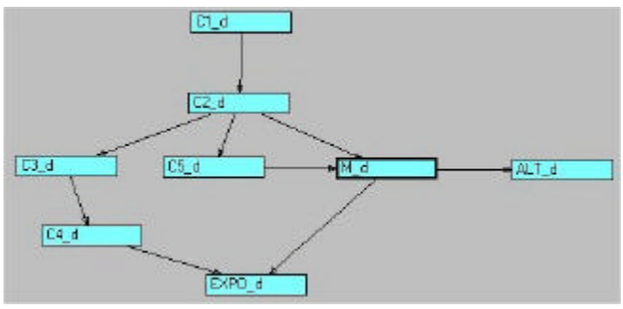
Es esta alternativa de clasificador (Multi-Net Bayes), la que se aplicó al problema en cuestión de asignar (clasificar) valores de vulnerabilidad a pixeles definidos en el raster del área de estudio [INFOR, 2002].

1.13. Análisis de la solución inicial.

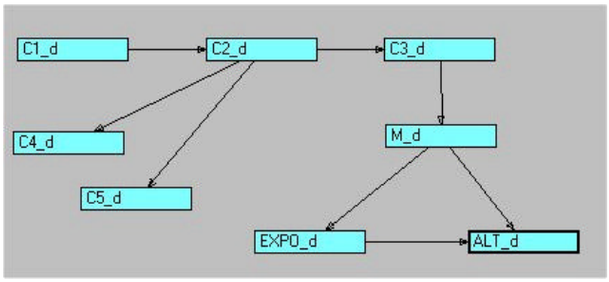
En el enfoque de Multi-net Bayes clasificador, se construyeron las redes locales basádonse en los valores del atributo del índice de vulnerabilidad, a partir del cual se plantean 5 subconjuntos que estructuran una red local específica. De acuerdo a los valores de cada subconjunto se plantean los siguientes niveles de vulnerabilidad a clasificar [INFOR,2002]:

Grado de vulnerabilidad	Descripción
5	Baja
4	Moderada
3	Media
2	Alta
1	Extrema

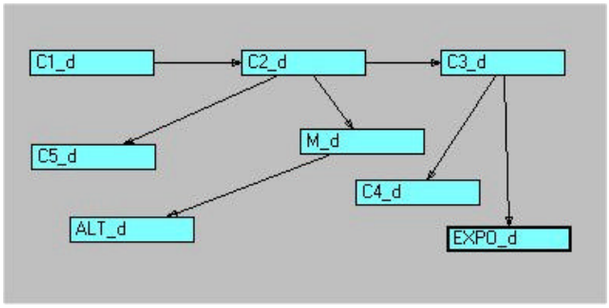
Las redes resultantes para cada grado de vulnerabilidad resultaron en:



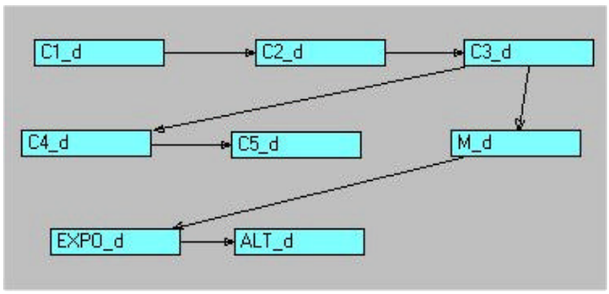
Red local bayesiana para grado de vulnerabilidad baja (5)



Red local bayesiana para grado de vulnerabilidad Moderada (4)



Red local bayesiana para grado de vulnerabilidad Media (3)



Red local bayesiana para grado de vulnerabilidad Alta y Extrema (2,1).

Donde:

- C1_d: Banda 1 landsat 7 ETM+
- C2_d: Banda 2 landsat 7 ETM+
- C3_d: Banda 3 landsat 7 ETM+
- C4_d: Banda 4 landsat 7 ETM+
- C5_d: Banda 5 landsat 7 ETM+
- M_d: Pendiente del terreno en grados asociada a cada pixel.
- Expo_d: exposición de grados en Pixel.
- Alt_d: altitud sobre el nivel del mar en metros.

1.14. Resultados.

Como resultado final de la aplicación del proceso de clasificación por la Multi-net de bayes antes descrita, se generó una asignación de valores de vulnerabilidad a cada pixel definido para clasificación en el área de estudio. El resultado de la clasificación frente a los datos de entrenamiento fue de 70,05% $\pm$ 6.09 al 95% del nivel de confianza, el cual fue considerado aceptable. La matriz de confusión generada para este clasificador corresponde a la siguiente:

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	0	0	**0**	0
3	0	2	**16**	3	2	**23**	69.56
4	1	10	27	**113**	16	**167**	67
5	0	1	1	2	**23**	**27**	85
Total	1	13	44	118	41	**152/217**	
%precisión en la clase	0	0	36.4	95.7	63.4		**70.05%**

1.15. **Conclusiones.**

Al analizar la matriz de confusión resulta evidente que, a pesar de una buena calidad de clasificación global (70.05) los datos utilizados para los grados de vulnerabilidad presentan baja cantidad (1 y 13). De este modo resulta extremadamente complejo para el algoritmo resolver con datos de entrenamiento tan pobres. Los resultados fueron considerados satisfactorios para los ejecutores de este proyecto. Sin embargo, consideraron importante, dada la dificultad de los datos y del problema, recurrir a otras tecnologías que se ajusten a estas necesidades, para tener otra alternativa que pudiese complementar y confirmar estos resultados.

La alternativa propuesta, y finalmente el trabajo fundamental de esta tesis, fue resolver este problema haciendo uso de técnicas de data mining y redes neuronales para el problema de asignación de niveles de vulnerabilidad del suelo a los pixeles del área de estudio.

Debemos agregar un antecedente clave, con respecto al desarrollo de este experimento inicial: las redes bayesianas fueron construidas, entrenadas, y testeadas con los mismos datos, es decir, no se conoce el desempeño de clasificación de éstas con datos desconocidos durante la fase de construcción.

Para resolver este problema, se construyó un conjunto de datos de entrenamiento y otro de test, y se diseñaron experimentos de redes neuronales, probando diversas configuraciones, arquitecturas y caracterizando su comportamiento en cada contexto(capítulo8: Solución utilizando redes neuronales Multilayer Perceptrón). Previo

a esto, se utilizaron los mismos conjuntos de datos y se construyeron experimentos utilizando redes Multi-net bayes (capítulo6: Modificación de la Solución inicial usando redes bayesianas), para hacer una comparación de ambas metodologías sometidas al mismo problema y contexto de datos.

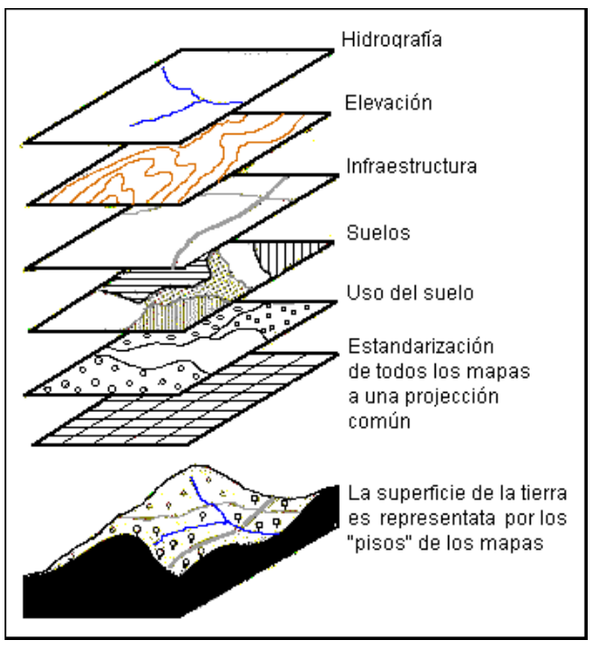
2. Capítulo3: Fundamentos sobre Gis.

2.1. ¿Qué es un GIS?

Existen diversas definiciones para caracterizar un GIS. Se puede decir que un GIS es una "Herramienta computacional" compuesta por equipos, programas, datos georreferenciados y usuarios que requieren organizar, analizar, automatizar procesos y producir información [Sol,2002].

El término Sistema de Información Geográfica o SIG se aplica actualmente a los sistemas computarizados de almacenamiento, elaboración y recuperación de datos con equipo y programas específicamente diseñados para manejar los datos espaciales de referencia geográfica y los correspondientes datos cualitativos o atributos.

En general la información espacial se representa en forma de "capas", en los que se describen la topografía, la disponibilidad de agua, los suelos, los bosques y praderas, el clima, la geología, la población, la propiedad de la tierra, los límites administrativos, la infraestructura (carreteras, vías férreas, sistemas de electricidad o de comunicaciones), como se representa en la siguiente figura [Sol,2002]:



De este modo, podemos enumerar las principales características de un GIS:

- Manejo de grandes volúmenes de información.
- Posibilidad de información de distintas fuentes y escalas.
- Rapidez en el procesamiento de la información y obtención de productos cartográficos.
- Capacidad de modelar información.
- Manejo de información georreferenciada.

2.2. ¿Cómo se representa la información en un GIS?

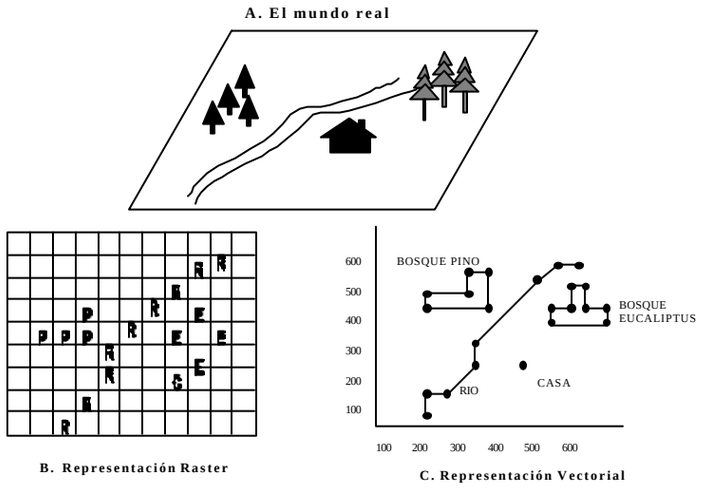
Existen dos Modelos de datos espaciales que permiten representar la información contenida en un GIS [Sol,2002]:

1. **Modelo Vectorial:** En él los objetos y sus atributos (condiciones) son representados por puntos y líneas que definen sus límites. La posición de cada objeto es definida por su localización en un "mapa" que es organizado a través de un sistema de coordenadas de referencia. Cada posición en el mapa tiene un único valor de coordenada.
2. **Modelo Raster:** El espacio es dividido regularmente en "celdas" (usualmente formadas por cuadrados).La localización de los objetos geográficos y sus atributos, está definida por la posición que las celdas ocupan en las columnas y las filas. El área que cada celda representa define la resolución de la información.

Los estudios realizados en este trabajo de tesis se aplican sobre un GIS basado en el modelo de datos espaciales Raster.

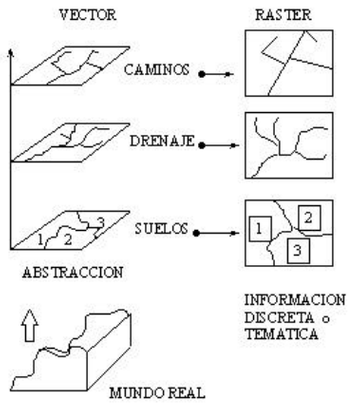
La siguiente figura ilustra las características principales de estos dos modelos, al representar una porción de terreno del mundo real. En ella existe dos bosques, uno de pino y otro de eucaliptus que se encuentran espacialmente separados por un río que cruza la porción de terreno. Junto al río existe una casa. En el caso del modelo Raster cada uno de estos elementos son representados por conjuntos de celdas vecinas con una

Modelos de Datos Raster y Vectorial

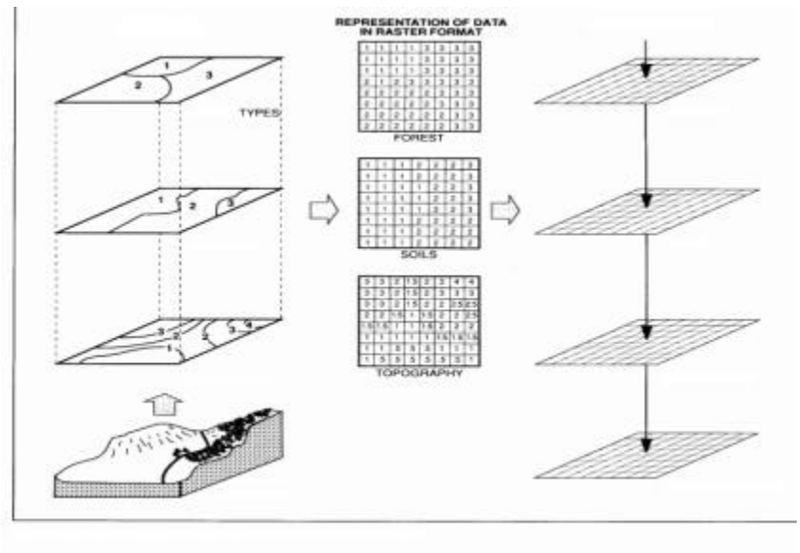


posición geográfica, dada por la posición fila - columna de cada una que forman parte de un arreglo de celdas que representa toda el área de interés. Estas celdas contienen un valor que indicará qué se está representando en ella. Este valor o atributo de la celda puede representar una porción de río (r), bosque de Eucaliptus (e) o de Pinos (p).

La fidelidad con que se representa una situación del mundo real, depende de la resolución utilizada con este arreglo de celdas. Puntualmente, la resolución está dada por el área real representada por la celda. Además, se asume que el valor de cada atributo asociado a ésta es homogéneo en toda el área que representa. En este sentido, este modelo de datos espacial es una discretización del mundo real, lo que permite hacer representaciones temáticas de la realidad, como se muestra en la siguiente figura:



En la siguiente figura, además podemos ilustrar el hecho que cada celda de cada capa tiene una celda correspondiente en cada una de las otras capas, producto de la georreferenciación [Sol,2002].



2.3. Ingreso de datos en un Gis.

La información que es ingresada en un GIS pertenece a dos grandes grupos: La **información espacial** que representa la localización geográfica (líneas, puntos y áreas) y la **información alfanumérica** de atributos asociados (no espacial), que representa información descriptiva de los elementos geográficos. Durante el proceso de ingreso de

datos, se debe tener especial cuidado de lograr una precisa correspondencia entre estos dos tipos de informaciones.

Para lograr el ingreso de datos en un Gis, podemos distinguir 4 grandes grupos de tecnologías comúnmente usadas:

- **Ingreso de datos por digitación (teclado):** utilizado para el ingreso de atributos, es decir toda la información no espacial. Se ingresan los datos a las tablas asociadas directamente con cada capa de información.

- **Ingreso de datos por digitación manual:** tecnologías utilizadas para el ingreso de datos espaciales: mesa digitalizadora y un cursor conectado a esta. Toda la información ingresada de este modo es inmediatamente procesada por el GIS.

- **Scanner:** Es el sistema más ampliamente utilizado en los últimos años. Es rápido, y automatizado para el ingreso de datos en forma digital. Se genera una imagen digital del mapa, cuya salida es una imagen en formato raster.

- **Dispositivos GPS (Global Positioning System):** Es un sistema basado en la interacción entre un dispositivo GPS y un conjunto de Satélites espaciales que son capaces de capturar la posición exacta en el tiempo de éste.

2.4. Algunas definiciones sobre percepción remota.

La percepción remota se define como el arte de y la ciencia de obtener información sobre un objeto, zona o fenómeno a través del análisis de información obtenida por un dispositivo que no se encuentra en contacto con el objeto, zona o fenómeno de investigación (Llilesand y Kiefer, 1994), [Sol,2002].

Existen diversas condiciones que deben cumplirse para llevar a cabo la percepción remota: Una fuente emisora de energía (el sol), un objeto a ser estudiado (una zona de bosques), un sistema capaz de capturar información (censores remotos). En el caso de estos últimos la información puede ser captada por emisión, por reflexión o por emisión – reflexión. En cualquiera de estos casos el flujo de energía entre la cubierta terrestre y el sensor constituye una forma de radiación electromagnética. Para que se logre la percepción remota es necesaria una interacción entre el sensor y los objetos de estudio. El tipo de datos susceptibles de ser obtenidos por percepción remota puede originarse en la distribución o cambios de diferentes fuerzas (gravedad, magnetismo), de ondas acústicas o de energía electromagnética. La forma más común para caracterizar las diversas ondas electromagnéticas es a través de la localización de las **longitudes de ondas en el espectro electromagnético.**

En percepción remota, el tipo de energía que se detecta es caracterizado por su posición en el espectro electromagnético.

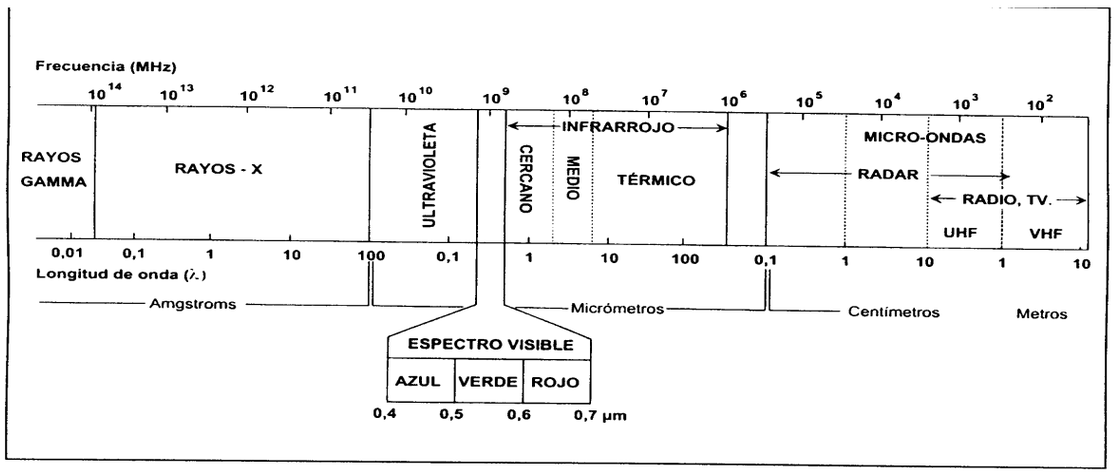


FIGURA 3. Espectro electromagnético (Chuvieco, 1996).

De acuerdo a la plataforma y al sensor que se utilice, los sistemas de sensores remotos se pueden caracterizar de acuerdo a la resolución temporal, espacial, espectral y radiométrica.

- **Resolución Temporal:** Es la frecuencia de observaciones del sensor sobre un objeto. No depende del sensor sino de la plataforma.
- **Resolución Espacial:** Es el mínimo detalle espacial (pixel) que registra un sensor. Depende del sistema óptico del sensor y de la altitud de la plataforma.
- **Resolución espectral:** se caracteriza por un determinado número de bandas, y sus respectivos rangos espectrales con los que se capta la información electromagnética. Los satélites pueden clasificarse como monoespectrales para caso de los radares que presenta sólo una banda, y multiespectrales, de 2 a varias bandas y más.
- **Resolución Radiométrica:** Es la capacidad de un sensor para registrar pequeños cambios de energía. Se mide en cantidad de niveles de grises o de cuentas digitales.

2.5. La plataforma espacial Landsat.

Consiste en una serie de satélites de los Estados Unidos comenzada en 1972. Inicialmente fueron denominados ERTS (Earth Resources Technology Satellite). A través de diversos instrumentos han estado observando la tierra para evaluación de la tecnología de percepción remota en estudios y manejo de recursos naturales. En la actualidad los datos del instrumento TM (Thematic Mapper) son comerciales[Sol,2002].

La siguiente tabla entrega algunas características técnicas de Landsat.

Resolución Temporal	16 días
Resolución Espacial	30 m. 120m (para banda 6)
Resolución espectral (micrómetros)	Banda1:0,45-0,52 Banda2:0,52-0,60 Banda3:0,63-0,69 Banda4:0,76-0,90 Banda5:1,55-1,75 Banda6:10,74-12,5 Banda7:2,08-2,35
Resolución Radiométrica	8 bits.

Cada una de las bandas, de acuerdo a las capacidades de la resolución espectral, es aplicada para explorar distintos objetos y fenómenos de la superficie terrestre. La siguiente tabla enumera las principales aplicaciones por banda.

Canal (banda espectral)	Principales Aplicaciones.
1	Mapeo de aguas costeras. Diferenciación entre suelo y vegetación. Diferenciación entre vegetación conífera y decídua .
2	Reflectancia de la vegetación verde sana.
3	Absorción de Clorofila. Diferenciación de especies vegetales.
4	Levantamiento de Biomasa. Delineamiento de cuerpos de agua.
5	Medidas de humedad de la vegetación. Diferenciación entre nubes y nieve.
6	Mapeo de estrés térmico en plantas. Otros mapeos térmicos.
7	Mapeo hidrotermal.

3. Capítulo 4: Fundamentos de data mining y conceptos básicos.

El almacenamiento de la información en los sistemas informáticos actuales es algo sencillo y barato. Estos sistemas tienen cada vez una capacidad mayor. Este incremento tiene el siguiente efecto: es poco costoso guardar datos del funcionamiento de nuestros negocios o de nuestros procesos, o de nuestros clientes, etc. , por lo que nuestras bases de datos crecen hasta límites insospechados[Dae,2002].

Cuando decidimos iniciar ese proceso de almacenamiento de datos, lo hacemos con la intención de analizarlos posteriormente. Sin embargo, cuando llega el momento, el análisis que se realiza suele ser bastante superficial, y guiado por los resultados que esperamos encontrar al analizarlos. Lo normal es utilizar un paquete estadístico, una hoja de cálculo, para localizar correlaciones entre variables, establecer medias y varianzas, e intentar modelar de esta forma nuestra información[Dae,2002].

En esta montaña de datos existe información que no puede ser encontrada con los procedimientos de investigación habituales. La minería de datos permite encontrar relaciones ocultas que pueden ayudarnos a gestionar mejor nuestro negocio o proceso.

3.1. Estructuración de los datos.

Para poder analizar nuestros datos con fiabilidad, es necesario que exista cierta estructuración y coherencia entre los mismos. Dentro de una organización, existen diversos sistemas de bases de datos que satisfacen necesidades especificas de información. Cada departamento posee su sistema de base de datos que interactúa de manera parcial con otro sistema. Sin embargo en este escenario nos podemos encontrar con distintos problemas:

1. Diferentes tipos de datos representando el mismo concepto. (Ejemplo: Representación de la fecha con 2 o 4 dígitos).

2. Diferentes claves para representar el mismo elemento: un cliente en un sistema puede ser representado por un código de cliente propio, mientras que en otro sistema de la empresa se encuentra representado por su Rut.
3. Diferentes niveles de precisión para representar un dato: los números reales no siempre se almacenan de la misma forma.

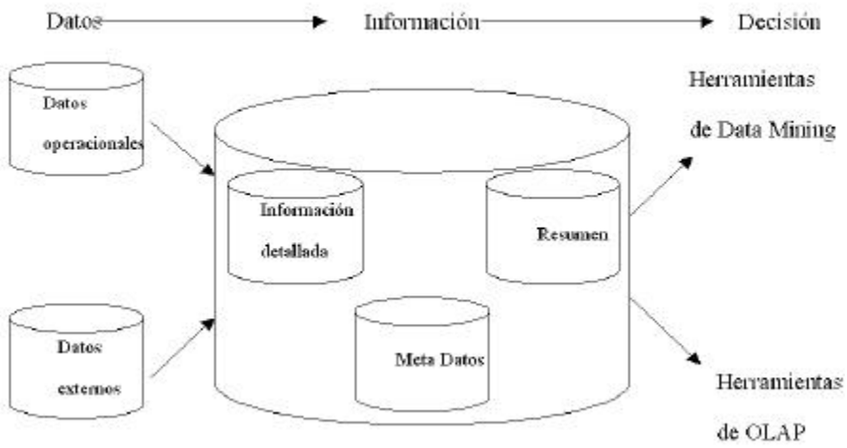
Es cierto que cada una de estas fuentes de datos puede ser manejada por separado. Seguro que hay quien opina que los datos están en diferentes bases de datos por que representan informaciones y procesos distintos, y que no tiene sentido estructurar la información más allá de lo que está. Pero no es menos cierto que nos estamos hurtando a nosotros mismos la posibilidad de descubrir un conocimiento que va más allá de cada uno de las parcelas de nuestro negocio: un conocimiento que representa la interacción de entre diferentes procesos, que es, precisamente, donde se encuentra la información más valiosa.

3.2. Data Warehousing.

Un data warehouse es un almacén estructurado de la información clave de nuestro proceso o negocio, que integra datos provenientes de todos los departamentos, sistemas, etc., y que nos permite analizar el funcionamiento de nuestra compañía y tomar decisiones sobre su gestión[Dae,2002].

La siguiente figura ilustra la arquitectura típica de un Data Warehouse.

Arquitectura de un Data Warehouse.



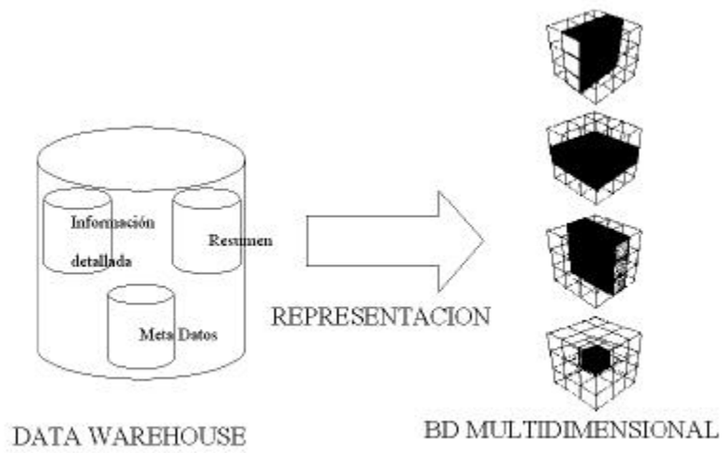
Fuente: Anahory, Murray (1997): Data Warehousing in the Real World.

No se trata de una simple agregación de las bases de datos locales. Existen algunas diferencias importantes:

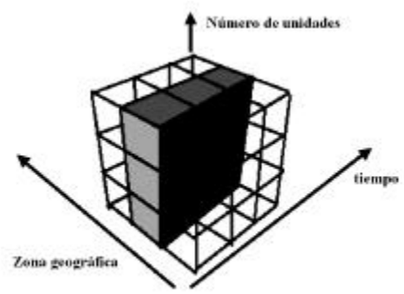
Base de Datos Operativa	**Data Warehouse**
Almacena la información de un sector del negocio o proceso.	Almacena grandes volúmenes de datos (información histórica e integración de datos de múltiples aplicaciones) .
Se opera mediante los cuatro mecanismos clásicos: añadir, eliminar, modificar, imprimir.	Se opera mediante análisis exploratorios de los datos. (base de datos multidimensional)
Se actualiza continuamente, cada vez que se elimina, añade, o modifica un dato.	Se actualiza a intervalos regulares, típicamente dentro de un proceso controlado.
Se orienta hacia la elaboración de informes periódicos.	Se ofrece información bajo demanda.
Orientada a la operación del sistema.	Orientada hacia el análisis del sistema.
Operación mediante OLPT	Operación mediante OLAP

La estructura de esta gran base de datos (data warehouse) es multidimensional, con diferentes puntos de vista que reflejan distintos aspectos del negocio o proceso. Así, los responsables del producto pueden analizar su evolución a lo largo del tiempo en diferentes sectores y localización geográfica.

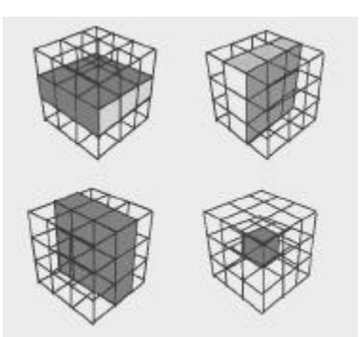
El ejemplo más clásico para representar un data warehouse es el de un cubo de datos, del que se pueden extraer diferentes rodajas o puntos de vista. Se puede analizar una parte concreta, o estudiar el conjunto global. El mecanismo mediante el cual se hace análisis a los datos contenidos en el data warehouse, se conoce como OLAP (Procesamiento Analítico en línea).



En una base de datos multidemsional ,frecuentemente se contrastan pares de atributos y su comportamiento a través del tiempo. Supongamos que deseamos analizar el número de unidades vendidas de un producto en una zona geografica determinada, y el comportamiento de esto a lo largo del tiempo. La siguiente figura muestra esta representación.



La consulta realizada sobre esta base de datos multidimencional, determinará qué porción del cubo será destacada por la herramienta OLAP. La siguiente figura ilustra el resultado de cuatro consultas distintas hechas sobre un cubo de datos.



3.3. Información oculta de los datos.

A estas alturas, ya va pareciendo claro que si almacenamos la información más relevante de nuestro proceso o negocio en un sistema que acumula y acumula datos sin parar, un análisis razonable nos puede permitir descubrir tendencias, localizar grupos de datos con comportamiento homogéneo, establecer relaciones, etc.

Esta información se encuentra oculta en los datos y será necesario utilizar todas las técnicas a nuestro alcance para obtenerla. El objetivo que nos planteamos es localizar relaciones entre atributos de nuestro data warehouse. Estas relaciones pueden ser del tipo:

- Para una gran superficie: Mas del 60% de las personas que adquieren queso fresco compran también algún tipo de mermelada.
- Para un departamento de fidelizacion de una campaña aérea: muchos usuarios que hacen vuelos de menos de 3 días a Berlín alquilan un coche en el aeropuerto.
- Para un operador de telefonía: Durante el mes siguiente al lanzamiento de una campaña de descuento en llamadas internacionales por parte de una compañía de la competencia, nuestros pequeños clientes redujeron su consumo en este sector, mientras que los grandes clientes lo mantuvieron.

Esta información puede ser extraída haciendo uso de diversas técnicas y ninguna de ellas debe ser despreciada, sino agregada al resto para obtener mejores resultados.

3.4. Qué es y qué no es minería de datos.

Minería de datos: es la extracción no trivial de información implícita, previamente desconocida y potencialmente útil, a partir de los datos.

Para conseguirlo, hace uso de diferentes tecnologías que resuelven problemas típicos de agrupamiento automático, clasificación, asociación de atributos, y detección de patrones secuenciales. La minería de datos es, en principio, una fase dentro de un proceso global denominado Descubrimiento del conocimiento en bases de datos, (knowledge discovery in databases o KDD), aunque finalmente haya adquirido el significado de todo el proceso, en lugar de la fase de extracción del conocimiento.

Es habitual que los expertos en estadística confundan la minería de datos con un análisis estadístico de éstos, de empresas dedicadas al procesamiento estadístico que venden sus productos como herramientas de minería de datos.

La diferencia fundamental es muy clara:

Supongamos que llegamos a la siguiente conclusión, después de un estudio:

“Mas del 60% de las personas que adquieren queso fresco compran también algún tipo de mermelada.”

- Utilizando un paquete estadístico, es necesario conocer a priori que existe una relación entre el queso fresco y la mermelada, y lo que realizamos con nuestro entorno estadístico es una cuantificación de dicha relación.
- Utilizando técnicas de data mining, la consulta que se realiza a la base de datos (data warehouse) busca relaciones entre parejas de productos que son adquiridos

por una misma persona en una misma compra. De esta información, el sistema deduce, junto a muchas otras, la afirmación anterior.

Como podemos ver, en este proceso se realiza un acto de descubrimiento de conocimiento real, puesto que no es ni siquiera necesario sospechar que existe una relación entre estos dos productos para encontrarla.

3.5. Niveles de abstracción del conocimiento extraido.

La evolución de la tecnología ha facilitado y automatizado en gran medida las tareas de análisis de información.

Para decidir cual es la técnica más adecuada para una determinada situación, es necesario distinguir el tipo de información que se desean extraer de los datos. Según su nivel de abstracción, el conocimiento contenido en los datos puede clasificarse en distintas categorías y requerirá una técnica más o menos avanzada para su recuperación[Dae,2002]:

- **Conocimiento evidente:** Información fácilmente recuperable bajo una simple consulta SQL.

Ejemplo:

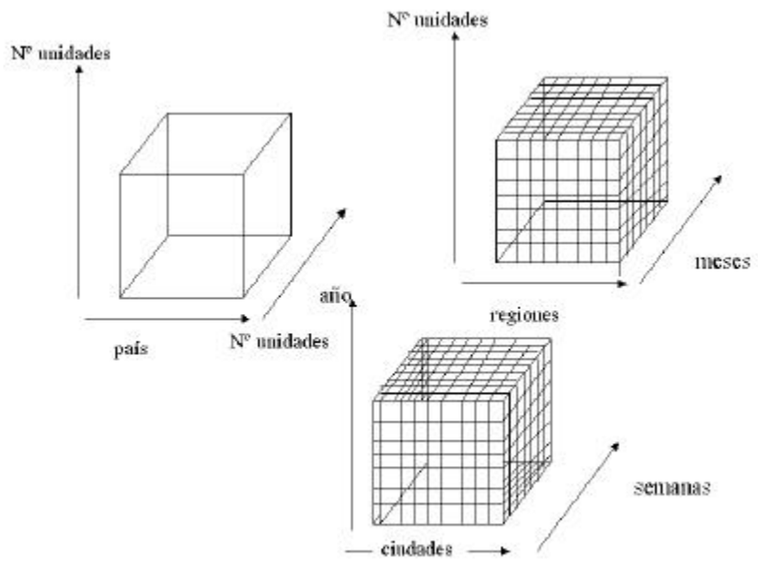
Cuales fueron las ventas de un determinado producto en Valdivia durante marzo?.

- **Conocimiento Multidimensional:** Se consideran los datos con una cierta estructura. Por ejemplo, en vez de considerar cada transacción individualmente, las ventas de una compañía pueden organizarse en función del tiempo y de la

zona geográfica, y analizarse con diferentes niveles de detalle.(país, región, localidad).

Técnicamente, se trata de reinterpretar una tabla con n atributos independientes como un espacio n-dimensional, lo que permite detectar algunas regularidades difíciles de observar con la representación monodimensional clásica. Este tipo de información es la que analizan las herramientas OLAP, que resuelven de forma automática cuestiones tales como: Cuáles fueron las ventas en Chile el pasado marzo? Aumentar el nivel de detalle: mostrar las de Valdivia.

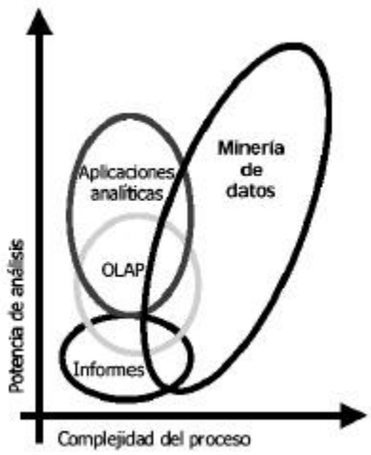
Cuando se aumenta el nivel de detalle en un análisis de los datos que se encuentran gráficamente representados en un cubo, se aumenta la precisión del análisis de este. La siguiente figura ilustra los niveles de detalle en el despliegue de la información sobre un cubo de datos.



Conocimiento oculto: Información no evidente, desconocida a priori y potencialmente útil, que puede recuperarse mediante técnicas de minería de datos, como reconocimiento de regularidades. Esta información es de gran valor, puesto que no se conocía y se trata de un descubrimiento real de nuevo

conocimiento, del que antes no se tenía idea, y que abre una nueva visión del problema. Un ejemplo de este tipo sería: qué tipo de cliente tenemos?,cuál es el perfil típico de cada clase de usuario?.

Como se ve, las técnicas disponibles para extraer información contenida en los datos son muy variadas y cada una de ellas es complementaria del resto, no exclusivas entre sí. Cada técnica resuelve problemas de determinadas características y , para extraer todo el conocimiento oculto, en general será necesario utilizar una combinación de varias.

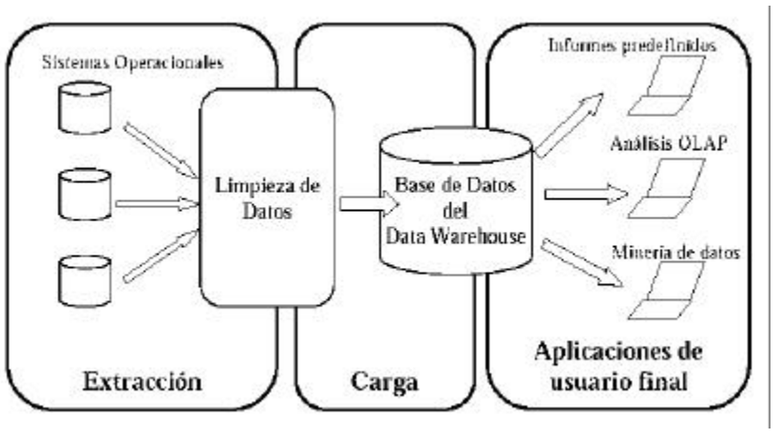


La figura representa la interacción existente entra las distintas técnicas disponibles para la extracción de conocimiento. Como puede verse, en la medida que aumentamos la complejidad del proceso y la potencia del análisis, la minería de datos es la tecnología más adecuada.

La minería de datos hace un análisis exploratorio, no corroborativo. Se trata de descubrir conocimiento nuevo, no de confirmar o desmentir hipótesis. Con cualquiera de otras técnicas es necesario tener una idea concreta de lo que se está buscando y, por lo tanto, la información que se obtiene con ellas esta condicionada con la idea preconcebida con que se aborde el problema. Con la minería de datos es el sistema y no el usuario el que encuentra las hipótesis, además de comprobar su validez.

La minería de datos permite obtener a partir de los datos un modelo del problema que se analiza, bien sean las ventas de un artículo para mejorarla campaña de marketing, las características técnicas de un producto en control de calidad o un proceso industrial cuyo control se desea optimizar, por citar algunos ejemplos. El modelo obtenido permitirá modelar el sistema o fenómeno real y obtener conclusiones aplicables en el día a día.

La Siguiente figura ilustra la extracción de conocimiento, en sus tres niveles de profundidad, sobre un Data Warehouse, proveniente de la síntesis de las base de datos corporativas:



3.6. Minería de Datos frente a OLAP Y DSS.

Los sistemas de ayuda a la decisión (DSS) son herramientas sobre las que se apoyan los responsables de una empresa, directivos y gestores, en la toma de decisiones. Para ello, utilizan:

- Un Data Warehouse, en el que se almacenan la información de interés para la empresa.
- Herramientas de análisis multidimensional (OLAP).

OLAP se define como análisis rápido de información multidimensional compartida.

Los DSS permiten la responsable de la toma de decisiones consultar y utilizar de manera rápida y económica las enormes cantidades de datos operacionales y de mercado que se generan en una empresa. Gracias al análisis OLAP, pueden verificarse hipótesis y resolverse consultas complejas. Además en el curso del análisis, la interpretación de los datos puede dar lugar a nuevas ideas y enfoques del problema, sugiriendo nuevas posibilidades de análisis.

Sin embargo, el análisis OLAP depende de un usuario que plantee una consulta o hipótesis.

Es el usuario el que la dirige y, por lo tanto, el análisis queda limitado por las ideas preconcebidas que aquel pueda tener.

La Minería de datos, junto a los DSS y el análisis OLAP, permite enriquecer el descubrimiento de conocimiento.

3.7. Diferencias entre Minería de Datos y OLAP para abordar el análisis

El análisis que realizan las herramientas OLAP es dirigido por el usuario, deductivo, parte de una hipótesis o de una pregunta del usuario y se analizan los datos para resolver esa pregunta concreta. Por el contrario, la minería de datos permite razonar de forma inductiva a partir de los datos para llegar a una hipótesis general que modele el problema.

Además las aplicaciones OLAP trabajan generalmente con datos agregados para obtener una visión global del negocio. Por el contrario, la mineria de datos trabaja con datos individuales, concretos, descubriendo las regularidades y patrones que presentan entre sí y generalizando a partir de ellos.

	OLAP	**MD**
Razonamiento	Deductivo	Inductivo
Trabaja con datos	Agregados	Concretos / Individuales.

Un ejemplo clarificará la diferencia entre ambas técnicas.

sistema OLAP/DSS:

pregunta: El año pasado se compraron más automóviles en Valdivia o en Osorno ?

respuesta: En Valdivia se compraron 5000 automóviles, mientras que en Osorno se compraron 6700.

Sistema de Data Mining:

Pregunta: hallar un modelo que determine las características mas relevantes de las personas que compran automóviles.

Respuesta: depende de la época del año y de la situación geografica. En invierno, los habitantes de Valdivia que pertenecen a un cierto grupo de edad y nivel de ingresos socioeconómicos probablemente compraran más automóviles que gente con las mismas características en Osorno.

Puesto que sus conclusiones son complementarias, puede ser conveniente utilizar Data Mining combinado con otras técnicas para obtener mejores resultados.

El objetivo final de cualquier proyecto de Data Mining puede resumirse en uno de estos dos:

- Ahorrar dinero mejorando la eficacia de sus actividades
- Ganar dinero descubriendo nuevas fuentes de sus beneficios.

Como se llega a estos resultados? .A partir de un conjunto de datos y de un conjunto de técnicas se puede llegar a unas determinadas conclusiones. Pero ¿ cómo se traducen los resultados de un proyecto de minería de datos en beneficios tangibles para la empresa?. Básicamente, estos resultados suponen una mejora de la información disponible y será al aplicar dicha información cuando se obtengan los beneficios.

Áreas donde se puede aplicar la técnica

Prácticamente en cualquier situación en la que se disponga de un conjunto de datos.

- **Deportes:** 28 de los 29 equipos que participan en la liga de baloncesto profesional americana NBA, utilizan técnicas de minería de datos para detectar patrones de comportamiento y relaciones entre variables de juego.
- **Marketing:** es uno de los campos cuyos resultados en minería de datos son más conocidos:

  El objetivo fundamental puede resumirse en detectar quien compra, cuando y dónde. Además se ha utilizado en targering, Fidelizacion de clientes. En Chile, los casos de Falabella y el BCI son conocidos.
- **Predicción:** Se pueden elaborar modelos que permitan estimar con precisión la evolución de una variable en el futuro. Este tipo de estudios se puede hacer en los campos más diversos, como la gestión comercial o el control de procesos.
- **Reducción de Riesgos:** Construcción de sistemas de evaluación automática de riesgos, basados en la experiencia previa.

- **Detección de Fraudes:** Obtención de modelos que permitan descubrir posibles fraudes, basándose en la detección de comportamientos anómalos, en comparación con datos históricos.
- **Control de calidad:** Detección de productos defectuosos, localización precoz de defectos, identificación de causas de fallos, análisis no destructivo.
- **Procesos industriales:** Automatización y optimización del control del proceso, Implementación de programas de mantenimiento predictivo.

4. Capítulo 5: Relación entre un Sistema de Información Geográfico y Data Mining.

Los sistemas de información geográficos son una representación de una realidad que surge de la relación entre distintas capas de información, dependiendo del fenómeno a ser modelado (topografía, características del suelo, bosques, etc., en el caso de una caracterización forestal). Además podemos representar otros aspectos de esa realidad geográfica, como puede ser, por ejemplo, toda la información entregada por los canales pertenecientes al sistema LandSat, como sucede en este trabajo de tesis.

Si siguiéramos agregando capas de información, estaríamos entregando a nuestro sistema GIS nuevos aspectos de la realidad que acontecen en esa parcela de suelo. De este modo, podemos hacer una analogía con los sistemas de información de una empresa. Cada subsistema de información representa distintas realidades que forman parte de los procesos de la empresa o finalmente, del negocio. Tenemos sistemas de información asociados al área de finanzas, contabilidad, marketing, de los procesos productivos, etc. Entonces se diseñaron estructuras como los Data Warehouse, que son una agregación de cada uno de estos subsistemas, que tienen un enfoque operativo, en un todo, el Data Warehouse cuyo sentido es el análisis de la información extraíble de esos datos con esa representación.

Lo que se plantea aquí, la motivación inicial de este trabajo de tesis, es que, un sistema de información geográfica, en la medida que cuente con datos pertenecientes a diversos dominios del conocimiento - diversas capas o coberturas geográficas- representa una estructura propicia para el análisis avanzado de datos, el Data Mining, del mismo modo como sucede con los Data Warehouse.

Finalmente, no es condición necesaria la construcción de un Data Warehouse para la aplicación de Data Mining. Lo que sucede es que el Data Mining tiene mayores probabilidades de éxito en la medida que contemos con datos que apunten a una misma realidad pero que provengan de los orígenes más diversos, como sucede con el data warehouse, o con un GIS de múltiples capas. Esta última condición sí es determinante para la aplicación de esta técnica.

Para reafirmar la idea anterior, los paquetes de software asociados al Data Mining, como sucede con Data Engine, se comunican con sistemas de Base de Datos mediante ODBC. En ellos los datos son importados a una tabla temporal mediante una query. De este modo el formato de los datos que alimentaran las redes neuronales, o los árboles de decisión, por nombrar dos tecnologías diversas asociadas al Data Mining, trabajan con tablas planas, o con consultas, no con la estructura de un modelo de datos.

5. Capítulo 6: Modificación de la solución inicial usando redes bayesianas.

5.1. Introducción

El presente informe tiene por objetivo definir una serie de experimentos usando la suite de redes bayesianas BNSOFT en sus módulos data preprocesor, data constructor y data predictor, para el uso de redes bayesianas de arquitectura multi-net, sobre los datos contenidos en la tabla frag_suelo. Lo que se pretende es caracterizar el comportamiento de dichas redes contrastándolas con las redes neuronales MLP1 a MLP7 (redes con datos de entrenamiento y test distintos) y las redes MLP1_1 a MLP1_7 (redes con datos de entrenamiento y test iguales), expuestas en el capítulo 8 "Solución utilizando redes neuronales Multilayer Perceptrón". Se ha demostrado experimentalmente que las redes con poblaciones de entrenamiento y test iguales (como es de esperar) poseen un desempeño de test superior; pero esto puede resultar engañoso, pues los datos de entrenamiento que permiten perfilar el comportamiento de las red son usados también en la fase de test.

Lo que se pretende aquí es realizar una serie de experimentos con redes bayesianas considerando los aspectos de entrenamiento y test (en este caso, llamados construcción de la red e inferencia, respectivamente) en ambos escenarios (poblaciones distintas o iguales). Además, en el caso de las redes bayesianas, se agrega un aspecto novedoso; estas redes permiten incluir el conocimiento experto, imponiendo relaciones de causalidad entre los nodos (variables), distintas de las relaciones que surgen entre los nodos de manera *natural* en la construcción de las red. Las características de la suite Bnsoft y el proceso de construcción de redes bayesianas multinet, junto con los algoritmos que rigen su construcción serán expuestos en el anexo nº2 "La Suite Bnsoft y la construcción de redes Multinet".

5.2. El experimento Lanco_bnsoft_6.

Este experimento consistió en la creación de un conjunto de redes bayesianas de arquitectura multi-net que se diferencian en tres aspectos principales:

- La construcción de la red y la inferencia de ésta con los mismos datos de entrenamiento y test, o con datos distintos.

- La inclusión o exclusión de conocimiento experto en la construcción de la red.

- El criterio usado por el software para permitir relacionar los nodos en la fase de construcción de la red; estos criterios son always, never if it performs better.

En base a estos aspectos, se construyeron las siguientes redes bayesianas:

- **Lanco2:** red construida en base a la información contenida en la tabla **datos_completos_disc.** Esta misma tabla fue usada en la fase de inferencia. No hay inclusión de conocimiento experto en la construcción de la red.

- **Lanco3:** red construida en base a la información contenida en la tabla **t_entrenamiento.** La inferencia se aplica sobre los datos de la tabla **t_test.** Ambas tablas son subconjuntos excluyentes entre sí, de la tabla **datos_completos_disc** No hay inclusión de conocimiento experto en la construcción de la red.

- **Lanco4:** esta red fue construida en base a los datos de la tabla **t_entrenamiento**, junto con el criterio de selección de características para la multi net **"always"**, además de la edición de las relaciones de los nodos usando conocimiento experto. La inferencia se hizo sobre los datos contenidos en **t_test.**

- **Lanco5:** esta red fue construida en base a los datos de la tabla **datos_completos_disc**, junto con el criterio de selección de características para la multi net **"always"**, además de la edición de las relaciones de los nodos usando conocimiento experto. La inferencia se hizo sobre los datos contenidos en **datos_completos_disc.**

- **Lanco6:** red construida en base a la información contenida en la tabla **t_entrenamiento.** Se seleccionó la opción de feature selector en la construcción de esta **"if it perform better".** Posteriormente se incluye conocimiento experto agregando relaciones entre nodos. La inferencia se aplica sobre los datos de la tabla **t_test.** Ambas tablas son subconjuntos excluyentes entre sí, de la tabla **datos_completos_disc.**

- **Lanco7:** red construida en base a la información contenida en la tabla **datos_completos_disc.** Esta misma tabla fue usada en la fase de inferencia. Se seleccionó la opción de feature selector en la construcción de esta **"if it perform better".** Posteriormente se incluye conocimiento experto agregando relaciones entre nodos.

La siguiente tabla resumen muestra los aspectos generales de configuración de dichas redes:

Redes: Lanco	**2**	**3**	**4**	**5**	**6**	**7**
Datos para construcción	Datos_ Completos _disc	T_entre Na Miento	T_entre Na Miento	Datos_ completos _disc	T_entre na miento	Datos_ completos _disc
Datos para inferencia	Datos_ Completos _disc	t_test	t_test	Datos_ completos _disc	T_test	Datos_ completos _disc
Inclusión de conocimiento experto	NO	NO	SI	SI	SI	SI
Feature selector: • **”always”** • **“if it perform better”.**	X	X	X	X	X	X

5.3. Descripción de los datos.

Los datos contenidos en la tabla **Datos_completos** representan píxeles que contienen información sobre la vulnerabilidad del índice de uso del suelo. La vulnerabilidad queda expresada en términos de un índice de vulnerabilidad y un valor de cluster asociado. Junto con esto existe información proporcionada por las 5 bandas satelitales C1, C2, C3, C4, C5, más un valor de pendiente del terreno y otro de altitud.

Existen 5 niveles de vulnerabilidad, por lo tanto son 5 intervalos dentro de los cuales cae un valor de índice.

La cantidad de registros asociados a cada cluster queda expresada en la siguiente tabla:

Cluster	**Cantidad**
1	1
2	5
3	11
4	95
5	4

Además de los datos anteriores, se cuenta con un atributo identificador del píxel Id_pixel.

En resumen, los atributos de la tabla **Datos_completos** son:

- id: identificador de registro que representa al conglomerado y la parcela de un punto dado. Ejemplo: idd=112, conglomerado=11, parcela=2.
- C1: banda 1 Landsat 7 ETM+
- C2: banda 2 Landsat 7 ETM+
- C3: banda 3 Landsat 7 ETM+
- C4: banda 4 Landsat 7 ETM+
- C5: banda 5 Landsat 7 ETM+
- m: pendiente del terreno en grados asociado a cada píxel
- expo: exposición en grados por píxel.
- Altitud: altitud sobre el nivel del mar en metros.
- Cluster : cluster de grado de vulnerabilidad.
- Indice: valor del índice de vulnerabilidad.
- Tabla: indica la procedencia del registro. Los valores que puede tomar son “e” (entrenamiento) y “t”(test).

5.4. Preparación de los datos.

En primer lugar, los datos deben ser preparados para tener un formato adecuado para la construcción de la red y la inferencia hecha por ésta. El modulo **BN data preprocessor** permite preparar los datos. La entrada para este módulo son los datos contenidos en la tabla **Datos_completos.**

Se debe indicar cuantas clases se desea que la red bayesiana pueda distinguir. En este caso, se trata de 5 clases que representan los niveles de vulnerabilidad. El software discretiza los datos, asignándoles un valor de intervalo para cada variable, en 5 grupos para cada una. Además se deben indicar cuales son las variables que deben ser discretizadas (las variables que representan los nodos de la red, como la variable que representa los valores de clasificación). De modo que el software genera 9 nuevas variables:

Indice_d : variable que representa las clases a clasificar o el número de redes, basándose en él las redes multi-net bayes.

- C1_d
- C2_d
- C3_d
- C4_d
- C5_d
- M_d
- Expo_d
- Alt_d

Estas nuevas variables son agregadas como atributos a una nueva tabla. Esta tabla será **Datos_completos_disc.**

El criterio seleccionado para definir los límites inferior y superior de cada clase fue **"equal width",** En base a los valores máximos y mínimos de la variable **indice_d**. Esto implica también que se generará una red especifica para cada uno de estos intervalos.

La definición de los intervalos se encuentra además determinada por la población utilizada para construir las redes. Como se ha indicado anteriormente, existen dos grupos: las redes construidas en base a la información contenida en **datos_completos_disc** (**Lanco2, Lanco5, Lanco7**), y las construidas en base a **t_entrenamiento (Lanco3, Lanco4, Lanco6).** De este modo, la siguiente tabla muestra los valores de dichos intervalos:

Cluster	**Intervalo (probabilidad) para Indice_d Redes: Lanco3, Lanco4, Lanco6**	**Intervalo (probabilidad) para Indice_d Redes: Lanco2, Lanco5, Lanco7**
1	>0.63857	>1.616528
2	>0.00201<0.63857	>0.9685761<1.616528
3	>-0.207785<0.00201	>0.3206241<0.9685761
4	>-0.464305<-0.207785	>-0.327328<0.3206241
5	<-0.464305	<-0.327328

Esta tabla es la que da origen a las tablas **t_entrenamiento** y **t_test**. La siguiente tabla muestra el numero de registros para cada clase por tabla:

Cluster	**Datos_completos_disc (Totales)**	**T_entrenamiento**	**T_test**
1	1	1	1
2	5	4	1
3	11	7	4
4	95	59	36
5	4	3	1

5.5. Creación de las redes e inferencia.

A continuación se describirá de manera individual cada uno de los experimentos y se expondrán sus resultados. La descripción incluye las arquitecturas individuales de cada uno de los experimentos, asociados a cada una de las clases o intervalos de índice de vulnerabilidad. Junto con esto, se expondrá la matriz de confusión resultante para cada caso. Los resultados serán discutidos en la sección "resultados finales y conclusiones".

5.5.1. Lanco2

Como se dijo anteriormente, esta red está construida en base a la información contenida en la tabla **datos_completos_disc.** Esta misma tabla fue usada en la fase de inferencia. No hay inclusión de conocimiento experto en la construcción de la red.

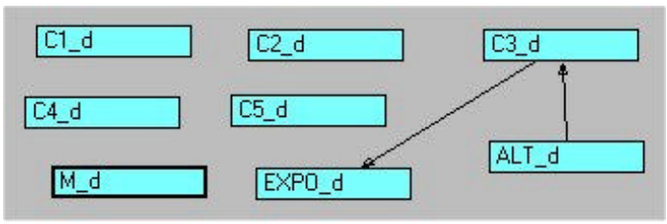
Para la construcción de la red se utilizó el módulo **Belief network Power predictor**, utilizando la opción que permite aprender una nueva red clasificadora de creencia desde los datos, y posteriormente utilizarla para clasificar datos. Se le indicó al software sobre la construcción de una multinet, en la sección **algorithm seetings (Learn a bayesian multinet classifier),** junto con opción de selección de características para la multinet **if it perform better.** El software genera un archivo que representa la arquitectura de la red, indicándose las subredes,

arquitecturas individuales, las conexiones, etc. Este archivo es clasificador_Lanco_2.bnc. y puede ser cargado nuevamente cuando se desee clasificar nuevos datos con dichas subredes.

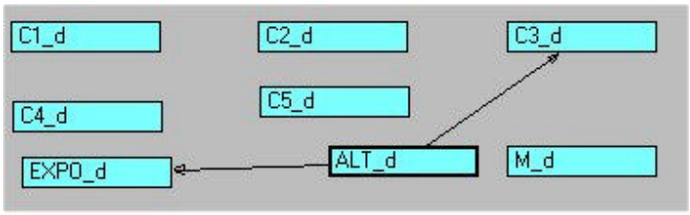
Antes de utilizar la red, existe la posibilidad de editarla, alterando su estructura, lo que implica suministrar conocimiento experto durante su creación.

Las arquitecturas de red, para cada una de las categorías quedan ilustradas en la siguiente figura:

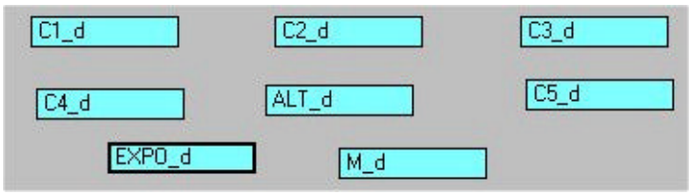
Indice_d <-0.327328 (cluster 5)



Indice_d>-0.327328<0.3206241 (cluster4), Indice_d >0.3206241<0.9685761 (cluster3),Indice_d >0.9685761<1.616528 (cluster2).



Indice_d >1.616528 (cluster1).



Los resultados de la clasificación son expuestos a continuación. La matriz de confusión siguiente muestra dichos resultados.

	1	2	3	4	5	total	% precisión en la clasificación
1	0	0	0	0	0	**0**	0
2	0	0	0	0	0	**0**	0
3	0	1	2	5	0	**8**	25
4	1	4	6	72	1	**84**	85.71
5	0	0	3	18	3	**24**	12.5
Total	**1**	**5**	**11**	**95**	**4**	**77/116**	
%precisión en la clase	0	0	18.182	75.789	75		**66.37931%**

Como los resultados de la red son conocidos, y la pertenencia de cada registro a su clase real, se puede construir una matriz de confusión contrastando las pertenencias reales (conocidas) de estos con las pertenencias asignadas durante la inferencia de las redes a los registros.

La posición $[i, j]$ representa el número de registros de la clase j, clasificados por la red como i.

Como era de esperar, dada la presencia de los datos en cada una de las clases, la clase mejor clasificada fue la 4, seguida de la clase 5. Las clases 2 y 1 simplemente no fueron reconocidas durante la inferencia. Se puede ver que la calidad general de clasificación fue de un 66.4% lo que no es malo, dada la poca información con la que se contó y la mala distribución de la presencia de registros en cada clase.

5.5.2. Lanco3

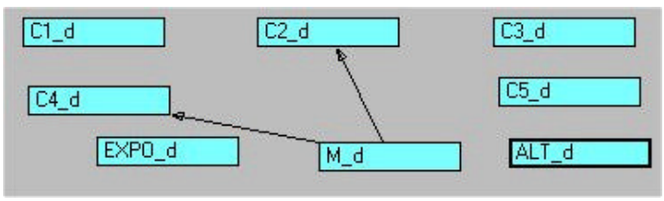
Red construida en base a la información contenida en la tabla **t_entrenamiento.** La inferencia se aplica sobre los datos de la tabla **t_test.** Ambas tablas son subconjuntos excluyentes entre sí, de la tabla **datos_completos_disc** No hay inclusión de conocimiento experto en la construcción de la red.

La matriz de confusión siguiente muestra las asignaciones de los registros hechos por la multinet a cada una de las clases.

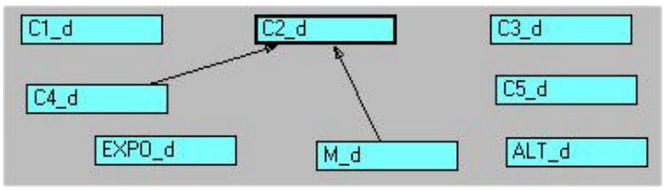
	1	2	3	4	5	total	% precisión en la clasificación
1	0	0	1	6	0	**7**	0
2	0	0	0	0	1	**1**	0
3	0	0	0	14	0	**14**	0
4	0	1	1	12	0	**14**	85.7
5	1	0	2	4	0	**7**	57.14
Total	**1**	**1**	**4**	**36**	**1**	**12/43**	
%precisión en la clase	0	0	0	33.3	0		**27.9%**

El siguiente diagrama muestra las arquitecturas de la red Lanco3:

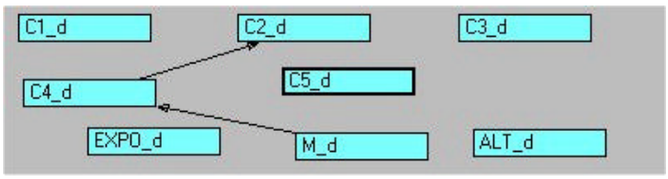
Indice_d <-0.464305 (cluster 5), Indice_d >-0.207785<0.00201 (cluster 3), Indice_d >0.63857 (cluster1):



Indice_d >-0.464305<-0.207785 (cluster4)



Indice_d >0.00201<0.63857 (cluster 2):



5.5.3. Lanco4

Esta red fue construida en base a los datos de la tabla **t_entrenamiento**, junto con el criterio de selección de características para la multi net **"always"**, además

de la edición de las relaciones de los nodos usando conocimiento experto. La inferencia se hizo sobre los datos contenidos en **t_test.**

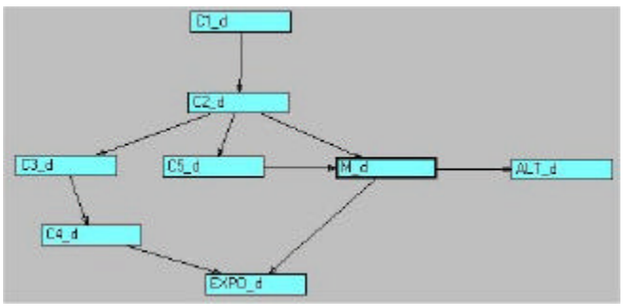
La matriz de confusión siguiente muestra el comportamiento de esta red durante la fase de clasificación:

	1	**2**	**3**	**4**	**5**	**total**	**% precisión en la clasificación**
1	0	0	0	6	0	**6**	0
2	0	1	1	11	0	**13**	7.7
3	1	0	1	11	0	**13**	7.7
4	0	0	0	3	1	**4**	75
5	0	0	2	5	0	**7**	0
Total	**1**	**1**	**4**	**36**	**1**	**5/43**	
%precisión en la clase	0	100	25	8.3	0		**11.62%**

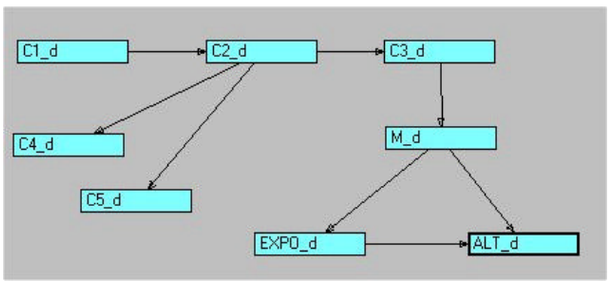
La inclusión de conocimiento experto logró aumentar la calidad de clasificación en las clases donde se cuenta con muy pocos ejemplares, lo que se refleja en el aumento de la calidad de precisión en cada clase, particularmente en las clases 2 y 3. Sin embargo, la clase 4 fue muy mal reconocida por las redes. Como esta clase es la con mayor presencia dentro de la población, esto influyó en la mala calidad general de la red (un 11.62%), considerándose ésta como muy poco confiable.

El siguiente diagrama muestra la arquitectura de la red Lanco4:

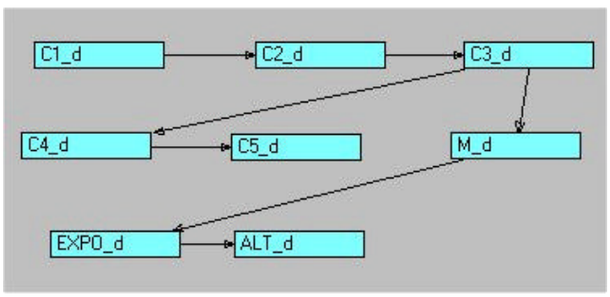
Indice_d <-0.464305 (cluster 5):



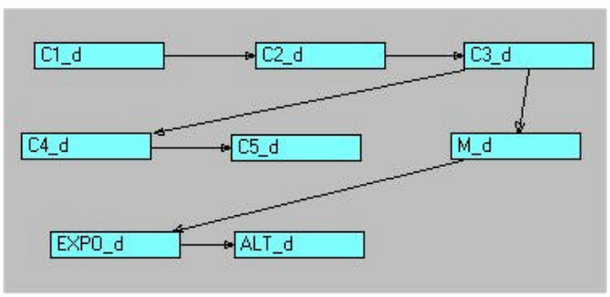
Indice_d >-0.464305<-0.207785 (cluster4)



Indice_d >-0.207785<0.00201 (cluster 3)



Indice_d >0.00201<0.63857 (cluster 2), Indice_d >0.63857 (cluster1)



5.5.4. Lanco5

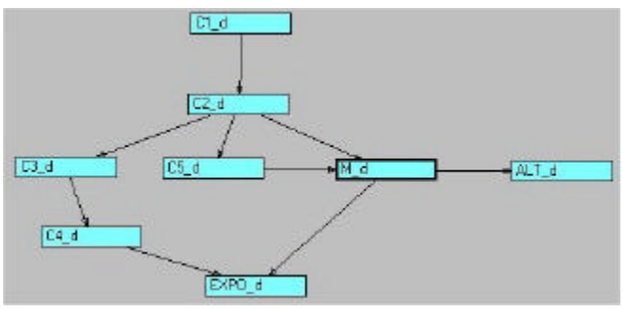
Esta red fue construida en base a los datos de la tabla **datos_completos_disc**, junto con el criterio de selección de características para la multi net **"always"**, además de la edición de las relaciones de los nodos usando conocimiento experto. La inferencia se hizo sobre los datos contenidos en **datos_completos_disc.**

La matriz de confusión arrojó los siguientes resultados:

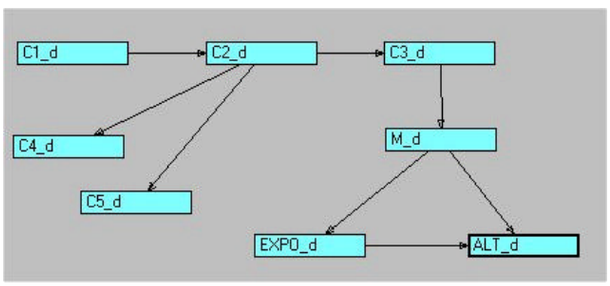
	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	1	0	**1**	0
3	0	0	**1**	5	1	**7**	14.28%
4	1	5	10	**69**	2	**87**	79.31%
5	0	0	0	20	**1**	**21**	4.76
Total	**1**	**5**	**11**	**95**	**4**	**71/116**	
%precisión en la clase	0	0	9.10%	72.63%	25%		**61%**

El siguiente diagrama muestra la arquitectura de la red Lanco5:

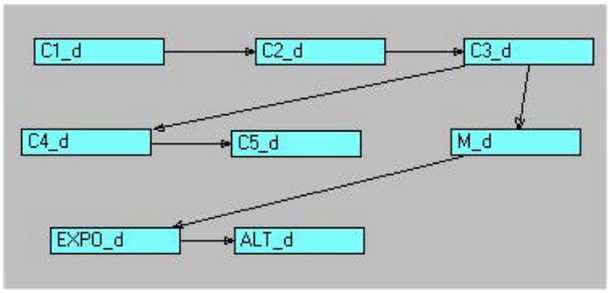
Indice_d <-0.327328 (cluster 5):



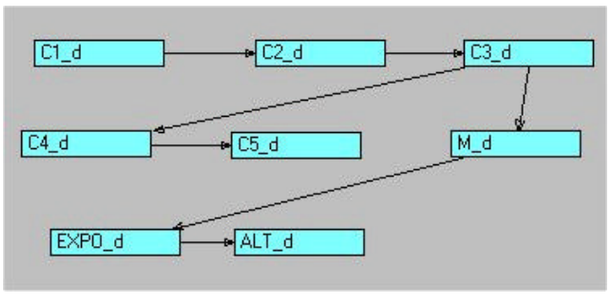
Indice_d >-0.327328<0.3206241 (cluster4)



Indice_d >0.3206241<0.9685761 (cluster 3)



Indice_d >0.9685761<1.616528 (cluster 2), Indice_d >1.616528 (cluster1)



5.5.5. Lanco6

Red construida en base a la información contenida en la tabla **t_entrenamiento.** Se seleccionó la opción de feature selector en la construcción de esta **"if it perform better".** Posteriormente se incluye conocimiento experto agregando relaciones entre nodos. La inferencia se aplica sobre los datos de la tabla **t_test.** Ambas tablas son subconjuntos excluyentes entre sí, de la tabla **datos_completos_disc.**

Arquitectónicamente, esta red presenta la misma estructura, definición de intervalos y poblaciones de datos para construcción e inferencia que el experimento **LANCO6.** La diferencia principal es la elección del parámetro**"if it perform better"** (si esta se desempeña mejor).

La matriz de confusión para esta red es como se muestra a continuación:

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	6	0	**6**	0
2	0	**1**	1	11	0	**13**	7.7%
3	1	0	**0**	11	0	**12**	0
4	0	0	1	**3**	1	**5**	60%
5	0	0	2	5	**0**	**7**	0
Total	**1**	**1**	**4**	**36**	**1**	**4/43**	
%precisión en la clase	0	100%	0	8.3%	0		**9.3%**

5.5.6. Lanco7

Red construida en base a la información contenida en la tabla **datos_completos_disc.** Esta misma tabla fue usada en la fase de inferencia. Se seleccionó la opción de feature selector en la construcción de esta **"if it perform better".** Posteriormente se incluye conocimiento experto agregando relaciones entre nodos.

Arquitectónicamente, presenta las mismas estructuras de las redes con conocimiento experto.

La matriz de confusión siguiente muestra el estado de clasificación de este modelo:

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	0	0	**0**	0
3	0	0	**1**	5	0	**6**	16.7%
4	1	5	10	**63**	2	**81**	77.8%
5	0	0	0	27	**2**	**29**	6.9%
Total	**1**	**5**	**11**	**95**	**4**	**66/116**	
%precisión en la clase	0	0	9.1%	66.32%	50%		**58.9%**

6. Capítulo 7: Fundamentos sobre Redes Neuronales.

6.1. Redes Neuronales.

6.1.1. Conceptos básicos.

Existen numerosas definiciones para caracterizar a una red neuronal, e intentar explicar en que consiste. Kohonen la define como "redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, los cuales intentan interactuar con los objetos del mundo real, del mismo modo que lo hace el sistema nervioso biológico" [Hil,2000].

Por lo tanto, podríamos decir que Las Redes neuronales imitan la estructura y funcionamiento del cerebro humano usando modelos matemáticos. Estas se basan en tres principios básicos del funcionamiento del cerebro humano:

1. El conocimiento es distribuido a través de muchas neuronas.
2. Las neuronas se pueden comunicar de manera local unas con otras.
3. El cerebro es adaptable.

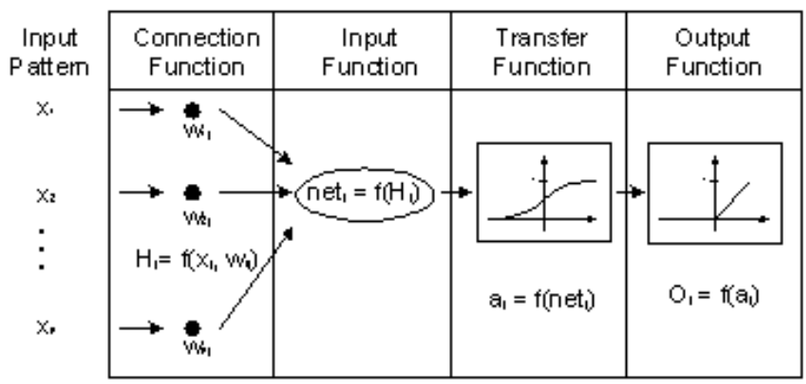
Estos tres aspectos nos ayudaran a encontrar una caracterización de una red neuronal. Podemos decir que una red neuronal artificial está compuesta de varias neuronas con una **estructura neuronal** característica, interconectadas en base a una **topología de red**, que utilizan una **regla de adaptación** de las conexiones entre neuronas o regla de aprendizaje.

6.1.1.1. **Estructura neuronal**.

Las Neuronas (unidades de procesamiento), con las cuales se construyen las redes neuronales, son elementos individuales que se componen de cuatro elementos:

1. Una función de conexión.
2. Una función de entrada (Input función).
3. Función de transferencia.
4. Una función de salida (Output Función.)

El procesamiento de información neuronal puede ser representado por la siguiente figura[Dat,2002]:

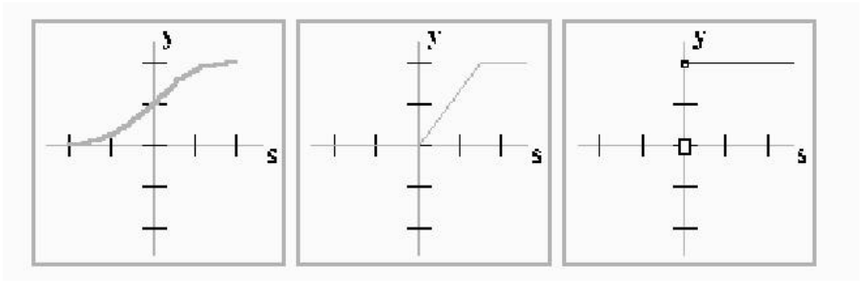


Una neurona recibe señales por medio de distintas conexiones de entrada (input connections). Estas conexiones son ponderadas a la entrada de la neurona mediante la función de conexión (connection function). Los pesos empleados aquí, son los que definen las sinapsis, o conexiones entre neuronas, y son establecidos finalmente por el proceso de aprendizaje de la red. Estos irán cambiando constantemente a lo largo del proceso de aprendizaje de acuerdo a cada patrón o vector de entrada que sea ingresado a ésta y la regla de aprendizaje utilizada.

La función de entrada es la encargada de transformar este valor en un valor escalar. Generalmente consiste en una sumatoria de términos provenientes de la función de conexión. En tales casos, la actividad de la red que proviene de la función de conexión y la función de entrada es la suma ponderada de los valores de entrada. La función de activación determina un nuevo estado basado en la actividad actual de la red, si es que es apropiado tomar el estado previo de la neurona en cuenta. Este nuevo estado de activación es transmitido por medio de las conexiones neuronales vía una función de salida, la cual es generalmente una función lineal. Haciendo una referencia a los sistemas biológicos neuronales, el estado de activación en la salida de una neurona es conocido como excitación de la neurona.

Existen varios tipos de funciones de activación, dependiendo de la regla de aprendizaje.

La siguiente figura muestra funciones de activación.

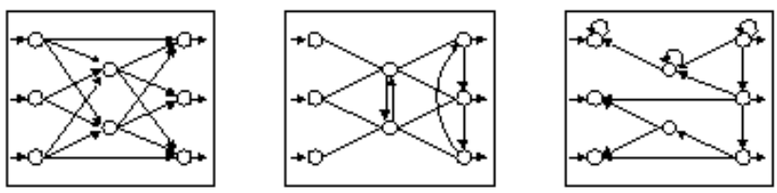


6.1.1.2. Topología de Red.

En una red neuronal, las neuronas son interconectadas, para formar una estructura neuronal rígida, con resultado de lo cual el algoritmo de aprendizaje raramente provoca las condiciones para la formación de nuevas conexiones y remoción de las viejas conexiones, tal como ocurre en los sistemas biológicos[Dat,2002].

Es empleada una estructura de conexión en capas. Existe una capa de entrada, capas escondidas y una capa de salida. Las neuronas se conectan generalmente entre capas. El número de capas encontradas en una red neuronal determina el desempeño de la red, junto con otros factores que serán explicados posteriormente.

Como muestra la siguiente figura, la topología de redes neuronales se pueden clasificar en tres grandes grupos: feed forward (redes conectadas hacia delante), feed Back (redes conectadas hacia atrás), y con conexiones laterales. En los dos primeros casos, la red neuronal puede tener un número variable de capas escondidas [Dat,2002].



6.1.1.3. Regla de adaptación

La regla de adaptación es el algoritmo encargado de definir los criterios mediante los cuales se establece la dinámica de los pesos o ponderaciones que conectan unas neuronas con otras. La red neuronal debe pasar por una serie de fases antes de encontrarse en condiciones para realizar la tarea para la cual fue diseñada. Estas fases son **entrenamiento** de la red, **testeo**, y su uso operativo. En Data Engine esta última fase es conocida como **recall.** Durante el entrenamiento de la red, son presentados ejemplos que perfilaran las capacidades de clasificación de tipos que pueda desempeñar.

En realidad, el conocimiento no es almacenado físicamente en la red, sino que es representado a través de la distribución de los pesos de las neuronas. Tal cual como sucede en los sistemas biológicos, el aprendizaje ocurre a través de una adaptación estructural de los pesos, donde podría ocurrir que algunas conexiones desaparezcan y otras nuevas aparezcan.

Inicialmente, durante el entrenamiento, se asignan valores al azar para las conexiones entre las neuronas. Entonces la red procesa vectores de entrada (los ejemplos de entrenamiento), y, tomando como referencia una salida deseada para cada vector de entrada, son ajustados los pesos usando un criterio impuesto por la regla o algoritmo de aprendizaje. Este proceso se repite hasta que se cumple con un criterio de convergencia donde se establece un umbral de error que debe ser alcanzado.

La fase de Test o prueba consiste en presentarle a la red ejemplos que nunca antes han sido expuestos a ésta, donde se mide la calidad del aprendizaje resultante de la fase de entrenamiento. De este modo se puede determinar hasta qué punto la red tiene aprendida la tarea para la cual fue diseñada. Para tener una buena referencia de la calidad de aprendizaje, la selección de ejemplos durante esta fase debe contemplar en lo posible todos los casos de patrones considerados en el dominio del problema.

La fase de recall es simplemente el uso de la red en la tarea para la cual fue diseñada. Se presentan entradas que deben ser clasificadas, de las cuales desconocemos la salida.

6.1.1.4. Tipos de aprendizaje.

Como se ha dicho anteriormente, el aprendizaje de una red neuronal es el proceso por el cual esta modifica sus pesos en respuesta a una información de entrada. Así, puede ser clasificado en dos grandes grupos:

1. Aprendizaje supervisado.
2. Aprendizaje no supervisado.

6.1.1.4.1. **Aprendizaje supervisado:** Se caracteriza por que el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor comprueba la salida de la red y en el caso de que ésta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada.

6.1.1.4.2. **Aprendizaje no supervisado:** Las redes que utilizan este tipo de aprendizaje no requieren influencia externa para ajustar los pesos de las conexiones entre las neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas redes son capaces de auto organizarse.

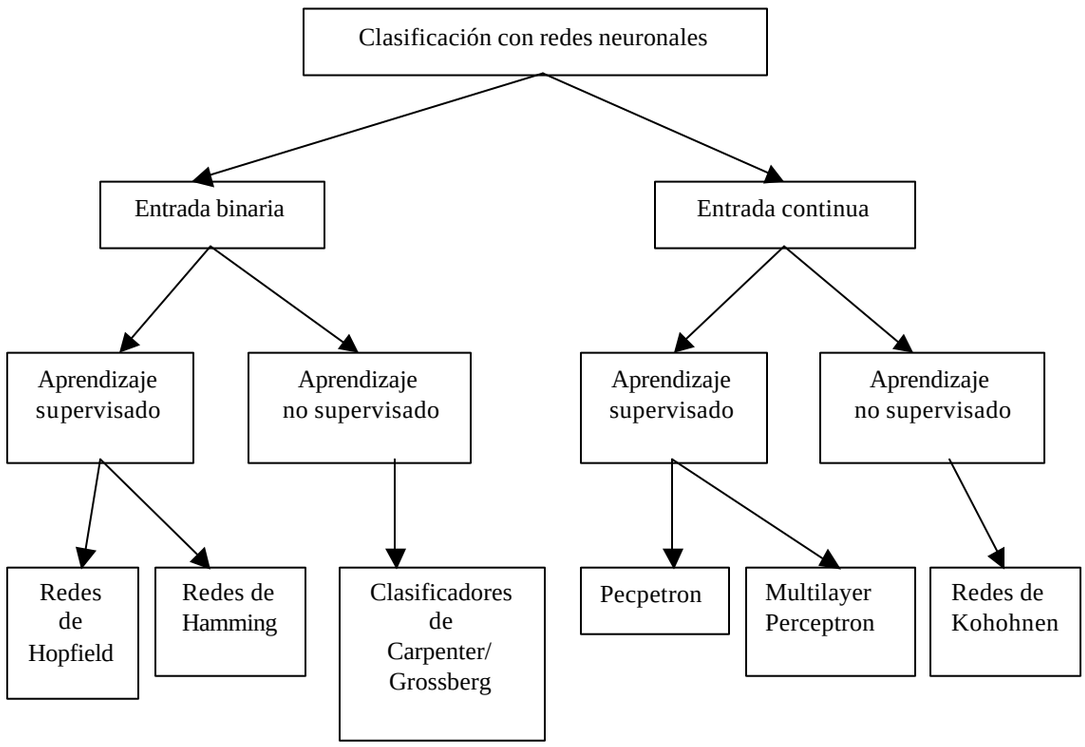
Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se pueden establecer entre los datos que se presentan en su entrada. Puesto que no

hay un supervisor que indique a la red la respuesta que debe generar ante una entrada concreta, cabría preguntarse precisamente por lo que la red genera en estos casos. Existen varias posibilidades en cuanto a la interpretación de las salidas de las redes, que dependen de su estructura y del algoritmo de aprendizaje empleado.

En algunos casos, la salida representa el grado de familiaridad o similitud entre la información que se le esta presentando a la entrada y las informaciones que se le han mostrado hasta entonces (en el pasado). En otro caso, podría realizar una clusterización (clustering) o establecimiento de categorías, indicando la red a la salida qué categoría pertenece la información presentada a la entrada, siendo la propia red quien deba encontrar las categorías apropiadas a partir de correlaciones entre las informaciones presentadas. Una variación de esta categorización es el prototipado. En este caso, la red obtiene ejemplares o prototipos representantes de las clases a las que pertenecen las informaciones de entrada.

6.1.2. Clasificación de los tipos de redes en tareas de clasificación

Finalmente, podemos hacer una clasificación de los tipos de redes utilizados en tareas de clasificación, dependiendo del tipo de entrada (binaria o continua), considerando el tipo de aprendizaje. La siguiente figura ilustra los tipos de redes implementados en Data Engine[Dat,2002]:



6.2. Multilayer Perceptrón.

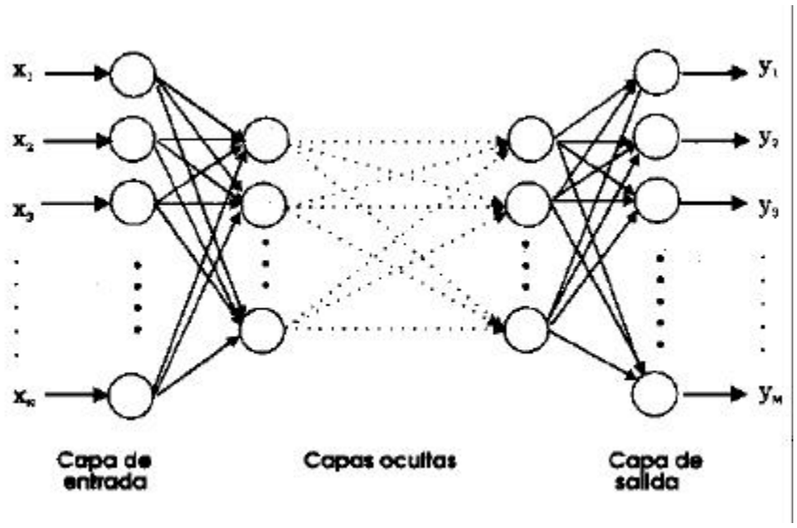
6.2.1. Introducción.

En 1986, Rumelhart, Hinton y Willlians [Rum,1986], basándose en trabajos de otros investigadores [Wer,1974], [Par,1982], formalizaron un método para que una red neuronal aprendiera la asociación que existe entre los patrones de entrada de la misma y las clases correspondientes, utilizando más niveles de neuronas que las que utilizó Rosenblantt para desarrollar el perceptrón. Este método, conocido en general como backpropagation (propagación del error hacia atrás), está basado en la generalización de la regla delta y, a pesar de sus propias limitaciones, ha ampliado de forma considerable el rango de las aplicaciones de las redes neuronales.

Este algoritmo puede ser aplicado en modelos de redes neuronales que poseen capas intermedias. Una característica importante de este algoritmo es su capacidad para representar conocimiento que se logra organizando la capa intermedia de la red, de manera de lograr cualquier correspondencia entre la entrada y la salida de la red.

Lo anterior es una característica muy importante; la red multilayer Perceptrón es una combinación de redes Perceptrón simples. Esta característica permite resolver problemas de funciones que eran imposibles de resolver como el problema de la función XOR. [Hil,2000].

Como muestra la siguiente figura, es un modelo de red en el cual las neuronas son configuradas en capas, donde generalmente, las neuronas de una capa se encuentran todas conectadas con las neuronas de la capa siguiente. Como el sentido de las conexiones existe solamente desde la capa de entrada hacia la capa de salida, esta es una red feedforward. (Alimentada hacia delante).



Esta red puede procesar patrones de entrada análogos y aprende en modo supervisado, empleando el algoritmo de backpropagación (propagación hacia atrás), motivo por el cual esta red es referida frecuentemente como la red de backpropagation. Las neuronas son expandidas por un factor de umbral, y la función sinusoidal es empleada como una función de activación, del mismo modo como el algoritmo de aprendizaje requiere de la función de activación para ser constante y diferenciable.

Como alternativa, existen diversos algoritmos de aprendizaje. Entre ellos se encuentran[Dat,2002]:

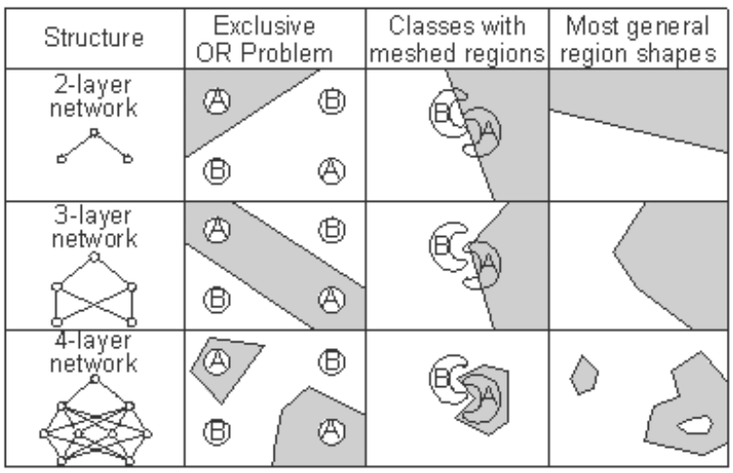
- quickpropagation (propagación rápida).
- Super SAP (modificación del algoritmo backpropagatión).
- Resilent propagation. (Propagación elástica).

Cada uno de estos algoritmos serán comentados más adelante.

6.2.2. Relación entre el desempeño de una red y el número de capas.

La capacidad de desempeño de la red esta fuertemente condicionada por el número de capas escondidas, así este factor determina las capacidades de representación de la red. [Dat,2002] La figura siguiente muestra la capacidad de partición para diferentes números de capas.

Como muestra la figura:



1. **Red de dos capas:** esta red puede efectuar solamente una separación lineal del espacio de decisión. Esto corresponde a una función de separación lineal.

2. **Red de tres capas:** estas redes combinan el resultado de distintas redes de dos capas y pueden delimitar el espacio objetivo con una línea continua. El número de aristas se corresponde con el número de neuronas escondidas. Esta región es convexa en capa, sin embargo. Así una expansión de la capa intermedia habilita los limites de separación con mayor precisión pero las restricciones de configuraciones convexas no mejora.

3. **Red de Cuatro capas:** desempeñan cualquier tipo requerido de separación requerido, incluido configuraciones no convexas regiones inconexas y inclusiones.

Debido a su fácil comprensión algorítmica y su eficiencia, el multilayer Perceptrón es desarrollado en muchos campos de aplicaciones prácticas. Debe destacarse aquí, sin embargo, que el que el proceso de entrenamiento requiere altos niveles de capacidades de computo, y podría requerir por lo tanto un gran consumo de tiempo en PCs convencionales.

También, la convergencia del algoritmo no está siempre garantizada. Consecuentemente, muchas variantes del algoritmo han sido desarrolladas. En la práctica, se han visto muchas aplicaciones que muestran buenos resultados sin tales modificaciones al algoritmo original.

6.2.3. Aplicaciones de la red Multilayer perceptrón.

Debido a sus características arquitectónicas y su capacidad de representar conocimiento, esta red ha sido utilizada en diversas áreas. A continuación se describirá de modo general algunas aplicaciones desarrolladas con MLP[Hil,2000].

Codificación de la Información: La idea principal consiste en que la información de entrada se recupere en la salida a través de un código interno. Si esto se consigue, en la fase de funcionamiento sólo habrá que proporcionar a la red la información codificada, compuesta por los estados de activación de las neuronas ocultas y los pesos correspondientes obtenidos durante la etapa de aprendizaje para generar correctamente la salida deseada.

Traducción de texto en lenguaje hablado: En la capa de entrada se codifica convenientemente cada una de las posibles letras, así como el contexto local, que consiste en las tres letras precedentes y posteriores, a través de una ventana que se va moviendo por el texto secuencialmente. En la capa de salida se codifica la información correspondiente a la pronunciación de una serie de fonemas, más ciertas características que dan idea de sus vecinos. Con un número adecuado de unidades ocultas, la red establece la correspondencia texto- fonemas que permitirá, durante la fase de funcionamiento, generar automáticamente los fonemas correspondientes a un determinado texto.

Reconocimiento del lenguaje hablado: Se trata de una aplicación difícil de atacar, por lo que su estudio es de máxima actualidad. La dificultad reside en saber tratar correctamente información temporal. Una de las alternativas que se han propuesto es introducir retardos que muestren explícitamente la existencia de una ventana temporal en la captura de información. Cada unidad oculta se encuentra conectada a cada unidad de la capa de entrada a través de diferentes conexiones, que dan cada una de ellas diferentes retardos. De esta forma, se pretende generar respuestas similares a patrones de entrada similares, pero que están desplazados en el tiempo. Para tratar correctamente este problema, el algoritmo backpropagation convencional ha de ser modificado correctamente. El error no sólo hay que propagarlo por todas las capas, sino temporalmente, lo cual introduce un mayor nivel de complejidad en al resolución del problema.

Reconocimiento óptico de caracteres: Se trata de una aplicación típica de reconocimiento de patrones, siendo en este caso los caracteres de un texto los patrones que hay que reconocer. Se puede utilizar una red MLP para aprender las diferentes formas de los caracteres que se representarán con imágenes compuestas por cierto

número de pixeles, generando la red como salida el código, por ejemplo ASCII, correspondiente al carácter de entrada. El objetivo es que la red tenga capacidad para generalizar; es decir, que reconozca formas de caracteres no utilizadas durante la fase de aprendizaje, siempre que no difieran excesivamente de los patrones de entrenamiento. también se puede utilizar una segunda red

MLP como memoria auto asociativa, o como diccionario que aprenda palabras completas y que permita depurar posibles errores en el reconocimiento de caracteres aislados llevado a cabo por la primera red. Se han obtenido buenos resultados con caracteres impresos de diferentes tamaños, aunque es motivo de investigación actual su aplicación en el caso de texto manuscrito.

6.2.4. Estructura de una red multilayer perceptrón.

6.2.4.1. Neuronas del multilayer Perceptrón.

La neurona de este tipo de red puede ser descrita vía su función de entrada, su función de activación y su función de salida. En Data Engine la suma de los pesos de todas las señales que son activas en las conexiones de entrada es empleada como la **función de entrada.**

$$I_i = \sum_j w_{ij} x_j$$

Donde:

- w_{ij} El peso de la conexión entre la neurona i(neurona destino) y la neurona j (neurona origen).
- x_j Señal de la conexión referida.

La suma es transformada por la función de activación. Todos los algoritmos de aprendizaje implementados en Data Engine requieren de una función de activación para ser diferenciable. En principio, por lo tanto, cualquier función diferenciable puede ser empleada. En Data Engine, las siguientes alternativas se encuentran disponibles[Dat,2002]:

	Función de activación	Derivada
Linear	$f(z) = z$	1
Sigmoid	$f(z) = \frac{1}{1+e^{-t}}$	$\dot{f}(z) = f(z)(1-f(z))$
Sym. Sigm.	$f(z) = 2\left(\frac{1}{1+e^{-t}} - 0.5\right)$	$\dot{f}(z) = 0{,}5(1-f(z)f(z))$
Mod. Sigm	$f(z) = \frac{1}{1+e^{-t}}$	$\dot{f}(z) = f(z)(1-f(z)) + 1$
Tahn	$f(z) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}}$	$\dot{f}(z) = 1 - f(z)^2$
Parábola	$f(z) = \begin{cases} -1, & z < 2 \\ \frac{1}{4}(z+2)^2 - 1, & -2 \le z \le 0 \\ 1 - \frac{1}{4}(z-2)^2, & 0 \le z \le 2 \\ \frac{1}{z} > 2 \end{cases}$	$\dot{f}(z) = \begin{cases} \sqrt{(1+f(z))}, z \le 0 \\ \sqrt{(1-f(z))}, z \ge 0 \end{cases}$

Como se puede notar, la derivada de cada una de las funciones puede ser expresada usado la misma función.

En Data Engine, una transferencia directa de la activación de una neurona a su salida es empleada como la función de salida. La **función de transferencia global de una neurona** es estructurada como:

$$o_i = a_i = f(\sum_j w_{ij} x_j)$$

- O_i Salida de la neurona i.
- a_i Activación de la neurona i.
- x_j Es idéntica a la salida de la neurona precedente, con el índice j del elemento observado.

6.2.5. El algoritmo Backpropagation

La regla de aprendizaje Backpropagation es una generalización de la regla delta (Kratzer, 1990), este habilita adicionalmente los pesos de las conexiones de las neuronas de las capas escondidas para que aprendan. El proceso de aprendizaje de un multilayer perceptrón con backpropagation es detalladamente como se describe a continuación[Dat,2002],[Hil,2000]:

1. Un vector de entrada es presentado a la red, es decir, estos elementos son aplicados a las neuronas de la capa de entrada.

2. Esta información es propagada a la red en dirección hacia delante, y de este modo, generando un vector de salida O.

3. La comparación de este vector de salida con el valor de salida objetivo provee un error para cada neurona de salida. La meta del proceso de aprendizaje es minimizar el error global de la red.

$$E = \frac{1}{2}\sum (Z_i - O_i)^2$$

Donde:

- z_i Valores de salida objetivos para cada neurona i.
- O_i Salida de la neurona i.

La regla delta logra esto alterando los pesos de las conexiones en la dirección del máximo gradiente de la función de error. Una adaptación de los pesos es efectuada de acuerdo a la ecuación:

$$\Delta w_{ij} = w_{ij}(t+1) - w_{ij}(t) = -\alpha \frac{\partial E}{\partial w_{ij}}$$

Donde α es definido como la **tasa de aprendizaje**. Esto resulta en:

$$\Delta w_{ij} = \alpha \cdot \delta_j \cdot x_j$$

4. El error local es calculado dependiendo del tipo de neurona. Si la neurona se encuentra en una capa escondida, su error se calcula como:

$$\delta_i = f'(I_i) \sum_j \delta_j w_{ij}$$

Donde:

δ_i : error local de la neurona i.

δ_j : error del elemento j de la capa siguiente conectado con la neurona i.

w_{ij} : peso de la conexión entre la neurona i y neurona j.

I_i : función de entrada de la neurona i.

Lo anterior puede ser interpretado como que para calcular el error de una neurona en una

Capa k, el error de todas las neuronas en la capa k+1 (la próxima capa en la dirección de la capa de salida) es requerida.

El error local de una neurona de la capa de salida es obtenido vía:

$$d_i = f(\dot{I}_i)(Z_i - O_i)$$

Donde:

- I_i : función de entrada de la neurona i.
- z_i Valores de salida objetivos para cada neurona i.
- O_i Salida de la neurona i.

Este error es, primero que todo, calculado y entonces propagado hacia atrás en la capa escondida antes de la capa de salida. Este proceso se continúa hasta que la capa de entrada es alcanzada.

5. En el instante que la capa de entrada es alcanzada, un error ha sido calculado para cada elemento. Los pesos de las conexiones pueden ser modificados de acuerdo a Δw_{ij} en la fase concluyente de este proceso.

En resumen, el algoritmo de backpropagation consiste en un aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo, empleando un ciclo de propagación-adaptación de dos fases:

1. Primero, se aplica un patrón de entrada como estimulo para las neuronas de la primera capa de la red.

2. Se va propagando a través de todas las capas superiores hasta generar una salida.

3. Se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener.

4. Se calcula un valor de error para cada neurona de salida.

5. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia que contribuya directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original.

6. Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada; es decir, el error disminuya.

6.2.6. Algunas consideraciones sobre la adaptación de los pesos.

6.2.6.1. Momentum.

Así como cada método de gradiente descendiente, la cuestión de la correcta configuración de la tasa de aprendizaje también se aplica al algoritmo de backpropagation. La función de error puede ser considerada como un espacio multidimensional en el cual un error es asignado a cada configuración de los pesos. Este espacio generalmente representa una dependencia no lineal extrema entre la configuración de los pesos y el error global. La meta del proceso de aprendizaje es encontrar el mínimo global en el, espacio de error. El hecho de que los pesos cambien con una función lineal de la derivada parcial corresponde a la asunción que el espacio de error es localmente lineal. El tamaño de un vecindario para ser considerado como local es descrito por la **tasa de aprendizaje** (α).

Sin embargo, una pequeña tasa de aprendizaje implica una gran capacidad de computo que será requerida y involucra un peligro de estancamiento en un mínimo local del espacio de error.

Este conflicto puede ser resuelto mediante la introducción de un término de **momentum**. La ecuación para adaptación de los pesos es modificada como:

$$\Delta w_{ij}(t) = \alpha \cdot \delta_j \cdot x_j + m \cdot \Delta w_{ij}(t-1)$$

m Es definido como el **momemtum**. Este valor está entre cero y uno. Tal modificación de la regla de aprendizaje toma en cuenta el cambio de los pesos desde los pasos de

aprendizaje previos (t-1) en el paso de aprendizaje actual (t), donde el término de la función de momentum funciona como un filtro pasa-bajos. Esto suprime el efecto oscilatorio y consolida la tendencia del plazo más largo en el proceso de cambios de pesos. El hecho es que una pequeña tasa de aprendizaje en sí misma contrarresta el alejamiento de las tendencias oscilatorias, lo que se agudiza por esta influencia. El resultado final es habilitar un aprendizaje más rápido a pesar de una tasa de aprendizaje baja.

6.2.6.2. Weight-Decay.

Las redes neuronales compuestas por pesos relativamente pequeños tienen la cualidad de poder generalizar de mejor modo. Para obtener estos pesos pequeños, un Weight-Decay (decaimiento de la ponderación o caída del peso) aplicado, provocaría dos efectos:

1. La prevención de grandes pesos que provocarían un comportamiento de aprendizaje desfavorable [Wer,1988], [Zel,1994] [Dat,2002]

2. La facilitación de poda (**pruning).** Cuando los pesos son artificialmente llevados a cero por el Weight-Decay, poseen una relevancia insignificante. De tal modo caen debajo del umbral de poda y son removidos de la red mediante poda (poda de pesos). [Wei,1992].

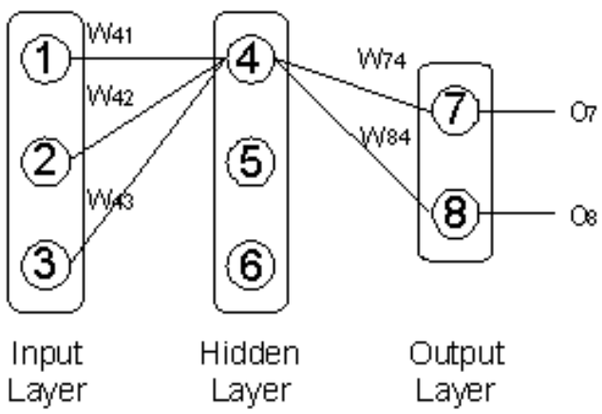
Esto se logra agregando a la ecuación de adaptación de los pesos, una constante de decaimiento de pesos, d, en ambas versiones (con y sin considerar un término de momentum). De este modo las ecuaciones se transforman en:

$$\Delta w_{ij}(f) = d \cdot (\boldsymbol{a} \cdot \boldsymbol{d}_j \cdot x_j)$$
$$\Delta w_{ij}(f) = d \cdot (\boldsymbol{a} \cdot \boldsymbol{d}_j \cdot x_j + \boldsymbol{m} \cdot \Delta w_{ij}(t-1))$$

valores típicos del factor Weight-Decay caen entre 1.0 - 10E-6 = 0.999999 y 1.0 - 10E-8 = 0.99999999.el caso donde esta constante esta más cercana a uno, es la que tiene un efecto menor.[Dat,2002]

6.2.7. Ejemplo calculando el cambio de peso de una neurona escondida.

Mostramos un ejemplo de cómo la actualización de los pesos se calcula. El término de la variable momentum no es calculado por razones de claridad en la explicación del ejemplo.



El cambio de peso Δw_{41} se calcula del siguiente modo:

$$\Delta w_{41} = \boldsymbol{a} \cdot \boldsymbol{d}_4 \cdot o_1$$

o_1 Comienza siendo la señal activa de esta conexión, llamada la salida de la neurona 1. Para calcular el error local $\boldsymbol{d}_4$ de la neurona 4, los errores de las salidas neuronales 7 y 8 son, primero que nada, calculados:

$$\boldsymbol{d}_i = f'(I_i) \cdot (z_i - o_i), \qquad i = 7,8$$

Estos errores son propagados hacía atrás a la capa escondida. $\boldsymbol{d}_4$ es calculado entonces usando:

$$\boldsymbol{d}_4 = f'(I_4) \cdot (\boldsymbol{d}_7 \cdot w_{74} + \boldsymbol{d}_8 \cdot w_{84})$$

I_4 es la suma de los pesos de las entradas de la neurona 4, es decir:

$$I_4 = o_1 \cdot w_{41} + o_2 \cdot w_{42} + o_3 \cdot w_{43}$$

Δw_{41} Puede entonces ser calculada de acuerdo a la fórmula anterior.

6.2.8. Otros algoritmos de aprendizaje.

6.2.8.1. Super SAP.

El algoritmo de entrenamiento superSAB es una modificación del algoritmo de backpropagation[Zel,1994], [Dat,2002]. Aquí, para cada peso, una tasa de entrenamiento individual es usada. Durante el entrenamiento, las tasas de entrenamiento individuales son adaptadas continuamente a la superficie del error de la red. Si el signo de la derivada parcial $\frac{\partial E}{\partial w_{ij}}$ no cambia a través de distintos pasos, la tasa de aprendizaje individual es incrementada. En el caso de modificación de el signo, la tasa de aprendizaje es reducida en contra de éste.

6.2.8.2. Resilent Propagation (propagación elástica).

El algoritmo resilent propagation opera con un delta de peso individual por conexión en lugar de una tasa de aprendizaje. Similarmente, como ocurre con el algoritmo superSAB, los pesos son modificados de acuerdo con los signos de los gradientes de la función de error ∂E hacia el peso ∂w_{ij}. Si el signo de las dos últimas derivadas parciales son iguales, la cantidad del cambio de peso se incrementa. En otro caso, se reduce[Dat,2002].

6.2.8.3. Quickpropagation.

Tanto el método de gradiente descendiente como el de backpropagation tienen la desventaja de ser particularmente lentos para converger. Si una simple afirmación puede ser hecha con respecto a la superficie del error, entonces algunas mejoras pueden ser hechas para el algoritmo de backpropagation. [Zel,1994], [Dat,2002].

Asumiendo que la superficie del error en la cercanía a ésta esencialmente pequeña forma una parábola abierta hacia arriba, entonces el mínimo de esta parábola puede ser computado directamente [Fah,1998]. Uno calcula el mínimo de la parábola usando el gradiente actual de la función de error (método de Newton), el gradiente previo y el último ajuste de los pesos. Si la superficie del error no es exactamente una parábola (como es el caso usual), entonces el valor calculado no será un verdadero mínimo, esto será, sin embargo, una mejora con respecto al algoritmo de backpropagatión. Repitiendo iterativamente este paso, el margen de error puede ser reducido, hasta que el mínimo es logrado.

El paso para calcular el ajuste de los pesos $\Delta w_{ij}(t)$ está basado sobre la suma de términos para el gradiente G(t), y el término de la parábola P(t):

$$\Delta w_{ij}(t) = G(t) + P(t)$$

Donde el gradiente G(t) se calcula como:

$$g(t) = \begin{cases} d \cdot \boldsymbol{a} \cdot \sum_p o_{pi} \boldsymbol{d}_{pi} & cuando (t = 0) \vee \Delta w_{ij}(t-1) = 0 \\ 0 & \text{sgn}(s(t-1)) = \text{sgn}(s(t)) \end{cases}$$

El término de la parábola P(t), se calcula como:

$$P(t) = \begin{cases} 0 & para \quad (t = 0) \vee \Delta w_{ij}(t-1) = 0 \\ \dfrac{S(t)}{S(t-1) - S(t)} & para \quad \begin{array}{l}(t \neq 0) \wedge \Delta w_{ij}(t-1) \neq 0 \wedge \\ \left|\dfrac{S(t)}{S(t-1) - S(t)}\right| \leq m\end{array} \\ m \cdot \Delta w_{ij}(t-1) & para \quad \begin{array}{l}(t \neq 0) \wedge \Delta w_{ij}(t-1) \neq 0 \\ \left|\dfrac{S(t)}{S(t-1) - S(t)}\right| > m\end{array} \end{cases}$$

donde,

$\alpha =$ tasa de aprendizaje

o_a = salida de la neurona i del modelo p

δ_a = señal de error de la neurona j del modelo p

d = caída de peso

$S(t)$ = componente de error del gradiente de la función en dirección w_{ij}

m = factor de ajuste máximo [Zel,1994].

Los valores que son requeridos para el cálculo de la parábola se obtienen automáticamente de los pasos del algoritmo Backpropagation. Solamente necesitan ser almacenados como estén disponibles para ser usados en el próximo ciclo de la quickpropagation.

En superficies de errores bien constituidas, el procedimiento de quickpropagation alcanza el error mínimo más rápido que el de backpropagation (de aquí su nombre), provocando que la función de error realmente pueda ser aproximada por parábola abierta hacia arriba.

Ciertas características de la superficie de la parábola necesitan ser consideradas cuando usamos quickpropagation así como para impedir efectos indeseables. En el caso de que la superficie de error comience como una parábola invertida, o en el caso de la vecindad de un local máximo, los ajustes de los pesos deberían tender a este local máximo. Para

impedir esto, se usa una combinación de la parábola con el gradiente normal descendiente. Oscilaciones no deseadas de los pesos pueden ser impedidas incorporando, en el algoritmo de quickpropagation, un limite superior en el ajuste de los pesos.

Típicamente, el límite de los pesos superior suele ser $m = 1.8$ con respecto al ajuste de pesos previo.

Para que los pesos no sean tan grandes, se sugiere en la literatura relacionada a quickpropagation usar un decaimiento de pesos con d=0.0001 [Zel,1994], [Fah,1998], [Dat,2002]. Este valor también es usado en Data Engine.

Particularmente, en los casos de reconocimiento de patrones usando redes neuronales, un gran número de casos existen donde la quickpropagation requiere notablemente menor tiempo que otros métodos. Para la aproximación de funciones, la optimización usando quickpropagation es fuertemente dependiente de la aplicación respectiva y del tamaño de la red involucrada.

6.2.9. El aprendizaje en multilayer perceptrón.

6.2.9.1. Control del umbral de excitación: Bias Neuron.

Tal como las neuronas del cerebro humano, las neuronas del multilayer Perceptrón a menudo incluyen un umbral con el cual la actividad de la función de transferencia es controlada. Una neurona natural es excitada sólo si la suma de los valores de entrada excede un cierto valor de umbral. En las primeras versiones del multilayer Perceptrón, estos valores de umbral podían ser seteados manualmente para cada neurona.[Dat,2002].

Los diseños más recientes consiguen esto con una capa neuronal interna adicional que consiste en exactamente una neurona cuya entrada es seteada en 1. Esta neurona llamada bias-neuron es conectada a todas las neuronas de la red actual como un ajuste de umbral para la excitación de todas las neuronas.

La gran ventaja de esta forma de modelamiento es que las conexiones del bias-neuron (y sus respectivos umbrales) son entrenados juntos con las conexiones normales del MLP haciendo una configuración manual o un proceso de entrenamiento especial innecesario.

El bias-neuron se encuentra disponible en el MLP de Data Engine. Los pesos del bias o los valores de umbral pueden ser vistos en la ventana de conexión.

6.2.9.2. Consideraciones sobre la estrategia de aprendizaje.

El algoritmo de backpropagation encuentra un valor mínimo de error (global o local), mediante la aplicación de pasos descendientes (gradiente descendiente). Cada punto de la superficie de error corresponde a un conjunto de valores de pesos de la red. Con el gradiente descendiente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error.

Por lo tanto, uno de los problemas que presenta este algoritmo de entrenamiento de redes multicapa es que busca minimizar la función de error, pudiendo caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función de error. Sin embargo, ha de tenerse en cuenta que no tiene por que alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

6.2.9.3. Adaptación de los pesos.

Los pesos de la red neuronal MLP son inicializados con valores randómicos. Posteriormente, en el proceso de aprendizaje de al red estos pesos, o conexiones entre las neuronas, irán cambiando. A continuación se exponen las consideraciones sobre la inicialización de los pesos y sus cambios posteriores.

6.2.9.3.1. Inicialización de los pesos.

Los valores de iniciación para los pesos de las conexiones de la red neuronal tienen una influencia sustancial en la tasa de convergencia del proceso de aprendizaje, y debe ser, por lo tanto, seleccionada con cuidado. Un buen rango inicial es entre 0.1 y 0.9, como resultado de lo cual los pesos quedan con un rango sensitivo de la función de transferencia sinusoidal. [Dat,2002]. Sin embargo, esto podría ser un expediente para inicializar los pesos con valores muy grandes, para entrar a la región de saturación de la función de transferencia directamente. Esto es apropiado cuando información binaria debe ser procesada.

Es recomendable después del primer entrenamiento comparar los pesos neuronales actuales con los pesos iniciales (en el editor de pesos neuronales). Si la diferencia entre

los pesos de inicialización y los pesos de entrenamiento es particularmente notable, entonces se deben ajustar los parámetros de inicialización adecuadamente. Esto es a menudo recompensado con una convergencia rápida de la red durante fases futuras de entrenamiento.

6.2.9.3.2. Criterios de adaptación de los pesos neuronales.

1. **Adaptación después de un solo paso:** los pesos son adaptados inmediatamente después de la presentación de cada ejemplo de entrenamiento. Esto resulta en un cambio en la dirección del avance del gradiente del error de la neurona individual. Este proceso también se conoce como la **estrategia delta**.

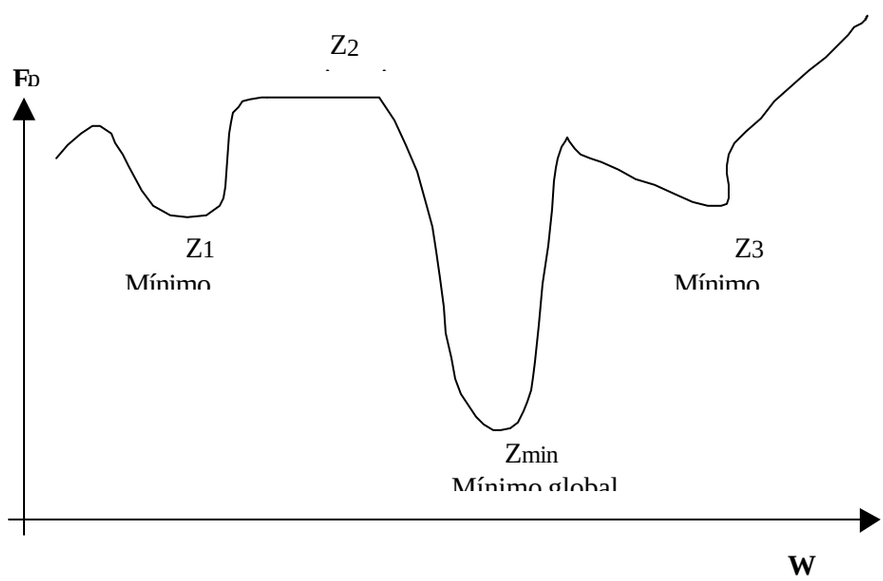
En este caso, se deben tomar algunas consideraciones con respecto al estudio que estemos haciendo: Si los ejemplos son presentados a la red en su secuencia natural, como pudiera ser el caso de secuencias de mediciones de algún fenómeno, el aprendizaje depende fuertemente de esta secuencia. Este efecto es favorable si el orden de aparición de los ejemplos representa fuertemente la información que es relevante, como sucede en el estudio de algunos modelos. Para tareas de clasificación, sin embargo, este efecto no es favorable y conviene presentar los ejemplos de modo aleatorio.

2. **Adaptación después de una época:** los pesos son adaptados sólo después de que un conjunto de ejemplos de aprendizaje han sido presentados. La cantidad de ejemplos después de los cuales un cambio en los pesos toma lugar, es referido como una época. La longitud de una época es generalmente idéntica al número de patrones de aprendizaje. Este método es llamado **backpropagation acumulativa.** Los errores de las neuronas individuales son acumulados sobre la longitud entera de la época, y al

final de la época, son sumados a los pesos previos en un paso único. Esto corresponde al cambio de los pesos en la dirección del avance del gradiente de la función de error global de la red. Este método no podría ser empleado con longitudes de épocas grandes, pues un gran número de operaciones es entonces requerido para una actualización individual de los pesos. Esto podría conducir a una función de convergencia muy lenta para el método.

6.2.9.4. Control sobre la convergencia: configuración de la tasa de aprendizaje (α).

La configuración de la tasa de aprendizaje está fuertemente relacionada con el control sobre la convergencia del algoritmo de aprendizaje. Esto se traduce en la manera como nos desplazamos sobre la curva o superficie de error, donde lo que nos interesa es acercarnos a un mínimo, el cual representa un error que consideramos aceptable. El valor de α, representa la velocidad con que nos desplazamos sobre la superficie de error. El aspecto de la superficie de error es como se muestra a continuación:



En la figura, W representa los posibles valores de la matriz de pesos de la red. Como se puede observar, con incrementos grandes se corre el riesgo de pasar por encima de un punto mínimo sin estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que ocurra esto. En consecuencia, existen diferentes criterios para configurar α.

6.2.9.5. Utilización de una tasa de aprendizaje constante.

Si utilizamos una tasa de aprendizaje constante, esta puede caer en dos categorías: grande o pequeña. El limite para definir si este valor es considerado pequeño o grande, depende de la naturaleza del problema y de la arquitectura de la red (o diseño). Si se escoge una tasa de aprendizaje alta, entonces la red aprende más rápido; mayores ajustes de los pesos conducen a una convergencia más rápida. Con esto, el proceso de aprendizaje toma lugar rápidamente.

Sin embargo, puede ocurrir que un ajuste demasiado grande de los pesos provoque que un valor óptimo sea sobrepasado, sin alcanzar valores óptimos para los pesos.

Por lo tanto, conviene utilizar una tasa de aprendizaje pequeña; así el resultado logrado es tan preciso como sea posible.

6.2.9.6. Utilización de una tasa de aprendizaje variable en dos fases.

Al usar MLP para la aproximación de funciones, se puede observar a menudo que el error, aún después de un largo periódo de entrenamiento, raramente cae debajo del orden de la magnitud de la tasa de aprendizaje usada. El resultado final resulta inexacto, significa que la red neuronal ha aprendido pobremente. [Zel,1994], [Brau,1995] ,[Dat,2002].

Un método para resolver este problema, en muchas ocasiones, es no usando una tasa de aprendizaje constante. en este caso se usa una tasa de aprendizaje alta en una primera fase y posteriormente se reduce considerando los siguientes aspectos:

Una alta tasa de aprendizaje es escogida, así que en el comienzo de un proceso de aprendizaje, se logra una rápida adaptación de los pesos. De ese modo, uno intenta alcanzar el óptimo a través de la superficie de error usando un largo paso para aproximándose al mínimo, tan rápido como sea posible. Después de que el actual mínimo fue sobrepasado, el error deja de caer, entonces se reduce la tasa de aprendizaje

Esta no es una regla universal en la cual el modo de la tasa de aprendizaje debería ser reducido papa lograr este efecto deseado. El usuario puede inspeccionar la gráfica de la curva de aprendizaje para establecer en cual punto el error deja de caer, él puede reducir la tasa de aprendizaje por sí mismo. Valores típicos para decrecer son partiendo por la mitad o reduciendo por un factor cercano a 10. Una clara mejora a la red puede ser lograda frecuentemente con experimentación de diferentes valores. [Brau,1995], [Sil,1990] , [Jac,1988].

En el caso de este procedimiento, los siguientes puntos deben ser considerados: si la tasa de aprendizaje es reducida muy bruscamente, el aprendizaje se vuelve demasiado lento y/o el aprendizaje adicional no ocurre, la red neuronal es impedida de producir el mejor resultado. Si uno reduce la tasa de aprendizaje si se está en una parte plana de la superficie de error o cerca de un mínimo local, uno puede seguir atrapado allí, típicamente sin lograr los resultados óptimos.

6.2.9.7. **Utilización de una tasa de aprendizaje continuamente decreciente.**

Otra variante para cambiar la tasa de aprendizaje es usar la tasa de aprendizaje continuamente decreciente. En cada paso, la tasa de aprendizaje es multiplicada por un valor cercano pero ligeramente inferior a 1.0 (tasa de decaimiento de aprendizaje). La multiplicación repetida por una constante de decaimiento continuamente transforma la tasa de aprendizaje inicial a un valor eventual de 0.0. Para valores cercanos a 1, el proceso corresponde al decaimiento exponencial de la tasa de aprendizaje. Una constante de decaimiento razonable se encuentra dentro del rango de 1.0 - 10E-6 = 0.999999 y 1.0 - 10E-9 = 0.999999999. En casos de duda, se debe escoger una constante cercana a uno.

6.2.10. Elección de neuronas en la capa escondida y el proceso de poda.

La elección de muy pocas neuronas en la capa escondida da lugar a una red incapaz de representar la mayoría de los problemas. Demasiadas neuronas, tienen la desventaja que la red pierde cualquier capacidad de generalización, la red tiende entonces a aprender los ejemplos de entrenamiento "de memoria" (overfitting). Llevado al extremo, demasiados pesos podrían llegar a estar disponibles en la red, donde cada patrón de entrenamiento es representado separadamente por un peso (o conexión) de la red neuronal. En cambio, la red está disponible para reproducir los modelos de entrenamiento exactamente, pero producirá un error considerable si es enfrentada a otra fuente de datos aún si cambiaran ligeramente con respecto a los datos originales de entrenamiento. Demasiados pesos reducen la capacidad de generalización (aplicabilidad general) de la red.

Como una guía, cuando escogemos el tamaño de la red, se puede decir que una red demasiado grande aprende de memoria, mientras que una red más pequeña produce mejores generalizaciones.

El tamaño de la red es escogido generalmente con heurística, donde el número de neuronas disponibles, o su grado de libertad, es idéntico a el numero de pesos sinápticos, entonces se coloca en relación con el número de ejemplos de entrenamiento. Como regla, el número de ejemplos debería ser entre dos y cuatro veces el número de pesos de la red neuronal. Tal aproximación ha sido utilizada con éxito en muchos casos reales de prueba; sin embargo, no se puede generalizar como una ley universal.

Otra forma de determinar el tamaño óptimo de la red es usar métodos adaptativos a la estructura. Aquí, la topología de la red neuronal puede ser modificada durante el proceso de entrenamiento usando dos estrategias básicas:

1. Aumentando el número de neuronas en la capa escondida, es decir, se comienza con una pequeña red y, si es necesario, se agregan más neuronas en cada una de las capas escondidas. [Fah,1998],[Dat,2002].

2. Reduciendo el número de conexiones entre neuronas. Para esto, uno comienza con una red muy grande y remueve las conexiones o sinapsis, que parecieran tener un efecto insignificante sobre el funcionamiento de la red (poda). [Ree,1998] ,[Dat,2002].

Las diferencias entre estos dos métodos pueden ser encontradas en diversas publicaciones científicas y en muchas aplicaciones diferentes [Brau,1995], [Zel,1994] [Dat,2002].

Para facilitar el uso de este proceso, diferentes algoritmos se han establecido particularmente bien como arquitecturas adaptativas para MLP. El proceso de poda está

basado en lo siguiente: Se comienza con una red grande, en el sentido que se encuentran presentes todas las conexiones entre todas las neuronas. Entonces las conexiones que carecen de importancia son removidas quitando sus pesos sinápticos correspondientes [Wei,1992],[Dat,2002].

Con los procedimientos de poda disponibles en Data Engine, los pesos que son considerados sin importancia, son los pesos que contribuyen menos a la suma total de los pesos de la neurona. Estos pesos contribuyen muy poco a la suma total y son aquellos que tienen valores muy próximos a cero, o esos que son activos pero que son multiplicados por cero.

Como un valor característico, la llamada relevancia r_{ij} de los pesos w_{ij} es usada, la cual corresponde a un filtro pasa bajos de la actual contribución de cada peso respectivo:

$$r_{ij}(t) = \frac{\boldsymbol{t}_p \cdot r_{ij}(t-1) + s_{ij}}{\boldsymbol{t}_p + 1}$$

Donde $\boldsymbol{t}_p$ es la constante de tiempo de la banda baja y s_{ij} es la contribución de los pesos w_{ij} de la señal de salida de la neurona.

El calculo de s_{ij} es diferente, dependiendo de la estrategia de aprendizaje adaptativo escogida. para la estrategia de aprendizaje acumulativo, esta es:

$$s_{ij} = \left| w_{ij} \right|$$

Mientras que para la estrategia de paso único es:

$$s_{ij} = \begin{cases} \left| w_{ij} \cdot o_{ij} \right| & para \quad \text{sgn}(\delta_i(t-1)) = \text{sgn}(\delta_i)) \\ 0 & para \quad \text{sgn}(\delta_i(t-1)) \neq \text{sgn}(\delta_i)) \end{cases}$$

Si la relevancia de los pesos cae debajo de un valor determinado (el umbral de poda),entonces el peso es removido del umbral de poda. El control del proceso de poda es administrado ajustando el umbral de poda y la constante de tiempo τ_p del filtro pasa bajos para el cálculo de la relevancia.

Es generalmente cierto que el umbral de poda debería ser ajustado solamente después de una cuidadosa consideración, puesto que un umbral demasiado alto puede provocar el retiro de conexiones que son relevantes. De este modo, la red es reducida al grado donde no puede resolver largamente el problema a mano. En casos extremos esto podría significar la remoción de todos los pesos de la red, junto con cualquier funcionalidad considerada en ellos.

Valores razonables para el umbral de poda no pueden ser generalizados, puesto que esto depende fuertemente de la topología de red considerada y del problema en sí mismo. Cuando el umbral de poda es muy pequeño (cercano a 0.0001), cualquier posibilidad de poda es eliminada. Altos valores de umbral (cercanos a 0.4) conducen a cambios drásticos en la arquitectura de la red.

La constante de tiempo τ_p para el filtro pasa bajo debería ser seleccionada de tal modo que una o más épocas son recordadas. Valores típicamente usados son entre 10 y 50 veces la longitud de época, esto puede, sin embargo, diferir dependiendo de los casos [Jan,1995]. Data Engine multiplica automáticamente la constante de tiempo usada por el

número de épocas especificadas, independientemente de la estrategia de aprendizaje, y en el caso del aprendizaje de un paso, por la longitud de la época.

Una constante grande, no representa un peligro, la relevancia comienza lentamente a bajar, y algunas conexiones serán removidas. Esto es efectivamente recomendado para comenzar la poda con una constante de tiempo t_t grande y un pequeño umbral de poda.

6.2.11. Selección de los parámetros de la red.

6.2.11.1. Arquitectura

Se ha probado que el MLP con regla de backpropagation es una herramienta de aproximación universal. Esto significa que con cualquier grado de exactitud necesario, puede existir una configuración de red que satisfaga los requerimientos necesarios [Hor,1991]. Desafortunadamente, esto no implica que exista ayuda para configurar los parámetros de modo que se logre esto. La configuración de la red es lograda usando principalmente criterios heurísticos. [Dat,2002].

Primero que nada, el tamaño de la red debe ser establecido. El tamaño de la red es determinado por el número de capas escondidas y el número de neuronas en dichas capas. Las siguientes guías están disponibles para determinar el número de capas escondidas. [Mas,1993], [Dat,2002].

1. Cuando la función consiste en un número de punto finitos, una red de tres capas es apropiada.
2. Cuando la función es continua la cual es definida en un conjunto compacto, una red de tres capas es disponible para aprender esta función. Conjunto compacto implica que las entradas tienen limites claros, es decir, sus rangos de valores no son ilimitados.

3. Una red de tres capas puede también aprender distintas funciones que no respondan al criterio dicho anteriormente. Bajo ciertas condiciones, por ejemplo, las funciones con entradas distribuidas normalmente, (no compactas), pueden ser aprendidas.
4. Todas las otras funciones pueden ser aprendidas en una red de cuatro capas.

Como es indicado en los tres primeros puntos, una red de tres capas es el modelo más apropiado en la mayoría de las situaciones prácticas. Cuatro capas son requeridas por ejemplo, cuando una función requiere ser aproximada cuando es continua por tramos, es decir, posee discontinuidades en distintos puntos. Tal tipo de funciones no pueden ser aprendidas en una red de tres capas. Las redes de cuatro capas también son empleadas en tareas de clasificación de modo gradual.

El número de neuronas de entradas se corresponde normalmente con el número de variables de entrada del proceso a ser modelado. Al seleccionar las neuronas de salida, es aconsejable que no se entrene la red para varias tareas simultáneamente. Si por ejemplo, una tarea de pronóstico es hecha sobre diferentes intervalos de tiempo (una semanalmente y la otra anualmente), datos de entrada diferentes serán requeridos para estas funciones diferentes. Para el caso de pronóstico semanal serán requeridos datos tomados diariamente o a cada hora, mientras que en el pronóstico anual estará basado en valores mensuales o semanales, donde la predicción anual estará sujeta a una fuente de error mayor que la semanal. Esto podría tener una fuerte influencia durante el proceso de entrenamiento, donde la predicción podría ser aprendida con una eficiencia menor con respecto al modelamiento de este problema en una red distinta.

Jansen [Jan,1995] propone la fórmula siguiente para el proceso heurístico relativo a la selección de las neuronas en la primera capa escondida[Dat,2002]:

$$\text{Número de neuronas} = 5 \cdot J \cdot max_{ij} \left\{ \Delta \hat{x}_j \, \hat{v}_{ijG} \right\}$$

Donde

- J :número de entradas
- $\Delta \hat{x}_j$ amplitud esperada de la variable de entrada j
- $\hat{v}_{ijG}$ frecuencia del limite superior deseado de la variable de salida i sobre la variable de entrada j.

No existe una regla correspondiente para determinar el número de neuronas de la segunda capa escondida. Una aproximación puramente empírica es necesaria aquí.

Otra pauta que se podría usar para determinar el número de neuronas es que la probabilidad de altas frecuencias que ocurren en el espectro de salida de la red neuronal, se incrementa como el número de capas escondidas se incrementa. Sin embargo, como la meta es obtener la aproximación más pareja, es más conveniente seleccionar un pequeño número de capas escondidas.

6.2.11.2. Estrategia de aprendizaje y orden de presentación.

Aprendizaje de un solo paso con secuencia de presentación aleatoria debería ser seleccionada como la estrategia de aprendizaje estándar. Aprendizaje acumulativo debería ser empleado sólo cuando pequeñas épocas son empleadas (un poco menos de 100 ejemplos de aprendizaje).

La presentación de ejemplos secuenciales ahorra tiempo de computación, como una nueva secuencia aleatoria no tiene que ser puesta para los ejemplos de entrenamiento previamente a cada época. Este modo de presentación debería ser usado sólo cuando los registros de datos de ejemplo ya están disponibles en secuencia aleatoria. La presentación secuencial también produce buenos resultados cuando uno está trabajando con datos con ruido.

6.2.12. Consideraciones sobre la Función de transferencia, tasa de aprendizaje, Momentum y tasa de decaimiento.

Las redes neuronales del tipo MLP tienen la ventaja de disponer de distintas tasas de aprendizaje para las diferentes capas de la red. La tasa de aprendizaje de la capa escondida debería ser generalmente más grande que la tasa correspondiente a la capa de salida.

Para asegurar una convergencia en el proceso de aprendizaje, la tasa de aprendizaje debe cumplir cabalmente la siguiente condición:

$$0 < tasa\_de\_aprendizaje < \frac{2}{numero\_de\_neuronas\_en\_la\_capa\_precedente}$$

En el caso del aprendizaje de un paso, la tasa de aprendizaje calculada de este modo debería ser multiplicada por un factor de seguridad de 1/10.

Cuando usamos las redes MLP para reconocimiento de patrones y para clasificación, funciones de transferencia no lineales (sigmoid, tanh) son usadas entre todas las capas de la red neuronal. Para la aproximación de funciones es recomendado usar funciones de transferencia lineal para las neuronas en la capa de salida lo cual conduce típicamente a resultados más precisos. En el mismo instante, estas neuronas reaccionan más sensiblemente a los ajustes de los pesos mientras aprenden. La tasa de aprendizaje debería tomar esto en cuenta, y para las conexiones lineales son más pequeñas que las usadas para las funciones sinusoidales (en el orden de 10 o 100 veces más pequeñas).

Cuando usamos el MLP para aproximar funciones se propone una tasa de aprendizaje entre 0.01 y 0.0001 para las neuronas lineales (salidas) y entre 0.05 y 0.001 para las neuronas escondidas (no lineales). Uno puede comparar esto con la teoría de aprendizaje estocástico. [Fu,1987], [Dat,2002].

Aunque el proceso de aprendizaje usando estas pequeñas tasas de aprendizaje puede ofrecer pérdidas para un largo tiempo, el decaimiento es en muchos casos justificado por resultados mucho mejores. La elección de los parámetros para momentum y decay es muy dependiente del tipo de estrategia de aprendizaje adoptada. La tasa de decaimiento es solamente significativa con aprendizaje de un solo paso (delta).

Rangos razonables para momentum son mostrados en la siguiente tabla:

Estrategia de aprendizaje	**Momentum**
Delta	07....0.9
Delta acumulativo	< 0.3

El uso de una tasa de decaimiento (para aprendizaje de un solo paso) normalmente mejora la generalización de la red, mediante la disposición de pesos uniformemente

pequeños para las neuronas. La tasa de decaimiento siempre debería ser cercana a 1. Cuando se cuenta con un número sustancial de ejemplos involucrados, la siguiente regla puede ser aplicada:

$$tasa\_de\_decaimiento = 1 - \frac{1}{10 \cdot numero\_ejemplos\_de\_entrenamiento}$$

6.2.13. Proceso de entrenamiento.

Un entrenamiento exitoso de la red neuronal es revelado con el hecho de que la red puede reproducir datos conocidos con un mínimo de error y clasificar datos desconocidos (es decir, datos no empleados en la fase de entrenamiento) correctamente. Esto es necesario así para interrumpir el entrenamiento de la red repetidamente para superar las fases de test estableciendo si la exactitud requerida de la aproximación ha sido lograda.

En todos los casos, la siguiente regla debe ser observada: el dato usado en testeo y en recall nunca debe estar contenido en los datos de entrenamiento de la red.

6.2.14. Overfitting. (Aprendizaje de los ejemplos “de memoria”).

Un problema denominado Overfitting aparece tempranamente durante la fase de entrenamiento. Overfitting se produce, cuando, la red, en vez de aprender las relaciones entre los datos, aprende los detalles individuales de los ejemplos de entrenamiento. Como resultado, la red presenta un error de los datos de test que sobrepasa el error de los datos de entrenamiento. Bajo circunstancias normales, el error de test podría ser levemente más alto que el error de entrenamiento. También es posible que los datos de

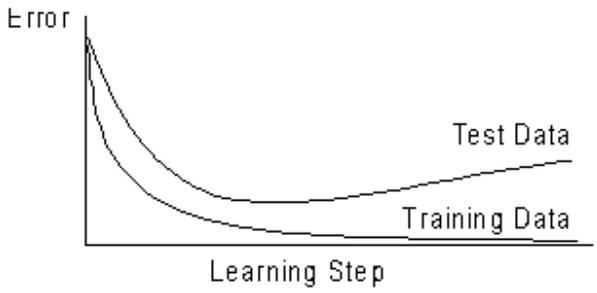
test no tengan las mismas reglas que los datos de entrenamiento. Datos de entrenamiento más apropiados deben ser usados.

El overfitting es el resultado, sobre todo, de un alto grado de libertad (llamado los pesos) de la red neuronal. Consecuentemente, el número de estos grados de libertad debería mantenerse tan bajo como sea posible. Para lograr esto, una red lo más pequeña posible debería ser seleccionada.

La tasa del número de ejemplos de entrenamiento a el número de pesos en la red es también significativo: grandes redes requieren un gran número de ejemplos de entrenamiento. El siguiente procedimiento puede ser aplicado como una regla para seleccionar una cantidad adecuada de datos de entrenamiento. el número de pesos de la red es calculado como:

$$número\_de\_pesos = \sum_{i=1}^{N-1} (n_i \cdot n_{i+1})$$

Donde N es el número de capas y n_i ni es el número de neuronas en la capa i. El número mínimo de ejemplos de entrenamiento es obtenido doblando este número. Es más recomendable tener cuatro veces este número de ejemplos de entrenamiento disponibles.
Las características típicas de una curva de error de los datos de entrenamiento y un error de los datos de test son mostrados en la siguiente figura:



Puede ser visto que el error de la fase de entrenamiento decrece y se aproxima a una asíntota. Desde cierto punto, sin embargo, el error de test comienza a crecer. Este efecto es causado por el overfitting de la red. Más allá de ciertos pasos de entrenamiento, la red comienza a aprender los datos de entrenamiento ”de memoria”. Para impedir esto, el proceso de entrenamiento debe ser parado en el mínimo de la curva de error de test. Con este fin, el proceso de entrenamiento es interrumpido en intervalos para testear la red con los datos de test. El error calculado en este test es mostrado en la ventana de estado de entrenamiento. Tan pronto como comienza a aumentar el error de test, el proceso de entrenamiento es parado. Cuando la red despliega una tendencia a overfitting en un momento muy inicial, hay dos maneras de combatir este efecto[Dat,2002]:

1. Reducir el numero de neuronas escondidas.
2. Usar más datos de entrenamiento.

Es recomendable guardar la red después de cada fase de test. Cuando el incremento del error de test ocurra, la red debe ser reseteada a su estado previo, cargando los parámetros guardados más recientes.

6.2.15. Preprocesamiento de datos.

Las redes neuronales ya han sido empleadas con gran éxito en diversas áreas de aplicación.

En algunos casos, el resultado obtenido, ha sido sorpresivamente de muy buena calidad. Dado que un conocimiento relativamente pequeño es empleado para entrenar una red a causa de su comportamiento de caja negra, los datos son presentados a la red sin ningún paso de procesamiento posterior.

Sin embargo, el grado de cuidado que se debe tener en preparar los datos es de importancia decisiva para la velocidad de aprendizaje de la red y la calidad de la aproximación obtenida con esta. Cada hora invertida en preparar los datos podría ahorrar días en entrenar la red exitosamente.

Las primeras preguntas consideradas aquí son de naturaleza muy general:

- Los datos disponibles, ¿son suficientes y contienen la información correcta?

- Las variables disponibles, ¿muestran el rango de variables pertinentes al problema, tan completamente como sea posible?

- ¿Están ahí los casos de frontera que no son cubiertos por los datos?

- ¿Contienen los datos información irrelevante?

- ¿Son estas transformaciones o combinación de variables (por ejemplo, tasas) las que describen el problema más eficientemente que las variables individuales en sí mismas?

Una vez que todos estos aspectos han sido clarificados, los datos necesitan ser transformados en una forma apropiada para la red. Muchos métodos de normalización son empleados para este propósito. Una característica común a todos los métodos es que una compensación de los ítems de los datos es lograda, después de lo cual son transformados por medio de un factor de escalamiento. Con respecto al proceso de normalización debe notarse que es equivocado normalizar los datos de entrenamiento, de test y de recall sobre las bases de sus respectivos mínimos y máximos. Más bien, el mismo proceso de normalización debe ser usado para todas las fases, asegurando que todos los ítems sigan dentro del mismo rango.

El método más comúnmente empleado para normalización implica mapear los datos linealmente sobre un rango especificado, donde cada valor de la variable x es transformado como sigue[Dat,2002]:

$$x' = \frac{\left(x'_{\max} - x'_{\min}\right)}{\left(x_{\max} - x_{\min}\right)}\left(x - x_{\min}\right) + x'_{\min} = \frac{\left(x'_{\max} - x'_{\min}\right)}{\left(x_{\max} - x_{\min}\right)}x + \left(x'_{\min} - \frac{\left(x'_{\max} - x'_{\min}\right)}{\left(x_{\max} - x_{\min}\right)}x_{\min}\right)$$
$$= factor \cdot x + Offset$$

Donde $x_{max}\ y\ x_{min}$ son los valores máximos y mínimos esperados de las variables respectivamente, y $x'_{max}\ y\ x'_{min}$ especifica el valor deseado de rango para la variable transformada. Un rango de aproximadamente 0.1 a 0.9 es apropiado para la transformación de la variable en un rango sensitivo de la función de transferencia sinusoidal.

Si las variables relativas son aproximadamente distribuidas Normal, ellas pueden ser dadas como una distribución normal estándar con una media 0 y una desviación estándar de 1 aplicando:

$$x' - \frac{x - m}{s}$$

Donde m denota la media y la desviación estándar de la distribución normal original. De este modo, es posible prevenir el efecto donde las señales con variabilidad pronunciada podrían tener preferencia en la fase de aprendizaje.

Jansen [Jan,1995], [Dat,2002] , propone seleccionar los datos de entrada de manera ortogonal, es decir, de tal modo que las variables de entrada sean lo más posible no correlacionadas, y son normalizadas sobre la misma energía de señal. Esto es logrado dividiendo los datos por la raíz de la media cuadrada de los datos de aprendizaje. Sin embargo, la ortogonalidad no es esencial para asegurar la convergencia en la fase de aprendizaje.

Como se intenta lograr la misma calidad de aproximación para todas las salidas de la red, los datos de entrenamiento y de test de la red deberían ser normalizados sobre el mismo rango. Como resultado, los errores de todas las salidas deberían tener la misma magnitud.

La normalización a la misma señal de energía también puede ser empleada aquí.

Aunque la normalización de los datos no es obligatoria, esta es algunas veces ineludible. Si por ejemplo, una función es válida solamente para un rango limitado, por ejemplo la función sinusoidal (entre 0 y 1)la red estará inhabilitada para generar cualquier valor de

salida fuera de este rango. Los datos para las fases de test y de entrenamiento deben ser, por lo tanto, normalizados.

En principio, no es absolutamente necesario normalizar los datos de entrada, cuando a la capa de entrada de la red es asignada una función lineal. Sin embargo, no es aconsejable no normalizar este dato. Como resultado de la normalización, todas las variables adquieren la misma significancia para el proceso de aprendizaje. Si la normalización no se lleva a cabo, las variables con grandes valores tendrán preferencia.

7. Capítulo 8: Solución utilizando Redes Neuronales.

7.1. Asignación de niveles de vulnerabilidad utilizando redes neuronales.

7.1.1. Introducción.

En este capítulo, basándose en los fundamentos arquitectónicos que rigen la construcción y configuración de modelos de redes neuronales, se pretende realizar una serie de experimentos cuyos objetivos son los siguientes:

- Asignar valores de niveles de vulnerabilidad utilizando solamente la información proveniente de las variables explicatorias a posiciones geográficas que carecen de mediciones directas desde terreno.

- Validar el uso de tecnologías de Data Mining (redes neuronales), como herramienta útil de asignación de niveles de vulnerabilidad.

- Caracterizar el comportamiento de las redes neuronales presentando distintas configuraciones de sus parámetros frente a un problema de clasificación puntual.

- Caracterizar el comportamiento de las redes neuronales cuando son entrenadas y testeadas con la misma población de datos, o con poblaciones distintas.

Para esto, se desarrollaron siete modelos de redes con distintas configuraciones desde el punto de vista de sus parámetros, número de neuronas en las capas escondidas, etc. Estos mismos modelos fueron probados en dos escenarios diferentes, frente a la situación de los datos:

1. Modelos de redes donde se utilizaron los mismos datos durante la fase de entrenamiento y test: estas redes son identificadas como MLP1_1,MLP2_2,MLP3_3, MLP4_4,MLP5_5, MLP6_6,MLP7_7. Estas redes difieren entre sí en sus configuraciones.

2. Modelos de redes donde se utilizaron datos diferentes en la fase de entrenamiento y en la fase de test: estas redes son identificadas como redes MLP1, MLP2, MLP3, MLP4, MLP5, MLP6, MLP7. Estas redes difieren entre sí en sus configuraciones.

Cada uno de estos modelos fue desarrollado en Data Engine. La arquitectura de red neuronal escogida fue Multilayer Perceptrón y el algoritmo de aprendizaje fue Back Propagation. Los aspectos de utilización de dicho software para el trabajo con redes neuronales serán expuestos en el siguiente capítulo.

7.1.2. Descripción general de los datos.

La tabla frag_suelo contiene información de las variables auxiliares asociadas a 217 pixeles (o registros). entre estas variables se cuenta con:

- idd: identificador de registro que representa al conglomerado y la parcela de un punto dado. Ejemplo: idd=112, conglomerado=11, parcela=2.
- C1: banda 1 Landsat 7 ETM+
- C2: banda 2 Landsat 7 ETM+
- C3: banda 3 Landsat 7 ETM+
- C4: banda 4 Landsat 7 ETM+

- C5: banda 5 Landsat 7 ETM+
- m: pendiente del terreno en grados asociado a cada pixel
- expo: exposición en grados por pixel.
- Altitud: altitud sobre el nivel del mar en metros.
- Cluster : cluster de grado de vulnerabilidad.

Como ha sido expuesto anteriormente, los posibles valores que puede tomar la variable cluster son expresados en la siguiente tabla:

Grado de vulnerabilidad	**Descripción**
1	Baja
2	Moderada
3	Media
4	Alta
5	Extrema.

Como se verá más adelante fueron consideradas las mismas variables involucradas en la construcción de las redes bayesianas, para la construcción de la red neuronal, con el fin de asignar niveles de vulnerabilidad a cada pixel.

Las siguientes tablas muestra como fueron distribuidos los registros disponibles de cada clase, en ambos casos para las fases de entrenamiento, test y recall:

Tabla: distribución de los datos para MLP1,2,3,4,5,6, Y 7 (datos de entrenamiento y test distintos)

frag_suelo	totales	Entrenamiento	test	recall
1	1	1	1	0
2	11	4	1	6
3	21	7	4	10
4	175	59	36	80
5	9	3	1	5

Tabla: distribución de los datos para MLP1_1,2_2,3_3,4_4,5_5,6_6, Y 7_7 (datos de entrenamiento y test iguales)

frag_suelo	totales	entrenamiento y test	recall
1	1	1	0
2	11	5	6
3	21	11	10
4	175	95	80
5	9	4	5

Los registros asignados a la fase recall, se unirán a los de la tabla auxil, para la clasificación de la red.

7.1.3. Aspectos generales sobre el proceso de aprendizaje y entrenamiento.

Las redes neuronales pasan por tres procesos claramente diferenciados:

Entrenamiento: En esta fase son presentados a la red una colección de registros (datos) compuestos de atributos de entrada (variables que permitirán caracterizar una clase de salida) y su respectiva salida (pertenencia del registro a una clase determinada). Con la presentación sucesiva de ejemplos, la red comenzará su proceso de aprendizaje que consiste en el ajuste de los pesos neuronales, minimizando el error (diferencia entre la salida obtenida y la salida deseada, o pertenencia a una clase). La forma como se ajustan los pesos está determinada por la configuración de los parámetros de la arquitectura de la red y por el algoritmo de aprendizaje.

Test: Presentación de ejemplos con la pertenencia a una clase conocidos. La red clasifica dichos ejemplos, pero no considera la pertenencia de éstos a una clase. Esta información (salida deseada), es contrastada con la salida obtenida. Con la relación entre la salida obtenida y la deseada, se construye una matriz de confusión. Esta matriz de confusión permite caracterizar la calidad de la clasificación hecha por la red, basándose en la salida obtenida (pertenencia de un registro a una determinada clase), y en la pertenencia real de un registro a una clase.

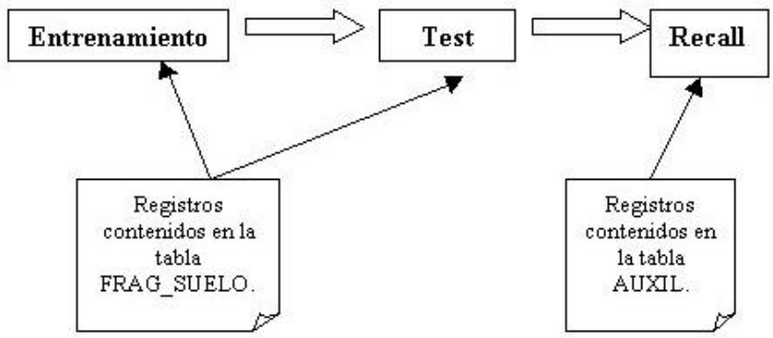
Recall: una vez entrenada la red (**Entrenamiento**) y conocida su capacidad de clasificación (**Test**) se procede a utilizar la red para clasificar conjunto de registros. El objetivo es asignar una clase de salida a cada uno de éstos. Se considera que la calidad de clasificación de la red de los nuevos datos, es como la calidad alcanzada durante la fase de test.

Luego, las redes son entrenadas y testeadas con datos donde conocemos los niveles de vulnerabilidad asociados a cada pixel y registro. En la fase de recall se asignan niveles de vulnerabilidad a la población de datos que deseamos clasificar. De este modo los datos del experimento se dividen en dos grupos:

- Información contenida en la tabla frag_suelo, con asignación de niveles de vulnerabilidad conocidos.(variables idd,c1,c2,c3,c4,c5,altitud,m,expo y cluster conocidos),correspondiente a 217 pixeles.

- Información que representa los pixeles donde solamente se tiene las variables explicatorias conocidas (idd,c1,c2,c3,c4,c5,altitud,m y expo),donde no se cuenta con datos de terreno que permitan calcular el índice de vulnerabilidad ambiental de suelo-agua y su cluster asociado. Esta información se encuentra contenida en la tabla auxil, y representan la población de datos objetivo que queremos clasificar con las redes.

La siguiente figura expone la utilización de estos datos en las fases que atraviesa la utilización de una red neuronal:



Todas las redes pasan por un **new training,** el cual consiste en la iniciación de los pesos neuronales con valores aleatorios. Este proceso finaliza después de mil épocas presentadas a la red (una época es una presentación de todos los registros contenidos en los datos de entrenamiento a la red). Posteriormente, se entrenó la red, a partir del estado anterior de los pesos, con 6000 épocas más. Es decir, el proceso de entrenamiento completo consideró 7000 épocas, cantidad suficiente para conocer con claridad la convergencia de la red, expresada en la curva de aprendizaje, como se verá más adelante.

7.1.4. Aspectos específicos sobre el proceso de aprendizaje y entrenamiento.

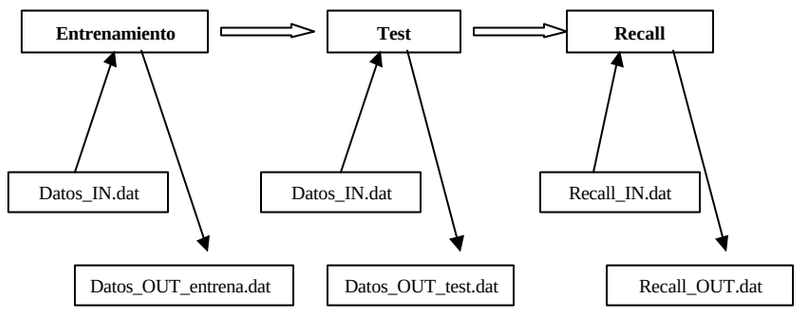
El entrenamiento y test de ambos grupos de redes, como se indicó anteriormente, se hizo con los datos contenidos en la tabla **Frag_suelo**.

En ambos grupos de experimentos fue necesario normalizar los datos para entrenamiento en el rango [0,1]. Este paso es necesario por dos motivos:

- Primero, se debe garantizar que todas las entradas a una red neuronal poseen la misma ponderación. Si las entradas de dos neuronas caen en diferentes rangos, la neurona que posee una escala absoluta más grande será favorecida durante el entrenamiento. Esto se produce por que los algoritmos utilizan reglas basadas en distancias.

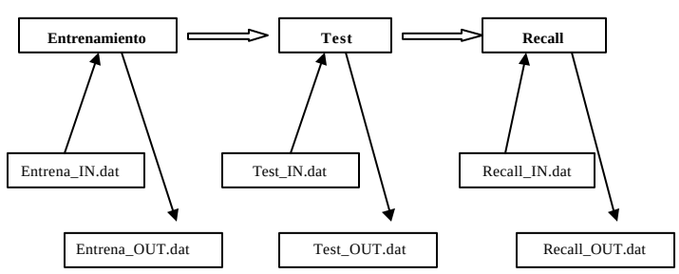
- Segundo, Por las funciones de transferencia utilizadas entre las capas de la red. Si es utilizada una función sigmoidea o una función tangente hiperbólica, se deben utilizar sobre un rango permitido de valores. Una neurona solamente produce un valor de salida permitido si el argumento de la función es menor que 1, puesto que la función de transferencia posee asíntotas en $f(x) = 1$ y en $f(x) = -1$

En el caso de las redes MLP1_1,MLP2_2,MLP3_3,MLP4_4,MLP5_5, MLP6_6,MLP7_7, estos datos pasan por un proceso de normalización, como se ha indicado anteriormente. De este modo DATOS_IN.dat contiene exactamente la misma información que Frag_suelo pero normalizada. La siguiente figura ilustra este hecho:



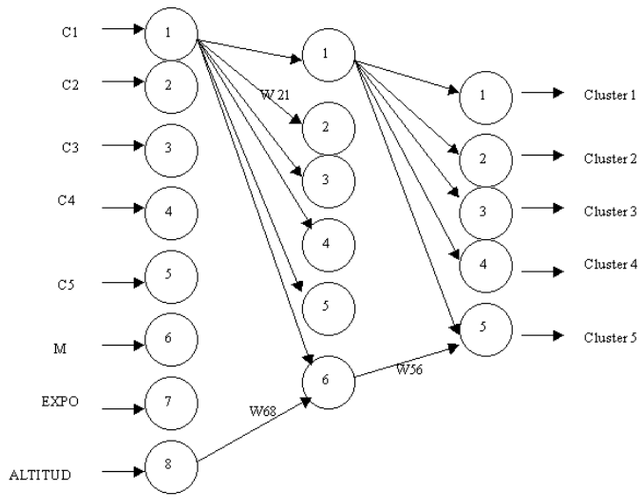
Para el otro conjunto de redes, MLP1, MLP2, MLP3, MLP4, MLP5, MLP6, MLP7. La tabla Frag_suelo fue dividida en dos grupos: datos para entrenamiento y datos para test.

Previo a esta división los datos fueron normalizados. Se considero utilizar aproximadamente el 70% de los registros para la fase de entrenamiento y el 30% para la fase de test. Los registros para entrenamiento constituyen la tabla Entrena_IN.dat. los registros para test forman la tabla Test_IN.dat, como se indica en la siguiente figura:



7.1.5. Arquitectura de las redes del experimento: características generales.

Para resolver el problema de la asignación de los niveles de vulnerabilidad del suelo, se construyeron los modelos de redes anteriormente citados basándose en la arquitectura de la siguiente figura:



Recordemos que, según la arquitectura de una red Multilayer perceptrón, cada neurona de una capa dada, se encuentra conectada con todas las neuronas de la capa siguiente. En la figura se omitieron dichas conexiones para facilitar la caracterización de la arquitectura. En el ejemplo, la capa de entrada de la red posee ocho neuronas, cada una de ellas representa las variables de las cuales tenemos información para cada registro de la tabla o pixel. Cada una de estas se encuentra conectada con la capa escondida. La capa escondida consta de seis neuronas que se encuentran conectadas con las cinco neuronas de la capa de salida. Cada una de las neuronas de salida representa una clase de nivel vulnerabilidad que conocemos (en los datos para entrenamiento y para test) o que pretendemos asignar (en el caso de la clasificación real, fase de recall).

También se ha indicado a modo de ejemplo algunos pesos de las conexiones dadas entre neuronas. w_{56} representa el peso neuronal de la conexión que va de la neurona

6 de la capa escondida, a la neurona 5 de la capa de salida. La neurona 5 representa al cluster 5.

Estos pesos son los que irán cambiando y se irán ajustando de acuerdo al algoritmo de aprendizaje, en este caso, Back Propagation.

7.1.6. Arquitectura y parametrización de las redes del experimento: características específicas.

Las siguientes tablas muestran la configuración de los parámetros y las características arquitectónicas de los experimentos. Los criterios utilizados para dicha configuración se basan en el capítulo 7. La literatura sugiere criterios de configuración de redes que hacen hincapié en la configuración de un parámetro o grupo de parámetros específicos. No existe un estándar que represente un criterio general de configuración de parámetros para redes multilayer Perceptrón; de manera que, se sabe sobre la incidencia de parámetros específicos en el comportamiento de la red, pero se sabe muy poco sobre la interacción entre múltiples parámetros y su incidencia en el desempeño.

Se dice frecuentemente que el diseño de una red neuronal usando algoritmo back-propagation es más un arte que una ciencia, en el sentido que muchos de los numerosos factores involucrados en el diseño son el resultado de nuestra experiencia personal [Hay,1999]. Por lo tanto, como se ha indicado en el capítulo anterior, existen heurísticas que permiten mejorar el desempeño del algoritmo.

Además debemos agregar como factor de éxito o fracaso, en la tarea de clasificación, la calidad de los datos utilizados para entrenamiento o test, que viene dado por la cantidad de información disponible.

En nuestro caso, la muestra utilizada para entrenamiento de la red, es una muestra de distribución aleatoria que obedece a un diseño estadístico, cuyo objetivo es explicar lo máximo posible la varianza de los datos. Debemos agregar a esto que las variables que representan el vector de entrada provienen de distintos contextos (variables de canales satelitales, y variables de terreno) lo que minimiza considerablemente la duplicación de información dada por variables que representan el mismo fenómeno.

De este modo, se realizaron una serie de experimentos probando distintas configuraciones de parámetros y se escogió aquella red que tenga un mejor comportamiento en la calidad de clasificación.

La definición de cada uno de los parámetros considerados se expone en el capítulo "Fundamentos teóricos de redes neuronales y Multilayer Perceptrón".

Parámetros de arquitectura.

	MLP1 MLP1_1	MLP2 MLP2_2	MLP3 MLP3_3	MLP4 MLP4_4	MLP5 MLP5_5	MLP6 MLP6_6	MLP7 MLP7_7
Número de capas escondidas	1	1	1	1	1	1	1
Función de transferencia input	Lineal	lineal	lineal	lineal	lineal	lineal	lineal
Función de transferencia capa escondida	Sigmoid	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid	Sigmoid
Función de transferencia Output	Lineal	sigmoid	Sigmoid	sigmoid	sigmoid	sigmoid	Sigmoid
Shortcuts	No	no	No	no	no	no	no
Número de neuronas en capa escondida	2	8	8	12	12	16	12

Parámetros sobre Método de aprendizaje.

	MLP1 MLP1_1	MLP2 MLP2_2	MLP3 MLP3_3	MLP4 MLP4_4	MLP5 MLP5_5	MLP6 MLP6_6	MLP7 MLP7_7
Learning method:	Backpropagation	backpropagation	backpropagation	backpropagation	Backpropagation	backpropagation	backpropagation
use momemtum term	No	si	Si	si	Si	si	si
use weight decay	No	no	No	no	No	no	no
Learning strategy:	single step	Single step	single step	single step	single step	single step	single step
Presentation order	Random	Random	random	random	Random	random	Random
enable pruning	No	No	Si	si	No	no	No
Relevanse treshold	0,02	0,02	0,02	0,02	0,02	0,02	0,02

Parámetros de Aprendizaje.

	MLP1 MLP1_1	MLP2 MLP2_2	MLP3 MLP3_3	MLP4 MLP4_4	MLP5 MLP5_5	MLP6 MLP6_6	MLP7 MLP7_7
Learning rate (1°escond.)	0,1	0,2	0,2	0,2	0,2	0,2	0,2
Momentum (1°escond.)	0,1	0,7	0,7	0,7	0,7	0,7	0,9
Weight decay (1°escond.)	0,999999	0,999999	0,998648 64	0,999999	0,999999	0,999999	0,999999
Learning rate (salida)	0,1	0,15	0,15	0,15	0,15	0,15	0,15
Momentum (salida)	0,1	0,7	0,7	0,7	0,7	0,7	0,7
Weight decay (salida)	0,999999	0,999999	0,998648 64	0,999999	0,999999	0,999999	0,999999
Learning rate (habilitación)	no	no	no	no	No	no	no
Learning rate decay (valor)	0,999999	0,999999	0,999999	0,999999	0,999999	0,999999	0,999999

Limites Superior e inferior para iniciación de los pesos

	MLP1 MLP1_1	MLP2 MLP2_2	MLP3 MLP3_3	MLP4 MLP4_4	MLP5 MLP5_5	MLP6 MLP6_6	MLP7 MLP7_7
range lower bound	-0,1	0,1	0,1	0,1	0,1	0,1	0,1
range upper bound	0,1	0,9	0,9	0,9	0,9	0,9	0,9

Condiciones de parada.

	MLP1 MLP1_1	MLP2 MLP2_2	MLP3 MLP3_3	MLP4 MLP4_4	MLP5 MLP5_5	MLP6 MLP6_6	MLP7 MLP7_7
stop if	n°epocas/1000	n°epocas/1 000	n°epocas/10 00	n°epocas/ 1000	n°epocas/ 1000	n°epocas/ 1000	n°epocas/ 1000
perform a test every 100 epoc.	No	no	no	no	no	no	no
keep best training state	No	no	no	no	no	no	no

7.2. Descripción de los resultados para cada modelo.

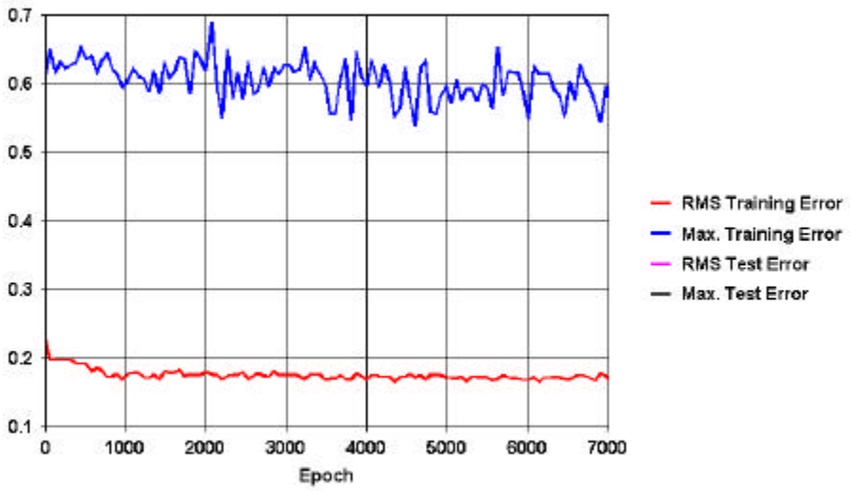
A continuación se describirán los resultados obtenidos en cada experimento basándose en la curva de aprendizaje y en la matriz de confusión. Ambos resultados permitirán identificar el mejor modelo de red y los mejores resultados en la clasificación.

La curva de aprendizaje nos entregará información sobre el comportamiento de la red a lo largo de su proceso de aprendizaje y la forma como ésta converge. Esta curva está caracterizada por dos medidas de error: el RMS training error y el Max training error. Lo que se espera al analizar estas medidas es que ambas curvas caigan para poder hablar de una convergencia por parte de la red. Cuando una red no converge estos valores tienden a crecer o a mantenerse estables a partir de las primeras épocas. La velocidad con que caen estas curvas se encuentra determinada por el valor de la tasa de aprendizaje α.

La descripción hecha sobre los aspectos de las curvas de aprendizaje y las características de los parámetros que se hagan sobre las redes MLP1 A MLP7, se extienden sobre las redes MLP1_1 a MLP7_7. Recordemos que se trata de la misma red. Es natural que las redes MLPX_X se desempeñen de mejor forma que sobre sus homólogas MLPX. Estos experimentos permiten ilustrar el efecto que tiene entrenar y testear con los mismos datos. También permite suplir la dificultad que posee la muestra en el sentido de contener muy pocos ejemplares en algunas clases en el caso de las redes MLPX.

MLP1

Curva de Aprendizaje.



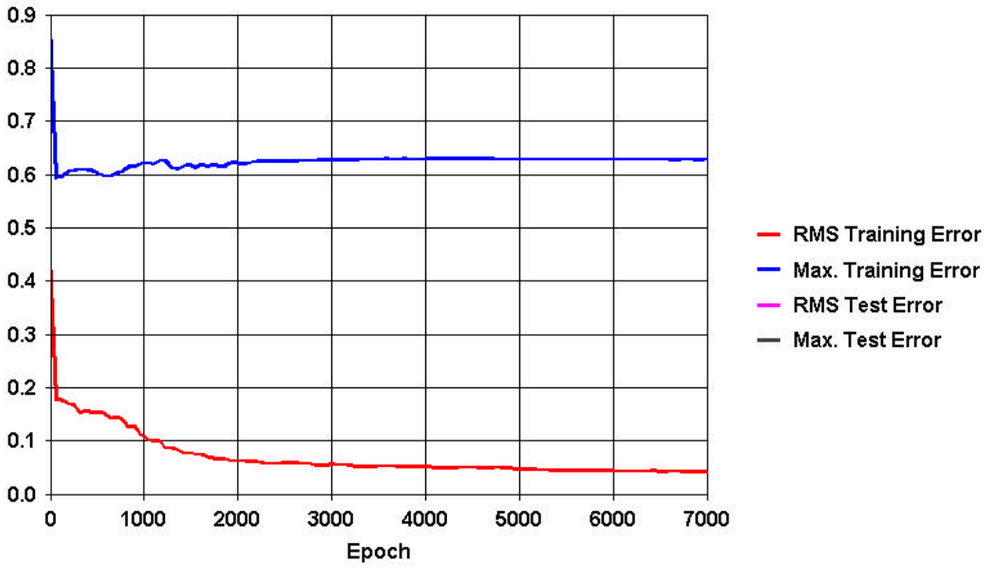
Matriz de Confusión.

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	0	0	**0**	0
3	0	0	**0**	0	0	**0**	0
4	0	1	4	**33**	1	**39**	84.61
5	1	0	0	3	**0**	**4**	0
Total	**1**	**1**	**4**	**36**	**1**	**33/43**	
%precisión en la clase	0	0	0	91.7	0		**76.74**

Se puede apreciar en la curva de aprendizaje las oscilaciones presentes debido al bajo valor de momentum considerado en ambas capas (0.1). Al aumentar este valor, se suavizará este efecto, como podremos apreciar en MLP2. El motivo de esto es que el momentum sobre una conexión neuronal permite considerar el valor anterior, en el tiempo, de la conexión para el cálculo del siguiente, suavizando las curvas de error. Podemos considerar que la red se estabiliza a partir de la época 1000. Existe una tendencia general, a pesar de las fuertes oscilaciones, de ir disminuyendo el error máximo de entrenamiento. Intencionalmente se ha dejado el valor de momentum fuera de un rango aconsejado por la literatura (cuando la estrategia de aprendizaje es

delta, un rango razonable para el momentum es entre 0.7 y 0.9). En la siguiente red **MLP2**, este valor es corregido.

MLP2



Curva de Aprendizaje.

Matriz de Confusión.

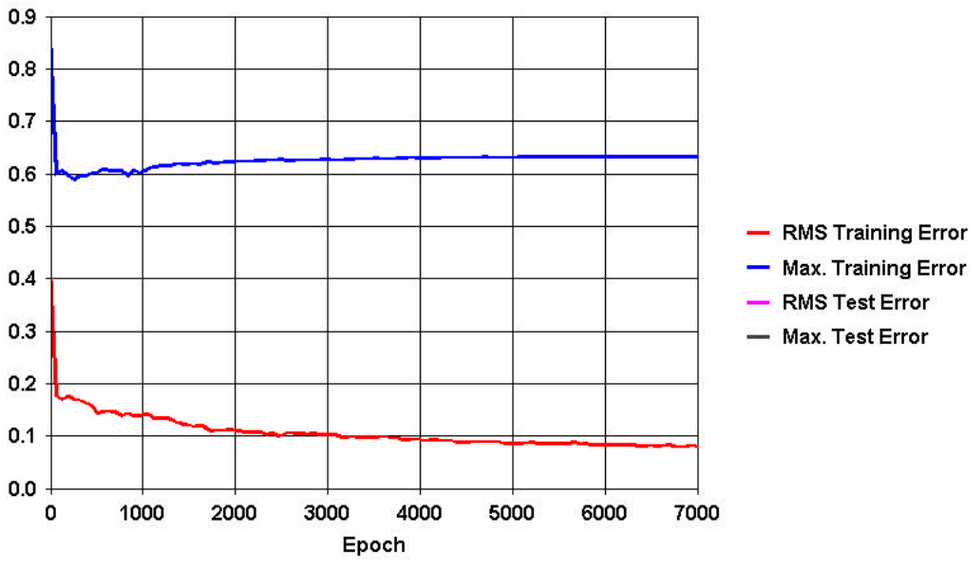
	1	2	3	4	5	total	% precisión en la clasificación
1	**1**	0	0	2	0	**3**	33.33
2	0	**0**	0	4	0	**4**	0
3	0	0	**0**	8	0	**8**	0
4	0	1	3	**20**	0	**24**	83.33
5	0	0	1	2	**1**	**4**	25
Total	**1**	**1**	**4**	**36**	**1**	**22/43**	
%precisión en la clase	100	0	0	55.56	100		**51.16**

En esta red han sido corregidos los valores de momentum para ambas capas (0.7). Al ponderar con un mayor valor el impacto de los pesos previos para calcular el valor de los nuevos pesos, se suprime el efecto oscilatorio. Existe en esta red un aumento del número de neuronas de la capa intermedia (de dos a 8 neuronas); esto provoca una mejor generalización por parte de la red, como se aprecia en las diferencias entre la matriz de confusión MLP1_1 A MLP2_2 (ver más adelante).El aumento de los valores de la tasa de aprendizaje también ha contribuido a una mejora del

desempeño de la red, disminuyendo el valor de RMS a partir de la época 1000. Se recomienda que la tasa de aprendizaje de la capa escondida sea generalmente más grande que la tasa de aprendizaje de la capa de salida, como sucede en esta red (0.2 y 0.15, respectivamente) esta configuración de las tasas de aprendizaje se mantendrá de aquí en adelante en el resto de los experimentos.

MLP3

Curva de Aprendizaje.



Matriz de Confusión.

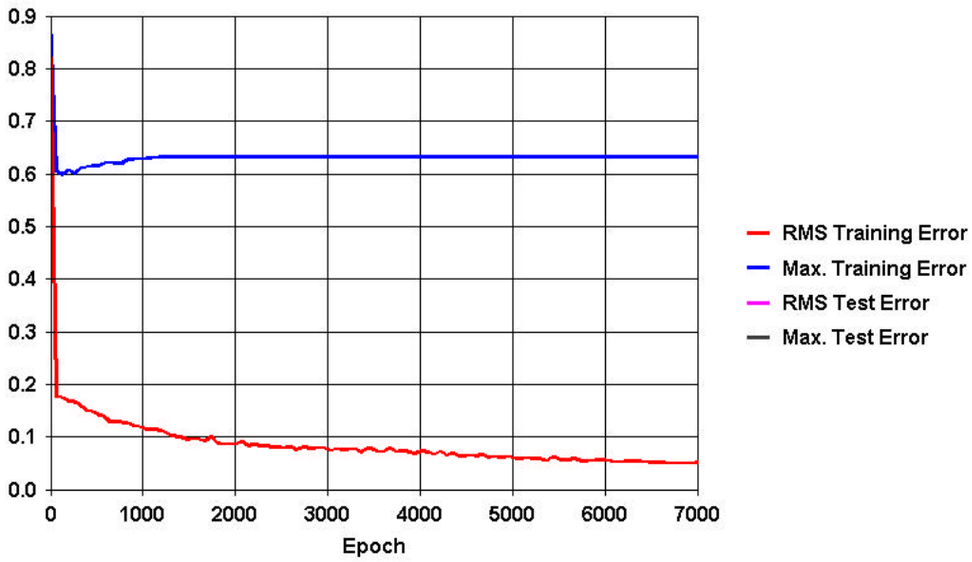
	1	2	3	4	5	total	% precisión en la clasificación
1	**1**	0	0	3	0	**4**	25
2	0	**0**	0	1	0	**1**	0
3	0	0	**2**	8	1	**11**	18.18
4	0	1	2	**24**	0	**27**	88.9
5	0	0	0	0	**0**	**0**	0
Total	**1**	**1**	**4**	**36**	**1**	**27/43**	
%precisión en la clase	100	0	50	66.7	0		**62.8**

La gran modificación que sufre esta red con respecto a MLP2 es la disminución del valor de weigth decay. Sirve para obtener pesos pequeños en cada w_{ij}.Las redes con

pesos pequeños generalizan de mejor modo, como podemos notar en la mejora de la clasificación, al observar las matrices de confusión de MLP2 y MLP3. Es probable que algunos pesos pequeños hallan sido removidos productos de la habilitación de poda (ver tabla de configuración de redes).

MLP4

Curva de Aprendizaje.



Matriz de Confusión.

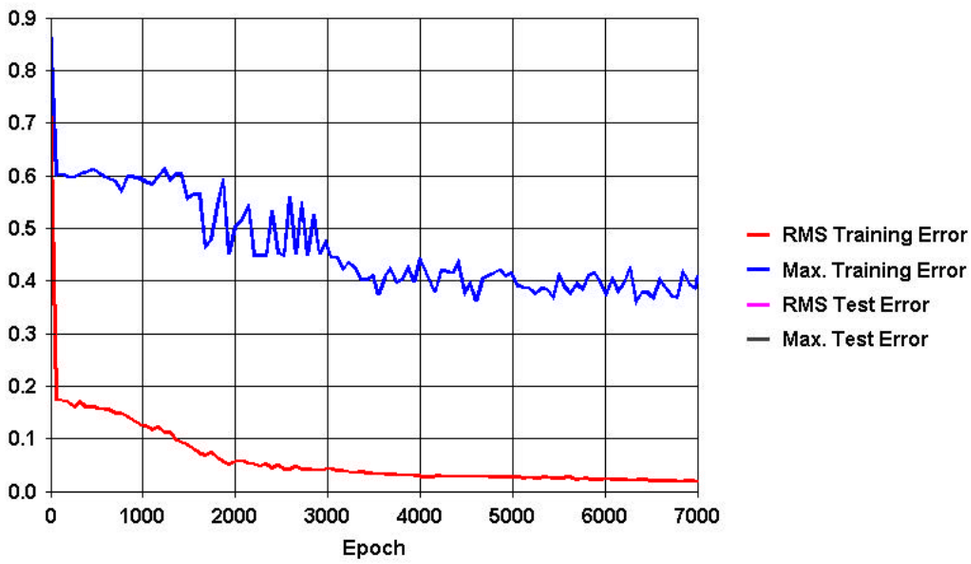
	1	2	3	4	5	total	% precisión en la clasificación
1	**1**	0	0	2	0	**3**	33.33
2	0	**0**	0	6	0	**6**	0
3	0	0	**0**	5	0	**5**	0
4	0	1	4	**23**	1	**29**	79.31
5	0	0	0	0	**0**	**0**	0
Total	**1**	**1**	**4**	**36**	**1**	**24/43**	
%precisión en la clase	100	0	0	63.9	0		**55.81**

Esta red es MLP2 con más neuronas en la capa escondida. Esto hizo aumentar la calidad de la clasificación (comparar las matrices de MLP2 con MLP4). Como se ha dicho anteriormente, el aumento de neuronas sobre la capa escondida provoca que la red tenga mejor capacidad de generalización, pero sin caer en overfitting

(aprendizaje de los ejemplos de entrenamiento “de memoria” e incapacidad de desempeñarse bien frente a ejemplos nuevos). Para esto se ha habilitado la poda sobre la red cuando estos pesos caen debajo de un cierto umbral (ver tablas de parametrización).

MLP5

Curva de Aprendizaje.



Matriz de Confusión.

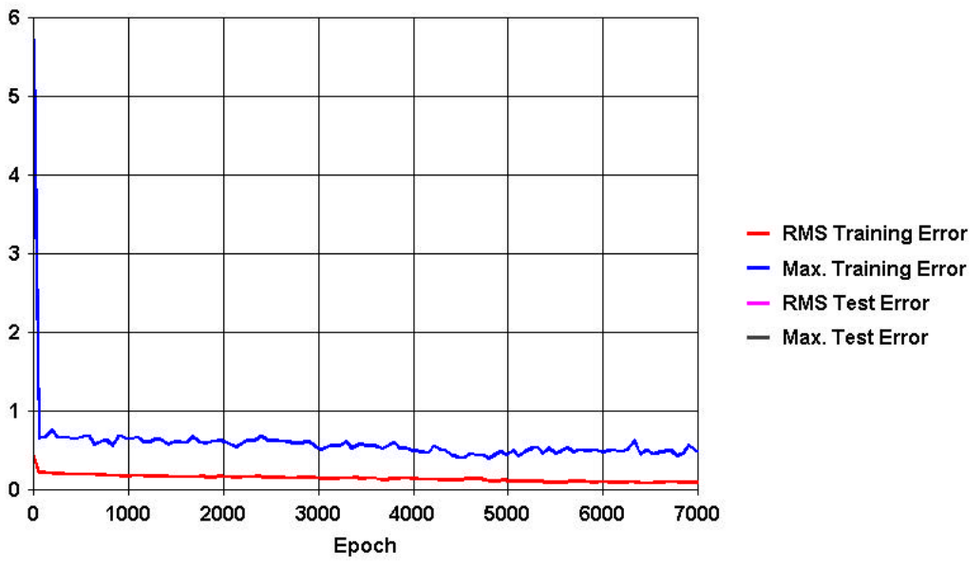
	1	2	3	4	5	total	% precisión en la clasificación
1	**1**	0	0	2	0	**3**	33.33
2	0	**0**	0	4	0	**4**	0
3	0	0	**0**	7	0	**7**	0
4	0	1	4	**22**	0	**27**	81.48
5	0	0	0	1	**1**	**2**	50
Total	**1**	**1**	**4**	**36**	**1**	**23/43**	
%precisión en la clase	100	0	0	61.1	100		**55.81**

MLP5 es idéntica MLP4, solamente que se ha quitado la posibilidad de poda. Esto altera notablemente la curva de error de entrenamiento máximo. En MLP4 existe poda de las conexiones cuando éstas caen debajo de un valor de (0.02). Esta restricción no existe aquí, en beneficio de una disminución del máximo error de

entrenamiento. Sin embargo, no existen grandes cambios en la matriz de confusión y en la curva de RMS test error. La posibilidad de poda resulta irrelevante. Por lo tanto, se mantienen las siguientes redes sin posibilidad de poda.

MLP6

Curva de Aprendizaje.



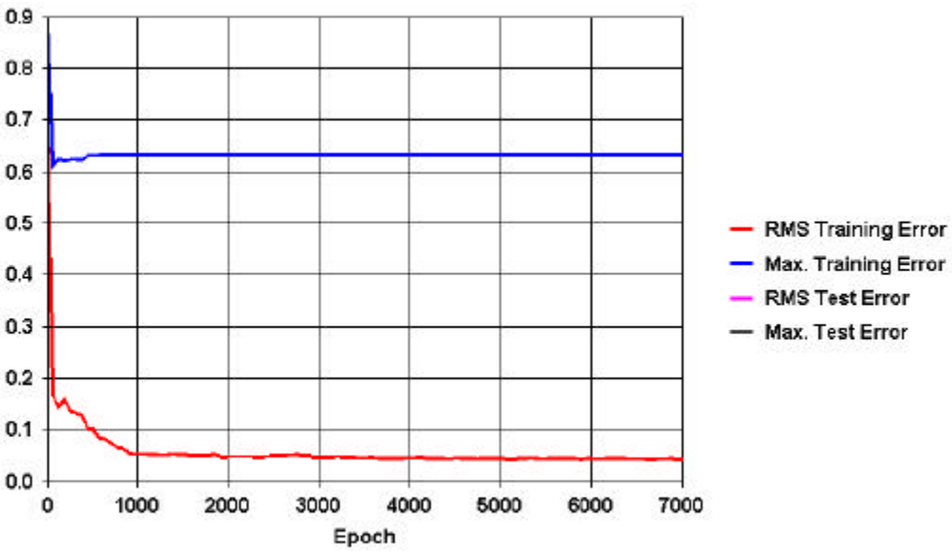
Matriz de Confusión.

	1	**2**	**3**	**4**	**5**	**Total**	**% precisión en la clasificación**
1	**1**	0	0	2	0	**3**	33.33
2	0	**0**	1	5	0	**6**	0
3	0	0	**1**	12	0	**13**	7.7
4	0	1	1	**15**	0	**17**	88.24
5	0	0	1	2	**1**	**4**	25
Total	**1**	**1**	**4**	**36**	**1**	**18/43**	
%precisión en la clase	100	0	25	41.67	100		**41.86**

Se han aumentado nuevamente en cuatro unidades las neuronas de la capa escondida, pero esto trae un efecto negativo en la calidad de la clasificación, como muestra la

matriz de confusión. Por lo tanto, la siguiente red tendrá nuevamente 12 neuronas en la capa intermedia.

MLP7



Curva de Aprendizaje.

Matriz de Confusión.

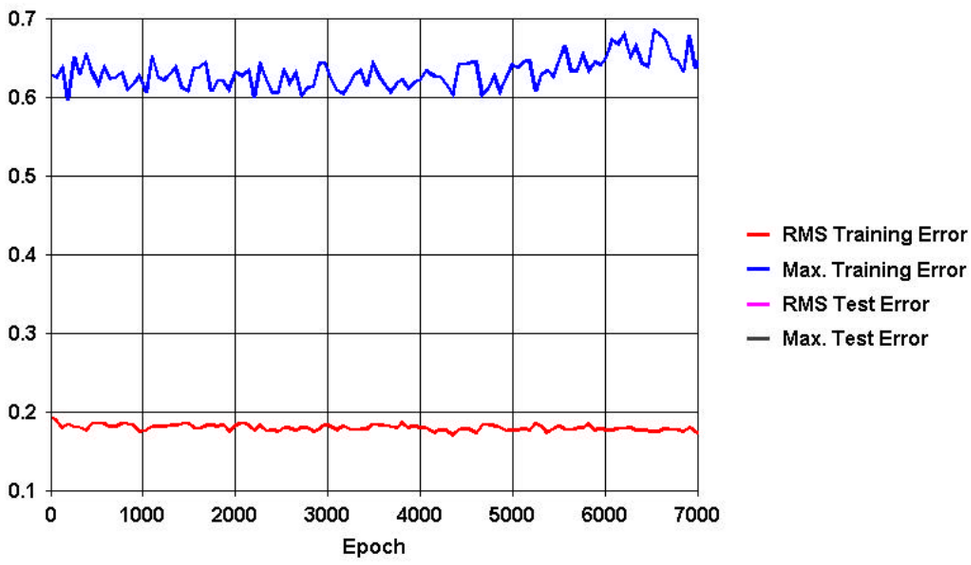
	1	2	3	4	5	total	% precisión en la clasificación
1	**1**	0	0	2	0	**3**	33.33
2	0	**0**	0	3	0	**3**	0
3	0	0	**0**	4	0	**4**	0
4	0	1	4	**27**	1	**33**	81.82
5	0	0	0	0	**0**	**0**	0
Total	**1**	**1**	**4**	**36**	**1**	**28/43**	
%precisión en la clase	100	0	0	75	0		**65.12**

Esta red presenta un aumento en el valor de momentum y una disminución del número de neuronas en la capa escondida. Se hace aquí, que el valor del momentum de la capa escondida sea mayor que el momentum de la capa de salida, colocando en ambas capas los valores extremos recomendados por la literatura, cuando el

aprendizaje es single step (0.9 en la capa escondida y 0.7 en la capa de salida). Recordemos que también tenemos distintos valores para la tasa de aprendizaje (0.2 para la capa escondida y 0.15 para la capa de salida). Como α representa la velocidad con que nos desplazamos sobre la superficie de error, se tomó la decisión de disminuirla en la capa de salida para que la red sea más sensible al desplazarse por la curva de error, aumentando la posibilidad de encontrarse con un buen óptimo. Para el α utilizado en la capa intermedia (más grande) podemos decir en cambio que la red aprende más rápido, por que existen mayores ajustes de los pesos neuronales, que conducen a una convergencia más rápida.

MLP1_1

Curva de Aprendizaje.

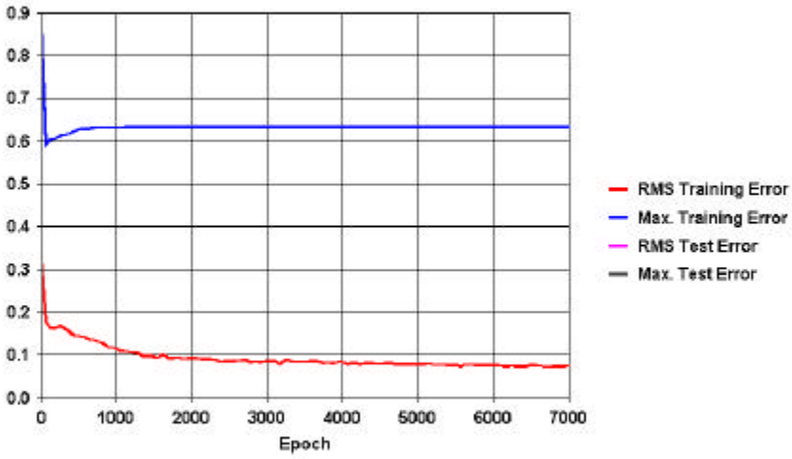


Matriz de Confusión.

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	0	0	**0**	0
3	0	0	**1**	0	0	**1**	100
4	1	5	10	**95**	4	**115**	82.60
5	0	0	0	0	**0**	**0**	0
Total	**1**	**5**	**11**	**95**	**4**	**96/116**	
%precisión en la clase	0	0	9.1	100	0		**82.76**

MLP2_2

Curva de Aprendizaje.

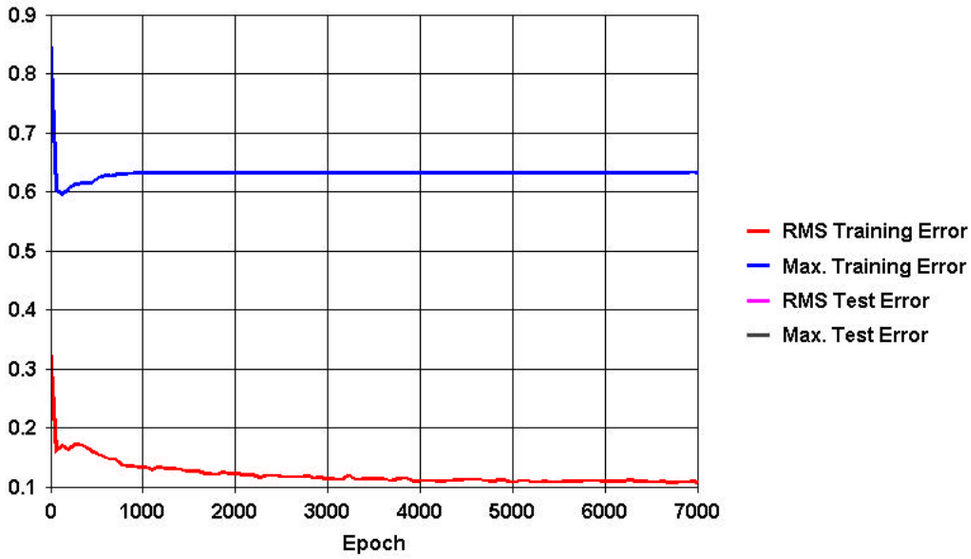


Matriz de Confusión.

MLP2_2	**1**	**2**	**3**	**4**	**5**	**total**	**% precisión en la clasificación**
1	**0**	0	0	0	0	**0**	0
2	0	**1**	0	0	0	**1**	100
3	0	1	**9**	2	1	**13**	69.23
4	1	3	2	**93**	0	**99**	93.94
5	0	0	0	0	**3**	**3**	100
Total	**1**	**5**	**11**	**95**	**4**	**106/116**	
%precisión en la clase	0	20	81.82	97.9	75		**91.38**

MLP3_3

Curva de Aprendizaje.

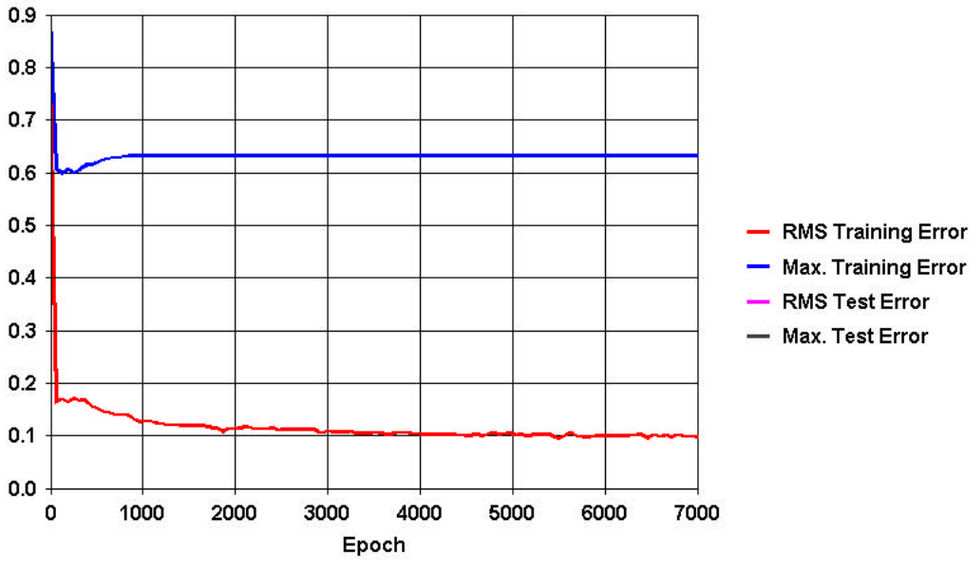


Matriz de Confusión.

	1	**2**	**3**	**4**	**5**	**total**	**% precisión en la clasificación**
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	0	0	**0**	0
3	0	0	**5**	0	0	**5**	100
4	1	5	6	**95**	2	**109**	87.16
5	0	0	0	0	**2**	**2**	100
Total	**1**	**5**	**11**	**95**	**4**	**116**	
%precisión en la clase	0	0	45.45	100	50		**96.23**

MLP4_4

Curva de Aprendizaje.

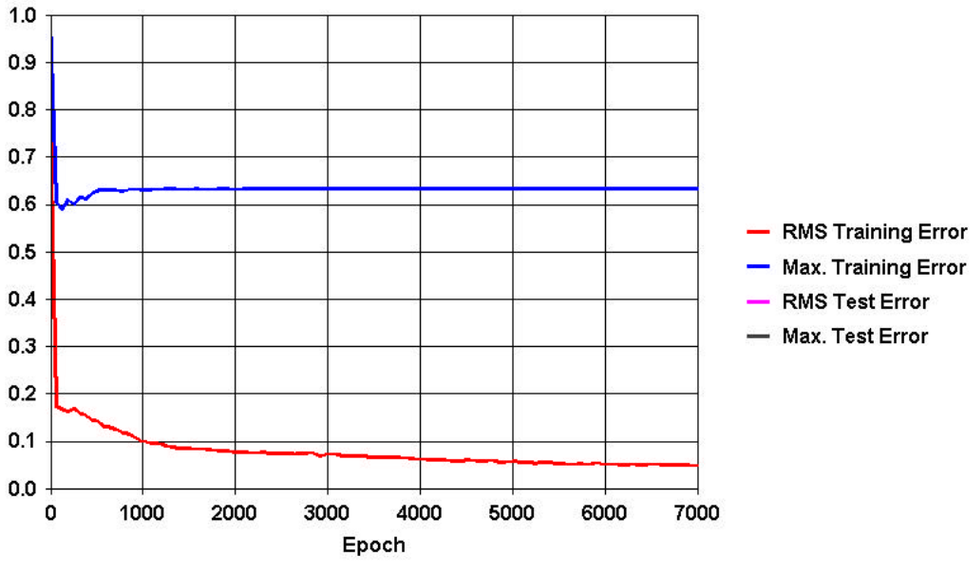


Matriz de Confusión.

	1	**2**	**3**	**4**	**5**	**total**	**% precisión en la clasificación**
1	**0**	0	0	0	0	**0**	0
2	0	**0**	0	0	0	**0**	0
3	0	1	**8**	3	0	**12**	66.67
4	1	4	3	**92**	1	**101**	91.09
5	0	0	0	0	**3**	**3**	100
Total	**1**	**5**	**11**	**95**	**4**	**103/116**	
%precisión en la clase	0	0	72.72	96.84	75		**88.8**

MLP5_5

Curva de Aprendizaje.

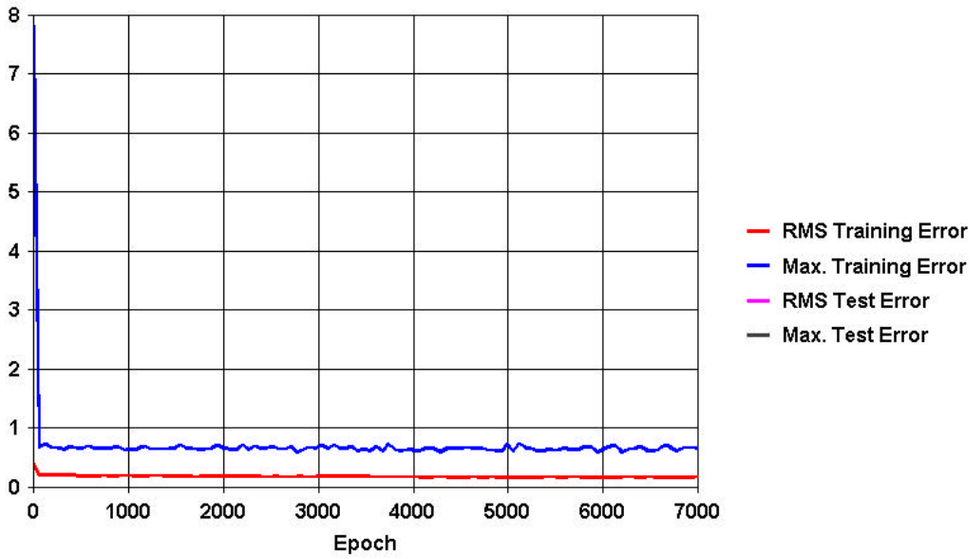


Matriz de Confusión.

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**2**	0	0	0	**2**	100
3	0	0	**8**	0	0	**8**	100
4	1	3	3	**95**	1	**103**	92.23
5	0	0	0	0	**3**	**3**	100
Total	**1**	**5**	**11**	**95**	**4**	**108/116**	
%precisión en la clase	0	40	72.72	100	75		**93.1**

MLP6_6

Curva de Aprendizaje.

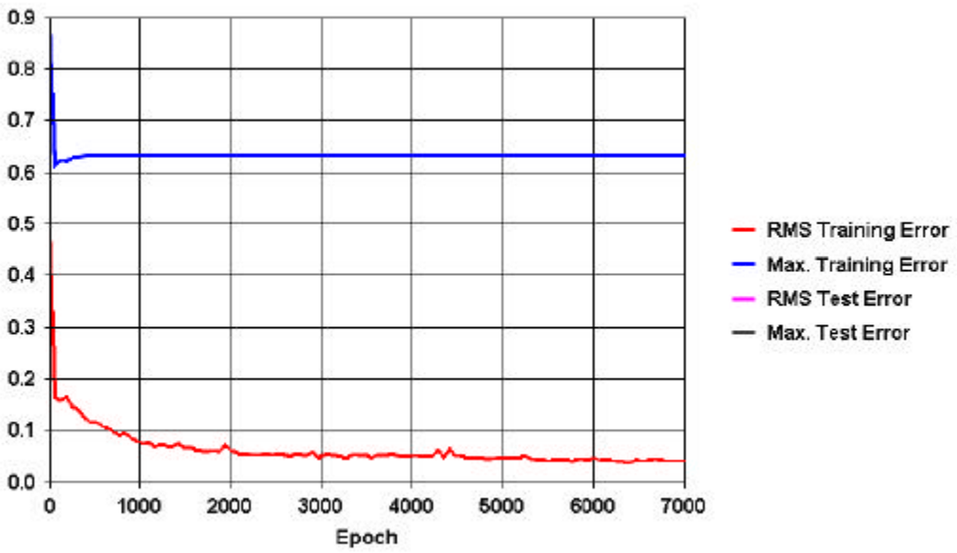


Matriz de Confusión.

	1	2	3	4	5	total	% precisión en la clasificación
1	**1**	0	0	1	0	**2**	50
2	0	**2**	1	0	0	**3**	66.67
3	0	0	**4**	1	0	**5**	80
4	0	3	6	**93**	0	**102**	93.93
5	0	0	0	0	**4**	**4**	100
Total	**1**	**5**	**11**	**95**	**4**	**104/116**	
%precisión en la clase	100	40	36.36	97.9	100		**89.66**

MLP7_7

Curva de Aprendizaje.



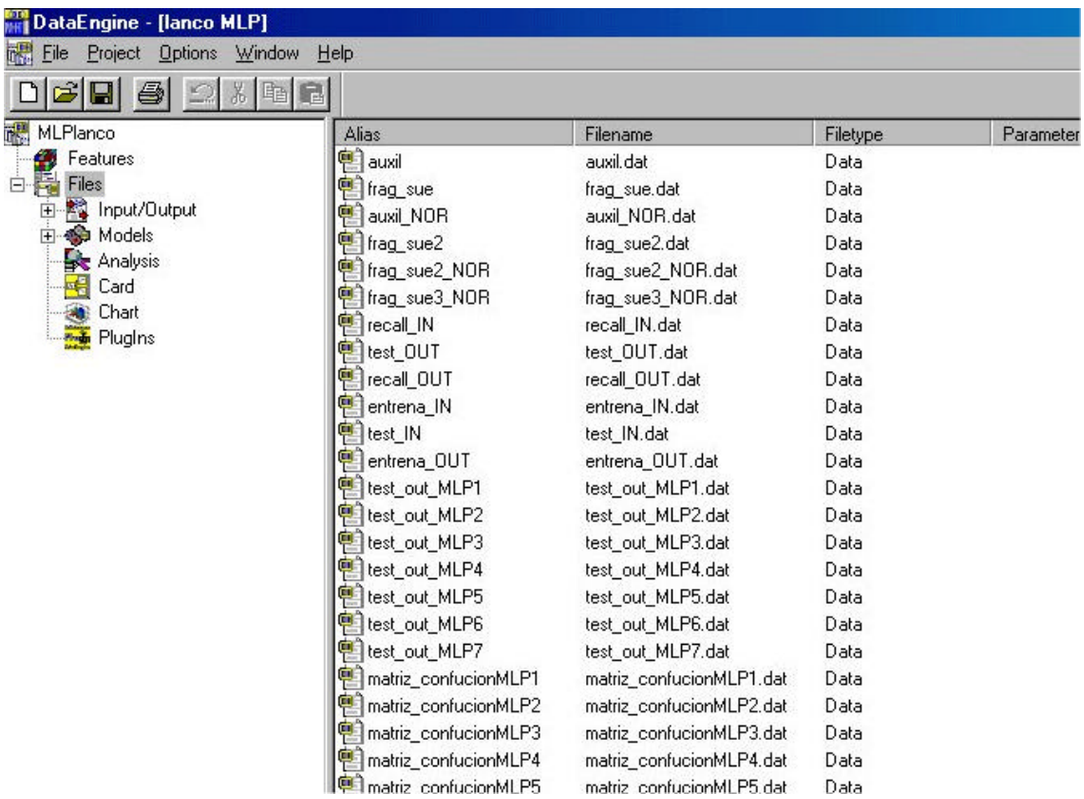
Matriz de Confusión.

	1	2	3	4	5	total	% precisión en la clasificación
1	**0**	0	0	0	0	**0**	0
2	0	**3**	0	0	0	**3**	100
3	0	1	**9**	1	0	**11**	81.81
4	1	1	2	**94**	0	**98**	95.92
5	0	0	0	0	**4**	**4**	100
Total	1	5	11	95	4	**110/116**	
%precisión en la clase	0	60	81.81	98.95	100		**94.83**

8. Capítulo 9: Implementación de la solución usando Data Engine.

Esta sección tiene como objetivo mostrar el proceso de configuración de un modelo de red neuronal multilayer Perceptrón, usando el software para Data Mining Data Engine 4.0. las características generales de este software son expuestas en el anexo nº1: "Software para Data Mining: Data Engine 4.0"

8.1. Interfaz principal.

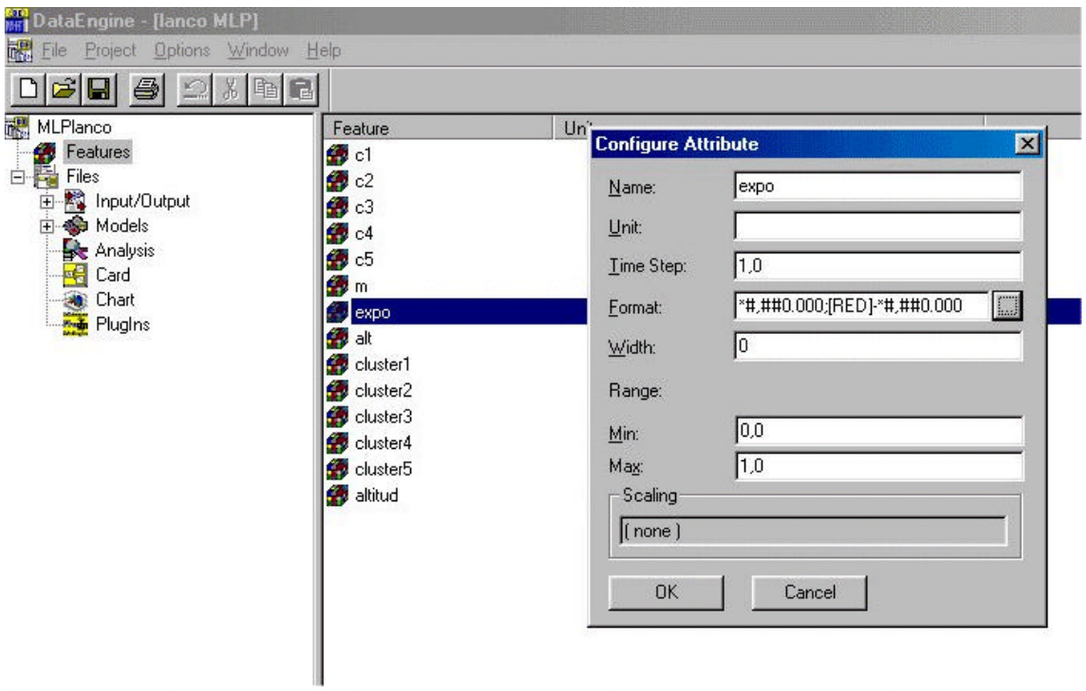


En la imagen superior se puede apreciar la pantalla principal de un proyecto desarrollado en Data Engine. En este caso, el proyecto lleva por nombre **MLPLanco**.

La estructura de árbol que se aprecia en la parte izquierda de la pantalla permite distinguir los elementos que conforman un proyecto: las características o variables involucradas en el estudio (**features**), los archivos de entrada y salida para los diferentes

modelos (**Files)**, y los distintos modelos utilizables (varios tipos de redes neuronales, árboles de decisión, modelos fuzzy, etc.).

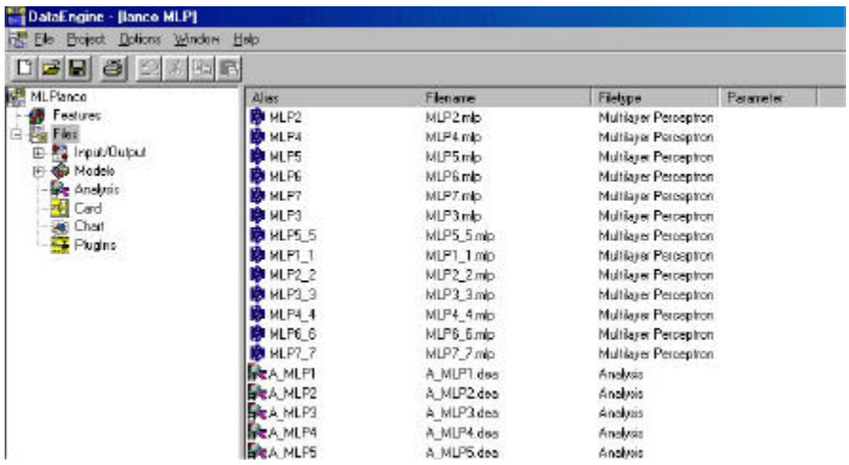
8.2. Configuración de características.



Como se puede apreciar en la imagen, se entiende por características a las variables de entrada y salida del modelo utilizado. En este caso son las variables de entrada y salida de una red neuronal de arquitectura multilayer perceptrón. Las características son declaradas a nivel global, lo que quiere decir que podemos tener diversos modelos en nuestro proyecto que hagan uso de subconjuntos de características de entrada y salida.

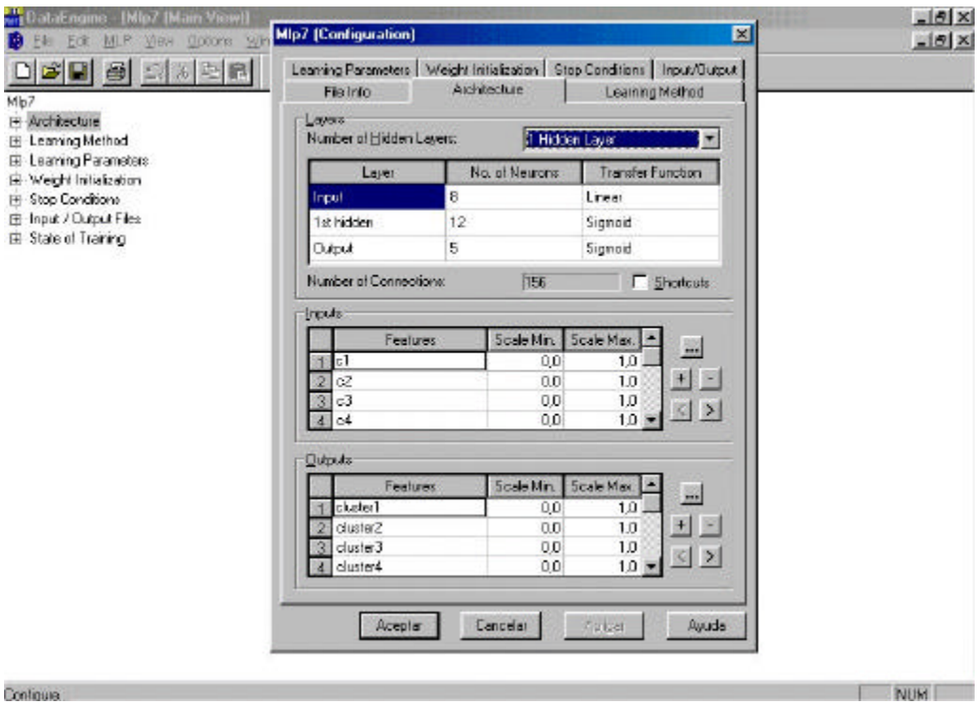
El cuadro **Configure attribute**, muestra los detalles de nombre, formato y rango de la variable de entrada exposición.

8.3. Declaración de modelos.



En el proyecto MLPLanco, se desarrollaron un conjunto de redes multilayer perceptrón que presentan diversas configuraciones de parámetros y arquitectónicas, como se aprecia en la imagen. Además contamos con objetos de análisis asociados a cada una de estas redes. Estos últimos tienen una serie de funcionalidades. Entre éstas se encuentra el cálculo de la matriz de confusión, y medidas de error.

8.4. Configuración de la red MLP7

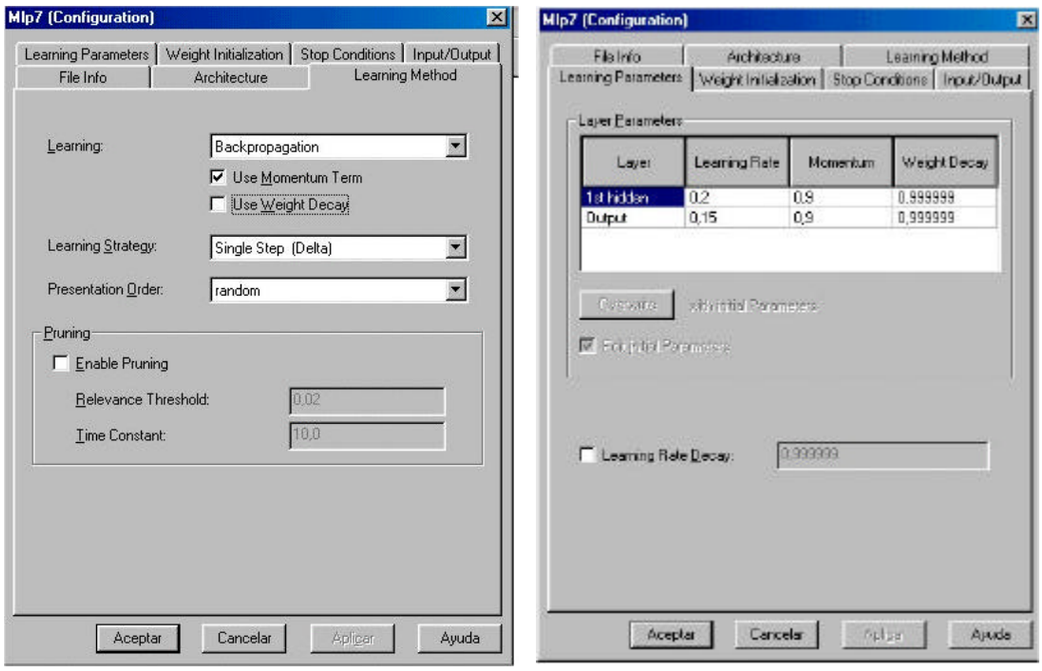


Esta interfaz permite configurar los diversos aspectos asociados a una red neuronal del tipo Multilayer Perceptrón. En esta sección tomaremos como ejemplo el caso de MLP7, que se extiende a todos los otros casos del proyecto MLPLanco.

La interfaz activa muestra la configuración de **arquitectura** del modelo. Se puede indicar el número de capas escondidas que deseamos para nuestro modelo (**hydden layers**), y el número de neuronas por capa. Los valores para las capas de entrada y salida se setean de acuerdo al número de **Inputs** y **Outputs** que consideremos en nuestra red. Recordemos que estos últimos corresponden a **features,** declaradas de manera global en el proyecto.

También se indica aquí cuales son las funciones de transferencia escogidas entre capa. Data Engine permite escoger el tipo de función de transferencia de las capas escondidas y de la capa de salida. La capa de entrada utiliza una función de activación del tipo lineal.

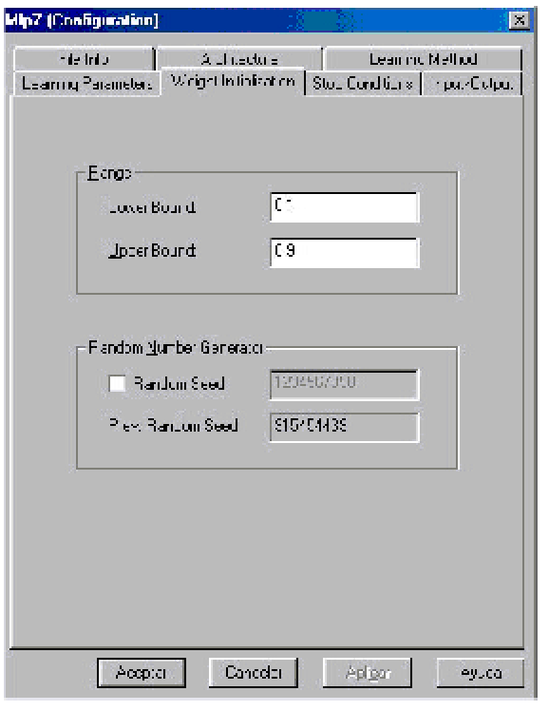
8.5. Método y parámetros de aprendizaje.

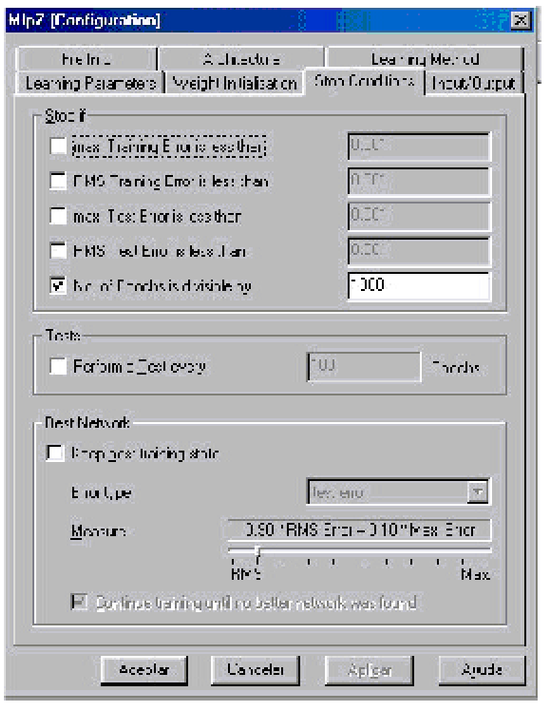


Learning method: Permite configurar los aspectos del algoritmo de aprendizaje utilizado en el proyecto. Este modelo fue entrenado con el algoritmo **Backpropagation,** haciendo uso del termino de **momentum,** para el ajuste de los pesos neuronales. El ajuste de los pesos va ocurriendo cada vez que se presenta un ejemplo distinto lo que se indica en **Learninf Strategy: Single Step (delta).** Los ejemplos son presentados a la red aleatoriamente (**Presentation Order: random).** No se utilizó la capacidad de podar las conexiones neuronales que caen bajo un umbral de poda, valor configurado como 0.02.

Learning Parameters: Aquí se indican los valores de la Tasa de aprendizaje α, momentum m, y weight-decay d, para la capa escondida y la capa de salida respectivamente.

8.6. Iniciación de los pesos neuronales y condiciones de parada para el entrenamiento.

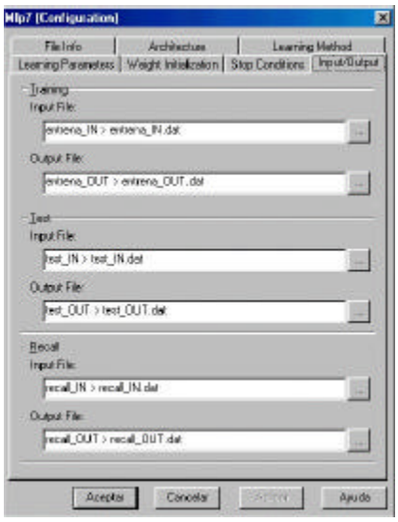




Weight Initialization: Permite indicar los valores máximos y mínimos definidos para el ajuste de los pesos neuronales. En **Random Number Generator** se configura el valor de la semilla utilizada en la generación aleatoria de los pesos neuronales de la red, cada vez que ocurre un nuevo entrenamiento (**New Training**).

Stop Conditions: Se puede escoger entre diversos criterios de parada para la fase de entrenamiento de la red. En este caso se utilizó un número de épocas divisibles por 1000: cada 1000 épocas (1000 veces que se presentan todos los ejemplos de entrenamiento), éste termina. Como se ha indicado anteriormente, se utilizó un New training, donde los pesos neuronales fueron generados aleatoriamente en base a la semilla, y 6 entrenamientos consecutivos, es decir 7000 épocas para entrenar la red.

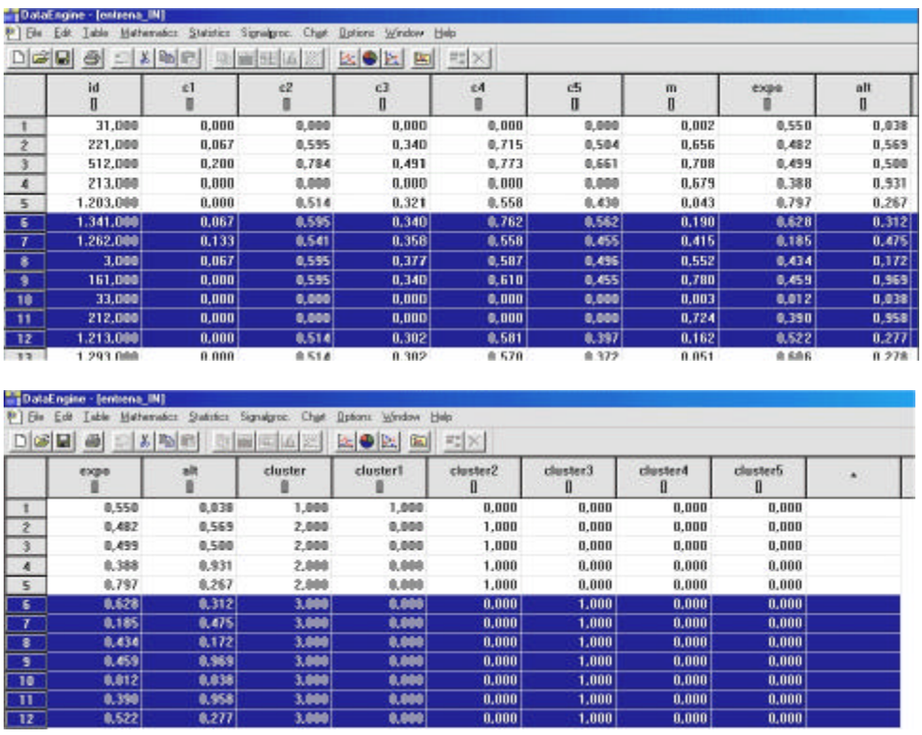
8.7. Configuración de Archivos de Entrada y Salida.



Se indican cuales son los archivos de entrada y salida para cada una de las fases por las que atraviesa la red neuronal.

8.8. Archivos de entrenamiento.

En la fase de entrenamiento (**training**) se indican los ejemplos que serán utilizados para entrenar la red (**input**). Estos ejemplos constan de las variables de entrada y sus respectivos atributos, y las variables de salida, representadas por las clases que queremos identificar en la clasificación de los registros. Este vector de salida binario indica la clase a la que pertenece el ejemplo: si tenemos un registro que pertenece a la clase 2 (**cluster2**) de las 5 que existen, se presenta a la red como (0,1,0,0,0). La figura siguiente muestra el aspecto de un grupo de registros o ejemplos que pertenecen a la clase 3 (**cluster3**). Recordemos que los datos de test_IN provienen de la tabla Frag_suelo. Estos datos han sido normalizados, información que conforma la entrada del entrenamiento.

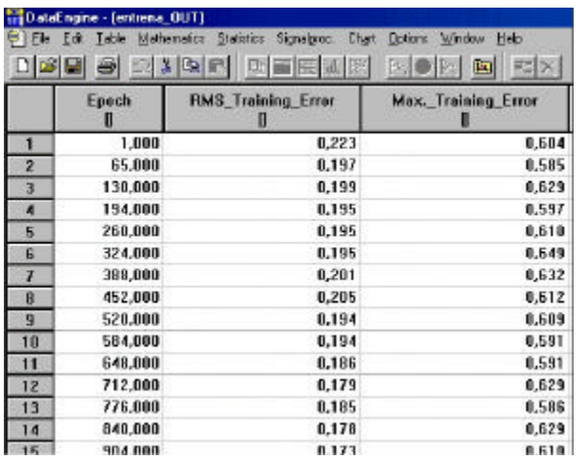


	id	c1	c2	c3	c4	c5	m	expo	alt
1	31,000	0,000	0,000	0,000	0,000	0,000	0,002	0,550	0,038
2	221,000	0,067	0,595	0,340	0,715	0,504	0,656	0,482	0,569
3	512,000	0,200	0,784	0,491	0,773	0,661	0,708	0,499	0,500
4	213,000	0,000	0,000	0,000	0,000	0,000	0,679	0,388	0,931
5	1.203,000	0,000	0,514	0,321	0,558	0,430	0,043	0,797	0,267
6	1.341,000	0,067	0,595	0,340	0,762	0,562	0,190	0,628	0,312
7	1.262,000	0,133	0,541	0,358	0,558	0,455	0,415	0,185	0,475
8	3,000	0,067	0,595	0,377	0,587	0,496	0,552	0,434	0,172
9	161,000	0,000	0,595	0,340	0,610	0,455	0,780	0,459	0,969
10	33,000	0,000	0,000	0,000	0,000	0,000	0,003	0,012	0,038
11	212,000	0,000	0,000	0,000	0,000	0,000	0,724	0,390	0,958
12	1.213,000	0,000	0,514	0,302	0,581	0,397	0,162	0,522	0,277

	expo	alt	cluster	cluster1	cluster2	cluster3	cluster4	cluster5
1	0,550	0,038	1,000	1,000	0,000	0,000	0,000	0,000
2	0,482	0,569	2,000	0,000	1,000	0,000	0,000	0,000
3	0,499	0,500	2,000	0,000	1,000	0,000	0,000	0,000
4	0,388	0,931	2,000	0,000	1,000	0,000	0,000	0,000
5	0,797	0,267	2,000	0,000	1,000	0,000	0,000	0,000
6	0,628	0,312	3,000	0,000	0,000	1,000	0,000	0,000
7	0,185	0,475	3,000	0,000	0,000	1,000	0,000	0,000
8	0,434	0,172	3,000	0,000	0,000	1,000	0,000	0,000
9	0,459	0,969	3,000	0,000	0,000	1,000	0,000	0,000
10	0,012	0,038	3,000	0,000	0,000	1,000	0,000	0,000
11	0,390	0,958	3,000	0,000	0,000	1,000	0,000	0,000
12	0,522	0,277	3,000	0,000	0,000	1,000	0,000	0,000

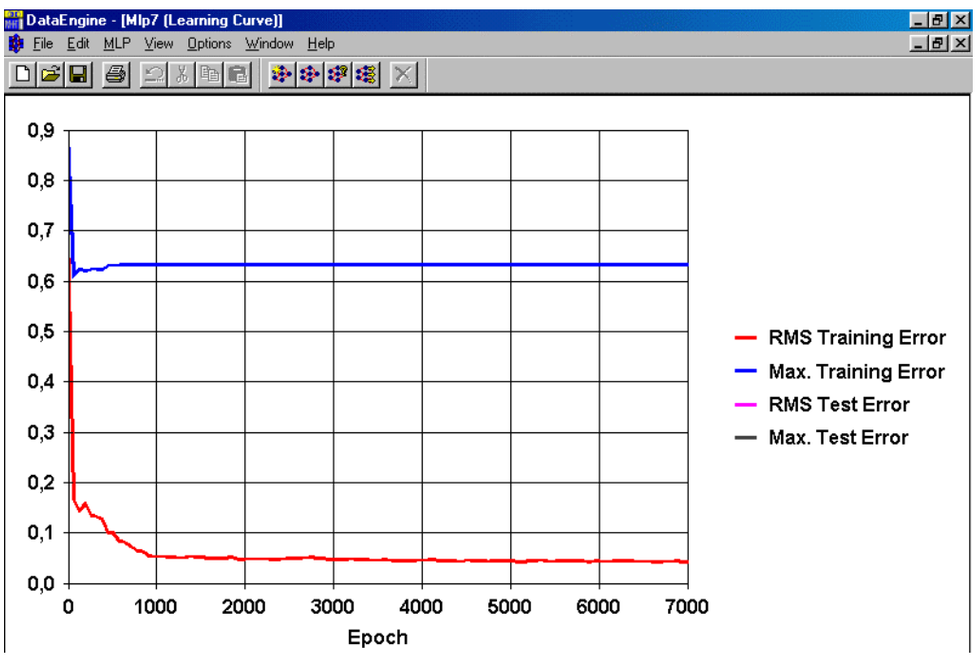
Antes de entrenar la red, como es lógico, el archivo de salida de la fase de entrenamiento se encuentra en bLanco. Después de que el entrenamiento ocurre, este archivo contiene la información para poder graficar la curva de aprendizaje de la red. Estos valores corresponden al cálculo del **RMS training Error** y **Máx. RMS training Error.** Si se mantiene abierta la vista curva de aprendizaje durante el entrenamiento, es posible apreciar como se va comportando la red durante su aprendizaje.

La fase de entrenamiento no entrega un resultado en el sentido literal de la palabra, puesto que el objetivo de esta fase es setear los pesos de las conexiones neuronales. El archivo de salida de esta fase contiene, como se dijo anteriormente, **RMS training Error** y **Máx. training Error.** El valor del **RMS training Error** se calcula en este caso sobre todas las salidas neuronales de la época de aprendizaje considerada, como se muestra en la tabla, mientras que **Máx. training Error** es el máximo valor de las diferencias entre la salida deseada y la obtenida. De todas formas, la definición del error RMS se verá en la próxima sección.


DataEngine - [entrena_OUT]

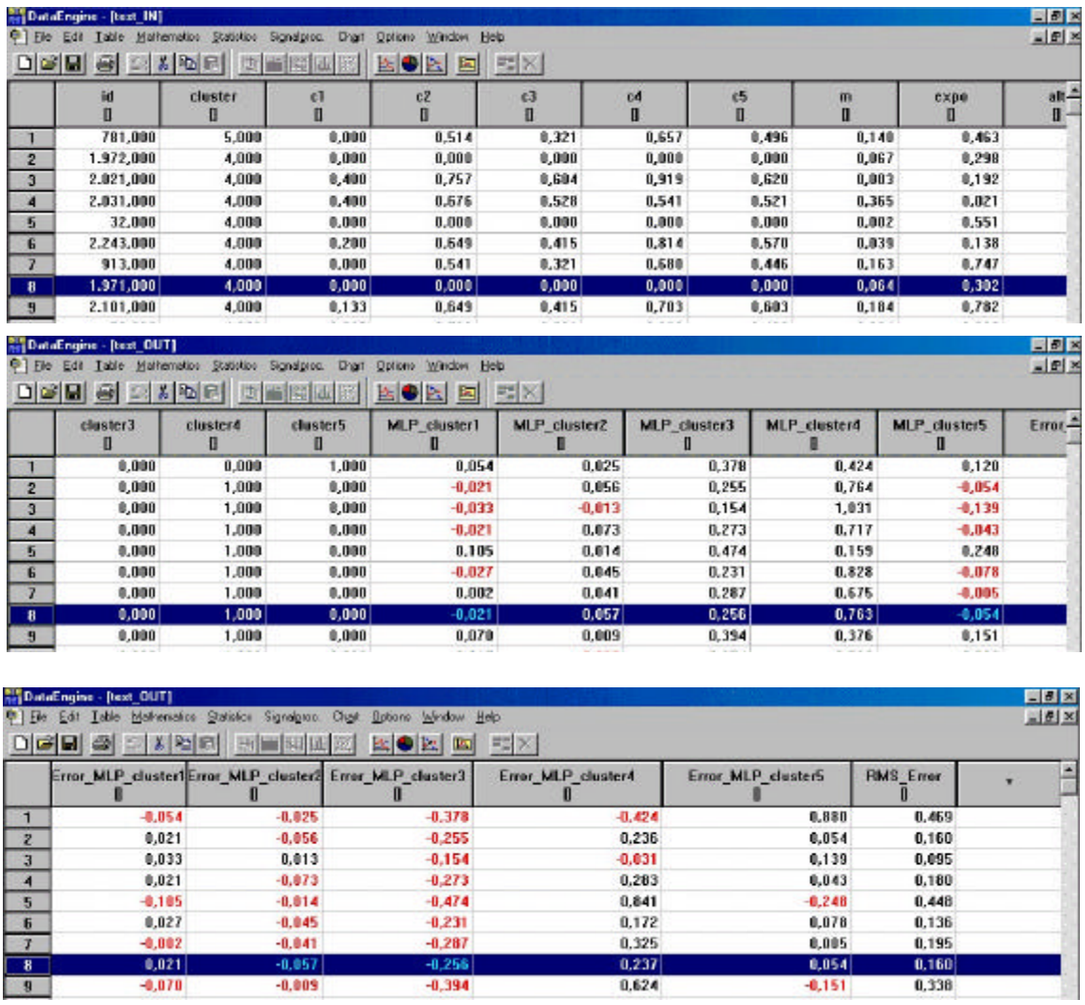
File Edit Table Mathematics Statistics Signalproc Chart Options Window Help

	Epoch []	RMS_Training_Error []	Max._Training_Error []
1	1,000	0,223	0,604
2	65,000	0,197	0,585
3	130,000	0,199	0,629
4	194,000	0,195	0,597
5	260,000	0,195	0,610
6	324,000	0,195	0,649
7	388,000	0,201	0,632
8	452,000	0,205	0,612
9	520,000	0,194	0,609
10	584,000	0,194	0,591
11	648,000	0,186	0,591
12	712,000	0,179	0,629
13	776,000	0,185	0,586
14	840,000	0,178	0,629
15	904,000	0,173	0,610



8.9. **Archivos de test.**

El archivo de test **Test_IN** fue utilizado para medir el desempeño de la red y conocer la calidad de la clasificación realizada por esta. **Test_OUT**, representa el archivo de salida. Podemos observar que este es el mismo **test_IN** al que se le agregan valores para ilustrar el desempeño.



DataEngine - [test_IN]

	id	cluster	c1	c2	c3	c4	c5	m	expo	alf
1	781,000	5,000	0,000	0,514	0,321	0,657	0,496	0,140	0,463	
2	1.972,000	4,000	0,000	0,000	0,000	0,000	0,000	0,067	0,298	
3	2.021,000	4,000	0,400	0,757	0,604	0,919	0,620	0,003	0,192	
4	2.031,000	4,000	0,400	0,676	0,528	0,541	0,521	0,365	0,021	
5	32,000	4,000	0,000	0,000	0,000	0,000	0,000	0,002	0,551	
6	2.243,000	4,000	0,200	0,649	0,415	0,814	0,570	0,039	0,138	
7	913,000	4,000	0,000	0,541	0,321	0,680	0,446	0,163	0,747	
8	1.971,000	4,000	0,000	0,000	0,000	0,000	0,000	0,064	0,302	
9	2.101,000	4,000	0,133	0,649	0,415	0,703	0,603	0,184	0,782	

DataEngine - [test_OUT]

	cluster3	cluster4	cluster5	MLP_cluster1	MLP_cluster2	MLP_cluster3	MLP_cluster4	MLP_cluster5	Error
1	0,000	0,000	1,000	0,054	0,025	0,378	0,424	0,120	
2	0,000	1,000	0,000	-0,021	0,056	0,255	0,764	-0,054	
3	0,000	1,000	0,000	-0,033	-0,013	0,154	1,031	-0,139	
4	0,000	1,000	0,000	-0,021	0,073	0,273	0,717	-0,043	
5	0,000	1,000	0,000	0,105	0,014	0,474	0,159	0,248	
6	0,000	1,000	0,000	-0,027	0,045	0,231	0,828	-0,078	
7	0,000	1,000	0,000	0,002	0,041	0,287	0,675	-0,005	
8	0,000	1,000	0,000	-0,021	0,057	0,256	0,763	-0,054	
9	0,000	1,000	0,000	0,070	0,009	0,394	0,376	0,151	

DataEngine - [test_OUT]

	Error_MLP_cluster1	Error_MLP_cluster2	Error_MLP_cluster3	Error_MLP_cluster4	Error_MLP_cluster5	RMS_Error
1	-0,054	-0,025	-0,378	-0,424	0,880	0,469
2	0,021	-0,056	-0,255	0,236	0,054	0,160
3	0,033	0,013	-0,154	-0,031	0,139	0,095
4	0,021	-0,073	-0,273	0,283	0,043	0,180
5	-0,105	-0,014	-0,474	0,841	-0,248	0,448
6	0,027	-0,045	-0,231	0,172	0,078	0,136
7	-0,002	-0,041	-0,287	0,325	0,005	0,195
8	0,021	-0,057	-0,256	0,237	0,054	0,160
9	-0,070	-0,009	-0,394	0,624	-0,151	0,338

En la figura podemos ver el registro nº8, que pertenece a la clase 4 (**cluster4**). Los atributos de la tabla que han sido agregados producto de la salida son MLP_cluster1 a MLP_cluster7, y éstos representan la ponderación asignada por las neuronas de salida para cada ejemplo de entrada. Puntualmente, el ejemplo del registro nº8 tiene su mayor ponderación en MLP_cluster4 (0.763), por lo que podemos decir que la red clasificó perfectamente ese ejemplo. Los siguientes atributos, ERROR_ MLP_cluster1 a ERROR_ MLP_cluster1 representa la diferencia entre la salida deseada para cada clase y

la salida real obtenida. Por ultimo, el valor de **RMS_error** (Root mean square error) se calcula como:

$$RMS = \sqrt{\frac{1}{Numero\_de\_clases}\left(valor\_actual\_clase_i - valor\_esperado\_clase_i\right)^2} =$$

$$\sqrt{\frac{1}{5}\left(MLP\_cluster_i - cluster_i\right)^2} = \sqrt{\frac{1}{5}\left(Error\_MLP\_cluster_i\right)^2}$$

tomando como ejemplo el registro nº8, tenemos:

$$\sqrt{\frac{1}{5}\left[(0.021)^2 + (-0.057)^2 + (-0.256)^2 + (0.237)^2 + (0.054)^2\right]} = \sqrt{\frac{0.128311}{5}} = 0.16019426$$

8.10. **Archivos de Recall.**

El archivo que contiene los registros que queremos clasificar, Recall_IN, contiene solamente los valores de las variables de entrada a la red, como se aprecia en la siguiente figura:

DataEngine - [recall_IN]

File Edit Table Mathematics Statistics Signalproc. Chart Options Window Help

	c1	c2	c3	c4	c5	m	expo	alt
1	0,067	1,000	0,717	0,081	0,810	0,000	0,821	0,000
2	0,000	0,541	0,358	0,692	0,562	0,094	0,647	0,312
3	0,067	0,568	0,340	0,657	0,471	0,262	1,000	0,324
4	0,067	0,595	0,396	0,622	0,455	0,000	0,861	0,000
5	0,267	0,784	0,528	0,791	0,727	0,055	0,581	0,321
6	0,333	0,757	0,528	0,721	0,678	0,635	0,333	0,882
7	0,333	0,757	0,528	0,837	0,777	0,618	0,335	0,868
8	0,067	0,649	0,377	0,837	0,612	0,737	0,483	0,390
9	0,000	0,568	0,340	0,738	0,554	0,227	0,485	0,279
10	0,000	0,514	0,283	0,599	0,446	0,087	0,391	0,301
11	0,000	0,000	0,000	0,000	0,000	0,512	0,480	0,221
12	0,067	0,595	0,340	0,767	0,521	0,156	0,491	0,271
13	0,067	0,514	0,283	0,477	0,388	0,502	0,055	0,199
14	0,067	0,541	0,321	0,552	0,413	0,465	0,021	0,194
15	0,000	0,486	0,283	0,471	0,380	0,424	0,023	0,206

Recordemos que la clasificación ha sido hecha sobre los datos contenidos en la tabla Auxil. En este caso, recall_IN contiene dichos datos normalizados. La salida de la clasificación se encuentra en el archivo recall_OUT.

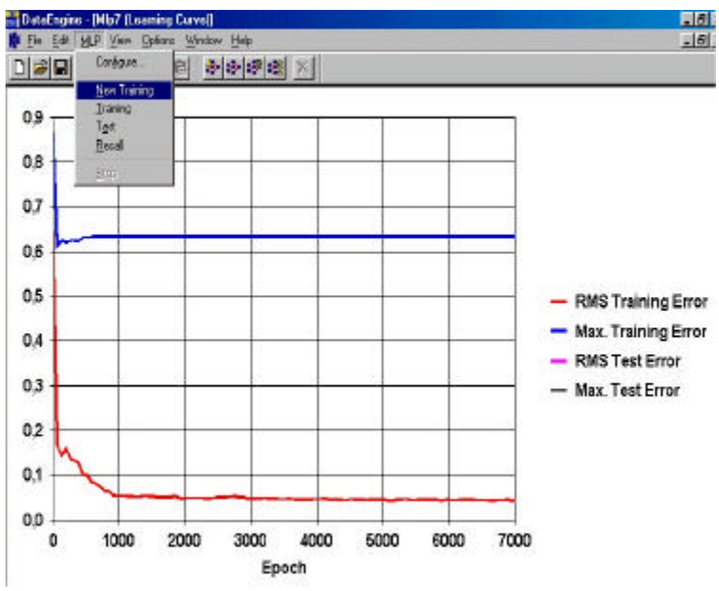
El archivo recall_OUT, agrega a los datos de entrada de la clasificación, los atributos de salida (cluster1 a cluster5).En este caso, el registro Nº7 de la tabla Auxil, por ejemplo, fue clasificado por la red como Cluster4, valor con mayor ponderación.

DataEngine - [recall_OUT]

File Edit Table Mathematics Statistics Signalproc Chart Options Window Help

	m	expo	alt	cluster1	cluster2	cluster3	cluster4	cluster5	.
1	0,000	0,821	0,000	0,192	-0,026	-0,027	0,890	-0,029	
2	0,094	0,647	0,312	0,171	-0,002	0,076	0,745	0,009	
3	0,262	1,000	0,324	0,184	-0,019	-0,003	0,861	-0,023	
4	0,000	0,861	0,000	0,189	-0,023	-0,018	0,879	-0,027	
5	0,055	0,581	0,321	0,189	-0,022	-0,014	0,872	-0,025	
6	0,635	0,333	0,882	0,135	0,039	0,249	0,502	0,074	
7	0,618	0,335	0,868	0,123	0,052	0,301	0,433	0,092	
8	0,737	0,483	0,390	0,111	-0,000	-0,051	1,037	-0,096	
9	0,227	0,485	0,279	0,134	0,033	0,209	0,572	0,053	
10	0,087	0,991	0,301	0,189	-0,024	-0,021	0,883	-0,028	
11	0,512	0,480	0,221	0,114	0,039	0,199	0,614	0,034	
12	0,156	0,491	0,271	0,163	-0,002	0,057	0,787	-0,006	
13	0,502	0,055	0,199	0,109	0,020	0,070	0,837	-0,035	
14	0,465	0,821	0,194	0,109	0,020	0,072	0,834	-0,035	

El proceso de entrenamiento, test y recall es dirigido por el usuario seleccionando la opción correspondiente en el menú **MLP**. En este caso, se han seguido los pasos del proceso teniendo activa la vista de la curva de aprendizaje **(vista Learning Curve)**, para observar el comportamiento de la red. El software también permite hacer esto observando los pesos de las neuronas como se muestra en la siguiente figura **(vista connections)**.



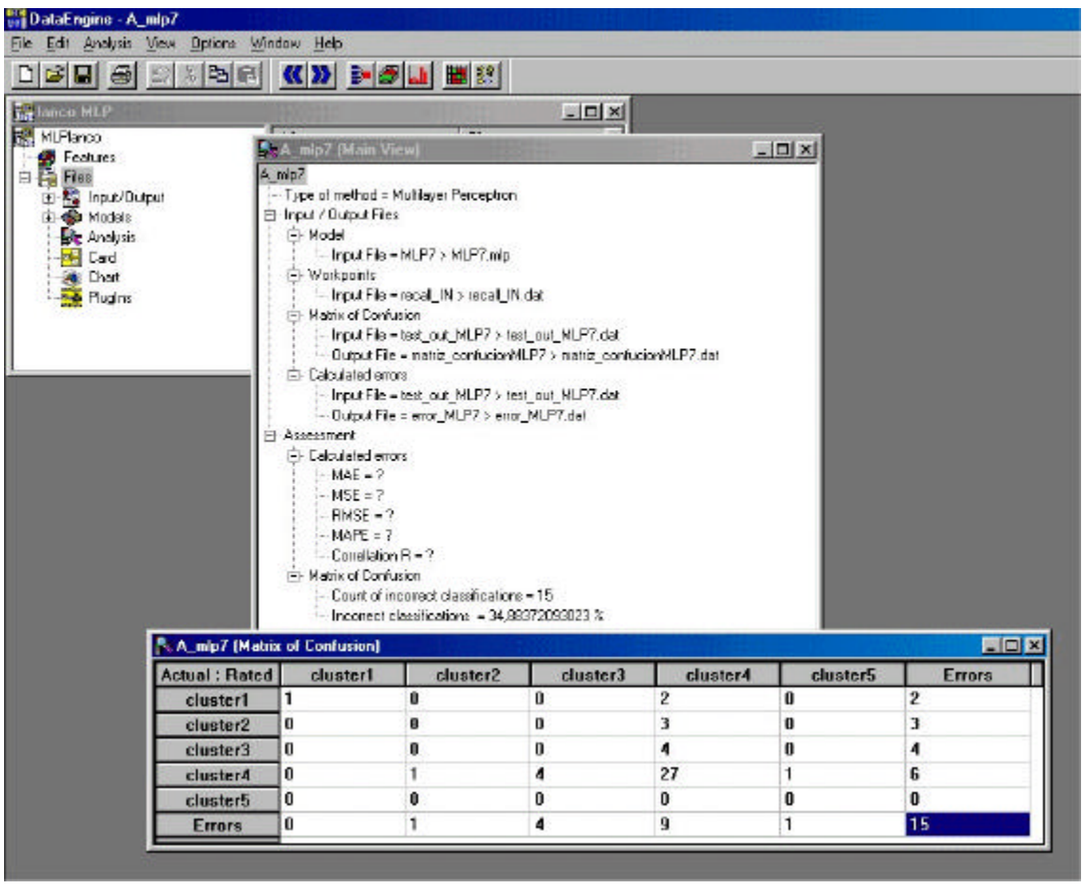
DataEngine - [Mlp7 (Connections)]

File Edit MLP Connections View Options Window Help

	From Neuron	To Neuron	Weight	Delta Weight	Relevance
30	c2	1st hidden 1	-6,305066	-0,0000001100	1,000000
31	c2	1st hidden 2	4,776711	0,0001323583	1,000000
32	c2	1st hidden 3	1,348850	0,0002002586	1,000000
33	c2	1st hidden 4	-7,719365	0,0003205203	1,000000
34	c2	1st hidden 5	2,896178	-0,0000014061	1,000000
35	c2	1st hidden 6	2,153397	0,0000156833	1,000000
36	c2	1st hidden 7	0,878252	0,0000331659	1,000000
37	c2	1st hidden 8	-2,279774	0,0011462319	1,000000
38	c2	1st hidden 9	0,160971	0,0000107395	1,000000
39	c2	1st hidden 10	11,313101	0,0003367559	1,000000
40	c2	1st hidden 11	-2,496092	0,0006669405	1,000000
41	c2	1st hidden 12	-3,587943	0,0001835089	1,000000
42	c3	1st hidden 1	-3,498084	-0,0000000636	1,000000
43	c3	1st hidden 2	7,784949	0,0000762057	1,000000
44	c3	1st hidden 3	3,740326	0,0001144562	1,000000

8.11. Análisis del modelo.

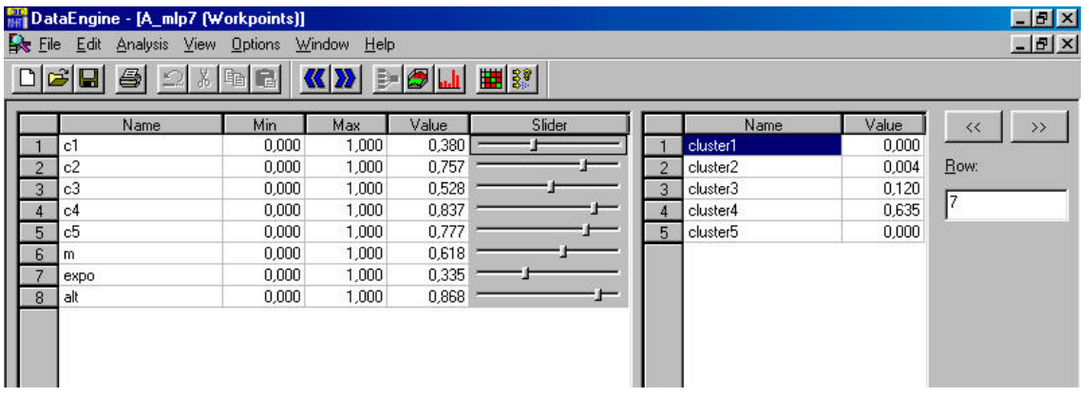
La siguiente imagen muestra algunas interfaces disponibles para un objeto de análisis. En este caso, se han creado tantos análisis como modelos de redes presentes en el experimento. Cada análisis se encuentra asociado a un objeto de modelo, y éste se alimenta de los archivos de entrada y salida del mismo. En nuestro caso, podemos observar que con los datos contenidos en el test_OUT de MLP7 se ha construido una matriz de confusión que da cuenta de la calidad de la clasificación hecha por la red clase por clase.



Actual : Rated	cluster1	cluster2	cluster3	cluster4	cluster5	Errors
cluster1	1	0	0	2	0	2
cluster2	0	0	0	3	0	3
cluster3	0	0	0	4	0	4
cluster4	0	1	4	27	1	6
cluster5	0	0	0	0	0	0
Errors	0	1	4	9	1	15

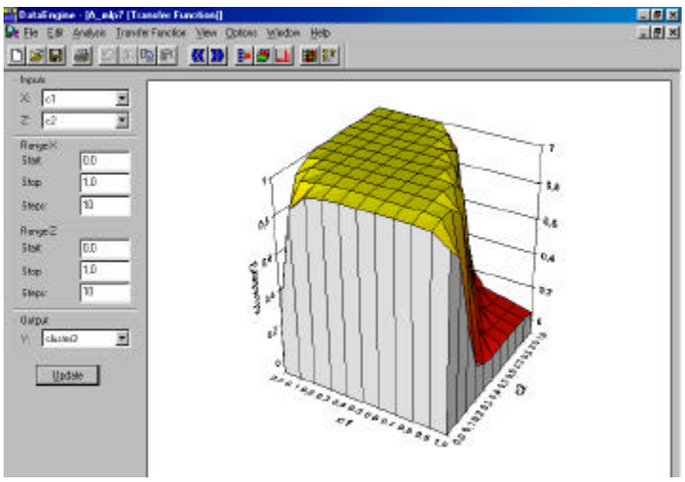
En un objeto de análisis se encuentran disponibles distintas formas de análisis: se puede estudiar la tabla de errores, hacer un análisis de sensibilidad sobre el modelo y por último estudiar el comportamiento de la fusión de transferencia.

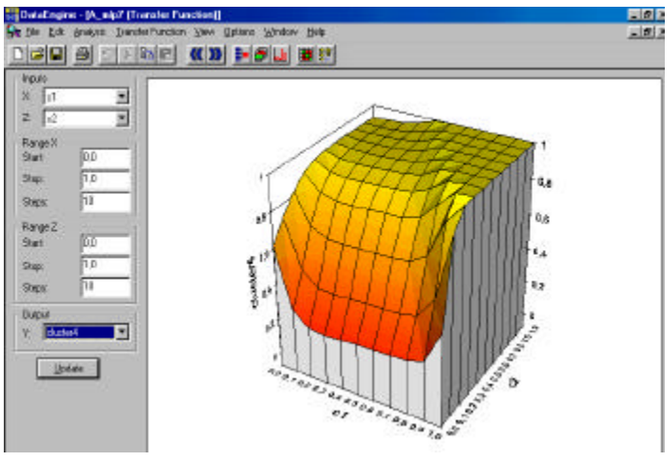
La interfaz de **workpoints** permite determinar como se comporta una salida deseada para un registro dado, cuando hacemos variar el valor de una variable de entrada de la red, registro a registro, como se aprecia en la siguiente figura:



Para poder realizar estos análisis, es necesario entrenar y testear la red, por motivos obvios. En este caso, este análisis posee con archivo de entrada los datos del recall. Al mover el control de **Slider** se pueden observar directamente el cambio del vector de salida. En la figura se está alterando el valor de la variable de entrada c1 de la fila nº7 de la tabla, y se pueden observar los cambios de cluster1 a cluster5.

Esta interfaz permite observar gráficamente la salida generada por dos variables de entrada sobre una de las clases, registro a registro. En este caso, se han considerado como entradas a c1 y c2 y como variable de salida a cluster3 (imagen superior) y cluster4 (imagen inferior), sobre un mismo registro.





9. Capítulo 10: Elección de la mejor solución, resultados finales y Conclusiones.

En este capítulo, se expondrá la elección de la mejor solución al problema encontrada mediante la reformulación de la solución original, usando redes bayesianas.

También se expondrá la elección de la mejor red neuronal junto con un análisis de los resultados globales de cada experimento.

Posteriormente se escogerá la mejor solución al problema basándose en la elección de aquel modelo que presente una mejor calidad en la clasificación, dada por la comparación de matrices de confusión de la mejor red neuronal y la mejor red bayesiana.

9.1. Elección de la mejor red bayesiana.

En el Capítulo 6, **"Modificación de la solución inicial usando redes bayesianas.",** se presenta la creación y las matrices de confusión arrojadas en la fase de test de un conjunto de experimentos (Lanco2, Lanco3, Lanco4, Lanco5, Lanco6 y Lanco7). La tabla siguiente nos da una visión general de los resultados obtenidos en los experimentos anteriormente comentados.

	Lanco2	**Lanco3**	**Lanco4**	**Lanco5**	**Lanco6**	**Lanco7**
Datos para construcción	datos completos disc	t_entrena Miento	t_entrena Miento	datos completos disc	t_entrena miento	datos completos disc
Datos para inferencia	datos completos disc	t_test	t_test	datos completos disc	t_test	datos completos disc
conocimiento experto	no	No	Si	si	si	si
feature selector	if it perform better	if it perform Better	Always	always	if it perform Better	if it perform better
% clasific. Correctas:						
cluster1	0	0	0	0	0	0
cluster2	0	0	100	0	100	0
cluster3	18.2	0	2.5	9.10	0	9.1
cluster4	75.8	33.3	8.3	72.63	8.3	66.3
cluster5	75	0	0	25	0	50
%clasf. Correcta global	**66.38**	**27.9**	**11.62**	**61**	**9.3**	**58.9**

Como era de esperar, las redes con mejor desempeño fueron las redes construidas con los mismos datos donde se hace la clasificación (Lanco2, Lanco5 y Lanco7). De este conjunto, la mejor red fue la que no incluyó conocimiento experto(**Lanco2**). Esta red, además fue la que presentó mayor precisión en la clasificación de la clase con mayor presencia de ejemplos (clase4). La baja calidad de la clasificación en las demás clases se explica por la poca cantidad de ejemplares utilizados durante la fase de construcción (ver tablas datos_completos_dis, t_entrenamiento y t_test).

El segundo grupo de redes, donde se utilizan poblaciones distintas para construcción e inferencia logra su mejor clasificación en **Lanco3**. Esta red es nuevamente de su grupo, la que no utiliza conocimiento experto durante su construcción. También es la que presenta mejor calidad de clasificación en el cluster4.

9.2. Elección de la mejor red Neuronal.

9.2.1. Análisis de la matriz de confusión.

La mejor red neuronal es aquella que posee la mejor relación entre la **calidad del aprendizaje**, dada por la **curva de aprendizaje**, la calidad de la **clasificación**, dada por la **matriz de confusión**. Estos criterios deben aplicarse a aquella red que presente un mejor comportamiento en ambos casos, redes MLP1 a MLP7 y redes MLP1_1 a MLP7_7. Se ha tomado la decisión de hacer estos últimos experimentos, dada la dificultad que presenta la muestra; existen muy pocos ejemplares para las clases cluster1,2 y 5. Por lo tanto las matrices de confusión de estos grupos entregan una visión más clara de la calidad de la clasificación, considerando también que este comportamiento ocurra en el caso real (poblaciones de entrenamiento y test distintas).

	MLP1	MLP2	MLP3	MLP4	MLP5	MLP6	MLP7
%clasif.	**Correctas**	**por clase**					
Cluster1	0	100	100	100	100	100	100
Cluster2	0	0	0	0	0	0	0
Cluster3	0	0	50	0	0	25	0
Cluster4	91.7	55.56	66.7	63.9	61.1	41.67	75
Cluster5	0	100	0	0	100	100	0
%clasif.	**Correctas**	**Global**					
	76.74	51.16	62.8	55.81	55.81	41.86	**65.12**

	MLP1_1	MLP2_2	MLP3_3	MLP4_4	MLP5_5	MLP6_6	MLP7_7
%clasif.	**Correctas**	**por clase**					
Cluster1	0	0	0	0	0	100	0
Cluster2	0	20	0	0	40	40	60
Cluster3	9.1	81.82	45.45	72.72	72.72	36.36	81.81
Cluster4	100	97.9	100	96.84	100	97.9	98.95
Cluster5	0	75	50	75	75	100	100
%clasif.	**Correctas**	**Global**					
	82.76	91.38	96.23	88.8	93.1	89.66	**94.83**

Al observar las matrices de confusión de ambos grupos de redes, podemos notar que la mejor red, dada su capacidad de hacer el **mejor porcentaje de clasificaciones correctas por clase y global**, corresponde a MLP7 (MLP7_7). (Recordemos que se trata de la misma red, desde el punto de vista de la parametrización y arquitectura). En el grupo de experimentos MLP7 (MLP7_7) pareciera ser que la mejor red, desde el punto de vista de la clasificación global es MLP1, lo que es incorrecto: este porcentaje se produce por la precisión alta alcanzada en la clase cluster4. Sucede que esta clase representa al 81% de la población total de la muestra. Por lo tanto esta clase es la que condiciona el porcentaje de precisión global. Veamos como se comporta esta red con más ejemplares por clase (tabla2). Sigue siendo un buen clasificador de cluster4, pero es mal clasificador en las clases restantes. La red MLP7, es claramente la mejor red al presentar la mejor clasificación global y, sobre todo, la mejor calidad de clasificación por clase.

9.2.2. Análisis de la curva de aprendizaje.

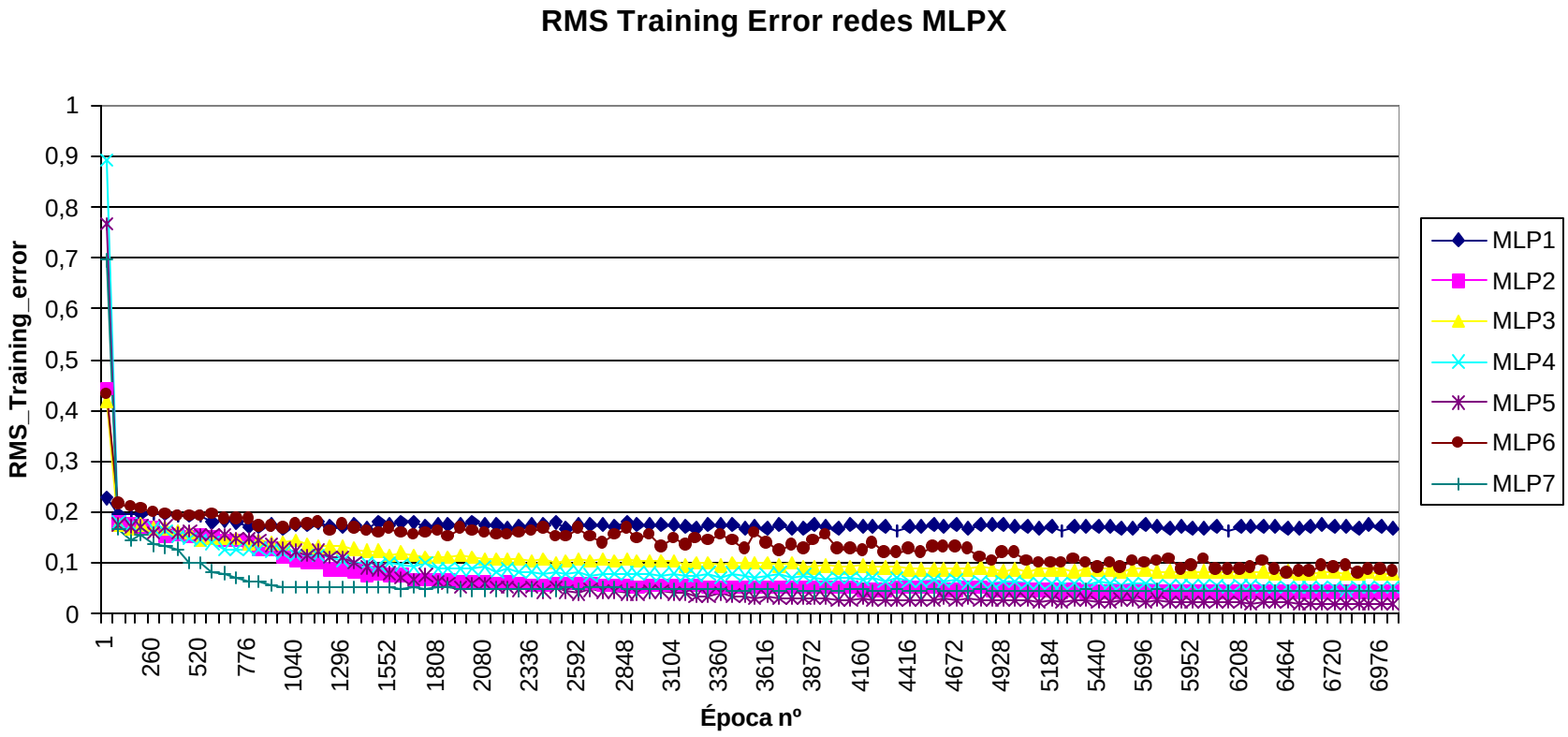
Al analizar las curvas de aprendizaje, elegimos la red cuya curva RMStest error presente una mejor convergencia, es decir la que se establezca constante en el menor valor, comparativamente con las otras redes. Este comportamiento se provoca por la interacción entre la tasa de aprendizaje, momentum y weight decay.

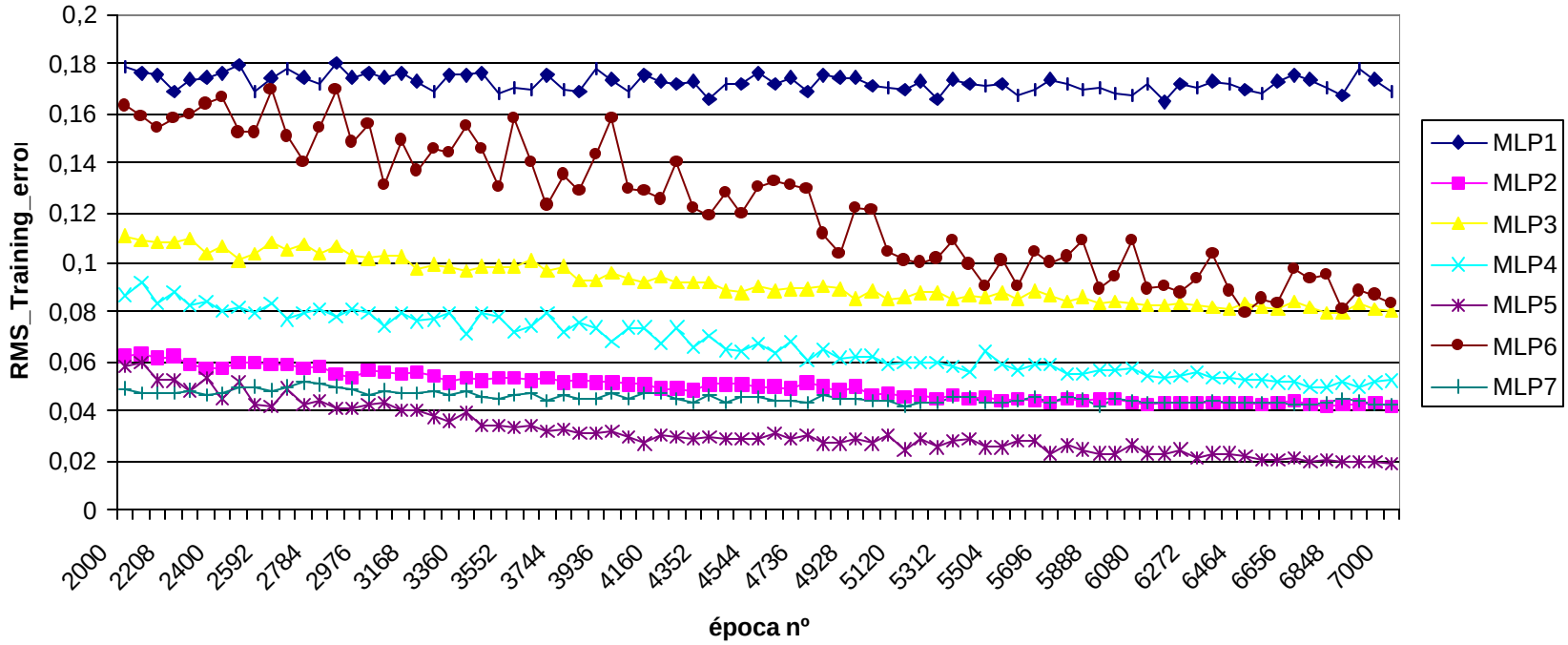
Los siguientes gráficos comparan las curvas de RMS_Training error y Maximo_training_error para cada grupo de redes. Estos gráficos permiten ponderar las redes desde la perspectiva del aprendizaje durante el entrenamiento. En este sentido, serán mejores las redes que presenten curvas formadas por valores más pequeños. La siguiente tabla es un ranking de las redes en ambos casos:

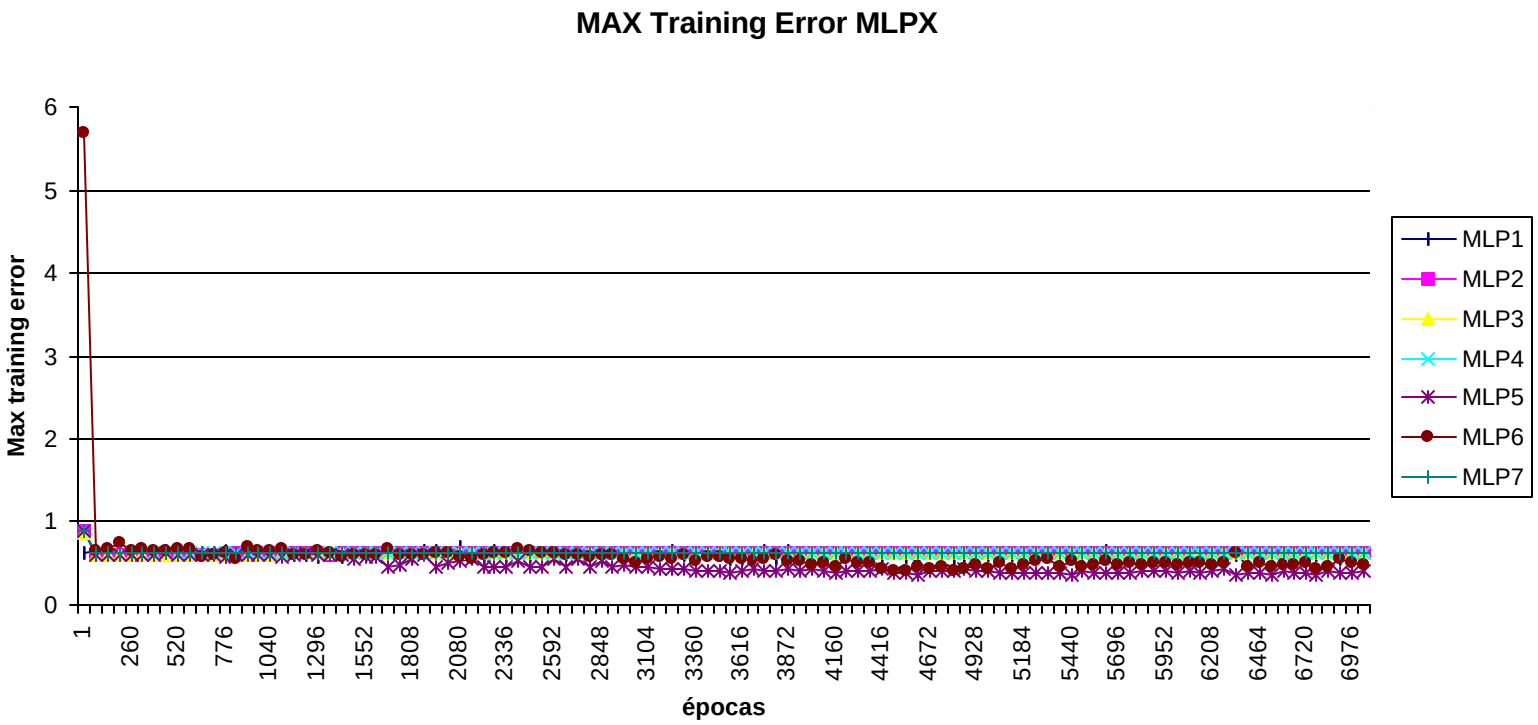
	Max training Error		**RMS Training Er ror**	
Lugar	**Mejor red (MLPX)**	**Mejor red (MLPX_X)**	**Mejor red (MLPX)**	**Mejor red (MLPX_X)**
1	MLP5	MLP1_1	MLP5	MLP7_7
2	MLP6	MLP4_4	MLP7	MLP5_5
3	MLP1	MLP6_6	MLP2	MLP2_2
4	MLP3	MLP5_5	MLP4	MLP4_4
5	MLP2	MLP2_2	MLP3	MLP3_3
6	MLP7	MLP7_7	MLP6	MLP6_6
7	MLP4	MLP3_3	MLP1	MLP1_1

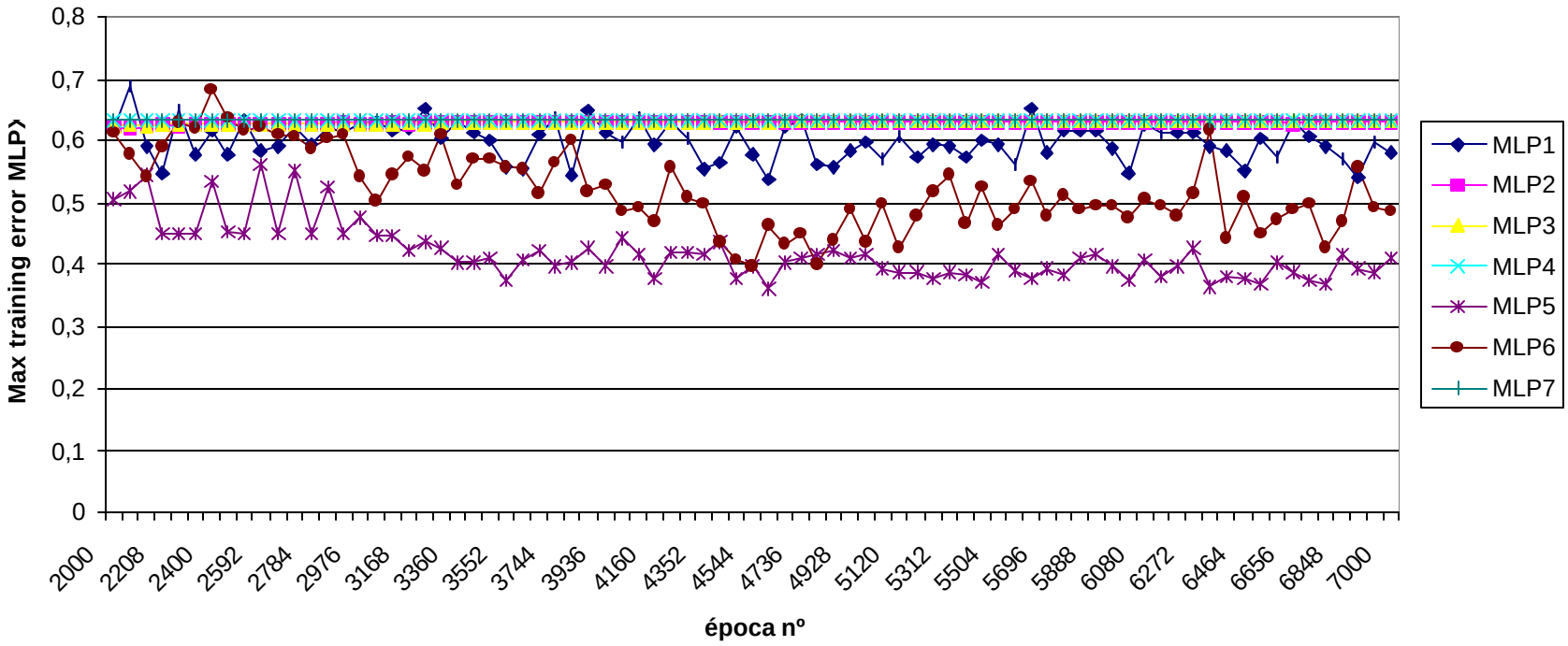
El RMS Training Error es el error cuadrático medio ocurrido durante el aprendizaje y se calcula en este caso sobre todas las salidas neuronales de la época de aprendizaje considerada, mientras que el Máx. training error es el máximo valor de las diferencias entre la salida deseada y la obtenida para una época dada. La tabla nos indica que la red MLP7 (y su versión MLP7_7) es una de las redes con mejor aprendizaje promedio, a pesar de ser una de las que presenta un más alto error de entrenamiento.

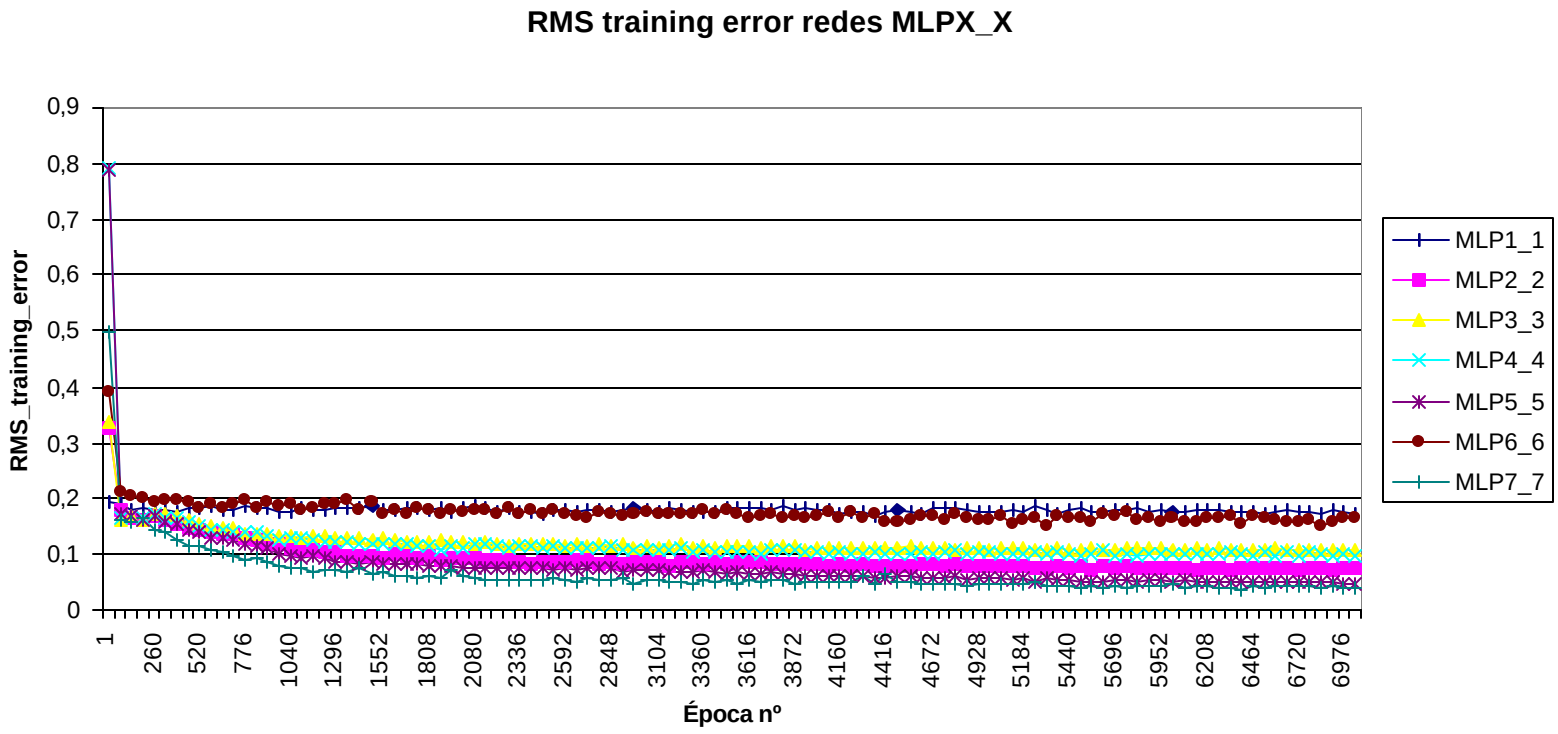
Al contar con esta información y considerando la calidad de la clasificación, podemos concluir que la mejor red es MLP7 (MLP7_7).

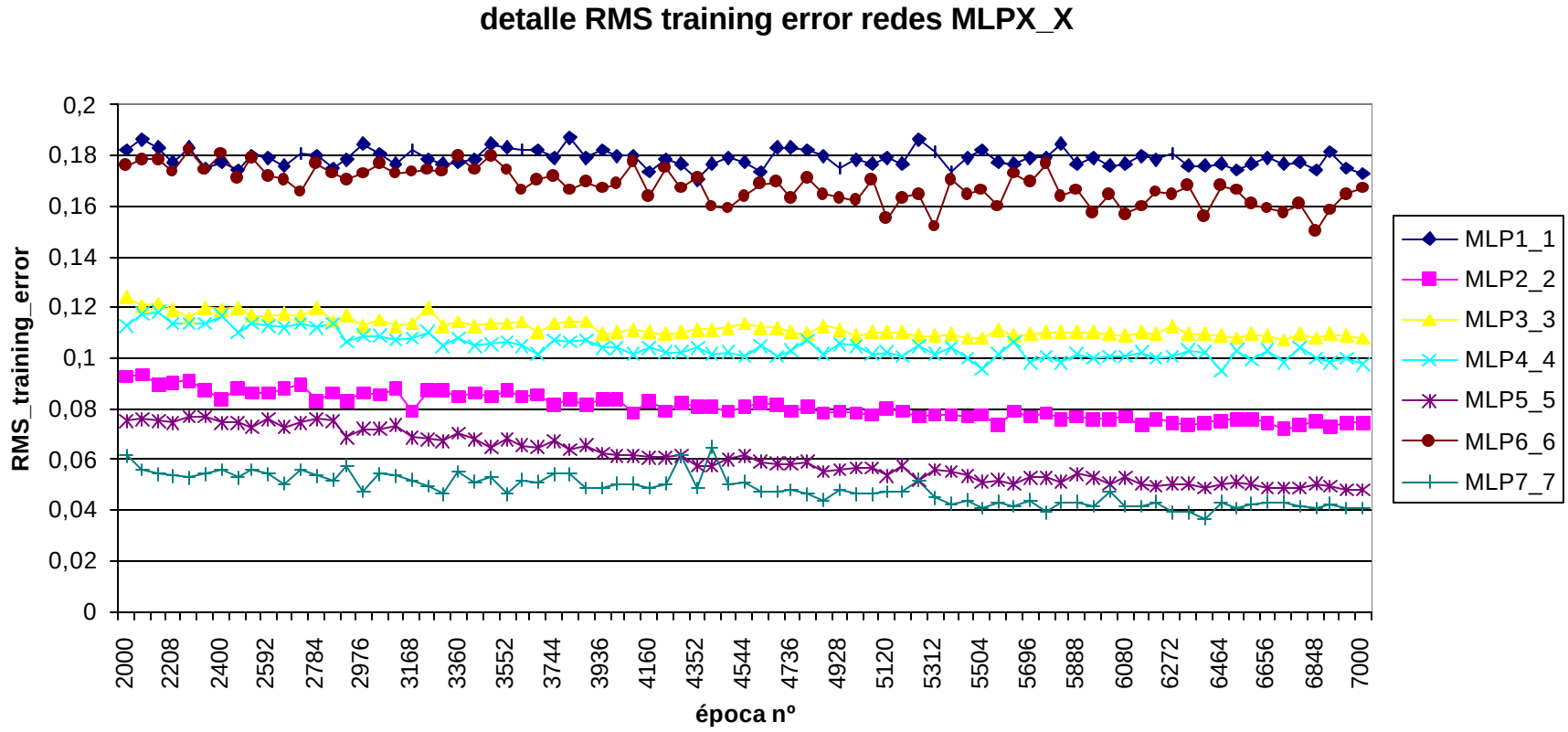

RMS Training Error redes MLPX
RMS_Training_error
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
1
260
520
776
1040
1296
1552
1808
2080
2336
2592
2848
3104
3360
3616
3872
4160
4416
4672
4928
5184
5440
5696
5952
6208
6464
6720
6976
Época nº
MLP1
MLP2
MLP3
MLP4
MLP5
MLP6
MLP7

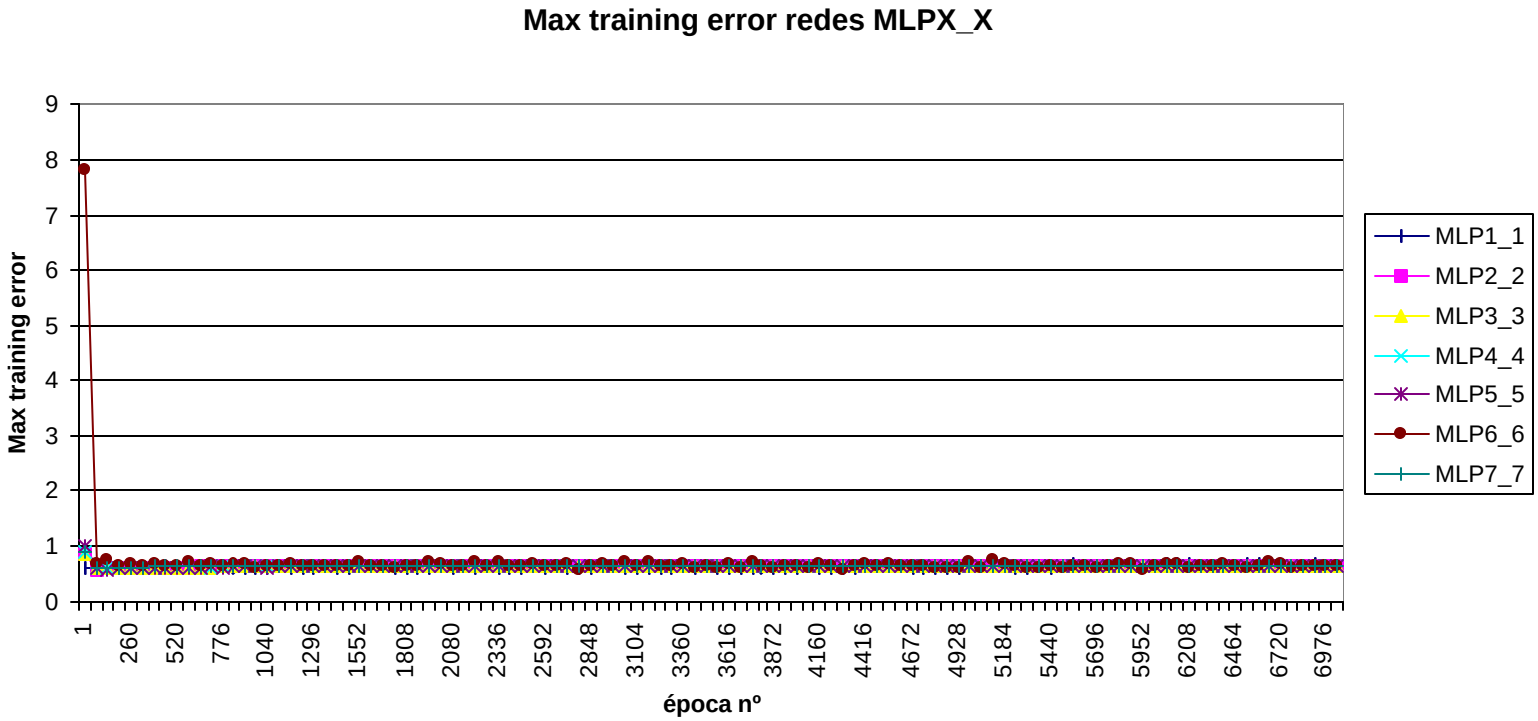

Detalle RMS training error redes MLPX
RMS_Training_error
0,2
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
2000
2208
2400
2592
2784
2976
3168
3360
3552
3744
3936
4160
4352
4544
4736
4928
5120
5312
5504
5696
5888
6080
6272
6464
6656
6848
7000
época nº
MLP1
MLP2
MLP3
MLP4
MLP5
MLP6
MLP7

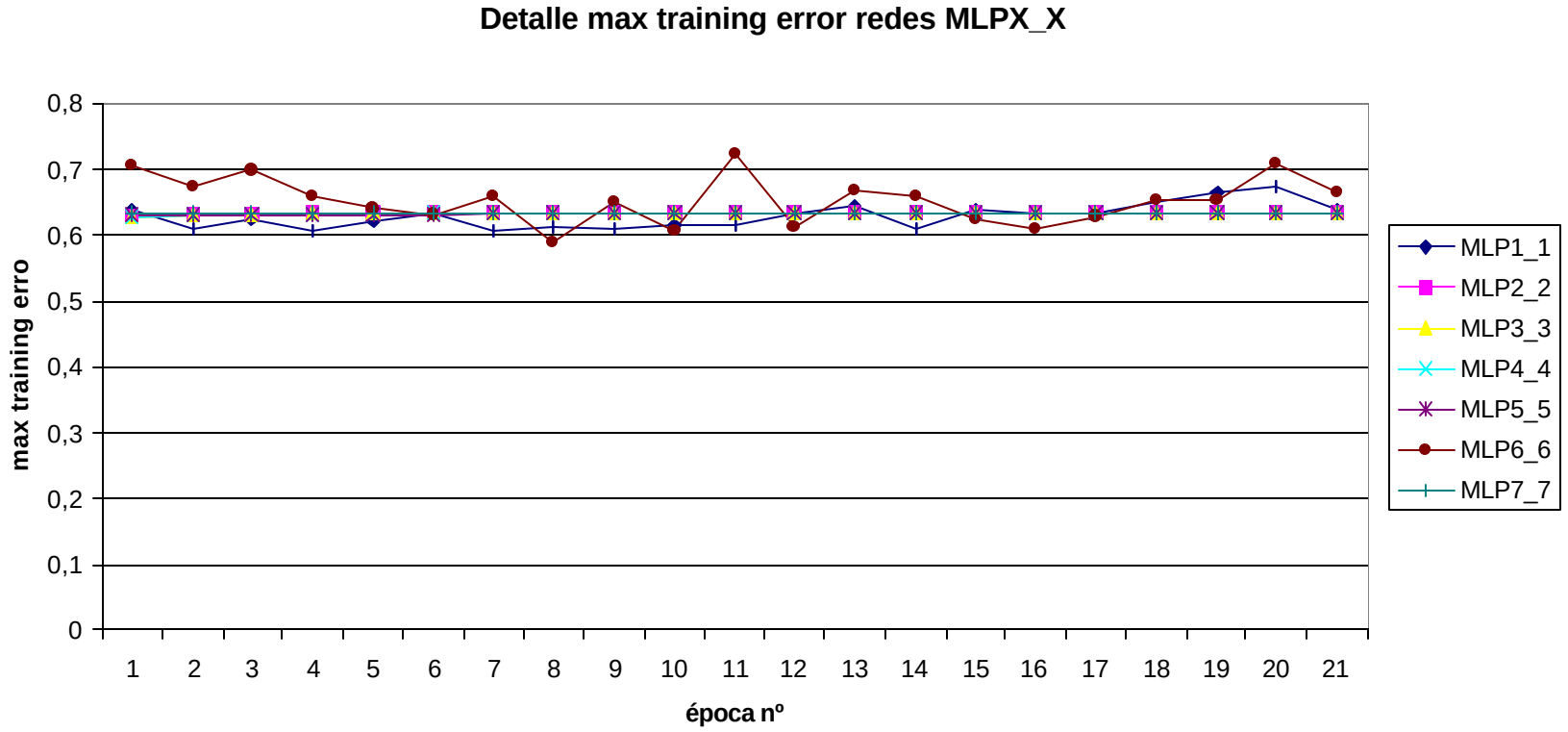

MAX Training Error MLPX
Max training error
6
5
4
3
2
1
0
1
260
520
776
1040
1296
1552
1808
2080
2336
2592
2848
3104
3360
3616
3872
4160
4416
4672
4928
5184
5440
5696
5952
6208
6464
6720
6976
épocas
MLP1
MLP2
MLP3
MLP4
MLP5
MLP6
MLP7


Detalle Max Training error redes MLPX
Max training error MLP)
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
2000
2208
2400
2592
2784
2976
3168
3360
3552
3744
3936
4160
4352
4544
4736
4928
5120
5312
5504
5696
5888
6080
6272
6464
6656
6848
7000
época nº
MLP1
MLP2
MLP3
MLP4
MLP5
MLP6
MLP7


RMS training error redes MLPX_X
RMS_training_error
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
1
260
520
776
1040
1296
1552
1808
2080
2336
2592
2848
3104
3360
3616
3872
4160
4416
4672
4928
5184
5440
5696
5952
6208
6464
6720
6976
Época nº
MLP1_1
MLP2_2
MLP3_3
MLP4_4
MLP5_5
MLP6_6
MLP7_7

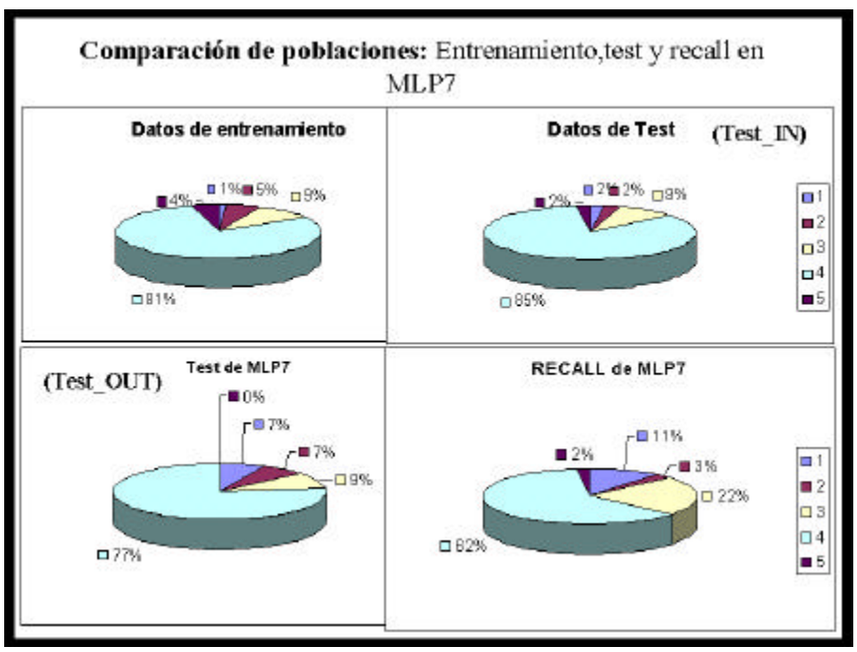

detalle RMS training error redes MLPX_X
RMS_training_error
0,2
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
2000
2208
2400
2592
2784
2976
3168
3360
3552
3744
3936
4160
4352
4544
4736
4928
5120
5312
5504
5696
5888
6080
6272
6464
6656
6848
7000
época nº
MLP1_1
MLP2_2
MLP3_3
MLP4_4
MLP5_5
MLP6_6
MLP7_7


Max training error redes MLPX_X
Max training error
época nº
0
1
2
3
4
5
6
7
8
9
1
260
520
776
1040
1296
1552
1808
2080
2336
2592
2848
3104
3360
3616
3872
4160
4416
4672
4928
5184
5440
5696
5952
6208
6464
6720
6976
MLP1_1
MLP2_2
MLP3_3
MLP4_4
MLP5_5
MLP6_6
MLP7_7


Detalle max training error redes MLPX_X
max training erro
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
época nº
MLP1_1
MLP2_2
MLP3_3
MLP4_4
MLP5_5
MLP6_6
MLP7_7

9.3. Comentarios sobre la clasificación usando Redes Neuronales.

Los siguientes diagramas de proporciones permiten tener una representación visual del comportamiento en la clasificación de la red MLP7 (MLP7_7). Podemos además apreciar la presencia de clases que encontró en la etapa de clasificación real (recall de MLP7).Podemos ver que la muestra para testear la red (test_IN) fue elegida de manera de mantener las proporciones de clase que presentan los datos de aprendizaje (datos de entrenamiento) y cómo ésta fue bien caracterizada en la salida de test (test_out). Sucede que en este problema, se tenía conocimiento aproximado de las presencias de las clases en la población a clasificar. Se sabe que la clase con presencia más fuerte es cluster 4 y se sabe que esta será la clase que mayor presencia tendrá en los registros después de la clasificación. Es interesante notar cómo las presencias de las clases en la población es aproximadamente un patrón que se repite en las tres poblaciones: entrenamiento, test y recall.



9.4. Elección de la mejor solución.

Ya nos encontramos en condiciones de poder comparar ambas mejores soluciones, basándonos en los resultados arrojados por las matrices de confusión.

Ya contamos con los mejores modelos de ambos enfoques (bayesiano y neuronal), en ambas situaciones de datos (entrenamiento y test con las mismas poblaciones o con poblaciones distintas). Sabemos que la red neuronal pasa por dos procesos iniciales: la fase de entrenamiento y la fase de test (clasificación sobre datos cuya pertenencia a las clases es conocido). Las redes bayesianas por su parte poseen una fase de construcción y una fase de inferencia. Cuando la inferencia se hace sobre datos con pertenencia a clases conocidas, nos encontramos en una fase de "test bayesiano". En virtud de lo anterior compararemos ambos grupos de soluciones.

9.4.1. Elección de la mejor solución con datos iguales.

Esto significa, en el caso de las redes neuronales, construcción de la red con la misma población de datos con que se hará el test. En el caso de las redes bayesianas significará que la inferencia se hará sobre los mismos datos que fueron utilizados en la fase de construcción. Las siguientes matrices de confusión representan ambos casos: MLP7_7 y LANCO2.

mlp7_7	1	2	3	4	5	total	% precisión en la clasificación	mlp7
1	0	0	0	0	0	0	0	1
2	0	3	0	0	0	3	100	2
3	0	1	9	1	0	11	81.81	3
4	1	1	2	94	0	98	95.92	4
5	0	0	0	0	4	4	100	5
Total	1	5	11	95	4	110/116		Total
%precisión en la clase	0	60	81.81	98.95	100		94.83	%precisión en la clase
LANCO2	1	2	3	4	5	total	% precisión en la clasificación	LANCO3
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	2
3	0	1	2	5	0	8	25	3
4	1	4	6	72	1	84	85.71	4
5	0	0	3	18	3	24	12.5	5
Total	1	5	11	95	4	77/116		Total
%precisión en la clase	0	0	18,182	75,789	75		66.38	%precisión en la clase

Como se observa en las matrices de confusión, la mejor calidad de clasificación, tanto a nivel de clases como a nivel global fue desempeñada por el modelo de red neuronal multilayer perceptron MLP7_7.

9.5. Elección de la mejor solución con datos distintos.

Esto significa, en el caso de las redes neuronales, construcción de la red con población de datos distinta con la que se hará el test. En el caso de las redes bayesianas significará que la inferencia se hará sobre datos distintos que los que fueron utilizados en la fase de construcción. Las siguientes matrices de confusión representan ambos casos: MLP7 y LANCO3.

mlp7	1	2	3	4	5	total	% precisión en la clasificación
1	1	0	0	2	0	3	33.33
2	0	0	0	3	0	3	0
3	0	0	0	4	0	4	0
4	0	1	4	27	1	33	81.82
5	0	0	0	0	0	0	0
Total	1	1	4	36	1	28/43	
%precisión en la clase	100	0	0	75	0		65.12
LANCO3	1	2	3	4	5	total	% precisión en la clasificación
1	0	0	1	6	0	7	0
2	0	0	0	0	1	1	0
3	0	0	0	14	0	14	0
4	0	1	1	12	0	14	85.7
5	1	0	2	4	0	7	57.14
Total	1	1	4	36	1	12de43	
%precisión en la clase	0	0	0	33.3	0		27.9%

Como se observa en las matrices de confusión, la mejor calidad de clasificación, tanto a nivel de clases como a nivel global fue desempeñada por el modelo de red neuronal multilayer perceptron MLP7.

9.6. Conclusiones finales.

Ambas comparaciones permiten concluir, que dada la naturaleza de los datos, la cantidad de información contenida en éstos y el conjunto de variables involucradas para la construcción de ambos modelos, la mejor solución es dada por el diseño del experimento usando redes neuronales.

Lo anterior no quiere decir que una tecnología sea superior a otra, de hecho ambos enfoques tienen sus particularidades que pueden ser complementarias y que ayudan a enriquecer el conocimiento global del fenómeno que es modelado y de la interacción entre las variables. De hecho, se podría situar al enfoque probabilístico de las redes bayesianas como una tecnología más, que puede formar parte del proceso de data mining.

Es importante hacer un esfuerzo por comparar las características de cada enfoque. Ambos enfoques hacen uso de las variables de entrada a sus modelos para construirlos. Las redes multinet bayes representan un aporte en el sentido de permitir incluir conocimiento experto dado por la relación de causalidad de los nodos y por el orden aplicado a éstos. Puede suceder, sin embargo, que esta información influya de manera negativa sobre el desempeño en la clasificación, tal como sucedió en este trabajo.

En el caso de las redes neuronales, la arquitectura de la red es rígida y estructuralmente no presenta modificaciones durante la fase de entrenamiento, salvo el ajuste de los pesos que tiene un impacto sobre la intensidad con que se comunican dos nodos (excepto en el caso de poda, que es indicada antes del entrenamiento con un criterio de relevancia de las conexiones). En el caso de las redes multinet bayes, el proceso de poda es una fase natural de la creación de las subredes.

En las redes neuronales los valores que pueden tomar la variable de salida (valores de la clasificación), son determinados por el número de neuronas de la capa de salida, de manera que tenemos una estructura global que posee en sí simultáneamente, las caracterizaciones de cada clase de salida. Esto no sucede con las redes multinet bayes, donde contamos con una estructura para cada valor posible de clasificación.

La pertenencia de un individuo durante la clasificación real, a cada una de las clases, se encuentra dado en el caso de las redes neuronales, a la sinapsis más fuerte con respecto a las neuronas de la capa de salida. En el caso de las redes multinet bayes, esta pertenencia se encuentra dada por un intervalo de probabilidad definido para cada clase.

Por último, es importante contar con información de calidad que contribuya positivamente sobre la construcción de estos modelos, de manera de optimizar su desempeño en la calidad de la clasificación. Ésta viene dada por la correcta selección de muestras que representen la realidad contenida en los datos y por la selección de variables que provengan de distintos ámbitos del conocimiento, que influyen directa o indirectamente sobre el fenómeno estudiado. De hecho, ésa es una de las principales características que perfilan a la Minería de Datos como una fuente de conocimiento novedosa en nuestro presente científico y tecnológico, y en el futuro.

Bibliografía.

Análisis del problema inicial:

- [Kal,1999] Kaly U., Briguglio L., McLeod H. Schmall S., Pratt and Pal R.,1999. Environmental Vulnerability Index (EVI) to Summarize national environmental vulnerability profiles. SOPAC. Technical report 275. 66 p.
- [INFOR,2002] INFOR, "Desarrollo y aplicación de alternativas de manejo para el abastecimiento continuo de bienes y servicios con bosques nativos", FDI-CORFO/INFOR. Capítulos III y IV.
- [San,1982] Sanchez A.1982. Las provincias de Chile: un estudio multivariado usando análisis de componentes principales. Inst. de Estudios Regionales, Univ. De Chile. Documento de trabajo n°5. 39 p.

Redes bayesianas y Suite Bnsoft:

- [Che,1997] Cheng J., Bell D. A, Liu W., "Learning Belief Networks from Data: An Information Theory Based Approach", School of Information and Software Engineering", University of Ulster at Jordanstown, United Kingdom.
- [Cho,1998] Chow, C.K. and Liu, C.N., Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory, 14, 462-467, 1968.
- [Coo,1992] Cooper, G.F., Herskovits, E., A Bayesian Method for the induction of probabilistic networks from data, Machine Learning, 9, 309-347, 1992.
- [Her,1991] Herskovits, E.H., Computer-based probabilistic network construction, Doctoral dissertation, Medical information sciences, Stanford University, Stanford, CA, 1991.

- [Per,1988] Pearl, J. , Probabilistic reasoning in intelligent systems: networks of plausible inference, Morgan Kaufmann, 1988.
- [Spi,1991] Spirtes, P., Glymour, C. and Scheines, R., An algorithm for fast recovery of sparse causal graphs, Social Science Computer Review, 9, 62-72, 1991.
- [Suz,1996] Suzuki, J., Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique, Proceedings of the international conference on machine learning, Bally, Italy, 1996.
- [URL,1] Belief Network (BN) PowerConstructor: http://www.cs.ualberta.ca/~jcheng/bnpc.htm
- [URL,2]Belief Network (BN) PowerPredictor: http://www.cs.ualberta.ca/~jcheng/bnpp.htm
- [URL,3]Data PreProcessor: http://www.cs.ualberta.ca/~jcheng/prep.htm

Fundamentos de Minería de Datos:

- [DAE,2000]“Minería de Datos, conceptos y objetivos”, DAEDALUS, Data, Decisions and Languaje S.A,(www.daedalus.es)
- [Fay,1996] Fallad M. U., “Advances in Knowledge discovery and Data Mining”.
- [Web,2000]Weber R.(2000), “Data Mining en la Empresa y las Finanzas utilizando tecnologías inteligentes”. Revista de Ingeniería de Sistemas, Volumen XIV, numero 1. pp 61-78.

Fundamentos sobre GIS:

- [Sol,2002] Solivelles U. (2002) “sistemas de información geográfica (SIG) y percepción remota”., apuntes de clases para la asignatura “Seminario de Informática y Matemática aplicada”, INFO 258.

Fundamentos sobre Redes Neuronales:

- [Brau,1995] Brause, R.: Neuronale Netze. B.G. Teubner, Stuttgart, 1995.
- [Dat,2002]Data Engine Documentation, Part IV: Theory, Cap. 3: Neural Networks.
- [Fah,1998] Fahlman, S.E.: An empirical study of learning speed in back propagation networks. Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufman, 1988.
- [Fu,1997] Fu, K.S.: Sequential Methods in Pattern Recognition. Academic Press, New York, 1987.
- [Hay,1999]Haykin Simon (1999),”Neural Networks” a comprehensive Foundation.
- [Hil,2000]Hilera J. R., Marinez V.(2000), “Redes Neuronales Artificiales” Fundamentos, Modelos y Aplicaciones.
- [Hor,1991] Hornik, K.: Approximation capabilities of multilayer feedfor-ward networks. Neural Networks 4:2, 1991, pp. 251-257.
- [Jac,1988] Jacobs, R.: Increased Rates of convergence through Learning Adaptation. Neural Networks, Volume 1, pp. 295-307, 1988.
- [Jan,1995] Jansen, M.: Globale Modellbildung und garantiert stabile Regelung von Robotern mit strukturierten neuronalen Netzen. Dissertation im Fachbereich Elektrotechnik, Universität Duisburg 1995.
- [Mas,1993] Masters, T.: Practical Neural Network Recipes in C++. Academic Press 1993.
- [Par,1982] Parker D., "Learning Logic", Invention Report. S81-64, File 1. Office of technology Licensing, Standford University, 1982.

- [Ree,1998] Reed, R.: Pruning Algorithms A Survey. IEEE Trans. On Neural Networks, Volume 4, Number 5, 1993, pp. 707-740.
- [Rum,1986] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Backpropagation. In: Rumelhart, D.E., McClelland (Eds.): Parallel Distributed Processing - Explorations in the Microstructure of Cognition. Vol. 1: Foundations, MIT Press, Cambridge, 1986, pp. 318-362.
- [Sil,1990] Silva, F., Almeida, L.: Speeding up Backpropagation, in: Advanced Neural Computers, Hrsg: Eckmiller, R., North-Holland, Amsterdam, 1990.
- [Wei,1992] Weigand, A., Hubermann, B., Rumelhardt, D.: Predicting sunspots and exchange rates with connectionist networks, in: Casdagli, M., Eubank, S. (Eds), Nonlinear modelling and forecasting, SFI Studies in the Sciences of Complexity, Proceedings Volume XII, Addison Wesley, 1992.
- [Wer,1974] Werbos P., "Beyond Regression: New tools for prediction and analisys in the behavioral sciences", ph.D. Thessis. Hardvard University, 1974.
- [Wer,1988] Werbos, P.: Backpropagation: Past and Future. Procee-dings of the Int. Conf. On Neural Networks, Volume I, pp. 343-353, IEEE Press, New York, 1988.
- [Zel,1994] Zell, A.: Simulation neuronaler Netze. Addison-Wesley, 1994.

Data Engine:

- [Beil,2001]Beilke Sabine, “Data Engine; from data to Information and Knowledge”, MIT - Management Intelligenter Technologien GmbH.
- [URL,4] Información sobre el producto Data Engine y sus características: http://www.dataengine.de/english/sp/index.htm
- [URL,5] Ejemplos de empresas que han aplicado Data Engine en sus procesos de Data Mining: http://www.dataengine.de/english/sp/testimonial/testimonial.htm

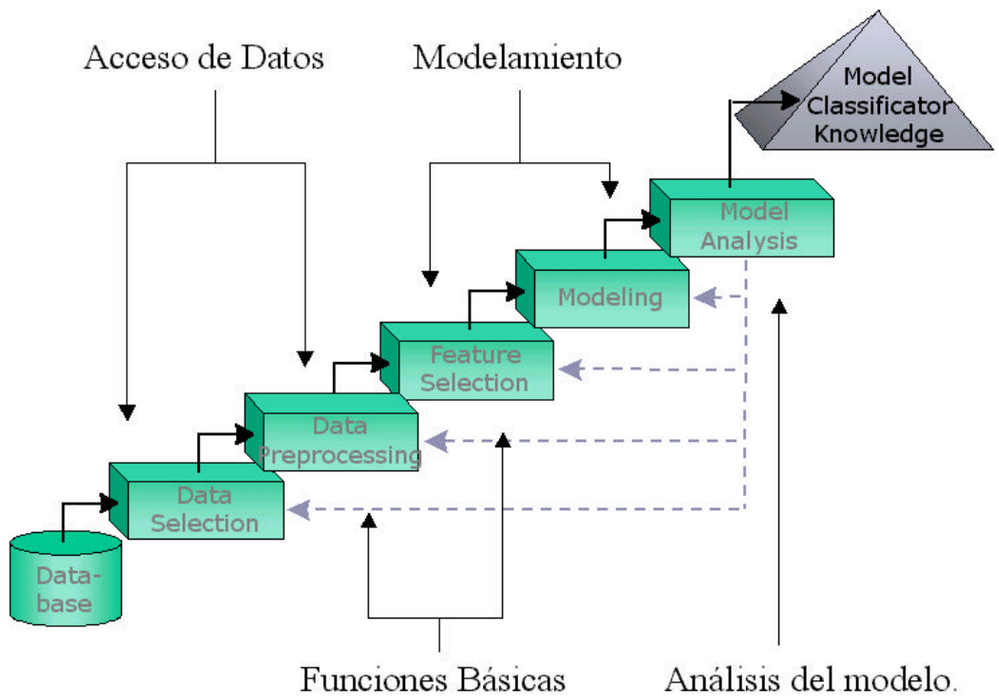
- [URL,6] Ejemplos de proyectos de DataMining usando Data Engine desarrollados por DAEDALUS, Data, decisions and languaje S.A.: http://www.daedalus.es/DocumentacionMD.asp

Anexo nº1: Software para data mining: Data Engine 4.0

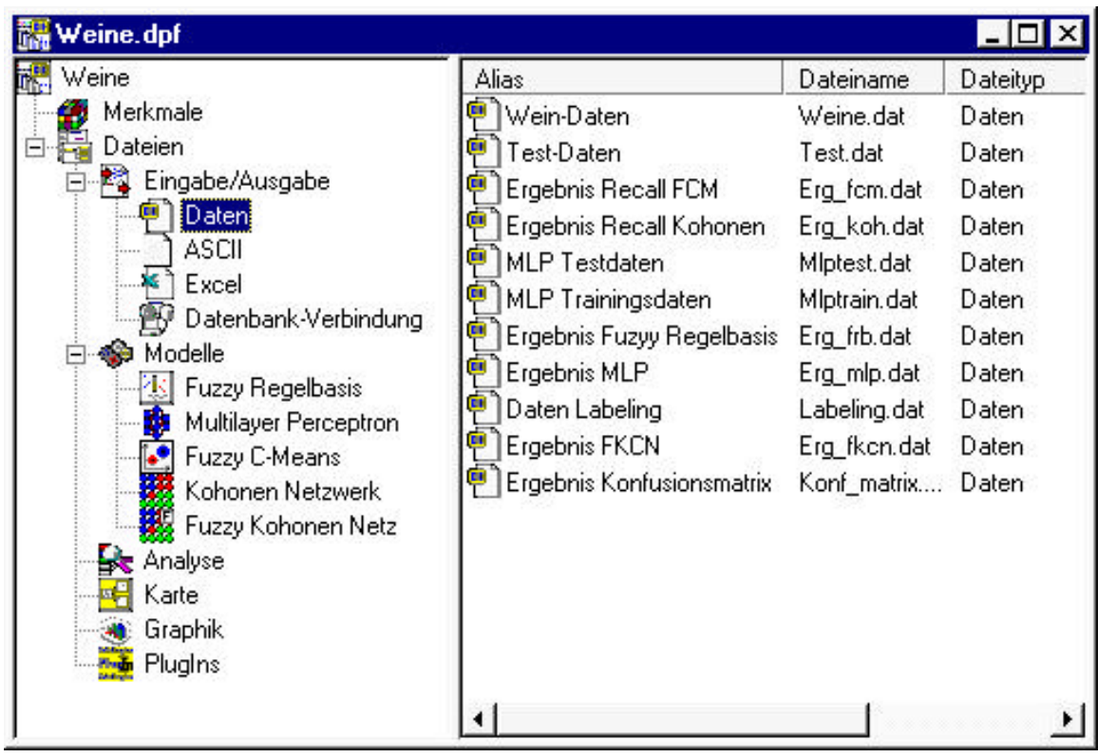
Data Engine es una caja de herramientas desarrollada por el MIT Management Integillenter Technologient GMBH.

Esta es una herramienta de software para análisis de datos inteligente que combina tecnologías de Redes Neuronales, métodos estadísticos y tecnologías fuzzy.

Data Engine soporta todos los pasos del proceso de extracción de conocimiento (KDD). Desde los datos iniciales hasta la extracción final de conocimiento[Bei,1999].



La interfaz Proyect Window es una estructura de árbol que permite gestionar las fuentes de datos, modelos aplicados a estos y gráficos. Corresponde a la interfaz principal de todo proyecto de Data Mining.



Un proyect window sirve como una estructura de trabajo para todos los archivos, modelos y información, sobre las características de los datos que se encuentran relacionados con un mismo proyecto de estudio.

Acceso a Datos

Data Engine permite manipular los datos:

1. En formato ASCII
2. En formato Excell
3. Desde una base de datos remota (vía OL/DB – ODBC)
4. Desde otras aplicaciones vía interfaces especiales o pulg. –in.

Funciones Básicas

- **Análisis estadístico:** Se puede comenzar el análisis de datos con un análisis estadístico. Las funciones de análisis estadístico de Data Engine, incluyen varianza, promedios, mínimos y máximos, desviación estándar, análisis por lotes,etc.
- **Funciones Matemáticas:** De propósito general, logarítmicas, trigonométricas, hiperbólicas, de redondeo, escalamiento, etc.
- **Funciones de procesamiento de señal:** FFT, FFT inversa, filtro digital, smoothing, etc.
- **Chart:** Series de tiempo, proyecciones en 2D Y 3D,etc.

Modelamiento

Data Engine provee varios modelos que pueden ser aplicados a los datos. Entre ellos se cuenta con reglas fuzzy, fuzzy clustering, redes neuronales, redes neuronales fuzzy, etc.

- **Reglas fuzzy:** Las reglas difusas representan el conocimiento experto que no puede o no debería ser representado en forma matemática explícita.
- **Fuzzy clustering:** El agrupamiento clásico (clustering) asigna cada objeto a exactamente una clase, mientras que en el agrupamiento difuso (Fuzzy clustering) los elementos son asignados a cada clase con un grado de pertenencia a cada una de ellas. El grado de difusión de las clases puede ser determinado.

Los métodos de fuzzy clustering están disponibles para dividir objetos en clases con características homogéneas. Data Engine provee la implementación del algoritmo fuzzy c means.

Puede ser aplicado en diversas áreas, como es el caso de control de calidad, segmentación, identificación de procesos y análisis de escenarios.

- **Redes neuronales:** Las redes neuronales aprenden relaciones desde los datos. Tipos diferentes de redes, métodos de aprendizaje y parámetros permiten una adaptación flexible.

Data Engine provee los siguientes tipos de redes neuronales:

1. **Multilayer Perceptrón:** Es una red multi-nivel, de aprendizaje supervisado, con una arquitectura flexible (número de capas, pesos neuronales, etc.) donde se pueden aplicar tipos diferentes de aprendizaje. Este método puede ser aplicado en tareas tales como predicción de ventas, modelamiento de procesos, tareas de control y procesamiento de imágenes.

2. **Redes de Kohonen:** Utilizan mapas de características auto-organizativas. Es un tipo de red de aprendizaje no supervisado. Son usadas principalmente en tareas de clasificación. Estas redes pueden representar cualquier forma de clase deseada en el espacio de características. En este tipo de clasificación, cada objeto es asignado exactamente una clase. El número de clases es aprendido desde los datos.

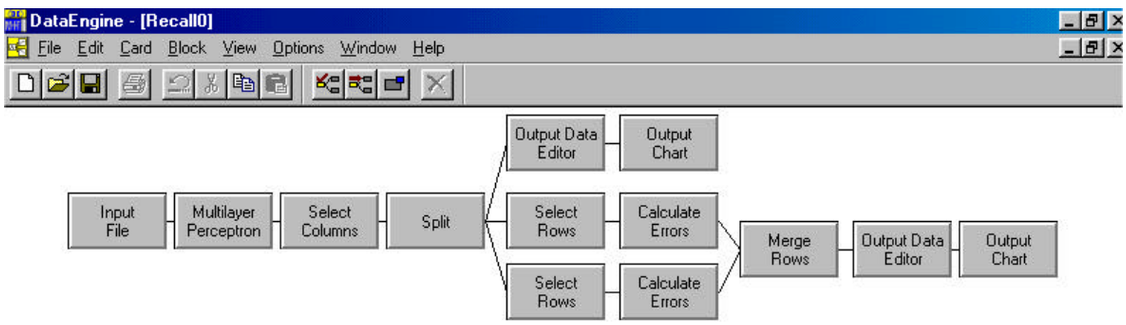
Análisis del Modelo.

El análisis de componentes permite validar y mejorar los modelos. Los valores de las características de los objetos pueden ser variados y los cambios en las salidas de los modelos son desplegados. Un “slider” puede ser usado para variar los valores. Con esto se puede conocer el impacto de la variación de una característica en la salida global del modelo. De este modo, se pueden realizar diversos análisis de sensibilidad.

Para tareas de clasificación, una matriz de confusión permite determinar qué tipos de errores existen en los datos de test.

Graphical Macro languaje

El lenguaje Macro-gráfico de Data Engine habilita una representación simplificada de los pasos de procesamiento de datos.



Los comandos macros llamados bloques de función son colocados en una tarjeta.

Visualización de los Datos y Resultados.

Data Engine proporciona un potente módulo que permite visualizar el comportamiento de los datos graficando el comportamiento de características individuales de los datos, contraste de características, etc.

Adaptación de Data Engine a los requerimientos específicos

Se pueden integrar a Data Engine nuevas funcionalidades definidas por el usuario, como es el caso de los bloques de funciones.

Se encuentran disponibles módulos adicionales para tareas específicas tales como Feature selector, Fuzzy pattern classification, Decisión Xpert y ProcessAnalyser. Uno de estos pulg.–in, Advanced clustering, inlcluye los siguientes métodos:

- Fuzzy C-Means
- Gustavson-Kessel-Algorithmus
- achsparallele Variante des Gustavson-Kessel-Algorithmus
- Gath & Geva-Algorithmus
- achsparallele Variante des Gath & Geva-Algorithmus
- jeweils in probabilistischer und possibilistischer Form

Campos de aplicación de Data Engine.

Data Engine ha sido aplicado a diversas áreas donde se han hecho estudios de Data Mining. Entre estas áreas se incluyen:

1. Análisis de Procesos y Modelamiento.
2. Control de calidad.
3. Diagnosis y Monitoreo.
4. Control.
5. Análisis de Riesgo.
6. Database Marketing.
7. Predicción.

Apoyo a decisiones.

Anexo nº2 "La Suite BNSOFT y la construcción de redes Multinet".

La suite BNSOFT.

Esta suite corresponde a un sistema que permite desarrollar redes bayesianas de diversas arquitecturas para tareas de clasificación, utilizando capacidades de inferencia. Se encuentra formada por la interacción de tres módulos principales:

- Data Prepocessor.
- BN Power Constructor.
- BN Data Predictor.

Data Preprocessor.

Este módulo permite detectar y discretizar los atributos o campos pertenecientes a una base de datos que contienen datos continuos. Soporta conexión a distintos tipos de bases de datos mediante ODBC. Permite separar los datos, en datos para entrenamiento o construcción de la red y datos para test (o inferencia), como es requerido por BN power predictor. Posee una interfaz tipo wizard, donde se guía al usuario los pasos necesarios para la discretización de los datos[URL,3]:

- **Paso1:** se le indica al software cuál el tipo de formato de BD, dónde se encuentran los datos a ser dicretizados, seleccionando de una lista el formato adecuado. En caso de encontrarse en una BD remota se procede a conectarse mediante ODBC.

- **Paso2:** En este paso se indica la ubicación física de nuestra BD, así como los atributos para lograr una conexión en caso que estemos utilizando ODBC, como

es el caso de DSN (Data Source Name), usuario, password, database, driver, server.

- **Paso3:** se especifica la tabla donde se encuentran los datos a ser procesados.

- **paso4:**se especifican los atributos que serán discretizados por el software, como el criterio utilizado para discretizar: Equal Width, Equal Frequency and Supervised (entropy based).

Una vez que los datos han sido procesados, existe la posibilidad de ingresar directamente a cualquiera de los otros módulos: BN Data Constructor (para construir la red a partir de los datos discretizados) o BN data predictor (para inferencia sobre los datos).

BN Power Constructor.

Este módulo permite construir una red de creencia basándose en los datos discretizados por Data Preprocessor. Este motor de construcción de redes de creencia se basa en test de independencia condicional, y en los algoritmos descritos más adelante. Soporta la inclusión de dominio del conocimiento al permitir ingresar el orden completo de los nodos, o el orden parcial, causas y efectos directos entre nodos, enlaces no permitidos, y especificación de nodos raíz y hoja. Este módulo genera como salida un archivo donde se especifica la estructura completa de la red (archivo.bnc). Posee una interfaz tipo wizard, donde se guía al usuario los pasos necesarios para la construcción de la red[URL,1]:

- **Paso 1:** se selecciona el formato de base de datos apropiado de los datos que se utilizarán para construir la red.

- **Paso 2:** En este paso se indica la posición física de nuestra BD, así como los atributos para lograr una conexión en caso que estemos utilizando ODBC, como es el caso de DSN (Data Source Name), usuario, password, database, driver, server.
- **Paso 3:** Se selecciona la tabla especifica que se desea utilizar. Una vez seleccionada la tabla, se despliega la lista de atributos que potencialmente pueden representar nodos en la red. Aquí se indica cuáles son los atributos de la tabla que se desean utilizar.
- **Paso 4:** Se despliega una interfaz que permite capturar el dominio del conocimiento, en el caso que se desee incluir conocimiento experto para la construcción de la red.
- **Paso 5:** este paso es usado para capturar la información del proceso en un archivo.log. un archivo .log almacena información de resultados intermedios en el proceso de construcción de la red, a partir de un conjunto de datos. Esto permite acelerar el proceso de construcción cuando el mismo conjunto de datos es procesado nuevamente. Se pueden hacer modificación en la inclusión de conocimiento experto sin alterar la integridad de este archivo.

Al finalizar dicha fase, existe la opción de guardar la red creada para un futuro uso, en forma de un archivo .BNC. Además se puede editar el modelo de red usando una interfaz visual, incluyendo aristas nuevas (nuevas relaciones entre nodos) o excluyendo aristas existentes. Luego, existe la posibilidad de utilizar **BN data predictor** para utilizar la red creada en la inferencia de nuevos datos.

BN Power Predictor.

Permite aprender clasificadores de redes bayesianas generales y clasificadores de redes multinet desde los datos de entrenamiento, y usar dichas estructuras para clasificar nuevos datos. BN Power Predictor puede ser utilizado en tres modalidades distintas[URL,2]:

1. **Modo1:** Aprender un nuevo clasificador de red de creencia desde los datos preprocesados y utilizarlo para clasificar nuevos datos.

2. **Modo2:** Modificar la estructura de una red de creencia ya creada y utilizarla para clasificar nuevos datos.

3. **Modo3:** Usar una red de creencia existente para clasificar nuevos datos.

Como cada una de las modalidades de uso apuntan a objetivos distintos, inicialmente se muestra una interfaz de usuario que pregunta por la modalidad a utilizar. Dependiendo de la modalidad en uso, la sucesión de pasos que sigue el asistente es distinta:

1. Si se usa en **modo1**, el asistente sigue todos los pasos.
2. Si se usa en **modo2**, los pasos a seguir por el asistente son 1,4,5,6,7 y8.
3. Si se usa en **modo3**, se siguen los pasos 6,7,8.

Posee una interfaz tipo wizard, donde se guía al usuario los pasos necesarios para la construcción de la red:

- **Paso1:** Selección de un conjunto de datos de entrenamiento y de las variables.

- **Paso2:** Ingresar el dominio del conocimiento.

- **Paso3:** Ingresar información, vía archivos .log y comienzo del proceso de construcción

- **Paso4:** Para mostrar, editar y guardar la estructura de la red creada (archivo.bnc).

- **Paso5:** edición de la tabla de costos de missclasification.

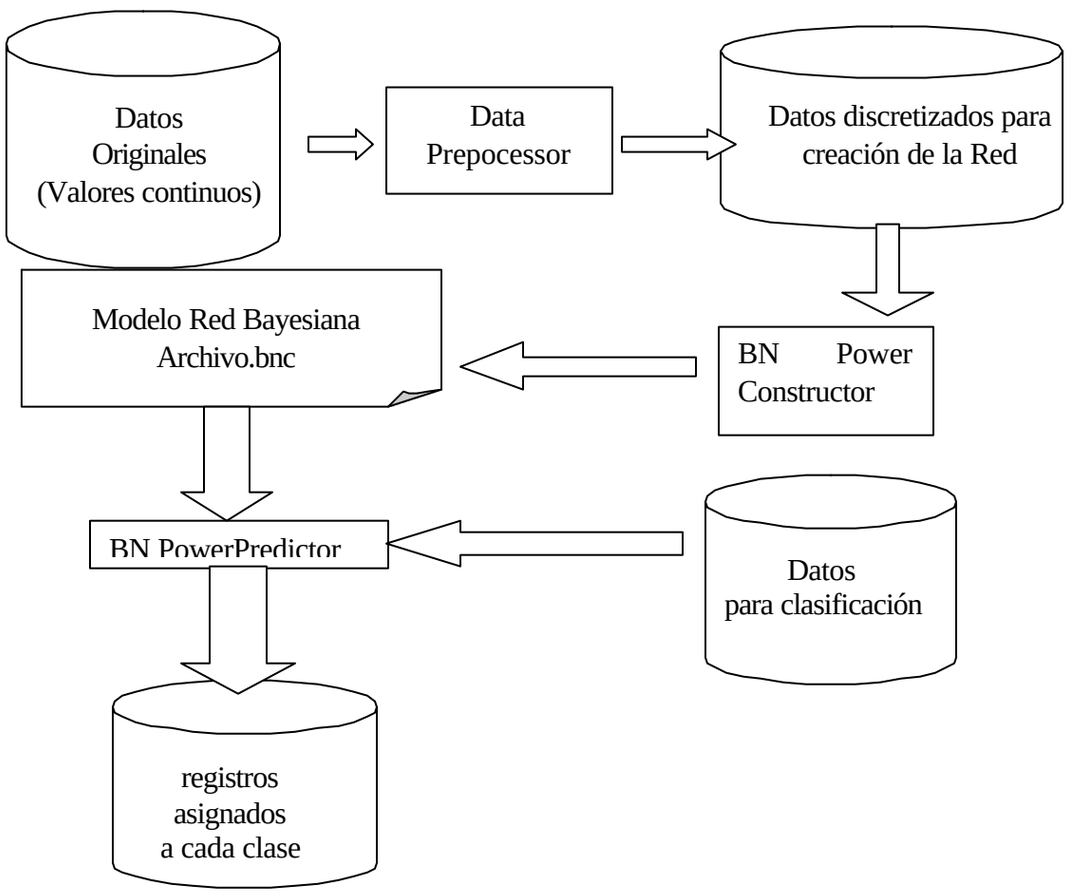
- **Paso6:** Selección del formato de la base de datos para test y generación de la clasificación.

- **Paso7:** Ingresar el nombre de la conexión de la base de datos mediante ODBC.

- **Paso8:** Selección del conjunto de datos de test especificados en la base de datos del paso anterior.

Este módulo genera como salida un archivo donde se muestra el resultado de la clasificación. Este resultado corresponde a una columna, agregada a los datos de test, donde se asigna a cada uno de los registros de la tabla la pertenencia de este a cada una de las clases definidas por la discretización de los datos y la designación del atributo que servirá para etiquetar las clases. Este punto se discutirá con más detalle en las siguientes secciones.

El siguiente diagrama ilustra la interacción de estos tres módulos. Nótese que se ilustra la funcionalidad del módulo **BN Power Predictor** usado en **modo3**:



Descripción del algoritmo.

A continuación se describirán los algoritmos que utiliza el módulo **BN Power Constructor** para la creación de la red bayesiana.

Existen dos grandes grupos en los que podemos clasificar los algoritmos para la construcción de redes de creencia, basados en la teoría de la información: el primero de ellos, **algoritmos basados en puntaje y búsqueda**, requieren como información de entrada el orden de los nodos (es decir, la relación de causalidad entre los nodos que conforman la red). La segunda categoría corresponde a **algoritmos basados en test de independencia** condicional (CI test), donde no se requiere la información de orden de los nodos. El algoritmo de construcción de multinet cae en la segunda categoría. Éste toma como entrada una base de datos y genera una red de creencia multinet como salida[Che,1997].

Introducción.

Una red de creencia bayesiana es una poderosa representación del conocimiento y del razonamiento bajo condiciones de incertidumbre. También es posible caracterizarla como un grafo acíclico dirigido (DAG) con una distribución de probabilidad condicional para cada nodo.

La estructura de un DAG, en tales redes, contiene nodos representando variables del dominio y arcos entre los nodos que representan dependencias probabilísticas en la construcción de redes bayesianas desde BD. Los nodos de nuestro DAG representan atributos de la base de datos.

En los últimos años, muchos algoritmos de construcción de redes de creencia han sido desarrollados. Generalmente, estos algoritmos pueden ser agrupados en dos categorías:

- Algoritmos que usan métodos de búsqueda heurística para construir un modelo y la evaluación de estos usando métodos de ponderación. Este proceso continua hasta que la ponderación del nuevo modelo no es significativamente mejor que la del modelo previo. Diferentes criterios de ponderación han sido aplicados en estos algoritmos tales como, métodos de ponderación bayesiana [Coo,1992], [Che,1997]; Métodos basados en entropía [Her,1991] [Che,1997], y métodos de longitud de descripción mínima [Suz,1996], [Che,1997].

- Algoritmos de construcción de redes bayesianas analizando relaciones de dependencia entre nodos. Las relaciones de dependencia son medidas usando alguna clase de test de independencia condicional. Los algoritmos descritos en [Spi,1991], [Che,1997], y el algoritmo presentado a continuación pertenecen a esta categoría.

En el desarrollo de este algoritmo, se tomaron dos hechos en consideración:

1. Todas las situaciones del mundo real usuales se pueden representar con redes dispersas, y las redes densamente conectadas revelan relaciones de muy poca independencia, y así contienen información de muy poco valor. Por lo tanto el algoritmo debería ser particularmente eficiente cuando la base de datos posee una red principalmente dispersa.

2. Puesto que los test de CI con conjuntos de condiciones grandes son computacionalmente caros, los autores tratan de impedir este hecho usando el menor conjunto de test CI posible. Considerando lo anterior, se desarrolla para este software un algoritmo novedoso que puede impedir muchos test innecesarios con grandes conjuntos de condiciones y además reducir el número de estos.

Con este algoritmo, se consideran dos ventajas principales sobre otros algoritmos basados en test de CI:

- La construcción de redes bayesianas a través de tres fases: bosquejo, espesado y poda. Esta es una extensión del algoritmo de Chow y Liu para redes multiconectadas. En el caso donde el orden de los nodos es conocido, este algoritmo preserva las características del algoritmo citado recientemente, en el sentido que requiere de test CI en el orden de $O(N^2)$.

- A diferencia de otros métodos de CI que usan test de c^2 [Spi,1991], [Che,1997], para chequear si dos variables son dependientes, este algoritmo utiliza información muta como test de CI lo cual puede informarnos qué tan cercana es la relación de esas variables, además de la relación en sí. Por lo tanto, este algoritmo tiene la capacidad de comparar los resultados de modo cualitativo.

Una aproximación a la teoría basada en la información.

Este algoritmo construye redes de creencia, analizando relaciones de independencia condicional entre los nodos. antes de revisar este algoritmo, es necesario introducir el concepto de d-separación introducido por pearl [Per,1988], [Che,1997]:

- Para tres conjuntos discontinuos de nodos cualquiera :X,Y,Z en una red de creencia, se dice que X se encuentra d-separado de Y por Z si este no tiene un camino adyacente entre X e Y.

- Un camino adyacente es un camino entre dos nodos sin considerar la direccionalidad de los arcos.

- Una ruta entre X e Y es activa, dada Z si:
  - Cada **collider** está en Z o es descendiente de Z.
  - Cada otro nodo en la ruta se encuentra fuera de Z.

Un **collider** [Spi,1991] de una ruta, es un nodo donde dos arcos en la ruta se encuentran en sus extremos. En una red de creencia, si ésta es un arco desde a hasta b, podemos decir que a es padre de b, y que b es hijo de a. También podemos decir que a se encuentra en la vecindad de b y que b se encuentra en la vecindad de a.

También se puede entender la d-separación del siguiente modo:

Se puede caracterizar una red de creencia como un sistema de red de canales de información donde cada nodo es una válvula que se encuentra siempre activa o

inactiva y las válvulas son conectadas por canales de información ruidosos. La información que fluye puede pasar a través de una válvula activa, pero no a través de una válvula inactiva. Cuando todas las válvulas (nodos) de una ruta adyacente entre dos nodos se encuentran activas, podemos decir que la ruta se encuentra **abierta.** Si cualquier válvula en la ruta se encuentra **inactiva**, podemos decir que la ruta se encuentra **cerrada**. Cuando todas las rutas entre dos nodos se encuentran cerradas, dado el estado de un conjunto de válvulas (nodos), podemos decir que dos nodos se encuentran d-separados por el conjunto de nodos. El estado de las válvulas puede ser cambiado a través de la instancia de un conjunto de nodos. La cantidad de información que fluye entre dos nodos puede ser medida usando información mutua cuando no existen nodos instanciados , o información mutua condicional cuando otros nodos son instanciados.

En teoría de la información, la información mutua de dos nodos x_i, x_j es definida como:

$$I(x_i, x_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \qquad \textbf{(1)}$$

y la información mutua condicional es definida como:

$$I(x_i, x_j | C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j | C)}{P(x_i | C)P(x_j | C)} \qquad \textbf{(2)}$$

donde x_i, x_j son dos nodos y C es un conjunto de nodos. En el algoritmo usado en **BN Power Constructor**, se utiliza $I(x_i, x_j)$ como un test de independencia condicional para medir la información promedio entre dos nodos, cuando el estado de algunos valores son cambiados mediante un conjunto de condiciones C. Cuando $I(x_i, x_j / C)$ es más pequeño que un valor umbral ε, decimos que x_i, x_j son d-separados por un conjunto de condición C, y son condicionalmente independientes.

Usando información mutua en construcción de modelos probabilísticos, se puede revisar el modelo de algoritmos para construcción de árboles de Chow y Liu [Cho,1968], [Che,1997]. En 1987, Rebane y Pearl extendieron el algoritmo de Chow y Liu a la construcción de poli árboles basada en causalidad [Per,1988]. El algoritmo utilizado en **Power Constructor** se basa en estos algoritmos para la construcción de redes de creencia bayesianas.

Este algoritmo, utiliza dos supuestos:

1. Los atributos de la base de datos son discretos y no existen valores nulos.
2. El volumen de los datos es suficientemente grande como para realizar test de independencia condicional (CI).

Fase del algoritmo.

Este algoritmo se compone de tres fases principales:

1. Bosquejo.
2. Espesado.
3. Poda.

La fase de **bosquejo** es esencialmente el algoritmo de construcción de árboles de Chow y Liu. Las fases de espesado y poda son diseñadas para extender la construcción de árboles a redes de creencia bayesianas. En la primera fase, este algoritmo computa la información mutua de cada par de nodos como una medida de cercanía y crea un bosquejo basándose en dicha información. El bosquejo es un grafo conexo simple (un grafo sin ciclos). En un caso especial, cuando las redes bayesianas son árboles o poli-árboles, esta fase puede construir la red correctamente, y la segunda y tercera fase no provocarán cambios sobre su estructura. En la fase de **espesado** el algoritmo adiciona aristas al grafo cuando un par de nodos no puede ser d-separado. Como resultado de esta fase, se tiene un mapa de independencia del modelo de dependencia principal, dado que el modelo principal es un DAG. En la **poda,** cada arista del mapa de independencia es examinada usando test de independencia condicional y puede ser removida si es que el par de nodos conectados por ella pueden ser d-separados. El resultado de esta fase es la estructura de un *mapa perfecto*, cuando el modelo principal es un DAG normal fidedigno. Cuando finaliza esta fase, el algoritmo genera también las orientaciones de las aristas del grafo.

Bosquejo.

1. Iniciar un grafo $G(V, E)$, donde $V =$ (todos los nodos del conjunto de datos) , E es inicialmente una lista vacía L.
2. Para cada par de nodos (v_i, v_j) donde $v_i, v_j \in V$, calcular la información mutua basándose en la ecuación :

$$I(x_i, x_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}$$

   Para todo par de nodos cuyo valor $I(x_i, x_j) > e$, ordenarlos de mayor a menor, basándose en e y colocarlos en la lista $L.$ crear un puntero $p.$ que apunte la primer par de nodos en $L.$.
3. Obtener el primer par de nodos de la lista $L.$ y removerlos de ella. Agregar las correspondientes aristas a E . Mover el puntero $p.$ al par de nodos siguientes.
4. Obtener el par de nodos desde $L.$, que se encuentran donde apunta $p.$ si no hay una ruta adyacente entre estos dos nodos, adicionar la correspondiente arista a E y remover este par de nodos desde $L.$
5. Mover el puntero $p.$ al par de nodos siguientes e ir al paso 4, a menos que $p.$ se encuentre apuntando al final de la lista o G contenga $n-1$ aristas. (n es el número de nodos en G).

Espesado

6. Mover el puntero p. a el primer par de nodos en L.
7. Obtener el par de nodos en L. que son apuntados por p. llamar al procedimiento **tratar_separar_A(grafo actual, nodo1, nodo2)**, para ver si este par de nodos puede ser separados en el grafo actual. Si es así, ir al paso 8. Si no, conectar el par de nodos, adicionando la arista correspondiente en E.
8. Mover el puntero p. al próximo par de nodos y volver al paso 7, a menos que p. apunte al final de L.

Poda.

9. Para cada arista en E, si hay otros caminos , además de esta arista, entre este los dos nodos, remover esta arista temporalmente de E y llamar al procedimiento **tratar_separar_A(grafo actual, nodo1, nodo2).** Si los dos nodos son dependientes, adicionar esta arista de vuelta a E; en caso contrario, removerla permanentemente de E.

10. Para cada arista en E, si hay otros caminos , además de esta arista, entre este los dos nodos, remover esta arista temporalmente de E y llamar al procedimiento **tratar_separar_B(grafo actual, nodo1, nodo2).** Si los dos nodos son dependientes, adicionar esta arista de vuelta a E; en caso contrario, removerla permanentemente de E.

11. Llamar al procedimiento **orientar_aristas(grafo actual).**

Procedure tratar_separar_A(grafo actual, nodo1, nodo2).

1. Encontrar los vecinos de nodo1 y nodo2 que se encuentran en las rutas adyacentes entre nodo1 y nodo2. Colocarlos en dos conjuntos N1 y N2, respectivamente.

2. Remover los hijos actuales conocidos de nodo1 que se encuentran en N1.
   Remover los hijos actuales conocidos de nodo2 que se encuentran en N2.

3. Si la cardinalidad de N1 es mayor que la de N2, intercambie N1 por N2 (swap N1 y N2).

4. Usar N1 como un conjunto de condiciones de C.

5. Hacer una prueba de CI (Independencia Condicional), usando la siguiente ecuación:

$$I(x_i, x_j | C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j | C)}{P(x_i | C) P(x_j | C)}$$

   Asignar a $v = I(x_i, x_j | C)$. Si $v \leq \varepsilon$, retornar “d-separados”.

6. Si C contiene solamente un nodo, ir al paso 8; de otro modo, para cada i, hacer la asignación $C_i = C$ (el i-ésimo nodo de C). $v_i = I(nodo1, nodo2 | C_I)$. encontrar el valor más pequeño v_m de $v_1, v_2, ...$

7. Si $v_m < e$, retornar “d-separado”; de otro modo, si $v_m > v$, ir al paso 8. Si no, asignar $v = v_m$, $C = C_m$. Ir al paso 6.

8. Si N2 no ha sido usado, use N2 como un conjunto de condición C e ir al paso 5; de otro modo retornar “falla”.

Procedure tratar_separar_B(grafo actual, nodo1, nodo2).

1. Encontrar los vecinos de nodo1 y nodo2 que se encuentran en las rutas adyacentes entre nodo1 y nodo2. Colocarlos en dos conjuntos N1 y N2, respectivamente.

2. Encontrar los vecinos de los nodos de N1 que están en las rutas adyacentes entre nodo1 y nodo2, y que no pertenecen a N1. Colocarlos en $N1'$.

3. Encontrar los vecinos de los nodos de N2 que están en las rutas adyacentes entre nodo1 y nodo2, y que no pertenecen a N2. Colocarlos en $N2'$.

4. Si $|N1 + N1'| < |N2 + N2'|$, crear el conjunto $C = N1 + N1'$. Si no, crear el conjunto $C = N2 + N2'$.

5. Hacer un test de independencia condicional usando la siguiente ecuación:

$$I(x_i, x_j | C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j | C)}{P(x_i | C) P(x_j | C)}$$

6. Ejecutar $v = I(nodo1, nodo2 | C)$. Si $v < \varepsilon$, retornar "d-separados".

7. Hacer $C' = C$. Para cada $i \in [1, |C|]$, hacer $C_i = C$ (el i-ésimo nodo de C), $v_i = I(nodo1, nodo2 | c_i)$. Si $v_i < \varepsilon$, retornar "d-separados". Sino, si $v_i < v + e$ entonces $C' = C'$ (el i-ésimo valor de C). (e es un valor pequeño).

8. Si $|C'| < |C|$ entonces, hacer $C = C'$, ir al paso 5;de otro modo, retornar "falla".

Procedure Orientar_aristas(grafo actual)

1. Para cualquier par de nodos $s1$ y $s2$ que no están directamente conectados y donde existe al menos un nodo que es vecino, tanto de $s1$ como de $s2$, encontrar los vecinos de $s1$ y $s2$ que están en las rutas adyacentes entre $s1$ y $s2$. Colocarlos en dos conjuntos N1 y N2, respectivamente.

2. Encontrar los vecinos de los nodos en N1 , que están en las rutas adyacentes entre $s1$ y $s2$, y que no pertenecen a N1. Colocarlos en $N1'$.

3. Encontrar los vecinos de los nodos en N2 , que están en las rutas adyacentes entre $s1$ y $s2$, y que no pertenecen a N2. Colocarlos en $N2'$.

4. Si $|N1+N1'| < |N2+N2'|$, crear el conjunto $C = N1+N1'$. Si no, crear el conjunto $C = N2+N2'$.

5. Hacer un test de independencia condicional usando la siguiente ecuación:

$$I(x_i, x_j | C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j | C)}{P(x_i | C) P(x_j | C)}$$

ejecutar $v = I(nodo1, nodo2 | C)$. si $v < \varepsilon$, ir al paso 8.

6. Hacer $C' = C$. Para cada $i \in [1, |C|]$, hacer $C_i = C$ (el i-ésimo nodo de C), $v_i = I(s1, s2 | c_i)$. si $v_i < v + e$, entonces $C' = C'$ (el i-ésimo valor de C), convertir a $s1$ y $s2$ a parientes de el i-ésimo nodo de C. Si $v_i < \varepsilon$, ir al paso 8. (e es un valor pequeño).

7. Si $|C'| < |C|$ entonces, hacer $C = C'$, ir al paso 5.

8. Volver al paso 1 , hasta que todos los pares de nodos hallan sido examinados.

9. Para cualquier par de nodos a, b, c, si a es pariente de b, b y c son adyacentes, a y c no son adyacentes, y la arista (b,c) no es orientada, hacer que b sea pariente de c.

10. Para cualquier arista (a, b) que no esté orientada, si esta es una ruta directa de a con b, hacer a a pariente de b.

11. Ir al paso 9 hasta que no más aristas puedan ser orientadas.