UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ALGORITMOS DE MINADO DE DATOS COMO UN SERVICIO EN LA NUBE

T E S I S

QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS
(COMPUTACIÓN)

P R E S E N T A:
RUIZ REYES DAVID RICARDO

DIRECTOR DE TESIS:
DR. JAVIER GARCÍA GARCÍA

Facultad de Ciencias

MÉXICO, D. F. MAYO 2015

Agradecimientos

Expreso mi agradecimiento al Dr. Javier García García por aceptar ser mi asesor y por todo su apoyo durante toda la maestría y por abrirme las puertas de su casa.

A mi novia Liliana Vázquez por su ayuda en la redacción.

Al CONACYT por el apoyo económico para realizar los estudios de posgrado.

Y a mi familia en especial a mi madre Patricia.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

Introducción

En la actualidad la gran cantidad de datos digitales que se generan diariamente es enorme y provienen de una amplia variedad de fuentes digitales, esto hace que los datos tengan múltiples formatos, dominios incluso de significado. De una manera no formal podemos clasificar los datos digitales en dos grandes grupos "estructurados" y "no estructurados".

Por estructurados consideramos a todos aquellos datos que se encuentran almacenados o se generan bajo un determinado modelo (llámese modelo entidad-relación, modelo objeto relacional, modelo orientado a objetos) y en los que se tiene un control sobre su dominio, formato, almacenamiento etc., este tipo de datos mayormente es almacenado y manipulado por medio de Sistemas Manejadores de Bases de Datos (SMBD).

Lo datos no estructurados son generados sin ningún tipo de modelo o estructura a seguir. En otras palabras estos datos son todos aquellos que no se encuentran almacenados en algún tipo de SMBD. Por ejemplo los comentarios generados en Twitter no responden a ningún tipo de estructura solo a que deben ser no mayores a 140 caracteres.

Con la generación de esta gran cantidad de datos, ha surgido la necesidad de desarrollar técnicas que permitan analizar de una manera eficiente los datos, estas técnicas se agrupan en el proceso de descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases KDD).

1.1. Minería de Datos y KDD

La minaría de datos es un paso dentro del proceso de descubrimiento de conocimiento (KDD, por sus siglas en inglés), en muchos casos se utilizan indistintamente ambos términos pero existen diferencias, ya que el KDD es un proceso más global que la minería de datos. La materia prima del proceso KDD (también de la minería de datos) son los datos, estos datos tienen un valor relativo, lo que realmente importa es el conocimiento que se pueda inferir a partir de los datos y la capacidad de utilizar este conocimiento. Se define la minería de datos [1] como el proceso de extraer conocimiento útil y comprensible, previamente, desde grandes cantidades de datos almacenados en distintos formatos. En otras palabras, el objetivo fundamental de la minería de datos es hallar modelos a partir de los datos. El proceso de KDD involucra diversas etapas como se muestra en la figura 1.1, estas etapas son:

- Selección: Aquí se ubican las diversas fuentes de las que provienen los datos, que pueden ser SMBD, XML, etc., y se determina de cuáles se obtendrán los datos.
- Limpieza de los Datos: Esta etapa es la que mayor tiempo requiere debido a la calidad del conocimiento descubierto pues no sólo depende del algoritmo de minado de datos que seleccionemos, sino de la calidad de los datos suministrados. Esta etapa representa un reto debido a que cada fuente utiliza diferentes formatos de registro, diferentes modelos de datos, diferentes tipos de errores, datos faltantes, valores que no se ajustan al comportamiento general de los datos, etc.
- Transformación: En esta etapa se consolidan los datos para tener como resultado un conjunto de estos y luego se les aplique los algoritmos de minado de datos. Esta transformación consiste en agrupar todos los datos bajo un mismo modelo, se pueden construir nuevos atributos, discretizar atributos, entre otros; todo con el fin de hacer más fácil el proceso de minería.
- Minería de Datos: El objetivo de esta etapa es el encontrar patrones ocultos. Esto se logra con la construcción de un modelo basado en los datos procesados,

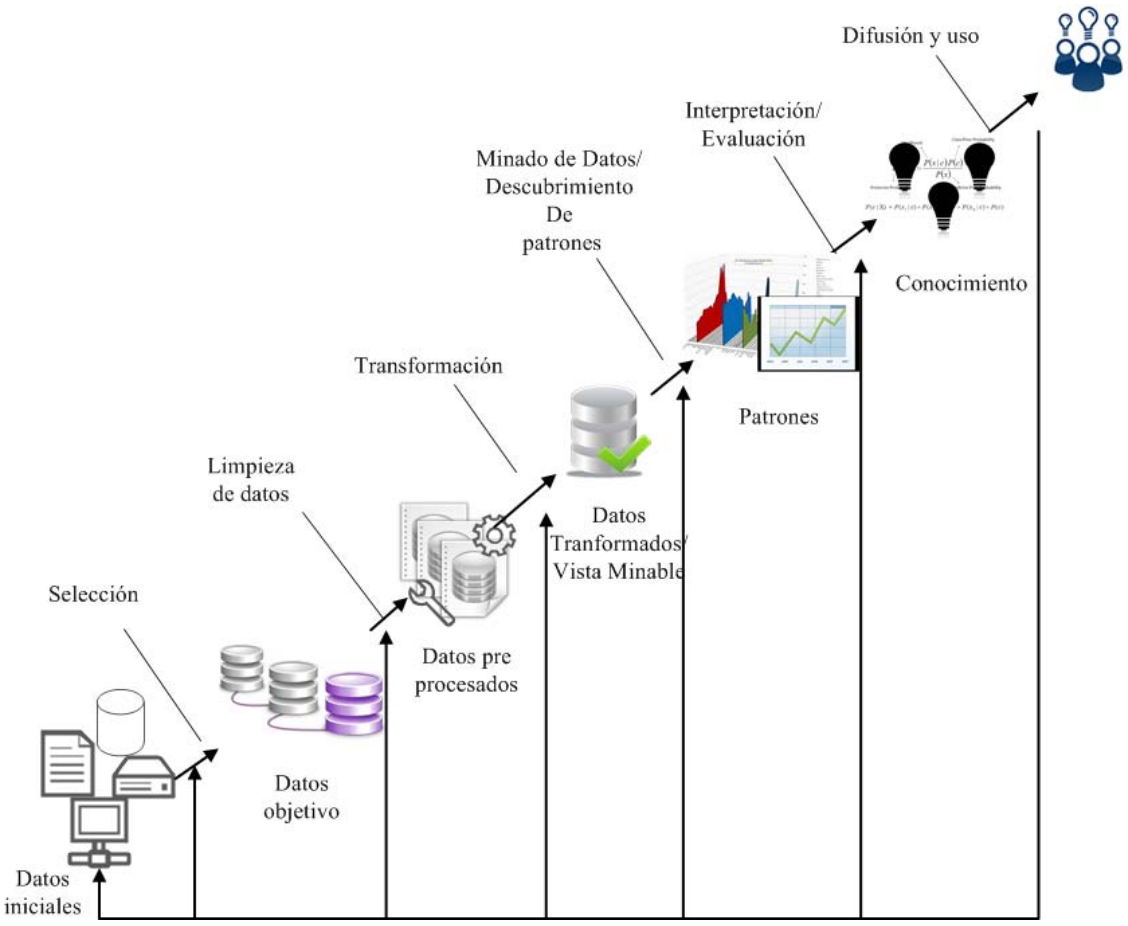


Figura 1.1: Proceso de descubrimiento de conocimiento.

dicho modelo debe ser una descripción de los patrones y las relaciones que existan entre los datos.

- Evaluación e Interpretación: En esta fase se requiere de un experto en el área del cual provienen los datos y será el encargado de evaluar los patrones hallados e interpretarlos para los usuarios finales. Los patrones descubiertos deberán ser precisos, comprensibles y útiles, aunque no es necesario se cumplan las tres características en la misma medida, se puede priorizar la precisión sobre la facilidad de comprensión de un patrón o viceversa, esto dependerá de las circunstancias.

- Difusión y uso: después de hallar los patrones se debe construir un modelo y validarlo. Dicho modelo se puede utilizar para predecir nuevos datos o para validar los que ya se tienen. Se debe monitorear el desempeño del modelo pues los datos que genera éste podrían estar en función del tiempo.

1.2. Objetivos

a. El objetivo de este trabajo es el desarrollo de una aplicación web que permite a un usuario el acceso a algoritmos de minado de datos que un proveedor de servicios pudiera proporcionar en la nube (o sea, un servidor proporcionado por el proveedor de servicios de los algoritmos). Tales algoritmos también se pudieran proveer localmente (en la computadora del usuario - cliente) o en forma híbrida tanto local como en la nube. La selección del modo de procesamiento, en la nube, local o híbrida; se decide por la aplicación de acuerdo al cálculo de una serie de métricas como pueden ser: la velocidad de la red, el poder de procesamiento tanto local como en la nube, el volumen de los datos, etc. De esta forma, la aplicación recomienda cuál es la mejor y más eficiente ejecución. El usuario puede modificar esta recomendación y ejecutar el algoritmo de la forma en que desee.

b. De una manera más puntual los objetivos son:

    i. Implementar optimizaciones al realizar operaciones sobre los datos. Las cuales se efectuarán con la ayuda de la biblioteca de LAPACK , estas optimizaciones se desarollarán con respecto a diversos cálculos que se

aplican a los datos, por ejemplo el cálculo de los valores y vectores propios de una matriz, la inversa de una matriz, etc.

ii. Estudiar algunos algoritmos de búsqueda estocástica para la selección de variables en el contexto de modelos lineales.

iii. Implementar el algoritmo *Stochastic Search Variable Selection* (SSVS).

iv. Experimentar con el software y sus diferentes modos de ejecución, esencialmente en forma local y nube, los algoritmos de minado de datos.

1.3. Motivación

La tendencia actual se dirige a ofrecer diversos servicios de tecnologías de la información por medio de internet o como comúnmente se le llama, en la nube. Los servicios que se proveen son diversos, desde poder crear y editar documentos de textos hasta almacenar documentos en la nube y poder acceder a ellos desde cualquier computadora con acceso a internet. En otras palabras, se trata de proveer todo lo que un servicio informático puede hacer pero como un servicio a través de internet. La motivación es determinar si los algoritmos de minado de datos son factibles de proveerlos como un servicio en la nube, las razones por las cuales se quiere hacer de los algoritmos de minado de datos un servicio disponible por medio de internet se deben a:

a. No se requiere de un software especializado en estadística y probabilidad, solo se requiere de un algoritmo muy en específico y su utilización es esporádica.

b. No se cuenta con el hardware necesario para el procesamiento de los datos de una manera rápida.

c. No se cuenta con los recursos humanos ni económicos para la infraestructura que se requiere para el minado de datos.

d. Se requiere poder acceder a los datos y a los algoritmos desde diversas partes geográficas.

1.4. Aportaciones

Se construyó una aplicación web con algoritmos de minado de datos tanto en la nube (servidor) como locales (en el SMBDR del cliente) los cuales están optimizados con bibliotecas de LAPACK y no requieren de la exportación de los datos fuera del SMBDR. Para precisar, a continuación las principales contribuciones:

a. Utilización de funciones definidas por el usuario programadas en *C* y agregadas al SMBDR,lo anterior se realiza debido a que SQL es un lenguaje diseñado para realizar consultas y no para realizar iteraciones, aunque tiene la posibilidad de utilizar apuntadores para realizar dicha tarea esto no es eficiente, por lo cual con la ayuda de *C* se opera sobre los datos donde se requiere realizar iteraciones.

b. En la implementación, los datos nunca se procesan fuera del SMBDR, el objetivo de lo anterior es evitar la pérdida de integridad de los datos.

c. Utilización de librerías de LAPACK en los procedimientos almacenados para optimizar el procesamiento de los datos. La optimización se realiza en términos de las operaciones que requiere cada algoritmo efectuar sobre los datos, con la ayuda de las funciones que provee LAPACK. Operaciones como regresión de mínimos cuadrados, transpuesta de una matriz, solución de ecuaciones lineales etc., se realizan de una manera más óptima, la versión de LAPACK utilizada en este trabajo es una versión optimizada para procesadores Intel.

d. En cuanto a la implementación de algoritmos, la mayor contribución es la implementación del algoritmo *Stochastic Search Variable Selection,* dicho algoritmo se basa en modelos jerárquicos bayesianos y muestreo de Gibbs para poder hacer la selección de variables de un modelo lineal. Este algoritmo es el de mayor complejidad para implementar dentro del SMBDR debido a la gran cantidad de cálculos que requiere.

1.5. Trabajos Relacionados

El tema de este trabajo es básicamente la optimización de algoritmos de minado de datos al implementarlos con UDFs y SPs y agregarlos al SMBDR, y también el

desarrollo en de una aplicación web que provea los algoritmos de minado de datos como un servicio en la nube. Respecto a la implantación existen diversos trabajos relacionados, uno de estos trabajos [2] describe los beneficios respecto a la optimización al implementar el análisis de componentes principales (PCA por sus siglas en inglés) por medio de UDFs y SP.

La mayoría de los sistemas comerciales actuales al realizar la minería de datos exportan los datos fuera del SMDB para procesar los datos, esta acción implica que se pierdan ventajas como la integridad de los datos, la seguridad y demás. La razón para exportar los datos se debe a la facilidad de desarrollar algoritmos eficientes en lenguajes de programación tradicionales. El modelo relacional con el que están guardados los datos dificulta la integración de algoritmos de minado de datos y el lenguaje SQL no está diseñado para realizar cálculos complejos. Por lo anterior se desarrolló un sistema que se describe en el artículo [3] mismo que puede analizar las tablas dentro de un SMBDR sin la necesidad de exportar los datos. El sistema utiliza UDFs y SPs en C++ y luego los agrega como funciones al SMBDR, de esta forma pueden entonces ser utilizados por medio del lenguaje SQL.[4]

La razón de utilizar UDFs y SPs se debe a que su ejecución se realiza en la memoria principal, y en conjunto con la flexibilidad de C++ para realizar cálculos matemáticos dando como resultado un buen desempeño en al procesamiento de los datos dentro del SMBDR como se describe en [5]

El no extraer los datos del SMBDR y realizar los cálculos con UDFs resulta ser en muchos casos más eficiente que utilizar herramientas o paquetes estadísticos como queda demostrado en el artículo Efficient, donde se realiza la implementación de PCA, la cual se realiza mediante la descomposición de valores singulares SVD, por medio de dos métodos. El primero por medio de consultas por medio de SQL y el segundo utilizando las UDFs.[6].En otro artículo, [7], se procede de una manera similar pero ahora con métodos como la regresión lineal y Naive Bayes. En el artículo [8] se presenta el algoritmo SSVS que es el principal algoritmo a implantar en el software.

CAPÍTULO 2

Marco Teórico

Las bases para la realización de este trabajo se sustentan en dos artículos principalmente, [9] y [8]. El primer artículo es el origen del software de minado de datos como un servicio en la nube.

El segundo artículo es la introducción a un método estadístico, Stochastic Search Variable Selection (SSVS), para seleccionar las variables más importantes dentro de un modelo con un gran número de variables independientes, se debe mencionar que este algoritmo es un algoritmo de KDD que según lo presentado en el capítulo introductorio, pertenece a la fase de Transformación 1.1. En el capítulo dedicado a SSVS se abordará de una manera detallada este algoritmo.

2.1. Software de minado de datos

El software desarrollado en el presente trabajo parte del trabajo desarrollado para el articulo [9], la principal aportación del suscrito fue la incorporación del algoritmo SSVS, el cual representó uno de los mayores retos en este trabajo debido a la gran cantidad de conocimiento previos que se requerían en diversas áreas de probabilidad, estadística y modelos jerárquicos bayesianos. En términos generales el software que se presenta es un servicio web basado en la tecnología de los sistemas manejadores de bases de datos relacionales (SMBDR) donde determinados algoritmos de minado de datos se ofrecen como un servicio. El SMBDR local o del usuario se conecta a otro SMBDR que llamaremos el SMBDR servidor a través de internet, es necesario decir que el servidor se supone con una capacidad de hardware mayor que el del usuario.

Las principales características de este software se listan a continuación:

- Se evita sacar los datos del SMBDR.
- Se evita la transmisión de grandes volúmenes de datos al SMBDR servidor.
- Se incorporan funciones de la biblioteca de LAPACK para la optimización de operaciones en matrices.
- Los algoritmos de minado de datos se programaron en *C* y se incorporaron al SMBDR en forma de UDFs y SPs.
- Se tienen dos tipos de ejecución, local y en la nube.
- Se incorporan optimizaciones en los conjuntos de datos con el uso de estadísticas suficientes.

La razón para decidir evitar en lo posible sacar los datos del SMBDR ss mantener la integridad de los datos, pues al sacarlos del SMBDR se puede llegar a comprometer dicha integridad. Otro motivo es evitar duplicar los datos, es decir se tienen los datos originales en el SMBDR y una copia fuera de este para procesarlos con algún software de minería de datos. Tal vez esto no parezca un gran problema, pero cuando se tiene un gran volumen de datos el tiempo para exportarlos puede ser considerable, además del espacio en disco que consumirá.

Para evitar la transmisión de grandes cantidades de datos el software determina el tiempo en que se procesarían los datos de manera local y en la nube, sugiere la forma con menor tiempo, esto lo realiza al comparar el tiempo que le tomaría procesar un conjunto de datos en el cliente (menor capacidad de procesamiento) y realizar la tarea en el servidor más el tiempo de transmisión de los datos del cliente al servidor.

En cuanto al manejo de bibliotecas de álgebra lineal, LAPACK (Linear Algebre PACKage) es un biblioteca de software libre escrita en FORTRAN para realizar operaciones de álgebra lineal que proporciona una serie de funciones para la soluciones de sistema de ecuaciones lineales, regresión de mínimos cuadrados, PCA (*Principal*

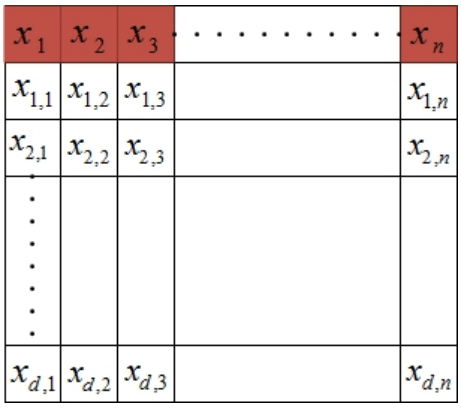


Figura 2.1: Ejemplo de conjunto de datos

component analysis), SVD (*singular value decomposition*). Estas funciones fueron incorporadas en la programación de las UDFs y SPs.

El servicio web fue programado en C, las vistas con ASP.NET, las UDFs y SP también en C todo bajo la plataforma de Microsoft .NET utilizando el entorno de desarrollo visual studio 2010 y el SMBDR que se utilizó fue SQL Server 2008.

2.2. Estadísticas suficientes (NLQ)

Para realizar eficientemente los cálculos de modelos estadísticos multivariables de dimensiones se hace uso de estadísticas suficientes, el uso de estas estadísticas [10] tiene como objetivo concentrar información y esto es debido a que proporciona casi la misma información que la muestra original pero con un conjunto de datos menores.

Sea $X = \{x_1, ..., x_n\}$ un conjunto de datos dado con NLQ puntos, donde cada punto es de dimensión d, el conjunto de datos X es almacenado en la base de datos y se le agrega una columna extra con un identificador único. En otras palabras n es el número de columnas y d es el número de renglones como se muestra en la imagen 2.1

Para obtener las estadísticas suficientes requerimos de;

Sea L un vector columna de $d \times 1$ que contiene la suma lineal de los puntos de

la forma L_i

$$L = \sum_{i=1}^{n} x_i = \begin{bmatrix} \Sigma x_1 \\ \Sigma x_2 \\ \vdots \\ \Sigma x_d \end{bmatrix} \tag{2.1}$$

Sea Q una matriz $d \times d$ que contiene la suma de los cuadrados del producto cruz de los puntos de la forma Q_{ij}

$$Q = XX^T = \sum_{i=1}^{n} x_i x_i^T \tag{2.2}$$

$$Q = \begin{bmatrix} \Sigma x_1^2 & \Sigma x_1 x_2 & \cdots & \Sigma x_1 x_d \\ \Sigma x_2 x_1 & \Sigma x_2^2 & \cdots & \Sigma x_2 x_d \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma x_d x_1 & \Sigma x_d x_2 & \cdots & \Sigma x_d^2 \end{bmatrix} \tag{2.3}$$

La matriz Q 2.3 tiene en la diagonal principal la suma de cuadrados y una suma de productos cruz fuera de la diagonal. La propiedad más importante de L y Q es que son bastante menores que X cuando $d \ll n$. L y Q resumen una gran cantidad de propiedades sobre el conjunto X que pueden ser aprovechadas en diversas técnicas de minería de datos, esto se debe a que en cualquier ecuación donde se presente $\sum x_i$ puede ser sustituida por L y para cuando aparezca XX^T se va a sustituir por Q. Lo anterior se hace con el fin de no utilizar el conjunto de datos X el cual depende de n, ya que L y Q depende de d que es menor que n lo que ayuda a hacer eficientes los cálculos. Por último n es un escalar que es el número de filas o tamaño de nuestros datos.

$$n = \sum_{i=1}^{n} 1 \tag{2.4}$$

2.3. Breve introducción a SSVS

En este apartado se describe brevemente en qué consiste "Stochastic Search Variable Selection" [8] o en español Selección de Variables por Búsqueda Estocástica, que de una manera informal consiste en determinar el subconjunto de las variables independientes que mejor expliquen un modelo lineal.

2.3.1. Selección de variables

En los modelos de regresión lineales se describe una relación entre una variable de salida o variable dependiente y una conjunto de variables explicativas o variable independientes. El modelo de regresión lineal es el siguiente:

$$Y = \beta_1 X_1 + ... + \beta_q X_q + \varepsilon \tag{2.5}$$

Donde $\beta = (\beta_1, \beta_2, ..., \beta_q)$ son los coeficientes de regresión lineal y ε es el error blanco Gaussiano, dicho lo anterior, un problema en los modelos de regresión múltiple es determinar cuáles variables explicativas son las que mejor escriben el modelo. En un primer caso se podría pensar en incluir todas las variables independientes en el modelo, pero esto tiene dos grandes inconvenientes, el primero es que cuando q sea muy grande se puede requerir demasiado tiempo en procesar todas las variables y el segundo radica en que nada garantiza que al incrementar q, las variables independientes, se mejore la calidad del modelo.

2.3.2. Determinar el mejor modelo

Para encontrar el mejor modelo en una primera aproximación se puede iniciar con probar todas las posibles combinaciones de las variables independientes o explicativas, lo que lleva a probar 2^q combinaciones, esto no sería ningún problema para cuando q es relativamente pequeña, pero la función de crecimiento de las posibles combinaciones es exponencial lo que provoca que cuando q represente un valor relativamente grande sea en algunos casos prohibitivo computacionalmente calcular en un tiempo razonable todas las combinaciones.
Lo que se pretende con SSVS es evitar realizar todas las combinaciones y se parte del supuesto que no todos los modelos tienen la misma probabilidad de aparecer y se presenta un modelo jerárquico bayesiano introducido por George and McCulloch. La

característica clave de este modelo es que los autores lo consideran como parte de un largo modelo jerárquico (los conceptos manejados en este apartado serán tratados a profundidad en el capítulo dedicado a SSVS) y cada componente de β se modelan como provenientes de dos distribuciones normales mixtas con diferentes varianzas.

CAPÍTULO 3

Algoritmos de minado de datos

En este capítulo se describirán brevemente los algoritmos de minado de datos implementados en este trabajo, Así como cálculos matriciales para la implementación de estos algoritmos .

3.1. Descomposición en Valores Singulares (SVD)

Una factorización especial para cualquier matriz A de tamaño $m \times n$ es la descomposición en valores singulares (*SVD por sus siglas en inglés*) [11], esta factorización de matrices es bastante útil en temas de álgebra lineal. La forma de descomposición de la matriz A es la siguiente:

$$A = U\Sigma V^T \tag{3.1}$$

La descomposición de A implica una matriz Σ de $m \times n$ de la forma;

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$$

El primer elemento de Σ es una matriz D que sólo contienen una diagonal de $r \times r$ para alguna r que no supere el valor de m y n. U es una matriz unitaria y sus columnas, $u_i, ..., u_n$, son los vectores singulares izquierdos. V es una matriz unitaria y sus columnas, $v_i, ..., v_n$, son los vectores singulares derechos, esto se puede apreciar visualmente en la figura 3.1 Llamaremos valores singulares de la matriz A

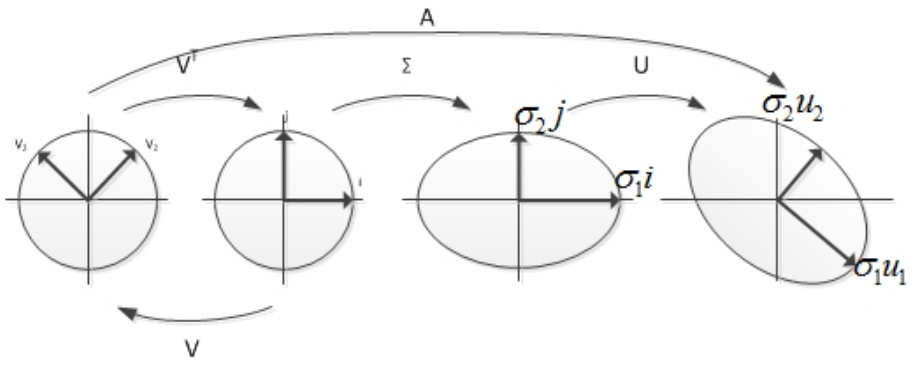


Figura 3.1: SVD Gráficamente

a las raíces cuadradas de los valores propios asociados a la matriz simétrica $A^T A$. Estos valores se denotan por $\sigma_1, \sigma_2, ..., \sigma_n$ y se colocan en orden decreciente.

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$$

Y se tiene que: $\sigma_i = \sqrt{\lambda_i}$ para $1 \leq i \leq n$.

Para entender mejor este concepto utilizaremos el siguiente ejemplo:

Ejemplo 3.1 Encontrar los valores singulares de la matriz

$$A = \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix}$$

La matriz A la podemos descomponer de la siguiente manera: $A = U\Sigma V^T$. Ahora nos ayudaremos de dos ecuaciones útiles para la descomposición de la matriz A

$$A^t A = V\Sigma\Sigma^t V^t$$

$$AV = U\Sigma$$

$$A^t A = \begin{bmatrix} 5 & -1 \\ 5 & 7 \end{bmatrix} \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} = \begin{bmatrix} 26 & 18 \\ 18 & 74 \end{bmatrix}$$

. Ahora debemos obtener el determinante:

$$\det\left(A^t A - \lambda I\right)$$

Sustituyendo los valores queda así:

$$\det\left[\begin{bmatrix} 26 & 18 \\ 18 & 74 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right] = \det\begin{bmatrix} 26-\lambda & 18 \\ 18 & 74-\lambda \end{bmatrix}$$

Calculando el determinante se tiene:

$$\det\left(A^tA - \lambda I\right) = (\lambda^2 - 100\lambda + 1600)$$

Factorizando se obtiene:

$$\det\left(A^tA - \lambda I\right) = (\lambda - 20)(\lambda - 80)$$

Los auto valores o valores propios de A son $\lambda_1 = 80, \lambda_2 = 20$ y los valores singulares son $\sigma_1 = \sqrt{80} = 4\sqrt{5}$ $\sigma_2 = \sqrt{20} = 2\sqrt{5}$
Para $\lambda_2 = 20$

$$A^tA - 20I = \begin{bmatrix} 26-20 & 18 \\ 18 & 54 \end{bmatrix} = \begin{bmatrix} 6 & 18 \\ 18 & 54 \end{bmatrix}$$

Para obtener el vector propio se debe satisfacer el siguiente sistema:

$$\begin{bmatrix} 6 & 18 \\ 18 & 54 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Restando el segundo renglón se tiene:

$$\begin{bmatrix} -12 & -36 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Y al realizar la multiplicación de las matrices y simplificar se tiene:

$$-12v_1 - 36v_2 = 0$$
$$-v_1 - 3v_2 = 0$$

Pero el vector propio debe ser unitario, por lo que se debe sacar su módulo quedando así el primer vector propio de A:

$$V_2 = \begin{bmatrix} \frac{-3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{bmatrix}$$

Para $\lambda_1 = 80$ se sigue el mismo procedimiento anterior lo que nos da el vector propio.

$$V_1 = \begin{bmatrix} \frac{1}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} \end{bmatrix}$$

Para obtener la matriz U se tiene que:

$$u_1 = \frac{Av_1}{\sigma_1} = \frac{1}{4\sqrt{5}} \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} \end{bmatrix} = \begin{bmatrix} \frac{10}{2\sqrt{50}} \\ \frac{10}{2\sqrt{50}} \end{bmatrix}$$

$$u_2 = \frac{Av_2}{\sigma_2} = \frac{1}{2\sqrt{5}} \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} \begin{bmatrix} \frac{-3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{bmatrix} = \begin{bmatrix} \frac{-10}{2\sqrt{50}} \\ \frac{10}{2\sqrt{50}} \end{bmatrix}$$

Ahora se tiene:

$$U = \begin{bmatrix} \frac{10}{2\sqrt{50}} & \frac{-10}{2\sqrt{50}} \\ \frac{10}{2\sqrt{50}} & \frac{10}{2\sqrt{50}} \end{bmatrix} \Sigma = \begin{bmatrix} 4\sqrt{5} & 0 \\ 0 & 2\sqrt{5} \end{bmatrix} V^t = \begin{bmatrix} \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \\ \frac{-3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \end{bmatrix}$$

$$\begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} = \begin{bmatrix} \frac{10}{2\sqrt{50}} & \frac{-10}{2\sqrt{50}} \\ \frac{10}{2\sqrt{50}} & \frac{10}{2\sqrt{50}} \end{bmatrix} \begin{bmatrix} 4\sqrt{5} & 0 \\ 0 & 2\sqrt{5} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \\ \frac{-3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \end{bmatrix}$$

$$\begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} = \begin{bmatrix} \frac{4\sqrt{5}}{\sqrt{2}} & \frac{-2\sqrt{5}}{\sqrt{2}} \\ \frac{4\sqrt{5}}{\sqrt{2}} & \frac{2\sqrt{5}}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \\ \frac{-3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \end{bmatrix}$$

$$\begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix}$$

A continuación se muestran los resultados obtenidos de descomponer la matriz del ejemplo 3.1 utilizando Matlab, únicamente con la intención de que corroboremos que los resultados coinciden.

Listing 3.1: Codigo Matlab

```
>> A=[5  5;-1  7]

A =

     5     5
    -1     7

>> [U,S,V] = svd(A)

U =

   0.707106781186548   0.707106781186547
```

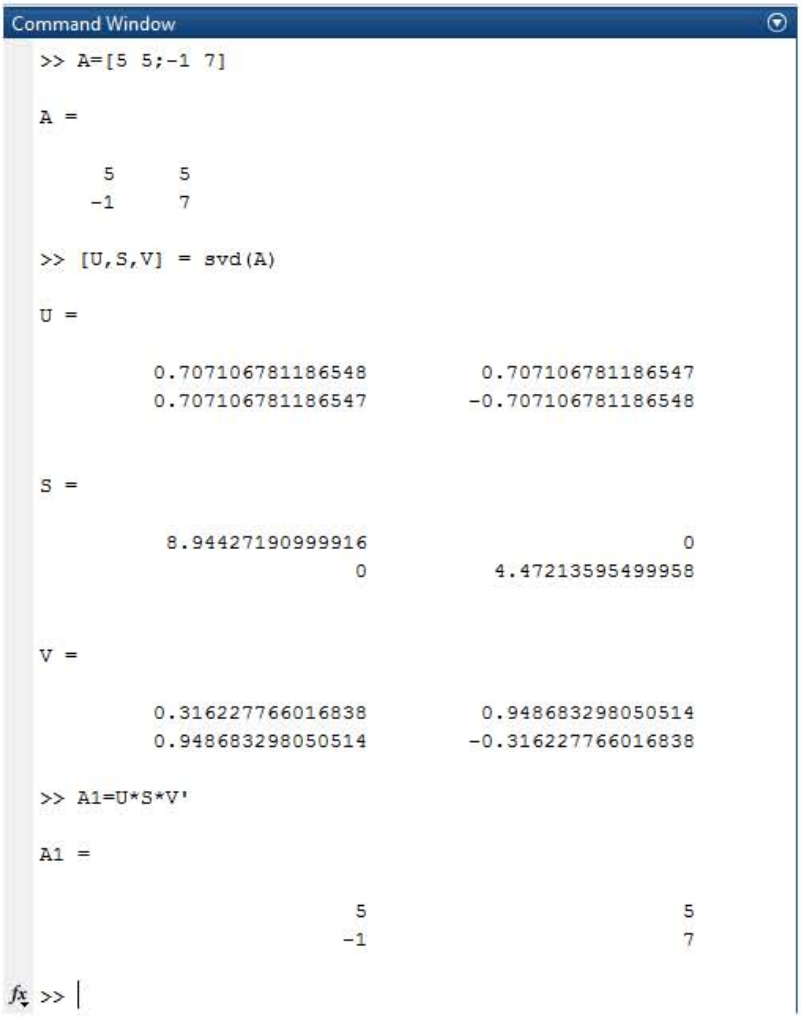


Figura 3.2: Matlab SVD

```
          0.707106781186547   -0.707106781186548

S =
          8.94427190999916                    0
                         0    4.47213595499958

V =       0.316227766016838    0.948683298050514
          0.948683298050514   -0.316227766016838

>> A1=U*S*V'
A1 =
                          5                    5
                         -1                    7
```

3.1.1. Aplicación de SVD

La compresión de imágenes digitales es una de las más importantes y conocidas aplicaciones de la descomposición de valores singulares, se utiliza SVD para trasmitir imágenes digitales de una manera eficiente por medios electrónicos.
El problema principal es saber cuál es la cantidad mínima de información acerca de la imagen que se requiere transmitir sin que se pierda la calidad de la imagen (tener una imagen nítida), ni las partes esenciales de la imagen y se ahorre almacenamiento.

Para aclarar lo anterior pensemos en una matriz A de $m \times n$ que representa una imagen en escala de grises donde cada punto (i, j) puede contener el valor de 0 a 255, que representa el tono de gris de la imagen.
Si se conoce sus componentes principales de la matriz A;

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + ... + \sigma_r u_r v_r^T \tag{3.2}$$

Con los valores singulares no nulos $\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_r > 0$. Los valores singulares más pequeños son los que menor información proveen de la imagen y que pueden ser despreciados. Esto es la ecuación 3.2 se define como:

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + ... + \sigma_k u_k v_k^T \tag{3.3}$$

Donde $k \leqslant r$, por lo que A_k es similar a A, con la variante que únicamente se consideraron los primeros k valores singulares y sus correspondientes vectores singulares.

3.2. Análisis de Componentes Principales

El tener un alto grado de dimensionalidad en los datos que se están analizando se convierte en un problema cuando se intenta obtener información de estos, a dicho problema se le ha llegado a llamar "la maldición de la dimensionalidad". Si tenemos demasiadas dimensiones o variables respecto a la cantidad de muestras, nos encontramos en una situación muy desfavorable debido a que se tienen muchos grados de libertad que pueden provocar que los patrones extraídos no sean fiables. Otro inconvenientes es el visual, sólo se puede representar tres dimensiones.

La reducción de la dimensionalidad se puede realizar por selección de un subconjunto de atributos o variables, o por medio de una transformación que sustituye el conjunto de variables originales o iniciales por otros diferentes.

El análisis de componentes principales (PCA por sus siglas en inglés de "Principal Component Analysis") [12], también llamada método de Karhunen-Loeve es la técnica más conocida para la reducción de la dimensionalidad y consiste en transformar las variables originales $x_1, x_2, ..., x_m$ en otro conjunto de atributos $f_1, f_2, ..., f_p$ donde $p \leqslant m$.

El propósito de esta transformación es que las nuevas variables se generan de manera que son independientes entre sí y, además, se debe lograr que las primeras variables f deben ser las que sean más relevantes, las que contengan más información. En otras palabras f_1 debe ser la variable o el atributo más relevante del nuevo conjunto entonces se tiene que f_1 es más relevante que f_2,f_2 es más relevante que f_3 y así sucesivamente, siendo f_p la variable menos relevante. Lo anterior permite seleccionar los k primeros atributos, y así garantizar que se seleccionarán las variables más relevantes. Si se desea representar los datos mediante un gráfico 3D se tomará $k = 3$.

El principal objetivo de PCA es convertir un vector original $x_1, x_2, ..., x_m$ en otro

vector $f_1, f_2, ..., f_p$, si suponemos que $m = p$ se puede representar de la siguiente manera:

$$\underset{m\times 1}{\bar{f}} = \begin{bmatrix} \bar{a}_1^T \\ \bar{a}_2^T \\ \vdots \\ \bar{a}_m^T \end{bmatrix} \underset{(m\times 1)}{\overline{x}} = \underset{(m\times m)}{A^T} \underset{(m\times 1)}{\overline{x}} \tag{3.4}$$

Se necesita obtener una matriz de $m \times m$ de coeficientes, tal que multiplicada por cada vector de atributos originales se obtenga otro vector de atributos en el nuevo espacio de dimensiones. ¿Cómo calcular los vectores de coeficientes $\bar{a}_i$ de tal manera que se consiga que los primeros atributos de los nuevos vectores f sean más relevantes que el resto?

La forma de conseguir lo anterior en PCA es asumir que lo que se necesita es que la varianza de los nuevos atributos sea mayor para los primeros atributos que para los últimos. Para llevar acabo esto se requiere calcular en primer lugar la matriz de covarianzas:

$$\underset{(m\times m)}{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\bar{u}_i) (\bar{u}_i)^T \tag{3.5}$$

Donde:
S: es la matriz de varianza-covarianza, donde la diagonal contiene la varianza y los elementos que se encuentran fuera de la diagonal son la covarianza de cada elemento de $S_{i,j}$.
n: es el número de ejemplo o muestras que tenemos de nuestros datos.
m: es el número de atributos o variables.
$\bar{u}_i$: es el vector de diferencias que se calcula restando cada ejemplo i(vectores de atributos $x_1, x_2, ..., x_m$) con la media de los atributos de nuestros datos.
Esto se expresa como sigue;

$$\bar{u}_i = \bar{x}_i - \bar{m} \tag{3.6}$$

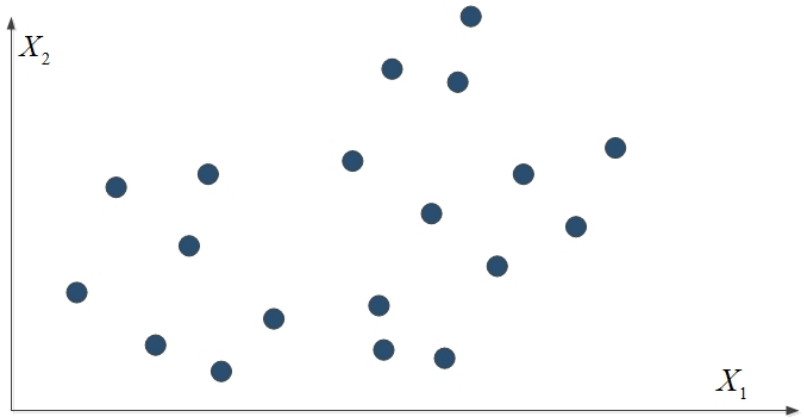


Figura 3.3: Datos Originales

$$\bar{m} = \frac{1}{n}\sum_{i=1}^{n} \bar{x}_i \tag{3.7}$$

Donde $\bar{u}_i$,$\bar{x}_i$ y $\bar{m}$ son vectores de $m \times 1$. Después de haber calculado la matriz de varianza-covarianza S, se debe obtener los valores propios de la matriz de S

$$\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_m \geqslant 0 \tag{3.8}$$

Los valores propios son un conjunto de escalares asociados a un sistema lineales de ecuaciones y con ayuda de los vectores propios de S se calcula los vectores $\bar{a}_i$ que son llamados ejes principales. Es importante mantener el orden de los vectores propios que se obtiene con cada valor propio, ya que este orden es el que asegura que el atributo f_1 sea el más importante y así sucesivamente.

Para comprender mejor lo que PCA realiza es conveniente presentarlo de una manera gráfica, supongamos que se tiene un conjunto de datos como se muestra en la imagen 3.3, estos datos están referenciados a los vectores X_1 y X_2, ambas variables tienen la misma cantidad de información.

Lo que se tiene que hacerse con respecto a nuestros datos es referenciarlos a nuestras nuevas variables que llamaremos componentes principales, esto se verá reflejado en la ecuación 3.9, la cual es una simplificación de la ecuación 3.4 para nuestro

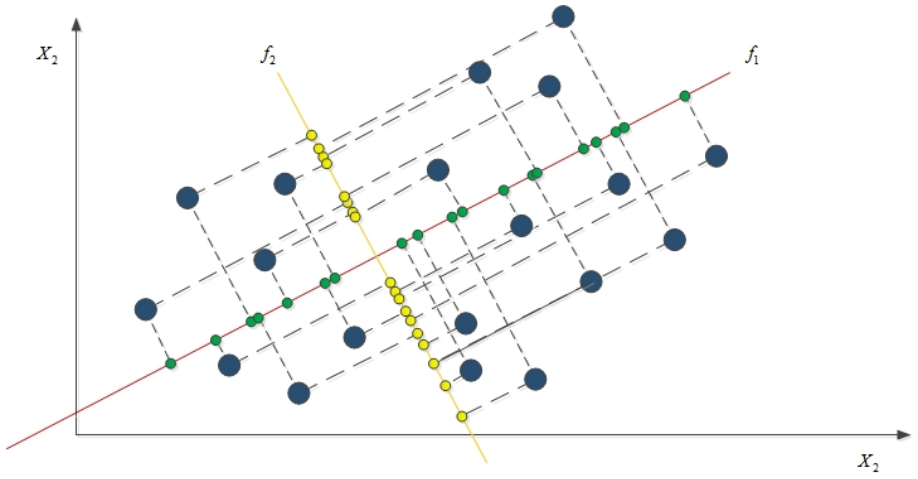


Figura 3.4: PCA

ejemplo.

$$\begin{aligned} f_1 &= a_{1,1}X_1 + a_{1,2}X_2 \\ f_2 &= a_{2,1}X_1 + a_{2,2}X_2 \end{aligned} \tag{3.9}$$

Al obtener f_1 y f_2 (línea roja con los puntos verdes y línea anaranjada con los puntos amarillos respectivamente) visibles en la imagen 3.4, se puede ver claramente cómo la componente f_1 es la que tiene la mayor cantidad de información (mayor varianza), esto se debe a que los puntos verdes los cuales son la proyección de los datos originales sobre la componente f_1. Nótese que estos puntos se encuentran más espaciados los unos de los otros. En el caso de la proyección sobre la componente f_2 se puede ver que contienen menor información (menor varianza) los puntos amarillos pues están muy juntos los unos de los otros e incluso en muchos casos, varios puntos ocupan el mismo lugar sobre la componente f_2.

3.3. K medias

El algoritmo de K medias (*K means*)[13] es uno de los algoritmos más viejos y ampliamente usados de agrupamiento por vecindad en el que se parte de un número determinado de prototipos o clases y un conjunto de ejemplos a agrupar.

La idea principal de K medias es situar a los prototipos o centroides en el espa-

cio, de forma que los datos pertenecientes al mismo prototipo tengan características similares. Una vez que los prototipos han sido correctamente situados, cada nuevo ejemplo o muestra es comparado con éstos y asociados a aquél al que esté más próximo, esto mediante la comparación de la medida de distancia, usualmente se utiliza la distancia euclidiana.

Las regiones son definidas minimizando la suma de la distancia cuadrática entre cada muestra de entrada y el centroide correspondiente a su clase, que está representado por el prototipo correspondiente. El algoritmo de K medias tiene dos variantes: por lotes y en línea. El primero se usa cuando todos los datos a clasificar están disponibles desde el inicio, mientras el segundo se utiliza cuando no se dispone de todos los datos desde el principio y se añaden ejemplos.

Cuando se dispone de todos los datos desde el inicio, se deben seleccionar al azar los prototipos así como el número de clases en las que se quiere clasificar los datos, esto quiere decir que cada clase debe disponer de al menos un ejemplo que en este caso es el prototipo.

Como todos los datos están disponibles, los centros de cada partición se calculan como la media de los ejemplos pertenecientes a la clase. Conforme se clasifica cada una de las muestras, algunas muestras cambian de clases por lo tanto se debe recalcular los centroides en cada paso.

El algoritmo tiene una fase de entrenamiento, esta fase puede variar en el tiempo de ejecución, este tiempo de ejecución depende del número de muestras a clasificar; pero una vez clasificados todos los datos, la clasificación de nuevos datos se hace de una manera muy rápida, debido a que la comparación de la distancia de cada nueva muestra sólo se realiza con los prototipos.

La forma en que opera el algoritmo es la siguiente:

- Se eligen al azar los prototipos y el número de clases en que queremos dividir.
- Se calcula, para cada ejemplo x_k, el prototipo más próximo A_g y se incluye en

la clase de dicho prototipo.

$$A_g = \text{mín}\left\{d\left(x_k, A_i\right)\right\} \tag{3.10}$$

- Después de haber procesado todas la muestras, cada prototipo A_k tendrá un conjunto de muestras a los que representa:

$$l\left(A_k\right) = \left\{x_{k_1}, x_{k_2}, ..., x_{k_n}\right\} \tag{3.11}$$

- Se desplaza el centroide hacia el centro de masa de su conjunto de muestras.

$$A_k = \frac{\sum_{i=1}^{m} x_{k_i}}{m} \tag{3.12}$$

- Se repite el procedimiento hasta que ya no se muevan los centorides

Con el uso de K medias el espacio de muestras de entradas se clasifica en k clases y el prototipo de cada clase se encuentra en el centro de la clase. Los centroides se determinan con el objetivo de minimizar las distancias cuadráticas euclidianas entre los patrones de entrada y el centroide más cercano, es decir se minimiza el valor de G:

$$G = \sum_{i=1}^{k} \sum_{n=1}^{m} M_{i,n} d(x_n - A_i)^2 \tag{3.13}$$

Donde m es el conjunto de patrones , d es la distancia euclidiana, x_n es la muestra de entrada n,A_i es el prototipo de la clase i, y $M_{i,n}$ es la función de pertenencia de la muestra n a la región i de forma que vale 1 si el prototipo A_i es el más cercano a la muestra x_ny 0 en caso contrario , es decir:

$$M_{i,n} = \begin{cases} 1 \text{ si } d\left(x_n - A_i\right) < d\left(x_n - A_s\right) \\ 0 \text{ en caso contrario} \end{cases} \tag{3.14}$$

Para entender mejor cómo funciona el algoritmo de K medias, supongamos que tenemos el conjunto de datos 3.5, que son diez llantas con sus respectivos puntajes de desempeño en dos tipos de terreno.

En la imagen 3.6 se puede ver una representación gráfica de las 10 llantas, lo que se quiere es agrupar en tres clases las 10 diferentes llantas en otras palabras $k = 3$.

Lo siguiente es seleccionar al azar nuestros tres prototipos que serán los centroides de nuestras clases, como este ejemplo es ilustrativo y ayudándonos de la figura 3.6,

Llanta	Mojado	Seco
LL1	2	10
LL2	2	5
LL3	8	4
LL4	3	8
LL5	7	5
LL6	6	4
LL7	1	2
LL8	4	9
LL9	6	2
LL10	1	9

Figura 3.5: Datos

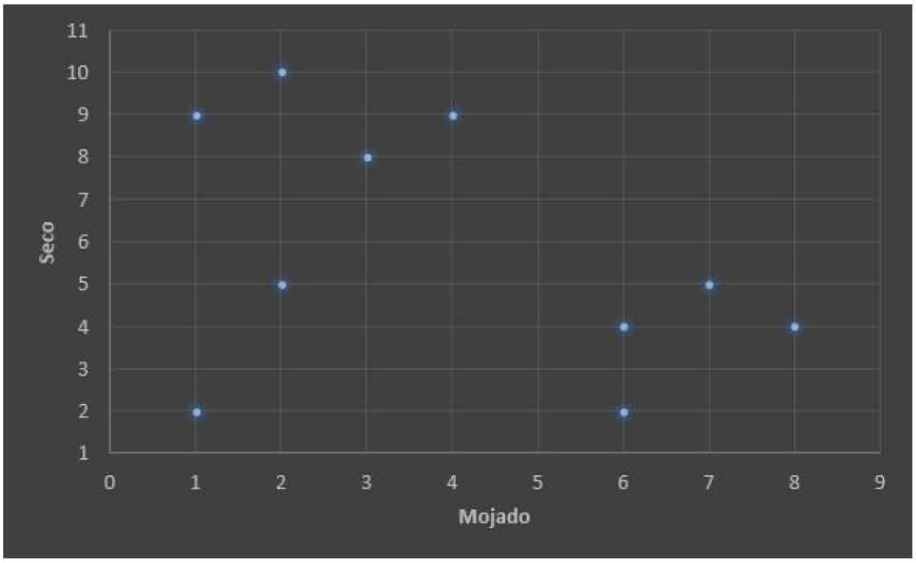


Figura 3.6: Representación gráficamente de los datos.

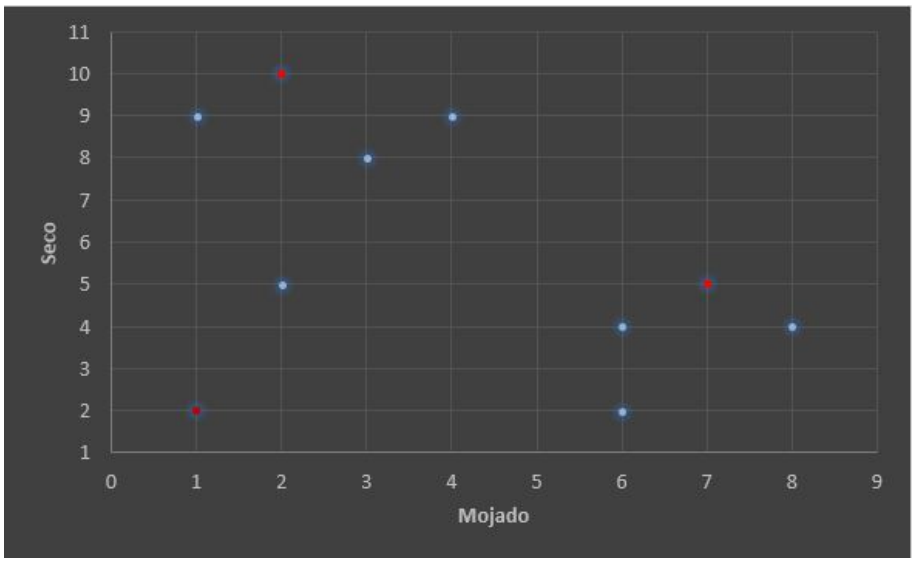


Figura 3.7: Representación gráficamente centroides.

los centroides serán $c_1 = (1, 2)$, $c_2 = (7, 5)$ y $c_3 = (2, 10)$ o lo que sería lo mismo c_1, c_2 y c_3 igual llanta siete, llanta cinco y llanta uno respectivamente. Lo centroides quedarían como se ve en la figura 3.7 , marcados con rojo

El siguiente paso es obtener la distancia de cada punto con cada uno de los centroides, a continuación se muestra un ejemplo:

$$d_{C1,LL1} = \sqrt{(1-2)^2 + (2-10)^2} = \sqrt{65} = 8{,}06225$$
$$d_{C1,LL2} = \sqrt{(1-2)^2 + (2-5)^2} = \sqrt{10} = 3{,}16227$$
$$\vdots$$
$$d_{C1,LL10} = \sqrt{(1-1)^2 + (2-9)^2} = 7$$

La ecuación anterior se calcula ahora para c_2 y c_3 los resultados completos se muestran en la figura 3.8, las celdas marcadas en rojo representan la mínima distancia del punto en cuestión con una determinada clase. La distancia mínima es la que

Llanta	Mojado	Seco	C1=LL7	C2=LL5	C3=LL1	Clase
LL1	2	10	8,0622577	7,0710678	0	C3
LL2	2	5	3,1622777	5	5	C1
LL3	8	4	7,2801099	1,4142136	8,4852814	C2
LL4	3	8	6,3245553	5	2,236068	C3
LL5	7	5	6,7082039	0	7,0710678	C2
LL6	6	4	5,3851648	1,4142136	7,2111026	C2
LL7	1	2	0	6,7082039	8,0622577	C1
LL8	4	9	7,6157731	5	2,236068	C3
LL9	6	2	5	3,1622777	8,9442719	C2
LL10	1	9	7	7,2111026	1,4142136	C3

Figura 3.8: Cálculo de distancias.

nos indicará a qué clase pertenece cada punto, en la columna "Clase" se muestra a qué clase pertenece cada punto o para nuestro ejemplo cada llanta.

Nuestros datos quedan clasificados de la siguiente manera $C_1 = \{LL2, LL7\}$, $C_2 = \{LL3, LL5, LL6, LL9\}$ y $C_3 = \{LL1, LL4, LL8, LL10\}$. Lo siguiente es calcular los nuevos centroides.

$$C_1 = \left(\frac{2+1}{2}, \frac{5+7}{2}\right) = (1{,}5, 6)$$
$$C_2 = \left(\frac{8+7+6+6}{4}, \frac{4+5+4+2}{4}\right) = (6{,}75, 3{,}75)$$
$$C_3 = \left(\frac{2+3+4+1}{4}, \frac{10+8+9+9}{4}\right) = (2{,}5, 9)$$

La figura 3.9 muestra en color rojo las nuevas posiciones de los centroides, luego se procede a calcular las distancias de estos nuevos centroides con cada uno de los puntos y determinar si alguno de los puntos cambió de clase.

Recordemos que nuestro ejemplo es únicamente ilustrativo y los prototipos se seleccionaron de una manera conveniente, en este caso ningún punto cambia de clase, como se ve en la figura 3.10, por lo que el algoritmo termina. En caso de que algún punto hubiera cambiado de clase se volverían a calcular los centroides de cada

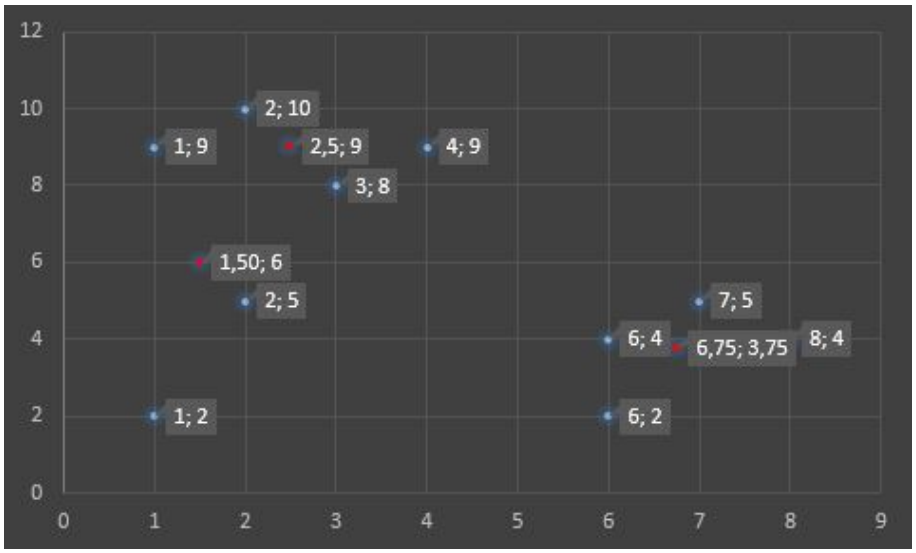


Figura 3.9: Nuevos centroides.

Llanta	Mojado	Seco	C1=LL7	C2=LL5	C3=LL1	Clase
LL1	2	10	4.031129	7.85016	1.11803399	C3
LL2	2	5	1.118034	4.91172	4.03112887	C1
LL3	8	4	6.800735	1.27475	7.43303437	C2
LL4	3	8	2.5	5.66789	1.11803399	C3
LL5	7	5	5.59017	1.27475	6.02079729	C2
LL6	6	4	4.924429	0.79057	6.10327781	C2
LL7	1	2	4.031129	6.01041	7.15891053	C1
LL8	4	9	3.905125	5.92663	1.5	C3
LL9	6	2	6.020797	1.90394	7.82623792	C2
LL10	1	9	3.041381	7.78621	1.5	C3

Figura 3.10: Cálculo de distancias.

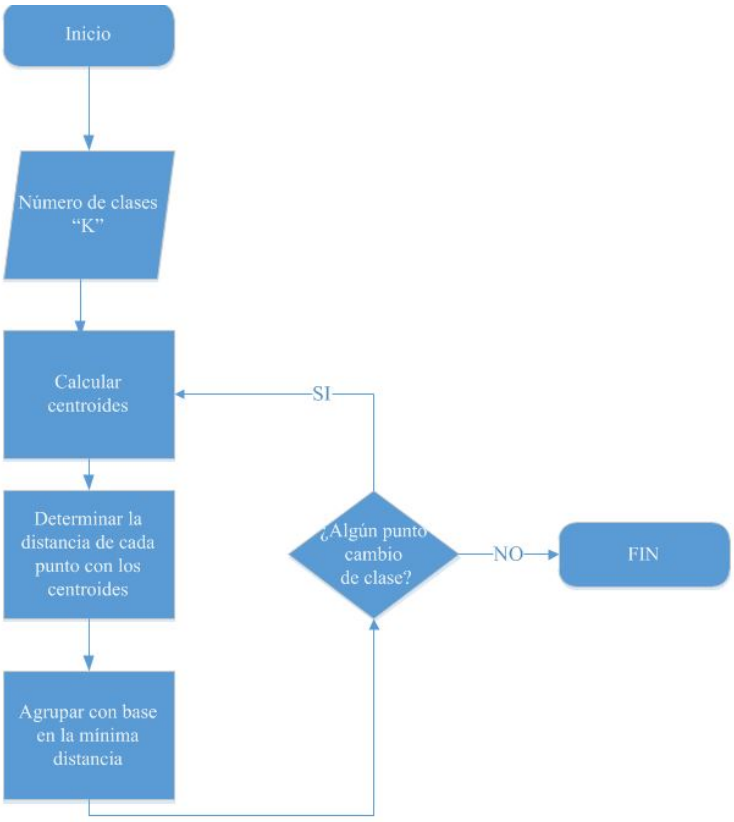


Figura 3.11: Algoritmo K medias.

clase y así hasta que ningún punto cambie de clase como se ve en el diagrama.

3.4. Bayesiano Ingenuo (Naïve Bayes)

A continuación se describe uno de los modelos más simples de clasificación basado en redes bayesianas, que se caracteriza por ser bastante práctico, permite interpretar el funcionamiento de otros métodos en términos probabilísticos.[14]

3.4.1. Teorema de Bayes

En teoría de probabilidades, el teorema de Bayes es la regla más básica para poder realizar inferencias. El teorema de Bayes nos permite actualizar las creencias que tenemos sobre algún fenómeno o conjunto de fenómenos con la ayuda de nueva evidencia u observaciones.

En otras palabras pasamos de la probabilidad *a priori* P(eventos) a la probabilidad *a posteriori* P(eventos—observaciones). La probabilidad *a priori* la interpretamos como la probabilidad inicial, es decir, la probabilidad que fijamos sin saber nada más sobre un evento. Por ejemplo, la probabilidad de que un conductor sufra un accidente al regresar a casa puede ser del .15. La probabilidad *a posteriori* es la que se obtiene después de conocer cierta información, puede verse como un refinamiento de nuestro conocimiento. Continuando con el ejemplo anterior si ahora sabemos que el conductor está alcoholizado se tiene que la probabilidad *a posteriori* será $P\left(accidente|alcoholizado\right)$ que por experiencia sabemos será mayor a la probabilidad $P\left(accidente\right)$. El teorema de Bayes se presenta en la ecuación 3.16

$$P\left(h|\,O\right) = \frac{P\left(O|h\right)P\left(h\right)}{p\left(O\right)} \tag{3.16}$$

Donde:

- $P\left(h|\,O\right)$ es la probabilidad *a posteriori* de h, probabilidad de que h sea cierta después de observar O.

- $P\left(O|h\right)$ es la probabilidad *a posteriori* de O, probabilidad de observar el conjunto O en un universo donde se verifica la hipótesis h. A esta probabilidad se le conoce como la verosimilitud de que la hipótesis h haya producido el conjunto de observaciones.

- $P(h)$ es la probabilidad *a priori* de la hipótesis h, probabilidad de h sin ninguna observación.

- $P\left(O\right)$ es la probabilidad *a priori* de O, probabilidad de observar O sin saber qué hipótesis se verifica.

Se plantea el caso de clasificación donde se tiene una variable que es la clase (C) y un conjunto de variables predictoras o atributos $\{X_1, ..., X_n\}$, la ecuación 3.16 quedaría así:

$$P(C|X_1...X_n) = \frac{P\left(X_1...X_n|C\right)P\left(C\right)}{P\left(X_1...X_n\right)} \tag{3.17}$$

Si C tiene k posibles valores $\{c_1, ...c_k\}$, lo que se debe hacer es determinar el valor más aceptable y devolverlo como el resultado de la clasificación. En el marco bayesiano la hipótesis más aceptable es la que tiene máxima probabilidad *a posteriori* o MAP (del inglés, *maximun a posteriori*). La clase a devolver está dada por la ecuación

$$\begin{aligned} c_{MAP} &= \arg\max_{c\in\Omega} p\left(c|X_1, .., X_n\right) \\ c_{MAP} &= \arg\max_{c\in\Omega} \frac{p\left(X_1, .., X_n|c\right)p(c)}{p\left(X_1, .., X_n\right)} \\ c_{MAP} &= \arg\max_{c\in\Omega} p\left(X_1, .., X_n|c\right)p(c) \end{aligned} \tag{3.18}$$

Donde Ω representa el conjunto de todos los valores que puede tomar la variable C.

CAPÍTULO 4

Búsqueda Estocástica de Selección de Variables(SSVS)

En este capítulo se describirá de una manera detallada el algoritmo *Stochastic Search Variable Selection*, (SSVS por sus siglas en inglés).[8]

4.1. Introducción

Al proponer un modelo de regresión múltiple se tiene el problema de determinar qué predictores incluir. Consideremos un modelo lineal con n observaciones y p variables predictoras

$$Y = X\beta + \varepsilon \tag{4.1}$$

Donde Y es un vector respuesta de $n \times 1$

X es una matriz de orden $n \times p$

ε es el error aleatorio.

Al intentar modelar la relación que existe entre la variable dependiente Y y el conjunto de variables predictoras $\mathrm{X}_1, X_2...., X_p$, surge el problema de determinar que variables predictoras se deben seleccionar. En otras palabras dado la variable dependiente Y y un conjunto de variables predictoras,$\mathrm{X}_1, X_2...., X_p$, se debe deter-

minar un subconjunto de estas variables predictoras que expliquen mejor el modelo, es decir el problema radica en ajustar el mejor modelo de la forma:

$$Y = X_1^*\beta_1^* + X_2^*\beta_2^* + \cdots + X_q^*\beta_q^* + \varepsilon \tag{4.2}$$

Donde $X_1^*, X_2^*..., X_q^*$ es un subconjunto de variables seleccionadas del conjunto original.

El número de subconjuntos posible dependerá el tamaño de p, si este número es muy pequeño no habría mucho problema en probar todas las posibles combinaciones, al hacer esto se calcularía 2^pcombinaciones, el conjunto potencia. Por ejemplo supongamos que tenemos 10 variables, tendríamos $2^{10} = 1024$posibles conjuntos con diferentes combinaciones de variables predictoras, este número es todavía computacionalmente manejable en un tiempo razonable, pero en la actualidad hay diversos fenómenos con más que simplemente 10 variables, predicción del estado del tiempo, modelos económicos, entre otros. Como se puede ver claramente el crecimiento de las posibles combinaciones es exponencial, para una P=20 se tiene $2^{20} = 1048576$posibles modelos, con solamente duplicar el número de variables.

Lo que se pretende con SSVS es evitar calcular todos los modelos para determinar cuál es el que mejor se ajusta al modelo (4.2).

Para realizar lo anterior SSVS asigna una distribución de probabilidad al conjunto de todos los posibles modelos de regresión de tal manera que los modelos prometedores se les da una mayor probabilidad, después se utiliza el muestreo de Gibbs para simular una muestra correlacionada de esta distribución. Los modelos más prometedores son entonces fácilmente identificados como aquellos que aparecen con más frecuencia en la muestra. La clave del potencial de SSVS es que el muestreo de Gibbs puede utilizar para simular una muestra informática de la parte *a posteriori* de forma rápida y eficiente.

4.2. Un modelo jerárquico Bayesiano para la selección de variables

Este modelo es propuesto por George and McCulloch(1993) [8] y determina que la relación entre la variable dependiente Y y el conjunto de potenciales predictores $X_1, X_2...., X_p$es un modelo lineal normal como sigue

$$Y \sim N_n \left(X\beta, \sigma^2 I_n\right) \tag{4.3}$$

Donde Y es $n \times 1$, N_n denota una distribución normal multivariable de n dimensiones, $X = [X_1, ..., X_p]$ es una matriz de $n \times p$, $\beta = (\beta_1, ..., \beta_p)'$, σ^2 es un escalar y I_n es la matriz identidad de $n \times n$.

Ambos β y σ^2 son desconocidos. Para el modelo jerárquico bayesiano (4.3) , para seleccionar un subconjunto de variables predictoras se inicializa con cero todos los β_i de las variables predictoras que no fueron seleccionadas.

De una manera informal se puede describir el modelo (4.3) de la siguiente manera;

La variable Y tiene una distribución normal multivariante con media $X\beta$ y varianza $\sigma^2 I_n$. Los β_i corresponde a los coeficientes de los regresión linea del modelo (4.1) para cada una de las variables independientes.

Cada elemento de β_i se modela usando una mezcla de dos distribuciones normales con diferentes varianzas y al incorporar γ como variable latente que puede tomar el valor $\gamma = 0$ o $\gamma = 1$ lo cual se puede expresar como

$$P\left(\beta_i|\gamma_i\right) = \left(1 - \gamma_i\right) N\left(0, \tau_i^2\right) + \gamma_i N\left(0, c_i^2 \tau_i^2\right) \tag{4.4}$$

$$P\left(\gamma_i = 1\right) = 1 - P\left(\gamma_i = 0\right) = w_i \tag{4.5}$$

Al asignar $\gamma_i = 0$ la ecuación (4.4) queda de la siguiente manera;

$$P\left(\beta_i|\gamma_i\right) \sim \left(1 - \gamma_i\right) N\left(0, \tau_i^2\right) \tag{4.6}$$

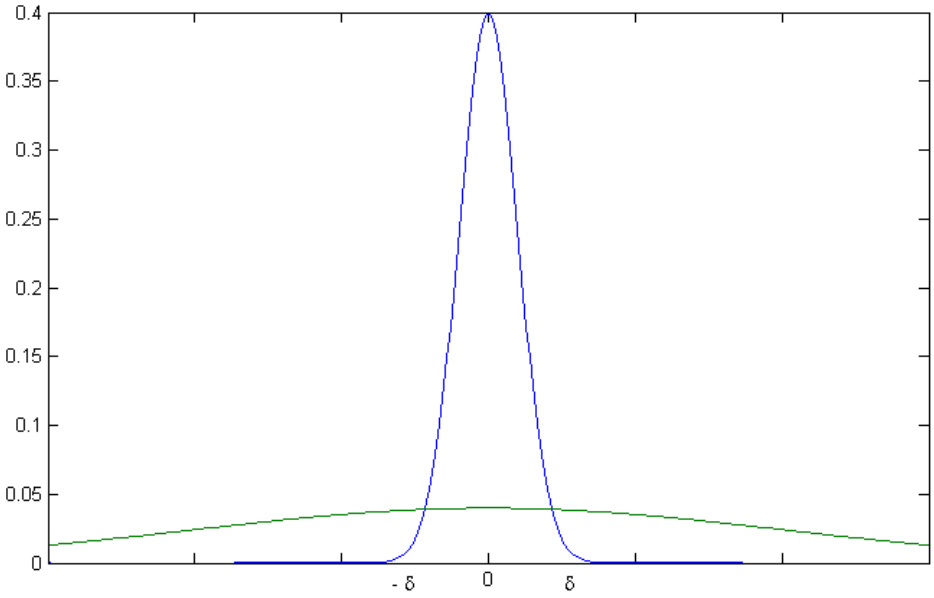


Figura 4.1: Distribuciones Mixtas

Y al asignar $\gamma_i = 1$ queda de la siguiente manera;

$$P\left(\beta_i|\gamma_i\right) \sim \gamma_i N\left(0, c_i^2\tau_i^2\right) \tag{4.7}$$

En la ecuación (4.4) se observan dos hiperparámetros c_i y τ_i, estos hiperparámetros se ajustarán para mejorar el desempeño de SSVS. Se asignará un valor a $\tau_i > 0$ pero pequeño de manera que si $\gamma_i = 0$, entonces β_i sería probablemente demasiado pequeña que podría ser considerada como cero. Al asignar un valor a $c_i > 1$ siempre y de manera que si $\gamma_i = 1$ entonces β_i es diferente de cero y deberá ser probablemente incluida en el modelo final. Basado en lo anterior $w_i = P\left(\gamma_i = 1\right)$ en 4.5, puede ser interpretado como la probabilidad *a priori* de que β_i sea diferente a cero, o en otras palabras de que X_i deba ser incluida en el modelo final.

La probabilidad *a priori* de cada componente de β puede ser obtenida si se usa una mezcla normal multivariante *a priori*.

$$\beta|\gamma \sim N_P\left(0, D_\gamma R D_\gamma\right) \tag{4.8}$$

Donde $\gamma = (\gamma_1, ..., \gamma_p)$ es un vector de variables latentes, R es la matriz de correlación *a priori* y D_γ es la matriz diagonal con i elementos iguales a $(1 - \gamma_i)\,\tau_i +$

$\gamma_i c_i \tau_i$. La mezcla a priori sobre β es obtenida con cualquier $P(\gamma)$ a priori no trivial en los 2^p posibles valores de γ.

El modelo de Bernoulli (4.5) se obtiene como cualquier distribución marginal discreta. Para obtener la γ el a priori se dispone de la siguiente ecuación;

$$P(\gamma) = \prod_{i=1}^{p} w_i^{\gamma_i}(1 - w_i)^{(1-\gamma_i)} \tag{4.9}$$

Y para finalizar nuestro modelo jerárquico (4.3) se debe obtener el valor *a priori* de σ^2 el cual se calcula al utilizar el a priori del conjugado de la inversa gama según [8]

$$\sigma^2|\gamma \sim IG\left(\frac{v_\gamma}{2}, \frac{v_\gamma \lambda_\gamma}{2}\right) \tag{4.10}$$

Se debe tener en cuenta que v_γ y λ_γ pueden depender de γ para incorporar dependencia entre β y σ^2. La obtención de v_γ y λ_γ se comentara en la sección 4.5.

La principal razón para incrustar un modelo lineal normal 4.3 en un modelo jerárquico mixto es la obtención de la distribución *a posteriori* γ,

$$P(\gamma|Y) \propto P(Y|\gamma)P(\gamma) \tag{4.11}$$

.

que provee información muy útil sobre la selección de variable. Basado en Y, el valor *a posteriori* $P(\gamma|Y)$ actualiza las probabilidades previas en cada uno de los 2^p posibles valores de γ. $P(\gamma)$ puede ser interpretada como la probabilidad *a priori* que los $x's$ correspondientes a un componente de γ diferente de cero deban ser incluidos en el modelo. Se identifica cada γ con el modelo donde $\gamma_i = 1$ por lo tanto x_i es incluido, los γ con una probabilidad alta *a posteriori* $P(\gamma|Y)$ identifican los modelos más prometedores, es decir los modelos que son más apoyados por los datos y por la distribución *a priori* de las estadísticas calculadas.

4.3. Seleccionando $P(\gamma)$

Para seleccionar el valor de $P(\gamma)$ se debe tomar en cuenta toda la información *a priori* disponible acerca de los subconjuntos de $x_i, ..., x_p$ que deben ser incluidos en el modelo final, lo cual puede ser muy difícil con 2^p posibles elecciones. El valor *a priori* de $P(\gamma) \equiv \frac{1}{2^p}$ es un caso especial de la ecuación 4.17 donde cada x_i tiene la misma probabilidad $\left(p_i = \frac{1}{2}\right)$ de ser incluida.

4.4. Seleccionando τ y c

El uso de valores altos de la probabilidad *a posteriori* para identificar los modelos más prometedores sugiere una estrategia para seleccionar los valores de τ_i y c_i. Sea δ_i el punto de intersección de $N\left(0, \tau_i^2\right)$ y $N\left(0, c_i^2\tau_i^2\right)$ ver figura 4.1

Cuando $|\beta_i| \leqslant \delta_i$ corresponde a la zona donde $N\left(0, \tau_i^2\right)$ domina a $N\left(0, c_i^2\tau_i^2\right)$, cuando $|\beta_i| > \delta_i$ corresponde a la zona donde $N\left(0, c_i^2\tau_i^2\right)$ domina a $N\left(0, \tau_i^2\right)$. Una gran probabilidad *a posteriori* de γ en $P\left(\gamma|Y\right)$ sugiere que para el caso cuando $|\beta_i| \leqslant \delta_i$ se tienen que $\gamma_i = 0$, y para el caso $|\beta_i| > \delta_i$ se tiene $\gamma_i = 1$. Tomando como base lo anterior τ_i y c_i se deben seleccionar de tal modo que si $|\beta_i| \leqslant \delta_i$ sería preferible poner $\beta_i = 0$ y excluir a x_i del modelo. Resumiendo

La selección de c_i en la 4.4 debe ser de tal manera que si $\beta \sim N(0, c_i^2\tau_i^2)$, entonces se puede considerar con seguridad $\beta \neq 0$ por lo cual debe ser incluido en el modelo final.

Uno podría querer seleccionar c_i lo suficientemente grande para que diera soporte a los valores de β_i que son sustancialmente diferentes de 0 pero no tan grandes que den soporte a valores irreales de β_i, se recomienda seleccionar los valores de c_i entre 10 a 100 para lograr una efectiva, pero no extrema, separación entre las dos densidades [8].

También es importante que aportir de esta distribución $N\left(0, c_i^2\tau_i^2\right)$ se obtengan probabilidades razonables para todos los valores de β_i. Para esto, dado los valores de c_i y δ_i, el valor de τ_i se obtiene como $\tau_i = \sqrt{2\log\left(c_i\right)c_i^2/(c_i^2 - 1)\,\delta_i}$ que es una alternativa más sofisticada para la selección de τ_i y c_i que se puede encontrar en el artículo de SSVS [8].

4.5. Elección de υ_γ y λ_γ

Por completitud, a continuación se razonará la selección de los valores de υ_γ y λ_γ en la distribución inversa gama 4.10, siguiendo el razonamiento encontrado

en [8]. Para lo anterior, uno puede suponer que estos valores llevan la información de un experimento previo imaginario donde v_γ es el número de observaciones y $[v_\gamma/(v_\gamma - 2)]\,\lambda_\gamma$ es la estimación *a priori* de σ^2.

Típicamente, estos valores serán constantes ($v_\gamma \equiv v, \lambda_\gamma \equiv \lambda$) o van a depender mayormente de γ sólo a través de $|\gamma|$ que es el número de componentes de γ diferentes de cero.

Por ejemplo, supongamos $[v_\gamma/(v_\gamma - 2)]\,\lambda_\gamma$ sea una función decreciente de $|\gamma|$ cuando se espera que modelos de dimensiones superiores obtengan una σ^2 más pequeña. Como se verá del valor *a posteriori* 4.16 la elección de $v_\gamma \equiv 0$, y cualquier valor para λ, se puede usar para representar la ignorancia o sea la falta de información previa.

4.6. Muestreo de Gibbs

La implementación de SSVS consiste en dos etapas, la primera es para determinar los valores iniciales de c_i,τ_i,v,λ y de $P(\gamma)$ de modo que los valores de γ correspondientes a los modelos prometedores se les asigne una alta probabilidad *a posteriori* bajo $P(\gamma|Y)$.
La siguiente etapa se trata de identificar los valores de γ con alta probabilidad *a posteriori* por medio del muestreo de Gibbs de manera eficiente. Este enfoque evita las dificultades de calcular las 2^p probabilidades *a posteriori* por medio de métodos numéricos. SSVS utiliza el muestreo de Gibbs para generar la secuencia.

$$\gamma^{(1)}, \gamma^{(2)}, ... \tag{4.12}$$

la cual converge a la distribución $\gamma \sim P(\gamma|Y)$. En particular esos valores de γ con alta probabilidad *a posteriori* pueden ser identificados como esos que aparecen con mayor frecuencia en la serie. Se debe notar que en contraste con muchas otras aplicaciones del muestreo de Gibbs la meta aquí no es la evaluación de la distribución $P(\gamma|Y)$. La mayoría de los 2^p valores de γ tendrán probabilidades muy pequeñas que aparecerán muy raramente, y pueden ser ignorados. SSVS utiliza el muestreo de Gibbs para buscar en la cadena generada en lugar de evaluar la cadena generada el valor *a posteriori* $P(\gamma|Y)$. Consecuentemente, la longitud de la secuencia 4.12

puede ser mucho más pequeña que 2^p, y aún servir para identificar los valores con alta probabilidad.

SSVS genera le secuencia 4.12 por medio del muestreo de Gibbs para obtener $P(\beta, \sigma^2, \gamma|Y)$. Esto produce la secuencia completa de los valores de los parámetros

$$\beta^{(0)}, \sigma^{(0)}, \gamma^{(0)}, \beta^{(1)}, \sigma^{(1)}, \gamma^{(1)}, ..., \beta^{(k)}, \sigma^{(k)}, \gamma^{(k)}, ..., \tag{4.13}$$

que es una cadena de Markov que converge a $P(\beta, \sigma^2, \gamma|Y)$ en la que la secuencia 4.12 esta incrustada. El muestreo de Gibbs se realiza como sigue. Primero $\beta^{(0)}$,$\sigma^{(0)}$ y $\gamma^{(0)}$ se inicializan con alguna suposición razonable tal como si se obtuvieran por medio de una regresión gradual y siguendo los razonamientos ya comentados. Los valores siguientes se obtienen por medio de la simulación sucesiva de los condicionales.

$$P\left(\beta|\sigma^2, \gamma, Y\right)$$

$$P\left(\sigma^2|\beta, \gamma, Y\right) = P\left(\sigma^2|\beta, Y\right) \tag{4.14}$$

$$P\left(\gamma_i|\beta, \sigma^2, \gamma_{-i}, Y\right) = P\left(\gamma_i|\beta, \gamma_{-i}\right)$$
$$i = 1, ..., p$$

donde $\gamma_{-i} = (\gamma_1, ..., \gamma_{i-1}, \gamma_{i+1}, ..., \gamma_p)$. Se aprecia que debido a la estructura jerárquica del valor *a priori* la condición para σ^2 depende únicamente de Y y β, y la condición para γ_i depende sólo de β y γ_{-i}. El éxito de SSVS se debe en gran parte al hecho de que la simulación sucesiva de 4.14 que proviene de una distribución estándar,puede ser llevada a cabo de forma rápida y eficientemente por métodos rutinarios (iterativos). Para empezar, el vector de coeficiente β^j se obtiene por el muestreo de

$$P(\beta|, \sigma^2, \gamma, Y) = N_p\left(\left(X^TX + \sigma^2 D_\gamma^{-2}\right)^{-1} X^TY, \sigma^2\left(X^TX + \sigma^2 D_\gamma^{-2}\right)^{-1}\right) \tag{4.15}$$

El paso más costoso es la actualización de $\left(X^TX + \sigma^2 D_\gamma^{-2}\right)$ con base a los nuevos valores de σ^2 y γ. Para obtener $\sigma^{(j)}$ se realiza por medio del muestreo de

$$P\left(\sigma^2|\beta, Y\right) = IG\left(\frac{n+v}{2}, \frac{|Y - X\beta|^2 + v\lambda}{2}\right) \tag{4.16}$$

La distribución inversa gamma actualizada desde 4.10 . Finalmente. El vector γ se obtiene mediante el muestreo de componente por componente de cada γ consecuentemente (y preferiblemente en orden aleatorio) de la distribución de Bernoulli con probabilidad

$$P(\gamma_i = 1|\beta, \gamma_{-i}) = \frac{a}{a+b} \tag{4.17}$$

donde

$$a = P\left(\beta|\gamma_{-i}, \gamma_i = 1\right) P\left(\gamma_{-i}, \gamma_i = 1\right)$$
$$b = P\left(\beta|\gamma_{-i}, \gamma_i = 0\right) P\left(\gamma_{-i}, \gamma_i = 0\right)$$

que, según ssvs, suponiendo una distribución Bernoulli independiente a priori para $P(\gamma)$ de 4.9 y 4.17 se simplifica a

$$P(\gamma_i = 1|\beta_i) = \frac{a}{a+b} \tag{4.18}$$

donde

$$a = P\left(\beta_i|\gamma_i = 1\right) \omega_i$$
$$b = P\left(\beta_i|\gamma_i = 0\right) (1 - \omega_i)$$

La generación de cada nuevo valor de γ calculado un paso a la vez, donde a cada paso el valor de algún γ_i se determina al azar de acuerdo con la distribución condicional de Bernoulli 4.17. Debido a que el cambio de γ_i corresponde esencialmente a la decisión de añadir o remover x_i de la regresión, la generación de la secuencia 4.12 es equivalente a realizar un procedimiento paso a paso dirigido estocásticamente.

4.7. Algoritmo SSVS

A continuación de describe el algoritmo de SSVS

Paso 1: Para inicializar $\beta^{(0)}, (\sigma^2)^{(0)}$, y $\gamma^{(0)}$ $\beta^{(0)}$ y $(\sigma^2)^{(0)}$. Se inicializan por medio una regresión de mínimos cuadrados., $\gamma^{(0)}$ se inicializa con todos sus elementos con valor uno, $\gamma^{(0)} = (1, ..., 1)^p$

Paso 2: Generar

$$\beta^{(j+1)} \sim N_p\left(\left(X^TX + \left(\sigma^2\right)^{(j)} D^{-2}_{\gamma^{(j)}}\right)^{-1} X^TY, \left(\sigma^2\right)^{(j)}\left(X^TX + \left(\sigma^2\right)^{(j)} D^{-2}_{\gamma^{(j)}}\right)^{-1}\right)$$

Paso 3: Generar

$$\left(\sigma^{-2}\right)^{j+1} \sim Ga\left(\frac{n + v_{\gamma^{(j)}}}{2}, \frac{\left|Y - X\beta^{(j)}\right|^2 + v_{\gamma^{(j)}}\lambda_{\gamma^{(j)}}}{2}\right)$$

Paso 4: Generar $\gamma_i^{(j+1)} \sim f\left(\gamma_i|Y, \beta^{(j)}, (\sigma^2)^{(j)}, \gamma_i\right)$

$$\gamma_i^{(j+1)} = f\left(\gamma_i|\beta^{(j)}, \left(\sigma^2\right)^{(j)}, \gamma_i\right) \tag{4.19}$$

la distribución 4.19 no depende de Y. Esto permite simplificar los cálculos y permite una rápida convergencia de la secuencia 4.12. Cada variable de la ecuación 4.19 se obtiene usando la distribución Bernoulli.

suponiendo una distribución Bernoulli independiente a priori para $P(\gamma)$ de 4.9 y 4.17 se simplifica a

$$P(\gamma_i = 1|\beta_i) = \frac{a}{a+b}$$

Donde

$$a = P(\beta_i|\gamma_i = 1)\,\omega_i$$

$$b = P(\beta_i|\gamma_i = 0)\,(1 - \omega_i)$$

Paso 5: Regresar al paso 2

CAPÍTULO 5

Software

En este capítulo se describirán las partes principales que componen el software de minado de datos, así como su implementación, las herramientas que se utilizaron para su programación y los entornos de desarrollo.

El software se divide esencialmente en tres partes que son:

- La aplicación web.
- Los algoritmos de minado de datos.
- Los procesos de cada algoritmo.

5.1. Aplicación Web

La aplicación web fue desarrollada en Visual Studio 2010, su función principal es la de proveer una interfaz web para el usuario. La interfaz despliega una serie de opciones mediante las cuales se pueden seleccionar los datos que se quieren manipular, el algoritmo que se quiere aplicar a los datos, el tipo de ejecución del algoritmo y una visualización básica de los resultados.

La aplicación web está dividida en 3 páginas "Sign In" que es el inicio de la aplicación por medio de la cual un usuario registrado pueda acceder a las funciones del software, se debe dar la dirección IP del servidor local que es la dirección del SMBDR, que en este caso es SQL Server; el puerto, el nombre de usuario y la contraseña.

Figura 5.1: Ventana de Sign In.

En la pestaña “New Process” de la aplicación es donde el usuario seleccionará los datos, el algoritmo y el modo de ejecución. La página consta de diversos campos a seleccionar, en un principio se debe proporcionar un “Cliente ID” con el que el software identificará a todos los procesos de un determinado usuario. Los siguientes pasos son la selección de una base de datos, la selección de la tabla que contiene los datos, la selección de las columnas que se requieran, la elección de un algoritmo de minado de datos; en esta fase se pueden habilitar otros campos dependiendo de si el algoritmo seleccionado requiere parámetros o especificaciones adicionales. Por último se selecciona el botón “Suggestion” con el cual se determina el costo en tiempo de ejecutar el algoritmo seleccionado en modo local o en la nube, con lo cual el usuario podrá decidir el modo de ejecución con base a la sugerencia del software o elegir otro método. Y por último presionará el botón “Run algorithm” con lo que el software iniciará la ejecución del algoritmo. La página “Results” básicamente se encarga de ver todos los procesos, las veces que un algoritmo ha sido ejecutado, así como su porcentaje de avance. Se pueden visualizar todos los procesos que un usuario ha ejecutado y puede ver de manera básica los resultados obtenidos. La pro-

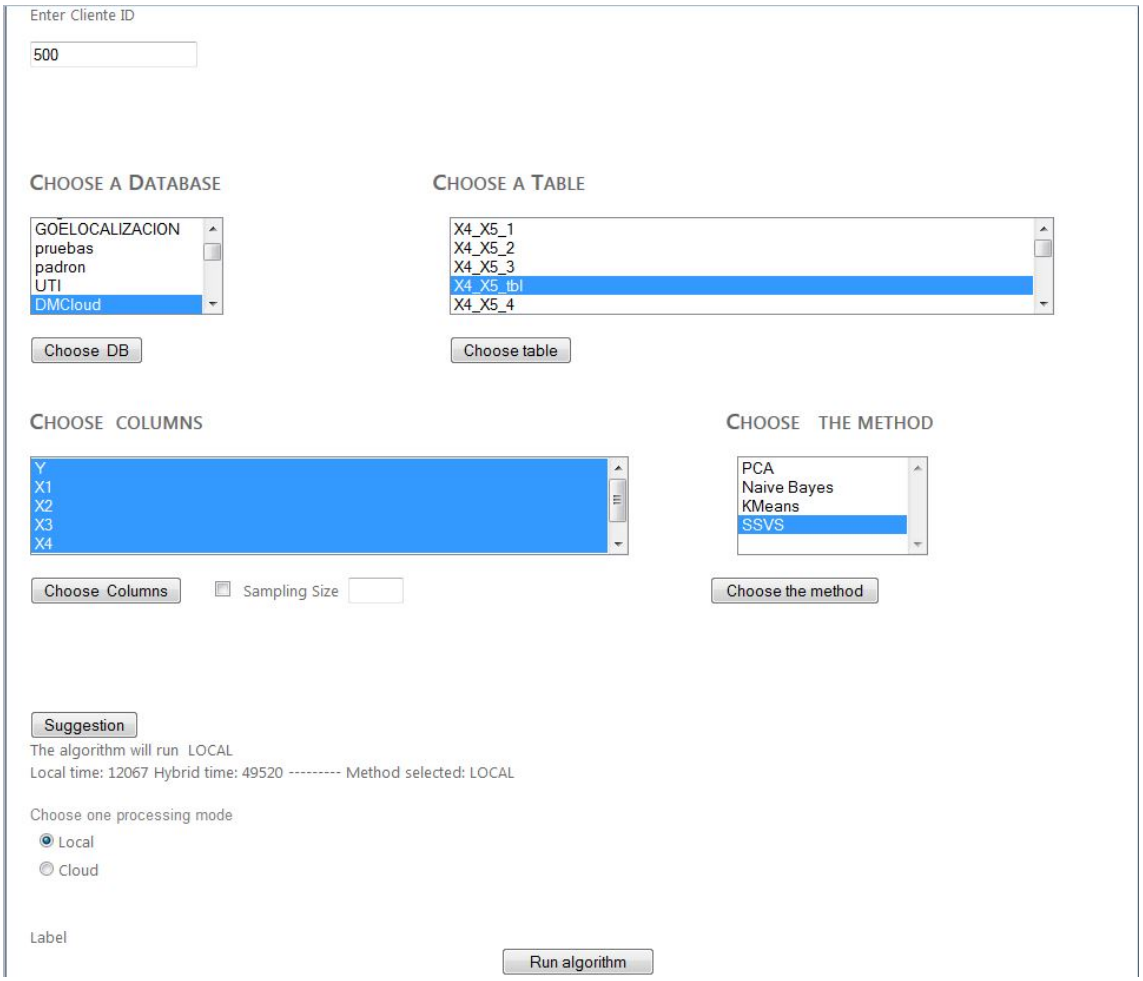


Figura 5.2: Ventana de New Process.

gramación de la aplicación web se realizó en C# y con ASP.NET. La programación se basó principalmente en el funcionamiento de los controles para el usuario y la llamada a los algoritmos de minado de datos y el cálculo del costo de ejecución de cada algoritmo.

5.2. Programación de los algoritmos de minado de datos

Cada uno de los algoritmos de datos se programó con C# en Visual Studio 2010. En la opción de proyecto para Base de Datos que da el entorno de programación

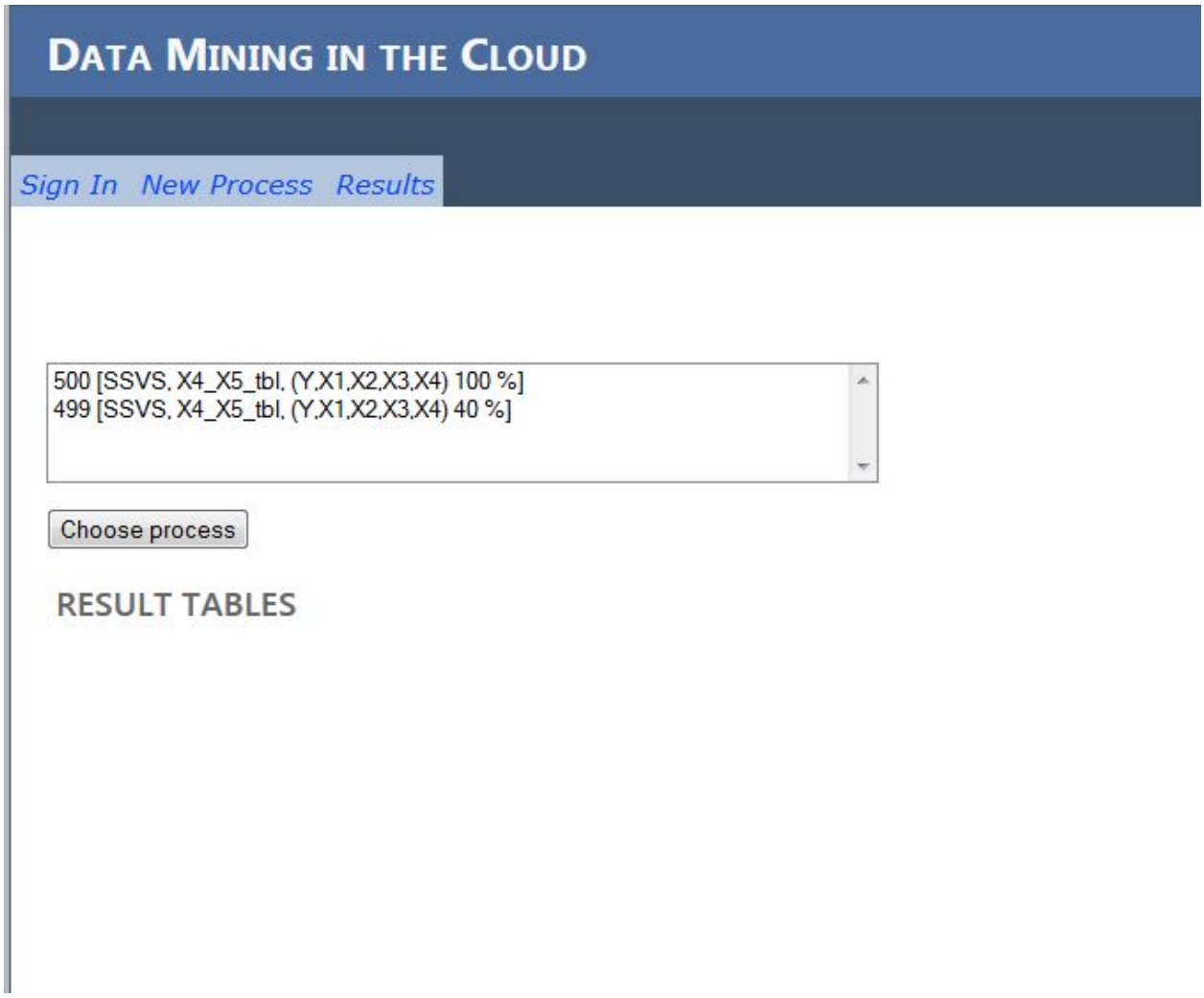
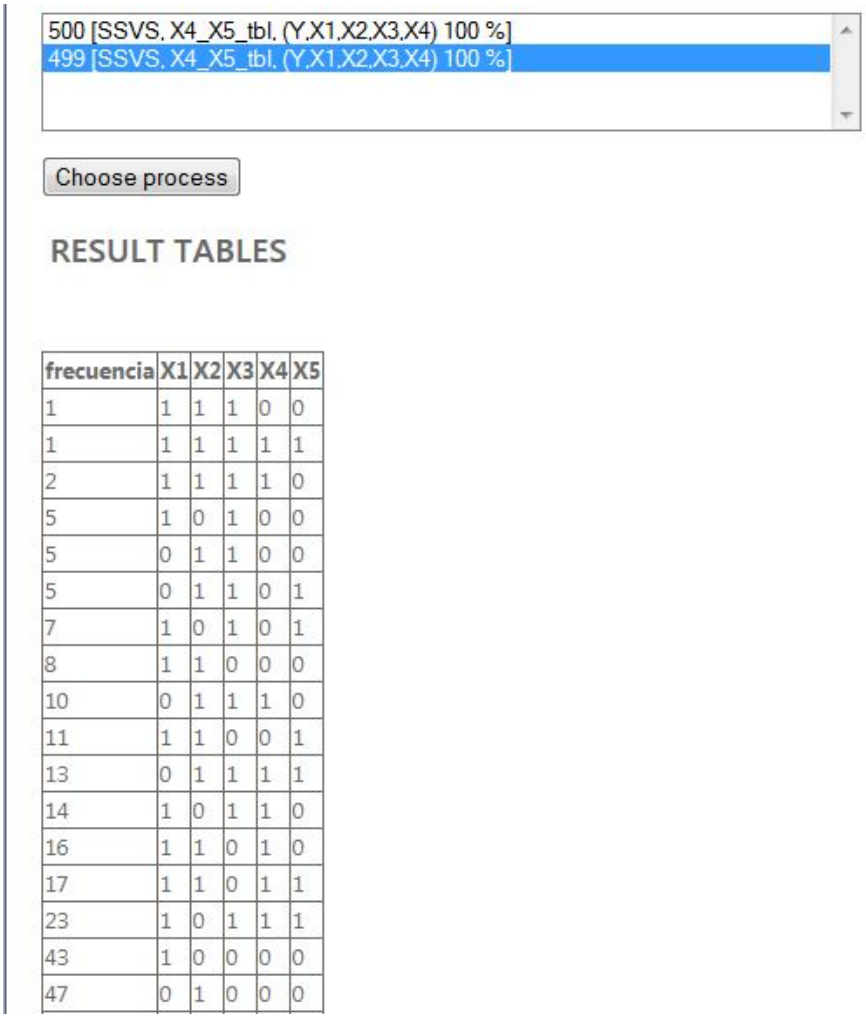


Figura 5.3: Ventana de Resultados.



frecuencia	X1	X2	X3	X4	X5
1	1	1	1	0	0
1	1	1	1	1	1
2	1	1	1	1	0
5	1	0	1	0	0
5	0	1	1	0	0
5	0	1	1	0	1
7	1	0	1	0	1
8	1	1	0	0	0
10	0	1	1	1	0
11	1	1	0	0	1
13	0	1	1	1	1
14	1	0	1	1	0
16	1	1	0	1	0
17	1	1	0	1	1
23	1	0	1	1	1
43	1	0	0	0	0
47	0	1	0	0	0

Figura 5.4: Ventana de Resultados SSVS.

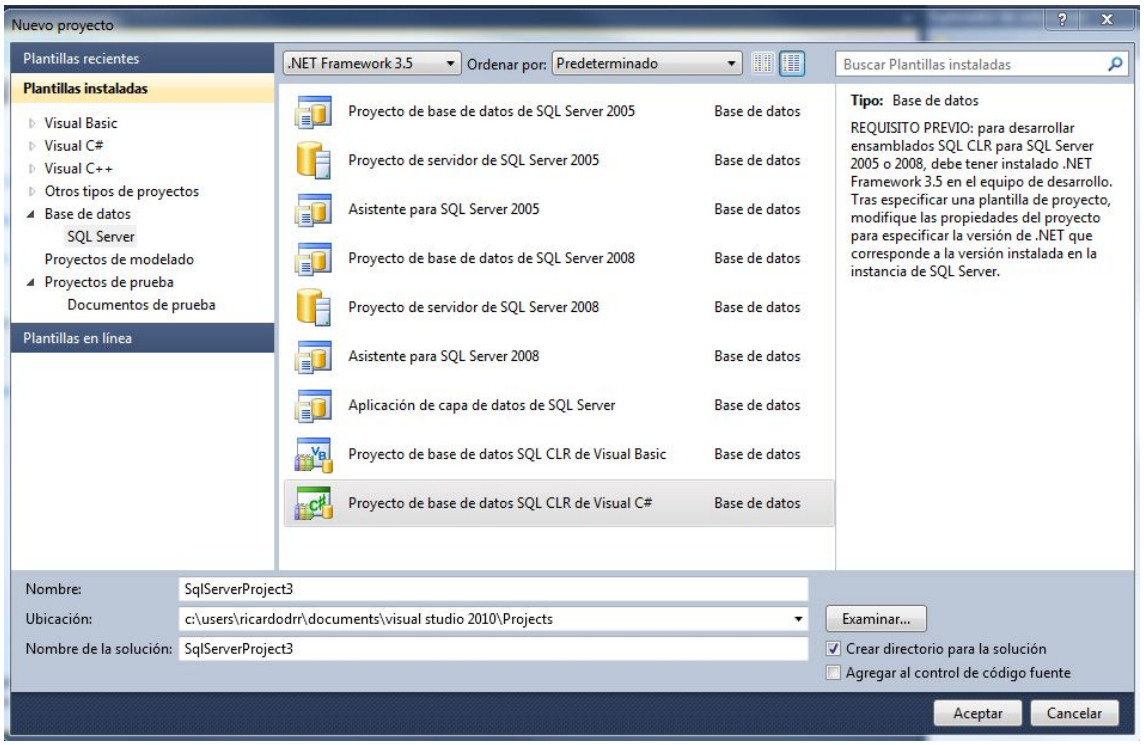


Figura 5.5: Opción para proyectos sobre SQL Server.

"Proyecto de base de datos SQL CLR de Visual C# " Se permite programar con C# diversas funciones para una base de datos como procedimientos almacenados, funciones de agregación, funciones definidas por el usuario, tipos de datos, y desencadenadores o disparadores "trigger". Lo anterior es posible gracias al (CLR "Common Language Runtime" CLR por sus siglas en inglés), que se podría traducir al español como: entorno en tiempo de ejecución de lenguaje común, que es parte fundamental en el entorno de desarrollo .NET.

Se programó un procedimiento almacenado por cada algoritmo. Para optimizar algunas de las operaciones que requieren cada uno de los algoritmos se utilizó la biblioteca de LAPACK que ofrece un sin fin de funciones de álgebra lineal, más en específico, se utilizó una variante llamada MKL "Math Kernel Library" que contiene la biblioteca de LAPACK con la ventaja que se encuentra optimizada para los procesadores de Intel.

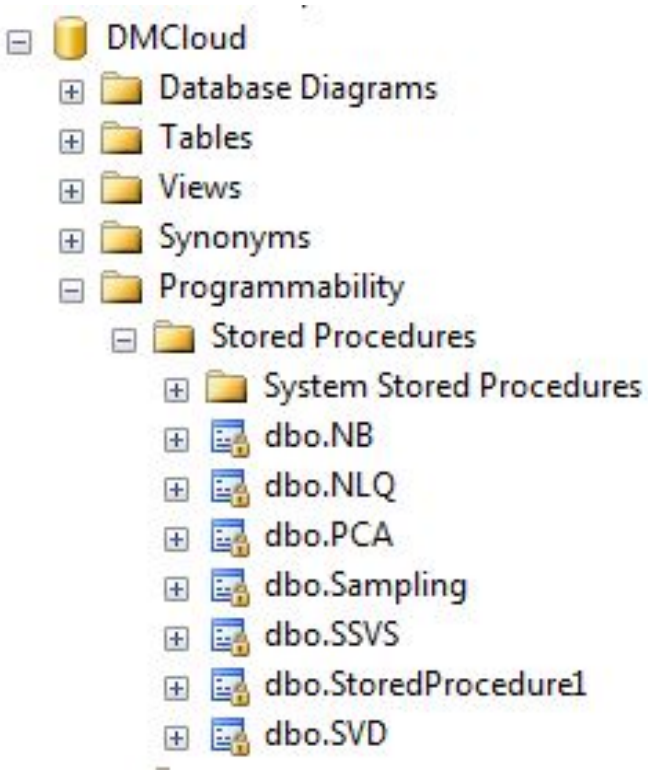


Figura 5.6: No se puede modificar los SP desde SQL Server.

Una vez programado cada algoritmo se procedió a incorporarlo a SQL Server como un procedimiento almacenado que puede ser llamado como cualquier otro procedimiento. Visual Studio se encarga de incorporar el procedimiento en la base de datos seleccionada, sólo se debe alterar la base de datos destino con "ALTER DATABASE Mi_Base_de_Datos SET TRUSTWORTHY ON" que permite la ejecución de los procedimientos almacenados que se compilan con CLR, estos procedimientos almacenados no podrán ser modificados desde SQL Server porque para su modificación se necesitará hacerlo desde el proyecto correspondiente en Visual Studio.

La principal razón de programar los algoritmos en C# y después incorporarlos, se debe a las ventajas de un lenguaje como C# que permite realizar cálculos con los datos de una manera más eficiente y controlada de lo que se podría realizar con Transact-SQL. Al ser SQL un lenguaje concebido para realizar consultas e incluso tener la opción por medio de apuntadores para realizar operaciones iterativas, es bastante ineficiente para las iteraciones, lo que complica la manipulación y operación

de datos cuando se requiere realizar algún tipo de cálculo que involucre una iteración.

5.3. Ejecución de los algoritmos de minado de datos

Al tener los algoritmos como procedimientos almacenados en la base de datos, estos serán llamados por medio de la aplicación web, cuando se ejecute un algoritmo por medio de la aplicación web se creará un hilo o "thread" con lo que se controlará y se podrá saber el estado de cada algoritmo por medio del hilo creado.

Lo anterior nos permitirá crear diversos hilos de un mismo algoritmo, incluso crear diversos hilos para diversos usuarios, y debido a que diversos hilos pueden ser creados mientras se ejecuta un algoritmo se debe tener un sistema de control; en otras palabras se necesita de un planificador de proceso. El tema sobre un planificador de procesos es muy extenso y corresponde a otro tema de estudio. Por lo cual en este trabajo nos limitaremos a mencionar el tipo de planificador elegido. El planificador seleccionado es el clásico FIFO "first input, first output" primero en entrar primero en salir, debido a que el software desarrollado en este trabajo tiene como uno de sus objetivos proveer algoritmos de minado y que se puedan ejecutar en modo local o en la nube la implementación de un planificador con diversos algoritmos para el manejo de los procesos, no es una prioridad por la cual se eligió el algoritmo de planificación anterior que es uno de los más sencillos de implantar.

Otro aspecto importante es determinar un tiempo estimado para la ejecución de cada algoritmo, tanto para su ejecución de modo local y de modo en la nube. Con esta información la aplicación dará una sugerencia al usuario de qué modo es más rápido ejecutar el algoritmo.

Para la estimación del tiempo de ejecución se toman en cuenta los siguientes aspectos; para el modo local, se necesita determinar el tamaño del conjunto de los datos lo cual se realiza con la instrucción "Table Scan" que realiza un barrido secuencial de una determinada tabla, se necesita determinar las veces que los datos serán leídos de la base de datos. Esta etapa dependerá mucho del tipo de algorit-

mo que sea seleccionado, para el caso particular del algoritmo de SSVS se debe hacer una lectura de todos los datos para calcular los mínimos cuadrados para inicializar los parámetros $\beta^{(0)}$ y $(\sigma^2)^{(0)}$, una operación recurrente en este algoritmo es la operación X^tX que puede ser sustituida por la estadística suficiente Q, con lo que el paso más costoso de la ecuación 4.16 es $\left(X^TX + \sigma^2 D_\gamma^{-2}\right)$, se puede reducir, pero aún así se debe leer la matriz Q tantas veces como muestras se quieran generar.

El tiempo estimado para SSVS de modo local se calcula

$$t_{SSVS} = (t_{tableScan})\,(n)\,(p) \tag{5.1}$$

Donde

$t_{tableScan}$ = es el tiempo que se tarda en hacer una recorrido secuencial de la tabla.

n = es el número de muestras de la secuencia del muestreo de Gibbs que se quiere generar.

p= es el número de variables predictoras.

Para el modo en la nube se toma en cuenta el tiempo local pero calculado con las características del servidor que se supone en un principio con un mejor hardware que el cliente, más una serie de parámetros para calcular el tiempo de transmisión de los datos al servidor. Se utiliza el procedimiento almacenado "$p_spaceuse$" que provee SQL Server con el cual se obtiene el espacio de disco reservado y el espacio de disco utilizado por una tabla, únicamente se utiliza el espacio que ocupa la tabla. Se utiliza el método "Send" de la clase "Ping" que provee el Framework .NET 4.5 de Visual Studio que intenta enviar un mensaje de eco ICMP (Protocolo de mensajes de control de Internet) a un equipo remoto y recibe un mensaje de respuesta de eco ICMP correspondiente del equipo remoto, después se utiliza la propiedad "RoundtripTime" de la misma clase con lo que se obtiene el número de milisegundos empleados en enviar un mensaje de solicitud de eco ICMP y recibir el mensaje de respuesta de

eco ICMP correspondiente. Con lo anterior se puede determinar el tiempo promedio que tardarían los datos al ser enviados al servidor en la nube. El tiempo estimado para le ejecución en la nube es;

$$t_{nube} = t_{local} + t_{transmision} \tag{5.2}$$

El tiempo de transmisión se calcula de la misma manera para todos los algoritmos, lo que llega a variar es el cálculo del tiempo de ejecución para cada algoritmo el cual depende de las veces que se tengan que leer los datos de la base de datos para realizar los cálculos.

CAPÍTULO 6

Experimentos

En este capítulo se experimentó con la herramienta descrita en capítulos anteriores y en particular con el algoritmo SSVS con diferentes conjuntos de datos y bajo diferentes condiciones de hardware. En la primera sección de este capítulo se experimentó con la precisión de la herramienta utilizando el algoritmo SSVS. En la siguiente sección se presentan los resultados de la experimentación del desempeño de la herramienta respecto al tiempo.

Condiciones iniciales.

Tabla 6.1: Características del Servidor y el Cliente

	Cliente	**Servidor**
Procesador	Intel Core 2 Duo 2.4 Ghz	Intel i7 2.5Ghz
Memoria RAM	8Gb	16Gb
Manejador de Base de Datos	SQL Server 2008	SQL Server 2008

6.1. Experimentación con la precisión del modelo

6.1.1. Experimento 1

Sea x_1, x_2, x_3, x_4, x_5 las variables predictoras o variables independientes y

$$Y = x_4 + 1{,}2(x_5) + e \tag{6.1}$$

la ecuación lineal que describe la relación entre las variables independientes y la variable dependiente Y, e es un error aleatorio. $n = 60$ que es el número de muestras

de Y, el número de muestras de Gibbs es 5000 ,$\tau = 0.44$ en promedio para las cinco variables, $\upsilon = 0$ y $\lambda = 0$ que como se menciona en el capítulo 4 apartado 5 es para representar la ignorancia, es decir que no se sabe nada del modelo de antemano, $c = 10$.

Las variables x_1, x_2, x_3, x_4, x_5 se generaron por medio de una distribución normal de media cero y varianza uno, el tamaño de la tabla es de 80Kb, la velocidad de transmision de los datos entre el cliente y el servidor es de 512 Kbit/s.

En estas condiciones el tiempo de transmisión fue menor a un segundo, el tiempo empleado por el cliente en procesar los datos fue de 18.7 segundos y el tiempo empleado por el servidor es de 12.3 ss., cabe hacer notar que incluso sumando el segundo del tiempo de transmisión es más rápido el servidor.

De las 5000 muestras generadas que se obtuvieron para este experimento, 29 modelos se repiten al menos una vez, los tres modelos más repetidos se muestran en la tabla 6.2

Tabla 6.2: Resultados experimento uno

repeticiones	**x1**	**x2**	**x3**	**x4**	**x5**
786	0	0	0	1	0
1002	0	0	0	0	1
1466	0	0	0	1	1

Como se puede observar en la Tabla 6.2 las veces que el modelo donde se incluyen las variables x_4 y x_5 es el que más se repite con 1466 veces, seguido por el modelo donde sólo se incluye la variable x_5 con 1002 repeticiones y por último con 786 el modelo donde precisamente se incluye la variable x_4. Por lo cual las variables que se seleccinarían serian x_4 y x_5

6.1.2. Experimento 2

Sean x_1, x_2, x_3, x_4, x_5 las predictoras o variables independientes y

$$Y = x_1 + 2x_4 + 1{,}2x_5 + e \tag{6.2}$$

Los parámetros para este experimento tienen los mismos valores que los del experimento 1, lo que se modifica es la función lineal, se agrega una variable x_1. El tiempo en el servidor fue de 12.7 ss. y en el cliente fue de 18.7 ss., de las 5000 muestras ahora se tienen 19 modelos que se repiten al menos una vez, de nueva cuenta se tomaron los 3 modelos que más se repiten. Ver tabla 6.3

Tabla 6.3: Resultados experimento 2

repeticiones	**x1**	**x2**	**x3**	**x4**	**x5**
547	0	0	0	1	0
1401	1	0	0	1	0
1483	1	0	0	1	1

El modelo que más se repite con 1483 es el que contiene las variables x_1, x_4 y x_5 que son justamente las variables que describen a la ecuación 6.2. Como se puede constatar en los dos experimentos anteriores el algoritmo SSVS funciona bastante bien. En las siguientes experimentos se realizarán algunas variaciones como por ejemplo aumentar el número de muestras de Y, disminuir la cantidad de muestras de Gibbs e incrementar el número de variables predictoras.

6.1.3. Experimento 3

Experimento 3 a

Se mantienen los mismos parámetros que en la experimento 2, se repite la misma ecuación 6.2, sólo el número de muestras de Gibbs que se quiere obtener ahora serán de 2500. El tiempo que se tardó el algoritmo en ejecutarse en el cliente fue de 10.46 ss. y en el servidor fue de 6.31 ss., esta mejora se debió a que se generó un número menor de muestras de Gibbs.

La mejora en tiempo es a costa de la precisión para identificar correctamente las variables, como se ve en la Tabla 6.4 el modelo con más repeticiones, con 1178, incorpora 2 variables, x_1 y x_4, deja fuera a la x_5. El modelo que incorpora las tres variables de la Ecuación 6.2 es el segundo con mayor repeticiones, con 800. El tercer modelo se repite 143 veces, incluso incorpora la variable x_2 que no está presente en la ecuación 6.2.

Tabla 6.4: Resultados experimento 3 a, con 2500 muestras de Gibbs

repeticiones	x1	x2	x3	x4	x5
143	1	1	0	1	0
800	1	0	0	1	1
1178	1	0	0	1	0

Experimento 3 b

Este experimento es idéntica a la anterior pero ahora el número de muestras de Y se duplicará a 120

Tabla 6.5: Resultados experimento 3 b, con 120 muestra de Y

repeticiones	x1	x2	x3	x4	x5
153	1	0	1	1	1
606	0	0	0	1	1
1378	1	0	0	1	1

Como se puede ver en la Tabla 6.5 al incrementar el número de muestras de Y la precisión para identificar correctamente las variables de la Ecuación 6.2 se incrementa. Ahora el modelo con 1378 repeticiones y que es el que más se repite y contiene las variables que describen la ecuación 6.2, el tiempo en el cliente es de 10.3 ss., en el servidor el tiempo fue de 6.3 ss. Como se puede observar el tiempo, tanto en el cliente y el servidor casi son el mismo que en el experimento tres, con la gran diferencia de que en este experimento el modelo con más repeticiones contiene a las variables predictoras x_1, x_4 y x_5 que son las que describen la ecuación 6.2.

6.1.4. Experimento 4

Sea la ecuación 6.1 pero con la pequeña variante de que introduciremos una variable correlacionada.

$$x_2 = x_4 + ,2z \tag{6.3}$$

Donde Z es un número aleatorio seleccionado de una distribución normal con media cero y varianza uno.

Tabla 6.6: Resultados experimento cuatro, variable correlacionada.

repeticiones	**x1**	**x2**	**x3**	**x4**	**x5**
295	0	1	0	0	1
1003	0	0	0	1	1
1207	0	0	0	0	1

Como se ve en la Tabla 6.6 el modelo con 1207 repeticiones contiene a x_5, el segundo modelo con 1003 contiene la x_4, x_5 que como sabemos es el modelo correcto, el tercer modelo se repite 295 y contiene la variable correlacionada x_2y x_5.

Ejecutando nuevamente el algoritmo y generando la variable correlacionada se tiene la Tabla 6.7

Tabla 6.7: Resultados experimento cuatro , variable correlacionada.

repeticiones	**x1**	**x2**	**x3**	**x4**	**x5**
384	0	0	0	1	1
514	1	0	0	0	1
661	0	1	0	0	1
1829	0	0	0	0	1

El modelo con 1829 es el que contiene la variable x_5, el segundo modelo con 661 repeticiones es el que contiene x_5 y la variable correlacionada x_2. El modelo buscado es el cuarto modelo con 384 repeticiones, como se puede ver en los cuatro modelos con más repeticiones en todos aparece la variable x_5 que es una de las dos variables que describen a Y. El modelo con 661 repeticiones y el modelo con 384 repeticiones se observa que se intercambian la variables x_2y x_4 las cuales son las variables involucradas en la correlación.

La existencia de variables altamente correlacionadas puede diluir un poco la capacidad del algoritmo SSVS para determinar qué variables son las más relevantes.

6.1.5. Experimento 5

Ahora se incrementarán las variables predictoras a $p = 30$, las muestras $n = 200$, el tamaño de la tabla es de 110Kb los demás parámetros se conservan como en el

experimento 1. Sea

$$Y = x_1 + 2\,(x_3) + x_6 + 1{,}3\,(x_9) + x_{11} + {,}5\,(x_{15}) + {,}9\,(x_{19}) + x_{22} + x_{26} + x_{30} + e \quad (6.4)$$

El tiempo en el servidor fue de 1.19 minutos y en el cliente fue de 16.46 minutos.

Tabla 6.8: Resultados experimento 5.

Repeticiones	**Variables**
49	x3,x6,x9,x11,x19,x22,x26,x30
73	x1,x3,x6,x9,x11,x22,x26,x30
127	x1,x3,x6,x9,x11,x19,x22,x30

En la Tabla 6.8 se pueden ver los tres modelos con mayor número de repeticiones de un total de 3230 modelos los cuales se repiten al menos una vez. Ninguno de los tres modelos contienen las diez variables que corresponden a la Ecuación 6.4 al modelo con 127 repeticiones le falta x_{15} y x_{26}, en ninguno de los tres modelos aparece la variable x_{15}.

6.1.6. Experimento 6

Ahora se realizará lo mismo que en el experimento 6 pero se aumentará el número $n = 500$, la tabla ahora pesa 400kb. El tiempo en el servidor fue de 1.4 minutos y en el cliente de 17.86 minutos.

Tabla 6.9: Resultados experimento 6.

Repeticiones	**Variables**
55	x1,x3,x6,x9,x11,x19,x22,x26,x28,x30
202	x1,x3,x6,x9,x11,x15,x19,x22,x26,x30
349	x1,x3,x6,x9,x11,x19,x22,x26,x30

Como se puede ver en la Tabla 6.9 el modelo con 349 repeticiones contiene nueve de las diez variables que debería, se agregó x_{26} la cual no aparece en la otra Tabla 6.8. Es de esperarse que al aumentar los datos que describen a Y se mejoran los resultados obtenidos.

6.1.7. Experimento 7

Ahora se incrementarán las muestras de Gibbs a generar a 10000 y con $n = 200$. El tiempo en el servidor fue de 2.7 minutos.

Tabla 6.10: Resultados experimemto 7.

Repeticiones	**Variables**
88	x1,x3,x6,x9,x11,x19,x22,x28,x30
115	x1,x3,x6,x9,x11,x15,x19,x22,x26,x30
359	x1,x3,x6,x9,x11,x19,x22,x26,x30

Como se puede ver en la tabla 6.10 los resultados son muy parecidos con las tablas 6.8 y 6.9 de los experimentos cinco y seis , al tener $n = 200$ que es relativamente pequeño la calidad de los modelos resultantes no se incrementa.

6.2. Experimentación de tiempo de ejecución

Esta sección es un conjunto de experimentos realizados con el fin de ver la variación del tiempo de procesamiento de SSVS en el servidor y en el cliente. Las condiciones para la ejecución de SSVS son las mismas que el experimento 1 de este capítulo, a excepción del número de variables las cuales se irán incrementando, los experimentos se realizaron variando la cantidad de las posibles variables predictoras. Se empieza con un modelo que tiene cinco variables predictoras hasta 40 variables.

Tabla 6.11: Tiempo de ejecución de SSVS.

Variables	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
Cliente	0.3	0.5	0.73	1	6	14.93	18.25	20.67
Servidor	0.2	0.36	0.58	0.7	1	1.28	2.11	2.73

En la Tabla 6.11 se muestra el tiempo en minutos que toma la ejecución del algoritmo SSVS en el cliente y en el servidor para un mismo número de variables predictoras. En un inicio la diferencia en el tiempo de ejecución es mínima de apenas segundos, pero conforme se incrementa el número de variables la diferencia entre el cliente y el servidor se incrementa llegando a minutos. La diferencia entre los

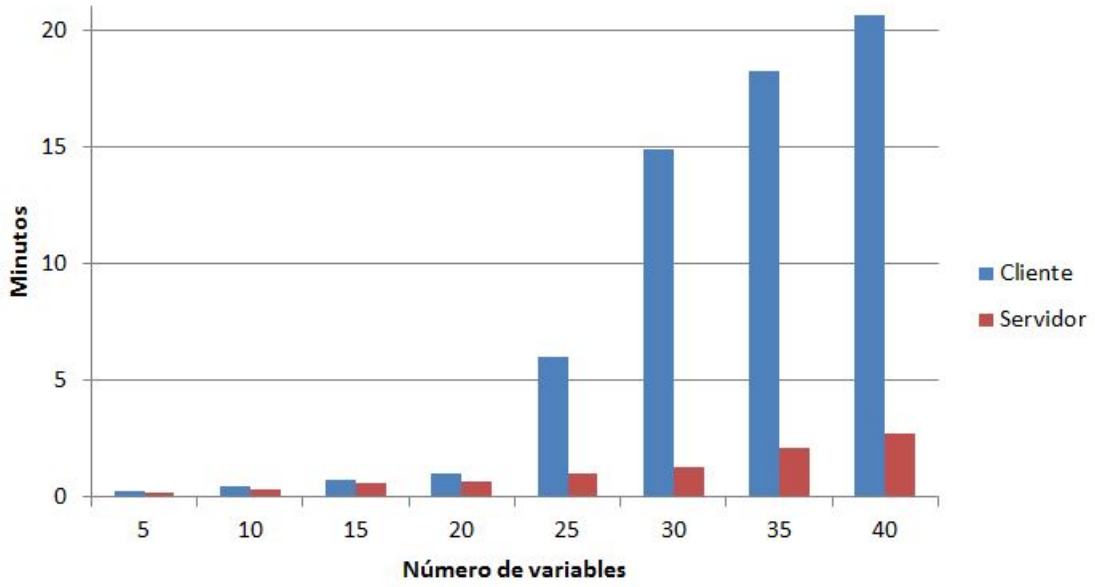


Figura 6.1: Gráfica de tiempos de ejecución de SSVS.

dos modos de ejecución se mantiene relativamente pequeña hasta las 20 variables predictoras, donde la diferencia es de segundos.

6.3. Costos

En este capítulo se buscará dar una idea de los costos de acceder a un servidor en la nube, suponiendo que el usuario cuenta ya con una máquina propia de similares condiciones a la utilizada como cliente en los experimentos, por lo cual supondremos el costo de ejecución en el cliente como cero. Existen diferentes modalidades, costo por demanda, costo por tiempo de uso, renta mensual. Los precios varían dependiendo de la opción de configuración de hardware así como si el servidor es virtual o físico.

Tal vez la opción más conocida de servicio de cómputo en la nube sea el servicio de Amazon Web Services (AWS), la cual prove una instancia llamadas Amazon Elastic Compute Cloud (Amazon EC2) que es un servicio basado en Web que permite a las empresas ejecutar programas de aplicación en el entorno informático de AWS. EC2 puede actuar como un conjunto ilimitado de máquinas virtuales. Existen una

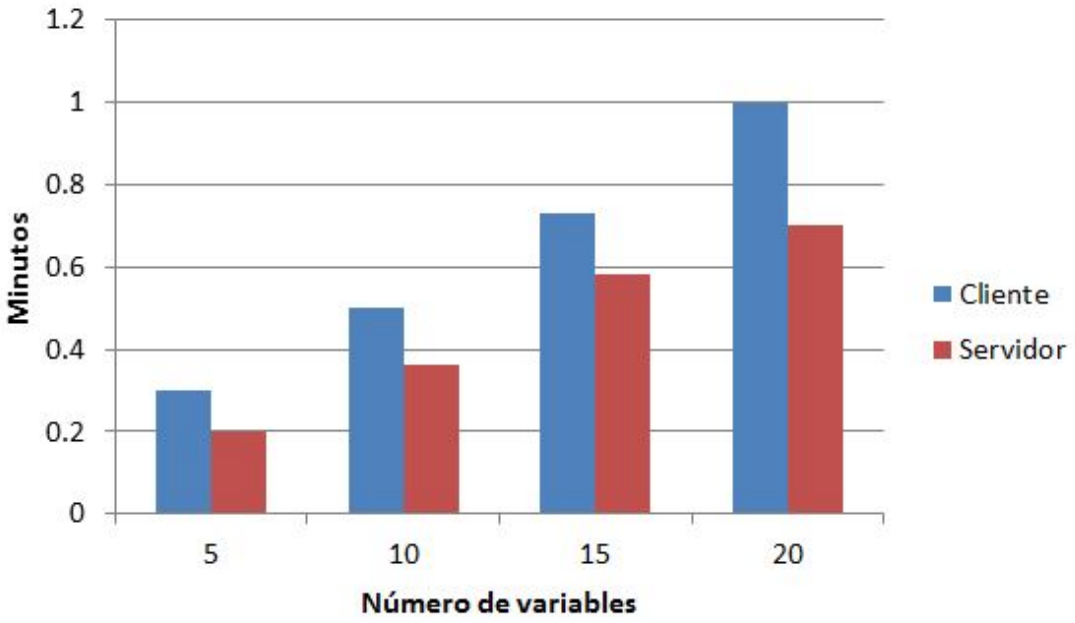


Figura 6.2: Gráfica de tiempos de ejecución de SSVS hasta 20 variables.

gran variedad de instancias de EC2, que varían según el tipo de almacenamiento, al capacidad de procesamiento, y la cantidad de memoria RAM.

Las instancias T2 son de uso general, proporcionan un rendimiento básico de CPU, con la opción de que proporciona ráfagas de rendimiento de CPU por encima del nivel básico. Se recomienda estas instancias para aplicaciones que no usan el CPU por completo de manera contante, pero que en algunos momentos requieren utilizar el CPU por encima de su desempeño básico. Las instancias M3 ofrecen un nivel de rendimiento constante, estas instancias proporcionan un procesamiento de CPU de alto nivel con un coste bajo, además de que representan una buena relación CPU y memoria. Las instancias C4 utilizan procesadores Intel Xeon, y se recomiendan para aplicaciones con un nivel de rendimiento mayor a lo requerido en las instancias M3. Las instancias R3 se encuentran optimizadas para el uso de la memoria principal, al ofrecer grandes cantidades de memoria estas instancias son excelentes para bases de datos relacionales y otras aplicaciones que requieran un uso intenso de la memoria. Los precios de la instancia bajo demanda con la opción de Windows con SQL Standard son:

Tabla 6.12: Precios instancia m3

Tipo	**vCPU**	**Memoria**	**Costo por hora(dolares)**
m3.medium	1	3.75	$ 0.374
m3.large	2	7.5	$ 0.747
m3.xlarge	4	15	$ 1.345
m3.2xlarge	8	30	$ 2.689

Las instancias del tipo M3 utilizan procesadores Xeon E5-2670 a 2.50 GHz.

Otra opción es Telmex la cual ofrece solo planes mensuales, siendo la más barata la opción de $1729. Pero no incluye ninguna licencia de SQL server, la licencia de SQL Standard User 2012 tiene un costo mensual de $ 337 y el costo extra por cada GB en RAM es de $500 al mes, a esta opción se le agregara 15 GB en RAM para que tenga las mismas condiciones que el servidor que se utilizó en el capítulo de experimentos lo que da un presupuesto mensual de:

Tabla 6.13: Precios Telmex

Renta Mensual	**$9566 (pesos)**
vCPU	2
RAM	16

Tomando como base los experiementos realizados al servidor y el cliente se puede estimar una ecuación polinómica de grado dos que mejor se ajuste a los datos obtenidos siendo la ecuación

$$ys = 0{,}0014x^2 + {,}0078x \tag{6.5}$$

la que describe el tiempo en el servidor.

Utilizando la ecuación 6.5 se tiene la tabla 6.14 con un estimado del tiempo para calcular hasta un modelo con mil variables y el costo utilizando una instancia de AWS m3.xlarge.

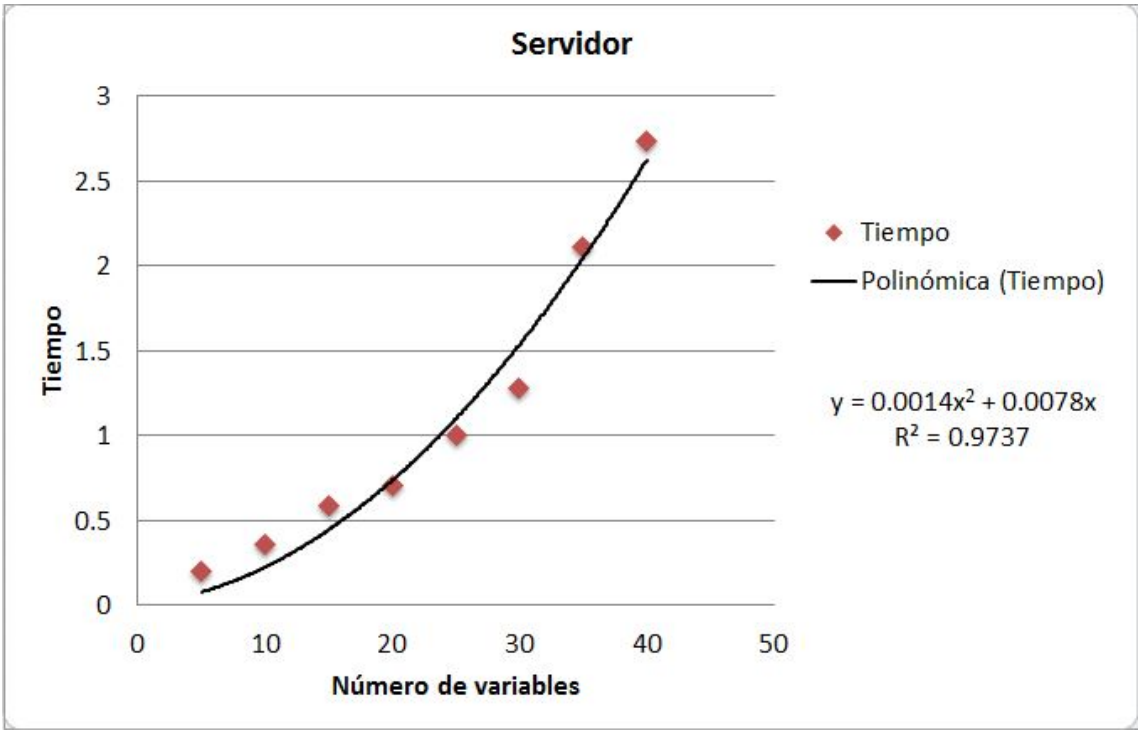


Figura 6.3: Ecuación polinómica para el estimado del tiempo.

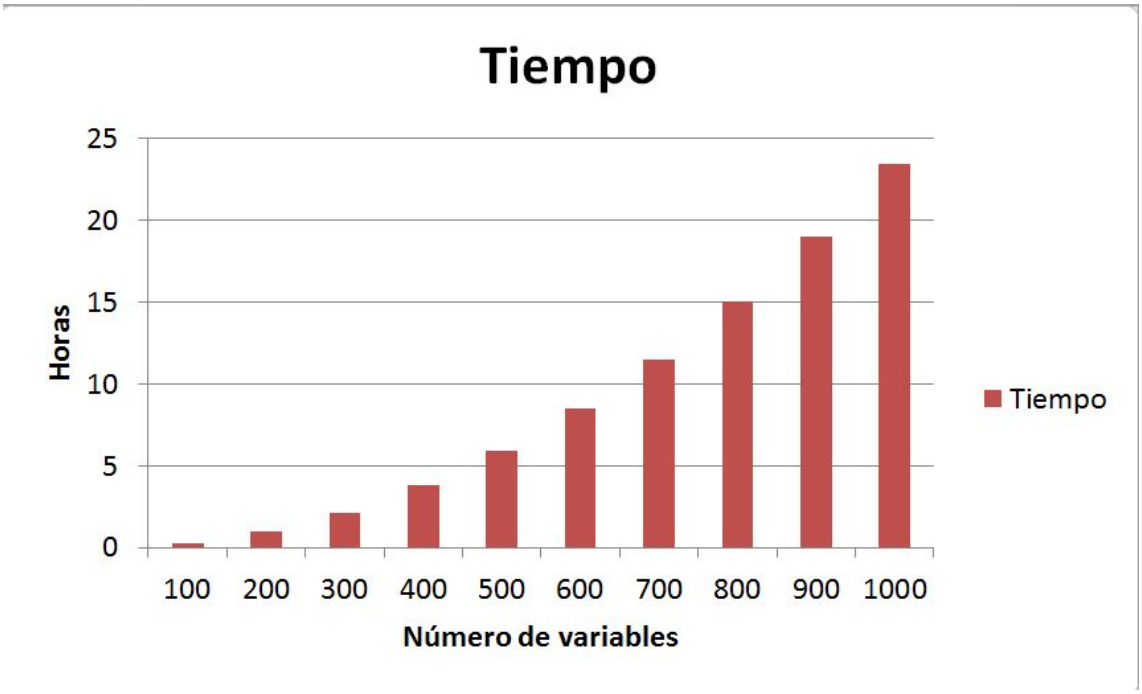


Figura 6.4: Tiempo estimado.

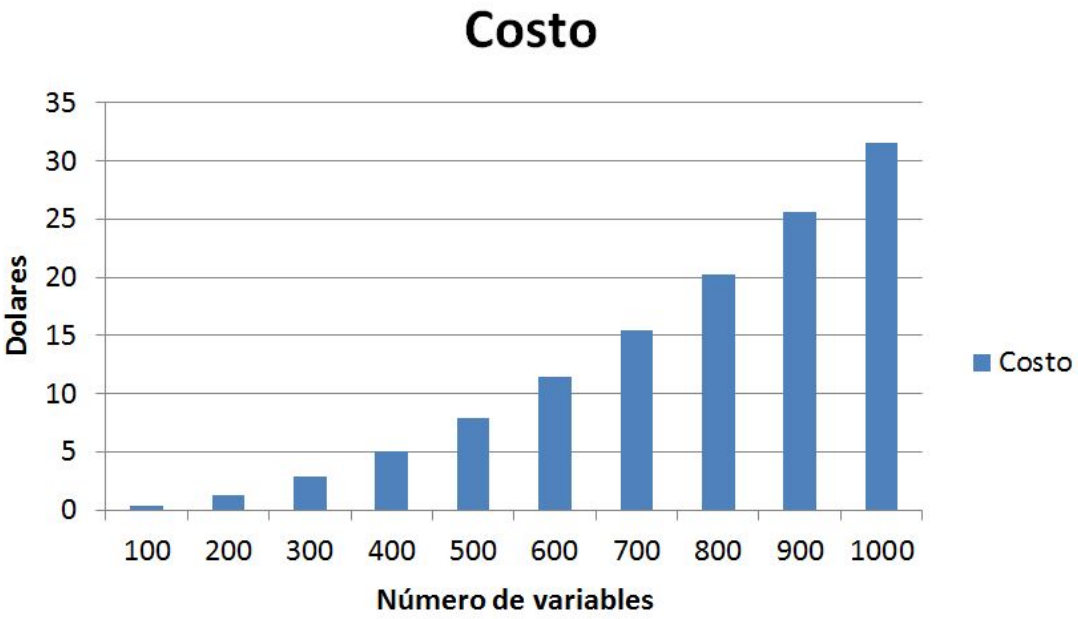


Figura 6.5: Costo en dolares.

Tabla 6.14: Costo total por el número de variables de modelo

Variables	**Tiempo Estimado(Horas)**	**Costo por hora(dolares)**	**Costo total**
100	0.24	$ 1.345	$ 0.33
200	0.95	$1.345	$ 1.29
300	2.13	$1.345	$ 2.87
400	3.78	$1.345	$ 5.09
500	5.89	$1.345	$ 7.93
600	8.47	$1.345	$ 11.40
700	11.52	$1.345	$ 15.50
800	15.03	$1.345	$ 20.22
900	19.01	$1.345	$ 25.57
1000	23.46	$1.345	$ 31.55

CAPÍTULO 7

Conclusiones y trabajos futuros

Se logró tener un prototipo funcional que provee algoritmos de minado de datos con dos modos de ejecución local, en la nube y sugiere el modo de ejecución más conveniente al utilizar diferentes parámetros para hacer una estimación del tiempo que tomará ejecutar determinado algoritmo. Asimismo puede gestionar diferentes procesos. Por otro lado se implementó el algoritmo SSVS que representó el algoritmo de mayor complejidad que se incorporó al software. De este algoritmo se determinó que la calidad de los modelos encontrados no necesariamente mejora si se aumenta la cantidad de muestras a generar de Gibbs.

Cuando los modelos contienen pocas variables predictoras y la cantidad de datos es grande es más conveniente ejecutar SSVS en el cliente debido a que la diferencia con ejecutarlo en el servidor es de unos cuantos segundos y esto puede variar dependiendo de la capacidad del procesamiento del cliente, mas como ya se mencionó anteriormente, se parte de la suposición de que el servidor tiene más recursos que el cliente, además la velocidad puede llegar a variar mientras se realiza la trasmisión de los datos.

La principal línea de investigación es realizar optimizaciones a la implementación de SSVS para evitar una lectura de los datos por cada iteración del algoritmo y así minimizar los accesos a disco. Respecto al software se puede trabajar en incorporar otros algoritmos de minado de datos. Respecto al planificador se pueden hacer mejoras al incorporar otros algoritmos de planificación para evitar que un proceso muy costoso no deje que procesos más ligeros puedan ejecutarse.

La visualización de los resultados es muy factible a ser mejorado, ya que el software se limita a dar una visualización básica, incluso se podría dar una opción a exportar los resultados fuera de la base de datos por medio del software.

BIBLIOGRAFÍA

[1] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[2] Carlos Ordonez, Naveen Mohanam, Carlos Garcia-Alvarado, Predrag T. Tosic, and Edgar Martinez. Fast pca computation in a dbms with aggregate udfs and lapack. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2219–2223, New York, NY, USA, 2012. ACM.

[3] Carlos Ordonez and Carlos Garcia-Alvarado. A data mining system based on sql queries and udfs for relational databases. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2521–2524. ACM, 2011.

[4] Carlos Ordonez and Sasi K Pitchaimalai. One-pass data mining algorithms in a dbms with udfs. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1217–1220. ACM, 2011.

[5] Carlos Ordonez and Sasi K Pitchaimalai. Fast udfs to compute sufficient statistics on large data sets exploiting caching and sampling. *Data & Knowledge Engineering*, 69(4):383–398, 2010.

[6] Mario Navas and Carlos Ordonez. Efficient computation of pca with svd in sql. In *Proceedings of the 2nd Workshop on Data Mining using Matrices and Tensors*, page 5. ACM, 2009.

[7] C. Ordonez. Statistical model computation with udfs. *Knowledge and Data Engineering, IEEE Transactions on*, 22(12):1752–1765, Dec 2010.

[8] Edward I George and Robert E McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

[9] Carlos Ordonez, Javier García-García, Carlos Garcia-Alvarado, Wellington Cabrera, Veerabhadran Baladandayuthapani, and Mohammed S. Quraishi. Data mining algorithms as a service in the cloud exploiting relational database systems. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1001–1004, New York, NY, USA, 2013. ACM.

[10] Carlos Ordonez. Building statistical models and scoring with udfs. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1005–1016. ACM, 2007.

[11] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition*, chapter 6. Singular Value Decomposition, pages 57–74. SIAM, 2007.

[12] Oscar Hérnadez Rodríguez. *Temas de Análisis Estadístico Multivariado*, chapter 1, pages 1–14. Universidad de Costa Rica, 1998.

[13] Cèsar Ferri Ramírez José Hernández Orallo, M.José Ramírez Quintana. *Introducción a la Minería de Datos*, chapter 16, page 428. Pea, 2004.

[14] Cèsar Ferri Ramírez José Hernández Orallo, M.José Ramírez Quintana. *Introducción a la Minería de Datos*, chapter 10, pages 257–260. Pe, 2004.