Univ. Michoacana de San Nicolas de Hgo.
Facultad de Ingeniería Eléctrica
División de estudios de Postgrado
Maestría y Doctorado en Ciencias en Ing Eléctrica
Opción Sistemas Computacionales

Notas de **Reconocimiento de Patrones**

José Antonio Camarena Ibarrola

Marzo de 2009

Índice general

Capítulo 1

Introducción

Estas notas fueron desarrolladas como apoyo a los estudiantes que cursan la materia de Reconocimiento de Patrones en la Maestría y el Doctorado en Ingeniería Eléctrica, opción Sistemas Computacionales. El programa de esta materia incluye temas de diversos libros los cuales son difíciles de conseguir y de algunos artículos científicos, las presentes notas son un compendio que se apega a los temas que indica el programa. Los ejemplos que se incluyen han sido desarrollados a mucho mayor detalle que lo que usualmente aparece en la bibliografía. Finalmente todavía ocurre sobre todo en los estudiantes de maestría (Quizás de Doctorado no) que se les dificulta la comprensión de material escrito en Inglés, estas notas son un apoyo también para aquellos alumnos con dificultades de lectura del inglés técnico del área de reconocimiento de patrones. Estas notas se encuentran a disposición de los estudiantes o de cualquier interesado en los temas de este curso en http://lc.fie.umich.mx/∼camarena/NotasReconPat.pdf. Se agradecen las observaciones y comentarios, favor de dirigirlos a camarena@umich.mx

Atentamente: Dr. José Antonio Camarena Ibarrola. Autor

Sistemas de *Reconocimiento de voz* (ASR por Automatic Speech Recognition); *Reconocimiento de huellas dactilares* (AFIS por Automatic Fingerprint Identification System); Reconocimiento de rostros, de firmas, de retina y muchos otros pertenecen a un área de la Inteligencia Artificial denominada *Reconocimiento de Patrones*. Un sistema basado en reconocimiento de patrones tiene el objetivo de identificar la clase a la que pertenece un patron que se le presenta, dicho de otra manera tiene la responsabilidad de clasificar objetos.

1.1. Esquema de un Sistema de Reconocimiento de Patrones

Para cumplir su misión, un sistema de reconocimiento debe primero extraer características a partir la señal cualquiera que sea su naturaleza (señal de audio, imagen, etc.), dichas características deben ser robustas, es decir deberían permanecer en la señal aunque esta sufra deformaciones por ruido u otras formas de degradación. Para extraer dichas características se utilizan una variedad de técnicas de *procesamiento digital de señales* que en muchas ocasiones hacen uso de alguna transformación. En la Figura 1.1 se muestra el proceso de reconocimiento de un patrón a partir de una señal, el preprocesamiento de la señal normalmente incluye la segmentación, es decir, la separación del área de interés del resto de la señal, por ejemplo, un sistema de reconocimiento de rostros debe ubicar el área donde se encuentra realmente el rostro en la imagen capturada para separarla del resto (Ej, del fondo de la imagen), el preprocesamiento implica casi siempre una normalización, por ejemplo, un sistema de reconocimiento de firmas debe ser independiente del color de tinta con el que un individuo ha firmado, por eso el preprocesamiento convertiría en esta etapa la imagen a color a una imagen en tonos de gris. El clasificador en sí compara las características extraídas de la señal con las características de los patrones conocidos a los que previamente se les extrajo dichas características.

1.1.1. Adquisición de Datos

Para poder aplicar las técnicas de procesamiento de señales que habrán de extraer características que permitan realizar la labor de un reconocedor de patrones se debe contar con un subsistema de adquisición de datos, el cual

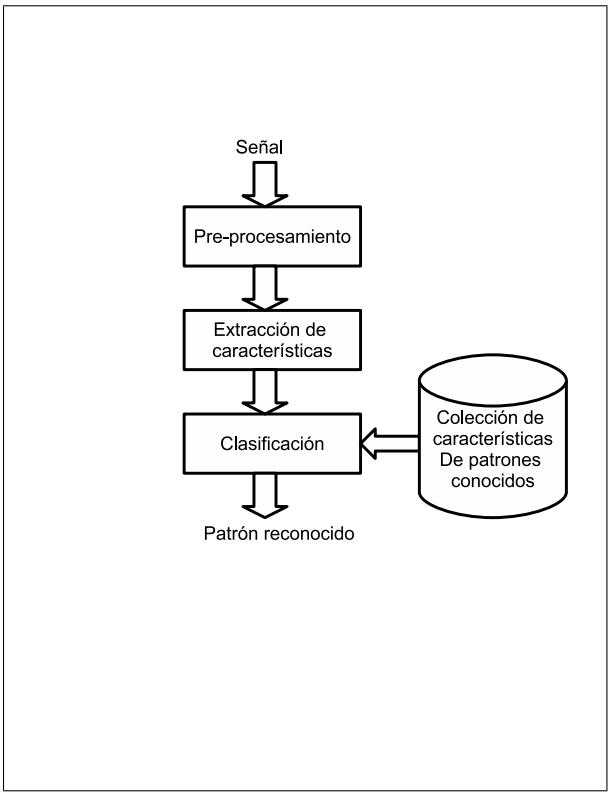

Figura 1.1: Proceso de reconocimiento de patrones

debe contar con sensores que no son otra cosa que transductores de variables físicas como presión, cantidad de luz, temperatura, etc a señales eléctricas analógicas. La señal eléctrica, normalmente una corriente o un voltaje debe convertirse en una señal digital para lo cual debe pasar por un convertidor analógico-digital o A/D. El convertidor A/D tiene algunos parámetros que hay que tomar en cuenta:

- Tiempo de conversión. Es el tiempo que le toma al convertidor convertir una muestra de la señal analógica a un vector de ceros y unos (El valor convertido a binario).
- Frecuencia de muestreo. Normalmente a un convertidor A/D se le indica el número de muestras por segundo que debe enviar para poder formar la señal. La frecuencia de muestreo no puede ser tan alta que el tiempo de conversión del convertidor no la permita, por otro lado si la frecuencia de muestreo es demasiado baja y no se cumple la regla de Nyquist que dice que se debe digitalizar al menos al doble de la frecuencia máxima que se contempla en la señal.
- Precisión. El número de bits por muestra permite tener mayor o menor resolulción en la señal digitalizada, por ejemplo si se digitaliza a 8 bits por muestra solo podremos tener 256 posibles valores en la señal digitalizada.
- Filtrado anti-aliasing. En vista de que al reconstruir la señal digitalizada no podremos tener los componentes de frecuencia que estén por arriba de la mitad de la frecuencia de muestreo, normalmente las tarjetas digitalizadoras incluyen un filtro antialiasing que de otra forma deformaría la señal (introduciendo alias o componente inexistente en la misma)

1.1.2. Extracción de Características

Las características de un patrón deben ser suficientes para diferenciar a los objetos que se pretenden reconocer, por ejemplo, para reconocer a una persona podemos nombrar sus cualidades como mujer, alta, rubia, ojos verdes, edad aproximada, etc. Sin embargo un buen reconocedor debería utilizar el menor número de características que le permitan realizar eficientemente su trabajo para no caer bajo la *maldición de la dimensionalidad* de la cual

hablaremos más adelante. En efecto, el número de características que se extraen de los patrones define el número de dimensiones del *espacio de características*. Asuma que cierto sistema de reconocimiento extrae solo dos características de los patrones con los que trabaja, por ejemplo, para separar pelotas en una linea de producción de una fábrica de juguetes, quizás solo necesite el tamaño y su brillantez, al poner en el eje horizontal el tamaño y en el eje vertical la brillantez tenemos un plano euclidiano como el de la Figura 1.2, en el caso de que se necesitaran tres características los objetos (patrones) estarían representados por puntos en un espacio 3D. Es común por ejemplo en reconocimiento de voz que un sonido fonético esté representado por 10 características en cuyo caso, cada diferente sonido que un ser humano puede emitir estaría representado por un punto en un hiper-espacio euclidiano de diez dimensiones.

1.1.3. Elección del Modelo

Un clasificador o reconocedor de patrones puede ser representado por un modelo, el modelo incluye las características que los patrones deben usar, la forma en que los datos de entrenamiento se usaran para efecto de realizar correctamente el reconocimiento. ¿Como saber cuando un modelo hipotético difiere significativamente del modelo inherente a los patrones que queremos clasificar?, tal vez decidamos que debemos usar otro modelo, y ¿Como sabremos cuando rechazar uno y probar otro?. ¿Debemos estar condenados al ensayo y error en la elección del modelo sin siquiera saber si debemos esperar una mejora en el desempeño del nuevo modelo?. Finalmente, ¿Existe una manera de automatizar la elección del modelo?

1.1.4. Entrenamiento

Un sistema de reconocimiento de patrones de aprender a clasificar objetos, este proceso se conoce como entrenamiento. En el entrenamiento supervisado los datos de entrenamiento consisten de parejas de entradas y sus salidas correspondientes, en el caso de las redes de neuronas los datos de entrenamiento sirven para ajustar la fuerza con la que se conectan las neuronas de una capa con las neuronas de la capa siguiente. Aunque el entrenamiento supervisado es el mas común, algunas veces no es posible tener etiquetados los datos de entrenamiento por lo que nos vemos obligados a realizar entrenamiento no supervisado, al entrenamiento no supervisado se le conoce también como

agrupamiento pues eso es lo que realmente se tiene que hacer para asignarles una etiqueta sintética.

1.1.5. Evaluación del Modelo

Para evaluar un sistema de reconocimiento de patrones existen varias técnicas, lo que queremos averiguar es que tanto se equivoca o que tanto el sistema puede ser optimizado para evaluar clasificadores podemos mencionar las siguientes herramientas:

- Matrices de confusión. Consiste en comparar los datos de prueba de forma en que todos se usan, al final se llena una matriz con las distancias de cada patron de prueba contra cada uno de los patrones de entrenamiento, dicha matriz tiene tantos renglones como patrones de prueba y en cada renglón se debe verificar un valor bajo para el patron que corresponde a su clase y valores (distancias) altos para aquellos que no corresponden a su clase.

- Curvas ROC. En la Teoría de detección de señales una curva ROC (acrónimo de Receiver Operating Characteristic, o Característica Operativa del Receptor) es una representación gráfica de la sensibilidad frente a $(1 - especificidad)$ para un sistema clasificador binario según se varía el umbral de discriminación. Otra interpretación de este gráfico es la representación de la razón de verdaderos positivos (VPR = Razón de Verdaderos Positivos) frente a la razón de falsos positivos (FPR = Razón de Falsos Positivos) también según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo). El análisis de la curva ROC, o simplemente análisis ROC, proporciona herramientas para seleccionar los modelos posiblemente óptimos y descartar modelos subóptimos independientemente de (y antes de especificar) el coste de la distribución de las dos clases sobre las que se decide. La curva ROC es también independiente de la distribución de las clases en la población (en diagnóstico, la prevalencia de una enfermedad en la población). El análisis ROC se relaciona de forma directa y natural con el análisis de coste/beneficio en toma de decisiones diagnósticas.

- histogramas de distancias. En un buen clasificador, el histograma de distancias reporta que no solo el patrón fue identificado correctamente

sino que la distancia al patrón de su misma clase es mucho menor que las distancia al resto de los patrones de otras clases, esto se puede observar en el histograma. En un mal clasificador, aún si el sistema atinó a identificar correctamente al patrón, resulta que la distancia al patron de otra clase que quedo en segundo lugar quedó peligrosamente cerca y estuvo a punto de hacer quedar mal al clasificador

1.2. Aprendizaje Supervisado

Existen diferentes formas de realizar el aprendizaje supervisado. Este tipo de aprendizaje necesita un profesor que mida el rendimiento del sistema.

- Aprendizaje por corrección de error: El entrenamiento consiste en presentar al sistema un conjunto de pares de datos, representando la salida y la entrada para dicha entrada. Este conjunto recibe el nombre de conjunto de entrenamiento. El objetivo consiste en minimizar el error entre la salida deseada y la actual, es un aprendizaje off-line

- Aprendizaje por refuerzo: Es más lento que el anterior y no se conoce la salida exacta para cada entrada sólo se conoce como debería ser el comportamiento general ante diferentes entradas. En este caso la función del supervisor es mas la de un crítico que la de un maestro. Es un aprendizaje on-line que produce una señal de refuerzo indicando el éxito o fracaso de la salida producida por la red. La red actualiza sus pesos en función de la señal de refuerzo que dado una acción tomada por el sistema es seguida de un estado satisfactorio dicha acción es reforzada y disminuida en caso contrario.

- Aprendizaje Estocástico Este aprendizaje consiste en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado y distribuciones de probabilidad.

1.3. Aprendizaje no supervisado

Aprendizaje no supervisado es un método de Aprendizaje Automático donde un modelo es ajustado a las observaciones. Se distingue del Aprendizaje supervisado por el hecho de que no hay un conocimiento a priori. El aprendizaje no supervisado toma como entrada un conjunto de datos y los

trata como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

El aprendizaje no supervisado puede ser usado junto con la Inferencia bayesiana para producir probabilidades condicionales (es decir, aprendizaje supervisado) para cualquiera de las variables aleatorias dadas. El Santo Grial del aprendizaje no supervisado es la creación de un código factorial de los datos, esto es, un código con componentes estadísticamente independientes. El aprendizaje supervisado normalmente funciona mucho mejor cuando los datos iniciales son primero traducidos en un código factorial.

El aprendizaje no supervisado también es útil para la compresión de datos: fundamentalmente, todos los algoritmos de compresión dependen tanto explícita como implícitamente de una distribución de probabilidad sobre un conjunto de entrada.

Otra forma de aprendizaje no supervisado es la agrupación (en inglés, clustering), el cual a veces no es probabilístico.

1.3.1. Agrupamiento de datos

1.4. Reforzamiento del Aprendizaje

El objetivo del aprendizaje por refuerzo es usar el premio-castigo para aprender una función, la cual permitirá tomar decisiones en el futuro de qué acción tomar a partir de una percepción del entorno. La función de agente utiliza la información contenida en él para realizar la toma de decisiones. De ahí el nombre de Aprendizaje por Refuerzo. Existen, no obstante, otros formalismos para aprender, mediante refuerzo, qué acción realizar en cada caso, como por ejemplo las Redes Neuronales. Este método de aprendizaje surge de una rama de estudios de psicología experimental, que pueden remontarse a las experiencias de Pavlov con el refuerzo condicionado, y por otro lado es heredero de los métodos de control óptimo que se originan a partir de los trabajos de Bellman. Dicho de forma breve, el aprendizaje por refuerzo es el problema de conseguir que un agente actúe en un entorno de manera que maximize la recompensa que obtiene por sus acciones. Este tipo de aprendizaje se encuadra en los denominados como aprendizaje supervisado

La señal de refuerzo puede ser inmediata o retardada. Inmediata es cuando se obtiene una crítica para cada acción efectuada justo después de su realización. La información aportada por el refuerzo en este caso es local a

cada acción tomada. Por el contrario, en el caso del refuerzo retardado se dará cuando éste no se obtiene inmediatamente después de la realización de cada acción, sino al completar la secuencia de acciones empleadas para resolver el problema. En este caso, el refuerzo obtenido es una estimación global del comportamiento.

Una condición para poder aplicar el aprendizaje por refuerzo es que éste sea modelizable mediante cadenas de Markov: la acción a escoger en una situación dada depende únicamente de esta situación y no del camino que se ha realizado para llegar a ella. Definimos al agente como el aprendiz encargado de observar su entorno para recoger información que le permita modificar su comportamiento para así aprender a resolver un determinado problema. Como dijimos anteriormente, el objetivo del aprendizaje por refuerzo es la utilización de las recompensas para la obtención de una función de agente. Por tanto nuestro agente será una función que, recibiendo como entrada una percepción del entorno, devolverá la acción siguiente a realizar.

Las aplicaciones del Aprendizaje por Refuerzo son múltiples, desde robots móviles que aprenden a salir de un laberinto, programas de ajedrez que aprenden cuáles son las mejores secuencias de movimientos para ganar un juego o un brazo robótico que aprende cómo mover las articulaciones para lograr el movimiento final deseado.

1.5. Clasificador Lineal Vs No-lineal

Si los patrones pueden separarse por una linea recta discriminante como en la Figura 1.2, entonces podemos implementar un *reconocedor lineal*. Si se requiere de una linea curva para separar las clases como en la Figura 1.3 entonces debemos implementar un reconocedor no-lineal o implementar un reconocedor lineal y aceptar que se equivoque de vez en cuando.

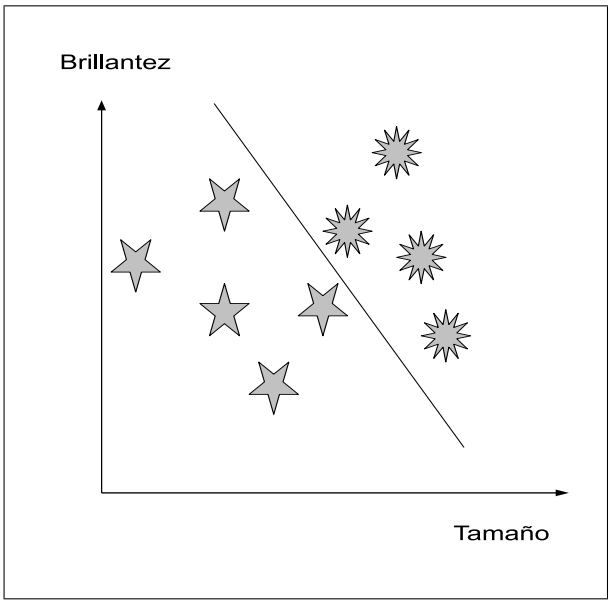

Figura 1.2: Discriminante lineal 2D

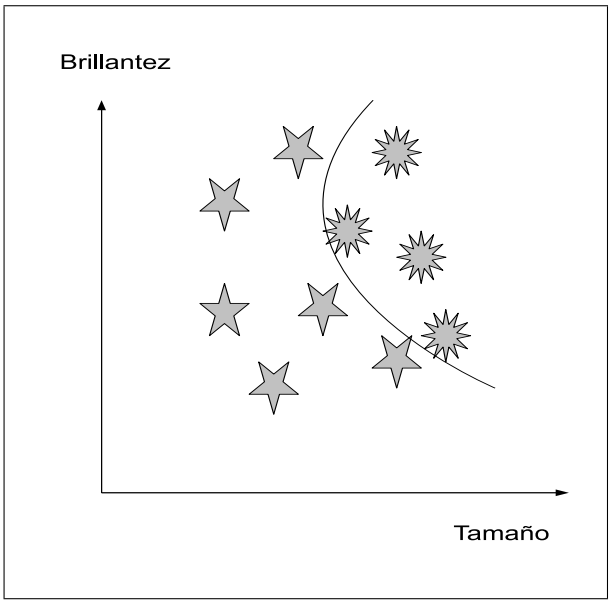

Figura 1.3: Discrimante no-lineal 2D

Capítulo 2

El clasificador Bayesiano

Suponga que tenemos solo dos clases, sean estas ω_1 y ω_2, si quisiéramos predecir a cual de ellas pertenecerá el siguiente patrón que se presente solamente conociendo la historia, es decir, la frecuencia con han aparecido los patrones de una y otra clase, entonces lo mejor que podríamos hacer es usar la *Primera regla de decision*.

La mas simple regla de decision nos indica que debemos optar por la clase ω_1 si $P(\omega_1) > P(\omega_2)$, en caso contrario debemos optar por ω_2. $P(\omega_1)$ es la probabilidad *a priori* de que la clase sea ω_1 y $P(\omega_2)$ es la probabilidad *a priori* de que la clase sea ω_2. Tanto $P(\omega_1)$ como $P(\omega_2)$ son estimadas como la proporción de clases que se han presentado hasta el momento, por ejemplo si han aparecido 80 % de patrones de la clase ω_1 y 20 % de patrones de la clase ω_2 podemos suponer que $P(\omega_1) = 0{,}8$ y que $P(\omega_2) = 0{,}2$. Por supuesto, con tan poca información la probabilidad de equivocarnos es bastante alta, en el peor de los casos si se han presentado tantos patrones de la clase ω_1 como de la ω_2, entonces $P(\omega_1) = P(\omega_2) = 0{,}5$ y nos estaremos equivocando la mitad de las veces que hagamos nuestra predicción y sin embargo es lo mejor que podemos hacer.

Normalmente no tenemos que decidir de que clase será el siguiente patrón que se presente. El problema mas común en reconocimiento de patrones es el que plantea que dado un nuevo patrón x debemos decidir a cual clase pertenece. Para ello podemos determinar $P(\omega_j|x)$, es decir, la probabilidad *a posteriori* de que pertenezca a la clase ω_j puesto que contamos con una *evidencia* (características del patrón x). Para resolver esto hacemos uso de la *formula de Bayes* (2.1)

$$P(\omega_j|x) = \frac{P(x|\omega_j)P(\omega_j)}{P(x)} \tag{2.1}$$

Donde, $P(x|\omega_j)$ es la denominada *Probabilidad condicionada a la clase*, es decir, la probabilidad de que un patrón de la clase ω_j tenga las características x, a esto le llamamos *verosimilitud*, es decir lo creíble (verosimil) que es que un objeto de la clase ω_j tenga las características x, si $P(x|\omega_j)$ es grande (cercano a 1.0) entonces x es común en la clase ω_j, por ejemplo, $P(1{,}2|pigmeo) = 0{,}9$ puede indicarnos que es muy frecuente (90 %) que la estatura de un adulto pigmeo sea de 1.2m, se trata de una estatura muy creíble (verosimil).

En la ecuación (2.1), $P(x)$ es la probabilidad *a priori* de que el patrón que se presente tenga las características x, a esto le podemos también llamar *evidencia*. $P(\omega_j)$ es la probabilidad *a priori* de que se presente un patron de la clase ω_j, entonces la ecuación (2.1) se puede entender como se plantea en la ecuación (2.2)

$$Posteriori = \frac{Verosimilitud * Priori}{evidencia} \tag{2.2}$$

La fórmula conocida como de la *probabilidad total* nos dice que $P(x)$ se puede determinar mediante (2.3)

$$P(x) = \sum_{i=i}^{c} P(x|\omega_i)P(\omega_i) \tag{2.3}$$

Donde c es el número de clases

La regla de decision de Bayes para el caso mas simple de solo dos clases ($c = 2$) nos indica que debemos optar por la clase ω_1 si $P(\omega_1|x) > P(\omega_2|x)$ y por ω_2 en caso contrario. Fácilmente podemos generalizar esta regla para el caso de c clases como :

Si $P(\omega_i|x) < P(\omega_j|x)\ \forall j \neq i$ decide ω_i

En la Figura 2.1 se pueden observar dos curvas, cada una corresponde a una clase y para cada valor de x cada curva reporta la probabilidad de que x se presente en la clase correspondiente, por eso tambien se le conoce como *probabilidad condicionada a la clase*. En la Figura 2.2 se muestran las probabilidades a posteriori correspondientes a las verosimilitudes de la Figura 2.1, para cada x, la suma de las probabilidades a posteriori es igual a 1.0, las

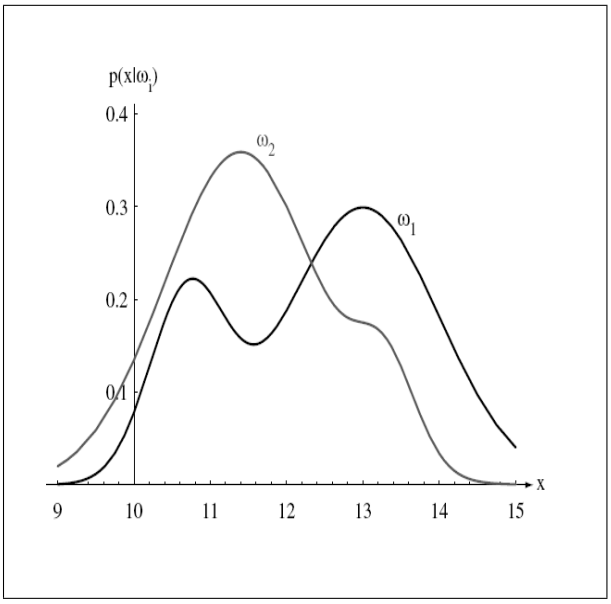

Figura 2.1: Verosimilitudes de dos clases, para cada valor de x cada curva reporta la probabilidad de que x se presente en la clase correspondiente

regiones en las que $P(\omega_1|x) > P(\omega_2|x)$ corresponden a la clase ω_1 y el resto a la clase ω_2.

2.1. Clasificador Bayesiano basado la minimización del régimen de errores

Un clasificador toma decisiones o ejecuta acciones, una posible decisión es que el patrón no corresponde con ninguna clase conocida. Las decisiones tienen consecuencias que se pueden modelar como costos, después de todo una decision puede ser mas costosa que otra. Una *función de pérdidas* nos permite asociar una decisión a una pérdida, incluso cuando el clasificador no reconoce al patrón como perteneciente a ninguna clase. Podríamos usar la función de pérdidas para optar por un clase que signifique una menor pérdida.

Sea $\lambda(\alpha_i|\omega_j)$ la pérdida absorbida al ejecutar la acción α_i cuando la clase a la que pertenece el patron es ω_j

Considere a $\mathbf{x}$ ahora no como un patrón sino como un vector de características de un espacio de características de dimensión D, es decir $\mathbf{x}\epsilon\Re^{\mathbf{D}}$. La

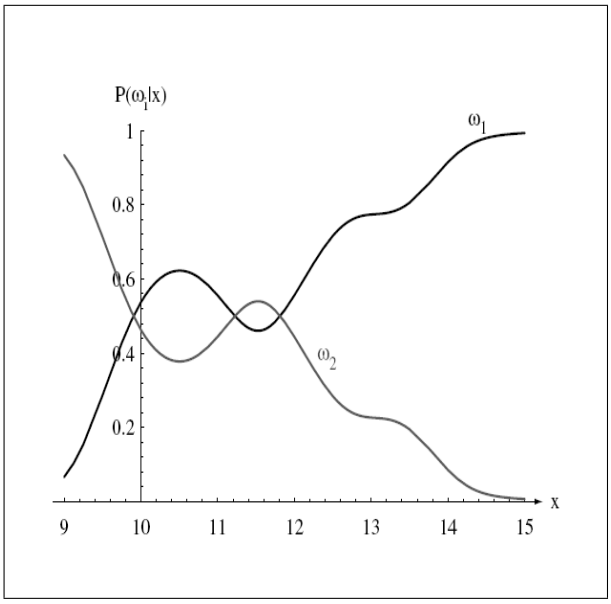

Figura 2.2: Probabilidades a posteriori correspondientes a las verosimilitudes de la Figura 2.1 para las probabilidades a priori $P(\omega_1) = 2/3$ y $P(\omega_2) = 1/3$

fórmula de Bayes es entonces (2.4).

$$P(\omega_j|\mathbf{x}) = \frac{\mathbf{P}(\mathbf{x}|\omega_\mathbf{j})\mathbf{P}(\omega_\mathbf{j})}{\mathbf{P}(\mathbf{x})} \tag{2.4}$$

El *Riesgo condicional* de tomar la acción α_i dado el patrón con características $\mathbf{x}$ de acuerdo a la ecuación (2.5) es la suma de todos los costos asociados con tomar la acción α_i para cada clase por la probabilidad de cada clase dado el patrón con características $\mathbf{x}$.

$$R(\alpha_i|\mathbf{x}) = \sum_{\mathbf{j=1}}^{\mathbf{c}} \lambda(\alpha_\mathbf{i}|\omega_\mathbf{j})\mathbf{P}(\omega_\mathbf{j}|\mathbf{x}) \tag{2.5}$$

El *Riesgo total* o simplemente *Riesgo* es aquel que integra todos los riesgos asociados a tomar la decisión que corresponde a todos los posibles valores de $\mathbf{x}$ por las probabilidades de que dichos $\mathbf{x}$ ocurran, la ecuación (2.6) define formalmente el Riesgo.

$$R = \int R(\alpha(\mathbf{x})|\mathbf{x})\mathbf{P}(\mathbf{x}) \tag{2.6}$$

Donde $\alpha(\mathbf{x})$ es la *función de decisión* que dado un vector de características de un patron $\mathbf{x}$ regresa una acción de entre el conjunto de posibles acciones $\alpha_1, \alpha_2, ...\alpha_a$, a es el número de posibles acciones.

Para minimizar el Riesgo se procede de la siguiente manera:

1. Calcula el Riesgo Condicional $R(\alpha_i|\mathbf{x})$ para $i = 1, 2, ..., a$
2. Elige la acción α_j para la cual $R(\alpha_j|\mathbf{x})$ es menor

Denominemos a la pérdida absorbida al elegir a la clase ω_i cuando la verdadera clase es la ω_j como $\lambda_{i,j}$ es decir:

$$\lambda_{i,j} = \lambda(\alpha_i|\omega_j) \tag{2.7}$$

Entonces de acuerdo a (2.5)

$$R(\alpha_1|\mathbf{x}) = \lambda_{\mathbf{1},\mathbf{1}}\mathbf{P}(\omega_\mathbf{1}|\mathbf{x}) + \lambda_{\mathbf{1},\mathbf{2}}\mathbf{P}(\omega_\mathbf{2}|\mathbf{x}) \tag{2.8}$$

$$R(\alpha_2|\mathbf{x}) = \lambda_{\mathbf{2},\mathbf{1}}\mathbf{P}(\omega_\mathbf{1}|\mathbf{x}) + \lambda_{\mathbf{2},\mathbf{2}}\mathbf{P}(\omega_\mathbf{2}|\mathbf{x}) \tag{2.9}$$

La regla de decisión del menor riesgo nos dicta que optemos por ω_1 si $R(\alpha_1|\mathbf{x}) < \mathbf{R}(\alpha_\mathbf{2}|\mathbf{x})$ y por ω_2 en caso contrario, sustituyendo (2.8) y (2.9) en esta desigualdad obtenemos

$$\lambda_{1,1}P(\omega_1|\mathbf{x}) + \lambda_{\mathbf{1},\mathbf{2}}\mathbf{P}(\omega_\mathbf{2}|\mathbf{x}) < \lambda_{\mathbf{2},\mathbf{1}}\mathbf{P}(\omega_\mathbf{1}|\mathbf{x}) + \lambda_{\mathbf{2},\mathbf{2}}\mathbf{P}(\omega_\mathbf{2}|\mathbf{x}) \tag{2.10}$$

Agrupando las probabilidades a posteriori obtenemos:

$$P(\omega_1|\mathbf{x})(\lambda_{\mathbf{1},\mathbf{1}} - \lambda_{\mathbf{2},\mathbf{1}}) < \mathbf{P}(\omega_\mathbf{2}|\mathbf{x})(\lambda_{\mathbf{2},\mathbf{2}} - \lambda_{\mathbf{1},\mathbf{2}}) \tag{2.11}$$

Es evidente que el costo de equivocarse es mayor que el costo de acertar, por lo tanto tanto $(\lambda_{2,2} - \lambda_{1,2})$ como $(\lambda_{1,1} - \lambda_{2,1})$ son cantidades negativas, eso explica el cambio de signo en la desigualdad, para obterner:

$$\frac{P(\omega_1|\mathbf{x})}{P(\omega_2|\mathbf{x})} > \frac{\lambda_{2,2} - \lambda_{1,2}}{\lambda_{1,1} - \lambda_{2,1}} \tag{2.12}$$

Usando la regla de Bayes y la regla de la herradura esta desigualdad se convierte en:

$$\frac{P(\mathbf{x}|\omega_\mathbf{1})\mathbf{P}(\omega_\mathbf{1})}{P(\mathbf{x}|\omega_\mathbf{2})\mathbf{P}(\omega_\mathbf{2})} > \frac{\lambda_{1,2} - \lambda_{2,2}}{\lambda_{2,1} - \lambda_{1,1}} \tag{2.13}$$

Finalmente:

$$\frac{P(\mathbf{x}|\omega_\mathbf{1})}{P(\mathbf{x}|\omega_\mathbf{2})} > \frac{(\lambda_{1,2} - \lambda_{2,2})P(\omega_2)}{(\lambda_{2,1} - \lambda_{1,1})P(\omega_1)} \tag{2.14}$$

La parte izquierda de la desigualdad (2.14) es una razón de verosimilitudes mientras que la parte derecha es una cantidad que ¡no depende de **x**!

Así pues, la regla de decisión del menor riesgo nos dice:

Decide ω_1 si la razón de verosimilitudes $\frac{P(\mathbf{x}|\omega_\mathbf{1})}{P(\mathbf{x}|\omega_\mathbf{2})}$ es mayor que un umbral que no depende de **x**. En la Figura 2.3 se graficó la razón de verosimilitudes para los posibles valores de **x**, para el umbral θ_a el clasificador determina que las características **x** corresponden a ω_1 en las regiones indicadas por R_1 ya que en estas la curva (razón de verosimilitudes) reporta un valor mayor que el umbral. Si el umbral se aumenta a θ_b las regiones R_1 se harían mas estrechas y las regiones R_2 que corresponden a la clase ω_2 se ensancharían, eso se debe a que se estaría penalizando mas el clasificar erroneamente a ω_2 como ω_1 que al reves ($\lambda_{1,2} > \lambda_{2,1}$)

Si simplemente contamos el número de veces que el clasificador se equivoca y buscamos minimizar ese valor, bueno, tal enfoque se puede obtener si se elige como función de costo a (2.15)

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & if\ i = j \\ 1 & if\ i \neq j \end{cases} \tag{2.15}$$

Observe que todos los errores cuestan lo mismo, en este caso, el Riesgo corresponde con la probabilidad de equivocarse como lo plantea la ecuación (2.16)

$$R(\alpha_i|\mathbf{x}) = \sum_{\mathbf{j=1}}^{\mathbf{c}} \lambda(\alpha_\mathbf{i}|\omega_\mathbf{j})\mathbf{P}(\omega_\mathbf{j}|\mathbf{x}) \tag{2.16}$$

Dada la función de costo (2.15), entonces (2.16) se convierte en

$$R(\alpha_i|\mathbf{x}) = \sum_{\mathbf{j}\neq\mathbf{i}} \mathbf{P}(\omega_\mathbf{j}|\mathbf{x}) \tag{2.17}$$

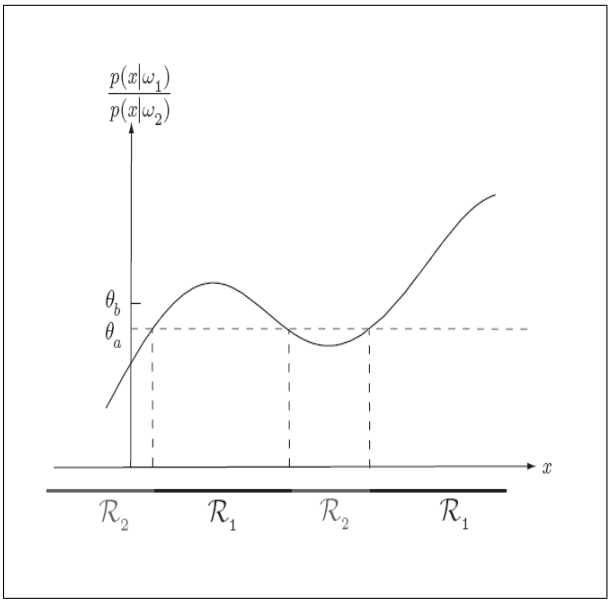

Figura 2.3: Gráfica de razon de verosimilitudes para diferentes valores de $\mathbf{x}$ y las regiones que define un determinado umbral

Que de acuerdo a la regla de complementación de teoría de probabilidad nos conduce a:

$$R(\alpha_i|\mathbf{x}) = \mathbf{1} - \mathbf{P}(\omega_\mathbf{i}|\mathbf{x}) \tag{2.18}$$

Por lo tanto, para minimizar el régimen de errores hay que optar por la mayor de las probabilidades a posteriori. La regla de decisión que minimiza el regimen de errores es:

Decide ω_i si $P(\omega_i|\mathbf{x}) > \mathbf{P}(\omega_\mathbf{j}|\mathbf{x})\ \forall j \neq i$

2.2. Criterio Minimax

Este criterio consiste en diseñar el clasificador de manera que se minimize el máximo Riesgo posible.

2.3. Funciones Discriminantes y Superficies de Decisión

2.3.1. Clasificador de mas de 2 clases

Es conveniente representar conceptualmente a un clasificador como una colección de funciones discriminantes, una para cada clase, de esta manera, el clasificador asigna $\mathbf{x}$ a la i-ésima clase si y solo si la función asociada a esa clase $g_i(x)$ reporta un valor mas grande que el resto de las funciones asociadas a las demás clases, es decir si se cumple:

$$g_i(x) > g_j(x) \; \forall \; i \neq j \tag{2.19}$$

Esta forma de ver a un clasificador se puede considerar como una generalización de los clasificadores, por ejemplo, para un clasificador Bayesiano, el cual está basado en la minimización del Riesgo:

$$g_i(x) = -R(\alpha_i|x) \tag{2.20}$$

También, para el clasificador que minimiza el régimen de errores:

$$g_i(x) = P(\omega_i|x) \tag{2.21}$$

Se puede reemplazar $g_i(x)$ por $f(g_i(x))$ siempre y cuando f sea una función monotónicamente creciente.

Al igual que las reglas de Decisión, las funciones discriminantes dividen al espacio de características en regiones de la siguiente manera:

Si $g_i(x) > g_j(x) \; \forall \; i \neq j$, entonces x está en la región R_i la cual es la región correspondiente a la clase ω_i

Las regiones están separadas por *Fronteras de decisión*. En la Figura 2.4, se pueden observar las funciones discriminantes que basadas en las probabilidades a posteriori siguiendo una distribución Normal (Gaussiana) definen fronteras de decisión en el espacio de características que en este caso es un plano bidimensional (el piso) por tratarse de dos características.

2.3.2. El Dicotomizador

Un dicotomizador es un clasificador de solo dos clases y utiliza solo una función $g(x)$ la cual se define como:

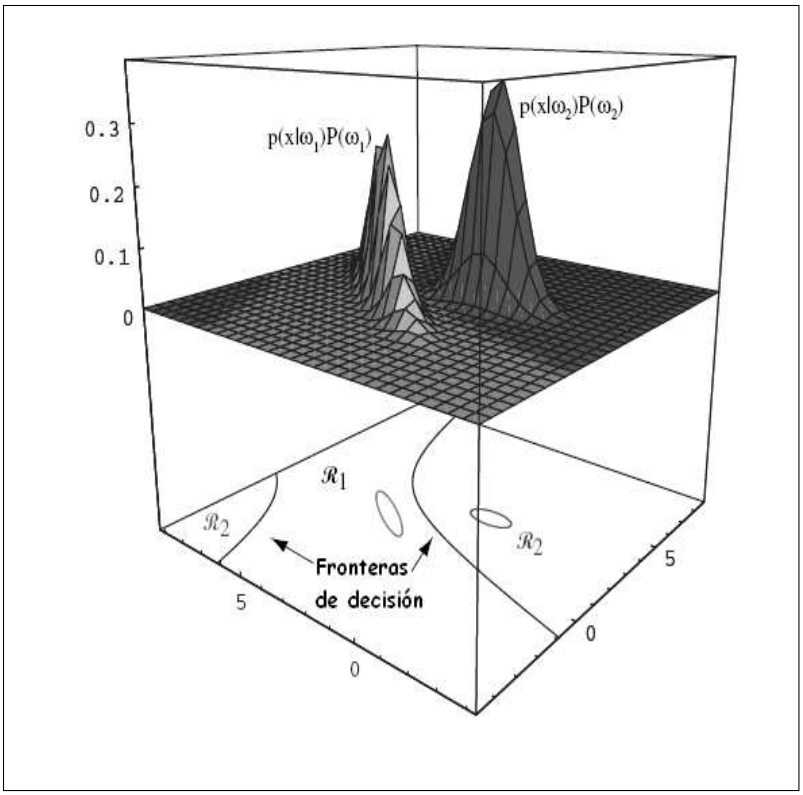

Figura 2.4: Regiones de Decisión de un clasificador de dos clases para un espacio de características bidimensional

$$g(x) = g_1(x) - g_2(x) \tag{2.22}$$

EL dicotomizador opta por la clase ω_1 si y solo si $g(x) > 0$, ahora bien si nos basamos en la minimización del régimen de errores tendremos que:

$$g(x) = P(\omega_1|x) - P(\omega_2|x) \tag{2.23}$$

Aplicando la regla de Bayes y la regla de la herradura:

$$g(x) = \frac{P(x|\omega_1)P(\omega_1)}{P(x)} - \frac{P(x|\omega_2)P(\omega_2)}{P(x)} \tag{2.24}$$

Como se dijo anteriormente, cualquier función f monotónicamente creciente se puede usar para reemplazar $f(g(x))$ por $g(x)$, en este caso nos conviene deshacernos de $P(x)$ puesto que es una cantidad que afecta por igual al minuendo y al sustraendo (no aporta nada para efecto de averiguar el signo), tambien nos conviene utilizar a la función logaritmo natural para aprovechar en el siguiente paso las leyes de los logaritmos, por estos motivos es que (2.24) se convierte en (2.25):

$$g(x) = ln(P(x|\omega_1)P(\omega_1)) - ln(P(x|\omega_2)P(\omega_2)) \tag{2.25}$$

Que gracias a las leyes de los logaritmos se reduce a:

$$g(x) = ln\Big(\frac{P(x|\omega_1)P(\omega_1)}{P(x|\omega_2)P(\omega_2)}\Big) \tag{2.26}$$

Y luego a:

$$g(x) = ln\Big(\frac{P(x|\omega_1)}{P(x|\omega_2)}\Big) + ln\Big(\frac{P(\omega_1)}{P(\omega_2)}\Big) \tag{2.27}$$

2.4. Funciones discriminantes para la densidad Normal

La clasificación para minimizar el régimen de errores puede lograrse usando la colección de funciones discriminantes:

$$g_i(x) = ln(P(x|\omega_i) + ln(P(\omega_i)) \tag{2.28}$$

La función de densidad Normal (Gaussiana) tiene la forma:

$$P(x) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^{\mathbf{t}}\mathbf{\Sigma^{-1}}(\mathbf{x}-\mu)} \tag{2.29}$$

Sustituyendo (2.29) en (2.28) y reduciendo obtenemos:

$$g_i(x) = -\frac{1}{2}(\mathbf{x}-\mu_\mathbf{i})^{\mathbf{t}}\mathbf{\Sigma_i^{-1}}(\mathbf{x}-\mu_\mathbf{i}) - \frac{\mathbf{d}}{\mathbf{2}}\mathbf{ln}(\mathbf{2}\pi) - \frac{\mathbf{1}}{\mathbf{2}}\mathbf{ln}|\mathbf{\Sigma_i}| + \mathbf{ln}(\mathbf{P}(\omega_\mathbf{i})) \tag{2.30}$$

2.4.1. Caso Matriz de Co-varianzas diagonal y única

Consideremos primero el caso mas simple en el que las características son variables independientes y tienen la misma varianza, es decir:

$$\Sigma_i = \sigma^2\mathbf{I} \tag{2.31}$$

Se cumple entonces que $|\Sigma_i| = \sigma^{2d}$ y que $\Sigma^{-1} = \frac{1}{\sigma^2}\mathbf{I}$, además, el segundo y tercer términos de (2.30) no dependen de i (No dependen de la clase), entonces son simples constantes aditivas y por tanto pueden ignorarse. Tomando todas estas consideraciones en cuenta, la ecuación (2.30) se transforma en:

$$g_i(x) = -\frac{1}{2\sigma^2}||\mathbf{x}-\mu_\mathbf{i}||^\mathbf{2} + \mathbf{ln}(\mathbf{P}(\omega_\mathbf{i})) \tag{2.32}$$

Para entender la función discriminante (2.32), observe que si $\mathbf{x}$ cae a igual distancia de los centroides de todas las clases (De toda μ_i), entonces las probabilidades a-priori ($P(\omega_i)$) sirven para desempatar.

En realidad no es necesario calcular las distancias del vector de características $\mathbf{x}$ a los centroides de las regiones de decisión μ_i como vamos a demostrar, desarrollando (2.32) obtenemos:

$$g_i(x) = -\frac{1}{2\sigma^2}(x^t x - 2\mu_i^t x + \mu^t\mu) + ln(P(\omega_i)) \tag{2.33}$$

Observe que el término $x^t x$ no depende de i (no depende de la clase) y por tanto se considera una constante aditiva y se puede eliminar, $g_i(x)$ adopta la forma:

$$g_i(x) = W_i^t x + w_{i0} \tag{2.34}$$

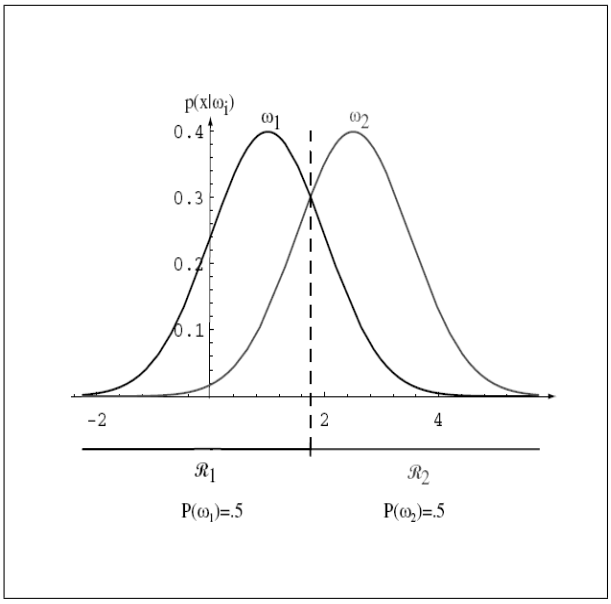

Figura 2.5: Regiones de Decisión definidas para el Caso I (Varianzas iguales) si son dos clases y el espacio de características es de una dimensión

Donde $W_i = -\frac{1}{\sigma^2}\mu_i$ y w_{i0} al que usamos como umbral de decisión es:

$$w_{i0} = -\frac{1}{2\sigma^2}\mu^t\mu + ln(P(\omega_i)) \tag{2.35}$$

Recuerde que el caso que se analiza en esta sección es aquel en el que todas las clases tienen una distribución de su verosimilitud Normal y sobre todo que TODOS TIENE LA MISMA VARIANZA. En las figuras 2.5, 2.6 y 2.7 se puede observar como se definen las regiones de decisión cuando el espacio de características es unidimensional, bidimensional y tridimensional respectivamente, en todas las figuras se manejan solo dos clases y sus correspondientes probabilidades a priori son también iguales (0.5) como consecuencia de ello, la frontera de decisión es una linea recta o un plano que divide el espacio justo a la mitad.

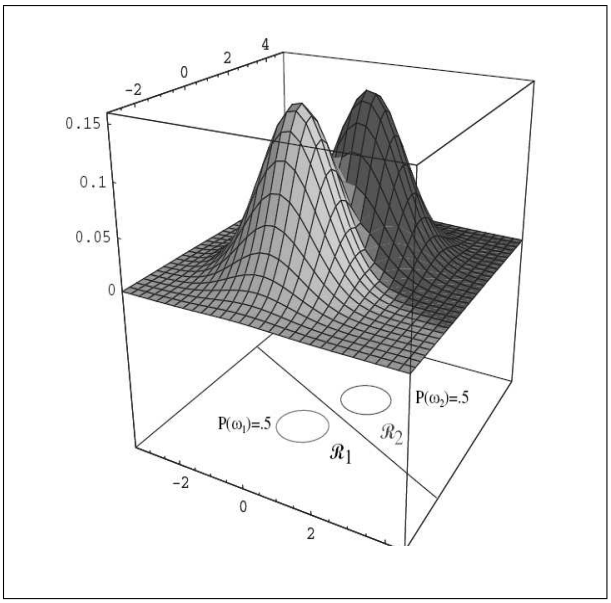

Figura 2.6: Regiones de Decisión definidas para el Caso I (Varianzas iguales) si son dos clases y el espacio de características es de 2 dimensiones

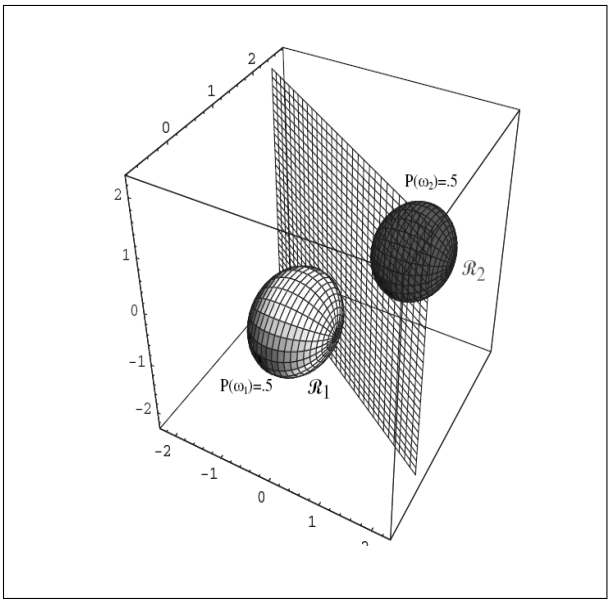

Figura 2.7: Regiones de Decisión definidas para el Caso I (Varianzas iguales) si son dos clases y el espacio de características es de 3 dimensiones

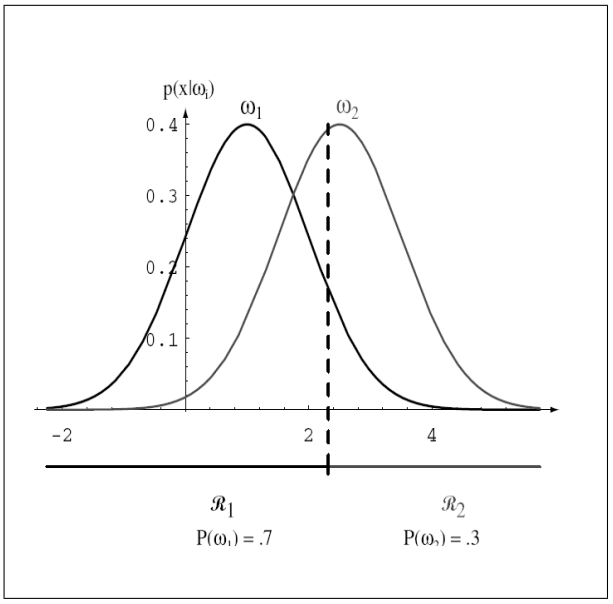

Figura 2.8: Regiones de Decisión definidas para el Caso I (Varianzas iguales) si son dos clases y el espacio de características es de una dimensión, las probabilidades a priori son diferentes

Si las probabilidades a priori no son iguales, las fronteras de decisión se moverán de manera que se alejen de la media de la distribución con mayor probabilidad a priori, de esta manera la región correspondiente a esa clase será mayor. En las figuras 2.8, 2.9 y 2.10se observa este efecto para espacios de características de una, dos y tres dimensiones respectivamente, en las mismas figuras se indican las probabilidades a priori de cada clase.

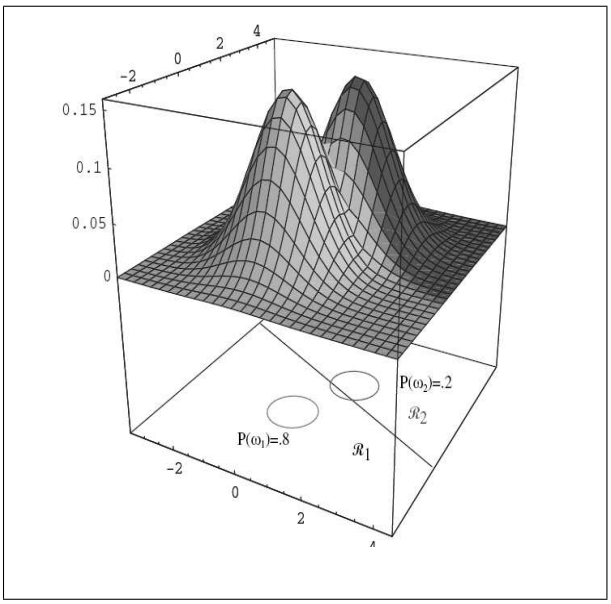

Figura 2.9: Regiones de Decisión definidas para el Caso I (Varianzas iguales) si son dos clases y el espacio de características es de 2 dimensiones, las probabilidades a priori son diferentes

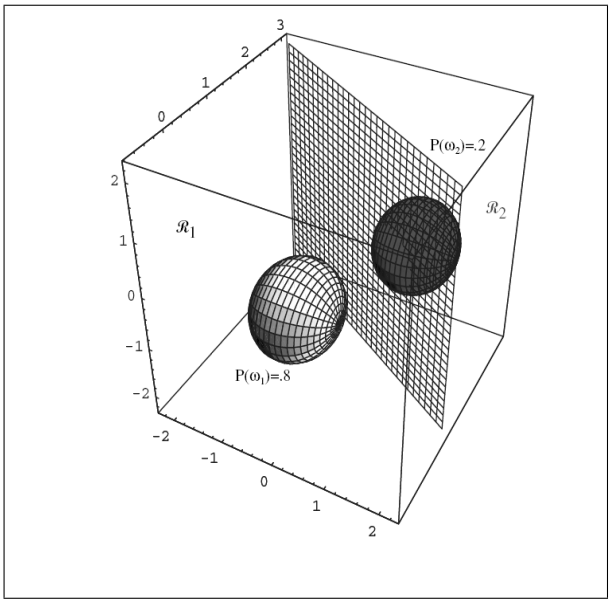

Figura 2.10: Regiones de Decisión definidas para el Caso I (Varianzas iguales) si son dos clases y el espacio de características es de 3 dimensiones, las probabilidades a priori son diferentes

Llamamos máquina lineal al clasificador que utiliza funciones discriminantes lineales, las superficies de decisión son pedazos de hiperplanos definidos por las ecuaciones lineales:

$$g_i(x) = g_j(x) \tag{2.36}$$

donde ω_i y ω_j son las clases con mayor probabilidad a priori.
La ecuación del hiperplano es:

$$w^t(x - x_0) = 0 \tag{2.37}$$

donde $w = \mu_i - \mu_j$ y x_0 es:

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{||\mu_i - \mu_j||^2} ln\Big(\frac{P(\omega_i)}{P(\omega_j)}\Big)(\mu_i - \mu_j) \tag{2.38}$$

Definida de esta forma, (2.37) es la ecuación de un hiperplano que paso por x_0 y es ortogonal al vector w que es el vector que une las medias. El hiperplano que separa las regiones R_i y R_j es ortogonal a la línea que une las medias μ_i y mu_j.

Analicemos ahora como afectan las probabilidades a-priori de las clases ω_i y ω_j. Primero, si $P(\omega_i) = P(\omega_i)$, entonces el segundo término de (2.38) se hace cero ya que $ln(1) = 0$ y entonces (2.38) se reduce a:

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) \tag{2.39}$$

Por lo tanto el hiperplano corta la línea que une las medias justo a la mitad

Si en cambio $P(\omega_i) \neq P(\omega_i)$, entonces x_0 (el punto donde el hiperplano corta a la línea que une a las medias) se aleja de la media mas probable (probabilidad a priori).

Si $P(\omega_i)$ es igual para toda i, es decir para todas las clases, entonces el término $P(\omega_i)$ se convierte en otra constante aditiva que puede ser ignorada y la regla óptima se convierte en:

1. Calcule $||x - \mu_i||$ para cada clase (cada i)

2. Asigne x a la media mas cercana

Observe que si σ^2 es pequeña comparada con $||\mu_i - \mu_j||$ entonces la posición de la frontera de decisión es prácticamente independiente de las probabilidades a priori y por tanto se puede aplicar este procedimiento.

A este clasificador se le conoce como de la *distancia mínima*, aquí la media es un representante de la clase, no es exactamente el método del *vecino mas cercano*.

2.4.2. Caso Matriz de Co-variancias única

En este caso todas las clases tienen la matriz de covarianzas, es decir:

$$\Sigma_i = \Sigma \tag{2.40}$$

Sin embargo, a diferencia del Caso I, las características no son variables independientes y por tanto la matriz de covarianzas Σ no es una matriz diagonal aunque puede serlo, si lo es caemos en el caso I. Así, el Caso II incluye al Caso I como caso particular.

Regresando a la función discriminante (2.30)donde asumimos que la verosimilitud tiene una distribución normal multivariable y que se escribe aquí de nuevo;

$$g_i(x) = -\frac{1}{2}(\mathbf{x} - \mu_\mathbf{i})^\mathbf{t}\mathbf{\Sigma^{-1}}(\mathbf{x} - \mu_\mathbf{i}) - \frac{\mathbf{d}}{\mathbf{2}}\mathbf{ln}(\mathbf{2}\pi) - \frac{\mathbf{1}}{\mathbf{2}}\mathbf{ln}|\mathbf{\Sigma}| + \mathbf{ln}(\mathbf{P}(\omega_\mathbf{i})) \tag{2.41}$$

De nuevo, el segundo y tercer términos de esta función son constantes aditivas y se pueden ignorar

$$g_i(x) = -\frac{1}{2}(\mathbf{x} - \mu_\mathbf{i})^\mathbf{t}\mathbf{\Sigma^{-1}}(\mathbf{x} - \mu_\mathbf{i}) + \mathbf{ln}(\mathbf{P}(\omega_\mathbf{i})) \tag{2.42}$$

Y de nuevo, Si $P(\omega_i) = P(\omega_j)\ \forall i, j$ el último termino también será una constante aditiva y también se ignorará:

$$g_i(x) = -\frac{1}{2}(\mathbf{x} - \mu_\mathbf{i})^\mathbf{t}\mathbf{\Sigma^{-1}}(\mathbf{x} - \mu_\mathbf{i}) \tag{2.43}$$

Reconocemos aquí a la *Distancia de Mahalanobis* entre x y μ_i que nos reporta que tan cerca de una distribución se encuentra un vector tomando en cuenta no solo su media sino las varianzas y covarianzas de dicha distribución.

Entonces, la regla de decisión para este caso consiste en:

1. Calcule la distancia de Mahalanobis $(\mathbf{x} - \mu_\mathbf{i})^\mathbf{t}\mathbf{\Sigma^{-1}}(\mathbf{x} - \mu_\mathbf{i})$ entre x y cada clase (cada i)

2. Asigne x a la clase mas cercana

Si no se cumple que $P(\omega_i) = P(\omega_j)\ \forall i, j$, entonces la decisión se carga hacia la categoría de mayor probabilidad a priori.

Desarrollemos la fórmula de la distancia de Mahalanobis

$$(\mathbf{x} - \mu_\mathbf{i})^\mathbf{t}\mathbf{\Sigma^{-1}}(\mathbf{x} - \mu_\mathbf{i}) = \mathbf{x^t\Sigma^{-1}x} - \mathbf{x^t\Sigma^{-1}}\mu_\mathbf{i} - \mu_\mathbf{i}^\mathbf{t}\mathbf{\Sigma^{-1}x} + \mu_\mathbf{i}^\mathbf{t}\mathbf{\Sigma^{-1}}\mu_\mathbf{i} \quad (2.44)$$

Observe que el primer término del lado derecho de esta ecuación no depende de i (de la clase), por lo tanto es una constante aditiva que puede ignorarse. Lo importante de eliminar este término es que se evidencia que EL CLASIFICADOR ES LINEAL, por tanto, la función discriminante $g_i(x)$ tiene la forma general del hiperplano, es decir:

$$g_i(x) = w_i^t x + w_{i0} \quad (2.45)$$

Donde $w_i = \Sigma^{-1}\mu_i$ y

$$w_{i0} = -\frac{1}{2}\mu_i^t\Sigma^{-1}\mu_i + ln(P(\omega_i)) \quad (2.46)$$

Las fronteras de decisión son de nuevo hiperplanos. Si las regiones R_i y R_j son contiguas el hiperplano es:

$$w^t(x - x_0) = 0 \quad (2.47)$$

Donde $w = \Sigma^{-1}(\mu_i - \mu_j)$ lo cual implica que el hiperplano (la frontera de decisión) no es generalmente ortogonal a la linea que une a las medias de las regiones contiguas y x_0 (el lugar donde se corta a la linea que une las medias) estaría dada por:

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{ln(P(\omega_i)/P(\omega_j))}{(\mu_\mathbf{i} - \mu_\mathbf{j})^\mathbf{t}\mathbf{\Sigma^{-1}}(\mu_\mathbf{i} - \mu_\mathbf{j})}(\mu_i - \mu_j) \quad (2.48)$$

La interpretación es que el hiperplano corta a la linea que une μ_i y μ_j si $P(\omega_i) = P(\omega_i)$ y en caso contrario el corte (x_0) se aleja de μ_i en la medida en que $P(\omega_i)$ es mayor que $P(\omega_i)$. En las figuras 2.11 y 2.13 podemos observar el plano que define las regiones para dos clases para un espacio de características

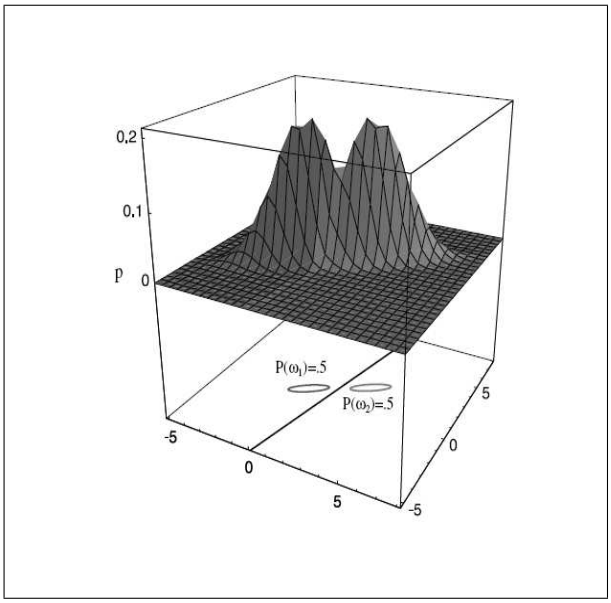

Figura 2.11: Regiones de Decisión definidas para el Caso II si son dos clases y el espacio de características es de 2 dimensiones, las probabilidades a priori son iguales

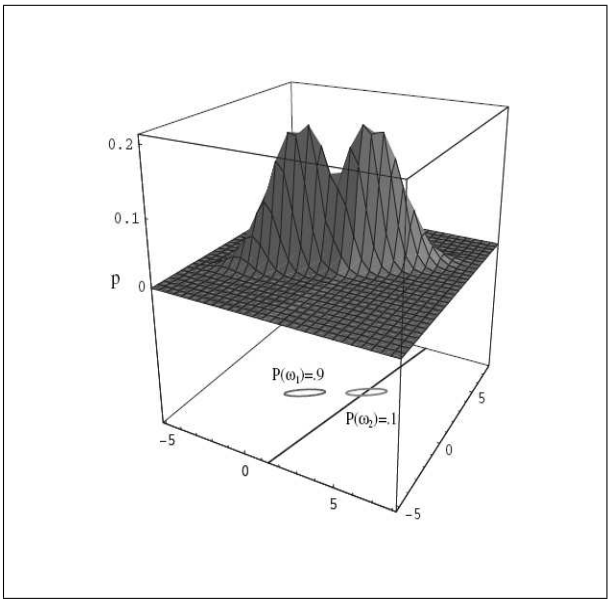

Figura 2.12: Regiones de Decisión definidas para el Caso II si son dos clases y el espacio de características es de 2 dimensiones, las probabilidades a priori son diferentes

bidimensional y tridimensional respectivamente cuando las probabilidades a priori son iguales, observe como el hiperplano no es ortogonal al vector que iría de la clase ω_1 a la clase ω_2. En las figuras 2.12 y 2.14 podemos observar el plano que define las regiones para dos clases para un espacio de características bidimensional y tridimensional respectivamente cuando las probabilidades a priori son diferentes, es claro que los hiperplanos se han movido alejándose de la media de la clase con mayor probabilidad a priori.

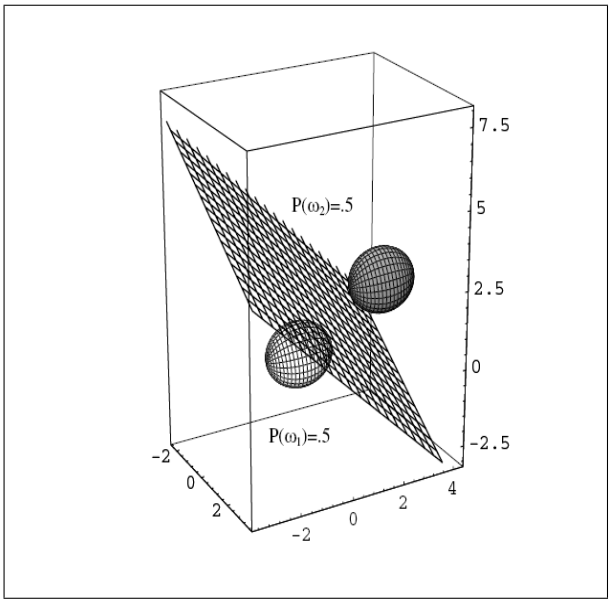

Figura 2.13: Regiones de Decisión definidas para el Caso II si son dos clases y el espacio de características es de 3 dimensiones, las probabilidades a priori son iguales

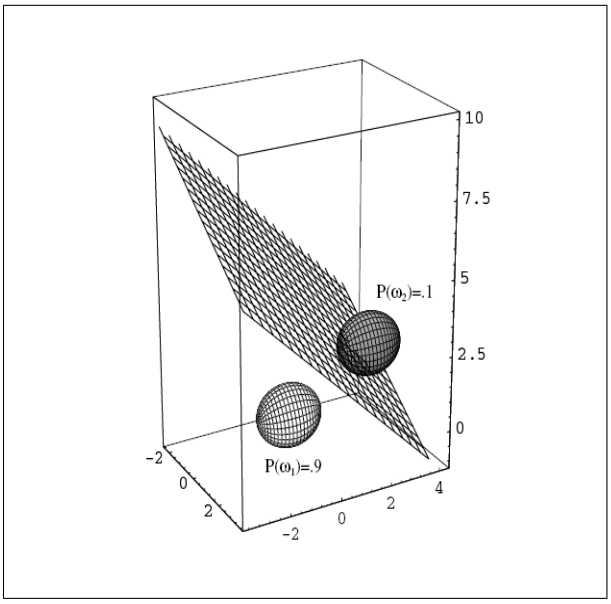

Figura 2.14: Regiones de Decisión definidas para el Caso II si son dos clases y el espacio de características es de 3 dimensiones, las probabilidades a priori son diferentes

2.4.3. Caso Matriz de Co-variancias arbitraria

En este caso, todas las clases tienen sus propias varianzas, el único término que se puede ignorar por ser una constante aditiva es $\frac{d}{2}ln(2\pi)$ y las funciones discriminantes son cuadráticas. La forma general de una cuadrática es:

$$g_i(x) = x^t W_i x + w_i^t x + w_{i0} \tag{2.49}$$

Donde: $W_i = -\frac{1}{2}\Sigma_i^{-1}$
$w_i = \Sigma_i^{-1}\mu_i$
y $w_{i0} = -\frac{1}{2}\mu_i^t\Sigma_i^{-1}\mu_i - \frac{1}{2}ln|\Sigma_i| + ln(P(\omega_i))$

Las superficies de decisión son hipercuadráticos, pueden ser hiperplanos, pares de hiperplanos, hiperesferas, hiperelipsoides o hiperparábolas. En las Figura 2.15 se puede observar el caso en el que la superficie de decisión es una hiperparabola. En la Figura 2.16 se puede observar el caso en el que la superficie de decisión es una hiperelipsoide. Observe en ambos casos que las distribuciones si bien son normales tienen distintas covarianzas.

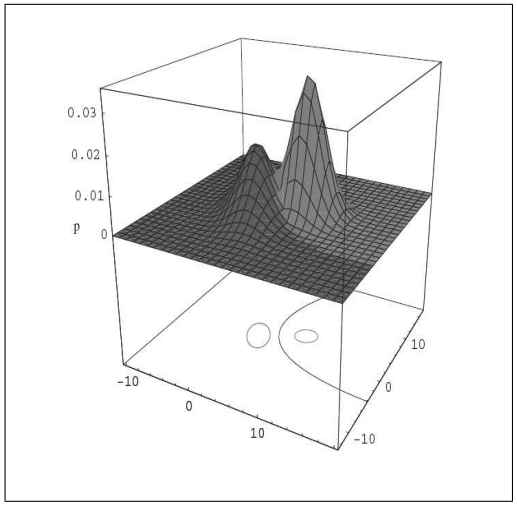

Figura 2.15: Regiones de Decisión definidas para el Caso III si son dos clases y el espacio de características es de 2 dimensiones, la superficie de decisión es un hiperparábola

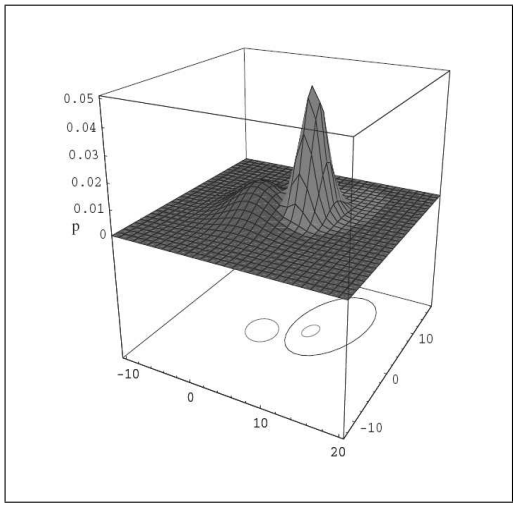

Figura 2.16: Regiones de Decisión definidas para el Caso III si son dos clases y el espacio de características es de 2 dimensiones, la superficie de decisión es un hiperelipsoide

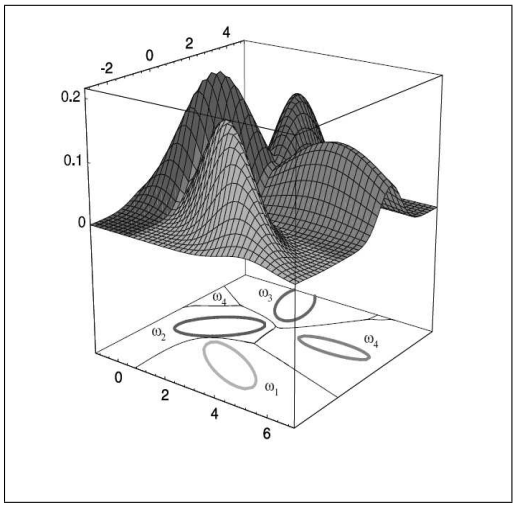

Figura 2.17: Regiones de Decisión definidas para el Caso III para cuatro clases y el espacio de características es de 2 dimensiones.

En la Figura 2.17 se observan las regiones de decisión para cuatro clases en un espacio de características bidimensional.

Como ejemplo considere los puntos (patrones) que se muestran en el espacio de características bidimensional de la Figura 2.18, los puntos que están arriba forman parte de la clase ω_1 y los puntos inferiores son de la clase ω_2. Utilizaremos como probabilidades a priori $P(\omega_1) = P(\omega_2) = 0{,}5$.

La colección de puntos (x_i, y_i) de la clase ω_1 forman el siguiente conjunto: $(3,4), (3,8), (2,6), (4,6)$ de donde:
$\mu_{1x} = (3+3+2+4)/4 = 3$
$\mu_{1y} = (4+8+6+6)/4 = 6$
La media de la clase ω_1 es entonces $\mu_1 = (3,6)$

$\sigma_{1xx} = \frac{1}{4}\sum_{i=1}^{4}(x_i - \mu_{1x})^2 = \frac{1}{4}[(3-3)^2 + (3-3)^2 + (2-3)^2 + (4-3)^2] = \frac{1}{2}$
$\sigma_{1xy} = \sigma_{1yx} = \frac{1}{4}\sum_{i=1}^{4}(x_i - \mu_{1x})(y_i - \mu_{1y}) =$
$\frac{1}{4}[(3-3)(4-6) + (3-3)(2-6) + (2-3)(6-6) + (4-3)(6-6)]=0$
$\sigma_{1yy} = \frac{1}{4}\sum_{i=1}^{4}(y_i - \mu_{1y})^2 = \frac{1}{4}[(4-6)^2 + (8-6)^2 + (6-6)^2 + (6-6)^2] = 2$

La Matriz de covarianzas de la clase ω_1 es entonces:

$$\Sigma_1 = \begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}$$

La colección de puntos (x_i, y_i) de la clase ω_2 forman el siguiente conjunto: $(3,0), (1,-2), (5,-2), (3,-4)$ de donde:
$\mu_{2x} = (3+1+5+3)/4 = 3$
$\mu_{2y} = (0-2-2-4)/4 = -2$
La media de la clase ω_2 es entonces $\mu_2 = (3,-2)$

$\sigma_{2xx} = \frac{1}{4}\sum_{i=1}^{4}(x_i - \mu_{2x})^2 = \frac{1}{4}[(3-3)^2 + (1-3)^2 + (5-3)^2 + (3-3)^2] = 2$
$\sigma_{2xy} = \sigma_{2yx} = \frac{1}{4}\sum_{i=1}^{4}(x_i - \mu_{2x})(y_i - \mu_{2y}) =$
$\frac{1}{4}[(3-3)(0-(-2)) + (1-3)(-2-(-2)) + (5-3)(-2-(-2)) + (3-3)(-4-(-2))]=0$
$\sigma_{2yy} = \frac{1}{4}\sum_{i=1}^{4}(y_i - \mu_{2y})^2 = \frac{1}{4}[(0-(-2))^2 + (-2-(-2))^2 + (-2-(-2))^2 + (-4-(-2))^2] = 2$

La Matriz de covarianzas de la clase ω_2 es entonces:

$$\Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

En vista de que las matrices de covarianzas son diferentes, es decir cumple

con el caso III ($\Sigma_1 \neq \Sigma_2$), entonces las funciones discriminantes tendrán la forma general:

$$g_i(x) = x^t W_i x + w_i^t x + w_{i0} \tag{2.50}$$

Para la clase ω_1, la función discriminante sería:

$$g_1(x) = x^t W_1 x + w_1^t x + w_{10} \tag{2.51}$$

Donde: $W_1 = -\frac{1}{2}\Sigma_1^{-1} = -\frac{1}{2}\begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}^{-1}$

$W_1 = -\frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 1/2 \end{bmatrix}$

Además $w_1 = \Sigma_1^{-1}\mu_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$

También: $w_{10} = -\frac{1}{2}\mu_1^t\Sigma_1^{-1}\mu_1 - \frac{1}{2}ln|\Sigma_1| + ln(P(\omega_1))$

$w_{10} = -\frac{1}{2}\begin{bmatrix} 3 & 6 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} 3 \\ 6 \end{bmatrix} - \frac{1}{2}ln(1) + ln(0{,}5) = -18 + ln(0{,}5)$

$w_{10} = -18{,}673147$

Sustituyendo W_1, w_1 y w_{10} en (2.51):

$$g_1(\mathbf{x}) = -\frac{\mathbf{1}}{\mathbf{2}}\mathbf{x^t}\begin{bmatrix} 2 & 0 \\ 0 & 1/2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 6 & 3 \end{bmatrix}\mathbf{x} - \mathbf{18{,}673147} \tag{2.52}$$

De manera similar, para la clase ω_2, la función discriminante sería:

$$g_2(x) = x^t W_2 x + w_2^t x + w_{20} \tag{2.53}$$

Donde: $W_2 = -\frac{1}{2}\Sigma_2^{-1} = -\frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}^{-1}$

$W_2 = -\frac{1}{2}\begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}$

Además $w_2 = \Sigma_1^{-1}\mu_2 = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 3/2 \\ -1 \end{bmatrix}$

También: $w_{20} = -\frac{1}{2}\mu_2^t\Sigma_2^{-1}\mu_2 - \frac{1}{2}ln|\Sigma_2| + ln(P(\omega_2))$

$$w_{20} = -\tfrac{1}{2}\begin{bmatrix} 3 & -2 \end{bmatrix}\begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} 3 \\ -2 \end{bmatrix} - \tfrac{1}{2}ln(4) + ln(0{,}5) = -4{,}616$$

Sustituyendo W_2, w_2 y w_{20} en (2.53):

$$g_2(\mathbf{x}) = -\frac{\mathbf{1}}{\mathbf{2}}\mathbf{x^t}\begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 3/2 & -1 \end{bmatrix}\mathbf{x} - \mathbf{4{,}616} \tag{2.54}$$

Haciendo $g_1(\mathbf{x}) = \mathbf{g_2}(\mathbf{x})$

$$-\tfrac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 6 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 18{,}673147 =$$

$$= -\tfrac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 3/2 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 4{,}616$$

Reduciendo

$$-\tfrac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2x_1 \\ \frac{1}{2}x_2 \end{bmatrix} + 6x_1 + 3x_2 - 18{,}763 =$$

$$= -\tfrac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} \frac{1}{2}x_1 \\ \frac{1}{2}x_2 \end{bmatrix} + \tfrac{3}{2}x_1 - x_2 - 4{,}616$$

$$-x_1^2 - \tfrac{1}{4}x_2^2 + 6x_1 + 3x_2 - 18{,}763 = -\tfrac{1}{4}x_1^2 - \tfrac{1}{4}x_2^2 + \tfrac{3}{2}x_1 - x_2 - 4{,}616$$

$$(6 - \tfrac{3}{2})x_1 + (3-1)x_2 - 18{,}763 + 4{,}616 = x_1^2 - \tfrac{1}{4}x_1^2$$

$$4{,}5x_1 + 4x_2 - 14{,}057 = \tfrac{3}{4}x_1^2$$

$$x_2 = \tfrac{14{,}057}{4} - \tfrac{4{,}5}{4}x_1 + \tfrac{3}{16}x_1^2$$

Finalmente:

$$x_2 = 3{,}514 - 1{,}125x_1 + 0{,}1875x_1^2$$

Esta cuadrática describe la parábola con vértices en $(3, 1{,}83)$ que se muestra en la Figura 2.18 y que constituye la frontera de decisión que divide las Regiones de las dos clases de este ejemplo.

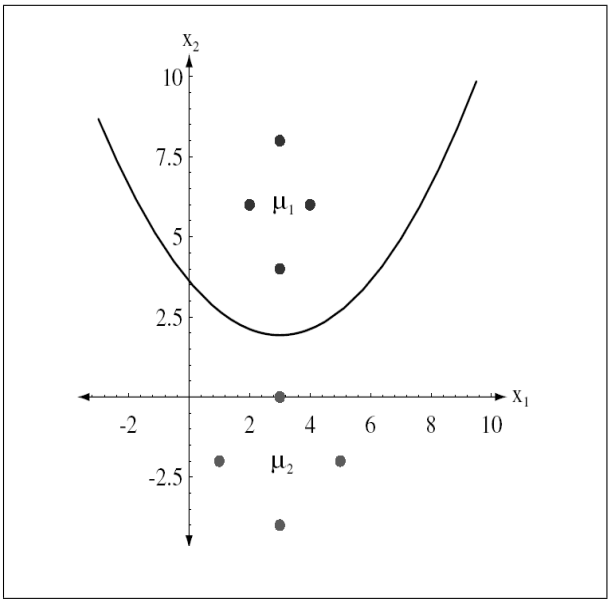

Figura 2.18: Ejemplo de patrones que pertenecen a dos clases, varianzas diferentes (caso III)

2.5. Acotamiento del error para densidades normales

Para un clasificador de solo dos clases (Un dicotomizador) solo hay dos maneras en que este puede incurrir en un error. La primera consiste en que el vector de características x quede dentro de la región R_1que el clasificador identifica como perteneciente a la clase ω_1 cuando el realidad x es un patrón de la clase ω_2. La segunda consiste en que el vector de características x quede dentro de la región R_2 que el clasificador identifica como perteneciente a la clase ω_2 cuando el realidad x es un patrón de la clase ω_1. De manera formal:

$$P(error) = P(x\epsilon R_1, \omega_2) + P(x\epsilon R_2, \omega_1) \tag{2.55}$$

$$P(error) = P(x\epsilon R_1|\omega_2)P(\omega_2) + P(x\epsilon R_2|\omega_1)P(\omega_1) \tag{2.56}$$

$$P(error) = \int_{R_1} P(x|\omega_2)P(\omega_2)dx + \int_{R_2} P(x|\omega_1)P(\omega_1)dx \tag{2.57}$$

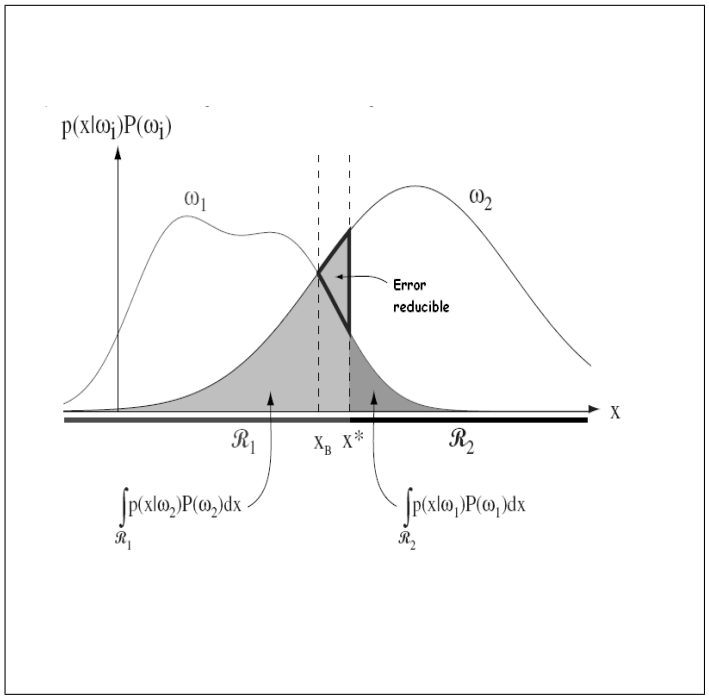

Figura 2.19: Zonas de error de un clasificador

En la Figura 2.20 se muestran las áreas de error y como estas dependen de la frontera de decisión, al mover dicha frontera es posible reducir el error.

Podemos generalizar la definición para c clases:

$$P(correcto) = \sum_{i=1}^{c} P(x \epsilon R_i, \omega_i) \tag{2.58}$$

$$P(coorecto) = \sum_{i=1}^{c} P(x \epsilon R_i | \omega_i) P(\omega_i) \tag{2.59}$$

$$P(correcto) = \sum_{i=1}^{c} \int_{R_i} P(x|\omega_i) P(\omega_i) dx \tag{2.60}$$

Por supuesto $P(error) = 1 - P(correcto)$

2.5.1. Cota de Chernoff

La regla de decisión de Bayes minimiza el régimen de errores pero no nos permite saber cual es ese error, sabemos sin embargo que ese error no puede ser mayor que la cota de error de Chernoff. Primero debemos entender la desigualdad:

$$min[a, b] \leq a^{\beta} b^{1-\beta} \tag{2.61}$$

Suponga que $a \geq b$, entonces la desigualdad (2.61) se convierte en:

$$b \leq b\Big[\frac{a}{b}\Big]^{\beta} \tag{2.62}$$

La cual obviamente se cumple ya que $\frac{a}{b}$ es un factor mayor a 1,0.
Sabemos que:

$$P(error) = \int min[P(x|\omega_1)P(\omega_1), P(x|\omega_2)P(\omega_2)]dx \tag{2.63}$$

aplicando la desigualdad (2.61):

$$P(error) \leq \int [P(x|\omega_1)P(\omega_1)]^{\beta}[P(x|\omega_2)P(\omega_2)]^{1-\beta}dx \tag{2.64}$$

Podemos sacar los factores que no dependen de la variable de integración

$$P(error) \leq [P(\omega_1)]^{\beta}P(\omega_2)]^{1-\beta} \int [P(x|\omega_1)]^{\beta}[P(x|\omega_2)]^{1-\beta}dx \tag{2.65}$$

Asumiendo gaussianidad:

$$P(error) \leq [P(\omega_1)]^{\beta}P(\omega_2)]^{1-\beta}e^{-k(\beta)} \tag{2.66}$$

donde $e^{-k(\beta)}$ es una función que encuentra su mínimo para $\beta = 0{,}66$ como se ve en la Figura 2.20, este valor nos conduce a la cota de Chernoff

2.5.2. Cota de Bhattacharyya

Para $\beta = 0{,}5$ la función $e^{-k(\beta)}$ tiene un valor bastante cercano al mínimo pero que nos facilita mucho los cálculos, este valor nos conduce a la cota de Bhattacharyya.

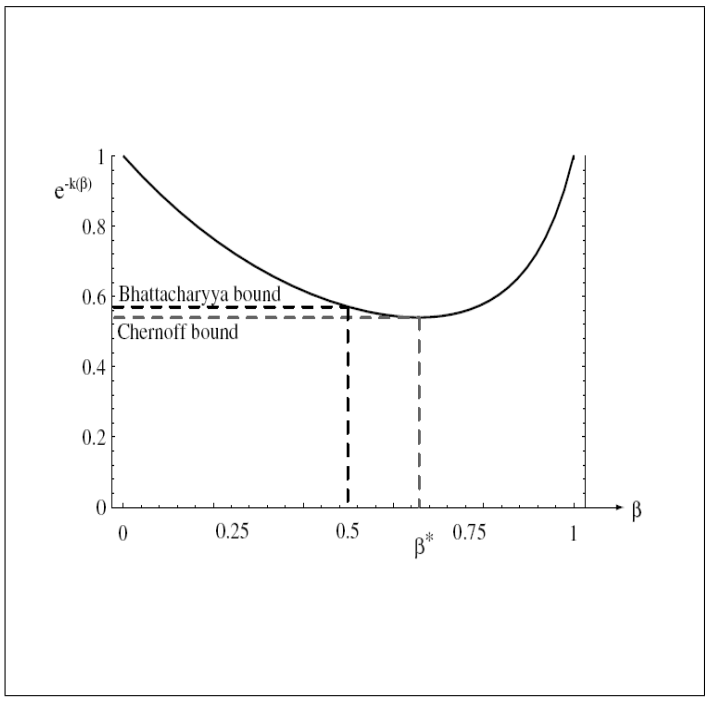

Figura 2.20: Valores de β que nos conducen a la cota de Chernoff (β^*) y a la de Bhattacharyya ($\beta = 0{,}5$)

Si $\beta = 0{,}5$ entonces:

$$P(error) \leq \sqrt{P(\omega_1)P(\omega_2)}e^{-k(0,5)} \tag{2.67}$$

$e^{-k(0,5)}$ se puede obtener fácilmente si se conocen las medias y las matrices de covarianzas, de hecho:

$$k(1/2) = \frac{1}{8}(\mu_2 - \mu_1)^t \Big[\frac{\Sigma_1\Sigma_2}{2}\Big]^{-1}(\mu_2 - \mu_1) + \frac{1}{2}ln\Big[\frac{|\Sigma_1 + \Sigma_2|/2}{\sqrt{|\Sigma_1||\Sigma_2|}}\Big] \tag{2.68}$$

2.6. Curvas ROC

Como hemos visto, un procedimiento común de clasificación consiste en medir la distancia entre un vector de características $\mathbf{x}$ y la media de la distribución de cierta clase posiblemente utilizando la distancia de Mahalanobis. Si la distancia es inferior a un valor umbral th y el vector efectivamente pertenece a dicha clase, decimos que estamos en presencia de un *positivo verdadero*. Si la distancia entre el patron y la clase es superior a th pero el patrón sí pertenece a la clase, entonces decimos que se trata de un *negativo falso*. En cambio, si la distancia entre el patrón y la clase es inferior th pero el patron en realidad no pertenece a dicha clase, lo llamamos *positivo falso*. Finalmente, si la distancia entre el patrón y la clase es superior a th y se efectivamente el patron no pertenece a la clase en cuestión, es un *negativo verdadero*. La Tabla 2.1 resume estas definiciones.

Tabla 2.1: Definiciones para análisis de sensibilidad de un clasificador

	$dist < th$	$dist > th$
misma clase	Positivo verdadero (TP)	Negativo Falso (FN)
diferente clase	Positivo Falso (FP)	Negativo Verdadero (TN)

Considere la siguiente situación: Asuma que 40,000 comparaciones entre canciones fueran llevadas a cabo en un experimento y sabemos con certeza que de ellas habrá 4,000 ocasiones en las que una canción será comparada con otra versión de la misma canción y el resto de las comparaciones (36,000) serán hechas entre canciones distintas. Suponga que dado un cierto umbral

th el sistema calcula una distancia entre canciones inferior a dicho umbral (por tanto las declara un *match*) para 3,900 de las 4,000 referidas .El resto (100) fueron erróneamente consideradas como canciones distintas (por tener una distancia entre ellas superior al umbral). Por otra parte, 35,950 de las 36,000 fueron adecuadamente consideradas como diferentes y solo 50 fueron incorrectamente consideradas como la misma canción. En la Tabla 2.2 estos resultados hipotéticos son resumidos.

Tabla 2.2: Ejemplo de positivos y negativos para un cierto umbral *th*

	$dist < th$	$dist > th$	Total
Canciones iguales	$TP = 3900$	$FN = 100$	4000 Positivos
Canciones distintas	$FP = 50$	$TN = 35950$	36000 Negativos
Totales	3950	36050	40000

El *Régimen de Positivos verdaderos* (TPR) es la fracción de canciones que el sistema reconoce correctamente (positivos verdaderos) respecto al total de canciones que debería haber reconocido (*Positivos*). Al TPR se le conoce también como also *sensibilidad* o *recall* y se determina con la ecuación (2.69). TPR es también igual a $1 - FRR$ donde FRR es el conocido *Régimen de Rechazo Falso*.

$$TPR = \frac{TP}{TP + FN} \tag{2.69}$$

Para el ejemplo descrito aquí, $TPR = 3900/(3900 + 100) = 0{,}975$=97.5 %

EL *régimen de Positivos Falsos* (FPR) es una medida de la frecuencia con la que el clasificador se equivoca y confunde una canción con otra y se define mediante la Ecuación (2.70). Al FPR se le conoce también como *Régimen de Falsas Alarmas* y es igual a $1 - especificidad$

$$FPR = \frac{FP}{FP + TN} \tag{2.70}$$

Para el ejemplo descrito aquí, $FPR = 50/(50{+}35950) = 0{,}00125$=0.125 % y entonces *specificity*=$1 - 0{,}00125 = 0{,}99875$=99.875 %

El espacio de Características de operación del receptor (ROC) es el plano en el cual el eje vertical es el TPR y el eje horizontal es el FPR. Un punto

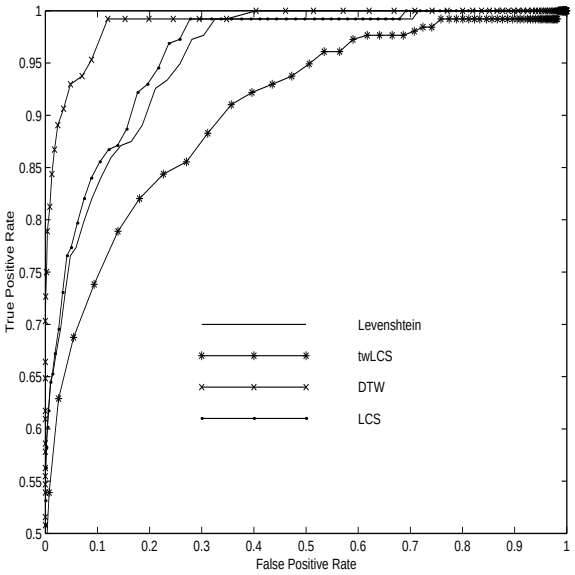

Figura 2.21: Curvas ROC para identificación de canciones utilizando la huella de audio de MPEG-7

en este plano representa el desempeño del sistema clasificador para un cierto valor umbral th. Al permitir que varíe th se genera una curva ROC.

El *Régimen de precisión* se define como la fracción de patrones correctamente reconocidos (positivos verdaderos) entre el número de consultas ejecutadas (positivos verdaderos mas positivos falsos) [1].

Para el ejemplo descrito aquí, el *régimen de precisión* sería 3900/(3900+50)=0.9873=98.73 %

En la Figura 2.21 se muestran las curvas ROC que resultaron al hacer experimentos de reconocimiento de canciones utilizando la huella de audio definida para MPEG-7 [2], en este caso cada curva corresponde a una técnica de alineamiento de series de tiempo distinta.

2.7. Clasificador Bayesiano para características discretas

En algunas ocasiones, las características solo pueden tomar valores de un conjunto discreto (Ej genero masculino o femenino, colores, etc.) Si x es un vector que toma valores de un conjunto $v_1, v_2, ..., v_m$ entonces la probabilidad

a posteriori sería:

$$P(\omega_j|x) = \frac{P(x|\omega_j)P(\omega_j)}{P(x)} = \frac{P(x|\omega_j)P(\omega_j)}{\sum_{i=1}^{c} P(x|\omega_i)P(\omega_i)} \tag{2.71}$$

Observe en la ecuación (2.71) que la probabilidad total para obtener la probabilidad a priori $P(x)$ se obtiene utilizando una sumatoria en lugar de una integral ya que x toma valores discretos.

La regla de decisión básicamente consiste en optar por la acción que reduzca a mínimo Riesgo total, es decir:

$$\alpha = arg \min_{i} R(\alpha_i|x) \tag{2.72}$$

2.7.1. Características binarias e independientes

Como caso particular del problema de características discretas considere el caso en el que solo hay dos clases y los vectores de características son binarios e independientes. Las vectores de características x tienen entonces la forma:

$$x = (x_1, x_2, ..., x_d)^t$$

donde x_i solo puede tomar el valor 0 o 1

Cada bit nos dice algo (si o no) del patron x. Llamemos p_i a la probabilidad de que el i-ésimo bit esté prendido en los patrones que pertenecen a la clase ω_1

$$p_i = P(x_i = 1|\omega_1) \tag{2.73}$$

Y llamemos q_i a la probabilidad de que el i-ésimo bit esté prendido en los patrones que pertenecen a la clase ω_2

$$q_i = P(x_i = 1|\omega_2) \tag{2.74}$$

Por ejemplo, Si dos fábricas hacen coches, podríamos decidir cual de las dos fábricas construyó un auto en particular basado en la funcionalidad (funciona o no funciona) de cada componente.

Considerando que la funcionalidad de cada parte es independiente de las demás partes, entonces:

$$P(x=1|\omega_1) = \prod_{i=1}^{d} p_i^{x_i}(1-p_i)^{1-x_i} \tag{2.75}$$

$$P(x=1|\omega_2) = \prod_{i=1}^{d} q_i^{x_i}(1-q_i)^{1-x_i} \tag{2.76}$$

Recuerde que cada x_i solo puede tomar dos valores (0 o 1)
La razón de verosimilitudes es:

$$\frac{P(x=1|\omega_1)}{P(x=1|\omega_2)} = \prod_{i=1}^{d} \Big(\frac{p_i}{q_i}\Big)^{x_i} \Big(\frac{1-p_i}{1-q_i}\Big)^{1-x_i} \tag{2.77}$$

Recordemos que la función discriminante para un dicotomizador es :

$$g(x) = g_1(x) - g_2(x) = P(\omega_1|x) - P(\omega_2|x) = \frac{P(x|\omega_1)P(\omega_1)}{P(x)} - \frac{P(x|\omega_1)P(\omega_1)}{P(x)} \tag{2.78}$$

$$g(x) = ln\frac{P(x|\omega_1)}{P(x|\omega_2)} + ln\frac{P(\omega_1)}{P(\omega_2)} \tag{2.79}$$

En este caso:

$$g(x) = \sum_{i=1}^{d} \Big[x_i ln\frac{p_i}{q_i} + (1-x_i) ln\Big(\frac{1-p_i}{1-q_i}\Big)\Big] + ln\frac{P(\omega_1)}{P(\omega_2)} \tag{2.80}$$

Entonces se trata de un discriminante lineal gracias a la independencia de las x_i's, por tanto:

$$g(x) = \sum_{i=1}^{d} w_i x_i + w_0 \tag{2.81}$$

Donde

$$w_i = ln\frac{p_i(1-q_i)}{q_i(1-p_i)} \tag{2.82}$$

y

$$w_0 = \sum_{i=1}^{d} \left[ln \frac{1-p_i}{1-q_i} \right] + ln \frac{P(\omega_1)}{P(\omega_2)} \tag{2.83}$$

Si $p_i = q_i$, entonces x_i no proporciona información acerca de cual fábrica hizo el coche y $\omega_i = 0$

2.7.2. Ejemplo

Suponga un clasificador de dos clases en las que las probabilidades a priori son iguales ($P(\omega_1) = P(\omega_2) = 0{,}5$), las características son binarias tridimensionales y se apegan a:

$p_i = P(x_i = 1|\omega_1) = 0{,}8$ Para i=1,2,3
$q_i = P(x_i = 1|\omega_2) = 0{,}5$ Para i=1,2,3
Como $p_1 = p_2 = p_3$, $q_1 = q_2 = q_3$además entonces:

$w_1 = w_2 = w_3 = ln \frac{(0{,}8)(0{,}5)}{(0{,}8)(0{,}2)} = 1{,}3863$

$w_0 = w_0 = \sum_{i=1}^{3} ln \frac{1-0{,}8}{1-0{,}5} + ln \frac{0{,}5}{0{,}5} = -2{,}7488$

Entonces la función discriminante es:

$g(x) = 1{,}3863x_1 + 1{,}3863x_2 + 1{,}3863x_3 - 2{,}7488$

En la Figura 2.22 se muestra la superficie de decisión que define esta función discriminante, para generarla hacemos $g(x) = 0$. Observe que si dos bit valen 1, entonces el patrón corresponde a la clase ω_1 dado que la probabilidad de que los bits de esta clase valgan uno es mayor.

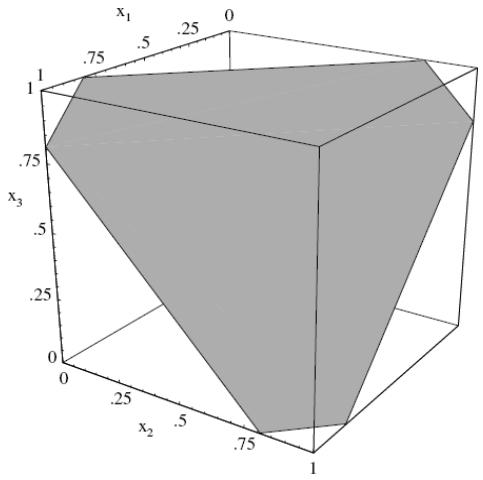

Figura 2.22: Superficie discriminante para el ejemplo de características binarias independientes

2.8. Clasificador Bayesiano para características ruidosas y/o ausentes

En ocasiones una característica puede no estar presente en el vector de características debido por ejemplo a la falla de algún sensor, también puede ocurrir que la característica este presente pero presente una valor poco común debido por ejemplo a problemas en la transmisión de la señal.

Si una característica no existe lo primero que podríamos pensar es reemplazarla el valor medio de dicha característica, sin embargo existe un mejor enfoque que consiste en realizar la clasificación exclusivamente utilizando las características buenas, el cual consiste en determinar la probabilidad de que el patrón pertenezca a cierta clase dadas las características buenas:

$$P(\omega_i|x_b) = \frac{P(\omega_i, x_b)}{P(x_b)} \tag{2.84}$$

donde x_b son las características buenas del patron x mientras que x_m es como nos referiremos a las características malas (Ej ausentes).

$P(\omega_i, x_b)$ la podemos determinar utilizando en concepto de probabilidad

marginal, es decir, integramos para todos los posibles valores de características malas. Entonces:

$$P(\omega_i|x_b) = \frac{\int P(\omega_i|x_b, x_m)P(x_b, x_m)dx_m}{P(x_b)} \tag{2.85}$$

Pero x es precisamente el vector formado por x_b y por x_m, de modo que:

$$P(\omega_i|x_b) = \frac{\int P(\omega_i|x)P(x)dx_m}{\int P(x_b)dx_m} \tag{2.86}$$

donde la probabilidad *marginal* $P(x_b$ de que ocurran las características buenas se obtiene integrando $P(x)$ para todas los posibles valores de las características malas. Finalmente, como $g_i(x) = P(\omega_i|x)P(x)$ obtenemos:

$$P(\omega_i|x_b) = \frac{\int g_i(x)dx_m}{\int P(x_b)dx_m} \tag{2.87}$$

Capítulo 3

Estimación de parámetros

3.1. Introducción

Un clasificador Bayesiano es óptimo, desafortunadamente la probabilidad a priori $P(\omega_i)$ y la verosimilitud o probabilidad condicionada a la clase ($P(x|\omega_i)$) normalmente se desconocen. Utilizando los datos de entrenamiento, se puede hacer una buena estimación de $P(\omega_i)$, sin embargo, para estimar $P(x|\omega_i)$ nunca parece haber suficientes datos. Si el conocimiento del problema nos permite asumir el tipo de densidad de $P(x|\omega_i)$, entonces el problema se reduce a una *estimación de parámetros* de la densidad.

Dos procedimientos muy exitosos para realizar estimación de parámetros son:

1. **Método de la máxima verosimilitud. Esta técnica se basa en maximizar la probabilidad de obtener los datos de entrenamiento.**

2. **Estimación Bayesiana. En este método, los parámetros son variables aleatorias de alguna distribución *a priori*, los datos de entrenamiento la convierten en una distribución *a posteriori*, a esta técnica se le conoce como *Aprendizaje Bayesiano***

3.2. Estimador de Máxima Verosimilitud

Podemos definir a un conjunto de entrenamiento como el conjunto:

$\{D_1, D_2, ..., D_c\}$

Donde D_j es un conjunto de datos de entrenamiento tal que todos sus elementos pertenecen a la clase ω_j la cual tiene una distribución $P(x|\omega_j$, estos datos son i.i.d. (independientes e idénticamente distribuidos).

Llamemos θ_j al conjunto de parámetros a estimar.

Si $P(x|\omega_j) \sim N(\mu_j, \Sigma_j)$, es decir si tiene una distribución Normal (Gaussiana) con media μ_j y matriz de covarianzas Σ_j, entonces θ_j consiste de μ_j y Σ_j.

Escribimos $P(x|\omega_j)$ como $P(x|\omega_j, \theta_j)$ para enfatizar la dependencia de $P(x|\omega_j)$ de θ_j.

Suponemos que D_i solo proporciona información concerniente a θ_i pero no proporciona ninguna información acerca de θ_j, por lo tanto, lo que tenemos que resolver son c problemas idénticos. Plantearemos entonces uno de esos problemas de manera genérica.

Sea $D = \{x_1, x_2, ..., x_n\}$

La *Verosimilitud* de θ es la Probabilidad de que ocurra el conjunto de prueba D cuando los parámetros son los que tiene θ. Como asumimos independencia en los elementos de D (dijimos que eran i.i.d.), entonces:

$$P(D|\theta) = \prod_{k=1}^{n} P(x_k|\theta) \tag{3.1}$$

θ es un vector de dimensión p, formalmente:

$$\theta = [\theta_1, \theta_2, ..., \theta_p]^t \tag{3.2}$$

El operador:

$$\nabla_\theta = \begin{bmatrix} \frac{\partial}{\partial\theta_1} \\ \frac{\partial}{\partial\theta_2} \\ \vdots \\ \frac{\partial}{\partial\theta_p} \end{bmatrix} \tag{3.3}$$

Llamamos $\hat{\theta}$ al valor de θ que maximiza la verosimilitud $P(D|\theta)$. Como el logaritmo es una función monotónicamente creciente, entonces el mismo valor $\hat{\theta}$ maximiza la log-verosimilitud a la cual definimos como:

$$l(\theta) = ln(P(D|\theta)) \tag{3.4}$$

Entonces:

$$\hat{\theta} = arg \max_{\theta} l(\theta) \tag{3.5}$$

$$l(\theta) = ln(P(D|\theta)) = ln \prod_{k=1}^{n} P(x_k|\theta) \tag{3.6}$$

El logaritmo de un producto es igual a la suma de los logaritmos, por tanto:

$$l(\theta) = \sum_{k=1}^{n} ln(P(x_k|\theta)) \tag{3.7}$$

Para maximizar esta cantidad:

$$\nabla_\theta l(\theta) = \sum_{k=1}^{n} \nabla_\theta ln(P(x_k|\theta)) \tag{3.8}$$

$$\nabla_\theta l(\theta) = \sum_{k=1}^{n} \begin{bmatrix} \frac{\partial ln(P(x_k|\theta))}{\partial \theta_1} \\ \frac{\partial ln(P(x_k|\theta))}{\partial \theta_2} \\ \vdots \\ \frac{\partial ln(P(x_k|\theta))}{\partial \theta_p} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.9}$$

Este es un sistema de p ecuaciones simultáneas a resolver. La solución de este sistema de ecuaciones puede ser un máximo o un mínimo local. De entre todas las soluciones, una d ellas cooresponde al máximo global, hay que checar cada solución con la segunda derivada.

Un estimador de máxima verosimilitud es un estimador MAP (*Maximum A Posteriori*), los MAP's maximizan $l(\theta)P(\theta)$. Si $P(\theta)$ es uniforme, el MAP es un estimador de máxima verosimilitud.

3.2.1. Ejemplo de estimación de vector de medias asumiendo distribución Normal

Consideremos uno de los casos mas comunes que consiste en que el conjunto de datos de entrenamiento de cada clase sigue una distribución gaussiana,

entonces los parámetros que tendríamos que estimar son los componentes del vector que representa la media de la distribución así como los componentes de la matriz de covarianzas. Sin embargo, para ejemplificar el procedimiento conviene comenzar por un caso mas simple, es decir, asumamos que conocemos la matriz de covarianzas y que los únicos parámetros que se requieren estimar son los componentes de μ, es decir, solo vamos a estimar por ahora la media de la distribución.

Asumiendo entonces que la verosimilitud o probabilidad condicionada a la clase sigue una distribución gaussiana, entonces la probabilidad de ocurrir (presentarse) de cualquier elemento x_k del conjunto de entrenamiento estaría dada por:

$$p(x_k|\mu) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x_k-\mu)^t\Sigma^{-1}(x_k-\mu)} \tag{3.10}$$

Aplicando logaritmo natural a ambos miembros:

$$ln[p(x_k|\mu)] = -\frac{d}{2}ln(2\pi) - \frac{1}{2}ln|\Sigma| - \frac{1}{2}(x_k-\mu)^t\Sigma^{-1}(x_k-\mu) \tag{3.11}$$

Derivando respecto a μ

$$\frac{\partial ln[p(x_k|\mu)]}{\partial \mu} = -\frac{1}{2}(x_k-\mu)^t\Sigma^{-1}(-1) - \frac{1}{2}(-1)^t\Sigma^{-1}(x_k-\mu) \tag{3.12}$$

Reduciendo:

$$\nabla_\theta[ln[p(x_k|\mu)] = \Sigma^{-1}(x_k-\mu) \tag{3.13}$$

Igualamos a cero para encontrar un punto crítico (Ej un máximo), esto se debe cumplir para cada uno de los componentes de μ:

$$\sum_{k=1}^{n} \Sigma^{-1}(x_k-\hat{\mu}) = 0 \tag{3.14}$$

$$\sum_{k=1}^{n} \Sigma^{-1}x_k - \sum_{k=1}^{n} \Sigma^{-1}\hat{\mu} = 0 \tag{3.15}$$

Sumar n veces algo es igual a multiplicarlo por n

$$\sum_{k=1}^{n} \Sigma^{-1} x_k - n\Sigma^{-1}\hat{\mu} = 0 \tag{3.16}$$

Agrupando Σ^{-1}

$$\Sigma^{-1}\Big[\sum_{k=1}^{n} x_k - n\hat{\mu}\Big] = 0 \tag{3.17}$$

La solución no trivial sería:

$$\sum_{k=1}^{n} x_k - n\hat{\mu} = 0 \tag{3.18}$$

Despejando $\hat{\mu}$

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \tag{3.19}$$

Este era un resultado bastante predecible pero resulta muy satisfactorio sirve para ver que el método de máxima verosimilitud funciona adecuadamente.

3.3. Estimador Bayesiano

Como se comentó al inicio del capítulo, el estimador Bayesiano asume que el vector de parámetros que se quiere estimar (θ) es una variable aleatoria de alguna distribución *a priori*, los datos de entrenamiento la convierten en una distribución *a posteriori*.

3.3.1. Aprendizaje Bayesiano

El clasificador Bayesiano es óptimo pero requiere que se conozca la verosimilitud y la probabilidad a priori, lo mejor que podemos hacer es usar toda la información que esté a nuestro alcance, por ejemplo, si se conoce el tipo de distribución (Ej Gaussiana), el rango en el cual se deben encontrar los valores de los parámetros buscados o información que está inmersa en los datos de entrenamiento. La fórmula de Bayes que usamos para determinar $P(\omega_i|x)$ a partir de $P(x|\omega_i)$, $P(x)$ y $P(\omega_i)$ se puede reescribir de manera que

se destaque el hecho de que la probabilidad a priori, la verosimilitud y la evidencia de que se dispone son valores relativos a (obtenidos de) los datos de entrenamiento D de que se disponen y que la probabilidad a posteriori a la que lleguemos también será determinada dados los datos de entrenamiento, con esto en mente, escribimos:

$$P(\omega_i|x, D) = \frac{P(x|\omega_i, D)P(\omega_i|D)}{P(x|D)} \tag{3.20}$$

$$P(\omega_i|x, D) = \frac{P(x|\omega_i, D)P(\omega_i|D)}{\sum_{j=1}^{c} P(x|\omega_j, D)P(\omega_j|D)} \tag{3.21}$$

Dado que la probabilidad a priori es fácil de determinar a partir del conjunto de entrenamiento podemos escribir $P(\omega_i)$ en lugar de $P(\omega_i|D)$. Po otra parte, en el aprendizaje supervisado que es el que nos interesa, los datos de entrenamiento están etiquetados, es decir $D = \{D_1, D_2, ..., D_c\}$. Con estas dos observaciones en mente podemos reescribir (**??**) para obtener:

$$P(\omega_i|x, D) = \frac{P(x|\omega_i, D_i)P(\omega_i)}{\sum_{j=1}^{c} P(x|\omega_j, D_j)P(\omega_j)} \tag{3.22}$$

Asumimos que los datos de entrenamiento son i.i.d (independientes e idénticamente distribuidos) y resolvemos realmente c problemas separados. Para tratar uno (cualquiera) de estos problemas el índice i se puede obviar (no escribir).

$P(x)$ se desconoce pero podemos considerar que $P(x|\theta)$ sí se conoce así como la distribución a priori $P(\theta)$. El conjunto de entrenamiento convierte $P(\theta)$ en $P(\theta|D)$ que TENDRÁ SU MÁXIMO EN EL VALOR BUSCADO DE θ.

$$P(x|D) = \int P(x, \theta|D)d\theta = \int P(x|\theta, D)P(\theta|D)d\theta \tag{3.23}$$

Como x es obtenido en un proceso independiente del proceso de obtención de D, entonces $P(x|\theta, D) = P(x|\theta)$. Esto tambipen se puede entender como que $P(x)$ se conoce completamente si tan solo se conoce θ. Por lo tanto:

$$P(x|D) = \int P(x, \theta)P(\theta|D)d\theta \tag{3.24}$$

3.3.2. Caso Gaussiano univariado y multivariado

Para el caso en que $P(x|\mu) \sim N(\mu, \sigma^2$, es decir, los datos de entrenamiento siguen una distribución gaussiana y queremos estimar la media de esta distribución. El conocimiento previo que se puede tener de μ se puede expresar en términos de una distribución a priori para μ, es decir:

$$P(\mu) \sim N(\mu_0, \sigma_0) \tag{3.25}$$

Podemos entender μ_0 como nuestra suposición inicial acerca del valor de μ y σ_0 representa la incertidumbre que tenemos al respecto. Una vez que $P(\mu)$ está definida si se extraen valores n valores de los datos de entrenamiento D, entonces obtenemos una distribución a posteriori para μ, como a continuación:

$$P(\mu|D) = \frac{P(D|\mu)P(\mu)}{\int P(D|\mu)P(\mu)d\mu} \tag{3.26}$$

En vista de que estos n valores de D son i.i.d

$$P(\mu|D) = \alpha \prod_{k=1}^{n} P(x_k|\mu)P(\mu) \tag{3.27}$$

donde α es un factor de normalización que depende de D pero no de μ.

Lo que ocurre entonces es que la percepción que tenemos del valor que estamos estimando de μ cambia después de leer n datos del conjunto de entrenamiento, en la Figura 3.1 vemos como un valor inicial dado a la media cambió después de extraer 1,5,10,20 y 30 valores del conjunto de entrenamiento. En la Figura 3.2, el mismo problema pero en dos dimensiones vemos como un valor inicial dado a la media cambia después de extraer 5,12,24 y 50 valores del conjunto de entrenamiento.

3.4. Algoritmo EM (Maximización del valor esperado de la log-verosimilitud

El método de estimación de parámetros de la máxima verosimilitud puede ser muy complicado dependiendo del tipo de distribución que se asuma que tienen los datos. El Algoritmo EM o de Maximización del valor esperado de

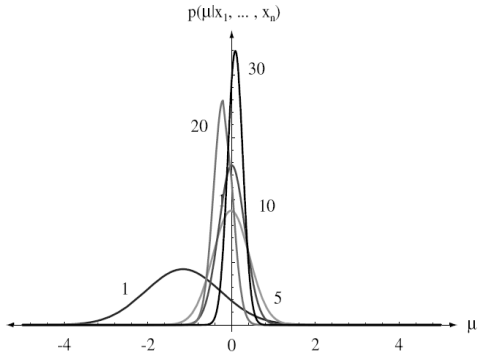

Figura 3.1: Estimación Bayesiana de la media de una distribución normal en una dimensión

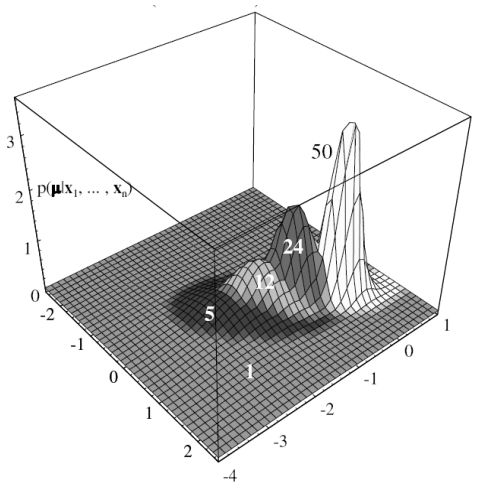

Figura 3.2: Estimación Bayesiana de la media de una distribución normal en dos dimensiones

la log-verosimilitud es un algoritmo iterativo, no intenta resolver de manera analítica el problema como lo hace el método de la máxima verosimilitud.

3.4.1. Estimación de Parámetros con datos faltantes

El Algoritmo EM fue diseñado para estimar parámetros de una distribución cuando los datos están incompletos o se han perdido algunos. En problemas como el de *mezcla de gausianas* se procede de manera que se inventan datos "perdidos" solo para poder utilizar el algoritmo EM [3].

Sea X que el conjunto de datos observado, este es un conjunto incompleto, se asume que un conjunto completo existe $Z = (X, Y)$. El conjunto de datos X sigue una distribución cuyos parámetros θ deseamos determinar. La verosimilitud de los datos completos es:

$$L(\theta|Z) = L(\theta|X, Y) = P(Z|\theta) = P(X, Y|\theta) = P(Y|X, \theta)P(X, \theta) \quad (3.28)$$

La verosimilitud de los datos incompletos es $L(\theta|X) = P(X|\theta)$

El Algoritmo EM primero encuentra el valor esperado de la log-verosimilitud de los datos completos respecto a los datos desconocidos dados los datos observados y las actuales estimaciones de los parámetros buscados, a este le llamamos el paso E.

$$Q(\theta, \theta^{i-i}) = E\Big[ln(P(X, Y|\theta)|X, \theta^{i-1}\Big] \quad (3.29)$$

Donde θ^{i-i} son las actuales estimaciones de los parámetros buscados y θ son los nuevos parámetros que se utilizan para maximizar Q.

A continuación se realiza el paso M, es decir, se maximiza el valor esperado, es decir:

$$\theta^i = arg\,\max_{\theta} Q(\theta, \theta^{i-1}) \quad (3.30)$$

Estos dos pasos (El paso E y el paso M) se repiten las veces que sea necesario hasta que no haya cambios significativos en los parámetros buscados. Una forma modificada de este algoritmo conocida como "Algoritmo EM Generalizado", en el paso M, en lugar de Maximizar $Q(\theta, \theta^{i-1})$, simplemente se encuentra un valor de θ^i tal que $Q(\theta^i, \theta^{i-1}) > Q(\theta, \theta^{i-1})$, Al igual que el Algoritmo EM, este algoritmo se garantiza que converge.

3.4.2. Aplicación del Algoritmo EM al entrenamiento de Modelos Ocultos de Markov

Un Modelos Ocultos de Markov (HMM) es un proceso estocástico que consta de un proceso de Markov no observable (oculto) $q = \{q_t\}_{t\epsilon N}$ y un proceso observable $O = \{O_t\_t\epsilon N$ cuyos estados son dependientes estocásticamente de los estados ocultos, es decir, es un proceso bivariado (q, O). Los HMM se pueden considerar también como sistemas generativos estocásticos, los cuales se emplean en la modelación de series de tiempo. Una cadena de Markov $q = \{q_t\}_{t\epsilon N}$ es un proceso estocástico de Markov discreto. Un proceso estocástico se llama de Markov, si conocido el presente, el futuro no depende del pasado, esto quiere decir, que dada una variable estocástica q_{t-1} que denota el estado del proceso en el tiempo $t-1$, entonces, la probabilidad de transición en el momento t se define como $P[q_t = \sigma_t | q_{t-1} = \sigma_{t-1}]$. Formalmente, una cadena de Markov se define como (Q, A), donde $Q = \{1, 2, ..., N\}$ son los posibles estados de la cadena y $A = (a_{ij})_{nxn}$ es una matriz de transición de estados en el modelo. Si $A(t) = a_{ij}(t)_{nxn}$ es independiente del tiempo, entonces el proceso se llama homogéneo y las probabilidades de transición de estados son de la forma $a_{ij}(t) = P[q_t = j | q_{t-1} = i]$ con las siguientes propiedades:

$$0 \leq a_{ij} \leq 1,\ 1 \leq i \leq N\ ,\ 1 \leq j \leq N$$

$$\sum_{j=1}^{N} a_{ij} = 1,\ 1 \leq i \leq N$$

La condición fundamental de que sea una cadena de Markov, establece que las probabilidades de transición y emisión, dependen solamente del estado actual y no del pasado, esto es:

$$P[q_t = j | q_{t-1} = i, q_{t-2} = k, ...] = P[q_t = j | q_{t-1} = i] = a_{ij}(t). \tag{3.31}$$

Un HMM es una cadena de q junto con un proceso estocástico que toma valores de un alfabeto Σ, el cual depende de q. Estos sistemas evolucionan en el tiempo pasando aleatoriamente de estado a estado y emitiendo en cada momento al azar algún símbolo del alfabeto Σ. Cuando se encuentra en el estado $q_{t-1} = i$, tiene la probabilidad a_{ij} de moverse al estado $q_t = j$ en el

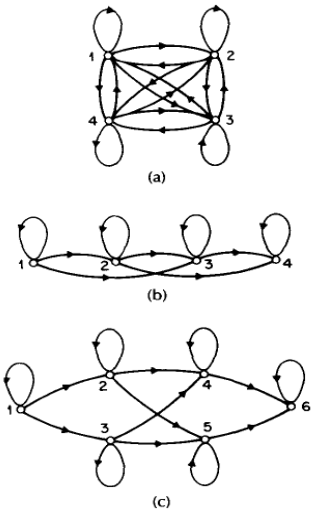

Figura 3.3: Ilustración de tres distintos HMM

siguiente instante y la probabilidad $b_j(k)$ de emitir el símbolo $o_t = v_k$ en el tiempo t. Solamente los símbolos emitidos por el proceso q son observables, pero no la ruta o secuencia de estados q, de ahí, el calificativo de "oculto", ya que el proceso de Markov q es no observable.

Un HMM puede ser representado como un grafo dirigido de transiciones y emisiones como se ilustra en la Figura 3.3. La arquitectura específica que permita modelar de la mejor forma posible las propiedades observadas, depende en gran medida de las características del problema. Las arquitecturas más usadas son:

1. Ergódicas o completamente conectadas, en las cuales cada estado del modelo puede ser alcanzado desde cualquier otro estado en un número finito de pasos (Figura 3.3a).

2. Izquierda-derecha, hacia adelante o Bakis, las cuales tienen la propiedad de que en la medida que el tiempo crece se avanza en la secuencia de observación asociada O y en esa misma medida, el índice que señala el estado del modelo permanece o crece, es decir, los estados del sistema van de izquierda a derecha (Figura 3.3b). En secuencias biológicas y en reconocimiento de la voz estas arquitecturas modelan bien los aspectos

lineales de las secuencias.

3. Izquierda-derecha paralelas, son dos arquitecturas izquierda-derecha conectadas entre sí (Figura 3.3c).

Formalmente, un HMM discreto de primer orden se define como una tupla de cinco elementos.

$$\lambda = (\Sigma, Q, A, B, \pi) \tag{3.32}$$

Donde $\Sigma = \{v_1, v_2, ..., v_M\}$ es un alfabeto o conjunto discreto finito de M símbolos. $Q = \{1, 2, ..., N\}$ es un conjunto finito de N estados. $A = (a_{ij})_{NxN}$ es una matriz de probabilidades de transición donde a_{ij} es la probabilidad de transición desde el estado i al estado j. $B = (b_j(o_t))_{NxM}$ es una matriz de probabilidades de emisión de símbolos, uno por cada estado, donde $b_j = \{b_{j1}, b_{j2}, ..., b_{jM}\}$ es la probabilidad de emisión del símbolo v_k del alfabeto en el estado j. $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$ es un vector de probabilidades del estado inicial q_o en Q.

Las probabilidades de inicialización, transición y emisión son los parámetros del modelo. Un HMM puede ser usado como un generador de secuencias de observaciones $O = \{o_1, o_2, ..., o_T\}$ donde, o_t es uno de los símbolos de Σ para $t\epsilon\{1, 2, ..., T\}$. T es la longitud de la secuencia de observación O, es decir, el número de observaciones en la secuencia. $\lambda = (A, B, \pi)$ son los parámetros del modelo. Un HMM define una medida de probabilidad μ, sobre el espacio de secuencias Σ^*. Existen tres problemas básicos relacionados con los HMM.

1. Calcular eficientemente $P(O|\lambda)$ la probabilidad de la secuencia de observación O dado el modelo $\lambda = (A, B, \pi)$ y la secuencia de observación $O = (o1, o2, ..., o_T)$.

2. Encontrar la trayectoria más probable $q = (q_1, q_2, ..., q_T)$ dado el modelo λ y la secuencia de observación $O = (o1, o2, ..., o_T)$, es decir, $q = arg\,\text{máx}_{r\epsilon Q^*} P(r)$.

3. Ajustar los parámetros A, B, π para maximizar $P(O|\lambda)$.

Una solución a los tres problemas puede encontrarse en [4].

3.4.3. Aplicación a Estimación de Parámetros en mezclas de gaussianas

Los modelos de mezclas de Gaussianas (GMM Gaussian Mixtures Models) han mostrado ser una herramienta poderosa para distinguir fuentes acústicas con diferentes propiedades generales. En reconocimiento de hablante, esta habilidad se ha explotado comúnmente, modelando cada hablante con un GMM. Un GMM está compuesto, básicamente, de una superposición de K funciones de densidad de probabilidad gaussianas, donde cada una está ponderada por un coeficiente de peso

$$p(x) = \sum_{k=1}^{K} N(x|I_k, O_k) \tag{3.33}$$

donde I_k y O_k son la media y la matriz de co varianza de la función de densidad gaussiana asociada a la componente de mezcla k. Por cada clase se estiman los parámetros de los GMM, que incluyen los coeficientes de ponderación, y las medias y matrices de covarianza de cada gaussiana. Los parámetros del modelo suelen estimarse empleando el algoritmo EM, que es una versión generalizada de la estimación de máxima verosimilitud.

En el paso E del algoritmo calcula empleando un conjunto de parámetros iniciales y en el paso M, los parámetros se estiman usando las ecuaciones de actualización.

3.5. Análisis de Componentes principales

El análisis de componentes principales (en español ACP, en inglés, PCA) es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos. Intuitivamente la técnica sirve para determinar el número de factores subyacentes explicativos tras un conjunto de datos que expliquen la variabilidad de dichos datos.

Técnicamente, el PCA busca la proyección según la cual los datos queden mejor representados en términos de mínimos cuadrados. PCA se emplea sobre todo en análisis exploratorio de datos y para construir modelos predictivos. PCA comporta el cálculo de la descomposición en autovalores de la matriz de covarianza, normalmente tras centrar los datos en la media de cada atributo.

El ACP construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de

mayor tamaño del conjunto de datos es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente. Para construir esta transformación lineal debe construirse primero la matriz de covarianza o matriz de coeficientes de correlación. Debido a la simetría de esta matriz existe una base completa de vectores propios de la misma. La transformación que lleva de las antiguas coordenadas a las coordenadas de la nueva base es precisamente la transformación lineal necesaria para reducir la dimensionalidad de datos. Además las coordenadas en la nueva base dan la composición en factores subyacentes de los datos iniciales.

Una de las ventajas de ACP para reducir la dimensionalidad de un grupo de datos, es que retiene aquellas características del conjunto de datos que contribuyen más a su varianza, manteniendo un orden de bajo nivel de los componentes principales e ignorando los de alto nivel. El objetivo es que esos componentes de bajo orden a veces contienen el "más importante" aspecto de esa información.

Supongamos que existe una muestra con n individuos para cada uno de los cuales se han medido m variables (aleatorias) F_j. El ACP permite encontrar un número de factores subyacentes $p < m$ que explican aproximadamente el valor de las m variables para cada individuo. El hecho de que existan estos p factores subyacentes puede interpretarse como una reducción de la dimensionalidad de los datos: donde antes necesitábamos m valores para caracterizar a cada individuo ahora nos bastan p valores. Cada uno de los p encontrados se llama *componente principal*, de ahí el nombre del método.

Existen dos formas básicas de aplicar el ACP:

1. Método basado en la matriz de covarianzas, que se usa cuando los datos son dimensionalmente homogéneos y presentan valores medios similares.

2. Método basado en la matriz de correlación, cuando los datos no son dimensionalmente homogéneos o el orden de magnitud de las variables aleatorias medidas no es el mismo.

El método basado en las covarianzas es el más usado cuando todos los datos son homogéneos y tienen las mismas unidades. Cuando se usan valores muy variables o magnitudes que tienen unidades resulta más adecuado para interpretar los resultados el método basado en correlaciones, que siempre es aplicable sin restricción alguna.

El método basado en correlaciones parte de la matriz de correlaciones, consideremos el valor de cada una de las m variables aleatorias F_j. Para cada uno de los n individuos tomemos el valor de estas variables y escribamos el conjunto de datos en forma de matriz:

$$(F_j^\beta)_{j=1,...,m}^{\beta=1,...,n}. \tag{3.34}$$

Obsérvese que cada conjunto

$$\mathcal{M}_j = \{F_j^\beta | \beta = 1, ..., n\} \tag{3.35}$$

puede considerarse una muestra aleatoria para la variable F_j. A partir de los $m \times n$ datos correspondientes a las m variables aleatorias, puede construirse la matriz de correlación muestral viene definida por:

$$\mathbf{R} = [r_{ij}] \in M_{m\times m} \qquad \text{con } r_{ij} = \frac{\text{cov}(F_i, F_j)}{\sqrt{\text{var}(F_i)\text{var}(F_j)}} \tag{3.36}$$

Puesto que la matriz de correlaciones es simétrica entonces resulta diagonalizable y sus valores propios λ_i, verifican:

$$\sum_{i=1}^{m} \lambda_i = 1 \tag{3.37}$$

Debido a la propiedad anterior estos m valores propios reciben el nombre de pesos de cada uno de los m componentes principales. Los factores principales identificados matemáticamente se representan por la base de vectores propios de la matriz $\mathbf{R}$. Está claro que cada una de las variables puede ser expresada como combinación lineal de los vectores propios o componentes principales.

La aplicación del ACP está limitada por varias asunciones

- Asunción de linealidad: Se asume que los datos observados son combinación lineal de una cierta base.

- Importancia estadística de la media y la covarianza: el ACP utiliza los vectores propios de la matriz de covarianzas y sólo encuentra las direcciones de ejes en el espacio de variables considerando que los datos se distribuyen de manera gaussiana.

3.6. Técnicas no paramétricas de estimación de parámetros

Los métodos no paramétricos no asumen nada acerca de la función de distribución de probabilidades que se quiere estimar dada una colección de datos de entrenamiento. Los datos le van dando forma a la función de distribución de probabilidades, la cual es además suavizada por el kernel elegido (Por ejemplo una ventana de Parzen) [5].

Para resolver el problema de estimar una función de distribución de probabilidades a partir de los datos de entrenamiento, se puede hacer el siguiente planteamiento:

La probabilidad P de que el vector x caiga dentro de la region R esta dada por:

$$P = \int_R p(x')dx' \tag{3.38}$$

Entonces P es una versión suavizada de la función $p(x)$. Suponga que n muestras son extraídas de manera i.i.d.. La probabilidad de que k de estas muestras caigan dentro de la Región R esta dada por:

$$P_k = \binom{n}{k} P^k (1-P)^{n-k} \tag{3.39}$$

El valor esperado para k es $E[k] = nP$

La razón k/n es una buena estimación para la probabilidad P, esta aproximación es particularmente buena cuando n es grande. Si asumimos que $p(x)$ es continua y que la región R es tan pequeña que p puede considerarse constante dentro de dicha región, entonces:

$$\int_R p(x')dx' \approx p(x)V \tag{3.40}$$

donde x es un punto dentro de R y V es el volumen que R abarca
De lo anterior se desprende que:

$$p(x) \approx \frac{k/n}{V} \tag{3.41}$$

Si el volumen V permanece constante, el valor k/n seguramente será un valor finito. Por otra parte, en vista de que el número de muestras no es ilimitado, entonces el volumen V no puede ser arbitrariamente pequeño, si ignoramos esto, desde el punto de vista teórico, podemos contemplar el siguiente procedimiento: Para estimar la densidad en x formamos una secuencia de $R_1, R_2, ..$ que contienen a x, la primera región solo tiene una muestra, la segunda dos y así sucesivamente. Sea V_n el volumen de R_n, k_n el número de muestras que caen dentro de R_n y $p_n(x)$, la n-ésima estimación para $p(x)$ sería:

$$p_n(x) = \frac{k_n/n}{V_n} \tag{3.42}$$

Para que $p_n(x)$ converja a $p(x)$ se requieren tres condiciones:

1.
$$\lim_{n\to\infty} V_n = 0 \tag{3.43}$$

2.
$$\lim_{n\to\infty} V_n = 0 \tag{3.44}$$

3.
$$\lim_{n\to\infty} V_n = 0 \tag{3.45}$$

Existen dos maneras de obtener secuencias de regiones que satisfagan estas condiciones. Una de ellas consiste en comprimir una región inicial especificando el volumen V_n como función de n, por ejemplo $V_n = 1/\sqrt{n}$, luego se muestra que las variables aleatorias k_n y k_n/n se comportan apropiadamente hasta el punto que $p_n(x)$ converge a $p(x)$, este método es básicamente el método de las ventanas de Parzen.

El segundo método consiste en espcificar k_n como una función de n, por ejemplo $k_n = \sqrt{n}$. Aquí el volumen V_n se hace crecer hasta que englobe a k_n vecinos de x. este es el método de los k-vecinos más cercanos a x. En la Figura 3.4 se muestran estos conceptos, en los cuadros de arriba el método de ventanas de Parzen y en los cuadros de abajo el de los k-vecinos.

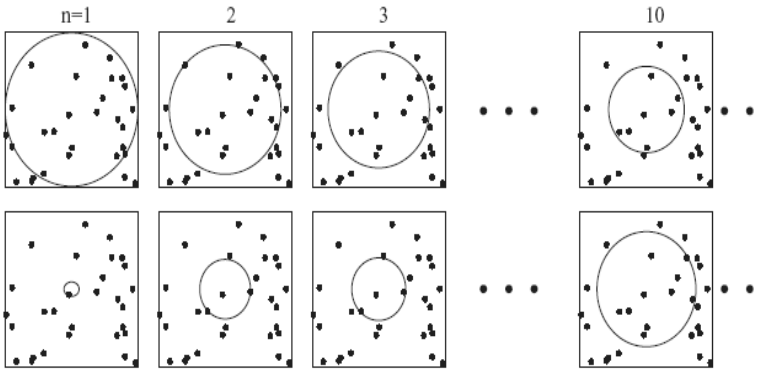

Figura 3.4: Dos métodos no paramétricos de estimación de la función de densidad de probabilidades de datos aleatorios

3.6.1. Método de Ventanas de Parzen

Asuma que la región R_n es un hipercubo en d dimensiones, si h_n es la longitud de una orilla de ese hipercubo, entonces el volumen V_n esta dado por:

$$V_n = h_n^d \tag{3.46}$$

El número de muestras que caen dentro del hipercubo esta dada por:

$$k_n = \sum_{i=1}^{n} \phi\Big(\frac{x - x_i}{h_n}\Big) \tag{3.47}$$

donde:

$$\phi(u) = \begin{cases} 1 & |u_j| \leq 1/2 \; j = 1, ...d \\ 0 & de \;\; otra \;\; manera \end{cases} \tag{3.48}$$

$\phi(u)$ define un hipercubo unitario centrado en el origen, entonces $\phi(u)(x - x_i)/h_n$ regresa 1 si x_i cae dentro del hipercubo centrado en x y cero en caso contrario. Substituyendo (3.47) en (3.42) obtenemos:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\phi\Big(\frac{x - x_i}{h_n}\Big) \tag{3.49}$$

En lugar de limitarnos a una función ventana tipo hipercubo podemos utilizar una clase de funciones ventana mas general. La ecuación (3.49) calcula una estimación de $p(x)$ mediante un promedio de funciones de x, podemos ver a la función ventana para hacer interpolación donde cada muestra contribuye con una cantidad que depende de su distancia a x.

Definiendo la función ventana de Parzen $\delta_n(x)$ de ancho h_n mediante:

$$\delta_n(x) = \frac{1}{V_n}\phi\Big(\frac{x}{h_n}\Big) \tag{3.50}$$

entonces podemos reescribir (3.49)

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(x - x_i) \tag{3.51}$$

En la Figura 3.5 podemos ver la ventana de Parzen $\delta(x)$ para diferentes valores de h y en la Figura 3.6 podemos ver las estimaciones hechas para cierto conjunto de datos utilizando estas ventanas.

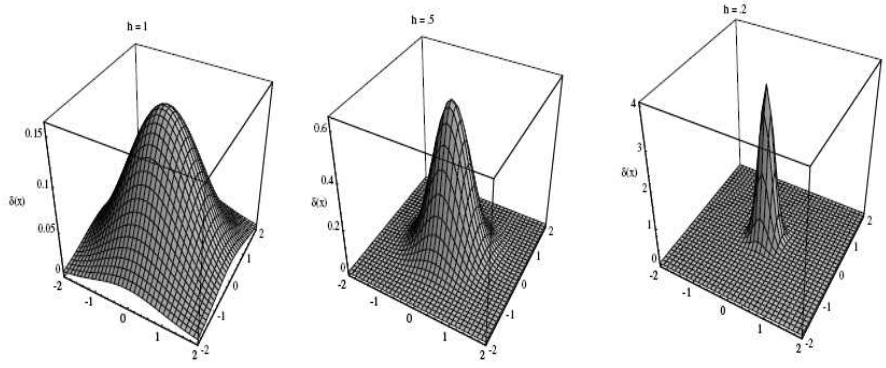

Figura 3.5: La ventana de Parzen para $h = 1, 2, 3$

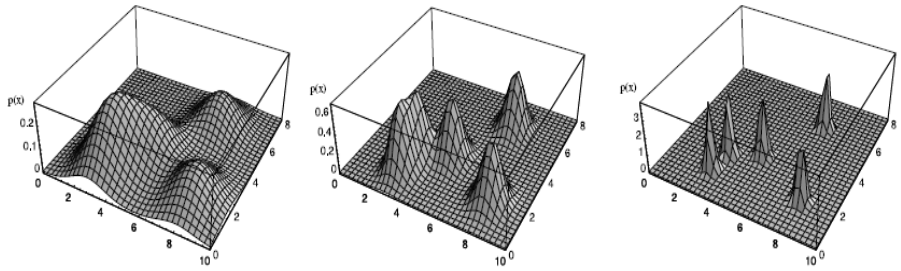

Figura 3.6: Estimaciones de $p(x)$ usando las ventanas de parzen de la figura 3.5

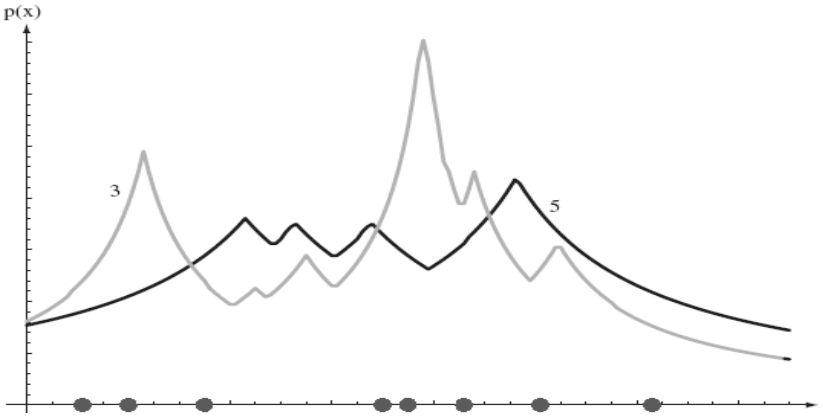

Figura 3.7: Estimaciones de $p(x)$ usando el método de k-vecinos con $k = 3$ y $k = 5$

3.6.2. Método de los K-Vecinos más cercanos

La solución al problema que plantea el método de las ventanas de Parzen de averiguar el mejor tipo de ventana (el ancho) para cada caso es dejar que la ventana sea dependiente de los datos. Podemos entonces estimar p(x) centrando un área alrededor de x y permitir que esta crezca hasta hasta que encierre a k_n muestras (k_n puede por ejemplo ser de $\sqrt{n}$), estas muestras son los k-vecinos mas cercanos a x. Si el área requirió crecer mucho, entonces seguramente $p(x)$ reporta un valor bajo y si el área requirió crecer poco, $p(x)$ tiene un valor alto de acuerdo a:

$$p_n(x) = \frac{k_n/n}{V_n} \tag{3.52}$$

En la Figura se muestra la estimación de $p(x)$ de un conjunto de 8 datos mediante el método de k-vecinos usando $k = 3$ y $k = 5$.

Capítulo 4

Redes Neuronales

Las redes de neuronas artificiales (denominadas habitualmente como RNA o en inglés como: "ANN") son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida. Esta salida viene dada por tres funciones:

1. Una función de propagación (también conocida como función de excitación), que por lo general consiste en el sumatorio de cada entrada multiplicada por el peso de su interconexión (valor neto). Si el peso es positivo, la conexión se denomina excitatoria; si es negativo, se denomina inhibitoria.

2. Una función de activación, que modifica a la anterior. Puede no existir, siendo en este caso la salida la misma función de propagación.

3. Una función de transferencia, que se aplica al valor devuelto por la función de activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que queramos darle a dichas salidas. Algunas de las más utilizadas son la función sigmoidea (para obtener valores en el intervalo [0,1]) y la tangente hiperbólica (para obtener valores en el intervalo [-1,1]).

Biológicamente, un cerebro aprende mediante la reorganización de las conexiones sinápticas entre las neuronas que lo componen. De la misma man-

era, las RNA tienen un gran número de procesadores virtuales interconectados que de forma simplificada simulan la funcionalidad de las neuronas biológicas. En esta simulación, la reorganización de las conexiones sinápticas biológicas se modela mediante un mecanismo de pesos, que son ajustados durante la fase de aprendizaje. En una RNA entrenada, el conjunto de los pesos determina el conocimiento de esa RNA y tiene la propiedad de resolver el problema para el que la RNA ha sido entrenada.

Por otra parte, en una RNA, además de los pesos y las conexiones, cada neurona tiene asociada una función matemática denominada función de transferencia. Dicha función genera la señal de salida de la neurona a partir de las señales de entrada. La entrada de la función es la suma de todas las señales de entrada por el peso asociado a la conexión de entrada de la señal. Algunos ejemplos de entradas son la función escalón de Heaviside, la lineal o mixta, la sigmoide y la función gaussiana, recordando que la función de transferencia es la relación entre la señal de salida y la entrada. Ventajas

Las redes neuronales artificiales (RNA) tienen muchas ventajas:

- Aprendizaje: Las RNA tienen la habilidad de aprender mediante una etapa que se llama etapa de aprendizaje. Esta consiste en proporcionar a la RNA datos como entrada a su vez que se le indica cuál es la salida (respuesta) esperada.

- Auto organización: Una RNA crea su propia representación de la información en su interior, descargando al usuario de esto.

- Tolerancia a fallos: Debido a que una RNA almacena la información de forma redundante, ésta puede seguir respondiendo de manera aceptable aun si se daña parcialmente.

- Flexibilidad: Una RNA puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada (ej. si la información de entrada es la imagen de un objeto, la respuesta correspondiente no sufre cambios si la imagen cambia un poco su brillo o el objeto cambia ligeramente)

- Tiempo real: La estructura de una RNA es paralela, por lo cuál si esto es implementado con computadoras o en dispositivos electrónicos especiales, se pueden obtener respuestas en tiempo real.

4.1. El Perceptrón generalizado

El Perceptrón es un tipo de red neuronal artificial desarrollado por Frank Rosenblatt, también puede entenderse como perceptrón la neurona artificial y unidad básica de inferencia en forma de discriminador lineal, que constituye este modelo de red neuronal artificial, esto debido a que el perceptrón puede usarse como neurona dentro de un perceptrón más grande u otro tipo de red neuronal artificial.

El perceptrón usa una matriz para representar las redes neuronales y es un discriminador terciario que traza su entrada x (un vector binario) a un único valor de salida $f(x)$ (un solo valor binario) a través de dicha matriz.

$$f(x) = \begin{cases} 1 & w \cdot x - u > 0 \\ 0 & \text{en caso contrario} \end{cases} \tag{4.1}$$

Donde w es un vector de pesos reales y $w \cdot x$ es el producto punto (que computa una suma ponderada). u es el 'umbral', el cual representa el grado de inhibición de la neurona, es un término constante que no depende del valor que tome la entrada.

El valor de $f(x)$ (0 o 1) se usa para clasificar x como un positivo o un caso negativo, en el caso de un problema de la clasificación binario. El umbral puede pensarse de como compensar la función de activación, o dando un nivel bajo de actividad a la neurona del rendimiento. La suma ponderada de las entradas debe producir un valor mayor que u para cambiar la neurona de estado 0 a 1. En la Figura 4.1 puede verse un perceptrón de dos entradas x_i y x_2 con sus correspondientes pesos w_1 y w_2, la salida Y depende del umbral θ.

El algoritmo de aprendizaje es el mismo para todas las neuronas, todo lo que sigue se aplica a una sola neurona en el aislamiento. Nosotros definimos algunas variables primero:

$x(j)$ denota el elemento en la posición j en el vector de la entrada
$w(j)$ el elemento en la posición j en el vector de peso
y denota la salida de la neurona
δ denota la salida esperada
α es una constante tal que $0 < \alpha < 1$

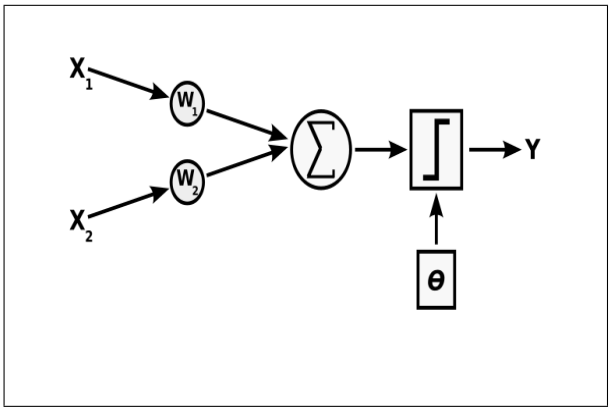

Figura 4.1: El perceptrón

4.1.1. La regla delta

Los pesos son actualizados después de cada entrada según la regla de actualización siguiente:

$$w(j)' = w(j) + \alpha(\delta - y)x(j) \tag{4.2}$$

Por lo cual, el aprendizaje es modelado como la actualización del vector de peso después de cada iteración, lo cual sólo tendrá lugar si la salida y difiere de la salida deseada δ. Para considerar una neurona al interactuar en múltiples iteraciones debemos definir algunas variables más:

x_i denota el vector de entrada para la iteración i
w_i denota el vector de peso para la iteración i
y_i denota la salida para la iteración i
$D_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ denota un periodo de aprendizaje de m iteraciones

En cada iteración el vector de peso es actualizado como sigue:

- Para cada pareja ordenada (x, y) en $D_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
  Pasar (x_i, y_i, w_i) a la regla de actualización

$$w(j)' = w(j) + \alpha(\delta - y)x(j) \tag{4.3}$$

El periodo de aprendizaje D_m se dice que es separable linealmente si existe un valor positivo γ y un vector de peso w tal que: $y_i \cdot (\langle w, x_i \rangle + u) > \gamma$ para todos los i.

Novikoff (1962) probo que el algoritmo de aprendizaje converge después de un número finito de iteraciones si los datos son separables linealmente y el número de errores esta limitado a:

$$\left(\frac{2R}{\gamma}\right)^2. \tag{4.4}$$

Sin embargo si los datos no son separables linealmente, la línea de algoritmo anterior no se garantiza que converja.

4.2. Topología de redes neuronales

Una red neuronal artificial puede estar formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple).

El perceptrón multicapa

El perceptrón multicapa puede ser totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa i es entrada de todas las neuronas de la capa $i+1$, mientras que el segundo, cada neurona de la capa i es entrada de una serie de neuronas (región) de la capa $i+1$. A las capas de neuronas las denominamos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y las salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

En la Figura 4.2 se muestra el perceptrón multicapa.

En el perceptrón multicapa, las funciones de transferencia de los elementos de procesado (neuronas) han de ser derivables. También se reportan las siguientes limitaciones:

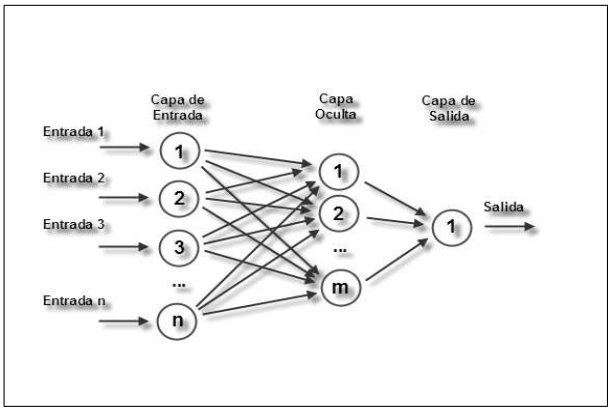

Figura 4.2: El perceptrón multicapa

- El Perceptrón Multicapa no extrapola bien, es decir, si la red se entrena mal o de manera insuficiente, las salidas pueden ser imprecisas.
- La existencia de mínimos locales en la función de error dificulta considerablemente el entrenamiento, pues una vez alcanzado un mínimo el entrenamiento se detiene aunque no se haya alcanzado la tasa de convergencia fijada.

Cuando caemos en un mínimo local sin satisfacer el porcentaje de error permitido se puede considerar: cambiar la topología de la red (número de capas y número de neuronas), comenzar el entrenamiento con unos pesos iniciales diferentes, modificar los parámetros de aprendizaje, modificar el conjunto de entrenamiento o presentar los patrones en otro orden.

De acuerdo a su topología, es decir del patrón de conexiones que presentan, las redes de neuronas artificiales se clasifican en tres tipos básicos:

1. Las reddes Monocapa. Ejemplos: perceptrón, Adaline.
2. Las redes de propagación hacia delante o acíclicas en las que todas las señales van desde la capa de entrada hacia la salida sin existir ciclos, ni conexiones entre neuronas de la misma capa. Ejemplos: perceptrón multicapa.
3. Las redes recurrentes que presentan al menos un ciclo cerrado de activación neuronal. Ejemplos: Elman, Hopfield, máquina de Bolzman.

4.3. Funciones de activación

La Función de Activación de un nodo define la salida de un nodo dada una entrada o un set de entradas. Se podría decir que un circuito estándar de computador se comporta como una red digital de funciones de activación al activarse como "ON" (1) u "OFF" (0), dependiendo de la entrada. Esto es similar al funcionamiento de un Perceptrón en una Red neuronal artificial. La función de activación combina el potencial postsináptico, que nos proporciona la función de propagación, con el estado actual de la neurona para conseguir el estado futuro de activación de la neurona. Sin embargo, es muy común que las redes neuronales no tomen su propio estado como un parámetro y que por tanto no se considere . Esta función es normalmente creciente monótona y podemos citar las funciones más comunes:

- Lineal: Algunas redes neuronales usan esta función de activación como el Adeline por su eficiencia y facilidad.
- Escalón: Esta función es la más usada para redes neuronales binarias ya que no es lineal y es muy simple. Algunas redes que usan esta función son el Perceptrón y Hopfield. Para redes que trabajan en el rango [-1,1] se usa la función signo.
- Hiperbólicas o tangenciales: Las redes con salidas continuas, como el Perceptron multicapa con retropropagación, usan esta función ya que su algoritmo de aprendizaje necesita una función derivable

En las redes neurales inspiradas sobre la biología, la función de activación es usualmente una abstracción representando un tasa de potencial de activación en la celda. En su forma simplificada, esta función es binaria, esto es, se activa la neurona o no. La función se ve como $\phi(v_i) = U(v_i)$, donde U es la función escalón. En este caso, un gran número de neuronas deben ser usadas en computación más allá de la separación lineal de las categorías.

Una función rampa también puede ser usada para reflejar el incremento del potencial de activación que ocurre cuando la entrada se incrementa. La función podría ser de la forma $\phi(v_i) = \mu v_i$, donde μ es la pendiente. Esta función de activación es lineal, y por consiguiente tiene los mismos problemas que la función binaria. En adición, las redes neurales construidas usando este modelo tienen convergencia inestable porque a la larga, las entradas a la neurona tienden a incrementarse sin límite, esta función no es normalizable.

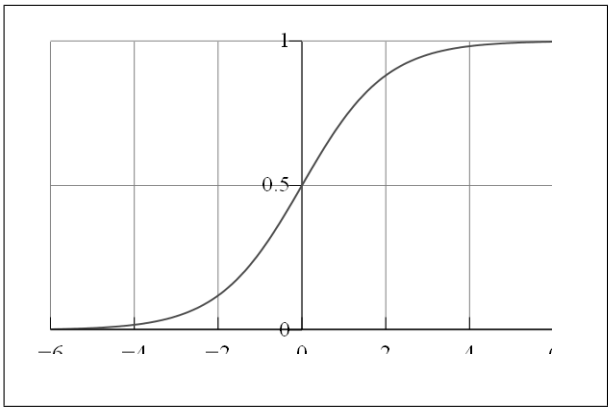

Figura 4.3: La Función Sigmoidea

Los problemas mencionados anteriormente, pueden ser manejados usando una función de activación sigmoidal. Un modelo realístico permanece en cero hasta que una entrada es recibida, en este punto la frecuencia de activación se incrementa rápidamente, pero gradualmente llega a ser asíntota cuando la frecuencia es 100 %. Matemáticamente, esto se ve como $\phi(v_i) = U(v_i)tanh(v_i)$, donde la función de tangente hiperbólica puede también ser cualquier función sigmoidal. Esta conducta es realísticamente reflejada en la neurona, ya que las neuronas no pueden físicamente activarse más rápido que una cierta tasa.

4.3.1. Función sigmoidea

El modelo final que es usado en perceptrones multicapa es el modelo de activación sigmoidal en la forma de tangente hiperbólica. Dos formas de esta función son comúnmente usados: $\phi(v_i) = tanh(v_i)$ cuyos rangos son normalizados desde -1 hasta 1. El de la función:

$$\phi(v_i) = \frac{1}{1 + e^{-v_i}} \tag{4.5}$$

es verticalmente normalizado desde 0 a 1. Este último es frecuentemente considerado más biológicamente realístico. En la Figura 4.3 se muestra la función sigmoidea.

4.4. Aprendizaje

Una forma de clasificación de las redes neuronales es en función del tipo de aprendizaje de que es capaz (si necesita o no un conjunto de entrenamiento supervisado). Para cada tipo de aprendizaje encontramos varios modelos propuestos:

- Aprendizaje supervisado: necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce. Ejemplos de este tipo de redes son: el perceptrón simple, la red Adaline, el perceptrón multicapa y la memoria asociativa bidireccional.
- Aprendizaje no supervisado o autoorganizado: no necesitan de tal conjunto previo. Ejemplos de este tipo de redes son: las memorias asociativas, las redes de Hopfield, la máquina de Bolzman y la máquina de Cauchy, las redes de aprendizaje competitivo, las redes de Kohonen o mapas autoorganizados y las redes de resonancia adaptativa (ART)
- Redes híbridas: son un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia. Un ejemplo de este último tipo son las redes de base radial.
- Aprendizaje reforzado: se sitúa a medio camino entre el supervisado y el autoorganizado.

4.4.1. Propagación hacia atrás

La propagación hacia atrás de errores o retropropagación (del inglés backpropagation) es un algoritmo de aprendizaje supervisado que se usa para entrenar redes neuronales artificiales. El algoritmo consiste en minimizar un error (comúnmente cuadrático) por medio de gradiente descendiente, por lo que la parte esencial del algoritmo es cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

Los algoritmos en Aprendizaje Automático pueden ser clasificados en dos categorías: supervisados y no supervisados. Los algoritmos en aprendizaje supervisado son usados para construir “modelos” que generalmente predicen un ciertos valores deseados. Para ello, los algoritmos supervisados requieren que se especifiquen los valores de salida (output) u objetivo (target) que se asocian a ciertos valores de entrada (input). Ejemplos de objetivos pueden

ser valores que indican éxito/fallo, venta/no-venta, pérdida/ganancia, o bien ciertos atributos multi-clase como cierta gama de colores o las letras del alfabeto. El conocer los valores de salida deseados permite determinar la calidad de la aproximación del modelo obtenido por el algoritmo.

La especificación de los valores entrada/salida se realiza con un conjunto consistente en pares de vectores con entradas reales de la forma (x,t), conocido como conjunto de entrenamiento o conjunto de ejemplos. Los algoritmos de aprendizaje generalmente calculan los parámetros W de una función $N(x;W)$ que permiten aproximar los valores de salida en el conjunto de entrenamiento.

Si $(x^{(q)}, t^{(q)})$, $q = 1, \ldots, p$, son los elementos del conjunto de entrenamiento, la calidad de la aproximación en el ejemplo q se puede medir a través del error cuadrático:

$$E(\boldsymbol{x}^{(q)}; \boldsymbol{W}) = \frac{1}{2}\|N(\boldsymbol{x}^{(q)}; \boldsymbol{W}) - \boldsymbol{t}^{(q)}\|^2 \tag{4.6}$$

donde $\|\cdot\|$ es la norma euclidiana.

El error total es la suma de los errores de los ejemplos:

$$E(\boldsymbol{W}) = \sum_{q=1}^{p} E(\boldsymbol{x}^{(q)}; \boldsymbol{W}). \tag{4.7}$$

Un método general para minimizar el error es el actualizar los parámetros de manera iterativa. El valor nuevo de los parámetros se calcula al sumar un incremento $\Delta\boldsymbol{W}$ al valor actual:

$$\boldsymbol{W} := \boldsymbol{W} + \Delta\boldsymbol{W} \tag{4.8}$$

El algoritmo se detiene cuando $\boldsymbol{W}$ converge o bien cuando el error alcanza un mínimo valor deseado.

Si la función $N(\boldsymbol{x}; \boldsymbol{W})$ usada para aproximar los valores de salida es diferenciable respecto a los parámetros $\boldsymbol{W}$, podemos usar como algoritmo de aprendizaje el método de gradiente descendiente. En este caso, el incremento de los parámetros se expresa como

$$\Delta\boldsymbol{W} = -\gamma \frac{\partial E(\boldsymbol{W})}{\partial \boldsymbol{W}} \tag{4.9}$$

donde $0 < \gamma < 1$ es un parámetro conocido como factor de aprendizaje.

Definimos $\bar{\boldsymbol{v}} = (v_1, \ldots, v_n, 1)^T$ como el vector extendido del vector $\boldsymbol{v} = (v_1, \ldots, v_n)^T$. El par $(\boldsymbol{x}, \boldsymbol{t})$ representará a un elemento del conjunto de entrenamiento y una relación de entrada-salida, a menos que se indique otra cosa.

4.4.2. Red con una capa oculta

Los elementos que constituyen a la red neuronal son los siguientes:

- s es una función de valores reales, conocida como la función de transferencia.
- $\bar{\boldsymbol{o}}^{(0)}$ es la capa de entrada, considerado como el vector extendido $\boldsymbol{o}^{(0)} = \boldsymbol{x} = (x_1, \ldots, x_n)^T$.
- $\bar{\boldsymbol{o}}^{(1)}$ es la capa oculta, el vector extendido de $\boldsymbol{o}^{(1)} = (o_1^{(1)}, \ldots, o_k^{(1)})^T$.
- $\boldsymbol{o}^{(2)} = (o_1, \ldots, o_m)^T$ es la capa de salida, considerado como el vector que aproxima al valor deseado $\boldsymbol{t} = (t_1, \ldots, t_m)^T$.
- $\boldsymbol{W}^{(1)}$ es una matriz de tamaño $(n+1) \times k$ cuyos valores $W_{ij}^{(1)}$ son los pesos de la conexión entre las unidades $\bar{o}_i^{(0)}$ y $o_j^{(1)}$.
- $\boldsymbol{W}^{(2)}$ es una matriz de tamaño $(k+1) \times m$ cuyos valores $W_{ij}^{(2)}$ son los pesos de la conexión entre las unidades $\bar{o}_i^{(1)}$ y $o_j^{(2)}$.

De estos elementos, únicamente las matrices $\boldsymbol{W}^{(l)}$ son consideradas como los parámetros de la red, ya que los valores $\bar{\boldsymbol{o}}^{(l)}$ son el resultado de cálculos que dependen de las matrices de pesos, del valor de entrada $\bar{\boldsymbol{x}}$ y de la función de transferencia s.

La función de transferencia s que consideraremos en nuestro algoritmo es conocida como función sigmoidal, y esta definida como

$$s(u) = \frac{1}{1 + e^{-u}} \tag{4.10}$$

esta función además de ser diferenciable, tiene la particularidad de que su derivada se puede expresar en términos de sí misma:

$$\frac{ds(u)}{du} = s(u)(1 - s(u)). \tag{4.11}$$

esto nos servirá para simplificar los cálculos en el algoritmo de aprendizaje descrito a continuación:

1. Calcular la salida de la red $\boldsymbol{o}^{(2)}$ a partir de uno de los conjuntos de valores de prueba x.

2. Comparar con la salida correcta t y calcular el error según la fórmula:

$$E(\boldsymbol{x}; \boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}) = \frac{1}{2} \sum_{i=1}^{m} (o_i^{(2)} - t_i)^2. \tag{4.12}$$

3. Calcular las derivadas parciales del error con respecto a los pesos $\boldsymbol{W}^{(2)}$ que unen la capa oculta con la de salida.

4. Calcular las derivadas parciales del error con respecto a los pesos $\boldsymbol{W}^{(1)}$ que unen la capa de entrada con la oculta.

5. Ajustar los pesos de cada neurona para reducir el error.

6. Repetir el proceso varias veces por cada par de entradas-salidas de prueba.

4.5. Memorias Asociativas

Se entiende por memoria asociativa el almacenamiento y recuperación de información por asociación con otras informaciones.

Un dispositivo de almacenamiento de información se llama memoria asociativa si permite recuperar información a partir de conocimiento parcial de su contenido, sin saber su localización de almacenamiento. A veces también se le llama memoria de direccionamiento por contenido

Los computadores tradicionales no usan este direccionamiento; se basan en el conocimiento exacto de la dirección de memoria en la que se encuentra la información.

Sin embargo, se cree que el cerebro humano no actúa así. Si queremos recordar el nombre de una persona, no nos sirve saber que fue el nombre número 3274 que aprendimos. Es más útil saber que su nombre empieza y termina por 'N' y que es un famoso científico inglés. Con esta información, es casi seguro que recordaremos exitosamente a "Newton".

Las memorias asociativas son una de las redes neuronales artificiales más importantes con un amplio rango de aplicaciones en áreas tales como: Memorias de acceso por contenido, identificación de patrones y control inteligente.

Una memoria asociativa puede almacenar información y recuperarla cuando sea necesario, es decir, una red retroalimentada, cuya salida se utiliza repetidamente como una nueva entrada hasta que el proceso converge. Puede recuperar dicha información basándose en el conocimiento de parte de ésta (clave). El patrón clave puede ser una versión con ruido de un patrón memorizado, es decir, que difiere de él en pocas componentes. La memoria humana recuerda a una persona aunque vaya vestida de forma diferente o lleve gafas.

Tipos de Memorias Asociativas:

- Memorias heteroasociativas: establecen una correspondencia de x (vector de entrada) en y (vector de salida), de distinta dimensión. Dichos patrones se llaman memorias principales o de referencia.

- Memorias autoasociativas: establece la misma correspondencia que la memoria heteroasociativa pero siendo los patrones de entrada y de salida los mismos.

Capítulo 5

Métodos estocásticos

El aprendizaje juega un rol central en el diseño de clasificadores de patrones, el enfoque tradicional como hemos visto consiste en especificar un modelo y luego estimar los parámetros del mismo a partir de los datos de entrenamiento. Estos métodos se vuelven insatisfactorios a medida que el modelo se hace mas complejo. Los métodos estocásticos han venido a resultar útiles para la determinación de parámetros cuando la aleatoriedad juega un papel importante tanto en el aprendizaje como en la búsqueda, en general los métodos estocásticos sesgan la búsqueda hacia regiones donde que permitan aleatoriedad de manera que los "buenos parámetros" tengan cierta ventaja de ser encontrados aún en los modelos mas complejos.

5.1. Recocido simulado

El algoritmo de recocido simulado está basado en una analogía entre la simulación de recocido de sólidos y la problemática de resolver problemas de optimización combinatoria de gran escala. Por esta razón el algoritmo se conoce como recocido simulado. Recocido denota un proceso de calentamiento de un sólido a una temperatura en la que sus granos deformados recristalizan para producir nuevos granos. La temperatura de recocido o de recristalización, depende del tipo de material, del grado de deformación del mismo, además de su uso futuro. Seguida a la fase de calentamiento, viene un proceso de enfriamiento en donde la temperatura se baja poco a poco. De esta manera, cada vez que se baja la temperatura, las partículas se reacomodan en estados de más baja energía hasta que se obtiene un sólido con sus

partículas acomodadas conforme a una estructura de cristal. Si se comienza con un valor máximo de la temperatura, en la fase de enfriamiento del proceso de recocido, para cada valor de la temperatura T debe permitirse que se alcance su equilibrio térmico. Sin embargo, si el proceso de enfriamiento es demasiado rápido y no se alcanza en cada etapa el equilibrio térmico, el sólido congelará en un estado cuya estructura será amorfa en lugar de la estructura cristalina de más baja energía. La estructura amorfa está caracterizada por una imperfecta cristalización del sólido.

El equilibrio térmico está caracterizado por la distribución de Boltzmann. De acuerdo a esta distribución, la probabilidad de que el sólido esté en un estado i con energía E_i a la temperatura T, viene dada por

$$P_t(X = i) = \frac{1}{Z(T)} e^{\frac{E_i}{k_B T}} \tag{5.1}$$

donde X es una variable aleatoria que denota el estado actual del sólido. $Z(T)$ es una constante de normalización llamada la función partición, que está definida como

$$Z_T = e^{\frac{E_j}{k_B T}} \tag{5.2}$$

donde la sumatoria se extiende sobre todos los posibles estados y k_B es una constante física conocida como la constante de Boltzmann. El factor $e^{\frac{E_j}{k_B T}}$ se conoce como el factor de Boltzmann. Obviamente (5.1) es una función de densidad de probabilidad ya que siempre es mayor o igual a cero y la suma sobre todos los valores es igual a la unidad. Se puede observar en (5.1) que cuando el valor de T disminuye, la distribución de Boltzmann se concentra en los estados de menor energía mientras que si la temperatura se aproxima a cero, únicamente los estados con mínima energía tienen una probabilidad de ocurrencia diferente de cero.

Por lo dicho anteriormente, el proceso de recocido consta de dos pasos fundamentales que son:

1. Incrementar la temperatura del barro térmico a un valor máximo.

2. Decrementar cuidadosamente la temperatura del barro térmico hasta que las partículas se reacomoden por sí mismas en un estado de mínima energía, denominado el estado fundamental del sólido.

El proceso físico de recocido puede moderarse exitosamente usando métodos de simulación, el algoritmo introducido para tal propósito se basa en técnicas Monte Carlo y genera una sucesión de estados del sólido de la siguiente manera. Dado un estado i del sólido con energía E_i, se genera un estado subsecuente j aplicando un mecanismo de perturbación que transforma el estado actual en el siguiente estado por medio de una pequeña distorsión, por ejemplo, por el desplazamiento de una partícula. La energía del siguiente estado es E_j. Si la diferencia de energía, $E_j - E_i$, es menor o igual a cero, el estado j se acepta como el estado actual. Si la diferencia de energía, es mayor que cero, el estado j se acepta con una probabilidad que está dada por

$$e^{fracE_j k_B T} \tag{5.3}$$

donde T denota la temperatura del baño térmico y k_B es la constante de Boltzman. La regla de decisión descrita arriba se conoce como el criterio de Metropolis y al algoritmo se le conoce como algoritmo de Metropolis.

Si la temperatura se baja poco a poco, el sólido puede alcanzar su equilibrio térmico en cada temperatura. En el algoritmo de Metropolis esto se lleva a cabo generando un número grande de transiciones para un valor dado de la temperatura.

5.1.1. El Algoritmo de Recocido Simulado

La simulación del proceso de recocido puede usarse para describir un proceso de generación de una sucesión de soluciones de un problema de optimización combinatoria, en donde se vayan obteniendo, conforme el proceso avanza, mejores soluciones al mismo. Para este propósito, se puede observar una analogía entre el sistema físico y un problema de optimización combinatoria en donde cada solución del problema puede verse como un estado del sólido y el valor de la función objetivo para la el nivel de energía del sólido. En resumen, se puede pensar en las siguientes equivalencias.

- Las soluciones de un problema de optimización combinatoria son equivalentes a los estados de un sistema físico.
- El costo de una solución es equivalente a la energía de un estado.

Además, se introduce un parámetro que juega el papel equivalente de la temperatura. Este parámetro se llama el parámetro de control. El algorit-

mo de recocido simulado puede verse como una iteración del algoritmo de Metropolis, evaluado en valores decrecientes del parámetro de control.

Sea (S, f) una Instancia de un problema de optimización combinatoria, y denote por i y j dos soluciones con costo $f(i)$ y $f(j)$, respectivamente. Entonces el criterio de aceptación determina si j se acepta de i a parir de aplicar la siguiente probabilidad de aceptación:

$$P_c(aceptar\ j) = \begin{cases} 1 & f(j) \leq f(i) \\ e^{\frac{f(i)-f(j)}{c}} & f(j) > f(i) \end{cases} \tag{5.4}$$

donde $c \epsilon R^+$ denota el parámetro de control.

Claramente, el mecanismo de generación corresponde al mecanismo de perturbación en el algoritmo de Metropolis, mientras que el criterio de aceptación corresponde al criterio de Metropolis.

Una transición es una acción combinada que transforma la solución actual en una subsecuente. Esta acción consiste de los siguientes dos pasos: (i) aplicación del mecanismo de generación, (ii) aplicación del criterio de aceptación.

Sea C_k el valor del parámetro de control y L_k el número de transiciones generadas en la k-ésima iteración del algoritmo de Metropolis. Entonces el algoritmo de recocido simulado puede describirse en pseudo-código.

El algoritmo de la Figura 5.1 comienza llamando a un procedimiento de inicialización donde se definen la solución inicial, la temperatura inicial y el número inicial de generaciones necesarias para alcanzar el equilibrio térmico para la temperatura inicial. La parte modular del algoritmo consta de dos ciclos. El externo Repite ... hasta y el interno para ...finpara. El ciclo interno mantiene fijo el parámetro de control hasta que se generan L_k soluciones y se acepta o se rechaza la solución generada conforme los criterios de aceptación ya discutidos. El ciclo externo disminuye el valor de la temperatura mediante el procedimiento CALCULA-CONTROL y calcula el número de soluciones a generar para alcanzar equilibrio térmico mediante el procedimiento CALCULA-LONGITUD. Este ciclo finaliza cuando la condición de paro se cumple.

Un rasgo característico del algoritmo de recocido simulado es que, además de aceptar mejoramientos en el costo, también acepta soluciones más malas en costo. Inicialmente. para valores grandes donde c, puede aceptar soluciones con un valor objetivo mucho mayor a la solución actual; cuando c decrece, únicamente pequeñas desviaciones serán aceptadas y finalmente, cuando el

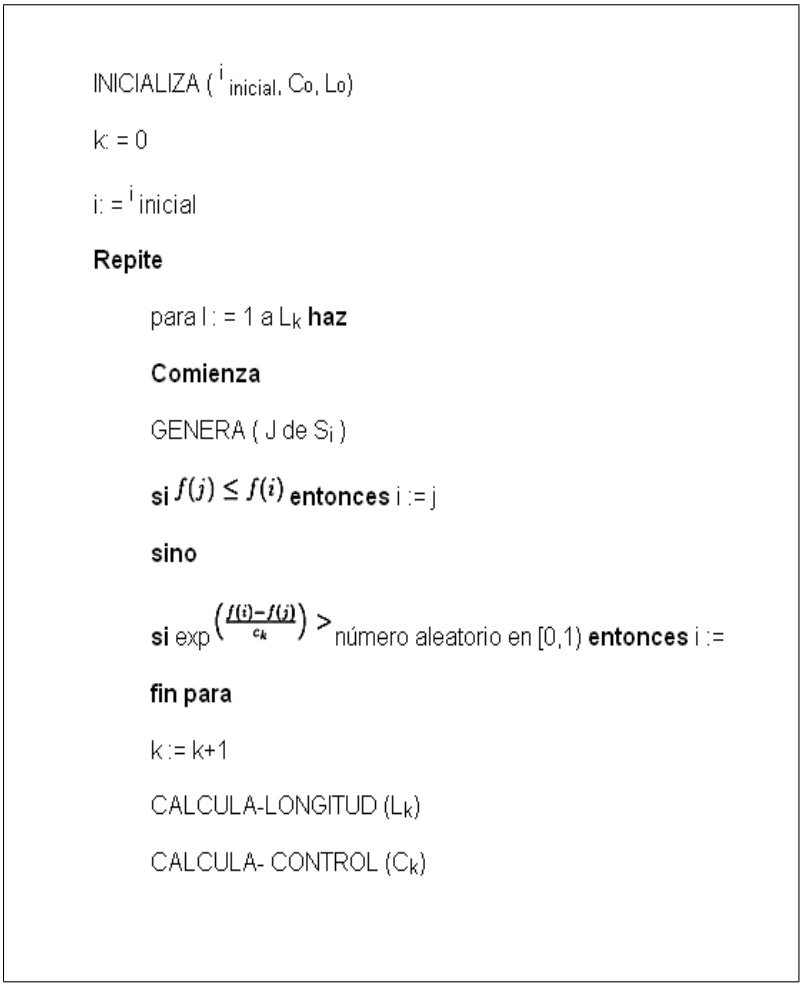

Figura 5.1: Algoritmo de Recocido simulado

valor de c se aproxima a cero, no se aceptarán desviaciones. Este hecho significa que el algoritmo de recocido simulado, en contraste con el algoritmo de búsqueda local, puede escapar de mínimos locales además de exhibir los rasgos favorables de los algoritmos de búsqueda local; es decir, simplicidad y aplicabilidad general.

5.1.2. El recocido simulado como una variante del "Hill-Climbing"

El recocido simulado es una variación del ascenso a colina. Al inicio, este algoritmo, permite explorar una buena parte del espacio de estado, de tal forma que la solución final puede resultar insensible al estado inicial. En consecuencia, la probabilidad de quedar atrapado en un máximo local, en una meseta o en un risco, se hace mínima.

El recocido simulado es un proceso computacional que refleja los pasos establecidos en el proceso físico de tratamiento térmico de materiales. En el recocido, por ejemplo, un metal es llevado a elevados niveles energéticos, hasta que alcanza su punto de fusión. Luego, gradualmente es enfriado hasta alcanzar un estado sólido, de mínima energía, previamente definido. Por su naturaleza, este algoritmo debe ser formulado como un descenso a valle en el que la función objetivo es el nivel energético.

Las sustancias físicas usualmente se mueven desde configuraciones de alta energía a las de menor energía, así que el descenso al valle, ocurre en forma natural. Pero, eventualmente pueden haber transiciones a niveles energéticos más altos, con una probabilidad dada por:

$$p = e^{frac\Delta EkT} \tag{5.5}$$

donde $\Delta E = E_{nuevo} - E_{actual}$, T es la Temperatura absoluta y k es la constante de Boltzmann.

El procedimiento que se va a seguir para enfriar el sistema, se llama programa de recocido. Su forma óptima depende de cada tipo de problema y usualmente se lo descubre empíricamente. El programa de recocido, debe incluir los siguientes ingredientes:

1. El valor inicial de la temperatura.
2. El criterio que será utilizado para decidir cuando reducir la temperatura.

3. La cantidad en que será reducida la temperatura.

4. Cuando finalizar el proceso.

El algoritmo para el recocido simulado, es ligeramente diferente del procedimiento simple de ascenso a colina. Las diferencias son:

1. Se debe respetar el programa de recocido.

2. Movimientos a estados peores que el actual, pueden ser aceptados.

3. Se debe mantener, además del estado actual, el mejor estado encontrado hasta el momento. Así, si por alguna razón el estado final resulta peor que el mejor encontrado anteriormente, siempre será posible regresar a él

5.2. Redes de Boltzman

La máquina de Boltzmann es útil para el reconocimiento de patrones, intentando recuperar información no disponible de un estado (es decir completando las partes que no conocemos) Las redes de Boltzmann consisten en neuronas conectadas entre sí que pueden estar conectadas bidireccionalmente y que tienen salidas binarias. Las neuronas se distinguen en dos grupos: las visibles y las no visibles. Las primeras constituyen la interfaz de la red y las segundas son sólo para un mejor desempeño de la red. Hay dos arquitecturas principales para las máquinas de Boltzmann: Completación de Boltzmann y la red de Boltzmann de entrada-salida. La diferencia está en la capa visible, pues en las de completación sólo hay un tipo y están conectadas entre ellas de forma bidireccional (todas las neuronas con todas, inclusive las no visibles); en la red de entrada-salida las visibles se dividen en las neuronas de entrada y en las de salida, siendo las de entrada únicamente conectadas unidireccionalmente con la capa no visible y las neuronas de salida. Las de salida se conectan con todas las que no son de entrada de forma bidireccional. La característica principal de las redes de Boltzmann es que la función de salida es estocástica con probabilidad:

$$P_k = \frac{1}{1 + e^{-\frac{net_k}{T}}} \tag{5.6}$$

donde net_k es la diferencia de energía en el sistema cuando $x_k = 0$ y $x_k = 1$ (donde x_k es la salida de la k-ésima unidad) y está dada por la siguiente expresión:

$$net_k = \sum_{j=1, j\neq k}^{n} w_{kj} x_j \tag{5.7}$$

El parámetro T representa la temperatura del sistema. Para simular el proceso del annealing usamos el siguiente algoritmo. Sea x' el vector de entrada con componentes desconocidos.

1. Asignar los valores conocidos del vector de entrada x' a las neuronas visibles.

2. Imputar todas los valores desconocidos y de las neuronas no visibles con valores aleatorios en el conjunto 0,1.

3. Seleccionar una unidad x_k aleatoriamente y calcular su valor de entrada a la red (net_k)

4. Sin importar el valor actual de la unidad seleccionada, asignar el valor $x_k = 1$ con probabilidad P_k (definida anteriormente). Esta elección estocástica se puede implementar con una comparación entre P_k y un valor z seleccionado al azar de la distribución uniforme. Con z entre 0 y 1 y menor o igual que P_k.

5. Repetir los pasos 3 y 4 hasta que todas la unidades tengan una probabilidad de ser seleccionadas para una actualización. Este número de actualización de unidades se llama ciclo de procesamiento. El realizar un ciclo completo no garantiza que todas la unidades hayan sido actualizadas.

6. Repetir el paso 5 hasta llegar al equilibrio térmico.*

7. Bajar la temperatura T y repetir pasos 3 a 7.

La convergencia al estado de mínima energía se asegura por el teorema de German y German que asegura que si las temperaturas del k-ésimo paso del algoritmo está acotada inferiormente por $\frac{T_0}{log(1+k)}$ donde T_0 es una constante suficientemente grande se alcanzará un estado de energía con diferencia ϵ al estado de mínima energía.

El algoritmo se detiene cuando T se ha reducido hasta un valor pequeño. Cuando esto sucede la red se ha estabilizado y el resultado final será las salidas de las neuronas visibles. Esperamos que el resultado final sea un vector x que tenga todos sus componentes conocidos.

5.2.1. Aprendizaje en las máquinas de Boltzmann

El aprendizaje en la máquina de Boltzmann se fundamenta en el annealing simulado y de ahí su origen estocástico. Utiliza además, el método de descenso de gradiente sobre la función de divergencia asimétrica definida anteriormente.

Dado que las salidas de las neuronas son de carácter probabilístico, el aprendizaje de Boltzmann intentará distribuir estas salidas de acuerdo a la muestra de los vectores utilizados en el proceso de aprendizaje. El problema es, ahora, determinar la distribución de los vectores de entrenamiento. Como en la mayoría de los casos no tenemos más información para elegir esta distribución elegiremos la uniforme.

Distingamos dos modos de operar de la red: el primero, tendiendo las neuronas visibles fijas y el segundo, con dichas neuronas libres. En ambos casos el objetivo es que después de realizar el proceso de annealing se espere un estado de energía mínimo.

Las cantidades p_{ij}^{+} y p_{ij}^{-} representan la probabilidades de coocurrencia de que las neuronas i y j estén activas ($x_i = 1$ y $x_j = 1$) al mismo tiempo. En el algoritmo estimaremos estas cantidades y así modificar los pesos. Ahora podemos dar el algoritmo:

1. Fijar algún vector de entrenamiento para las neuronas visibles
2. Realizar el proceso de annealing hasta que se alcance el equilibrio a la temperatura mínima
3. Realizar este proceso varias veces y para cada uno de ellos determinar qué pares de neuronas conectadas están activas simultáneamente
4. Estimar la coocurrencia (clamped) para el vector de entrenamiento
5. Repetir de 1 a 4 para cada vector de entrenamiento y estimar p_{ij}^{+}.
6. Liberar las neuronas visibles y realizar el proceso de annealing hasta que se alcance el equilibrio a la mínima temperatura

7. Realizar este proceso muchas veces y después de cada una determinar los pares de neuronas conectadas que están activas simultáneamente
8. Estimar la coocurrencia con el paso 7
9. Repetir del 6 al 8 el mismo número de veces que se repitió el paso 5 y estimar p_{ij}^{-}
10. Calcular y aplicar los cambios de pesos $W_{ij} = \epsilon(p_{ij}^{+} - p_{ij}^{-})$
11. Repetir del 1 al 10 hasta que ΔW_{ij} sea muy pequeño.

5.2.2. Costos computacionales de la Máquina de Boltzmann

El gran problema de la máquina de Boltzmann es la lentitud de convergencia y complejidad del algoritmo, pues dentro de él hay 4 ciclos anidados:

- Ajustar los pesos hasta alcanzar un mínimo satisfactorio.
- Cada iteración que estima p_{ij} en el modo libre y para el modo fijo.
- Para cada estimación, realizar el proceso de annealing.
- A cada temperatura, buscar equilibrio térmico.

Además, dentro del proceso de annealing, no podemos realizar el descenso de temperaturas tan brusco como se quiera, pues para que la red se estabilice en un nivel de energía suficientemente bajo (alguno que otorgue una estabilidad suficiente a la red), las temperaturas en cada iteración están acotadas inferiormente por el teorema de German y German.

5.3. Algoritmos genéticos

Los algoritmos genéticos se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una Selección de acuerdo con algún criterio, en función del cual

se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados. También es denominado algoritmos evolutivos, e incluye las estrategias evolutiva, la programación evolutiva y la programación genética.

Un algoritmo genético es un método de búsqueda dirigida basada en probabilidad. Bajo una condición muy débil (que el algoritmo mantenga elitismo, es decir, guarde siempre al mejor elemento de la población sin hacerle ningún cambio) se puede demostrar que el algoritmo converge en probabilidad al óptimo. En otras palabras, al aumentar el número de iteraciones, la probabilidad de tener el óptimo en la población tiende a 1 (uno).

Los algoritmos genéticos establecen una analogía entre el conjunto de soluciones de un problema, llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados los genes. Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de interacciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud. Las siguientes generaciones (nuevos cromosomas), llamada descendencia, se forman utilizando dos operadores genéticos, de sobrecruzamiento y de mutación.

Los algoritmos genéticos son de probada eficacia en caso de querer calcular funciones no derivables (o de derivación muy compleja) aunque su uso es posible con cualquier función. Deben tenerse en cuenta también las siguientes consideraciones:

- Si la función a optimizar tiene muchos máximos/mínimos locales se requerirán más iteraciones del algoritmo para .ªsegurar.el máximo/mínimo global.
- Si la función a optimizar contiene varios puntos muy cercanos en valor al óptimo, solamente podemos .ªsegurar"que encontraremos uno de ellos (no necesariamente el óptimo).

Un algoritmo genético puede presentar diversas variaciones, dependiendo de cómo se aplican los operadores genéticos (cruzamiento, mutación), de cómo se realiza la selección y de cómo se decide el reemplazo de los individuos para formar la nueva población. En general, el pseudocódigo consiste de los siguientes pasos:

- Inicialización: Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.

- Evaluación: A cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber qué tan "buena.es la solución que se está codificando.

- Condición de término El AG se deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el AG un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:

  Selección Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.

  Sobrecruzamiento El cruzamiento es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.

  Mutación modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.

  Reemplazo una vez aplicados los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la generación siguiente

En la Figura 5.2 se muestra un algoritmo genético generalizado, en la misma i representa la inicialización, $f(X)$ la evaluación, ? es la condición de término, Se es la selección, Cr es el cruzamiento, Mu la mutación, Re el reemplazo, y X^* es la mejor solución.

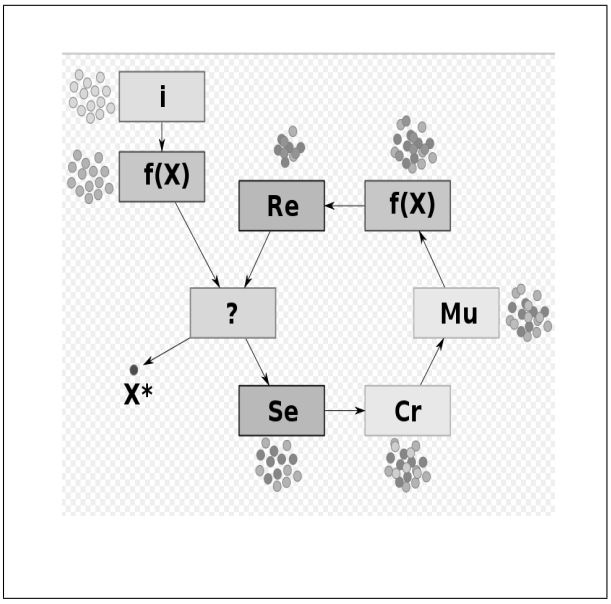

Figura 5.2: Algoritmos Genéticos

5.4. Programación genética

Programación Genética consiste en la evolución automática de programas usando ideas basadas en la selección natural (Darwin). No solo se ha utilizado para generar programas, sino que cualquier otro tipo de soluciones cuya estructura sea similar a la de un programa. Por ejemplo, fórmulas matemáticas, circuitos electrónicos. La evolución se produce en la naturaleza gracias a que:

- Existe reproducción entre individuos de una población.
- Las características de los individuos afectan su probabilidad de supervivencia.
- Existe herencia.
- Existen recursos finitos, que ocasiona competencia.

En programación genética se busca que poblaciones de programas evolucionen, transmitiendo su herencia de manera que se adapten mejor al medio. Los mejores individuos tienen mayores probabilidades de reproducirse. La medida de calidad del individuo dependerá del tipo de problema. Un algoritmo de programación genética sigue el siguiente esquema:

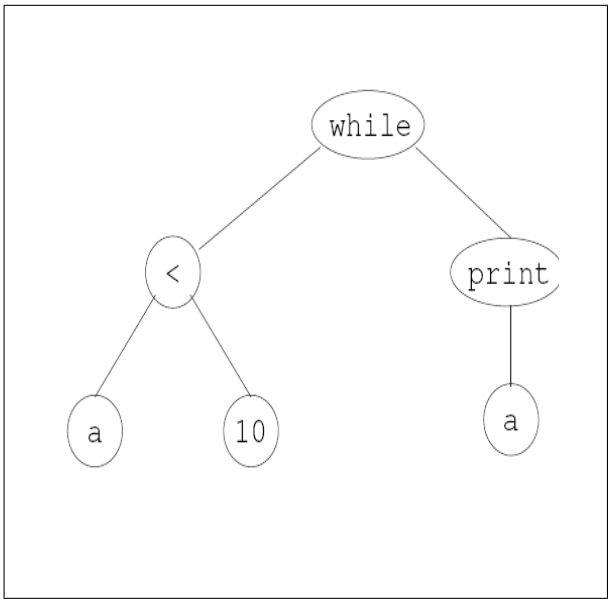

Figura 5.3: Arbol que representa código

1. Genera una población inicial.

2. Mientras no se cumple el criterio de terminación:

   Seleccionar individuos (para reproducción y eliminación), considerando su calidad.

   Combinar y/o variar individuos nuevos.

   Agregar y eliminar individuos.

En programación genética, los programas (o individuos) se representan como árboles. Es así como el segmento de código

```
while  (a<10) {
    print(a);
    a++
}
```

puede representarse por el árbol de la Figura 5.3

El primer paso en programación genética consiste en formar la población inicial de individuos. Uno de los parámetros principales para un algoritmo

genético es el tamaño máximo de un programa. Este límite puede estar impuesto sobre el número de nodos o sobre la profundidad del árbol. Usualmente, se utilizan dos métodos para generar esta población, el método de grow y el full .

5.4.1. El método grow

Sea T el conjunto de terminales y F el conjunto de funciones. Se elige aleatoriamente un elemento de F para que conforme la raíz del árbol. El contenido de los nodos hijos de la raíz se elige desde $F \cup T$. Si el valor elegido es una función, se repite este procedimiento con los hijos. (si el valor elegido es una constante, se termina esa rama del árbol.)

5.4.2. El método full

El método full hace crecer el árbol en forma similar al método grow, pero siempre se eligen elementos del conjunto de funciones, a menos que el nodo esté a profundidad máxima, en cuyo caso solo se eligen elementos de T. El resultado de este método son siempre árboles balanceados de profundidad máxima. Si se usa el número de nodos como límite de tamaño, el crecimiento se termina cuando el tamaño árbol ha alcanzado el límite.

5.4.3. Operadores Genéticos

Los operadores básicos de programación genética son:

Crossover

El operador de cruza (crossover) combina el material genético de individuos intercambiando pedazos dos progenitores para producir dos descendientes. Si los individuos se representan como árboles, se elige al azar un nodo de cada árbol y luego se intercambian los subárboles bajo estos nodos.

Mutación

Actúa sobre un solo individuo, generalmente, uno resultante de un crossover. La mutación act´ua con una probabilidad, en general, muy baja y que es un

parámetro del algoritmo de programación genética. Un tipo de mutación consiste en escoger en forma aleatoria un nodo del árbol y generar bajo ´este otro subárbol (por ejemplo, usando el método grow).

Reproducción

Es el operador más simple de todos. Se selecciona un individuo y se lo duplica, quedando dos copias dentro de la población.

5.4.4. Función de Calidad

La calidad es la medida usada por el algoritmo de programación genética durante la evolución simulada para medir que tan buena es una solución. Una función de calidad se dice estandarizada positiva si es que siempre asigna el valor 0 al individuo que es mejor solución para el problema. Una función de calidad se dice normalizada si es que siempre asigna el valores entre 0 y 1. Si f es una función estandarizada, entonces $g = 1/(1 + f)$ es una función normalizada que asigna valor 1 al individuo de mayor calidad. Si el objetivo de la PG es aprender una función matemática g, entonces una medida de calidad puede ser el error cuadrático medio. Así, I es un conjunto de entradas y $g(i) = o_i$ es la salida para una entrada i, entonces podemos medir la calidad de un programa p por:

$$f_p = \sum_{i \epsilon I} (p_i - o_i)^2 \tag{5.8}$$

donde p_i es la salida del programa ante la entrada i. La función de calidad depende claramente del tipo de problema. Por ejemplo, en un problema de clasificación en bases de datos, la calidad puede estar medida por el número de ejemplos bien clasificados.

5.4.5. Selección

La selección es el proceso por el cual se transmiten individuos de una generación a otra. Hay distintos tipos de selección tipos de selección. El primero de ellos se basa en algoritmos genéticos. Para una población de N individuos:

1. Elegir a dos individuos progenitores, privilegiando los con mejor calidad. Esto se hace eligiendo al individuo i con probabilidad.

$$P(i) = \frac{f(i)}{\sum_i f(i)} \tag{5.9}$$

   donde $f(i)$ es la medida de calidad (para que esto funcione puede ser necesario normalizar la función antes de hacer este cálculo)

2. Aplicar crossover (si corresponde, probabilísticamente).

3. Aplicar mutación (si corresponde, probabilísticamente).

4. Reproducir.

5. Repetir hasta completar una nueva generación de N individuos.

Otra técnica consiste en efectuar torneos:

1. Elegir dos grupos de n individuos aleatoriamente de la población.

2. Seleccionar al mejor elemento del primer grupo (padre), y al mejor elemento del segundo (madre).

3. Aplicar crossover (si corresponde, probabilísticamente).

4. Aplicar mutación (si corresponde, probabilísticamente).

5. Los dos nuevos individuos reemplazan a los peores de cada uno de los grupos.

Esta última técnica es generalmente preferida por razones de eficiencia.

Capítulo 6

Técnicas de Agrupamiento

Un algoritmo de agrupamiento (en inglés, clustering) es un procedimiento de agrupación de una serie de vectores de acuerdo con un criterio de cercanía. Esta cercanía se define en términos de una determinada función de distancia, como la euclídea, aunque existen otras más robustas o que permiten extenderla a variables discretas.

Generalmente, los vectores de un mismo grupo (o clústers) comparten propiedades comunes. El conocimiento de los grupos puede permitir una descripción sintética de un conjunto de datos multidimensional complejo. De ahí su uso en minería de datos. Esta descripción sintética se consigue sustituyendo la descripción de todos los elementos de un grupo por la de un representante característico del mismo.

En algunos contextos, como el de la minería de datos, se lo considera una técnica de aprendizaje no supervisada puesto que busca encontrar relaciones entre variables descriptivas pero no la que guardan con respecto a una variable objetivo.

Existen diversas técnicas de agrupamiento. Se dividen en dos grandes categorías:

1. Jerárquicas, que construyen una jerarquía de grupos escindiéndolos iterativamente.

2. De particionamiento, en los que el número de grupos se determina de antemano y las observaciones se van asignando a los grupos en función de su cercanía.

Existen diversas implementaciones de algoritmos concretos. Por ejemplo,

el de las k-medias, de particionamiento. Es uno de los más antiguos pero uso extendido a pesar de sus carencias y falta de robustez.

6.1. K-medias

Es uno de los más simples y conocidos algoritmos de agrupamiento, sigue una forma fácil y simple para dividir una base de datos dada en k grupos (fijados a priori). La idea principal es definir k centroides (uno para cada grupo) y luego tomar cada punto de la base de datos y situarlo en la clase de su centroide más cercano. El próximo paso es recalcular el centroide de cada grupo y volver a distribuir todos los objetos según el centroide más cercano. El proceso se repite hasta que ya no hay cambio en los grupos de un paso al siguiente. El problema del empleo de estos esquemas es que fallan cuando los puntos de un grupo están muy cerca del centroide de otro grupo, también cuando los grupos tienen diferentes tamaños y formas.

$$P(c_i|x_j) = \begin{cases} 1 & ||x_j - m_i|| < ||x_j - m_k|| \ \forall k \neq i \\ 0 & \text{en caso contrario} \end{cases} \tag{6.1}$$

1. Extraer el número de muestras n a utilizar y el número de clases c.
2. Inicializar los centros de las clases m_i y las probabilidades $P(c_i|x_j)$, $i = 1...c$; $j = 1...n$
3. Normalizar las probabilidades por medio de la ecuación $\sum_{i=1}^{c} P(c_i|x_j) = 1$; $i = 1...c$
4. Obtener m_i de acuerdo con la ecuación (6.6)
5. Recalcular $P(c_i|x_j)$ por medio de la ecuación (6.1).
6. Repetir los pasos 3 a 5 hasta que m_i y $P(c_i|x_j)$ no cambien o el cambio sea pequeño.

1. Extraer el número de muestras n a utilizar y el número de clases c.
2. Inicializar los centros de las clases m_i y las probabilidades $P(c_i|x_j)$, $i = 1...c$; $j = 1...n$

3. Normalizar las probabilidades por medio de la ecuación $\sum_{i=1}^{c} P(c_i|x_j) = 1$; $i = 1...c$
4. Obtener m_i de acuerdo con la ecuación (6.6)
5. Recalcular $P(c_i|x_j)$ por medio de la ecuación (6.7).
6. Repetir los pasos 3 a 5 hasta que m_i y $P(c_i|x_j)$ no cambien o el cambio sea pequeño.

Con el fin de acelerar la convergencia hemos añadido una condición adicional para la parada del algoritmo en el paso 6. En efecto, el proceso iterativo también puede detenerse si se ha alcanzado un cierto número K de iteraciones aún cuando no se hayan cumplido ninguna de las dos condiciones de cambio dadas en dicho paso. Finalmente, para determinar si un cambio es pequeño o no, se introduce un umbral ϵ en base al cual se determina si los centros o los grados de pertenencia se pueden considerar como que han cambiado o no respecto de la iteración anterior como exige el algoritmo. Es decir, si los valores de los cambios están por debajo de dicho umbral se considera que no cambian.

6.2. K-medias dinámico

El método de K-medias dinámico es una variante del método de K-medias clásico (KMC), el problema con KMC es que es necesario conocer el número de grupos o clases, es decir k antes de comenzar el procedimiento. Este requerimiento limita mucho las aplicaciones, suponga por ejemplo que se tiene un discurso grabado en un archivo de audio y queremos agrupar los diferentes sonidos fonéticos que existen en dicho discurso, si bien hay estudios acerca de los sonidos posibles que puede emitir un parlante en español mexicano, la verdad es que no estamos seguros de que en todos los discursos se usen todos los sonidos.

La solución que implementa el algoritmo de K-medias dinámico consiste en:

1. Realizar agrupamiento mediante el algoritmo de k-medias clásico
2. Dividir las clases demasiado pobladas (o que ocupen una región demasiado amplia) en dos clases (generar un centroide nuevo cerca del centroide del grupo a dividir

3. Fusionar las clases demasiado despobladas o que ocupen una región demasiado pequeña)

4. repetir desde el paso 1 hasta que ya no se generen nuevos grupos ni se elimine grupo alguno

Obviamente el problema con este método es que es requiere hacer muchos mas cómputos que el KMC.

6.3. K-medias difuso (KMD)

Dado un conjunto de c clases $\{c_1, c_2, ..., c_c\}$ y un conjunto $X = \{x_1, x_2, ..., x_n\}$ de n muestras, bajo la perspectiva difusa se considera que dada una muestra x_j, ésta puede pertenecer a más de una clase. Esta pertenencia se mide por lo que se conoce como grado de pertenencia, que designaremos como $P(c_i|x_j)$ indicando en qué medida la muestra x_j pertenece a la clase ci. Las clases se caracterizan por tener asociado un vector denominado centro de la clase, en el caso que nos ocupa para las c clases serían los siguientes: $m_1, m_2, ..., m_c$.

El algoritmo KMD busca un mínimo de la función de coste global heurística siguiente:

$$J_D = \sum_{i=1}^{c} \sum_{j=1}^{n} [P(c_i|x_j)]^b ||x_j - m_i||^2 \tag{6.2}$$

donde b es un parámetro libre, elegido para ajustar el solapamiento o mezcla de las diferentes clases, si b es 0, J_D es un mero criterio de suma de errores al cuadrado con cada patrón asignado a una única clase. Para $b > 1$, el criterio permite que cada patrón pertenezca a más de una clase. Los grados de pertenencia de cada patrón a las diferentes clases se normalizan como sigue:

$$\sum_{i=1}^{c} P(c_i|x_j) = 1 \tag{6.3}$$

La solución se obtiene encontrando los mínimos de J_D siguientes:

$$\partial J_D / \partial m_i = 0 \tag{6.4}$$

$$\partial J_D/\partial P(c_j) = 0 \tag{6.5}$$

donde $P(c_j)$ es la probabilidad a priori de que un patrón pertenezca a la clase c_j. Lo anterior conduce a las siguientes soluciones:

$$m_i = \frac{\sum_{j=1}^{n}[P(c_i|x_j)]^b x_j}{\sum_{j=1}^{n}[P(c_i/x_j)]^b} \tag{6.6}$$

$$P(c_i|x_j) = \frac{(1/d_{ij})^{1|(b-1)}}{\sum_{r=1}^{c}(1/d_{rj})^{1/(b-1)}} \tag{6.7}$$

donde $d_{ij} = ||x_j - m_i||^2$ y $b > 1$ es un escalar denominado *peso exponencial* de suerte que cuanto mayor sea su valor, menor es la contribución de las muestras a la función objetivo utilizada para obtener las ecuaciones (6.6) y (6.7); el número de clases c debe ser inferior al número de muestras y superior a 1, es decir $1 < c < n$; d_{ij} es la distancia Euclídea al cuadrado, si bien se podrían utilizar otras distancias. Debido a que las ecuaciones (6.6) y (6.7) rara vez tienen soluciones analíticas, los centros de las clases y las probabilidades se estiman iterativamente de acuerdo con el siguiente algoritmo:

1. Extraer el número de muestras n a utilizar y el número de clases c.
2. Inicializar los centros de las clases m_i y las probabilidades $P(c_i|x_j)$, $i = 1...c$; $j = 1...n$
3. Normalizar las probabilidades por medio de la ecuación $\sum_{i=1}^{c} P(c_i|x_j) = 1$; $i = 1...c$
4. Obtener m_i de acuerdo con la ecuación (6.6)
5. Recalcular $P(c_i|x_j)$ por medio de la ecuación (6.7).
6. Repetir los pasos 3 a 5 hasta que m_i y $P(c_i|x_j)$ no cambien o el cambio sea pequeño.

6.4. Vecinos mas cercanos mutuos

La noción de proximidad basada en vecindad fué modificada por Gowda y Krishna en 1978 [6] para medir la "vecindad mutua" de dos patrones.

Si x_j es el p-ésimo vecino mas cercano de x_i y a la vez x_i es el q-ésimo vecino mas cercano de x_j entonces el *valor de vecindad mutua* (VVM) entre x_i y x_j se define como $q+p$, mientras mas pequeño sea el VVM, mas parecidos serán los patrones. Esta noción de similaridad es mas fuerte que aquella que cuenta el número de vecinos compartidos de Jarvis y Patrick. El algoritmo de agrupamiento de los vecinos mutuos es el siguiente:

1. Determine los k vecinos mas cercanos de cada patrón
2. Calcule el VVM para cada pareja de patrones. Si los patrones x_i y x_j no son vecinos mutuos para cierto valor de k asigne a VVM(x_i y x_j) un valor arbitrariamente grande
3. Identifique todas las parejas de patrones con VVM de 2. Junte cada una de esas parejas en un grupo comenzando con el par que tenga menor distancia entre ellos
4. Repita el paso 3 para VVM de $3, 4, ..., 2k$ para generar una jerarquía

El parámetro k que controla la profundidad de la vecindad es crucial para el desempeño de este algoritmo. Valores pequeños de k producen grupos muy sólidos pero quizás demasiados mientras que valores grandes de k producen pocos grupos pero quizás débiles. De hecho k puede escogerse lo suficientemente grande como para que el algoritmo regrese un solo grupo. Gowda y Krishna demostraron que el algoritmo puede identificar grupos no esféricos, grupos que no se pueden separar linealmente, grupos con poblaciones desiguales y grupos con puentes de baja deensidad para $k = 5$ en dos dimensiones. Sin embargo no existe una heurística para determinar k para conjuntos arbitrarios de datos

6.5. Agrupamiento Jerarquico

Un método de ordenamiento jerárquico es un procedimiento para transformar una matriz de proximidad en una secuencia de particiones anidadas. Definimos H, el conjunto de n objetos a ser agrupados

$$H = \{x_1, x_2, ...x_n\} \tag{6.8}$$

Una partición C de H parte a H en m subconjuntos disjuntos $\{C_1, C_2, ..., C_m\}$ de tal manera que

$$C_i \cap C_j = \emptyset \ \ \forall \ \ i \neq j \tag{6.9}$$

$$\cap_{i=1}^{m} C_i = H \tag{6.10}$$

Un agrupamiento jerárquico es una secuencia de particiones en la que cada partición está anidada en la siguiente partición de la secuencia. Un algoritmo *aglomerativo* comienza por la *partición disjunta* la cual pone a cada uno de los n objetos en su propio grupo individual, el algoritmo aglomerativo indica como utilizar la matriz de proximidad para fusionar dos o mas de estos grupos triviales anidando de esta manera a la partición en la siguiente. El proceso se repite una y otra vez reduciéndose el número de grupos hasta que queda un solo grupo con n objetos. A la partición que consiste de un solo grupo se le conoce como *partición conjunta*. Un *Algoritmo divisivo* procede exactamente al reves.

Una *dendograma* es una estructura en forma de árbol que muestra gráficamente la manera en que un algoritmo de agrupamiento funciona. En la Figura 6.1 se , proporciona un ejemplo de dendograma cualquier corte horizontal hecho al dendrograma define un agrupamiento en particular.

Los algoritmos de agrupamiento jerárquico pueden ser de liga sencilla o de ligado completo, de manera que aparte de establecer si un algoritmo de agrupamiento jerárquico es aglomerativo o divisivo, es necesario indicar el tipo de ligado que utiliza.

Bajo el esquema de ligado sencillo, un objeto se agrega a un grupo si la distancia a cualquiera de los objetos que forman el grupo es menor a cierto umbral.

Bajo el esquema de ligado completo, un objeto se agrega a un grupo si la distancia a cada uno de los objetos que forman el grupo es menor a cierto umbral.

Un grafo de umbral para una colección de n objetos es un grafo no-dirijido con justamente n nodos dado que cada nodo representa un objeto en el que existe una arista entre dos nodos si la distancia entre los objetos correspondientes es menor al umbral definido para tal grafo. Considere por ejemplo la siguiente matriz de proximidad de una colección de 5 objetos:

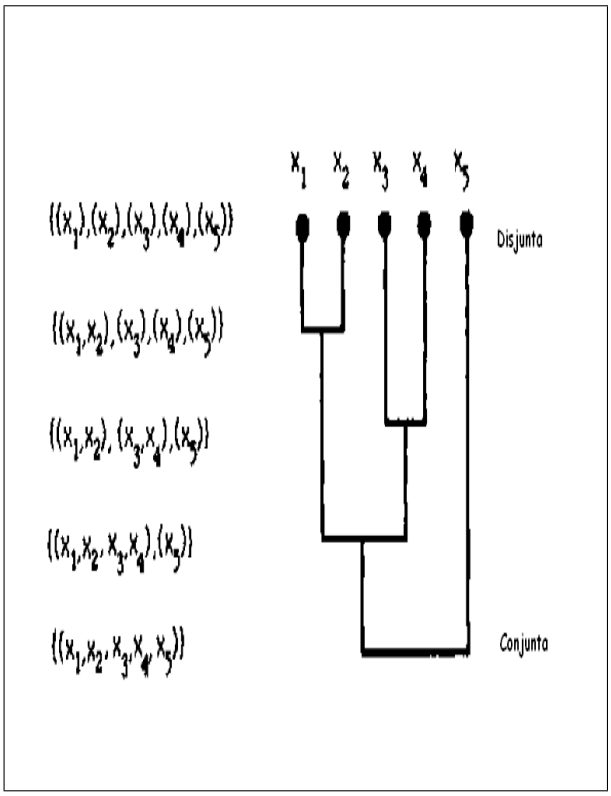

Figura 6.1: Dendograma

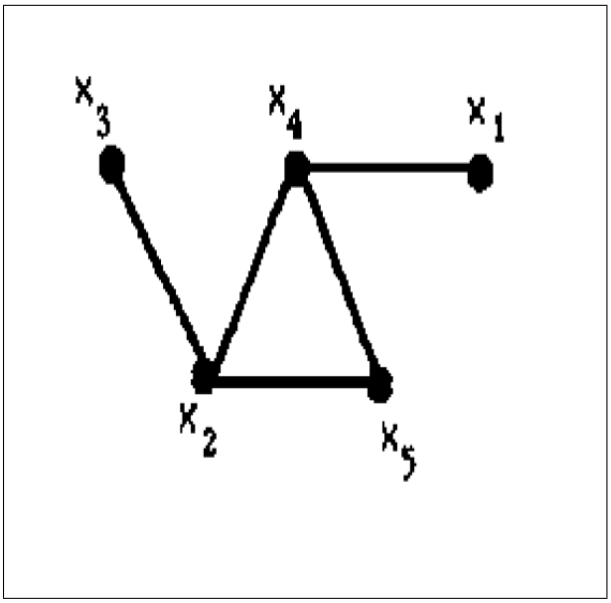

Figura 6.2: Grafo Umbral G(5) para la matriz de proximidad (6.11)

$$\begin{array}{ccccc} 0 & 6 & 8 & 2 & 7 \\ 6 & 0 & 1 & 5 & 3 \\ 8 & 1 & 0 & 10 & 9 \\ 2 & 5 & 10 & 0 & 4 \\ 7 & 3 & 9 & 4 & 0 \end{array} \tag{6.11}$$

Entonces para el valor umbral 5, obtenemos el grafo G(5) mostrado en la Figura 6.2:

El algoritmo aglomerativo para ligado sencillo consiste en:

1. Comience por el agrupamiento disjunto implicado por el grafo umbral $G(0)$ el cual no contiene aristas y coloca a cada objeto en su grupo individual. Haga $k = 1$

2. Forme el grafo umbral $G(k)$. Si el número de componentes (subgrafos conexos) en $G(k)$ es menor que el número de grupos en el agrupamiento actual, redefina el agrupamiento actual de manera que cada componente de $G(k)$ sea un grupo.

3. Si $G(k)$ consiste de un solo grafo conexo, deténgase, si no, haga $k = k+1$ y vaya al paso 2

6.6. Análisis de componentes principales

El análisis de componentes principales (en español ACP, en inglés, PCA) es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos. Intuitivamente la técnica sirve para determinar el número de factores subyacentes explicativos tras un conjunto de datos que expliquen la variabilidad de dichos datos.

Técnicamente, el PCA busca la proyección según la cual los datos queden mejor representados en términos de mínimos cuadrados. PCA se emplea sobre todo en análisis exploratorio de datos y para construir modelos predictivos. PCA comporta el cálculo de la descomposición en autovalores de la matriz de covarianza, normalmente tras centrar los datos en la media de cada atributo.

El ACP construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente. Para construir esta transformación lineal debe construirse primero la matriz de covarianza o matriz de coeficientes de correlación. Debido a la simetría de esta matriz existe una base completa de vectores propios de la misma. La transformación que lleva de las antiguas coordenadas a las coordenadas de la nueva base es precisamente la transformación lineal necesaria para reducir la dimensionalidad de datos. Además las coordenadas en la nueva base dan la composición en factores subyacentes de los datos iniciales.

Una de las ventajas de ACP para reducir la dimensionalidad de un grupo de datos, es que retiene aquellas características del conjunto de datos que contribuyen más a su varianza, manteniendo un orden de bajo nivel de los componentes principales e ignorando los de alto nivel. El objetivo es que esos componentes de bajo orden a veces contienen el "más importante" aspecto de esa información.

Supongamos que existe una muestra con n individuos para cada uno de los cuales se han medido m variables (aleatorias) F_j. El ACP permite encontrar un número de factores subyacentes $p < m$ que explican aproximadamente el valor de las m variables para cada individuo. El hecho de que existan estos p factores subyacentes puede interpretarse como una reducción de la dimensionalidad de los datos: donde antes necesitábamos m valores para caracterizar a cada individuo ahora nos bastan p valores. Cada uno de los p encontrados se llama *componente principal*, de ahí el nombre del método.

Existen dos formas básicas de aplicar el ACP:

1. Método basado en la matriz de covarianzas, que se usa cuando los datos son dimensionalmente homogéneos y presentan valores medios similares.

2. Método basado en la matriz de correlación, cuando los datos no son dimensionalmente homogéneos o el orden de magnitud de las variables aleatorias medidas no es el mismo.

6.6.1. Método basado en las covarianzas

Es el más usado cuando todos los datos son homogéneos y tienen las mismas unidades. Cuando se usan valores muy variables o magnitudes que tienen unidades resulta más adecuado para interpretar los resultados el método basado en correlaciones, que siempre es aplicable sin restricción alguna.

6.6.2. Método basado en correlaciones

El método parte de la matriz de correlaciones, consideremos el valor de cada una de las m variables aleatorias F_j. Para cada uno de los n individuos tomemos el valor de estas variables y escribamos el conjunto de datos en forma de matriz:

$$(F_j^\beta)_{j=1,...,m}^{\beta=1,...,n}. \tag{6.12}$$

Obsérvese que cada conjunto

$$\mathcal{M}_j = \{F_j^\beta | \beta = 1, ..., n\} \tag{6.13}$$

puede considerarse una muestra aleatoria para la variable F_j. A partir de los $m \times n$ datos correspondientes a las m variables aleatorias, puede construirse la matriz de correlación muestral viene definida por:

$$\mathbf{R} = [r_{ij}] \in M_{m\times m} \qquad \text{con } r_{ij} = \frac{\text{cov}(F_i, F_j)}{\sqrt{\text{var}(F_i)\text{var}(F_j)}} \tag{6.14}$$

Puesto que la matriz de correlaciones es simétrica entonces resulta diagonalizable y sus valores propios λ_i, verifican:

$$\sum_{i=1}^{m} \lambda_i = 1 \tag{6.15}$$

Debido a la propiedad anterior estos m valores propios reciben el nombre de pesos de cada uno de los m componentes principales. Los factores principales identificados matemáticamente se representan por la base de vectores propios de la matriz $\mathbf{R}$. Está claro que cada una de las variables puede ser expresada como combinación lineal de los vectores propios o componentes principales.

6.6.3. Limitaciones

La aplicación del ACP está limitada por varias asunciones

- Asunción de linealidad: Se asume que los datos observados son combinación lineal de una cierta base.
- Importancia estadística de la media y la covarianza: el ACP utiliza los vectores propios de la matriz de covarianzas y sólo encuentra las direcciones de ejes en el espacio de variables considerando que los datos se distribuyen de manera gaussiana.

6.6.4. Ejemplos

- Una análisis consideró las calificaciones escolares $n = 15$ estudiantes en m = materias (lengua, matemáticas, física, inglés, filosofía, historia, química, gimnasia). Los dos primeros componentes principales explicaban juntos el 82,1 % de la varianza. El primer de ellos parecía fuertemente correlacionado con las materias de humanidades (lengua, inglés, filosofía, historia) mientras que el segundo aparecía relacionado con las materias de ciencias (matemáticas, física, química). Así parece que existe un conjunto de habilidades cognitivas relacionadas con las humanidades y un segundo relacionado con las ciencias, estos dos conjuntos de habilidades son estadísticamente independientes por lo que un alumno puede puntuar alto en sólo uno de ellos, en los dos o en ninguno.
- Un análisis de metodología docente, consideró las calificaciones de $n = 54$ estudiantes de la facultad de Biología de la ULA y $m = 8$ tipos de

habilidades. El primer factor principal que explicaba las calificaciones era la inteligencia del estudiante y en segundo lugar la metodología de aprendizaje usada.

- Una análisis de 11 indicadores socieconómicos de 96 países, reveló que los resultados podían explicarse en alto grado a partir de sólo dos componentes principales, el primero de ellos tenía que ver con el nivel de PIB total del país y el segundo con el índice de ruralidad.

6.7. Redes auto-organizantes de Kohonen

Los mapas autoorganizados o SOM (Self-Organizing Map), también llamados redes de Kohonen son un tipo de red neuronal no supervisada, competitiva, distribuida de forma regular en una rejilla de, normalmente, dos dimensiones, cuyo fin es descubrir la estructura subyacente de los datos introducidos en ella. A lo largo del entrenamiento de la red, los vectores de datos son introducidos en cada neurona y se comparan con el vector de peso característico de cada neurona. La neurona que presenta menor diferencia entre su vector de peso y el vector de datos es la neurona ganadora (o BMU) y ella y sus vecinas verán modificados sus vectores de pesos.

Las neuronas de la SOM están distribuidas en forma de rejilla gular de una o dos dimensiones, dependiendo de la manera en que se quieran visualizar los datos. Las más comunes son las de dos dimensiones. Rejillas de dimensiones superiores son posibles, aunque son más difíciles de interpretar.

En las SOM de dos dimensiones, se pueden distinguir dos tipos de rejillas:

- Rejilla hexagonal: en ella cada neurona tiene seis vecinos (excepto los extremos).

- Rejilla rectangular: cada neurona tiene cuatro vecinos.

Cada neurona de la red tiene asociado un vector de pesos (o de prototipo) de la misma dimensión que los datos de entrada. Éste sería el espacio de entrada de la red, mientras que el espacio de salida sería la posición en el mapa de cada neurona.

Las neuronas mantienen con sus vecinas relaciones de vecindad, las cuales son claves para conformar el mapa durante la etapa de entrenamiento. Esta relación viene dada por una función.

En cada paso se introduce un vector de datos en cada neurona y se calcula la "similitud.entre éste y el vector de peso de cada neurona. La neurona más parecida al vector de entrada es la neurona ganadora (o BMU, Best-Matching Unit, Unidad con mejor ajuste). Para medir la similaridad se utiliza usualmente la distancia euclídeana. Tras ello, los vectores de pesos de la BMU y de sus vecinos son actualizados, de tal forma que se acercan al vector de entrada.

Bibliografía

[1] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," HP Labs Tech, Tech. Rep. HPL2003-4, 2003.

[2] M. A. Group, *Text of ISO/IEC Final Draft International Standar 15938-4 Information Technology - Multimedia Content Description Interface - Part 4: Audio*, July 2001.

[3] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," Department of Electrical Engineering and Computer Science U.C. Berkeley, Tech. Rep. TR-97-021, April 1998.

[4] R. L. Rabiner, "A tutorial on hidden markov models and selected aplications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* John Wiley and Sons Inc., 2001.

[6] A. Jain and R. Dubes, *Algorithms for clustering Data.* Prentrice Hall, 1988.