Aprendizaje Automático

La idea del aprendizaje consiste en utilizar las percepciones no sólo para actuar, sino también para mejorar la habilidad del agente para actuar en el futuro. El aprendizaje entra en juego cuando el agente observa sus interacciones con el mundo y sus procesos de toma de decisiones. Se llama hipótesis a la función que se utiliza para aproximar f(x). Entre múltiples hipótesis consistentes, la mejor es la más simple. Esto es correcto, pero es importante resaltar que la razón principal para esta elección es que la función mas simple es la que mejor generaliza y que las funciones complejas tienden a sobre ajustarse a los datos de entrenamiento.

Reconocimiento de Patrones

Podemos distinguir dos **categorías de reconocimiento**:

- **Reconocimiento perceptual**, como por ejemplo una forma (patrón espacial) o una secuencia (patrón temporal). **Objetos concretos**.
- **Reconocimiento conceptual**, tal como un viejo argumento o la solución de un problema. **Objetos abstractos**.

Puede tener 2 **objetivos**:

- Predicción: La mayoría de los casos, el objetivo es la clasificación.
- Comprender la relación entre las variables.

Patrón: descripción de un objeto definido como una relación entre sus características. Conjunto mínimo de características comunes a un universo de datos que permite identificar dichos datos como pertenecientes a diferentes clases. El reconocimiento de patrones está asociado al reconocimiento perceptual. Cuando una persona percibe un patrón, realiza una inferencia inductiva y asocia esta percepción con algunos conceptos generales o pistas derivados de su experiencia pasada.

- El **problema de reconocimiento** puede ser concebido como el de discriminar, clasificar o categorizar la información de entrada, no entre patrones individuales sino entre poblaciones, por medio de la búsqueda de características o atributos invariantes entre los miembros de una población.
- El **reconocimiento humano** es la estimación del parecido relativo entre datos de entrada y poblaciones conocidas. El **reconocimiento automático** es la clasificación de datos de entrada entre poblaciones mediante la búsqueda de características o atributos invariantes entre los miembros de cada población.

El diseño de un sistema de reconocimiento automático involucra por lo general las tareas siguientes (etapas para reconocimiento de patrones):

- Sensado.
- Extracción de Características.
- Clasificación.

Etapas

1. **Sensado**

Sensado se refiere a la representación de la información obtenida mediante algún tipo de sensor sobre los objetos a ser reconocidos. Cada cantidad medida describe una característica del objeto. Esto puede evidenciarse suponiendo, por ejemplo, que se obtiene información sobre caracteres alfanuméricos. Es la obtención de datos.

En este caso puede considerarse un esquema de medición de grilla como el mostrado en el lazo izquierdo de la figura.

Esta grilla puede representarse mediante un vector patrón como sigue:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad con \quad x_i = 0 \quad o \quad x_i = 1$$

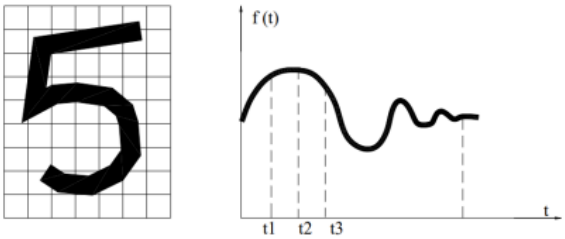


Fig. 2.1: Los objetos se representan mediante un vector patrón.

A la salida es común que la veamos como una escalar, pero también puede ser un vector. Donde se asigna a x_i el valor 1 si el elemento forma parte del carácter y 0 en caso contrario. En el caso de una imagen, x_i sería la intensidad del píxel i.

Variables cuantitativas: Se representan como números del tipo que sea necesario. Por ejemplo, peso, altura y edad son variables cuantitativas y se podrían representar con un float. En una imagen, las intensidades de los píxeles se pueden representar con enteros.

Variables categóricas o cualitativas: Pueden tomar solo un valor dentro de un conjunto limitado de valores. En el caso del nivel educativo, estos valores podrían ser "ninguno", "primario", "secundario", "terciario", "universitario o superior".

2. **Extracción de Características**

Lo que se logra con esto es reducir la dimensión de los patrones, pero con la posibilidad de perder un bajo porcentaje de la información contenida en ellos. Dentro de esta etapa podemos encontrar las siguientes tareas:

- Creación de nuevas características.
- Eliminación de características.
- Reducción de la dimensión.
- Otras transformaciones.

3. **Clasificación**

Determinar a qué clase pertenece cada vector de entrada: Para lograr esto, se requiere diseñar una mecanismo que, dado un conjunto de patrones de los cuales no se conoce a priori la pertenencia a la clase de cada uno de ellos, permita determinar a qué clase pertenece cada patrón.

Generación de límites de decisión entre regiones: Funciones de decisión. Se deben generar tantas funciones de decisión como clases haya. Tomando un dato como entrada,se evalúa en las x funciones, y el valor más alto es la clase a la que pertenece. Cuando 2 funcionesdan el mismo valor, el punto forma parte del límite entre las regiones.

Tipos de problemas reales: Linealmente separables y No linealmente separables. Si el sistema de reconocimiento puede autoajustar ciertos coeficientes internos que definen las funciones discriminantes estaremos en presencia de un sistema adaptivo o con capacidad de aprendizaje.

Definiendo a una clase como una **categoría** determinada por algunos atributos comunes y un patrón como la descripción de cualquier miembro de una categoría que representa a una clase de patrones, decimos que la teoría del Reconocimiento de Patrones se ocupa de buscar la solución al problema general de reconocer miembros de una clase dada en un conjunto que contienen elementos de muchas clases diferentes.

- Heurísticos: basados en la intuición y experiencia.
- Matemáticos:
    - Determinístico: de base estadística pero esta no está explicitada. Clasificadores entrenables.
    - Estadístico: de base estadística explícita. Ej: clasificador de Bayes.
- Sintácticos: basados en relaciones entre los objetos. Expresan una gramática. Útiles cuando los patrones no pueden ser apropiadamente descritos por mediciones (numéricamente).

Resultado de las etapas

El resultado de las etapas de sensado y extracción de características es un conjunto de vectores de características. Cada vector tiene la forma: $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}]$ donde x_i es el valor de la característica i.

Después de obtener los vectores de características se realiza una de las siguientes tareas:

- **Clasificación**. Asignar cada vector de características a una clase. Por ejemplo, para detectar qué carácter aparece en una imagen.
- **Predicción o regresión.** Predecir un valor (contínuo) para cada vector de características. Por ejemplo, para calcular el valor futuro de ciertas acciones bursátiles.

Los métodos de aprendizaje automático que vemos son clasificadores, pero con un mínimo cambio es posible adaptarlo para hacer predicciones.

Función de Decisión

Suponiendo un problema de reconocimiento de c clases distintas denominadas $\omega_1, \ldots, \omega_c$, puede considerarse al espacio de las entradas compuesto por c regiones, donde cada una de las cuales contiene a los elementos de una clase. La solución puede interpretarse como la generación de límites de decisión entre las regiones. Estos límites pueden estar dados por funciones de decisión o funciones discriminantes $d_1(x), \ldots, d_c(x)$, que son funciones escalares de los vectores de entrada. La función que obtiene el mayor valor es la que determina la pertenencia del vector a la clase correspondiente a esa función, es decir, si $d_i(x) > d_j(x)\ para\ i, j = 1, \ldots, c$ y $i \neq j$, entonces x pertenece a la clase ω_i.

En los problemas, muy frecuentes, donde la salida solo puede pertenecer a una clase de un total de dos clases (clasificación binaria), usualmente se utiliza una única función de decisión $d(x)$. Si $d(x > 0)$, x pertenece a una clase, en caso contrario, a la otra.

Clasificador Lineal

El caso más simple de función de decisión es la función de decisión lineal. El modelo de clasificación que implementa esta función se llama clasificador lineal. Para un vector de entradas $x^T = [x_1, \ldots, x_n]$, la función de decisión lineal es $d(x) = w_1 x_1 + \cdots + w_n x_n + w_{n+1}$, donde los coeficientes $w_i \in R$ son los parámetros del clasificador. El vector formado por todos los coeficientes se llama vector de parámetros $w = [w_i, \ldots, w_{n+1}]$. Es útil calcular el valor de la función de decisión como un producto entre vectores, para lo cual el vector x se redefine como $x^T = x_1, \ldots, x_n, 1]$, lo que lleva el nombre de vector de entradas aumentado. Con el nuevo vector x, la función de decisión se puede calcular como $d(x) = w_x$.

Para el caso una clasificación binaria, x pertenecerá a la clase ω_1 si $d(x) < 0$ y pertenecerá a la clase ω_2 si $d(x) > 0$. Entonces, la frontera entre las clases en el espacio de las entradas queda definida por el hiperplano $d(x) = 0$. Los elementos de la clase ω_1 están pintados de color amarillo y los de ω_2 en color azul. La función de decisión se ve como un plano inclinado y la frontera entre las clases se indica con la línea de puntos.

El entrenamiento del modelo se realiza ajustando los valores del vector de parámetros w. En primer lugar, se crea un vector $y = [y_1, \ldots, y_L]$ con las salida esperada para cada vector de entradas, donde $y_j = -1$ para $x_j \in \omega_1$ y $y_j = 1$ para $x_j \in \omega_2$. Después, se calcula el valor óptimo de w como la combinación de parámetros que minimiza la función del error cuadrático medio $mse = \frac{1}{L} * \sum_{j=1}^{L} (y_j - w x_j)^2$. Partiendo de la derivada del error cuadrático medio con respecto a los parámetros $\partial mse / \partial w$, se obtiene la expresión del vector de parámetros óptimo $w = (X \cdot X^T)^{-1} \cdot X \cdot y^T$, donde X es la matriz formada por los L vectores de entrada.

Redes Neuronales Artificiales

Al intentar construir máquinas inteligentes surge de forma natural un modelo: la mente humana. Por lo tanto, una idea obvia en el campo de la inteligencia artificial es la de simular directamente el funcionamiento del cerebro en una computadora. Desde la perspectiva de las aplicaciones prácticas del reconocimiento de patrones, el "realismo biológico" impondría restricciones completamente innecesarias, por lo tanto, si bien las redes neuronales artificiales están inspiradas en el modelo biológico, se debe pensar en ellas como modelos que extraen combinaciones lineales de las entradas, convirtiéndolas en características (features) derivadas y después modelan la salida como una función no lineal de esas característica; no como una simulación del cerebro.

Perceptron

El perceptron fue ideado por Frank Rosenblatt en 1958 y es el primer modelo de neurona artificial. Es un método de aprendizaje supervisado que realiza una clasificación binaria en base a una transformación lineal, al igual que el clasificador lineal.

El perceptron representa una neurona biológica donde las dendritas son las entradas del perceptron, el axón es la salida, las sinapsis son los coeficientes de la función de decisión y el comportamiento (muy simplificado) se replica acumulando la intensidad de los impulsos recibidos que, al superar cierto umbral, "activan" la neurona emitiendo una señal por la salida.

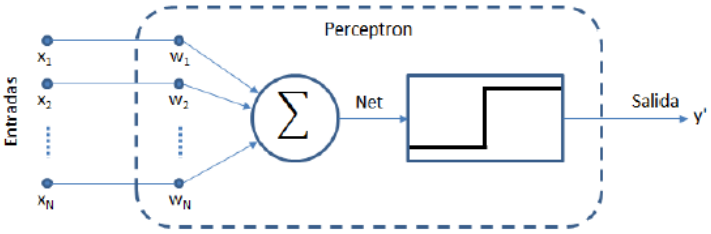


Figura 8.5: Esquema del perceptron.

Para el vector de entrada $x = [x_1, x_2, \ldots, x_N]$, un vector de pesos $w = [w_1, w_2, \ldots, w_N]$, el umbral de activación θ y la función umbral f, la salida y' del perceptron se calcula como:

$$y' = f\left(\left(\sum_{i=1}^{N} w_i x_i\right) - \theta\right)$$

donde

$$f(Net, \theta) = \begin{cases} 0 & \text{si} \quad Net - \theta < 0 \\ 1 & \text{si} \quad Net - \theta \geq 0 \end{cases}$$

Net: la suma ponderada de las entradas $\sum_{i=1}^{N} w_i x_i$ para simplificar las expresiones.

Para hacer más simple el cálculo, usualmente se incluye el umbral θ dentro de Net añadiendo un nuevo peso sináptico $w_0 = -\theta$. Para esto se redefine el vector de entrada como $x = [1, x_1, x_2, \ldots, x_N]$ con el mismo criterio utilizado al aumentar el vector de entrada del clasificador lineal. A partir de estos cambios, la salida se calcula como:

$$y' = f\left(\sum_{i=0}^{N} w_i x_i\right)$$

donde

$$f(Net) = \begin{cases} 0 & \text{si} \quad Net < 0 \\ 1 & \text{si} \quad Net \geq 0 \end{cases}$$

En la figura 8.6 se muestra el esquema más usual del perceptron, donde se reflejan los cambios realizados sobre el umbral. Notar la entrada fija en 1 del peso w_0.

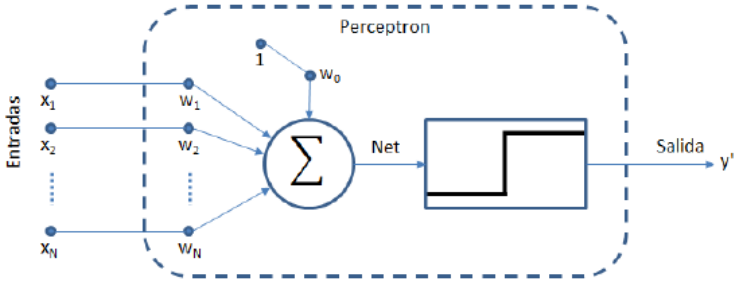


Figura 8.6: Esquema del perceptron para $w_0 = -\theta$ y vector de entrada aumentado.

El ajuste de los pesos del perceptron se realiza con la siguiente ley de aprendizaje: $\Delta w_i = \alpha(y - y')x_i$. Donde y es la salida esperada y α es la tasa de aprendizaje. El proceso es el siguiente:

- Inicializar los pesos con valores aleatorios.
- Mientras $error > 0$ para algún vector de entradas, hacer:
  - Ejecutar ciclo completo de entrenamiento ($epoch$). Mientras existan vectores de entrenamiento hacer:
    * Tomar un vector entrada/salida del conjunto de datos de entrenamiento.
    * Calcular la salida $y' = f\left(\sum_{i=0}^{N} w_i x_i\right)$
    * Calcular el $error = y - y'$
    * Calcular $\Delta_{w_i} = \alpha(y - y')x_i$
    * Modificar los pesos haciendo $w_{i_{t+1}} = w_{i_t} + \Delta_{w_i}$

Notar que tal como está declarado el método, y a menos que se defina una cantidad máxima de ciclos, el entrenamiento termina solo cuando todos los vectores están bien clasificados.

El perceptron tiene la misma desventaja que el clasificador lineal, solo es capaz de clasificar correctamente las entradas de problemas linealmente separables. Este problema NO se puede solucionar, el perceptron NO puede resolver el problema, sin embargo, hay una alternativa **teórica** que involucra otro modelo y abre un camino para tratar este tipo de problemas. Conectando perceptrones entre sí, en lo que se llama perceptron multicapa, se podría resolver el problema **si supiéramos cómo adaptar los pesos**.

¿Por qué no se puede aplicar este modelo? No sabemos cómo entrenarlo... No sabemos cómo modificar los pesos de los perceptrones de la primera capa porque no sabemos cuánto contribuyó cada uno al error de la última salida final.

Adaline

Adaline (ADAptative LINear Element) es un modelo de red neuronal artificial que tiene dos diferencias con respecto al perceptron.

La primera diferencia es la función de activación, en Adaline se utiliza una función lineal. La salida de Adaline es directamente $y' = Net$. Si este modelo se aplica a una predicción del tipo regresión (cosa imposible para el perceptron), la salida utilizada es Net. En el caso de la clasificación, será necesario utilizar una función escalón en algún momento. La función de activación de Adaline es lineal y, fuera del modelo, se aplica una función umbral para determinar la clase.

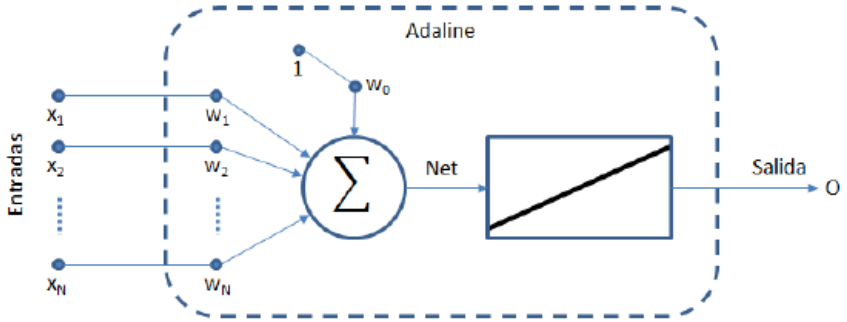


Figura 8.12: Esquema de Adaline.

La segunda diferencia es la ley de aprendizaje. En Adaline se utiliza la regla delta, basada en el mínima error cuadrático medio (Least Mean Squared o LMS error). Los pesos de la red se ajustan mediante una búsqueda guiada por el gradiente descendiente de la función del error.

- O: salida de la red (por *output*). En Adaline $O = \sum_{i=0}^{N} w_i x_i = Net$.
- k: identificador de un vector de entrenamiento.
- E: error cuadrático medio. $E = \frac{1}{2L} \sum_{k=1}^{L} (y_k - O_k)^2$, donde L es la cantidad de vectores de entrenamiento.

La adaptación de los pesos se lleva a cabo a través de una búsqueda sobre la superficie del error. Para encontrar el mínimo de la función del error los pesos se modifican en cantidades proporcionales al gradiente decreciente de la función E. La expresión de la ley de aprendizaje de Adaline es igual a la del perceptron, pero en el caso de Adaline el error puede tomar cualquier valor en R.

$$\begin{aligned} \Delta_{w_{ik}} &= -\alpha\delta\ x_i \\ &= -\alpha\left(-(y_k - O_k)\right) x_i \\ &= \alpha(y_k - O_k) x_i \end{aligned}$$

El método de entrenamiento también es similar al del perceptron, con la diferencia de que el error nunca es cero, así que se necesita otra condición de corte. Típicamente se corta el entrenamiento cuando se llega a un umbral de error previamente definido. En cuanto a las limitaciones, Adaline solo puede clasificar problemas que sean linealmente separables.

Metaheurísticas

Algoritmos Heurísticos

Los algoritmos heurísticos son los más fáciles de utilizar, ya que se basan en el conocimiento de una heurística que guía el proceso de búsqueda. El conocimiento del problema usualmente ayuda a encontrar una heurística razonable que encontrará rápidamente una solución aceptable. Un algoritmo de este tipo sólo buscará dentro de un subespacio del área total a una solución buena (que no necesariamente es la mejor) que satisfaga las restricciones impuestas. La principal limitación es su incapacidad para escapar de óptimos locales encontrar soluciones parcialmente óptimas).

Algoritmos Metaheurísticos

Una metaheurística es un proceso iterativo maestro que guía y modifica las operaciones de una heurística subordinada para producir eficientemente soluciones de alta calidad. Las metaheurísticas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. La heurística subordinada puede ser un procedimiento de alto o bajo nivel, una búsqueda local, o un método constructivo. Entre los algoritmos metaheurísticos más conocidos están, el recocido simulado y los algoritmos genéticos.

Cuatro tipos:

1. Las metaheurísticas de relajación se refieren a procedimientos de resolución de problemas que utilizan relajaciones del modelo original (es decir, modificaciones del modelo que hacen al problema más fácil de resolver), cuya solución facilita la solución del problema original.

2. Las metaheurísticas constructivas se orientan a los procedimientos que tratan de obtener una solución a partir del análisis y selección paulatina de las componentes que la forman.

3. Las metaheurísticas de búsqueda guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explorar las estructuras de entornos asociadas.

4. Las metaheurísticas evolutivas están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.

   Las metaheurísticas evolutivas establecen estrategias para conducir la evolución en el espacio de búsqueda de conjuntos de soluciones (usualmente llamados poblaciones) con la intención de acercarse a la solución óptima con sus elementos. El aspecto fundamental de las heurísticas evolutivas consiste en la interacción entre los miembros de la población frente a las búsquedas que se guían por la información de soluciones individuales.

   Las diferentes metaheurísticas evolutivas se distinguen por la forma en que combinan la información proporcionada por los elementos de la población para hacerla evolucionar mediante la obtención de nuevas soluciones.

   Los algoritmos genéticos y meméticos y los de estimación de distribuciones emplean fundamentalmente procedimientos aleatorios, mientras que las metaheurísticas de búsqueda dispersa o de reencadenamiento de caminos (Path-Relinking) emplean procedimientos sistemáticos.

Metaheurísticas – Algoritmos Genéticos

En ocasiones la computación se basa en procesos observados de la naturaleza para resolver ciertos problemas: por ejemplo, las redes neuronales que replican los procesos de sinapsis entre las neuronas. En este caso, los algoritmos genéticos replican el modelo de selección natural propuesto por Darwin, y que resume la famosa frase 'la supervivencia del más fuerte o adaptado'.

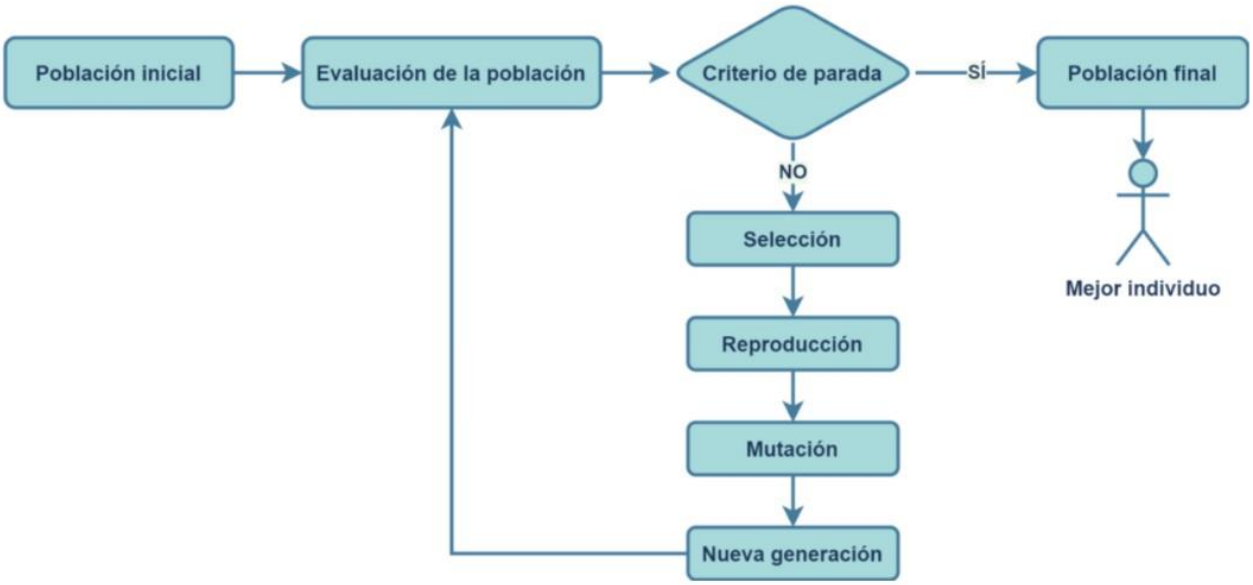
Este modelo básicamente dice que, dentro de una población, los individuos que sobreviven son aquellos que están más adaptados al medio, por lo tanto, las generaciones futuras de estos estarán mejor adaptadas ya que serán combinaciones de los mejores genes de sus antepasados. Además, esta teoría de la evolución introduce un concepto muy interesante que son las mutaciones. Una mutación es un pequeño cambio que se produce de manera aleatoria en ciertos individuos e introduce de esta manera versatilidad en las poblaciones. Habrá mutaciones que den lugar a cambios favorables y otros desfavorables.

Se utilizan para resolver problemas de Búsqueda y Optimización, ya que se basan en evolucionar poblaciones de soluciones hacia valores óptimos del problema.

Estructura de un algoritmo genético

Tener en cuenta:

- Individuo: los individuos de la población son las posibles soluciones al problema que se intenta resolver.
- Población: conjunto de individuos.
- Función fitness o de adaptación: función que evalúa a los individuos y les asigna una puntuación en función de que tan buenas sean las soluciones para el problema.
- Función de cruce: función que dados dos individuos genera dos descendientes a partir de la combinación de genes de sus padres, esta función depende del problema en cuestión.



- Fase inicial: se genera una población inicial de individuos (soluciones)
- Fase de evaluación: se evalúan los individuos de la población con la función fitness
- Fase de selección: se seleccionan los mejores individuos
- Fase de reproducción: se cruzan los individuos seleccionados mediante la función de cruce, dando lugar a una nueva generación que va a sustituir a la anterior
- Fase de mutación: se introducen mutaciones (pequeños cambios) en ciertos individuos de la nueva población de manera aleatoria o un entrecruzamiento cromosómico (también llamado crossover o recombinación)
- Se obtuvo una nueva generación, en general, con soluciones mejores que la anterior. Se vuelve al punto 2

Los algoritmos genéticos pueden finalizar cuando se alcanza un número de generaciones concreto o cuando cumplen una condición de parada.

Ventajas

Se desenvuelven bien en problemas con un paisaje adaptativo complejo: aquéllos en los que la función de aptitud es ruidosa, cambia con el tiempo, o tiene muchos óptimos locales, gracias a los cuatro componentes principales de los algoritmos genéticos, paralelismo, selección, mutación y cruzamiento, los que trabajan juntos para conseguir su buen desempeño.

Pueden explorar el espacio de soluciones en múltiples direcciones a la vez, por lo que, los algoritmos genéticos funcionan particularmente bien resolviendo problemas cuyo espacio de soluciones potenciales es realmente grande, demasiado vasto para hacer una búsqueda exhaustiva en un tiempo razonable

Los algoritmos genéticos realizan cambios aleatorios en sus soluciones candidatas y luego utilizan la función de aptitud para determinar si esos cambios producen una mejora. Como sus decisiones están basadas en la aleatoriedad, todos los caminos de búsqueda posibles están abiertos; en contraste a cualquier otra estrategia de resolución de problemas que dependa de un conocimiento previo

Desventajas

Si se elige mal una función de aptitud o se define de manera inexacta, puede que el algoritmo genético sea incapaz de encontrar una solución al problema, o puede acabar resolviendo el problema equivocado.

Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen tamaño de la población, el ritmo de mutación y cruzamiento, el tipo y fuerza de la selección.

El lenguaje utilizado para especificar soluciones candidatas debe ser robusto; es decir, debe ser capaz de tolerar cambios aleatorios que no produzcan constantemente errores fatales o resultados sin sentido. Una de las formas mas usadas es definir a los individuos como listas de números -binarios, enteros o reales- donde cada número representa algún aspecto de la solución candidata.

Modelos Bayesianos

Probabilidad a priori o incondicional

La probabilidad a priori o incondicional asociada a una proposición A es el grado de creencia que se le otorga en ausencia de cualquier otra información y se escribe como $P(A)$. Por ejemplo, dada la variable X cuyo dominio (posibles valores que puede tomar) es $\langle x1, x2\rangle$, si la probabilidad a priori de que $X = x1$ es $0{,}3$, deberíamos escribir: $P(X = x_1) = 0{,}3$ *o* $P(Test = positivo) = 0{,}3$.

Probabilidad condicional o a posteriori

Una vez que el agente obtiene alguna evidencia que afecta a la variable X, las probabilidades a priori ya no son aplicables a X. En su lugar usamos probabilidades a *posteriori o condicionales*. La probabilidad condicional del evento A dada la ocurrencia del evento B se escribe como $P(a|b)$. Por ejemplo, si la probabilidad de que $X = x_1$ dada la evidencia de que $Y = y_1$ *es* $0{,}7$, deberíamos escribir $P(X = x_1|Y = y_1) = 0{,}7$ o $P(x_1|y_1) = 0{,}7$

Es la probabilidad de que ocurra un evento ***A*** dado que ocurre otro evento ***B*** (evidencia). No implica causa-efecto. Ejemplo: $P(Test = Positivo|Enfermedad = Presente) = 0{,}7$

Las probabilidades condicionales pueden definirse en t´erminos de probabilidades no condicionales. La ecuaci´on que la define es:

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

Donde $P(a \wedge b)$ es la probabilidad conjunta de a y b, es decir, la probabilidad de que a y b ocurran al mismo tiempo. De la ecuación anterior se obtiene:

$$P(a \wedge b) = P(a|b)P(b)$$

Conocida como la *regla del producto*. Esta última ecuación es más intuitiva, ya que expresa que la probabilidad de que ocurran a y b es igual a la probabilidad de que ocurra a dado b, siempre y cuando ocurra b, o sea, por la probabilidad de b. La probabilidad condicional es una herramienta muy útil para representar información causal de la forma $P(efecto|causa)$.

Regla o teorema de Bayes

Me permite calcular una probabilidad condicional cuando tenemos la probabilidad condicional en sentido contrario.

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

Por ejemplo, si una alarma se dispara (por la razón que sea) con probabilidad $P(alarma)$, los robos ocurren con probabilidad $P(robo)$ y sabemos que la probabilidad de que nuestra alarma se dispare cuando hay un robo es $P(alarma|robo)$, con la regla de Bayes podemos calcular la probabilidad de que esté ocurriendo un robo cuando suena la alarma, $P(robo|alarma)$.

Un ejemplo para comprender la importancia del teorema de Bayes. Supongamos que Juan tiene un examen médico, que incluye una radiografía de tórax, como rutina de ingreso para su nuevo trabajo en un banco y que en la radiografía hay un hallazgo compatible con el cáncer de pulmón. A esto último lo representamos como $Rad = positivo$. Al recibir el resultado, Juan piensa que tiene la enfermedad ($Enf = verdadero$) y se preocupa mucho,

pero... ¿debería hacerlo? Sin conocer la exactitud del test, Juan realmente no puede saber qué tan probable es que tenga cáncer de pulmón. Cuando se entera que el test no es absolutamente concluyente, decide investigar y descubre que este tiene una tasa de falsos negativos de 0,4 y de falsos positivos 0,02. De los datos podemos deducir que:

$$P(Rad = positivo|Enf = verdadero) = 0,6$$

$$P(Rad = positivo|Enf = falso) = 0,02$$

Dadas estas probabilidades, Juan se siente un poco mejor. Sin embargo, nota que todavia no sabe cuál es la probabilidad de qué él tenga cáncer de pulmón. La probabilidad de que Juan tenga cáncer de pulmón es $P(Enf = verdadero|Rad = positivo)$, y esta no es una de las probabilidades listadas recién.

Juan finalmente recuerda el teorema de Bayes y se da cuenta de que todavía necesita otra probabilidad para determinar la que le interesa. La probabilidad faltante es $P(Enf = verdadero)$, que representa la probabilidad de que él tenga cáncer de pulmón antes de conocer el resultado de la radiografía. El otro dato útil que tiene Juan es que pertenece al grupo de personas que se realizó un examen preocupacional, no un examen motivado por síntomas. Entonces, cuando se entera de que solo 1 de cada 1000 nuevos empleados tiene cáncer de pulmón, asigna 0.001 a $P(Enf = verdadero)$. Ahora sí Juan aplica la regla de Bayes:

$$P(Enf = verdadero|Rad = positivo)$$

$$= \frac{P(Rad = positivo|Enf = verdadero)P(Enf = verdadero)}{P(Rad = positivo)}$$

$$= \frac{P(Rad = positivo|Enf = verdadero)P(Enf = verdadero)}{P(Rad = pos|Enf = ver)P(Enf = ver) + P(Rad = pos|Enf = fal)P(Enf = fal)}$$

$$= \frac{0,6 \times 0,001}{0,6 \times 0,01 + 0,02 \times 0,999}$$

$$= 0.029$$

Entonces, ahora Juan sabe que su probabilidad de tener la enfermedad es cercana a 0.03 y se relaja un poco mientras espera el resultado de otros estudios.

Supongamos que otra persona, Pedro, tiene el mismo diagnóstico de radiografía de tórax que tuvo Juan ($Rad = positivo$). Sin embargo, Pedro se hizo el estudio porque ha trabajado en minas durante 20 años, y sus empleadores se han preocupado porque notaron que cerca del 10% de sus trabajadores desarrollaron cáncer de pulmón después de trabajar varios años en las minas. ¿Cuál es la probabilidad de que Pedro tenga cáncer de pulmón? Basado en la información que tenemos sobre Pedro antes de que se realizara el test, le asignamos una probabilidad a priori $P(Enf = verdadero) = 0,1$. Repitiendo los cálculos de la regla de Bayes para este valor, podemos concluir que $P(Enf = verdadero|Rad = positivo) = 0,769$ para Pedro, no muy alentador.

Distribución Conjunta Completa

La distribución conjunta completa es la distribución de probabilidad conjunta que considera el conjunto completo de variables. Es decir, que contiene la probabilidad de ocurrencia de cada una de las combinaciones posibles entre los valores que puede tomar cada variable. La tabla siguiente muestra la distribución conjunta completa para el caso sencillo del ejemplo de la radiografía de tórax de Juan.

Table 10.1: Distribución conjunta completa para el ejemplo de Juan.

		Enf		
		verdadero	*falso*	$P(Rad)$
Rad	*positivo*	0,0006	0,01988	0,02058
	negativo	0,0004	0,97902	0,97942
	$P(Enf)$	0,001	0,999	1

Notar que la última fila y columna contienen la probabilidad de ocurrencia de cada valor de Enf y Rad respectivamente. Estas probabilidades ($P(x_i)$) se llaman marginales y se calculan sumando los valores de la fila o columna donde se encuentran, es decir sumando todas las probabilidades conjuntas donde la variable toma el valor x_i.

La siguiente tabla muestra la distribución conjunta completa para el ejemplo de Pedro.

Table 10.2: Distribución conjunta completa para el ejemplo de Pedro.

		Enf		
		verdadero	*falso*	$P(Rad)$
Rad	*positivo*	0,06	0,018	0,078
	negativo	0,04	0,882	0,922
	$P(Enf)$	0,1	0,9	1

Podríamos unir los ejemplos, pero para eso deberíamos definir una nueva variable $Trabajo$ con dominio $\langle en\_las\_minas, en\_otro\_lado \rangle$ y conocer la probabilidad de que una persona trabaje en las minas. Si esto fuera asi, la tabla tendría todas las combinaciones de las tres variables.

La distribución conjunta completa es una tabla que contiene toda la información necesaria para responder cualquier pregunta sobre las variables o la combinación de ellas.

Independencia

Dos eventos a y b son independientes si se cumple alguna de las siguientes condiciones:

1. $P(a|b) = P(a)\ y\ P(a) \neq 0, P(b) \neq 0.$
2. $P(a) = 0\ o\ P(b) = 0.$

Es decir, si $P(a)$ y $P(b)$ no son nulas, a y b son independientes cuando $P(a|b) = P(a)$. En ese caso, claramente la probabilidad de ocurrencia del evento a no cambia si ocurre o no ocurre b.

Solo si a y b son independientes, partiendo de $P(a|b) = P(a)$ y usando la regla del producto se puede ver que $P(a \wedge b) = P(a)P(b)$. Es importante tener en mente que la dependencia entre dos variables o eventos no implica que uno sea la causa del otro.

Independencia Condicional

Dos eventos a y b son condicionalmente independientes dado c, si $P(c) \neq 0$ y se cumple alguna de las siguientes afirmaciones:

1. $P(a|b \wedge c) = P(a|c)$ y $P(a|c) \neq 0, P(b|c) \neq 0.$

2. $P(a|c) = 0$ o $P(b|c) = 0.$

Un ejemplo para comprender esta propiedad. Supongamos que en un pueblo existen dos vecinos que no tienen ningún tipo de interacción entre ellos. Cada vecino tiene cierta probabilidad de salir de su casa con paraguas cuando hay pronóstico de lluvias. Llamemos p_1 y p_2 a los eventos “salir con paraguas” para el vecino 1 y para el vecino 2 respectivamente. Naturalmente, $P(p_1)$ y $P(p_2)$ son altas en caso de pronóstico positivo. Cuando uno de los vecinos sale con paraguas es más probable que el otro lo haga también, es decir, p_1 y p_2 no son independientes. Esto no

significa que se influyan mutuamente en el mundo real. El comportamiento se debe a que ambos eventos tienen la misma causa. Entonces, para un observador que no conoce el pronóstico, los eventos son dependientes. Si el observador conoce el pronóstico, los eventos se vuelven independientes dado el pronóstico. Es decir, si sabemos que va a llover, la probabilidad de ocurrencia del evento p_1 es condicionalemente independiente de la ocurrencia del evento p_2 y se escribe asi: $P(p_1|lluvia \wedge p_2) = P(p_1|lluvia)$, donde lluvia es el evento que representa al pronóstico positivo de lluvia.

La independencia condicional tiene particular importancia en la utilización de las redes bayesianas porque, como veremos más adelante, permiten representar toda la información necesaria mediante un conjunto reducido de probabilidades condicionales.

Redes Bayesianas

Una red bayesiana es una estructura de datos que representa las dependencias entre variables. Muestra una descripción compacta de cualquier distribución de probabilidad conjunta completa. Es un grafo dirigido en el que cada nodo contiene información probabilística cuantitativa. Especificación completa:

1. Un conjunto de variables aleatorias forma los nodos de la red. Las variables pueden ser discretas o continuas.

2. Un conjunto de arcos dirigidos conecta pares de nodos. Si hay un arco de un nodo $\boldsymbol{X}$ a un nodo $\boldsymbol{Y}$, se dice que **X** es un padre de **Y**.

3. Cada nodo $\boldsymbol{X_i}$ tiene una distribución de probabilidad condicionada $\boldsymbol{P(X_i | Padres(X_i))}$ que cuantifica el efecto de los padres del nodo.

4. El grafo no tiene ciclos dirigidos, entonces es un grafo acíclico dirigido, o GAD.

La topología de la red especifica las relaciones de independencia condicional que existen en el dominio. El significado intuitivo de un arco que sale de X y apunta a Y es, habitualmente, que X tiene una influencia directa sobre Y. Es generalmente sencillo para un experto del dominio decidir qué influencias directas existen en el área. Una vez que la topología de la red bayesiana está diseñada, necesitamos sólo especificar una distribución de probabilidad condicional para cada variable dados sus padres. La combinación de la topología y las distribuciones condicionales son suficientes para definir la distribución conjunta completa para todas las variables.

Inferencia en Redes Bayesianas

La tarea básica de cualquier sistema de inferencia probabilista es calcular la distribución de probabilidad a posteriori para un conjunto de variables pregunta, dado algún evento observado (esto es, alguna asignación de valores para un conjunto de variables evidencia). Notación en este contexto:

$\boldsymbol{X}$ es la variable pregunta

$\boldsymbol{E}$ es el conjunto de variables evidencias $\boldsymbol{E_1, \ldots, E_m}$. Y $\boldsymbol{e}$ es un evento observado particular

$\boldsymbol{Y}$ denota las variables no evidencia $Y_1, \ldots, Y_l$ (a veces llamadas variables ocultas).

Conjunto completo de variables: $X = \boldsymbol{X} \cup \boldsymbol{E} \cup \boldsymbol{Y}$. Una pregunta típica pide la distribución de probabilidad a posteriori $P(X|e)$.

Ejemplo

Planteamos una pregunta sobre el ejemplo del robo del libro.

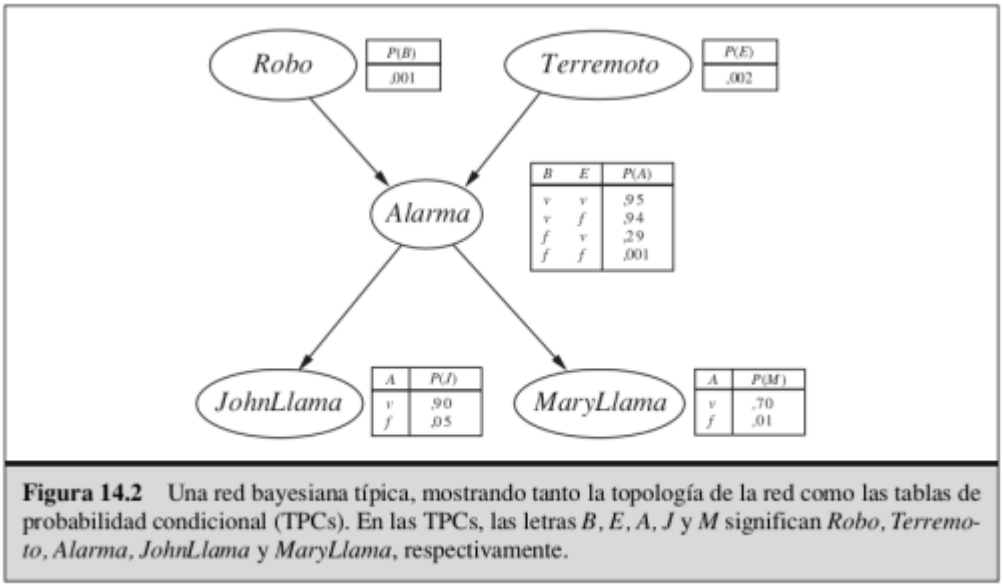


Figura 14.2 Una red bayesiana típica, mostrando tanto la topología de la red como las tablas de probabilidad condicional (TPCs). En las TPCs, las letras B, E, A, J y M significan *Robo*, *Terremoto*, *Alarma*, *JohnLlama* y *MaryLlama*, respectivamente.

Figura 10.1: Red bayesiana del ejemplo del robo en [RNR04].

En la red del robo, podemos observar el evento en el que $JohnLlama = cierto$ y $MaryLlama = cierto$. Podríamos entonces preguntarnos por la probabilidad de que haya ocurrido un robo: $P(Robo|JohnLlama = cierto \wedge MaryLlama = cierto)$. Para hacer más cortas las expresiones, reemplacemos los nombres de las variables por:

La primera letra, por ejemplo R para Robo. Como las variables de este ejemplo son binarias, vamos a usar la misma letra (en minúscula) para indicar el caso verdadero y la letra negada para el caso falso, por ejemplo $\langle r, \neg r \rangle$ para R. Entonces, la pregunta se puede escribir como $P\ (R|j\ \wedge\ m)$ si queremos conocer tanto la probabilidad de que el robo haya ocurrido como la de que el robo no haya ocurrido (devuelve dos valores) y $P(r|j\ \wedge\ m)$ si sólo nos interesa la probabilidad de que el robo sea cierto.

Usando las distribuciones conjuntas totales

Si bien es el método más simple y directo para realizar inferencias sobre el dominio, técnicamente no es un método de inferencia sobre redes bayesianas porque requiere que las probabilidades estén expresadas como probabilidades conjuntas.

Almacenar probabilidades conjuntas no es eficiente. Se podrían calcular a partir de las probabilidades condicionales, pero esto no mejoraría la eficiencia. Además, aunque se partiera de las probabilidades conjuntas el método tiene complejidad algorítmica muy alta.

A pesar de estas desventajas, es importante comprender cómo y por qué funciona, ya que es la base para el resto de los métodos. La explicación está en la sección 13.4 del libro.

Con la regla de Bayes

Respondamos la pregunta usando la regla de Bayes:

$$P(r|j \wedge m) = \frac{P(j \wedge m|r)P(r)}{P(j \wedge m)} \tag{10.3}$$

Sabemos que $P(r) = 0,001$, pero no conocemos $P(j \wedge m|r)$ ni $P(j \wedge m)$. Tenemos que calcularlas. Empecemos por $P(j \wedge m)$, para lo cual es necesario conocer $P(a)$ porque

$$P(j \wedge m) = 0,9 \times 0,7 \times P(a) + 0,05 \times 0,01 \times P(\neg a). \tag{10.4}$$

Entonces, calculamos $P(a)$ como

$$\begin{aligned} P(a) &= 0,95P(r)P(t) + 0,94P(r)P(\neg t) + 0,29P(\neg r)P(t) + 0,001P(\neg r)P(\neg t) \\ &= 0,95 \times 0,001 \times 0,002 + 0,94 \times 0,001 \times 0,998 + 0,29 \times 0,9990,002 + \\ &\quad 0,001 \times 0,999 \times 0,998 \\ &\approx 0,002516442. \end{aligned}$$

Ahora sí (recordando que $P(\neg a) = 1 - P(a)$),

$$P(j \wedge m) \approx 0,002084100239. \tag{10.5}$$

Ahora sí (recordando que $P(\neg a) = 1 - P(a)$),

$$P(j \wedge m) \approx 0,002084100239. \tag{10.5}$$

El siguiente paso es calcular $P(j \wedge m|r)$ como

$$P(j \wedge m|r) = 0,9 \times 0,7 \times P(a|r) + 0,05 \times 0,01 \times P(\neg a|r). \tag{10.6}$$

Notar que el cálculo es parecido al de 10.4, pero en este caso usamos $P(a|r)$ porque suponemos que es cierto que hubo un robo, entonces no debemos usar la probabilidad *a priori* $P(a)$.

No conocemos todavia $P(a|r)$. Para calcularla suponemos que el robo existió, pero no sabemos nada sobre el terremoto, asi que debemos tener en cuenta las dos posibilidades (con y sin terremoto).

$$P(a|r) = P(a|r \wedge t)P(t) + P(a|r \wedge \neg t)P(\neg t)$$

$$= 0,95 \times 0,002 + 0,94 \times 0,998$$

$$\approx 0,94002.$$

Con lo anterior calculado, volvemos a la ecuación 10.6:

$$P(j \wedge m|r) = 0,9 \times 0,7 \times 0,94002 + 0,05 \times 0,01 \times (1 - 0,94002)$$

$$\approx 0,59224259$$

Finalmente, podemos responder la pregunta inicial completando el proceso iniciado en la ecuación 10.3:

$$P(r|j \wedge m) = \frac{P(j \wedge m|r)P(r)}{P(j \wedge m)}$$

$$\approx \frac{0,59224259 \times 0,001}{0,002084100239}$$

$$\approx 0.284171835364393$$

La probabilidad de que haya ocurrido un robo, dado que Mary y John llamaron para avisar, es cercana a 0.28.

Algunas consideraciones

En las redes grandes que pueden aparecer en casos reales hay aspectos muy importantes a tener en cuenta:

- Llevar a cabo un proceso de deducción y análisis similar al del ejemplo anterior es prácticamente imposible.
- La representación de las probabilidades conjuntas totales tendría un tamaño inmanejable, por eso usamos las probabilidades condicionales.
- Los mecanismos de inferencia automático se deben enfocar en la eficiencia, hasta el punto donde, para casos de gran tamaño, el cálculo exacto es demasiado ambicioso.
- No es posible crear estas redes de forma manual. Tanto el conocimiento del dominio para definir la topología, como las probabilidades conocidas, no son suficientes. En estos casos, se deben utilizar métodos automáticos para aprender las redes.

Inferencia Automática: ver libro