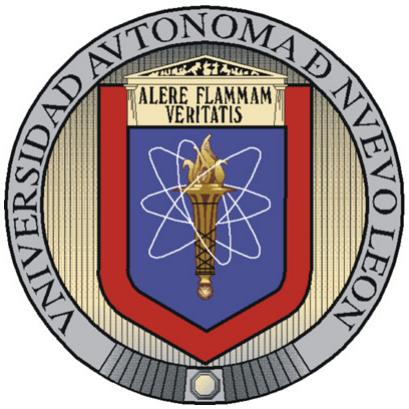
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



"Uso de técnicas de agregación en el diseño de clasificadores de vectores soporte"

Tesis profesional presentada por

Nancy Rocío Garza Padilla

en opción

al grado de Maestro en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, N.L. Otoño de 2005

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

"Uso de técnicas de agregación en el diseño de clasificadores de vectores soporte"

Tesis profesional presentada por

Nancy Rocío Garza Padilla

en opción
al grado de Maestro en Ciencias
en Ingeniería de Sistemas

San Nicolás de los Garza, N.L. Octubre de 2005

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Facultad de Ingeniería Mecánica y Eléctrica
División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la tesis **"Uso de Técnicas de Agregación en el Diseño de Clasificadores de Vectores Soporte"**, realizada por la alumna Nancy Rocío Garza Padilla, matrícula 0893474, sea aceptada para su defensa como opción al grado de Maestro en Ciencias en Ingeniería de Sistemas.

Comité de Tesis

Dr. Óscar Leonel Chacón Mondragón Asesor	Dr. Igor S. Litvinchev Revisor
Dr. Francisco R. Angel-Bello Acosta Revisor	Dr. Guadalupe Alan Castillo Rodríguez Subdirector División de Estudios de Posgrado

6 de Octubre de 2005

Gracias a todos los que hicieron esto posible; primero que nadie Gracias Dios por haber estado conmigo en este tiempo, gracias a mi familia que me apoyó y confió en mí y gracias a mis maestros, compañeros y amigos quienes estuvieron ahi para ayudarme y darme ánimo.

Índice general

Resumen

En un sistema eléctrico de potencia, los relevadores del sistema de protección de líneas de transmisión cortan la energía al momento de ocurrir una falla. Por tal motivo se necesita un procesamiento de señales eléctricas para la obtención de rasgos distintivos que permita una clasificación correcta del tipo de falla que ocurrió.

Existen técnicas de inteligencia artificial, como los mecanismos de vectores soporte (SVM) que son mecanismos de aprendizaje, que realizan eficientemente las tareas de clasificación binaria (reconocimiento de patrones) y son usados para el diagnóstico de fallas de un sistema en particular.

En este trabajo se propone como alternativa de solución la técnica SVM para generar una clasificación correcta del tipo de falla.

Con el fin de simular fallas en las líneas de transmisión, se utilizó el paquete computacional PSCAD (Power Systems Computer Aided Design) que realiza las simulaciones requeridas.

Durante las últimas dos decadas las técnicas agregación/desagregación se han convertido en una herramienta importante en investigación de operaciones, ambas para propósitos de modelación y para proveer soluciones aproximadas de problemas matemáticos a gran escala. Debido a que el problema al que se propone encontrar una solución es de gran escala, se considera la utilización de algoritmos “clustering”. Estos algoritmos son técnicas para agrupar objetos (caracterizados por una estructura de

datos) tales que, las similitudes de los objetos dentro de un mismo grupo sea mayor que aquéllos que están en los otros grupos.

Esta metodología propuesta consta de los siguientes pasos: (1) Generación de los datos de entrenamiento para cada tipo de falla; (2) Aplicación del algoritmo clustering a estos datos para reducir el tamaño del problema, quedando como datos sólo los prototipos; (3) Aplicación del algoritmo SVM a los prototipos resultantes para obtener un discriminante lineal; (4) Evaluación del discriminante generado con los datos del paso (1): si existen datos mal clasificados, se toman como datos de entrenamiento los datos que corresponden a los clusters que resultaron ser vectores soporte y se aplica el algoritmo SVM del paso (3), de lo contrario se continúa en el siguiente paso; (5) Termina el algoritmo.

Aplicando esta metodología a los resultados de las simulaciones de fallas realizadas en el paquete computacional PSCAD encontramos que el clasificador que se diseñó tuvo un buen comportamiento ya que no se encontró ningún error de clasificación y se hizo con un tiempo de cómputo eficiente aún para modelos muy grandes.

Capítulo 1

Introducción

La importancia de los métodos de clasificación de señales pueden ser vistos en una amplia área de aplicaciones industriales, siendo aquélla de nuestro interés la del diagnóstico de falla en sistemas eléctricos de potencia de alto voltaje.

Las líneas de transmisión para sistemas eléctricos de potencia juegan un rol muy importante en la industria, existiendo una gran demanda de confiabilidad y operación segura. Las fallas generadas por estos sistemas pueden incurrir en costos de millones de dólares.

Además de métodos basados en modelos tradicionales y sistemas expertos, los métodos de clasificación numérica se han vuelto muy populares en el diagnóstico de fallas de sistemas eléctricos. La clasificación basada en Mecanismos de Vectores Soporte nos proporciona una buena aproximación para diagnósticos de falla de sistemas de potencia en combinación con los métodos de aprendizaje no supervisado, como técnicas de clustering, que son más apropiadas para el manejo de información de este tipo de problemas de gran escala.

Figura 1.1: Líneas de Trasmisión

1.1. Motivación

La línea de transmisión es un elemento de gran importancia para el sistema eléctrico de potencia, ya que permite transferir energía eléctrica de las zonas de generación a las zonas de carga. Debido a que la línea de transmisión es un elemento de gran longitud generalmente se ve sometido a condiciones adversas, como viento, lluvia, nieve, accidentes, lo cual puede provocar cortocircuitos entre fases, conductores caídos a tierra u otro tipo de fallas. Cuando una falla de este tipo aparece en una línea es necesario desconectarla del resto del sistema, pues no hacerlo implicaría un esfuerzo considerable sobre elementos adyacentes, como transformadores y generadores, provocando una reducción considerable en su vida útil, además se pone en peligro la integridad del sistema.

Cualquier falla que aparezca en una línea de transmisión debe ser desconectada lo más pronto posible, con el propósito de reducir los efectos provocados por el disturbio; de no hacerlo puede dar origen a apagones generales como el que se registró el 14 de agosto 2003, afectando a Nueva York y otras ciudades del noreste de Estados Unidos y sureste de Canadá. El efecto en cadena fué provocado por una falla mal liberada dejando sin suministro de energía eléctrica a más de 50 millones de ususarios y pérdidas

multimillonarias a nivel comercial e industrial.

La creciente inclinación a instalar líneas de transmisión con voltajes cada vez mayores, así como la tendencia de operar el sistema eléctrico de potencia en un mercado desregulado, trae consigo la aparición de problemas nuevos y complicados desde el punto de vista de protecciones, tales como la necesidad de protección de fallas de manera ultra-rápida y selectiva; por lo tanto es necesario el desarrollo de nuevas propuestas de protección para su aplicación en lineas de transmisión.

Con base en lo anterior, la motivación central del tema de tesis consiste en proponer un algoritmo de identificación de fallas para la proteción de líneas de transmisión, basado en la medición de corriente por el relevador.

1.2. Antecedentes

Diferentes métodos basados en técnicas de inteligencia artificial se han hecho comunes en diagnóstico de fallas y monitoreo. Por ejemplo, Lógica Difusa y Redes Neuronales (NN) han sido usados en modelar y en la toma de decisiones en esquemas de diagnósticos. También, métodos de clasificación numérica son ampliamente usados en el área de modernos diagnósticos de fallas, y en particular, diagnósticos de falla en mecanismos. Por ejemplo, en [2] y [24] usaron clasificadores basados en NN para el diagnóstico de rodillos.

Clasificación basada en Mecanismos de Vectores Soporte (SVM) es un método de clasificación relativamente nuevo, y afirma tener mejores propiedades de generalización de clasificadores que NN. Otra interesante característica del clasificador basado en SVM es que su desempeño no depende del número de atributos de entidades clasificadas, es decir, la dimensión de los vectores clasificados. Eso es, porque tiende a ser especialmente eficiente en problemas de clasificación a gran escala. En procesos de diagnósticos de falla,

esta propiedad es muy útil porque el número de atributos elegidos para ser la base del diagnóstico es ilimitado.

SVM ha sido exitosamente aplicado a diagnósticos de falla en mecanismos eléctricos en [35] se usó este método para la clasificación de fallas en un motor de inducción.

Muchos trabajos de investigación se han enfocado en usar métodos de aprendizaje supervisado para construir modelos de predicción de fallas. A nuestra consideración, no ha habido mucho o muy poco esfuerzo dedicado a algoritmos de aprendizaje no supervisado.

1.3. Objetivo

El objetivo de esta tesis consiste en desarrollar un nuevo algoritmo para la protección de líneas de transmisión en sistemas eléctricos de potencia, que consiste en diseñar un clasificador para discriminación entre 2 clases basado en el riesgo regularizado.

1.4. Justificación del trabajo

Para problemas en gran escala, el problema de riesgo regularizado genera un modelo muy grande con un tiempo de cómputo alto conforme se incrementa el número de patrones de entrenamiento.

1.5. Planteamiento del problema

En la implementación de funciones de decisión para mecanismos de aprendizaje, una forma de limitar su capacidad es trazar un margen de separación grande entre las clases de datos. Para encontrar ese margen se usa el algoritmo SVM (Support Vector

Machines) que busca el hiperplano separador con mayor margen.

Para simplificar este problema, se consideraron técnicas de agregación y desagregación para los datos, lo cual consiste en dividir los puntos en subconjuntos (clusters) y tratarlo como una agrupación adaptativa lo cual hace el número de clusters fijo, pero los puntos que son agrupados están cambiando de iteración en iteración, para finalmente converger en clusters que contengan a los vectores soporte.

1.6. Hipótesis

Usando métodos de agregación se pueden obtener clasificadores basados en el riesgo regularizado de tal forma que en un proceso iterativo el clasificador se modifica hasta lograr la condición óptima final con un riesgo regularizado mínimo para los patrones de entrenamiento de ambas clases.

1.7. Estructura de la tesis

En el capítulo 1 se explicó la necesidad del desarrollo de algoritmos de detección de fallas en líneas de transmisión. También se dió un breve resumen de las propuestas más relevantes que han sugerido algunos autores sobre el tema.

En el capítulo 2 se ve detalladamente que son los vectores soporte y sus demostraciones matemáticas.

En el capítulo 3 se explican las funciones de agregación, que incluyen el clustering que fue el método que usamos para la resolución del problema.

En el capítulo 4 se analiza la prueba del método que utilizamos.

En el capítulo 5 se dan los resultados computacionales que se obtuvieron.

En el capítulo 6 se detallan las conclusiones, aportaciones científicas y recomenda-

ciones para trabajos futuros.

Capítulo 2

Mecanismos de Vectores Soporte

2.1. Introducción

En este capítulo se presenta la teoría de Mecanismos de Vectores Soporte y el modelo matemático del problema que estamos tratando.

Clasificación basada en Mecanismos de Vectores Soporte (SVM) es un método de clasificación relativamente nuevo, y afirma tener mejores propiedades de generalización de clasificadores que redes neuronales. Otra interesante característica del clasificador basado en SVM es que su desempeño no depende del número de atributos de entidades clasificadas, es decir, la dimensión de los vectores clasificados. Eso explica el porque tiende a ser especialmente eficiente en problemas de clasificación a gran escala. En procesos de diagnósticos de falla, esta propiedad es muy útil, porque el número de atributos elegidos para ser la base del diagnóstico es ilimitado.

SVM ha sido exitosamente aplicado a diagnósticos de falla en mecanismos eléctricos en [35] se usó este método para la clasificación de fallas en un motor de inducción; en nuestro trabajo se usa para clasificación de fallas en líneas de transmisión de un sistema eléctrico de potencia.

La clasificación basada en Mecanismos de Vectores Soporte (SVM) es un método

de mecanismo de aprendizaje relativamente nuevo basado en la teoría estadística de aprendizaje presentada por Vapnik [38].

2.2. Vectores Soporte

Básicamente, el mecanismo de vectores soporte es un mecanismo lineal con algunas propiedades. Para explicar como funciona, iniciaremos con el caso más sencillo que es el de datos separables. En este contexto la idea principal de un mecanismo de vectores soporte es construir un hiperplano como la superficie de decisión de tal manera que el margen de separación entre clases(positivas y negativas) es maximizado.

El mecanismo de vectores soporte es una implementación aproximada del método de minimización de riesgo regularizado. Este principio es basado en el hecho de que el rango de error de un mecanismo de aprendizaje en los datos de prueba es acotado por la suma del rango de error de entrenamiento y el término que depende de la dimensión Vapnik-Chervonenkis(VC). De acuerdo a esto, el mecanismo de vectores soporte provee un buen desempeño en problemas de clasificación de patrones a pesar del hecho de que no incorpora conocimiento al dominio del problema, lo cual es un atributo único en los mecanismos de vectores soporte.

2.2.1. Hiperplano óptimo para datos linealmente separables

Consideremos la muestra de entrenamiento $x_i(i = 1, ..., M)$ donde x_i son los datos de entrada n-dimensional pertenecientes a la $ClaseI$ y $ClaseII$ y $y_i = 1$ y $y_i = -1$ son las etiquetas asociadas a $ClaseI$ y $ClaseII$ respectivamente. Se asume que las clases son linealmente separables. La ecuación de una superficie de decisión en la forma de un

hiperplano que efectúa la separación es:

$$w^T x + b = 0$$

donde x es un vector de entrada, w es un vector n-dimensional de peso ajustable, y b es un escalar (inclinación o tendencia). El vector de peso w define la dirección (gradiente) del hiperplano separador $f(x)$ y b (bias) define la distancia del hiperplano desde el origen. Podemos escribirlo de esta manera

$$w^T x_i + b \geq 0 \quad \text{para} \quad y_i = +1$$
$$w^T x_i + b < 0 \quad \text{para} \quad y_i = -1$$

Para datos linealmente separables, podemos determinar un hiperplano $f(x)$ que separa las clases. Si los datos x pertenecen a la clase positiva ($ClaseI$), la función $f(x)$ del hiperplano separador cumple con la relación $f(x) \geq 0$ y si los datos pertenecen a la clase negativa ($ClaseII$) con la relación $f(x) < 0$. En general, se representa y cumple con las siguientes expresiones:

$$f(x) = w \cdot x + b = \sum_{j=1}^{n} w_j x_j + b \tag{2.1}$$

$$y_i f(x_i) = y_i(w \cdot x_i + b) \geq 0, \quad \text{para} \quad i = 1, ..., M, \tag{2.2}$$

El hiperplano separador que tiene la máxima distancia entre el hierplano y el dato mas cercano, esto es, el máximo margen, es llamado *hiperplano separador óptimo*. Un ejemplo del hiperplano separador óptimo de dos conjuntos de datos, es presentado en la figura 2.1.

Se considera que se tienen datos de clases separables y se dispone de datos de entrenamiento $\{(x_i, y_i)\}_{i=1}^{M}$. Se tiene un hiperplano separador $w^T x + b$, donde w es la normal al hiperplano, $\frac{|b|}{||w||}$ es la distancia perpendicular del hiperplano al origen y $||w||$ es la norma euclidiana de w. Sea $d_+(d_-)$ la distancia más corta del hiperplano de

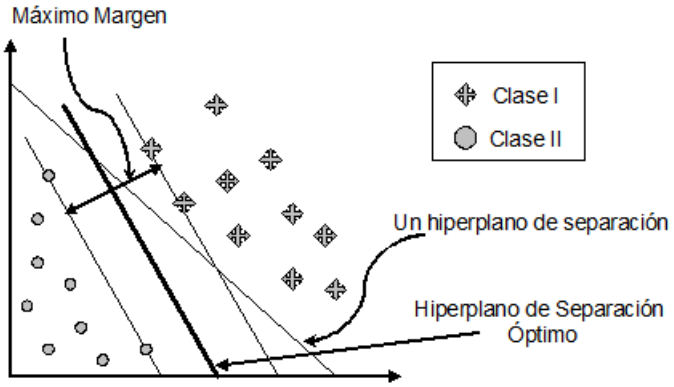


Figura 2.1: Hiperplano óptimo

separación al dato positivo(negativo) más cercano. Definamos el "margen" de separación como $d_+ + d_-$.

En la Figura 2.2 se tiene el hiperplano H_1 para el cual se cumple la igualdad $w^T x_i + b = 1$ para $y_i = +1$. Así mismo H_{-1} cumple con la igualdad $w^T x_i + b = -1$ para $y_i = -1$. El vector w es la normal a H_1 y la distancia perpendicular al origen es $\frac{|1-b|}{||w||}$. El vector w también es normal a H_{-1} y la distancia perpendicular al origen es $\frac{|-1-b|}{||w||}$. La distancia perpendicular de $H_0 = \{x|w^T x + b = 0\}$ al origen es $\frac{|-b|}{||w||}$, por lo tanto $d_+ = d_- = \frac{1}{||w||}$, y el margen de separación corresponde al valor de $\frac{2}{||w||}$.

Los hiperplanos H_1 y H_{-1} son paralelos y no existen datos entre ellos; por lo tanto, se puede obtener ambos hiperplanos que maximizen el margen de sepración mediante la minimización de $\frac{1}{2}||w||^2$.

El hiperplano óptimo puede ser obtenido resolviendo el siguiente problema de optimización cuadrático convexo [1]:

$$\begin{aligned} \text{minimizar} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{sujeto a} \quad & y_i(w^T x_i + b) \geq 1 \end{aligned} \tag{2.3}$$

Los puntos de entrenamiento para los cuales se cumplen las igualdades de las restricciones de 2.3 y cuya eliminación cambia la solución al problema, se les llama Vectores

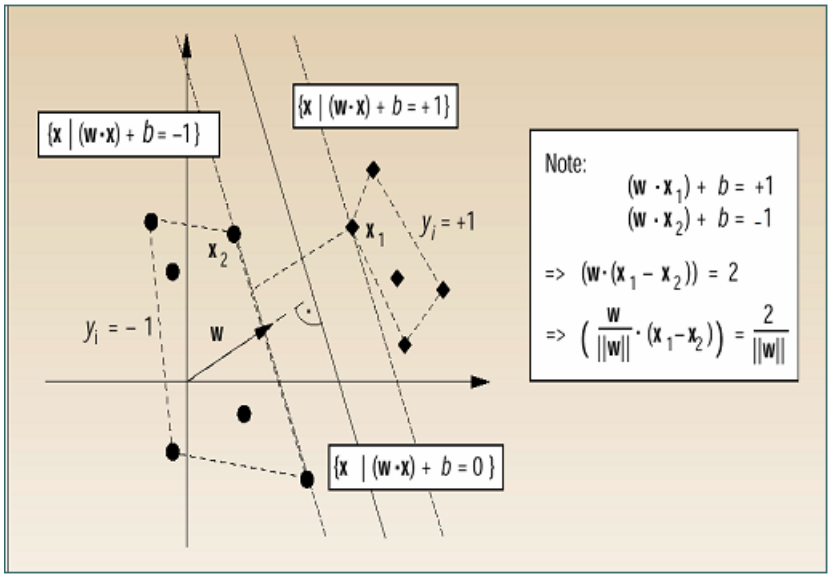


Figura 2.2: Vectores e Hiperplanos Soporte

Soporte.

2.2.2. Optimización Cuadrática para encontrar el hiperplano óptimo

Nuestro objetivo es desarrollar un procedimiento computacionalmente eficiente usando la muestra de entrenamiento $\{(x_i, y_i)\}_{i=1}^{M}$ para encontrar el hiperplano óptimo, sujeto a la restricción $y_i(w^T x_i + b) \geq 1$ para $i = 1, 2, ..., M$.

El problema de optimización que tenemos que resolver, puede establecerse como sigue:

Dada la muestra de entrenamiento $\{(x_i, y_i)\}_{i=1}^{M}$ *encontrar los valores de* w *y bias* b *tales que satisfagan las restricciones*

$$y_i(w \cdot x_i + b) \geq 1 \text{ para } i = 1, 2, ..., M$$

y se minimice la función de costo:

$$\phi = \frac{1}{2} w^T w$$

Este problema de optimización es llamado el problema primal. Este está caracterizado como sigue:

- La función de costo ϕ es una función convexa de w.
- Las restricciones son lineales en w.

Si el número de atributos de la muestra es grande, podemos simplificar considerablemente los cálculos convirtiendo el problema mediante las condiciones de Kuhn-Tucker en el equivalente problema dual Lagrangiano[20]. De acuerdo a esto, podemos resolver este problema de optimización usando el método de los multiplicadores de Lagrange [6]. La función de lagrange para 2.3 es:

$$L(w, b, \alpha) = \frac{1}{2}(w^T w) - \sum_{i=1}^{M} \alpha_i [y_i((w_i^T x_i) + b) - 1], \tag{2.4}$$

donde $\alpha = (\alpha_1, ..., \alpha_M)$ son los multiplicadores de Lagrange. La solución al problema de optimización es determinada por el punto estacionario de la función Lagrangiana $L(w, b, \alpha)$, el cual tiene que ser minimizado con respecto a w y b; también tiene que ser maximizada con respecto a α. De este modo, si derivamos 2.4 con respecto a w y b e igualando los resultados a cero, obtenemos las siguientes dos condiciones de optimalidad:

$$\text{Condición 1:} \quad \frac{\partial L}{\partial w}(w, b, \alpha) = 0 \tag{2.5}$$

$$\text{Condición 2:} \quad \frac{\partial L}{\partial b}(w, b, \alpha) = 0 \tag{2.6}$$

Aplicando 2.5 y 2.6 a 2.4 se produce lo siguiente:

$$w = \sum_{i=1}^{M} y_i \alpha_i x_i \tag{2.7}$$

$$\sum_{i=1}^{M} y_i \alpha_i = 0 \tag{2.8}$$

El problema trabaja con una función de costo convexa y restricciones lineales. Dado un problema de optimización con restricciones, es posible construir otro problema llamado problema dual. Este segundo problema tiene el mismo valor óptimo que el problema primal, pero con los multiplicadores de Lagrange proveyendo la solución óptima.

Para hacer el problema dual de nuestro problema primal, primero expandimos la ecuación 2.4 término por término, como sigue:

$$L(w, b, \alpha) = \frac{1}{2}(w^T w) - \sum_{i=1}^{M} \alpha_i y_i w^T x_i - b \sum_{i=1}^{M} \alpha_i y_i + \sum_{i=1}^{M} \alpha_i \tag{2.9}$$

El tercer término (de izquierda a derecha) de 2.9 es cero, esto es por la condición de optimalidad 2.8. Además, de 2.7 tenemos,

$$w^T w = \sum_{i=1}^{M} \alpha_i y_i w^T x_i = \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j x_j^T x_i \tag{2.10}$$

Con todo lo anterior, podemos establecer el problema dual como sigue:

Dada la muestra de entrenamiento $\{(x_i, y_i)\}_{i=1}^{M}$*, encontrar los multiplicadores de Lagrange* $\alpha = (\alpha_1, ..., \alpha_M)$ *que maximice la función objetivo*

$$\begin{aligned} &\text{maximizar} \quad W(\alpha) = \textstyle\sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ &\text{sujeto a} \qquad \textstyle\sum_{i=1}^{M} y_i \alpha_i = 0 \\ &\qquad\qquad \alpha_i \geq 0 \quad \text{para } \ i = 1, ..., M \end{aligned} \tag{2.11}$$

Note que, el problema dual está enteramente en términos de los datos de entrenamiento (el número de variables del problema dual es el número de datos de entrenamiento). Por otra parte, la función $W(\alpha)$ a ser maximizada depende sólo de los patrones de entrenamiento en la forma de un conjunto de producto escalar.

Podemos establecer el siguiente teorema de dualidad 2.1 [6]:

Theorem 2.1 (Teorema de Dualidad). *1. Si el problema primal tiene una solución óptima, el problema dual también tiene una solución óptima, y los correspondientes valores óptimos son iguales.*

2. Para que w^* sea una solución óptima primal y α^* sea una solución óptima dual, es necesario y suficiente que w^* sea factible para el problema primal, y

$$\phi(w^*) = L(w^*, b^*, \alpha^*) = min_w L(w, b^*, \alpha^*)$$

Asumamos que la solución óptima del problema dual es α^*. De acuerdo al teorema de Karush-Kuhn-Tucker la condición de igualdad en 2.2 se sostiene para (x_k, y_k) sólo si el α_k^* asociada no es 0. En este caso, la muestra de entrenamiento x_k es un vector soporte. Resolviendo 2.11 para $\alpha = (\alpha_1, ..., \alpha_M)$, podemos obtener los vectores soporte para la $ClaseI$ y $ClaseII$. Entonces, el hiperplano separador óptimo es puesto en igual distancia entre los vectores soporte de las clases I y II, y b^* es dado por:

$$b* = -\frac{1}{2}\sum_{k=1}^{M} y_k \alpha_k^* (s_1 \cdot x_k + s_2 \cdot x_k), \tag{2.12}$$

donde s_1 y s_2 son vectores soporte arbitrarios para las clases I y II respectivamente.

2.2.3. Hiperplano óptimo para datos linealmente no separables

La discusión hasta este momento se ha enfocado a patrones linealmente separables. Sin embargo, es necesario considerar soluciones en las cuales pueden existir errores en la clasificación. Consideremos en estos términos el concepto de *márgen de separación suave*. El margen de separación entre clases se dice que es *suave* si un punto en los datos (x_i, y_i) viola la condición $y_i(w^T x_i + b) \geq 1$.

Esta violación puede representarse de dos formas:

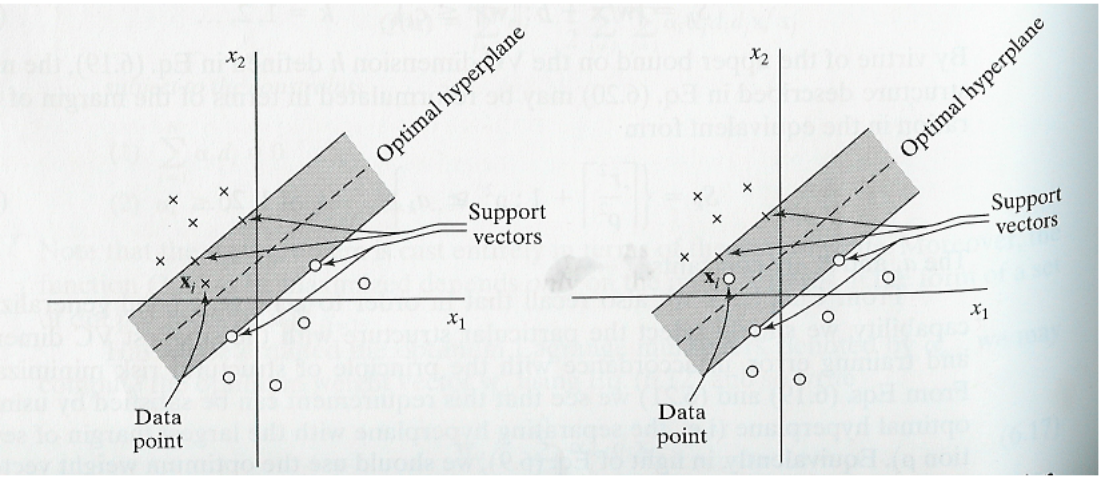


Figura 2.3: Hiperplano Óptimo

1. Si el punto (x_i, y_i) cae dentro de la región de separación pero en el lado correcto de la superficie de desición Figura 2.3(izquierda).

2. Si el punto (x_i, y_i) cae en el lado equivocado de la superficie de decisión Figura 2.3(derecha).

En la Figura 2.3 el punto x_i cae dentro de la región de separación, en el lado correcto de la superficie de decisión (figura izquierda). El punto x_i cae en el lado equivocado de la superficie de decisión (figura derecha). Se observa que tenemos correcta clasificación en el caso 1 pero mala en el caso 2.

En estas condiciones, sería deseable encontrar el hiperplano óptimo que minimice la probabilidad de error de clasificación, por consiguiente, introducimos un nuevo conjunto de variables escalares no negativas, $\{\epsilon_i\}_{i=1}^{M}$ dentro de la definición del hiperplano separador como se muestra a continuación:

$$y_i(w^T x_i + b) \geq 1 - \epsilon_i, \quad i = 1, ..., M \tag{2.13}$$

Las variables ϵ_i son llamadas *variables de holgura*; ellas miden la desviación del punto de la condición ideal del patrón de separación. Para $0 \leq \epsilon_i \leq 1$, el punto cae dentro de la

región de separación en el lado correcto de la superficie de decisión Figura 2.3(izquierda). Para $\epsilon_i > 1$, cae en el lado equivocado del hiperplano de separación Figura 2.3(derecha). Los vectores soporte son aquéllos puntos que satisfacen 2.13. Observa que si una muestra con $\epsilon_i \geq 0$ es dejada afuera del conjunto de entrenamiento, la superficie de decisión debería cambiar. Los vectores soporte están así definidos en exactamente la misma manera para ambos linealmente y no linealmente separables.

Nuestro objetivo es encontrar un hiperplano separador para el cual el error de clasificación en el conjunto de entrenamiento es minimizado. Nosotros podemos hacer esto minimizando el funcional

$$\phi(\epsilon) = \sum_{i=1}^{M} I(\epsilon_i - 1)$$

con respecto al vector de peso w, sujeto a la restricción descrita en 2.13 y la restricción en $\|w\|^2$. La función $I(\epsilon)$ es un indicador de función, definido por

$$I(\epsilon) = \begin{cases} 0 & \text{Si} \quad \epsilon \leq 0 \\ 1 & \text{Si} \quad \epsilon > 0 \end{cases}$$

Desafortunadamente, la minimización de $\phi(\epsilon)$ con respecto a w es un problema de optimización no convexo llamado NP-completo [10] [11] [18].

Para hacer el problema de optimización matemáticamente tratable, aproximamos el funcional $\phi(\epsilon)$ escribiendo

$$\phi(\epsilon) = \sum_{i=1}^{M} \epsilon_i$$

Además, una manera de simplificar los cálculos es formulando el funcional a ser minimizado con respecto al vector de peso w como sigue:

$$\phi(w, \epsilon) = \frac{1}{2} w^T w + C \sum_{i=1}^{M} \epsilon_i \tag{2.14}$$

Como hemos visto antes, al minimizar el primer término de 2.14 estamos minimizando la dimensión VC del mecanismo de vector soporte, y el segundo témino es una cota

superior del número de errores de la prueba. El teorema de Vapnik [38] establece lo siguiente:

Theorem 2.2 (Teorema de Vapnik)**.** *: Sea D el diámetro de la bola más pequeña que contiene todos los vectores de datos $x_1, x_2, ..., x_N$. El conjunto de los hiperplanos óptimos descrito por la ecuación*

$$w_0^T x + b_0 = 0$$

tiene una dimensión VC llamada h acotada por arriba

$$h \leq min\left\{\left\lceil\frac{D^2}{\rho^2}\right\rceil, m_0\right\} + 1 \tag{2.15}$$

donde el signo de techo $\lceil\cdot\rceil$ significa el número entero más pequeño mayor o igual al número de los datos que se encuentran dentro, ρ es el margen de separación igual a $2/||w_0||$, y m_0 es la dimensionalidad del espacio de entradas.

El parámetro C controla el intercambio entre la complejidad del mecanismo y el número de puntos no separables; esto puede ser visto también como un parámetro de "regularización". Este parámetro tiene que ser seleccionado por el usuario y puede hacerse de dos formas:

- El parámetro C es determinado experimentalmente vía el uso estándar de un conjunto de prueba(validación)/entrenamiento, la cual es una forma cruda de remuestreo.

- Es determinado analíticamente estimando la dimensión VC via 2.15 y así usando las cotas en el desarrollo del mecanismo basado en la dimensión VC.

En cualquier evento, el funcional $\phi(w, \epsilon)$ es optimizado con respecto a w y ϵ_i, sujeto a 2.13 y $\epsilon_i \geq 0$. Ahora, la norma cuadrada de w es tratada como una cantidad a ser

minimizada con respecto a los puntos no separables en vez de una restricción impuesta en la minimización del número de puntos no separables.

Con todo lo anterior, podemos establecer el problema primal para el caso de datos no separables como:

Dada una muestra de entrenamiento $\{(x_i, y_i)\}_{i=1}^{M}$*, encontrar el valor de* w *y bias* b *tales que satisfagan las restricciones*

$$\begin{aligned} y_i(w^T x_i + b) &\geq 1 - \epsilon_i, \quad i = 1, ..., M \\ \epsilon_i &\geq 0 \qquad\qquad \text{para toda } \ i \end{aligned}$$

tal que se minimice el costo funcional

$$\phi(w, \epsilon) = \frac{1}{2} w^T w + C \sum_{i=1}^{M} \epsilon_i$$

Usando el método de los multiplicadores de lagrange y procediendo en una manera similar a lo descrito en la sección anterior, podemos formular el problema dual para caso de datos no separables como sigue:

Dada una muestra de entrenamiento $\{(x_i, y_i)\}_{i=1}^{M}$*, encontrar los multiplicadores de Lagrange* $\alpha = (\alpha_1, ..., \alpha_M)$ *que maximice la función objetivo*

$$\text{maximizar} \quad W(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\begin{aligned} \text{sujeto a} \qquad & \sum_{i=1}^{M} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \ \text{ para } \ i = 1, ..., M \end{aligned}$$

donde C *es un parámetro positivo especificado por el usuario.*

El problema dual para el caso de patrones no separables es similar para el caso de patrones separables excepto por una pequeña, pero muy importante diferencia. La función objetivo $W(\alpha)$ a ser maximizada es la misma en ambos casos. El caso de no

separables difiere del separable en que la restricción $\alpha_i \geq 0$ es reemplazada con la restricción $0 \leq \alpha_i \leq C$.

Capítulo 3

Funciones de agregación

3.1. Introducción

Agregación es una herramienta para resolver problemas a gran escala reduciendo el tamaño de estos. La agregación significa que un conjunto de puntos con similaridad especifica se pueden agrupar y representar con un prototipo de igual dimensión. La posición geométrica del punto agregado (prototipo) está definida mediante los pesos de la agregación como cierta combinación convexa de los puntos que forman el cluster. De esta manera, en el problema agregado (o dual agregado) la búsqueda del hiperplano separador con mayor margen se reduce a solamente dos puntos agregados.

Este problema tiene la solución que es fácil encontrar (ya que tiene solamente 2 variables). Existen varias maneras de representar el punto agregado mediante la combinación convexa de los puntos que forman el cluster; para cierta combinación de los pesos existe cierto valor del margen de separación para el correspondiente hiperplano separador.

Es importante que después de la resolución del problema se puedan calcular las cotas de la agregación, o bien que se pueda estimar la diferencia entre el valor óptimo agregado y el valor óptimo verdadero. Si la diferencia es suficientemente pequeña, se

puede parar y considerar la solución desagregada actual como la solución aproximada del problema.

En el procedimento arriba referido, los clusters son fijos y los puntos que están agrupados en el punto prototipo son los mismos y no cambian de iteración en iteración.

Otra dirección interesante para la investigación es la agrupación (clustering) adaptativa. En este caso el número de clusters es fijo pero los puntos, que son agrupados, están cambiando de iteración en iteración. En otras palabras, en cada iteración los conjuntos I+; I-; están re-definidos junto con los pesos de agregación.

3.2. Clustering

3.2.1. Introducción

Los métodos de clustering son usados para exploración de datos y proveen prototipos que son usados en los clasificadores. Análisis de cluster [15] [21] [22] [36] es un método estadístico multivariado que agrupa un conjunto de objetos, como una colección de datos, tal que cada objeto pertenece solo a un grupo (o cluster) y la unión de los grupos contiene todos los objetos; un ejemplo de ésto es la Figura 3.1. La agrupación debe ser tal que todos los objetos dentro de un grupo en particular tengan mayor similitud entre ellos que entre los objetos de otros grupos. Análisis de cluster es un método bien establecido que tiene numerosas aplicaciones; las más antiguas se encuentran en las áreas de biología y zoología, pero estos métodos también han sido aplicados en los campos de arqueología, astronomía, medicina, investigación de mercado, lingüística, procesos de señales, entre otras [15] [40].

El análisis de cluster es también considerado como uno de los métodos de aprendizaje no supervisado. El término “aprendizaje no supervisado” o “aprendizaje sin un

maestro" está generalmente asociado con la idea de usar una colección de observaciones $X_1, ..., X_n$ como muestra de una distribución p(X), que describe las propiedades de p(X). Esta definición es muy genérica ya que el término por si mismo no tiene mucho significado, y necesita estar orientado apropiadamente. En la práctica, el término universalmente se refiere a:

- Clustering: Procedimiento que identifica "grupos" en un conjunto de datos.
- Descomposición de mezcla: Área de la estadística que se centra en identificar las densidades paramétricas de poblaciones individuales compuesta de una "super-población".
- Descubrimiento de la regla de asociación: Una técnica de minería de datos que encuentra una colección de atributos que frecuentemente aparecen juntos en las observaciones.
- Escalamiento multidimensional: El proceso de identificar un espacio Euclidiano de dimensiones pequeñas y posiblemente un mapeo no lineal del espacio original al nuevo espacio, en una manera tal que la distancia entre los pares de puntos de entrenamiento en el espacio original, es casi igual a las distancias entre sus proyecciones.

Los resultados del análisis de cluster producen una estructura identificable que puede ser usada para generar hipótesis de los datos observados. Es difícil dar una definición universal del término "cluster"(grupo); todos los métodos producen una partición de un conjunto de datos (una división de los datos dentro de grupos mutuamente excluyentes).

Algunos métodos supervisados tales como reglas de decisión y generador de árboles[30], construyen modelos de clasificación de clases de datos etiquetados. Si la composición y posibilidad del número de grupos que están en el cluster es conocido a priori, los

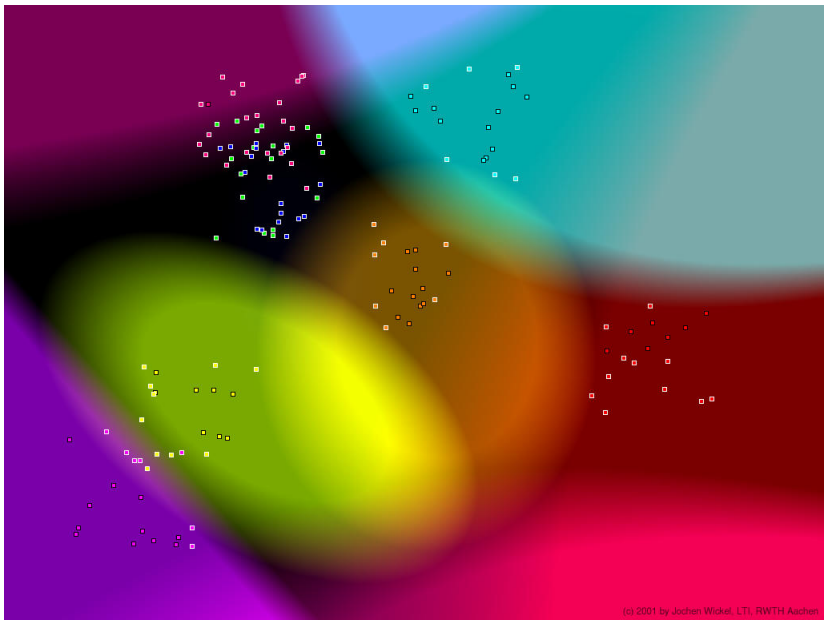


Figura 3.1: Ejemplo de clusters

métodos estadísticos supervisados y mecanismos de aprendizaje, son fundamentalmente diferentes de la técnica de análisis de cluster. Ambos métodos, supervisados y no supervisados, han sido extensamente aplicados. Muchos trabajos de investigación se han enfocado en usar métodos de aprendizaje supervisado para construir modelos de predicción de fallas. A la consideración de algunos autores [45], no ha habido mucho o es muy poco el esfuerzo dedicado a algoritmos de aprendizaje no supervisado.

Una gran variedad de técnicas han sido desarrolladas para hacer predicciones; Clasificación Difusa es una de ellas. Clustering Difuso fue usado por Yuan [42] para predecir el número de fallas en módulos, en este trabajo se proporciona un conjunto de patrones que reflejan el conocimiento del experto de como asignar etiquetas al conjunto de datos. Esta técnica es muy efectiva; es dificil derivar reglas difusas basadas en la experiencia, es mejor construirlas usando procesos de entrenamiento.

Otra técnica para resolver el problema de clasificación y clustering es basado en el algoritmos conexionistas o redes neuronales. El concepto no es nuevo (McCulloch

y Pitts sugirieron la descripción de una neurona como un límite lógico en 1943). El elemento escencial de una red neuronal, es la neurona. Una neurona típica j recibe un conjunto de señales x_i como datos de otras neuronas conectadas, cada una de las cuales es multiplicada por un factor de peso w_{ij}. Toda la activación de los pesos son sumados para producir la activación del nivel para la neurona j.

Los ajustes de los pesos de los nodos de la red neuronal permite a la red "aprender". En aprendizaje supervisado, un conjunto de muestras de entrenamiento es presentado uno por uno a la red. La red después calcula las salidas basada en sus datos actuales. La salida resultante es comparada con una salida deseada para una muestra de entrenamiento particular. Los pesos de la red son entonces ajustados para reducir el error.

Para el clustering, Stepp(1987) lo describe como la nueva frontera de la Inteligencia Artificial. El proceso de clustering ocurre en dos fases:

1. La fase de agregación

2. La fase de caracterización

La fase de agregación produce los grupos y la fase de caracterización intenta asignar alguna interpretación significativa a los grupos. Usando esas técnicas, las clases de objetos similares son básicamente encontradas haciendo comparaciones entre todos los elementos de datos, éstos algoritmos de clustering son llamados seriales.

En aprendizaje no supervisado, los modelos de la red son primero presentados como un vector de entrada de un conjunto de posibles redes. El aprendizaje de la red ajusta los pesos de tal forma que las muestras de entrada son agrupadas en clases basado en sus propiedades estadísticas (Dalton y Deshmane,1991;Rumelhart, Hinton, y Williams, 1986).

3.2.2. Técnicas de Clustering

Cuando se aplica una técnica de clustering es necesario entender el comportamiento (significado) de los datos y encontrar la técnica en particular que mejor actúe sobre los datos[39]. Algunas técnicas de clustering son las siguientes:

1. *Métodos jerárquicos*, el cual deriva un clustering de una matriz de disimilitudes dada;

2. *Particiones rápidas*, métodos para obtener una partición como una inicialización a aproximaciones más elaboradas;

3. *Modelos de mezcla*, los cuales expresan la función de densidad de probabilidad como una suma de densidades componentes;

4. *Métodos de la suma de cuadrados*, el cual minimiza un criterio de error de la suma de cuadrados, incluye el k-means, fuzzy k-means, cuantificación del vector y cuantificación estocástica del vector;

5. *Validación del cluster*, nos lleva al problema de la selección del modelo.

Métodos jerárquicos

Los procedimientos de clustering jerárquico son los métodos más comunmente usados en estructura de datos. Un árbol jerárquico es un conjunto anidado de particiones representado por un diagrama de árbol o dendograma que se muestra en la Figura 3.2. Seccionando un árbol en un nivel en particular produce una partición de k grupos disjuntos. Si dos grupos son elegidos de diferentes particiones (el resultado de particiones a diferentes niveles) entonces cualquiera de los grupos son disjuntos o un grupo contiene completamente el otro. Un ejempo de clasificación jerárquica es la clasificación del reino

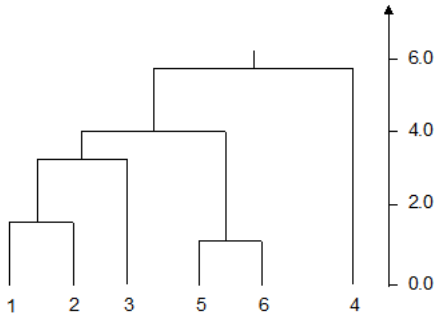


Figura 3.2: Dendograma

animal. Cada serie pertenece a una serie de clusters anidados de tamaño creciente con un número decreciente de características comunes. Para producir un diagrama de árbol es necesario ordenar los puntos de tal manera que las ramas no se crucen. Este ordenamiento es un tanto arbitrario pero no altera la estructura del árbol, sólo su apariencia. Hay un valor numérico asociado con cada posición arriba del árbol donde las ramas se unen. Ésta es una medida de la distancia o disimilitud entre dos clusters unidos.

Hay muchos algoritmos para encontrar un árbol jerárquico. Un *Algoritmo Aglomerativo* inicia con n subclusters, cada uno conteniendo solo un dato como punto, y en cada estado se unen los dos grupos más similares para formar un nuevo cluster, y así reducir el número de ellos. El algoritmo procede hasta que todos los datos caen dentro de un solo cluster. Un *algoritmo divisivo* opera por grupos partidos sucesivamente, iniciando con un solo grupo y continuando hasta que haya n grupos, cada uno con un elemento individual. Generalmente, algoritmos divisivos son computacionalmente ineficientes (excepto cuando la mayoría de las variables son binarias).

Del diagrama de árbol se puede definir un nuevo conjunto de distancias entre individuos, con la distancia entre el individuo i y el individuo j siendo la distancia entre

los dos grupos que los contengan, cuando éstos grupos son amalgamados (la distancia del nivel del enlace más bajo unido a ellos). Así, el procedimiento para encontrar un diagrama de árbol puede ser vista como una transformación del conjunto original de disimilitudes d_{ij} a un nuevo conjunto $\hat{d}_{ij}$, donde $\hat{d}_{ij}$ satisface la *desigualdad ultramétrica*

$$\hat{d}_{ij} \leq max(\hat{d}_{ik}, \hat{d}_{jk}) \text{ para todos los objetos } i, j, k$$

Esto significa que la distancia entre tres grupos puede ser usada para definir un triángulo que es equilátero o isoscéles (cualquiera de las tres distacias son iguales o dos son iguales y la tercera es menor). Una transformación $D : d \rightarrow \hat{d}$ es llamada como *transformación ultramétrica*.

Es apropiado aquí introducir el concepto de *árbol de mínima expansión*. Un árbol de mínima expansión no es un árbol jerárquico, pero un árbol expande un conjunto de puntos tales que cada dos vértices son conectados y no existen ciclos. Asociado con cada enlace en el árbol hay un valor o distancia, y el árbol de expansión es aquel tal que la suma de todas las distancias es mínima.

Particiones rápidas

Estas técnicas requieren una partición inicial de los datos. Los métodos de mezcla de distribuciones normales de probabilidad requieren estimados iniciales de la media y varianza de las matrices. Esta estimación podría ser basada en la muestra derivada de una partición inicial. El algoritmo k-means también requiere un conjunto inicial de medias. Cuantificación del vector jerárquico requiere estimados iniciales de los vectores clave, y similarmente los mapeos topográficos requieren inicialización de los vectores de peso.

Supongamos que tenemos un conjunto de n muestra de datos y deseamos encontrar una partición inicial de k grupos, o encontrar k vectores semilla. Se puede encontrar un

vector, dado un grupo de objetos, tomando la media del grupo. También podemos hacer una partición de un conjunto dado, con k vectores, usando una regla de asignación del vecino mas cercano. Hay muchos métodos heurísticos de partición, sólo consideraremos algunos de ellos.

1. **Selección aleatoria de k** Si se desea tener k diferentes vectores, se selecciona uno aleatoriamente de todo el conjunto de datos, otro de las restantes $n-1$ muestras en el conjunto de datos y así sucesivamente.

2. **División de la variable** Se elige una sola variable. Esta puede ser seleccionada de una de las variables medidas o de una combinación lineal de las variables. Se divide en k intervalos iguales que expanden el rango de la variable. Los datos son particionados de acuerdo al grupo en que ellos caen y k vectores semilla son encontrados de las medias de cada grupo.

3. **Algoritmo líder** El algoritmo cluster líder (Hartigan,1975; Spath,1980) particiona un conjunto de datos de tal forma que para cada grupo hay un objeto líder y todos los demás objetos dentro del grupo están dentro de una distancia T del líder. Algunas notas importantes de éste algoritmo son las siguientes:

   - Todos los centros de los clusters están al menos a una distancia T de cada uno de ellos.
   - Es rápido y requiere sólo una operación a través del conjunto de datos.
   - Es aplicable a una matriz de disimilitudes dada.
   - Es dependiente del orden del conjunto de datos. El primer punto es siempre un cluster líder. Los clusters iniciales tienden a ser más largos que los últimos.
   - La distancia T es especificada, no el número de clusters.

Modelos de mezcla

En el método de mezcla de clustering, cada grupo en la población se describe por una distribución de probabilidad diferente. Alternadamente, las mezclas pueden constar de las sumas de diferentes componentes de densidad (para modelar diferentes efectos tales como una señal o ruido). La población es descrita por una *mezcla finita de distribución* de la forma

$$p(x) = \sum_{i=1}^{k} \pi_i p(x; \theta_i)$$

donde π_i son las proporciones de mezcla ($\sum_{i=1}^{k} \pi_i = 1$) y $p(x; \theta_i)$ es una función de probabilidad p-dimensional dependiendo de un vector de parámetro θ_i. Hay tres conjuntos de parámetros a estimar: los valores de π_i, los componentes del vector θ_i y el valor de k, el número de grupos en la población.

Métodos de la suma de cuadrados

Los métodos de la suma de cuadrados encuentran una partición de los datos que maximice un criterio de clustering predefinido en base a la clase contenida y entre clases de matrices dispersas. Los métodos difieren en la elección del criterio de clustering optimizado y el procedimiento de optimización adoptado. Sin embargo, el problema que todos los métodos buscan resolver es: dado un conjunto de n datos de muestra, hacer una partición de k clusters de tal manera que el criterio de clustering es optimizado.

La mayoría de los métodos son subóptimos. Los requerimientos computacionales prohiben esquemas óptimos, aún para valores moderados de n. Por lo tanto requerimos métodos que, aunque produzcan una partición subóptima, den un valor del criterio de clustering que no sea mucho mayor que el óptimo.

Criterio de Clustering

Sea n muestra de datos $x_1, ..., x_n$. La matriz de covarianza $\hat{\Sigma}$ de la muestra está dada por

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)(x_i - m)^T$$

donde $m = \frac{1}{n}\sum_{i=1}^{n} x_i$ es la media de la muestra. Sean k clusters. La matriz dispersa dentro-clase (within-cluster) es:

$$S_W = \frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n} z_{ji}(x_i - m_j)(x_i - m_j)^T,$$

la suma de las sumas de cuadrados y las matrices de los productos cruzados sobre los k grupos, donde $z_{ji} = 1$ si x_i pertenece al grupo j, 0 de otra forma, $m_j = \frac{1}{n_j}\sum_{i=1}^{n} z_{ij}x_i$ es la media del cluster j y $n_j = \sum_{i=1}^{n} z_{ji}$ es el número de datos x_i en el cluster j. La matriz dispersa entre-clases(between-class) es:

$$S_B = \hat{\sum} - S_W = \sum_{j=1}^{k}\frac{n_j}{n}(m_j - m)(m_j - m)^T$$

y describe la dispersión de las medias de los clusters con respecto a la media total.

En los métodos de clustering se buscan clusters que sean internamente cohesivos pero aislados de otros clusters, pero no se conoce el número de clusters. He aquí algunos criterios de optimización del método de clustering:

1. **Minimización de** $Tr(S_W)$ El trazo de S_W es la suma de los elementos diagonales $Tr(S_W) = \frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n} z_{ji}||x_i - m_j||^2$

   $$Tr(S_W) = \frac{1}{n}\sum_{j=1}^{k} S_j$$

   donde $S_j = \sum_{i=1}^{n} z_{ji}||x_i - m_j||^2$ es la suma de los cuadrados del grupo j. Así, la minimización de $Tr(S_W)$ es equivalente a minimizar el total de la suma de los

cuadrados de cada grupo para todos los centroides k. Los métodos de clustering que minimiza ésta cantidad son algunas veces llamados como "suma de cuadrados" o "métodos de mínima varianza".

2. **Minimización de** $||S_W||/|\hat{\Sigma}||$ Este criterio es invariante a una transformación lineal no singular de los datos. Para un conjunto de datos dado es equivalente a encontrar la partición de los datos que minimice $||S_W||$ (La matriz $\hat{\Sigma}$ es independiente de la partición).

3. **Maximización de** $Tr(S_W^{-1}S_B)$ Este es una generalización del método de la suma de cuadrados en el cual los clusters no son hiperesferas sino hiperelipsoides. Este método es equivalente a minimizar la suma de los cuadrados bajo la medida de Mahalanobis. También es invariante a una transformación lineal no singular de los datos.

4. **Minimización de** $Tr(\hat{\Sigma}^{-1}S_W)$ Este es idéntico a la minimización de la suma de los cuadrados para los datos que han sido normalizados para hacer la matriz de dispersión total igual a la identidad.

La búsqueda de una partición no trivial de n objetos dentro de k grupos para los cuales el criterio elegido es optimizado, da lugar a un problema de optimización combinatoria. Encontrar la partición óptima requiere la examinación de cada posible partición. El número de particiones no triviales de n objetos dentro de k grupos es

$$\frac{1}{k}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}i^n$$

con el término final de la sumatoria siendo el más significante si $n >> k$. Esto se incrementa rápidamente de acuerdo al número de objetos. Por ejemplo, hay $2^{59}-1 \approx 6\times 10^{17}$ particiones de 60 objetos dentro de 2 grupos. Esto hace una enumeración

exhaustiva infactible de todos los posibles subconjuntos. Se describe a continuación algunas de las aproximaciones más populares.

Algoritmo K-means El objetivo del algoritmo k-means (el cual también responde a los nombres de c-means o relocalización iterativa o ISODATA básico) es la partición de los datos en k grupos de tal manera que la suma de los cuadrados dentro de cada grupo, es minimizado. La forma más simple del algoritmo k-means se basa en alternar dos procedimientos. El primero es el de asignar los objetos a los grupos. Un objeto es usualmente asignado al grupo de cuya media es la más cercana en el sentido de la distancia Euclidiana. El segundo procedimiento es el cálculo de las medias de los grupos basado en las asignaciones. El movimiento termina cuando no hay movimiento de un objeto a otro grupo que reduzca la suma de los cuadrados en los grupos. Un ejemplo puede ilustrar mejor este método y se presenta a continuación: Consideremos los datos de entrada de 2-dimensiones mostrados en la Figura 3.3. Sea $k = 2$ y seleccionamos dos vectores del conjunto de datos como vectores de media de clusters iniciales; sean estos los puntos 5 y 6. Ahora nos movemos a través del conjunto de datos y colocamos los datos en los grupos A y B cuyos valores de media están representados por los vectores iniciales 5 y 6 respectivamente. Los datos $1, 2, 3, 4$, y 5 son asigandos a A y el dato 6 a B. Nuevas medias son calculadas y la suma de los cuadrados dentro de cada grupo es evaluada dando un total de 6.4.

Los resultados de esta iteración son resumidos en la Tabla 3.1. El proceso se vuelve a repetir usando los nuevos vectores de media como vectores de referencia. Esta vez, los objetos $1, 2, 3$, y 4 son asignados al grupo A y 5 y 6 en el grupo B. La suma de los cuadrados dentro de cada grupo decreció a un total de 4.0. Un tercer proceso de iteración no produce cambio en la suma de los cuadrados.

El algoritmo K-means[42] puede ser visto como un proceso de optimización que tiene

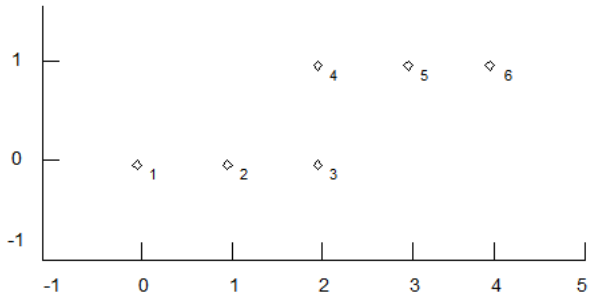


Figura 3.3: Datos para ilustrar el procedimiento del k-means

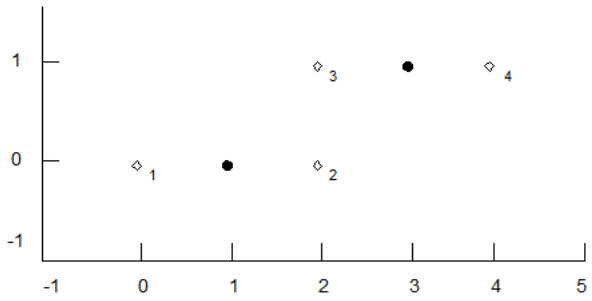


Figura 3.4: Datos para ilustrar el óptimo local del k-means

Iteración	Grupo A		Grupo B		$\mathrm{Tr}(S_w)$
	Miembros	Media	Miembros	Media	
1	1,2,3,4,5	(1.6,0.4)	6	(4.0,1.0)	6.4
2	1,2,3,4	(1.25,0.25)	5,6	(3.5,1.0)	4.0
3	1,2,3,4	(1.25,0.25)	5,6	(3.5,1.0)	4.0

Tabla 3.1: Iteraciones del k-means

Algoritmo: Clustering k-means

Entrada: Un conjunto de N vectores de datos $X = \{x_1,\ldots,x_N\}$ en R^d y el número de clusters es K.

Salida: Una partición de vectores de datos dada por el vector identidad del cluster $Y = \{y_1,\ldots,y_N\}$, $y_n \in \{1,\ldots K\}$.

Pasos:

1. *Inicialización*: Inicializa los vectores centroides del cluster $\{\mu_1,\ldots,\mu_k\}$;
2. *Asignación de datos*: Para cada vector de datos x_n, el conjunto $y_n = \underset{k}{argmin}\|x_n-\mu_k\|^2$;
3. *Estimación del centroide*: Para un cluster K, sea $X_k=\{x_n|y_n=K\}$, el centroide se estima como $\mu_k = 1/|x_k| \sum_{x\in X_k} x$;
4. Parar si Y no cambia, de otra forma regresar al paso 2.

Figura 3.5: Algoritmo de clustering K-means

como propósito minimizar la siguiente función objetivo (error de suma de cuadrados)

$$MSE = \frac{1}{N}\sum_{n} \|x_n - \mu_{y_n}\|^2, \tag{3.1}$$

donde $y_n = arg\ min_k\|x_n - \mu_k\|^2$ es el cluster identidad del vector de datos x_n, μ_{y_n} es el centroide del cluster y_n y N es el número de datos. A menos que se especifique otra cosa, $\|\cdot\|$ representa la norma L_2. Esto es, $\|x\| = \sqrt{\sum_{i=1}^{N} x(i)^2}$, donde $x(i) \in \Re^d$. El algoritmo se encuentra en la Fig. 3.5.

El procedimiento iterativo de colocar objetos a grupos en la media del grupo más cercano, seguido por el recálculo de la media de los grupos, da la versión del k-means llamado HMEANS por Spath (1980). También es llamado el método de Forgy o el método básico ISODATA.

Las técnicas descritas en esta sección minimizan un error cuadrado de la función

objetivo. El procedimiento *k-means* es un caso especial de todas las técnicas. Es ampliamente usado en reconocimiento de patrones, formando la base para muchas técnicas de clasificación supervisada. Este produce un código de los datos en el que cada patrón pertenece sólo a un cluster.

Fuzzy k-means es un algoritmo que permite a patrones pertenecer a más de un cluster, controlado por una función de membresía. Los clusters resultantes de una aproximación fuzzy k-means son suavemente traslapados, en general, con el grado de traslape controlado por un parámetro especificado por el usuario. El procedimiento de aprendizaje determina la partición.

Los algoritmos de clustering basados en el centroide son frecuentemente más eficientes que los algoritmos de clustering basados en similitudes [44]. El algoritmo de clustering k-means [42] y el EM (Expectation Maximization)[4] [8] pertenecen a ésta categoría y es por eso que elegimos el primer algoritmo (k-means) para realizar éste trabajo. Usualmente los algoritmos basados en similitudes tienen una complejidad computacional de al menos $O(N^2)$, donde N es el número de instancias de datos. Por el contrario, los algoritmos basados en el centroide son más manejables, con una complejidad computacional de $O(NKM)$ [45], donde K es el número de clusters y M el número de iteraciones. La razón de usar k-means es debido a su popularidad y su eficiencia.

3.3. Aplicaciones

En las aplicaciones de análisis de clusters de métodos jerárquicos se incluyen las siguientes:

- Monitoreo de vuelos. Eddy [14] considera un clustering de enlace sencillo de conjuntos de datos muy extenso (más de 400000 observaciones) de dimensiones muy grandes relacionados con vuelos de avionetas sobre los Estados Unidos.

- Datos clínicos. DÁndrea [3] aplica el método del centroide más cercano a datos relacionados con niños de padres alcohólicos.

- En un estudio comparativo de siete métodos de análisis de cluster jerárquicos en 20 conjuntos de datos, Morgan and Ray [31] examinaron la extensión de inversiones en dendogramas. Ellos concluyeron que las inversiones están esperando ser encontradas y recomendadas para no sólo usar los métodos de la mediana y los centroides.

El algoritmo de clustering k-means es ampliamente usado como un preprocesador para clasificación supervisada, con el fin de reducir el número de prototipos:

- Petrografía del carbón. En un estudio para clasificar los diferentes constituyentes del carbón (Mukherjee [32]), el algoritmo k-means fue aplicado para el entrenamiento de imágenes (vectores de entrenamiento consistentes de valores de nivel RGB) para determinar cuatro tipos de clusters conocidos (vitrinite, inertinite, exinite y background). Esos clusters están etiquetados y las imágenes de prueba son clasificadas usando los vectores de entrenamiento etiquetados.

- Clasificación de corte. Conway [9] usa el algoritmo k-means para segmentar imágenes de radar como parte de un estudio dentro de la clasificación de corte. k-means es usado para identificar los conjuntos de regiones de imagen que compartan atributos similares antes de etiquetarlos. Los datos fueron extraídos de un campo de cinco tipos de corte conocidos y pueden ser claramente separados dentro de dos clusters.

Algunas aplicaciones del algoritmo fuzzy k-means son las siguientes:

- Diagnósis médico. Li [25] usó un algoritmo fuzzy k-means para segmentar imágenes

en un estudio en clasificación automática y tejido etiquetado de imágenes de resonancia magnética del cerebro humano en dos dimensiones.

- Control de calidad acústico. Meier [29] describe la aplicación de fuzzy k-means a clusters de vectores de 6-dimensiones como parte de un sistema de control de calidad para azulejos cerámicos. Las señales son derivadas por golpear los azulejos y digitalizando y filtrando las señales grabadas. Las clases resultantes son representados como buenos o malos azulejos.

- Calidad del agua. Mukherjee [33] comparó el fuzzy k-means con dos algoritmos alternativos para segmentación de imágenes en un estudio para identificar y contar las bacterias de imágenes.

Vea también el estudio en fuzzy clustering por Yang [41] para más referencias de aplicaciones de fuzzy k-means.

Algoritmo Bayesiano Un ejemplo del algoritmo Bayesiano a una modelación de mezcla es digno de mencionarse. Dellaportas [13] considera la aplicación del modelo de mezcla a la clasificación para herramientas de piedras neolíticas. Una metodología Bayesiana es adoptada y desarrollada en tres principales formas de aplicar a los datos (147 medidas en 4 variables) consistente de variables de tipo mezclado -continuas y categóricas. Los valores perdidos y las medidas de error (errores en variables) en las variables continuas, también son tratadas.

Algoritmo Auto-Organizado Existen muchos ejemplos de aplicaciones del algoritmo de plano auto-organizado:

- Aplicaciones en ingeniería. Kohonen [23] da un resumen de éste algoritmo y describe muchas aplicaciones en ingeniería, incluyendo detección de fallas, análisis

de procesos y monitoreo, visión por computadora, reconocimiento de discursos, control robótico y el área de telecomunicaciones.

- Análisis de proteína humana. Ferrán [16] usa éste algoritmo para agrupar secuencias de proteínas dentro de familias. Usando 1758 secuencias de proteínas humanas, agrupó usando el plano en dos-dimensiones de varios tamaños y etiquetó los nodos en la red usando proteínas pertenecientes a secuencias conocidas.

- Clasificación de tarjetas de radar. Stewart [37] desarrolló el algoritmo de plano auto-organizado y el algoritmo de aprendizaje del vector de cuantificación a clasificación de tarjetas de radar usando un plato giratorio de datos de cuatro tipos de tarjeta. Los datos consisten de vectores de 33-dimensiones y fueron usados 36000 patrones por tarjeta. Es reportado el desempeño como una función del número de los centros del cluster, con el desempeño del vector de aprendizaje de cuantificación mejor que el algoritmo del vecino más cercano.

- Clasificación de la huella digital. Halici y Ongun [19], en un estudio de clasificación automática de huellas digitales; usó este algoritmo y uno modificado para el preprocesamiento de los vectores para combinarlos con vectores "certeros" que den no veracidades en las imágenes de las huellas digitales. Los resultados muestran un mejoramiento en estudios previos usando un perceptrón en una base de datos de tamaño 2000.

Algoritmo de Suma de Cuadrados Los métodos de la suma de cuadrados han sido aplicados a desorden de lenguajes. Powell [34] usó un algoritmo de mezcla normal y el método de la suma de cuadrados en un estudio de 86 casos que se refieren a una terapia de discurso. Cuatro grupos fueron encontrados los cuales son etiquetados como severos, altamente moderados, bajo moderados y ligeros.

Algoritmo de Vector de Cuantificación El vector de cuantificación ha sido ampliamente aplicado como un preprocesador en muchos estudios:

- Reconocimiento de discursos. Zhang [43] calcula tres diferentes vectores de cuantificación como preprocesadores para modelos de Markov en un pequeño problema de discurso. Ellos encontraron que el modelo de mezcla normal dió el mejor desempeño del clasificador subsecuente. Ver también Bergh [5].

- Diagnósticos médicos. Cosman [12] calcula la calidad de las imágenes del vector de cuatificación del árbol estructurado por el desempeño del diagnóstico de radiologística en un estudio en identificación de tumores. Resultados iniciales sugirieron que 12 bits por pixel (bpp) calcular tomografías de imágenes que pueden ser comprimidas entre 1 bpp y 2 bpp con ningún cambio significativo en la presición del diagnóstico.

- Reconocimiento de oradores. Recientes avances en reconocimiento de oradores son resumidos por Furui [17]. Los métodos de cuantificación del vector son usados para comprimir los datos de entrenamiento y produjo códigos de libros de vectores representativos caracterizando las especificaciones del orador. Un código de libro es generado por cada orador por medio de agrupar los vectores de entrenamiento. En el estado de recnocimiento del orador, una entrada absoluta es cuatificada usando el código de libro de cada orador y reconocimiento desarrollado asignando lo absoluto al orador del cual el código de libro produce menor distorsión. Un tutorial de la cuatificación del vector para discursos es dada por Makhoul [26].

3.4. Conclusiones del capítulo

En este capítulo hemos cubierto un amplio rango de técnicas para la partición de un conjunto de datos. Esto ha incluído algoritmos basados en métodos de análisis de cluster y métodos de cuantificación de vectores. Aunque ambos algoritmos tienen mucho en común -ambos producen una disección de un conjunto de datos- existen algunas diferencias. En análisis de clusters, intentamos bucar grupos "naturales" en los datos que pueden ser etiquetados en los términos del objeto principal del dato. Por el contrario, los métodos de cuantificación de vectores son desarrollados para optimizar algún criterio apropiado de la teoría de comunicación. Un área que hemos discutido en este capítulo es que los métodos de optimización con implementaciones específicas están en terminos del algoritmo k-means para el análisis de cluster.

Hasta que el análisis de cluster o clasificación sea llevado a cabo, no existe una sola técnica que sea mejor. Diferentes métodos de clustering pueden llevar a diferentes resultados y algunos métodos fallarán al detectar clusters. La razón de esto es que cada método implícitamente forza una estructura en los datos dados. Por ejemplo, los métodos de la suma de cuadrados producen clusters hiperesféricos. También, el hecho de que hay un amplio rango de métodos disponibles en parte se deriva de la basta definición de la palabra "cluster". No hay un acuerdo universal acerca de los que constituye un cluster, por ese motivo una sola definición es insuficiente.

Una dificultad adicional en el análisis de clusters es definir el número de ellos. Este problema es en parte debido a la dificultad de decidir que es un cluster y en parte porque los algoritmos de clustering tienden a producirlos aún cuando son aplicados a datos aleatorios.

Las dificultades mencionadas se pueden superar en algún grado considerando diferentes clasificaciones posibles o comparando clasificaciones de cada mitad de conjunto

de datos (por ejemplo McIntyre and Bashfield [28]; Breckenridge [7]). La interpretación de éstos es más importante que una inferencia rígida del número de grupos. Pero la pregunta es: ¿cuál método debemos emplear?. Existen algunas ventajas y desventajas de los algoritmos que hemos mencionado. Los métodos de optimización tienden a requerir una gran cantidad de tiempo computacional (y constantemente pueden resultar infactibles para conjuntos de datos muy grandes). De los métodos jerárquicos el de enlace sencillo es preferido por muchos usuarios. Es también invariante sobre transformaciones monótonas de las medidas de disimilitud. El método Ward también es muy popular. Los métodos del centroide y mediana deben ser evitados si las inversiones pueden hacer que la clasificación resultante sea muy dificil de interpretar.

Hay muchos aspectos del análisis de clusters que hemos mencionado solo brevemente en este capítulo, por lo que referimos al lector a la literatura de análisis de clusters [15] para detalles adicionales.

3.5. Recomendaciones

Existen un gran número de técnicas para elegir y con la disponibilidad de los paquetes computacionales pueden ser accesiblemente desarrollados. Sin embargo, hay algunas pautas a seguir cuando estamos llevando a cabo la clasificación.

1. Detectar y quitar los valores atípicos. Muchas técnicas de clustering son sensibles a la presencia de valores atípicos (aquéllos que se salen del rango). Además, algunas de las técnicas discutidas deben ser usadas para detectarlos y posiblemente quitarlos.

2. Dibuja los datos en dos dimensiones si es posible, en orden a entender la estructura de los datos.

3. Realiza cualquier preprocesamiento de los datos. Esto puede inlcuir una reducción del número de variables o la estandarización de las variables a media cero o varianza unitaria.

4. Si los datos no están en la forma de una matriz de disimilitudes, entonces una medida de disimilitud debe ser elegida (para algunas técnicas) y una matriz de disimilitud formada.

5. Elige una técnica apropiada. Las técnicas de optimización son más caras computacionalmente. De los métodos jerárquicos, algunos estudios favorecen el uso del método de enlace promedio, pero el método de enlace sencillo da soluciones que son invariantes a una medida de transformación monótona.

6. Evalúa el método. Tienes que calcular los resultados del método de clustering que has empleado. ¿Cómo difieren los clusters? Recomendamos que se parta el conjunto de datos en dos y se compare los resultados de cada clasificación en cada subconjunto. Los resultados similares sugeriran que una estructura útil ha sido encontrada. También, usa varios métodos y compara resultados. Para muchos de éstos métodos algunos parámetros deben ser especificados y esto vale la pena llevar a cabo la clasificación sobre un rango de valores de los parámetros para calcular la estabilidad.

7. Si tienes un problema de clasificación supervisada puedes usar la cuantificación del vector como un preproceso; un modelo para cada clase por separado es mejor que todo el conjunto de datos, y etiqueta los resultados.

8. Si requieres algunos prototipos representativos, se recomienda usar el algoritmo k-means.

Capítulo 4

Prueba del método

4.1. Introducción

En este capítulo se presentan los resultados de las pruebas a las que se sometió el algoritmo propuesto, así como las simulaciones de falla que se hicieron usando las componentes eléctricas de corriente generadas para el sistema de prueba. Los casos de fallas simulados incluyeron fallas trifásicas (las más severas), bifásicas y las monofásicas (las más comunes).

Las pruebas realizadas se hicieron con la finalidad de demostrar que el algoritmo puede detectar fallas en líneas de transmisión en los sistemas eléctricos de potencia; los resultados muestran el buen desempeño del algoritmo propuesto.

4.2. Simulador de Sistemas Eléctricos de Potencia

4.2.1. Historia del simulador

PSCAD es un simulador rápido, preciso y fácil de usar, para el diseño y modificación de todo tipo de mecanismos de potencia. Es un programa de simulación (Fig 4.1) del dominio de tiempo para sistemas de potencia de multifase y redes de control [27].

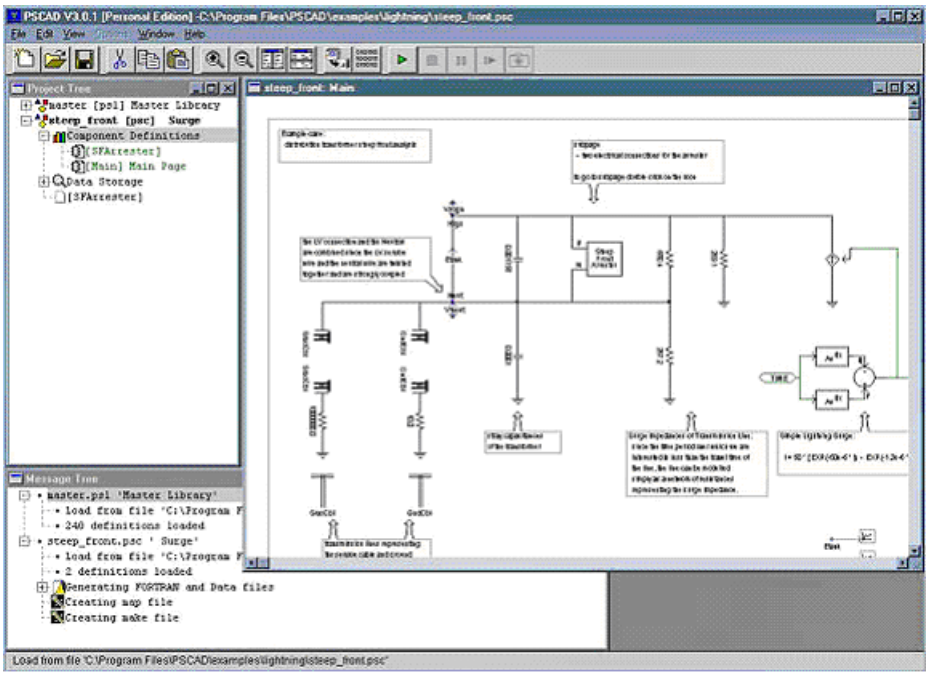


Figura 4.1: Simulador PSCAD

Está principalmente dedicado al estudio de transientes de los sistemas de potencia. Una interface gráfica y númerosas herramientas de control hacen de PSCAD una herramienta conveniente e interactiva para el análisis y diseño de cualquier sistema de potencia. PSCAD integra en un ambiente visual todos los aspectos para llevar a cabo una simulación, incluyendo ensamble del circuito, control de tiempo, análisis y reporte.

Este simulador te permite diseñar circuitos complejos en una fracción de tiempo y gracias a su avanzada tecnología puedes analizar sistemáticamente circuitos que de otra forma sería muy complicado. PSCAD caracteriza a un amplio rango de modelos para sistemas de potencia y estudios electrónicos de potencia tales como:

- Líneas de trasmisión dependientes de frecuencia
- Transformadores
- Varios mecanismos (sincrónicos, asincrónicos, DC)

- Varias turbinas (de agua, vapor, viento)
- Convertidores
- Bloques de control
- Relevadores
- etc.

Con el fin de simular las fallas en líneas de transmisión se utilizó el paquete computacional PSCAD (Power Systems Computer Aided Design). Este paquete proporciona la flexibilidad de diseñar el sistema de acuerdo a las necesidades preestablecidas de estudio y el único parámetro que se le proporciona al simulador es la resistencia de falla. El sistema es utilizado para la obtención de señales eléctricas (voltaje y corriente) para los casos de fallas monofásicas, bifásicas y trifásicas (10 tipos diferentes de fallas) en las líneas de transmisión [27].

4.3. Generación de datos de entrenamiento

Como se mencionó anteriormente, el único parámetro con que se alimenta al simulador es la resistencia de falla cuyos valores utilizados son entre 0 y 900 (con datos incrementales de 50 ohms) para cada tipo de falla específico (de 1 a 10). Se fija la resistencia de falla para efectuar toda una corrida donde se producen 160 archivos correspondientes a los 10 tipos de falla y a los 16 tiempos de inicio de falla (valores entre 0 y 15). Cada uno de estos archivos representa una respuesta del sistema para un tipo de falla, tiempo de falla y resistencia de falla determinadas. La Figura 4.2 muestra una pantalla de como el simulador está generando los datos: las ondas representan la señal de las fases y las agujas nos dicen el tiempo y el tipo de falla. El simulador nos

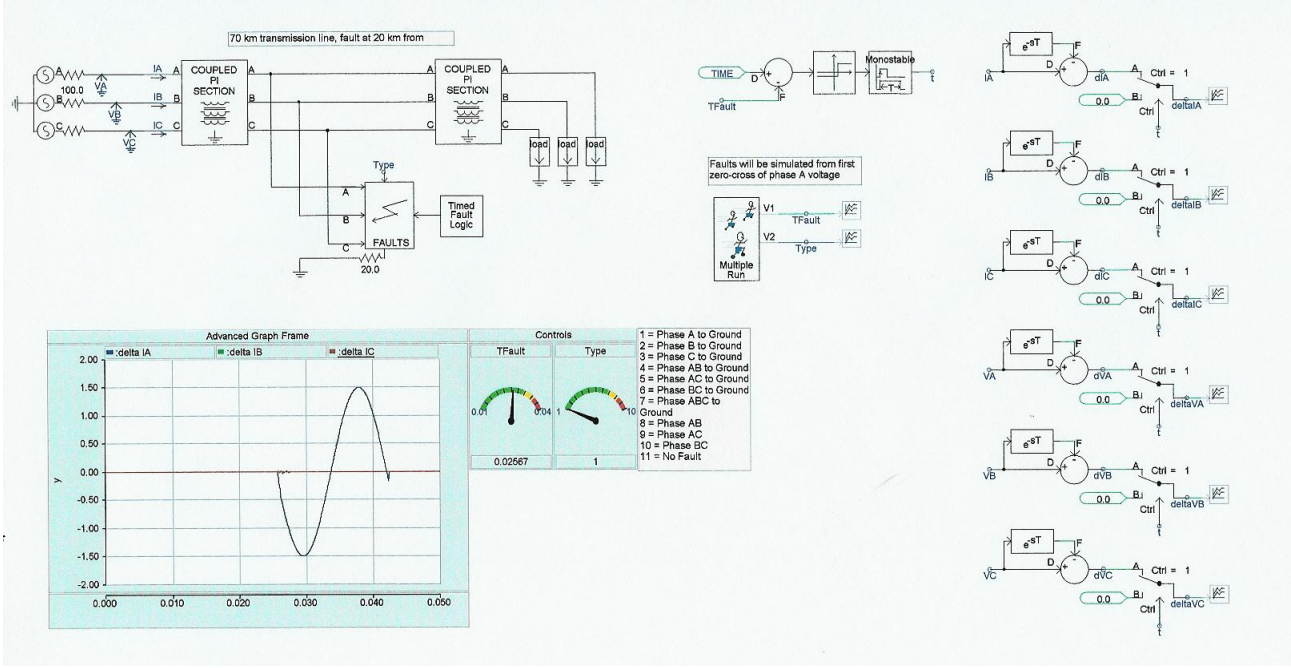


Figura 4.2: Generación de los datos en el simulador PSCAD

da las señales incrementales (SI) donde $SI(t) = SI(t) - SI(t-1)$ (esto es, la señal del período actual menos la señal del período anterior), por tal motivo de los 3 ciclos que reporta el simulador sólo un ciclo es significativo; dos de los tres ciclos resultan ser cero.

Los datos de señales incrementales que se obtienen del simulador para cada una de las 10 fallas (3 monofásicas, 3 bifásicas, 1 trifásica y 3 bifásicas a tierra), con una resistencia de falla fija, están contenidos en una matriz de 1440×7. Los datos proporcionados por el simulador se presentan en la Figura 4.3 (donde se ve una pequeña muestra de los datos); de las 7 columnas la primera indica un tiempo de integración en el simulador y las 6 restantes indican 3 voltajes de fase VC, VB, VA en ese orden y tres corrientes de fase IC, IB, IA en el mismo orden.

Las señales se obtienen con una frecuencia de 480 datos incrementales por ciclo y el simulador reporta 3 ciclos siendo en total 1441 (ya que el primer renglón es de ceros) datos; de ese total de datos solo 480 líneas son significativas, es decir solo un ciclo (el

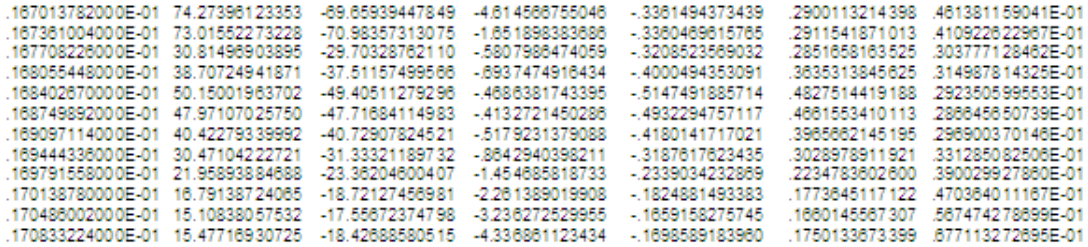


Figura 4.3: Muestra de archivo de datos

segundo) es el que me da el comportamiento de la falla y los 960 restantes son ceros; los datos incrementales toman un valor de cero un ciclo antes y un ciclo después de la falla.

Para cada falla existen 16 matrices de datos de este tipo, indicando 16 casos diferentes de inicio de falla. Por consiguiente, para cada resistencia de falla se dispondrá de 160 matrices de datos: 16 matrices/falla para 10 tipos de fallas. Los datos se reportan en 7 columnas de las cuales las columnas 2, 3, 4 corresponden a los voltajes de las fases C, B y A respectivamente; a su vez, los valores de las corrientes en las fases C, B y A se localizan en las columnas 5, 6, 7.

4.3.1. Casos de falla

Los casos de falla elegidos representan las fallas más comunes a las que están expuestas las líneas de transmisión. Los datos se obtienen inicialmente del simulador el cual proporciona para cada resistencia de falla los datos de voltaje y corriente para las tres fases, para cada inicio de falla y para los 10 tipos de falla. Los 10 tipos de falla en las líneas de transmisión son:

1. Monofásica fase A

2. Monofásica fase B

3. Monofásica fase C
4. Bifásica fases AB
5. Bifásica fases AC
6. Bifásica fases BC
7. Trifásica fases ABC
8. Bifásica (a tierra) fases AB
9. Bifásica (a tierra) fases AC
10. Bifásica (a tierra) fases BC

A partir de la falla se reportan 480 datos (1 ciclo). Se reportan 16 inicios diferentes de falla de un ciclo; este segmento contiene 16 matrices de 480×6.

4.3.2. Especificación del dominio de entradas

Las entradas no son más que las componentes de las *n-adas* que representan el objeto que en este caso es el sistema entero en un determinado tipo de falla. Elegimos trabajar con un subconjunto de los datos consistente en: tomar una cada treinta lecturas (16 de las 480) y sólo las 3 corrientes es decir, de la matriz de 480×6, queda un vector de $1 \times 48(16 \times 3)$ que representa un tipo y un tiempo de falla determinado. En total la reducción nos lleva a una matriz de 16×48 (por cada tipo de falla) datos que hemos ordenado en una *n-ada* de la siguiente manera: en la posición 1 a la 16, las dieciséis lecturas representan los tiempos de falla (0 al 15) y los 48 representan las tres corrientes de cada uno de los 16 datos.

Programa para determinar la solución al problema dual del problema de mayor margen de separación entre clases

Datos de entrada:

X1 : Patrones de la clase I (datos de un tipo de falla)
X2 : Patrones de la clase II (datos del otro tipo de falla)
X : Conjunto de patrones de entrenamiento
Y : Etiqueta asignada a los patrones de entrenamiento
+1 Patrones de la clase I
- 1 Patrones de la clase II

Datos de salida:

alpha : Vector de variables del dual
bias : Bias del clasificador lineal
nsv : Number of Support Vector

```
X=[X1;X2];
Y=[ones(size(X1,1),1);-ones(size(X2,1),1)];

% Función SVC
[nsv alpha bias] = svc(X,Y);
```

Figura 4.4: Programa SVC

4.4. Generación del clasificador

Una vez que se dispone de los patrones de entrenamiento (matriz de 16×48 para cada tipo de falla), el clasificador se genera mediante las ecuaciones 2.7 y 2.12:

$$w = \sum_{i=1}^{M} y_i \alpha_i x_i$$

$$b* = -\frac{1}{2} \sum_{k=1}^{M} y_k \alpha_k^* (s_1^T x_k + s_2^T x_k)$$

Para obtener los valores de α se procede a aplicar el algoritmo que me genera el clasificador de mayor margen de separación entre clases (fallas). Para que el algoritmo funcione se necesitan proveer los datos (ver Fig. 4.4) y por medio de este programa se obtienen los valores que estamos buscando. Los valores de *alpha* de la Figura 4.4 representan la solución del problema dual de mayor margen de separación entre clases (ver ecuación 2.11).

La herramienta de SVM (Support Vector Machine) fue diseñada para objetivos

de enseñanza por Steve Gunn (sgr@ecs.soton.ac.uk, University of Southampton), se diseñó en Matlab porque este es particularmente bueno en cuanto a la legibilidad de su codificación y simple para modificar. Esta herramienta está disponible solo para propósitos académicos.

Capítulo 5

Resultados Computacionales

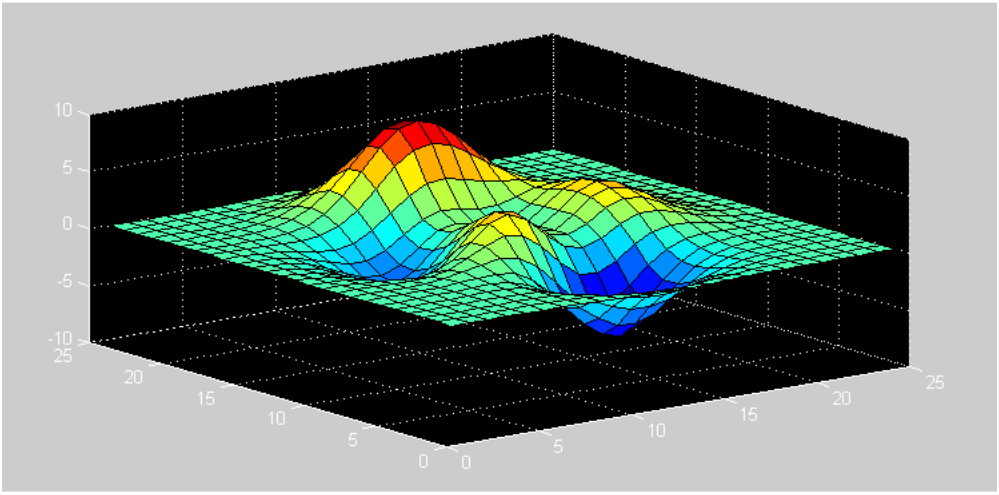
5.1. Introducción

En este capítulo se mostrarán los diferentes resultados que se obtuvieron al aplicar los procedimientos propuestos en el capítulo 4 al conjunto de patrones de entrenamiento generados. Para la programación de las rutinas propuestas se utilizó el lenguaje matemático MATLAB 6.5 y el optimizador GAMS y fueron ejecutadas en una computadora SUN^{TM} Ultra 10, con sistema operativo $Solaris^{TM}$ versión 7. Para la edición del texto se utilizó un editor llamado TexnicCenter para el desarrollo de documentos en Latex.

5.1.1. Generalidades del MATLAB

MATLAB integra computación, visualización y programación, en un ambiente fácil de usar donde los problemas y soluciones son expresadas en una notación matemática familiar. Los usos más típicos incluyen:

- Matemáticas y computación
- Desarrollo de algoritmos



- Adquisición de datos
- Simulación, modelación
- Análisis de datos, exploración y visualización
- Gráficos científicos y de ingeniería

MATLAB es un sistema interactivo cuyo cuyo elemento de dato básico es un arreglo que no requiere dimensionamiento. Esto permite resolver muchos problemas técnicos, especialmente aquéllos con formulaciones de matrices y vectores.

El nombre de MATLAB significa laboratorio de matrices. MATLAB ha evolucionado considerablemente en los últimos años. En las universidades, es la herramienta estándar para cursos introductorios y avanzados en matemáticas, ingeniería, y ciencia. En la industria, MATLAB es la herramienta de elección para investigación de alta productividad, desarrollo y análisis.

Toolbox es una colección de funciones de MATLAB (M-files) que ayudan para resolver clases de problemas particulares. Áreas en las cuales las toolboxes están disponibles incluyen procesamiento de señales, sistemas de control, redes neuronales, lógica difusa,

simulación, análisis financiero, estadística y muchos otros.

5.1.2. Generalidades del GAMS

The General Algebraic Modeling System (GAMS) está especificamente diseñado para modelar problemas de optimización lineal, no lineal y entero mixtos. El sistema modela problemas en una manera compacta y natural. El usuario puede cambiar la formulación fácil y rápidamente, cambiar de un solver a otro y también puede convertir problemas lineales a no lineales solo con cambiar una pequeña instrucción. El sistema es especialmente útil para problemas grandes y complejos. GAMS esta disponoble para uso de computadoras personales, estaciones de trabajo y super computadoras.

GAMS permite al usuario concentrarse en la modelación del problema haciendo su configuración simple. El sistema cuida los detalles del tiempo de la máquina y la implementación del sistema.

5.1.3. Generalidades del TexnicCenter

TeXnicCenter es un editor de texto para Microsoft Windows que realiza documentos en Latex. TeXnicCenter incluye muchas de las herramientas necesarias para el desarrollo de documentos con Latex. Se utiliza el editor de texto para escribir archivos en Latex y éste reconoce instrucciones de Latex y los despliega en diferentes colores para hacer más entendible la estructura de tus documentos.

El objetivo principal de TeXnicCenter es apoyar a los usuarios de Latex proveyendo las más importantes instrucciones vía menú de herramientas, por lo cual no es necesario recordar todos los detalles de sintaxis para construir el código. Además la interface al compilador de Latex y otras herramientas es proporcionada automáticamente.

5.2. Experimentación y Análisis

5.2.1. Experimentación con datos aleatorios

Para la generación de datos se realizan los siguientes pasos:

1. Se utiliza la función *dataelipse* Fig.5.1 que genera datos aleatorios dentro de una elipse, los parámetros de entrada que se le proporcionan a la función son los siguientes:

   - v: Centro
   - se: Longitud de los semiejes
   - teta: rotación de ejes en grados
   - N: número de muestras

   la función se manda llamar de la siguiente manera: $[X] = dataelipse(v, se, teta, N)$

2. A los datos de entrenamiento generados se les aplica el algoritmo de mayor margen Fig. 4.4 donde los parámetros de entrada son los siguientes:

   - ClaseI: Datos que pertenecen a la clase $+1$
   - ClaseII: Datos que pertenecen a la clase -1
   - X: Datos de entrenamiento que se generan asi: $X = [ClaseI; ClaseII]$;
   - Y: Etiqueta de las clases, este valor se genera así:

     $Y = [ones(size(ClaseI), 1); -ones(size(ClaseII), 1)]$

   después de haber generado los datos de entrenamiento se manda llamar la función $[nsv, alpha, b0] = svc(X, Y)$

Experimentación 1

Las primeras pruebas que se hicieron fueron para ver el comportamiento computacional de los datos en cuanto al tiempo. Se generaron datos aleatorios para tener los patrones de entrenamiento. En este caso se realizó el caso más simple, es decir en 2 dimensiones. Para la generación de estos datos se utilizó la función *dataelipse* Figura 5.1. Se inició el experimento con 2 clases de 35 datos cada una y el procedimiento se repite hasta clases de 1120 datos cada una; los resultados se reportan en la Tabla 5.1 para dos clases linealmente separables.

Datos/clase	Tiempo de Ejecución	Margen de Separación	Vectores Soporte
35	0.6 seg	13.737	2
70	1.5 seg	12.858	3
140	8.2 seg	13.249	2
280	76.4 seg	12.991	3
350	232.7 seg	12.713	2
560	1317.2 seg	12.686	2
1120	12801.2 seg	12.598	3

Tabla 5.1: Complejidad computacional con número de patrones o datos

Se puede ver claramente en la Figura 5.2 el comportamiento del tiempo de ejecución ya que aumenta considerablemente mientras aumenta el número de datos en las clases. El margen de separación y el número de vectores soporte no nos dicen mucho en este caso ya que permanecen sin cambio mientras aumentan los datos. Para resolver el problema de tiempo con la cantidad de datos, se propuso (Sección 3.2) usar "clustering" (Fig. 3.5) para simplificar el proceso computacional. También se realizaron las mismas pruebas en el modelador de GAMS usando el programa de la Figura 5.3 encontrándose resultados similares.

```
function [X]=dataelipse(v,se,teta,N)
%
%   PARAMETROS DE ENTRADA
%       v:  centro
%      se:  longitud de semiejes
%    teta:  rotación de ejes en grados
%       N:  numero de muestras
%   color:  color de los datos 'r*'
%
%   VALORES DE SALIDA
%
%       X:  Conjunto de n datos de dimensión similar
%           a 'v' y a 'se'.
%
tetr=teta*pi/180;
Trot=[cos(tetr) sin(tetr);-sin(tetr) cos(tetr)];
B=inv(Trot);
X=[];
for i=1:N
    gama=2*pi*rand(1);
    dist=rand(1);
    xo=dist*[cos(gama);sin(gama)];
    %x=v+B*(xo.*(se/1));
    x=v+B*diag(se)*xo;
    X=[X x];
end
```

Figura 5.1: Función para generar datos aleatorios

Figura 5.2: Complejidad computacional

```
PARAMETER  c(i, j)   matriz;
        c(i, j) = sum(p, x(p, i) * x(p, j));

VARIABLES
    a(I)
    z;

POSITIVE VARIABLE  a;
 EQUATIONS

   obj       funcion objetivo
   res1       restriccion;

obj..      z =E= SUM(i, a(i))-(SUM((i, j), y(i)*y(j)*a(i)*a(j)*c(i, j)))/2;
res1..     SUM((i), y(i) * a(i)) =E= 0;

MODEL prob/ALL/;

SOLVE prob USING NLP MAXIMIZING Z;

DISPLAY   a.l , z.l ;

FILE res/resultados.out/

PUT res
    put "VARIABLES" /;
    put @20, "a(i)"/;
    LOOP ((I), put @12, a.L(I): 8: 4/);
```

Figura 5.3: Algoritmo de Mayor Margen en Gams

5.2.2. Experimentación con datos del simulador PSCAD

Generación de datos

Para la generación de datos se realizan los siguientes pasos:

1. Se utiliza el simulador PSCAD (Sección 4.3) al que se le da como dato de entrada una resistencia de falla con la cual el simulador genera 160 archivos de los cuales los primeros 16 representan el tipo de falla monofásica fase A, los siguientes 16 el tipo de falla monofásica fase B y así sucesivamente hasta completar los 10 tipos de falla que estamos considerando.

2. Se analizan cada uno de los 160 archivos que generó el PSCAD; los pasos para la obtención de los datos de entrenamiento son los siguientes:

   a) Las señales se obtienen con una frecuencia de 480 datos por ciclo y el simulador reporta 3 ciclos por lo cual cada archivo tiene una matriz de datos de

1440×9, de los cuales solo 480 datos son significativos (es decir, solo un ciclo nos da el comportamiento de la falla) y de las 9 columnas, las columnas 3,4 y 5 corresponden a los voltajes de las fases C,B y A respectivamente y los valores de las corrientes se localizan en las columnas 6,7 y 9. Cuando ocurre una falla los voltajes tienden a caer y la corriente se dispara, por consiguiente se tomaron en cuenta solo las corrientes como rasgos distintivos de la falla. Entonces, el primer paso es eliminar los primeros 480 renglones y los últimos 480 renglones y las columnas 1, 2, 3, 4, 5, y 8 de la matriz de datos original (1440×9) reduciéndose a una matriz de 480×3 datos de entrenamiento por cada resistencia de falla para cada tipo de falla.

b) Se hace un diezmado cada 30 datos de la matriz de 480×3 quedando una matriz de 16×3 datos.

c) Se acomodan los datos de 16×3 en un renglón de 1×48 de la siguiente manera: los datos del primer renglón se quedan igual; los datos del segundo renglón se pasan al primer renglón pero a partir de la cuarta columna y así sucesivamente hasta terminar y formar una línea de 1×48.

d) Cada línea de 1×48 representa un tiempo de inicio de falla específico y como se consideran 16 tiempos de inicio de falla, se forma una matriz de 16×48 datos que representan un tipo de falla especifico.

e) Se estan considerando 18 resistencias de falla $(50, 100, 150, ..., 850, 900)$; si juntamos todas las resistencias de falla se forma una matriz de 288×48 (18*16=288) datos para cada tipo de falla.

3. A los datos de entrenamiento (i.e. Clasificación de datos de fallas en fase A contra datos de fallas de fase B) generados se les aplica el algoritmo de mayor margen Fig.4.4 donde los parámetros de entrada son los siguientes (Caso 1 para datos sin

usar clustering y Caso 2 usando clustering):

a) **Caso 1: Sin Clustering**

- ClaseI: Datos de corrientes de las tres fases que corresponden a la fase A (tipo falla 1).
- ClaseII: Datos de corrientes de las tres fases que corresponden a la fase B (tipo falla 2).
- X: Datos totales de entrenamiento que se generan asi: $X = [ClaseI; ClaseII]$
- Y: Etiquetas en donde la ClaseI tiene etiqueta de $+1$ y la ClaseII de -1; esta información se genera así:
  $Y = [ones(size(ClaseI), 1); -ones(size(ClaseII), 1)]$

b) **Caso 2: Con Clustering**

- Se aplica el algoritmo 3.5 de clustering $k-means$ donde los parámetros que se le proporcionan son los siguientes:
  - *centres*: son los centros o prototipos de los clusters, el número de clusters es especificado por el usuario ($numclusters$). Para la generación de los centros se hace lo siguiente:

    for $k = 1 : numclusters$
    $num = ceil(rand * size(Clase, 1))$
    $centres(k, :) = Clase(num, :)$
    end

  - *data*: esta matriz representa los datos que se requieren agrupar; para la ClaseI, entonces $data = ClaseI$ y para la ClaseII $data = ClaseII$.
  - *options*(14): es el máximo número de iteraciones que quieres realizar; para nuestro caso tomamos un valor de 200.

- Con los datos de entrenamiento, se manda llamar la función $k-means$ de la forma siguiente:

$$[centres, post] = kmeans(centres, data, options)$$

donde *post* es una matriz de *ceros* y *unos*, donde el número de renglones es igual al número de datos de la muestra y el número de columnas es igual al número de clusters especificado. En cada columna de esta matriz los valores de *uno* representan los datos que se encuentran en el cluster correspondiente a cada columna. La suma total de todos los *unos* debe ser igual al número total de datos para la clase correspondiente.

c) Con el proceso de clustering los datos de entrenamiento son ahora: $X = [centresI; centresII]$ donde $centresI$ y $centresII$ son los prototipos de los clusters de los datos de las $ClasesI$ y $ClasesII$ respectivamente y

$$Y = [ones(size(centresI), 1); -ones(size(centresII), 1)]$$

Procesamiento de datos

Después de haber generado los datos de entrenamiento (ya sea utilizando clustering o no) se manda llamar la función $[nsv, alpha, b0] = svc(X, Y)$ y se realizan los siguientes pasos para iniciar el procesamiento de los datos:

1. Uno de los datos de salida que arroja la función *svc* es el *bias* ($b0$). Con los datos (alpha) de la solución del dual (ecuación 2.11) se calcula el w de la siguiente manera:

$$w = (Y. * alpha)' * X$$

y con el w y $b0$ se forma el clasificador:

$$H = \{x | w^T x + b0 = 0\}$$

2. Para verificar el desempeño del clasificador diseñado, se revisan los datos de entrenamiento los cuales deben cumplir las siguientes condiciones:

$$w^T x + b0 \geq 1 \text{ para } x \in ClaseI$$

y

$$w^T x + b0 \leq -1 \text{ para } x \in ClaseII$$

después se realizan las predicciones de datos nuevos generados con resistencias de falla diferentes a los patrones de entrenamiento utilizados (en este caso se probó con resistencias de 20,30,40,90,160 y 999 ohms). La detección de respuestas erróneas del clasificador, se realiza con el procedimiento siguiente:

a) Determinar el valor de la función $f_H(x) = w^T x + b0$ para cada clase:

$$Z1 = ClaseI * w' + bias * ones(size(ClaseI, 1), 1)$$

$$Z2 = ClaseII * w' + bias * ones(size(ClaseII, 1), 1)$$

b) Determinar errores del clasificador para cada clase:

$$Sea \;\; NFZ1 = sum(Z1 < 0),$$

$$NFZ2 = sum(Z2 > 0).$$

Entonces, existen errores del clasificador en la $ClaseI$ y $ClaseII$ si $NFZ1 \neq 0$ y $NFZ2 \neq 0$ respectivamente.

c) Determinar las posiciones de los datos mal clasificados para cada clase:

$$Posf1Z1 = find(Z1 < 0)$$

$$Posf1Z2 = find(Z2 > 0)$$

d) Determinar los puntos mal clasificados para cada clase:

$$P1 = ClaseI(Posf1Z1, :)$$

$$P2 = ClaseII(Posf1Z2, :)$$

3. Si no hay datos mal clasificados el algoritmo termina; si existen malas clasificaciones los datos de entrenamiento se forman de los datos que se encuentran en los clusters que resultaron ser vectores soporte. El procedimiento es el siguiente:

   a) Se localizan los clusters que son vectores soporte ($\alpha \neq 0$).

   b) Se toman los puntos que hay dentro de cada cluster de la siguiente forma:

      1) En cada cluster se detectan las posiciones de los puntos en la matriz original:

$$\begin{aligned}
&Indx = zeros(150, numclusters);\\
&\text{for } \ j = 1 : numclusters\\
&\ \ temp = find(post(:, j) == 1);\\
&\ \ hil = size(temp, 1);\\
&\ \ \text{for } \ k = 1 : hil\\
&\ \ \ Indx(k, j) = temp(k, 1);\\
&\ \ \text{end}\\
&\text{end}
\end{aligned}$$

      2) Se obtienen los datos de los clusters de la siguiente manera (donde *sumtot* es el total de renglones de la nueva matriz que corresponde a los datos de entrenamiento; *nvecdos* son los clusters que resultaron ser vectores soporte y *datan* es la matriz de los datos que correspondian a esos clusters vectores soporte y que ahora forman mis nuevos datos de

entrenamiento):

$$
\begin{aligned}
&datan = zeros(sumtot, 48);\\
&sum = 0;\\
&\text{for } \; i = 1 : nvec\\
&\quad \text{for } \; j = 1 : suma(i)\\
&\quad\quad L = Indx(j, nvecdos(i));\\
&\quad\quad if \;\; L \neq 0\\
&\quad\quad\quad sum = sum + 1;\\
&\quad\quad datan(sum, :) = ClaseI(L, :);\\
&\quad\quad else\\
&\quad\quad \text{end}\\
&\quad \text{end}\\
&\text{end}
\end{aligned}
$$

c) Se continúa con el Paso 1 del procesamiento de datos.

Experimentación 2

Usando el procedimiento antes descrito se realizaron tres diferentes pruebas para la construcción del margen de separación entre fallas monofásicas los datos utilizados para cada fase fueron 288. Los resultados se muestran en la Tabla 5.2. No se encontró ninguna mala clasificación para los datos de entrenamiento ni para los 96 (en este caso se probó con resistencias de 20,30,40,90,160 y 999 ohms) datos de prueba.

Datos/clase	ClaseI	ClaseII	Margen	Vectores Soporte	T. Cómputo
288	Falla 1	Falla 2	0.6147	12 (2.1 %)	317.1 seg
288	Falla 1	Falla 3	0.6130	10 (1.7 %)	341.8 seg
288	Falla 2	Falla 3	0.4369	278 (48.3 %)	234.7 seg

Tabla 5.2: Características de entrenamiento del clasificador entre fallas monofásicas

Experimentación 3

Esta prueba es similar a la experimentación anterior pero ahora utilizando la metodología propuesta de clustering para analizar como se comporta el tiempo de ejecución así como la similitud entre sus gradientes $w's$. Para la generación de los datos de entrenamiento se siguieron los pasos de la sección 5.2.2.

Usando el procedimiento antes descrito se realizaron tres diferentes pruebas para la construcción del margen de separación entre fallas monofásicas aplicando clustering con $k = 10$ para cada clase (donde k es el número de clusters), es decir la ClaseI y la ClaseII tienen 10 datos cada una. Se observó que el tiempo de cómputo disminuyó considerablemente con respecto al experimento anterior y no se encontró ninguna mala clasificación para los datos de entrenamiento ni para los datos de prueba; los resultados se muestran en la Tabla 5.3.

Datos/clase	ClaseI	ClaseII	Margen	Vectores Soporte	T. Cómputo	
					Clustering	Svc
10	Falla 1	Falla 2	0.829	6 (30 %)	2.53 seg	0.6 seg
10	Falla 1	Falla 3	0.866	5 (25 %)	2.01 seg	0.0 seg
10	Falla 2	Falla 3	0.622	12 (60 %)	2.3 seg	0.1 seg

Tabla 5.3: Características de entrenamiento del clasificador entre fallas monofásicas aplicando clustering

Similitud del gradiente entre ambos Clasificadores

La proyección entre los gradientes de los dos clasificadores (con o sin clustering) nos ayuda a determinar la similitud existente entre ellos; es decir, para el experimento 2 y 3 se calcularon las proyecciones de los vectores de peso w (w_1 para el experimento sin usar clustering y w_2 para el experimento usando clustering).

La Proyección Ortogonal $p(w_1, w_2)$ de $w1$ sobre $w2$ se determina de la siguiente manera:

$$p(w1, w2) = \frac{w1 \cdot w2}{||w1||_2 \cdot ||w2||_2}$$

Con la formulación anterior se puede ver que en la medida que w_2 se aproxime a w_1, el valor de la proyección tiende a la unidad. La tabla 5.4 muestra el resultado de las proyecciones y queda como conclusión que los clasificadores que se formaron sin utilizar y utilizando clustering son muy similares entre sí.

ClaseI	ClaseII	Proyección
Falla 1	Falla 2	0.9928
Falla 1	Falla 3	0.9932
Falla 2	Falla 3	0.9958

Tabla 5.4: Proyección de Clasificadores

Experimentación 4

Para la generación de los datos de entrenamiento de esta prueba, se siguieron los pasos de la sección 5.2.2 pero en el paso 2 se cambia el inciso e), ya que en este experimento se probó para cada una de las 18 resistencias de falla por separado, con el fin de evaluar el comportamiento de los datos en la medida en que las resistencias de falla aumentan. Para estas pruebas la ClaseI son los datos que corresponden al tipo de falla 1 (fase A) y la claseII son los datos correspondientes a las 9 tipos de fallas restantes. Los resultados se muestran en la Tabla 5.5.

Se realizó el conteo de los vectores soporte para cuantificar la aportación de cada clase. En la Tabla 5.6 se presentan estos resultados y podemos observar que la aportación de la $ClaseII$ está en un rango del $63 - 76\,\%$.

Como se puede ver, los datos de la ClaseII contienen más vectores soporte que los de la claseI (Fig. 5.4). Para visualizarlo más claramente, en la Figura 5.5 se muestran los datos que resultan ser vectores soporte en un porcentaje no menor al $77\,\%$(14 veces) de

Resistencia	Margen	# vectores soporte	# datos mal clasificados	
			entrenamiento	prueba
50	2.5185	21 (13.1 %)	0	0
100	2.1176	22 (13.8 %)	0	0
150	1.7825	20 (12.5 %)	0	0
200	1.5254	19 (11.9 %)	0	0
250	1.3302	19 (11.9 %)	0	0
300	1.1783	19 (11.9 %)	0	0
350	1.0566	20 (12.5 %)	0	0
400	0.9571	20 (12.5 %)	0	0
450	0.8743	20 (12.5 %)	0	0
500	0.8045	20 (12.5 %)	0	0
550	0.7448	21 (13.1 %)	0	0
600	0.6933	20 (12.5 %)	0	0
650	0.6482	20 (12.5 %)	0	0
700	0.6086	20 (12.5 %)	0	0
750	0.5734	21 (13.1 %)	0	0
800	0.5420	20 (12.5 %)	0	0
850	0.5139	19 (11.9 %)	0	0
900	0.4884	19 (11.9 %)	0	0

Tabla 5.5: Resultados de la fase A contra las 9 fallas de fases restantes

las pruebas con las 18 resistencias de falla; estos son el tiempo de inicio de falla 1,3,4,14 y 15 (para la falla del tipo 1-Fase A); 12 y 13 (para la falla del tipo 2-Fase B); 1,2,5 y 6(para la falla del tipo 3-Fase C); y 9 y 11 (para la falla del tipo 4-Fase AB).

Experimentación 5

Para esta prueba se hizo exactamente lo mismo que la experimentación anterior solo que ahora la $ClaseI$ son los datos que corresponden a la fase B y la $ClaseII$ son los datos correspondientes a las 9 fases restantes. Los resultados se muestran en la Tabla 5.7.

Se realizó un análisis para ver el comportamiento de los vectores soporte y los resultados se reportaron en la Tabla 5.8. Para este caso observamos que también los datos de la $ClaseII$ contienen más vectores soporte que los de la $ClaseI$ (Fig. 5.6).

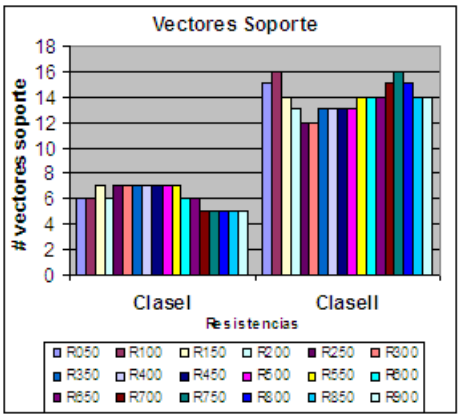


Figura 5.4: Vectores soporte del Tipo Falla 1 contra las 9 restantes

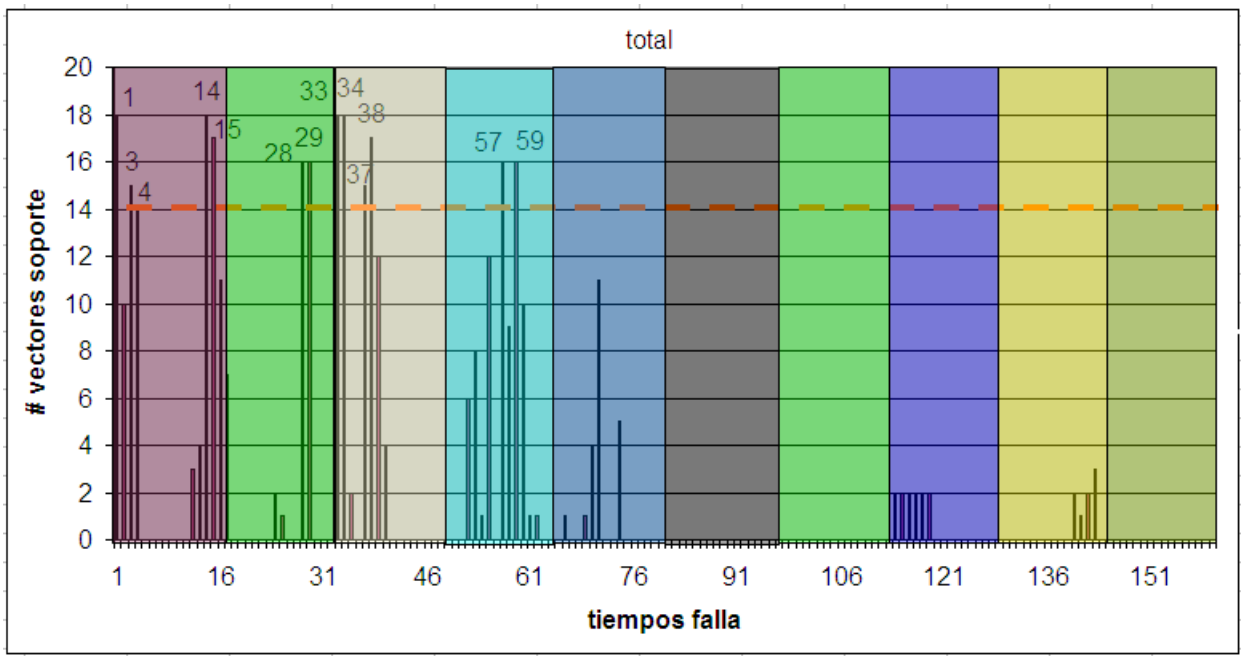


Figura 5.5: Datos que constantemente son vectores soporte

Resistencia	# vectores soporte	ClaseI 16	ClaseII 144
50	21	6	15
100	22	6	16
150	20	6	14
200	19	6	13
250	19	7	12
300	19	7	12
350	20	7	13
400	20	7	13
450	20	7	13
500	20	7	13
550	21	7	14
600	20	6	14
650	20	6	14
700	20	5	15
750	21	5	16
800	20	5	15
850	19	5	14
900	19	5	14

Tabla 5.6: Vectores soporte de la fase A contra las 9 fallas de fases restantes

En la Figura 5.7, al igual que el experimento 4, se muestran los datos que con mas frecuencia (no menor al 77 % de las 18 pruebas de resistencia de falla) aparecen como vectores soporte: esto son, el tiempo de falla 1, 11, 12 y 13 (para la falla del tipo 2-Fase B); 3 y 16 (para la falla del tipo 1-Fase A); 1,5 y 6 (para la falla del tipo 3-Fase C); 3,4,5 y 6 (para la falla del tipo 8-Fase AB a tierra).

Experimentación 6

Para esta prueba se hizo exactamente lo mismo que la experimentación 4 y 5 solo que en este caso la $ClaseI$ contiene los datos que corresponden a la Fase C y la $ClaseII$ contiene los datos correspondientes a las 9 fases restantes. Los resultados se muestran en la Tabla 5.9.

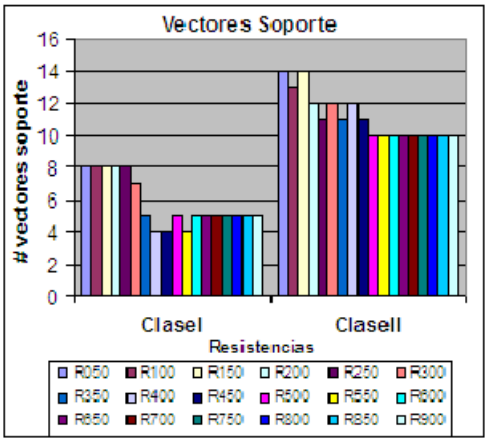


Figura 5.6: Vectores soporte del Tipo Falla 2 contra las 9 restantes

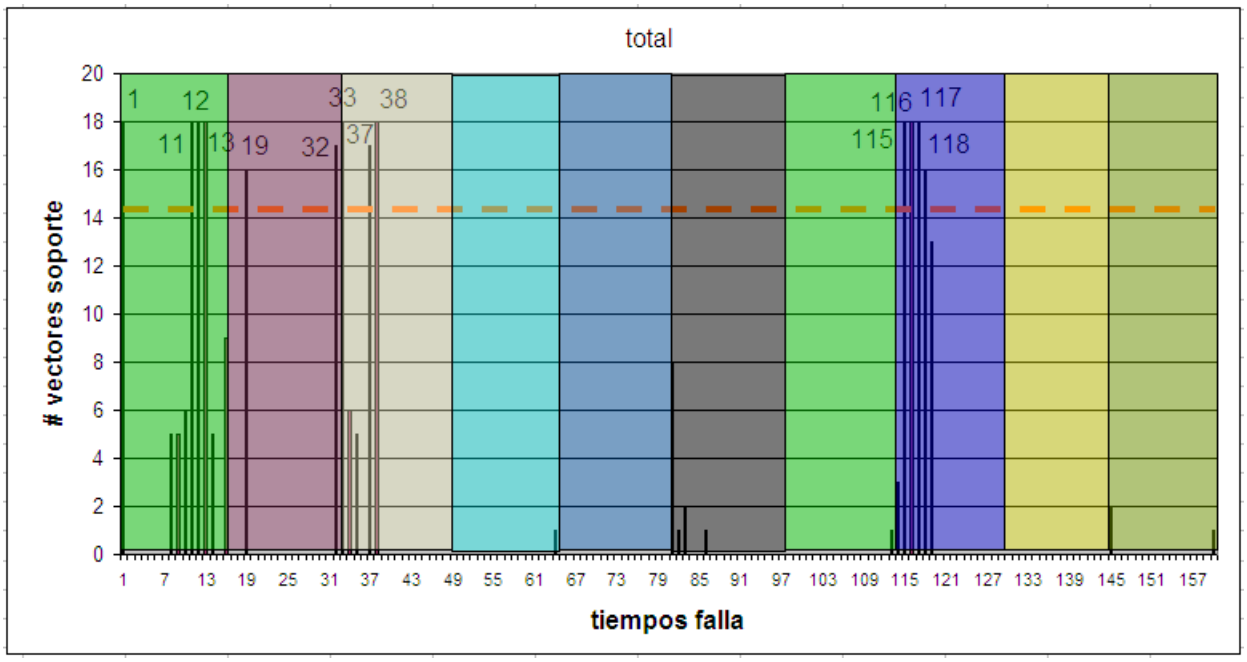


Figura 5.7: Datos que constantemente son vectores soporte

Resistencia	Margen	# vectores soporte	# datos mal clasificados	
			entrenamiento	prueba
50	2.5178	22 (13.8 %)	0	0
100	2.1136	21 (13.1 %)	0	0
150	1.7810	22 (13.8 %)	0	0
200	1.5253	20 (12.5 %)	0	0
250	1.3303	19 (11.9 %)	0	0
300	1.1774	17 (10.6 %)	0	0
350	1.0547	16 (10.0 %)	0	0
400	0.9544	16 (10.0 %)	0	0
450	0.8709	15 (9.40 %)	0	0
500	0.8006	15 (9.40 %)	0	0
550	0.7405	14 (8.80 %)	0	0
600	0.6886	15 (9.40 %)	0	0
650	0.6434	15 (9.40 %)	0	0
700	0.6036	15 (9.40 %)	0	0
750	0.5683	15 (9.40 %)	0	0
800	0.5369	15 (9.40 %)	0	0
850	0.5087	15 (9.40 %)	0	0
900	0.4833	15 (9.40 %)	0	0

Tabla 5.7: Resultados de la fase B contra las 9 fallas de fases restantes

Se realizó un análisis para ver el comportamiento de los vectores soporte y los resultados se reportaron en la Tabla 5.10. Se observa que, como en las 2 pruebas anteriores, los datos de la $ClaseII$ contienen más vectores soporte que los de la $ClaseI$ (Fig. 5.8). En la Figura 5.9 se muestran los datos que para la mayoría de las 18 resistencias de falla analizadas (mayor del 77 %) resultaron ser vectores soporte; esto son, el tiempo de falla 1,2,5, 6, 7 y 8 (para la falla del tipo 3-Fase C); 16(para la falla del tipo 1-Fase A); 1,12 y 13(para la falla del tipo 2-Fase B); 13,14 y 16 (para la falla del tipo 5-Fase AC).

Para los experimentos 4, 5 y 6 se observa que: 1) Mientras la resistencia de falla aumenta, el margen de separación disminuye; 2) El número de vectores soporte es considerado aceptable (menos del 15 %); 3) Se realizó la prueba del clasificador diseñado y se reportaron 0 malas clasificaciones tanto para las resistencias de falla de entrenamiento (50,100,150,...,900 ohms) como para las resistencias de falla de prue-

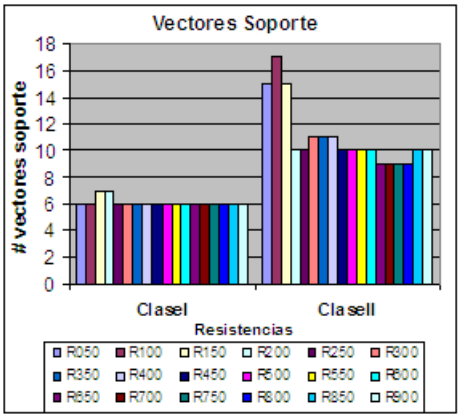


Figura 5.8: Vectores soporte del Tipo Falla 3 contra las 9 restantes

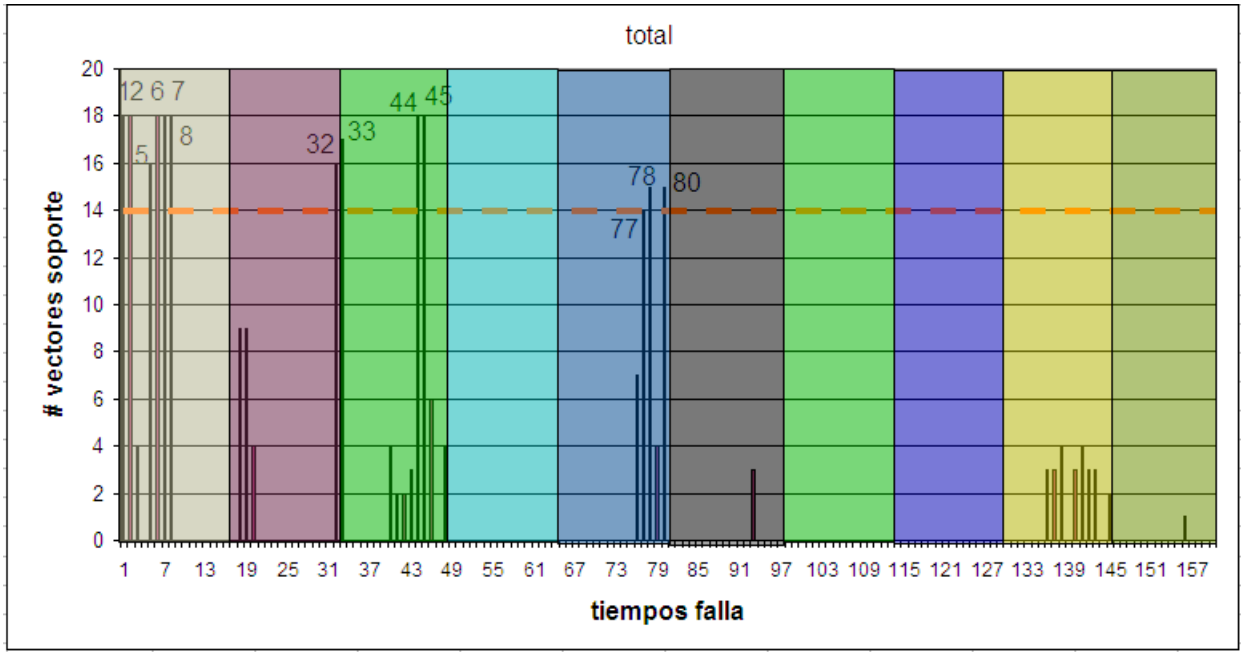


Figura 5.9: Datos que constantemente son vectores soporte

Resistencia	# vectores soporte	ClaseI 16	ClaseII 144
50	22	8	14
100	21	8	13
150	22	8	14
200	20	8	12
250	19	8	11
300	17	6	11
350	16	5	11
400	15	4	12
450	15	4	11
500	14	5	10
550	15	4	10
600	15	5	10
650	15	5	10
700	15	5	10
750	15	5	10
800	15	5	10
850	15	5	10
900	15	5	10

Tabla 5.8: Vectores soporte de la fase B contra las 9 fallas de fases restantes

ba (20,0,40,90,160,999 ohms); 4) Se probó con el clasificador basado en w y *bias* de los datos de fallas para 900 ohms (resistencia de falla) con el fin de clasificar todas las resistencias de prueba y entrenamiento, sin encontrar ninguna mala clasificación.

Experimentación 7

En esta prueba, se tomaron los datos de todas las 18 resistencias de falla (50, 100, 150, ..., 900 Ohms) teniendo un total de 288(18×16) datos de la $ClaseI$ y 2592(18×144) datos para la $ClaseII$, y se consideró el procedimiento de clustering por el incremento en los datos de entrenamiento. Se estudiaron 3 casos en los cuales, en el primero de ellos la $ClaseI$ contiene los datos de la falla monofásica de fase A y la $ClaseII$ los datos de los tipos de fallas restantes; para el segundo, la $ClaseI$ contiene los datos de

Resistencia	Margen	# vectores soporte	# datos mal clasificados	
			entrenamiento	prueba
50	2.5224	21 (13.1 %)	0	0
100	2.1158	23 (14.4 %)	0	0
150	1.7814	22 (13.8 %)	0	0
200	1.5255	17 (10.6 %)	0	0
250	1.3291	16 (10.0 %)	0	0
300	1.1753	17 (10.6 %)	0	0
350	1.0523	17 (10.6 %)	0	0
400	0.9520	17 (10.6 %)	0	0
450	0.8689	16 (10.0 %)	0	0
500	0.7988	16 (10.0 %)	0	0
550	0.7390	16 (10.0 %)	0	0
600	0.6873	16 (10.0 %)	0	0
650	0.6422	15 (9.40 %)	0	0
700	0.6026	15 (9.40 %)	0	0
750	0.5675	15 (9.40 %)	0	0
800	0.5362	15 (9.40 %)	0	0
850	0.5081	16 (10.0 %)	0	0
900	0.4828	16 (10.0 %)	0	0

Tabla 5.9: Resultados de la fase C contra las 9 fallas de fases restantes

la falla monofásica de fase B y los datos de los tipos de fallas restantes se encuentran en la $ClaseII$; para el tercero la $ClaseI$ contiene los datos de la falla monofásica de fase C y la $ClaseII$ los datos de los tipos de fallas restantes. Los resultados se muestran en la Tabla 5.11. Se aplicó el procedimiento $3b$ de la Sección 5.2.2 utilizando $k = 18$ clusters para la $ClaseI$ y $k = 36$ clusters para la $ClaseII$ para los tres casos, se realizó posteriormente el procesamiento de los datos.

Solamente en el segundo caso (Fase B vs fases restantes) de este experimento se presentaron errores de clasificación; los datos mal clasificados fueron 392 de la $ClaseII$ de un total de 2592 datos(Tabla 5.12). Con los resultados obtenidos se aplicó el Paso 3 de la Sección de Procesamiento de datos; los resultados fueron satisfactorios ya que el clasificador diseñado no reportó ningún error en la clasificación. Los resultados que se muestran en la Tabla 5.12 fueron los que se obtuvieron del clasificador generado en 2

Resistencia	# vectores soporte	ClaseI 16	ClaseII 144
50	21	6	15
100	23	6	17
150	22	7	15
200	17	7	10
250	16	6	10
300	17	6	11
350	17	6	11
400	17	6	11
450	16	6	10
500	16	6	10
550	16	6	10
600	16	6	10
650	15	6	9
700	15	6	9
750	15	6	9
800	15	6	9
850	16	6	10
900	16	6	10

Tabla 5.10: Vectores soporte de la fase C contra las 9 fallas de fases restantes

etapas.

ClaseI 18	ClaseII 36	M.Separación	V.Soporte	#datos mal clasif.	
				entrenamiento	prueba
Fase A	Fases restantes	0.5420	4 (7.4 %)	0	0
Fase B	Fases restantes	1.5450	4 (7.4 %)	392/2592	0
Fase C	Fases restantes	0.5274	4 (7.4 %)	0	0

Tabla 5.11: Fases monofásicas contra el resto de las fases

ClaseI	ClaseII	M.Separación	V.Soporte	#datos mal clasif.	
32	288			entrenamiento	prueba
Fase B	Fases restantes	0.5407	3 (5.3 %)	0	0

Tabla 5.12: Fases monofásicas contra el resto de las fases. Clasificador generado en 2 etapas

Capítulo 6

Conclusiones

En este capítulo, daremos a conocer las conclusiones finales de nuestro trabajo de tesis. Primeramente se resumen las conclusiones obtenidas de los experimentos realizados. Posteriormente enumeramos las aportaciones científicas de nuestro trabajo de investigación. Finalmente, emitimos algunas recomendaciones para trabajos posteriores.

6.1. Conclusiones Finales

El objeto de estudio en esta investigación es un problema de identificación de fallas para la protección de líneas de transimisión. Este problema fue modelado como un problema de programación cuadrática debido a las características de no linealidad de la función objetivo y las restricciones lineales. Como se ha mencionado antes, el problema es de gran escala y se requiere grandes cantidades de tiempo para su ejecución; debido a estas características se decidió emplear el algoritmo $K - means$ para la agregación de datos y reducción del tamaño del problema.

En este trabajo de tesis principalmente buscamos cumplir los siguientes objetivos:

- Entendimiento de la estructura matemática del problema que nos permita emplear adecuadamente las técnicas de solución.

- Efectuar una evaluación de los parámetros del algoritmo en estudio.
- Obtener soluciones a un número importante de instancias del problema.

Primeramente resolvimos las instancias creadas cuando construimos nuestra base de datos sin utilizar el simulador PSCAD y sin utilizar el algoritmo de clustering $k-means$. El resultado fue que no muchas de las instancias pudieron ser resueltas, y de las que sí pudieron solucionarse, empleaban demasiados recursos computacionales y necesitaban mucho tiempo para lograr obtener una solución.

Posteriormente, resolvimos las instancias creadas utilizando el simulador y el algoritmo clustering. El resultado fue que todas las instancias pudieron ser resueltas de manera efectiva (ya que el clasificador diseñado no presentó ningún problema de clasificación) y en un tiempo razonable. También obtuvimos muy buenos resultados para resistencias de falla (20,30,40,90,160 y 999 ohms) diferentes a las resistencias de falla de los conjuntos de aprendizaje (50, 100, 150,..., 850, 900 ohms).

Finalmente, la única instancia que presentó error de clasificación descrita en 5.2.2 (experimento 7) fue resuelta satisfactoriamente en una segunda etapa de la misma sección; en resumen el desempeño del algoritmo propuesto obtuvo muy buenos resultados.

6.2. Aportaciones Científicas

Entre las aportaciones de este trabajo se encuentran:

- Aplicación del algoritmo de vectores soporte para el diseño de un discriminante lineal de mayor margen de separación entre clases.
- Utilización de clustering para reducción de datos.

- Diseño de un algoritmo iterativo para obtención de soluciones, cuando el método de vectores soporte genera un discriminante lineal con errores de clasificación.
- Resultados y conclusiones satisfactorias respecto al problema tratado con la metodología propuesta.
- Nueva propuesta de representación de fallas, al sustituir la matriz de corriente de fases 3×16 a una de 1×48 ordenado por columnas de la matriz.
- Aportación del análisis, estudio y solución al problema.

Algunos resultados de este trabajo han sido presentados en:

1. N. Garza-Padilla. "Uso de técnicas de agregación en el diseño de clasificadores de vectores soporte". Ciclo de seminarios de PISIS, FIME, UANL, Mayo 2004.
2. N. Garza-Padilla. "Uso de técnicas de agregación en el diseño de clasificadores de vectores soporte", Escuela Nacional de Optimización y Análisis Numérico, Morelia, Michoacán, Abril 2005.
3. Dr. Oscar Chacón Mondragón - N. Garza Padilla - Dr. Ernesto Vázquez. "Support Vector Classification Through Clustering Process" Theforth IASTED International Conference on Computational Intelligence (SI2005) July 4-5 2005. Calgary, Alberta, Canadá.

6.3. Recomendaciones para Trabajos Posteriores

- Una recomendación es seguir incrementando esta base de datos, para que finalmente, los métodos con que se propone solucionar este problema, sea instancias

de una diversidad mayor, las cuales por supuesto, serán mucho más grandes. Esto se logra considerando diferentes forma de diezmado de la señal, reducción de tamaños de la muestra a un medio y/o cuarto de ciclo.

- Realizar experimentación para cuantificar la severidad y ubicación de la falla, basado principalmente en la resistencia de falla.
- Desarrollar un método jerárquico de discriminación de los 10 tipos de fallas.
- Utilización de un algoritmo para encontrar el número óptimo de clusters.

Bibliografía

[1] K.M. Agrawal, A.M. Jade, V.K. Jayaraman, and B.D. Kulkarni. Support vector machines: A useful tool for process engineering applications. *ProQuest Science Journals*, 99(1):57–62, 2003.

[2] I.E. Alguindigue, A. Loskiewicz-Buczak, and R. Uhrig. Monitoring and Diagnosis of Rolling Element Bearings Using Artificial Neural Networks. *IEEE Transactions on Industrial Electronics*, 40(2):209–217, 1993.

[3] L.M. D Andrea, G.L. Fisher, and T.C. Harrison. Cluster analysis of adult children of alcoholics. *International Journal of addictions*, 29(5):565–582, 1994.

[4] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49(3):803–821, 1993.

[5] A.F. Bergh, F.K. Soong, and L.R. Rabiner. Incorporation of temporal structure into a vector-quatization-based preprocessor for speaker-independient, isolated-word recognition. *AT&T Technical Journal*, 64(5):1047–1063, 1985.

[6] D.P. Bertsekas. Nonlinear programming. *Athenas Scientific*, 1995.

[7] J.N. Breckenridge. Replicating cluster analysis: method, consistency, and validity. *Multivariate Behavioral Research*, 24(32):147–161, 1989.

[8] I. V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 140–149, 2000.

[9] J.A. Conway, L.M.J. Brown, N.J. Veck, and R.A. Cordey. A model-based system for crop classification from radar imagery. *GEC Journal of Research*, 9(1):46–54, 1991.

[10] A. S. Cook. The complexity of theorem-proving procedures. *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[11] T.H. Cormen, C.E. Leiserson, and R.R. Rivest. Introduction to algorithms. *MIT Press*, 1990.

[12] P.C. Cosman, C. Tseng, R.M. Gray, R.A. Olshen, L.E. Moses, H.C. Davidson, C.J. Bergin, and E.A. Riskin. Tree-structured vector quantization of CT chest scans: image quality and diagnostic accurancy. *IEEE transactions on Medical Imaging*, 12(4):727–739, 1993.

[13] P. Dellaportas. Bayesian classification of neolithic tools. *Applied Statistics*, 47(2):279–297, 1998.

[14] W.F. Eddy, A. Mockus, and S. Oue. Approximate single linkage cluster analysis of large data sets in high-dimensional spaces. *Computational Statistics and Data Analysis*, 23:29–43, 1996.

[15] B.S. Everitt. *Cluster Analysis*. Wiley, New York, 1974.

[16] E.A. Ferrán, B. Pflugfelder, and P. Ferrara. Self-organized neural maps of human protein sequences. *Protein Science*, 3:507–521, 1994.

[17] S. Furui. Recent advances in speaker recognition. *Pattern Recognition Letters*, 18:859–872, 1997.

[18] M.R. Garey and D.S. Johnson. Computers and Intractability. *W.H. Freeman*, 1979.

[19] U. Halici and G. Ongun. Fingerprint classification through self-organising feature maps modified to treat uncertainties. *Proceedings of the IEEE*, 84(10):1497–5112, 1996.

[20] Simon Haykin. *Neural Networks*. Prentice Hall, Inc, 1999.

[21] AK Jain and RC Dubes. *Algorithms for Clustering Data*. Prentice-Hall, New Jersey, 1998.

[22] L Kaufman and Rousseeuw PJ. *Finding Groups in Data*. Wiley, New York, 1990.

[23] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organising map. *Proceedings of the IEEE*, 84(10):1358–1384, 1996.

[24] B. Li, M. Chow, Y. Tipsuwan, and J.C. Hung. Neural Network Based Motor Rolling Bearing Fault Diagnosis. *IEEE Transactions on Industrial Electronics*, 47(5):1060–1069, 2000.

[25] C. Li, D.B. Goldgof, and L.O. Hall. Knowledge-based classification and tissue labeling of MR images of human brain. *IEEE Transactions on Medical Imaging*, 12(4):740–750, 1993.

[26] J. Makhoul, S. Roucos, and H. Gish. Vector quatization in speech coding. *Proceedings of the IEEE*, 73(11):1511–1588, 1985.

[27] Dr. Ernesto Vázquez Martínez. Personal communication: evazquez@gama.fime.uanl.mx.

[28] R.M. McIntyre and R.K. Blashfield. A nearest-centroid technique for evaluating the minimum-variance clustering procedure. *Multivariate Behavioral Research*, 2:225–238, 1980.

[29] W. Meier, R. Weber, and H.J. Zimmermann. Fuzzy data analysis-methods and industrial applications. *Fuzzy Sets and Systems*, 61:19–28, 1994.

[30] TM Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[31] B.J.T. Morgan and A.P.G. Ray. Non-uniqueness and inversions in cluster analysis. *Applied Statics*, 44(1):117–134, 1995.

[32] D.P. Mukherjee, D.K. Banerjee, B. Uma Shankar, and D.D. Majumder. Coal petrography; a pattern recognition approach. *Pattern Recognition*, 28(2):269–281, 1994.

[33] D.P. Mukherjee, A. Pal, S.E. Sarma, and D.D. Majumder. Water quality analysis: a pattern recognition approach. *Pattern Recognition*, 28(2):269–281, 1995.

[34] G.E. Powell, E. Clark, and S. Bailey. Categories of aphasia: a cluster anlysis of Schuell test profiles. *British Journal of Disorders of Communication*, 14(2):111–122, 1979.

[35] S. Poyhonen, A. Arkkio, P. Jover, and H. Hyotyniemi. Coupling pairwise support vector machines for fault classification. *Control Engineering Practice*, 13:759–769, 2005.

[36] S Sharma. *Applied Multivariate Techniques*. Wiley, New York, 1996.

[37] C. Stewart, Y.C. Lu, and V. Larson. A neural clustering approach for high resolution radar target classification. *Pattern Recognition*, 27(4):503–513, 1994.

[38] V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[39] Andrew R. Webb. *Statistical Pattern Recognition*. John Wiley and Sons, Ltd., 2002.

[40] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[41] M.S. Yang. A survey of fuzzy clustering. *Mathematical and Computer Modelling*, 18(11):1–16, 1993.

[42] X. Yuan, T. M. Khoshgoftaar, E. Allen, and K. Ganesan. An application of fuzzy clustering to software quality prediction. In *Proc. IEEE Symposium on Application Specic Systems and Software Engineering Technology (ASSET'00)*, pages 85–90, 2000.

[43] Y. Zhang, C.J.S. de Silva, R. Togneri, M. Alder, and Y. Attikiouzel. Speaker-independient isolated word recognition using multiple hidden Markov models. *IEEE Proceeding on Vision, Image and Signal Processing*, 141(3):197–202, 1994.

[44] S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.

[45] Shi Zhong, Taghi M. Khoshgoftaar, and Naeem Seliya. Unsupervised Learning for Expert-Based Software Quality Estimation. *8th IEEE International Symposium on High Assurance Systems Engineering*, 2004.

Índice de Tablas

Índice de figuras

Apéndice A

Vectores y Matrices

Propiedades básicas y definiciones

Un vector $x(n,1)$ de n elementos está definido con una columna de dichos elementos

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

y una matriz $A(m,n)$ que tiene m renglones y n columnas, se representa como:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}$$

Producto Interno y Externo El producto interno (escalar,punto) de dos vectores n-dimensionales x y w es un escalar $a : a = x^T w = w^T x$. El producto externo de x y w

es una matriz A.($x \in \Re^m$ y $w \in \Re^n$.)

$$A = xw^T = \begin{bmatrix} x_1w_1 & x_1w_2 & \cdots & x_1w_j & \cdots & x_1w_n \\ x_2w_1 & x_2w_2 & \cdots & x_2w_j & \cdots & x_2w_n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_iw_1 & x_iw_2 & \cdots & x_iw_j & \cdots & x_iw_n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mw_1 & x_mw_2 & \cdots & x_mw_j & \cdots & x_mw_n \end{bmatrix}$$

Los resultados de la multiplicación de matrices son los siguientes:

- Una expresión terminando con un vector columna es un vector columna: ABx=c.
- Una expresión iniciando con un vector renglón es un vector renglón: $y^TBCD = r^T$.
- Una expresión iniciando con un vector renglón y terminando con un vector columna es un escalar: $x^TAy = s$.

Independencia Lineal de Vectores Sean $a_1, a_2, ..., a_n$ vectores en $\Re^m$, y $\alpha_1, \alpha_2, ..., \alpha_n$ escalares; los vectores a son linealmente independientes si

$$\sum_{i=1}^{n} \alpha_i a_i = 0 \Leftrightarrow \alpha_1, \alpha_2, ..., \alpha_n = 0.$$

Las columnas (renglones) de $A \in \Re^{m,n}$ son linealmente independientes si y solo si A^TA es una matriz no singular, $A^TA = |A^TA| \neq 0$.

Normas de Vectores Las normas son escalares (positivas) y son usadas como medidas de longitud, tamaño, distancia, etc, dependiendo del contexto. Una norma L_p $x(n, 1)$ se representa y se define como:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Principalmente, $p = 1, 2,\ o\ \infty$, y éstas normas son llamadas normas uno-, dos-, o infinita.

$\|x\|_1 = \sum_{i=1}^{n} \|x_i\|$.	(valor absoluto, norma L_1)
$\|x\|_2 = \sqrt{x^T x} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = (\sum_{i=1}^{n} x_i^2)^{1/2}$	(módulo, Euclidiana, norma L_2)
$\|x\|_w = \sqrt{x^T W x}$, W es simétrica positiva.	(norma Euclidiana de peso)
$\|x\|_\infty = max\|x_i\|$.	(infinito, Chebyshev, norma L_∞)
$\|x\| \geq 0$. Si, $x \neq 0, \|\alpha x\| = \|\alpha\|\|x\|$, para cualquier α.	(cualquier norma)
$\|x + y\| \leq \|x\| + \|y\|$.	(desigualdad triangular)

Dada una matriz A $m \times n$, denotamos el elemento del *i-ésimo* renglón y la *j-ésima* columna por a_{ij}. Una matriz A es llamada *cuadrada* si $m = n$, o bien A es una matriz (n,n). Una matriz (m,n) es una matriz *rectangular*. Los vectores también pueden ser vistos como matrices rectangulares donde $(n = 1)$. $[a_{ij}]_{m,n}$ es un elemento del *i-ésimo* renglón y la *j-ésima* columna de la matriz A.

Cuando los elementos de una matriz son reales, $A \in \Re^{m,n}$, las columnas de A se denotan por $a_i \in \Re^{m,1} = \Re^m, i = 1, 2, ..., n$, y A puede ser expresado en térmios de sus columnas por $A = [a_1 a_2 \cdots a_n]$.

La *transpuesta* A^T de una matriz A de $m \times n$ es una matriz de $n \times m$ y se comprueba que

$$(A^T)^T = A \qquad (AB)^T = B^T A^T \qquad (ABCD)^T = D^T C^T B^T A^T$$
$$A^T = [a_{ji}] \quad [A + B]^T = A^T + B^T$$

Una matriz cuadrada A es *simétrica* si a_{ij}=a_{ji} para toda i, j, es decir, $A = A^T$.

Una matriz es *diagonal* si $a_{ij} = 0$ para $i \neq j$, esto es, $A = (a_{11} a_{22} \ldots a_{nn})$. Si $a_{ii} = 1$, la matriz, es una matriz identidad.

El *trazo de una matriz* cuadrada A, denotado por $Tr\{A\}$, es la suma de sus elementos diagonales,

$$Tr\{A\} = \sum_{i=1}^{n} a_{ii}$$

El *determinante* de una matriz A, representado como $|A|$, se define como

$$|A| = \sum_{i=1}^{n} a_{ij} A_{ij} \text{ para } i = 1, ..., n$$

donde el cofactor A_{ij}, es el determinante de la matriz formada al eliminar el i-ésimo renglón y la j-ésima columna de A multiplicado por $(-1)^{i+j}$. La matriz de cofactores, $C(c_{ij} = A_{ij})$, es llamado el *contiguo* de A. Si A y B son matrices cuadradas del mismo orden, entonces $|AB| = |A||B|$. La *inversa* de una matriz A es aquélla única matriz A^{-1} con elementos tales que

$$A^{-1}A = AA^{-1} = I$$

donde I es la matriz identidad. Si la inversa existe, se dice que la matriz es *no singular*. Si la inversa no existe, la matriz es *singular* y $|A| = 0$.

Una matriz cuadrada es *ortogonal* si

$$AA^T = A^T A = I$$

esto es, los renglones y las columnas de A son ortogonales ($x^T y = 0$ *y* $x^T x = 1$, $y^T y = 1$ para dos columnas x y y diferentes). Una matriz ortogonal representa una transformación lineal que conserva distancias y ángulos, que constan de una rotación y/o reflección. Es claro, de la definición anterior, que una matriz ortogonal es no singular y la inversa de una matriz ortogonal es su transpuesta: $A^{-1} = A^T$. También, el determinante de una matriz ortogonal es ± 1 (-1 indica una reflección, +1 una rotación pura).

Una matriz cuadrada A es definida positiva si la forma cuadrática $x^T Ax > 0$ para toda $x \neq 0$. La matriz es semidefinida positiva si $x^T Ax \geq 0$ para toda $x \neq 0$.

El *rango* de una matriz es el máximo número de renglones linealmente independientes (o equivalente, el máximo número de columnas linealmente independientes).

Una matriz $n \times n$ tiene rango lleno si el rango es igual a n. En este caso, el determinante no es cero y la inversa existe. Para una matriz rectangular A de orden $m \times n$, $rango(A) \leq min(m, n)$ y

$$rango(A) = rango(A^T) = rango(A^T A) = rango(AA^T)$$

y el rango no se cambia por la pre- o post- multiplicación de A por una matriz no singular.

Suma, Resta, y Multiplicación de Matrices

$$C = A \pm B, \text{ o} \qquad c_{ij} = a_{ij} \pm b_{ij}$$
$$A \pm B = B \pm A \quad A \pm (B \pm C) = (A \pm B) \pm C$$
$$kA = Ak = [ka_{ij}] \qquad kA + kB = k(A + B)$$

El producto AB de una matriz A de (m, n) por una matriz B de (n, p) es una matriz C de (m, p):

$$C = AB \quad c_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj}, \quad i = 1, ..., m, j = 1, ..., p.$$
$$AB \neq BA \quad (AB)C = A(BC) \quad (A + B)C = AC + BC$$

Para matrices diagonales,

$$AD = DA, \text{ y } \quad ABC = BAC = CAB = CBA.$$

Problema Lineal de Mínimos Cuadrados El problema es encontrar el valor de x que minimice $\|Ax - y\|_2^2$, esto es, x debe minimizar la función de error

$$E(x) = \frac{1}{2}(Ax - y)^T (Ax - y)$$

La solución a este problema se obtiene de la condición necesaria y suficiente para un óptimo (mínimo)

$\frac{\partial E}{\partial x} = 0$

$\frac{\partial E}{\partial x} = 0 = \frac{\partial (Ax-y)^T (Ax-y)}{\partial x} = \frac{\partial (y-Ax)(y-Ax)^T}{\partial x}$

$\frac{\partial E}{\partial x} = \frac{\partial (yy^T + Axx^T A^T - Axy^T - yx^T A^T}{\partial x}$

$\frac{\partial E}{\partial x} = 0 + 2A^T Ax - A^T y - A^T y = 0.$

$2(A^T Ax - A^T y) = 0.$

$x^* = (A^T A)^{-1} A^T y.$

Apéndice B

Álgebra Lineal y Geometría Analítica

Considere dos vectores a y b de $(n, 1)$. El *producto escalar* (interno o punto) es dado como:

$$a^T b = b^T a = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n.$$

La longitud de un vector a es dada como $\|a\| = \sqrt{a^T a} = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$. El ángulo α entre dos vectores a y b puede ser obtenido de $a^t b = \|a\|\|b\| cos\alpha$ como

$$cos\alpha = \frac{a^T b}{\|a\|\|b\|} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}\sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}}$$

Claramente, podemos ver que cuando dos vectores son ortogonales ($\alpha = 90$ grados), entonces $cos\alpha = 0$. En otras palabras, cuando $a^T b = b^T a = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = 0$, los dos vectores son ortogonales.

El producto escalar es también igual al valor absoluto (la longitud) de uno de los vectores multiplicado por la proyección algebráica del otro vector en la dirección del primero: $a^T b = \|a\| b_a = \|b\| a_b$, Fig.B.1.

Hiperplano El conjunto de puntos $(x_1, x_2, ..., x_n) \in \Re^n$ satisfacen la ecuación

$$w_1 x_1, w_2 x_2, ..., w_n x_n + w_{n+1} = 0 \tag{B.1}$$

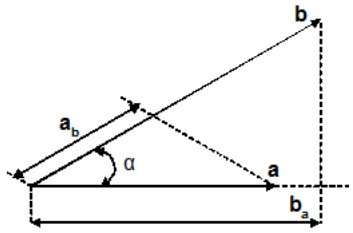


Figura B.1: Producto Escalar

donde las w_i no son todas cero, forman un hiperplano (una cubierta lineal de $n-1$), cualquier plano $\Re^n$ puede ser definido por la ecuación anterior.

Forma Cuadrática Una forma cuadrática es una función cuadrática de la forma siguiente:

$$Q = x^T Ax = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$$
$$Q = a_{11}x_1^2 + (a_{12} + a_{21})x_1x_2 + (a_{13} + a_{31})x_1x_3 + \cdots + a_{22}x_2^2 + \cdots),$$

donde x es un vector $(n, 1)$ y A es una matriz simétrica de orden n.

La forma cuadrática es *definida positiva* si $x^T Ax > 0$ para todos los vectores $x \neq 0$. Se dice que es *definida negativa* si $x^T Ax < 0$ para todos los vectores $x \neq 0$, y *semidefinida positiva* si $x^T Ax \geq 0$ para $x \neq 0$.

Apéndice C

Bases de Análisis Multivariable

Gradiente Definimos el *gradiente* de f en cada punto x donde existan todas las derivadas parciales como el vector

$$\nabla_x f(x) = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right)$$

Diferenciación de una Función Escalar con Respecto a un Vector

1. $\frac{\partial (a^T(x)b(x))}{\partial x} = \left[\frac{\partial (a(x))}{\partial x_j} \right]^T b(x) + \left[\frac{\partial (b(x))}{\partial x_j} \right] a(x) = [\nabla_x a^T(x)]b(x) + [\nabla_x b^T(x)]a(x).$

2. $\frac{\partial (x^T y)}{\partial x} = y.$

3. $\frac{\partial (x^T x)}{\partial x} = 2x.$

4. $\frac{\partial (x^T A y)}{\partial x} = Ay.$

5. $\frac{\partial (y^T A x)}{\partial x} = (y^T A)^T = A^T y.$

6. $\frac{\partial (x^T A x)}{\partial x} = (A + A^T)x,$ si A no es simétrica.

7. $\frac{\partial (x^T A x)}{\partial x} = 2x^T A = 2Ax,$ si A es simétrica.

8. $\frac{\partial [a^T x Q a(x)]}{\partial x} = 2[\nabla_x a^T(x)]Qa(x).$

Apéndice D

Bases de la Teoría de Probabilidad

Definición y terminología

Una colección de objetos se llama *conjunto*. Un miembro de esta colección se llama *elemento* del conjunto. Si a es un elemento de un conjunto S, diremos también que a está en S, y se denota por $a \in S$. Para denotar el hecho de que S está formado por los elementos $a, b, ...$ a menudo usamos la notación $S = \{a, b, ...\}$. Si S y S' son conjuntos y todo elemento de S' es un elemento de S, decimos que S' es un *subconjunto* de S ($S' \subset$).

El conjunto de todos los posibles resultados es llamado *espacio muestral* (Ω). Un evento A es un conjunto de resultados experimentales y corresponde a un subconjunto en Ω.

Operaciones con conjuntos

$\bar{A}$ complemento de A

$A \cup B$ unión de A y B

$A \cap B$ intersección de A y B

Propiedades

Conmutativa: $A \cup B = B \cup A \quad A \cap B = B \cap A$

Asociativa: $(A \cup B) \cup C = A \cup (B \cup C) \quad (A \cap B) \cap C = A \cap (B \cap C)$

Distributiva: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Probabilidad

Una *medida de probabilidad* es una función $P(A)$, con un conjunto A como argumento. Este puede ser considerado como la proporción de veces que A ocurre y tiene las siguientes propiedades:

1. $0 \leq P(A) \leq 1$.
2. $P(\Omega) = 1$ Si Ω es un evento cierto.
3. $P(\phi) = 0$ Si ϕ es un evento imposible.
4. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$. Si $P(A \cap B) = 0$ (los eventos son mutuamente excluyentes).
5. $P(A_1 \cup A_2 \cup A_3 \cup \cdots) = \sum_{i=1}^{\infty} P(A_i)$. Si $A_i \cup A_j = 0$ para cada $i \neq j$ y cuando hay un número infinito de eventos.

Una variable aleatoria x es una cantidad que puede tener diferentes valores de tal manera que para cada número real X dado la probabilidad $P(x \leq X)$ es definida. La variable aleatoria x pude ser *discreta*, esto es, puede tener un conjunto finito de distintos valores, o puede ser *continua*. Dado un sistema aleatorio y dos eventos cualquiera (A y B) que pueden ocurrir juntos, podemos formar un nuevo sistema tomando sólo aquéllos experimentos en donde B ocurre. La probabilidad de A en este nuevo sistema es llamada la *probabilidad condicional* de A dado B y se denota por $P(A|B)$. Si $P(B) > 0$ esta dado por:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Apéndice E

Teoría de Optimización

Consideraciones Generales

La palabra *optimización* se utiliza para referirse a funciones con un mínimo o máximo. Se trabaja con una función objetivo $f : D \subset \Re^n \rightarrow \Re$ donde D es un conjunto abierto. En un *problema de minimización con restricciones* se busca encontrar una x_0 tal que

$$f(x_0) = min\{f(x) : x \in K\}$$

donde $K \subset D$ es especificado. Un punto x_0 es un *punto de solución* y K es el conjunto de restricciones. Entonces el *problema general con restricciones* se representa como:

$$\begin{aligned} &\text{minimizar} \quad f(x) \\ &\text{sujeto a} \quad x \in K \end{aligned} \tag{E.1}$$

cuando $K = D$ entonces es un *problema de optimización sin restricciones*. Es convencional referirse a K como una *región factible* o conjunto de *puntos factibles*. Así, x es factible para E.1 si y solo si $x \in K$.

Condiciones de Karush-Kuhn-Tucker para Optimalidad Sean $f(x)$, $g_i(x)$, $h_t(x)$ valores reales de funciones diferenciables en $\Re^n$, para todo i, t. Considere el si-

guiente problema matemático:

$$\text{minimizar } \ f(x)$$
$$\text{sujeto a } \ g_i(x) \geq 0 \ \text{ para } \ i = 1, ..., m$$
$$h_t(x) = 0 \ \text{ para } \ t = 1, ..., p$$

El Lagrangiano Karush-Kuhn-Tucker para este problema es

$$L(x, \pi, \mu) = f(x) - \sum_{i=1}^{m} \pi_i g_i(x) - \sum_{t=1}^{p} \mu_t h_t(x),$$

donde π_i, μ_t son los multiplicadores de Lagrange asociados con las restricciones.

Las condiciones de Karush-Kuhn-Tucker (KKT) necesarias para optimalidad son:

$$[\partial L(x, \pi, \mu)/\partial x] = \nabla f(x) - \textstyle\sum_{i=1}^{m} \pi_i \nabla g_i(x) - \sum_{t=1}^{p} \mu_t \nabla h_t(x) = 0,$$
$$g_i(x) \geq 0 \ \text{ para } \ i = 1, ..., m$$
$$h_t(x) = 0 \ \text{ para } \ t = 1, ..., p$$
$$\pi_i g_i(x) = 0 \ \text{ para } \ i = 1, ..., m$$
$$\pi_i \geq 0 \ \text{ para toda } \ i$$

donde $\nabla f(x)$, son los vectores de las derivadas parciales. Si $\bar{x}$ es un mínimo local para este problema, existen los multiplicadores $\bar{\pi}, \bar{\mu}$ tal que $\bar{x}, \bar{\pi}, \bar{\mu}$ satisfagan las condiciones de KKT.

Conjuntos Convexos

Un subconjunto $K \subset \Re^n$ se dice que es un conjunto convexo si para cada combinación convexa de cualquier par de puntos en K está también en K. Esto es, si $\tilde{x} \in K$, y $\hat{x} \in K$, entonces $\alpha\tilde{x} + (1 - \alpha)\hat{x} \in K$ para toda $0 \leq \alpha \leq 1$.