AC.pdf

user_3428533

Aprendizaje Computacional

4º Grado en Ingeniería Informática

Facultad de Informática
Universidad de Murcia

Accede al documento original

Resumen Aprendizaje Computacional

Tema 1: Resumen del aprendizaje computacional (1º)

Aprendizaje en un sistema IA: una maquina aprende siempre que cambia su estructura, programa o datos de tal forma que su rendimiento mejora.

Aprendizaje computacional: trata de mejorar los rendimientos de los sistemas a base de modificar su estructura de forma automática o autónoma.

Tipos de aprendizaje:

1. **Supervisado** (semisupervisado)
2. No supervisado
3. Por refuerzo

Un modelo de agente es un modelo abstracto generalizado de entidad que actúa. Se obtienen "**percepciones**" del entorno y se determinan que acciones hacer en base a esa información (y al pasado, el conocimiento y capacidades que ya tenga). Las acciones son "**racionales**" (en el sentido de racionalidad IA)
Tipos de agentes en IA

1. AGENTE REACTIVO SIMPLE: Selecciona la acción dependiendo de la percepción actual sin tener en cuenta (ignorando) la "historia perceptual" (las cosas que percibió en el pasado)
2. AGENTE REACTIVO BASADO EN MODELOS: Mantiene un estado interno, tiene un modelo del mundo, tanto el estado, como el modelo del mundo es conocimiento.
3. AGENTE BASADO EN OBJETIVOS: Tienen objetivos a cumplir. Necesita hacer búsquedas y planificar en base a esos objetivos
4. AGENTE BASADO EN UTILIDAD: No solo tienes objetivos sino una función de utilidad que valora entre distintos grados de "felicidad". Busca varias formas de llegar al objetivo, pero escoge la más satisfactoria

Los agentes que aprenden incluyen varios módulos extra que le permiten:

- "reprogramar" sus reglas de comportamiento
- "reprogramar" el modelo del mundo
- "reprogramar" las consecuencias de sus acciones

REPROGRAMAR = APRENDER. Si la reprogramación es automática-> APRENDIZAJE COMPUTACIONAL.

Aprendizaje supervisado (T2): dado un conjunto de N pares ejemplo de entrada-salida (generadas por una función desconocida y=f(x), descubrir una función h que

aproxime la verdadera función f.
Asumimos un modelo definido según h : y = g(x | θ) en donde g() es el modelo y θ los parámetros.
En el problema de **regresión** la variable y es un número y la función g() es la función de regresión. En el problema de **clasificación** la variable y es una etiqueta de clase y la función g() es el discriminante. Si la función f es **estocástica**, se tiene que aprender la distribución condicionada de probabilidades: P(Y|x)

Los datos (T6): Como es fácil de suponer si el aprendizaje se hace a base de ejemplos, los datos son un componente fundamental del proceso de aprendizaje. El **preprocesado** de datos es una fase importante del desarrollo de modelos que, además, consume gran parte de los recursos de desarrollo.

¿Como saber si un modelo es mejor que otro? (T3)

Tema 2: Aprendizaje supervisado (3º)

Recordemos que en el aprendizaje supervisado se quiere encontrar el mapeado entre unas entradas **X**, y unas salidas **Y**.

Existirán varios mapeados alternativos (distintas hipótesis **h**, una vez escogida una representación de dicho mapeado (cjto **H**)), de entre todos los posibles mapeados (distintas Hs posibles). El mapeado "perfecto" se denomina **C**.
Para encontrar esa h nos basamos en la **hipótesis del aprendizaje inductivo** cualquier hipótesis encontrada, que aproxime la función objetivo sobre un conjunto suficientemente grande de ejemplos de entrenamiento, aproximará también dicha función en otros ejemplos no vistos antes.
Llamaremos **tasa de error** de una hipótesis a la proporción de errores que tiene, es decir la proporción de veces que h(x) ≠ y.

Clasificación binaria:

En una clasificación binaria hay ejemplos positivos (pertenecen a la clase) y ejemplos negativos (no pertenecen). El aprendizaje de una clasificación binaria es encontrar una descripción que compartan todos los ejemplos positivos y ninguno de los negativos.

Clases múltiples:

Con dos clases se puede buscar una hipótesis para la clase positiva y asumir los no clasificados como pertenecientes a la clase negativa También es posible buscar una hipótesis para una clase y otra hipótesis distinta para la clase negativa.

Regresión:

Si dado un punto de entrada, en vez de obtener una clase a la que pertenece se busca una salida numérica tendremos un problema de regresión. Técnicamente regresión es

cuando asumimos que la salida tiene ruido (sin ruido es ajuste/aproximación determinista). Ese ruido se explica como la presencia de variables ocultas no observables. Trataremos de encontrar modelos g(x) que aproximen la función subyacente f(x).

Sesgos inductivos:

Es un problema habitual en aplicaciones de aprendizaje que para hacer una buena aproximación se necesitarían todos los puntos (es un ill-posed-problem). Al tener solo unos pocos ejemplos hay muchas soluciones aceptables, con esos ejemplos, pero que se alejan del problema real original. Esto obliga a asumir ciertos supuestos sobre los datos a la hora de hacer el aprendizaje. A estos supuestos se les llama "**sesgos inductivos**".
Un espacio de hipótesis **no sesgado** es aquel que contiene a todo concepto representable: aquel que representa cada subconjunto posible de instancias de X
Un algoritmo de aprendizaje que no realiza ninguna asunción/supuesto a priori, sobre la forma del concepto objetivo, no tiene ninguna base racional para clasificar correctamente instancias no vistas aún. Si conseguimos caracterizar el sesgo de un algoritmo de aprendizaje, tendremos una descripción de alto nivel de cómo ese algoritmo trabaja para producir inferencias inductivas.

Selección del modelo:

Dado que no se puede aprender sin sesgo inductivo, la cuestión es escoger el sesgo inductivo adecuado. A esto se le llama selección del modelo, entre posibles Hs (clases de hipótesis). Un buen modelo presentará una buena **generalización**. La generalización es como de bien predice salidas no vistas habiendo entrenado con cierto conjunto de entrenamiento.
Se consiguen buenas generalizaciones si hemos ajustado bien la complejidad del modelo a la complejidad subyacente de los datos.
• Si nos quedamos cortos hay underfitting (altos errores en Training y Testing).
• Si nos pasamos hay overfitting (bajos errores en Training y altos en Testing).

Hay que encontrar:

- Una buena estructura/representación que pueda contener ese modelo (La clase H)
- Muchas estructuras tienen (hiper-)parámetros que las definen y que también hay que encontrar
- Los valores específicos que determinan un modelo (parámetros del modelo), la h.

(**H** es el conjunto de posibles modelos o la clase de modelos (por ejemplo, "todas las redes neuronales con n capas").**h** es el modelo específico dentro de H, con los parámetros e hiper-parámetros ajustados tras el entrenamiento).

Tipos de parámetros:

- Parámetros de la estructura (**hiper-parámetros**)
- Parametros del algoritmo de búsqueda (**hiper-parámetros**)
- Parametros del modelo

Si tenemos varios posibles modelos (distintas Hs) y queremos ver cual se ajustaría mejor a nuestros datos (generaliza mejor) podríamos primero usar parte del conjunto de entrenamiento para estimar lo bien que generaliza cada H.
Para ello se divide el conjunto de datos de Training en un conjunto de Training y otro de Validación

Tema 3: Evaluación y comparación de clasificadores

Los tres tipos de errores básicos que manejamos en aprendizaje computacional dependen de los conjuntos de entrenamiento, validación y test.

- Error de **entrenamiento** (training error): es el error que genera el clasificador entrenado a partir de los datos que ha utilizado para el entrenamiento.
- Error de **validación** (validation error): es el error que genera el clasificador entrenado a partir de datos reservados de la muestra total, no usados para el entrenamiento.
- Error de **test** (test error): una vez se ha generado un clasificador final, hay que evaluar su capacidad. Para eso se utiliza este tercer conjunto. Fíjese que este error no es una distribución de errores, sino una estimación puntual.

Cualquier conclusión generada del análisis, **está sujeta siempre al conjunto de datos usado**. Cuando decimos lo bueno que es un algoritmo estamos diciendo cómo de bien se ajusta su sesgo inductivo a las propiedades de los datos sobre los que estamos entrenando. **No existe el concepto “mejor algoritmo de aprendizaje”**

Ramdomización: Para que los resultados sean independientes es necesario que el orden en el que se ejecutan los experimentos sea aleatorio.
Replicación: Dada una configuración fija de factores controlables, el experimento se ha de ejecutar varias veces para promediar el efecto de los factores incontrolables. (Crossvalidacion)
Bloqueo: Reducir o eliminar la variabilidad en factores “molestos” que esté en nuestra mano.

Experimentos en AC:

- Objetivo del estudio
- Escoger la variable de respuesta
- Escoger factores y sus niveles
- Escoger el diseño experimental
- Realizar el experimento
- Análisis estadístico de los datos
- Conclusión y recomendaciones

VALIDACIÓN CRUZADA

Una forma de utilizar más eficientemente los datos de entrenamiento es generar, a partir de ese conjunto, diferentes pares de cjtos. Entrenamiento/Validación.

- Validación de k plieges
    - dividir X en K partes aleatoriamente. Luego, dividir cada parte en 50% para entrenamiento y 50% para validación.
    - validación cruzada (si X pequeño). Utilizar repetidamente el mismo X para generar los subcjtos. de entrenamiento/validación.
        - Validación cruzada de K-Pliegues

$$\begin{aligned} \mathcal{V}_1 &= \mathcal{X}_1 \quad & \mathcal{T}_1 &= \mathcal{X}_2 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ \mathcal{V}_2 &= \mathcal{X}_2 \quad & \mathcal{T}_2 &= \mathcal{X}_1 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ &\vdots \\ \mathcal{V}_K &= \mathcal{X}_K \quad & \mathcal{T}_K &= \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_{K-1} \end{aligned}$$

        - Validación cruzada 5-2
          (el subíndice hace referencia a la división de X en partes. El superíndice a la parte dentro de la división)

$$\begin{aligned} \mathcal{T}_1 &= \mathcal{X}_1^{(1)} \quad & \mathcal{V}_1 &= \mathcal{X}_1^{(2)} \\ \mathcal{T}_2 &= \mathcal{X}_1^{(2)} \quad & \mathcal{V}_2 &= \mathcal{X}_1^{(1)} \\ \mathcal{T}_3 &= \mathcal{X}_2^{(1)} \quad & \mathcal{V}_3 &= \mathcal{X}_2^{(2)} \\ \mathcal{T}_4 &= \mathcal{X}_2^{(2)} \quad & \mathcal{V}_4 &= \mathcal{X}_2^{(1)} \\ &\vdots \\ \mathcal{T}_9 &= \mathcal{X}_5^{(1)} \quad & \mathcal{V}_9 &= \mathcal{X}_5^{(2)} \\ \mathcal{T}_{10} &= \mathcal{X}_5^{(2)} \quad & \mathcal{V}_{10} &= \mathcal{X}_5^{(1)} \end{aligned}$$

    - En dominios con pocos datos como en aplicaciones médicas usamos **leave-one-out** (En cada iteración, se usa un único dato como conjunto de validación, mientras el resto se utiliza para entrenar)
- Bootstrapping: mejor método para conjuntos muy pequeños. para generar varios cjtos. a partir de un mismo X, podemos utilizar un muestreo con remplazamiento. Para un número k de nuevos cjtos. realizamos, k veces, con el mismo X, un muestreo de tantos elementos como en X, con remplazamiento.

OFERTAS BLACK FRIDAY

msi

Esta oferta

es como una doble victoria tuya: disfrútala ahora porque no pasa dos veces.

Ver ofertas

HASTA -40%

Tu viejo portátil ya dio lo que tenía que dar. Pásate a MSI: rápido, potente y sin dramas. Lo enciendes y estás listo para todo. Aprovecha las ofertas y despídete del modo “se cuelga cada dos por tres”.

Midiendo el rendimiento de los clasificadores:
En clasificación binaria se suele usar la **matriz de confusión** distinguiendo cuatro casos

	Predicted class		
True Class	Positive	Negative	Total
Positive	tp : true positive	fn : false negative	p
Negative	fp : false positive	tn : true negative	n
Total	p'	n'	N

Por otro lado, siendo N=|TP|+|FP|+|TN|+|FN| en el cjto. de validación, el ratio de error se define como: ratio error = (| FN |+ | FP |) / N

Name	Formula
error	$(fp + fn)/N$
accuracy	$(tp + tn)/N = 1 - \text{error}$
tp-rate	tp/p
fp-rate	fp/n
precision	tp/p'
recall	tp/p = tp-rate
sensitivity	tp/p = tp-rate
specificity	$tn/n = 1 -$ fp-rate

El TP-RATE (ratio de aciertos) o número de clasificados como positivos sobre el total de positivos
El FP-RATE (falsas alarmas) o número de clasificados como positivos siendo negativos sobre el total de negativos.

Curvas ROC representa el ratio de aciertos frente al ratio de falsas alarmas.
Suele haber algún parámetro en el algoritmo que, a cambio de incrementar el ratio de FP, aumenta también el ratio de aciertos. Esta curva la usamos como referencia para tomar la decisión de cuántas falsas alarmas estamos dispuestos a tolerar a cambio de más aciertos.

Estimación de intervalos
En la estimación de intervalos se intenta encontrar el intervalo en el cual se encuentra un determinado parámetro con cierto grado de confianza.

Comprobación de hipótesis
En ciertas aplicaciones, en vez de estimar un parámetro lo que se hace es usar la muestra para comprobar alguna hipótesis respecto al parámetro.
En la comprobación de hipótesis se define un estadístico que sigue cierta distribución si la hipótesis es correcta.
• Si la probabilidad de obtener ese estadístico de la muestra en cuestión es muy baja se rechaza la hipótesis,
• y si no es baja se “falla el rechazo”.

Test estadísticos:

- **Binomial**

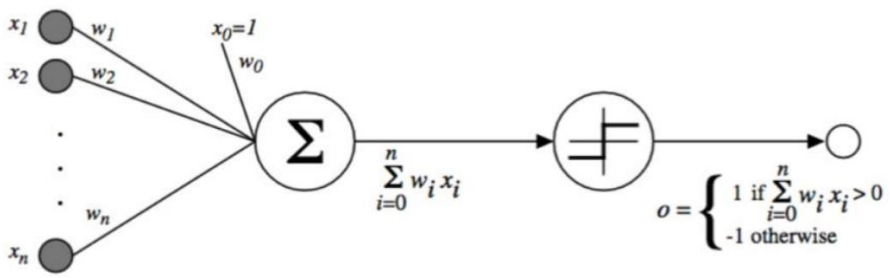
- **Normal aproximado**
- **1-Sample T Test**
- **Test t por pares en validación cruzada de k pliegues**
- **Test McNemar**
- **P-Value**

Tema 4: Redes neuronales artificiales

Paradigma neuronal: **NIMD** (Neural Instruction Multiple Data). Cada nodo tiene una memoria pequeña que almacena parámetros, cada uno implementa una función fija cuyo resultado se puede distribuir al resto.
Modelo aproximador de funciones reales, discretas y vectoriales a la salida.
Entrenamiento muy costoso, usar cuando el tiempo dedicado al aprendizaje no es relevante, cuando los datos contienen ruido, cuando no es necesario comprender la hipótesis generada a partir de D.

Usamos la noción de **clasificación basada en función discriminante.** Dicha función se va a especificar paramétricamente. El aprender un discriminante adecuado se reduce a buscar unos buenos parámetros. El vector ŵ es el vector de coeficientes, y w0 el sesgo (umbral) del discriminante. Ahora nuestro modelo básico de discriminante es **gi (x | θi)**. En donde θi es el conjunto de parámetros que definen nuestro discriminante. Cuando el problema no es linealmente separable, podemos usar un discriminante para cada par de clases i,j. Para un problema de K clases, usaremos K(K-1) funciones discriminantes.

EL PERCEPTRÓN: Unidad básica de una ANN típica.
Expresión matemática: **y =Σ wj x j + w0**



Representación de funciones booleanas:

La función AND puede representarse con un perceptrón, si asumimos +1 equiv. TRUE y 0 equiv. FALSE. El mismo perceptrón, puede usarse para representar OR. El resultado

general es que un perceptrón puede usarse para representar cualquier función m-de-n (al menos m de las n entradas a 1 para dar TRUE). Sin embargo, no podemos representar una función XOR, corresponde a un caso no separable linealmente.

Reglas del perceptrón:

1. Se inicializan pesos y el umbral de forma aleatoria.
2. Por cada ejemplo se calcula la salida del perceptrón

$$y = \sum_{j=1}^{d} w_j x_j + w_0$$

3. Se calcula el error entre la etiqueta verdadera y la salida del perceptrón (en caso de usarse umbral el error es 0 si se acierta y 1/-1 si se falla)
4. Se calculan los nuevos pesos con una fórmula que incluye el error y la tasa de aprendizaje η.

$$w_i = w_i + \eta * error * x_i$$

5. También se actualiza el umbral de forma similar

$$w_0 = w_0 + \eta * error$$

6. Se repite hasta que el error sea cero en todos los ejemplos.

K perceptrones en paralelo:
Con K > 2, tenemos K perceptrones, cada uno con su vector de pesos (i={1..K}). **Yi= Wx**
Ahora W es una matriz de K(d+1)

Entrenamiento del perceptrón: no separabilidad
Cuando el problema no es separable linealmente, la regla del perceptrón puede no converger. Como solución a ese problema tenemos la regla delta. Trataremos de minimizar

$$E(\vec{w}) = 1/2 \sum_{d \in D}^{d} (t_d - o_d)^2$$

Puede verse que el error depende de las salidas del perceptrón, en donde od es la salida de la máquina lineal, sin aplicar umbrales. Se basa en la aplicación de gradiente descendente.

Función de error
La representación depende del conjunto D particular que se use como muestra de la función a modelar. Cada punto de la función nos da una cantidad de error, dado el conjunto D y un perceptrón de dos variables de entrada. La optimización por gradiente, en función de los pesos tiene a moverse por la superficie de error, en la dirección de mayor variación del mismo. Esta operación de moverse paso a paso, se repite por cada d en D, presentado a la regla de entrenamiento que vamos a obtener ahora.

Regla del gradiente descendiente

Un gradiente descendente se puede aplicar cuando las hipótesis tienen parámetros continuos (los pesos) y cuando el error se puede diferenciar respecto a los parámetros de la hipótesis (la E(w) es diferenciable). La dirección de mayor variación del error nos la va a dar la expresión resultante de la derivada parcial del error con respecto a los pesos.

El algoritmo converge a la hipótesis de error mínimo, independientemente de si el problema es linealmente separable o no. Con un valor muy grande para el ratio de aprendizaje, corremos el riesgo de pasarlo de largo. Se suele disminuir gradualmente conforme avanza el aprendizaje. Se puede "atascar" en mínimos locales.

Principales desventaja: La convergencia a un mínimo local puede ser muy lenta. Si hay varios mínimos locales, no hay garantía de alcanzar el mínimo global. La Versión estocástica alivia esto mediante la modificación de los pesos para cada ejemplar de entrenamiento uno a uno, en lugar de hacerlo tras cada pasada del conjunto D completo. Lo que en el algoritmo aparecen como deltas, ahora son realmente modificaciones.

La **versión estocástica** del gradiente descendiente (SGD) actualiza los parámetros usando un solo dato (o minibatch pequeño) en cada iteración, en lugar de todo el conjunto de datos como en el **gradiente descendiente clásico**.

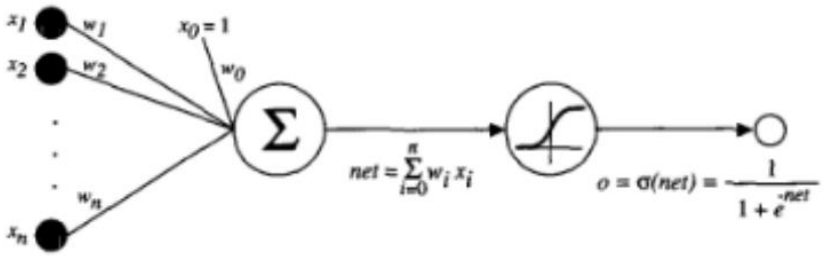
REDES NEURONALES MULTICAPA

Los perceptrones multicapa son aproximadores universales, pueden representar cualquier tipo de expresión lógica. Son aproximadores universales con una capa oculta y un número suficientemente grande de nodos ocultos. Con dos capas ocultas se ve fácilmente:

• Toda región del espacio puede delimitarse por hiperplanos mediante nodos ocultos en la primera capa.

• Un nodo oculto en la segunda capa los junta todos para hacer un AND y testar así si el punto cae en la región. El nodo de salida se conecta con éste último nodo mediante un peso cuyo valor es el valor de salida deseado.

Queremos representar regiones del espacio altamente no lineales y funciones derivables en el procesamiento de la salida. Para ello tenemos combinaciones lineales de variables, funciones umbral (salida en {-1, +1}). Necesitamos funciones no lineales, fácilmente derivables -> **sigmoide.**



ALGORITMO BACKPROPAGATION

- Minimiza el error cuadrático entre las salidas de la red y los valores objetivo
- Para ello aprende los pesos de los arcos de una red multicapa
- El proceso de minimización se basa en gradiente descendente

Al ser la superficie de error más compleja, pueden existir múltiples mínimos locales. Solo te garantiza la convergencia a un mínimo local. Posibles condiciones de terminación:
-Después de un número fijo de iteraciones
-Cuando el error conseguido cae bajo un determinado umbral
-Cuando la variación del error se estanca
-Cuando el error sobre otro cjto. es suficientemente bueno.
Demasiadas iteraciones -> overfitting
Modificación al algoritmo: **Momentum**
Se trata de aprovechar la inercia, causada por la actualización de los pesos según el último ejemplar presentado, en la actualización causada por el ejemplar actual. El efecto de esta modificación en el algoritmo es el de acelerar la convergencia en regiones de gradiente constante.

Resilient Propagation (Propagación elastica) **RPROP**
Actualiza los pesos de manera local. Nace como respuesta a los problemas en los que el momentum no acelera realmente la convergencia.

Convergencia: En problemas con superficies altamente no lineales, el número de mínimos locales es alto y Backpropagation tiene tendencia a caer en ellos. En el mundo real no es tan probelmatico.

- Redes con un número de arcos alto, corresponden a superficies de muchas dimensiones
- La red alcanza progresivamente y muy al final, las superficies no lineales

Recomendaciones para evitar los mínimos locales

- Usar el momentum
- Usar gradiente descendente estocástico

- Entrenar varias redes similares, con distintos pesos de inicio

Backpropagation estándar: Ajusta pesos usando el gradiente con una tasa de aprendizaje fija.
Backpropagation con momentum: Agrega una fracción del cambio anterior para acelerar la convergencia y evitar oscilaciones.
RPROP (Resilient Propagation): Usa solo el **signo** del gradiente, adaptando el tamaño del paso dinámicamente, ignorando la magnitud del gradiente.

Ninguno de los métodos garantiza alcanzar el **mínimo global** debido a la naturaleza no convexa de la superficie de error en redes neuronales.
Momentum y **RPROP** tienen **mejor capacidad** para evitar mínimos locales comparados con el backpropagation estándar.

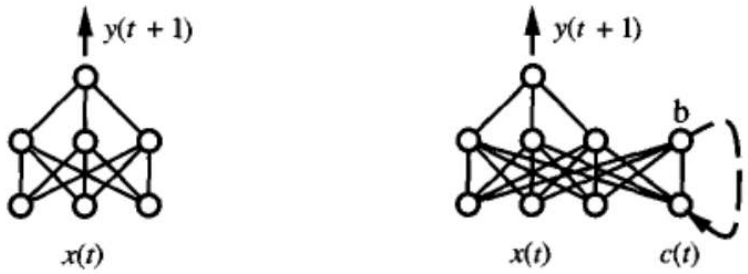
Generalización, sobreaprendizaje y parada

El **sobreaprendizaje** ocurre cuando el Backpropagation se dedica a ajustar la red a pequeñas particularidades locales en los datos de entrenamiento, irrelevantes para el problema general.
Estrategias para combatirlo:
• **Weight decay**: el tener pesos con valores muy altos posibilita regiones de decisión muy complejas
• El enfoque de las siguientes slides: usar un cjto. de datos de evaluación para controlar el error de generalización.
El criterio de **parada** es importante
•Parar demasiado pronto puede hacer inservible el modelo
•Parar demasiado tarde causa overfitting

REDES RECURRENTES:

Generamos un nuevo nodo oculto b y una nueva entrada c(t) cuyo valor es la salida de b en el instante t-1. Al depender b de x(t) y c(t) es capaz de representar dependencias

alejadas en el tiempo arbitrariamente.



(a) Feedforward network (b) Recurrent network

Podemos aplicar Backpropagation a este tipo de redes si desplegamos la recurrencia en el tiempo. Una vez finalizado el entrenamiento, nos quedamos con una sola red, sin el nodo b y la entrada c(t), y cada wij es la media de los wij de cada una de las subredes en la red desplegada.

Redes con topologías variables

El número ideal de nodos ocultos para un problema proporciona el mejor error de generalización.

Algoritmo **Cascada-correlación**, ideas básicas:

-Soluciona el problema del objetivo móvil en Backpropagation: cada nodo de la capa oculta busca su lugar en la red y no se comunican entre sí con lo que se produce el efecto manada:

-Sean las tareas A y B, los nodos oscilarán en resolver una de ellas y luego la otra de manera indefinida

-Se necesita mucho tiempo para resolver el problema

-Solución: permitir que únicamente un cjto. reducido de nodos cambie cada vez, manteniendo el resto cte.

-Ideas básicas: arquitectura en cascada y aprendizaje de pesos

Algoritmo cascada-correlación:

- Se añade un nodo que se conecta a la entrada y a los nodos ocultos anteriores.
- Ese nodo no se conecta (todavía) a la salida y se entrena con los residuos de la red (los errores que la red tiene en este momento).
- Cuando acaba este proceso se conecta ese nodo a las salidas, aunque los pesos de arcos que van desde la entrada a ese nodo oculto se congelan al añadirse (ya no se modificarán).

Se aplica de nuevo el entrenamiento con los pesos de los arcos que van desde los ocultos a la salida.

En el inicio (sin nodos ocultos) se entrenan los pesos de las entradas a las salidas mediante la regla del perceptrón, por ejemplo.

Cuando termina este entrenamiento, se evalúa la red sobre todo D y
- -Si el error es aceptable, paramos
- -Si no
  - -Añadimos un nuevo nodo mediante un algoritmo de creación de nodos
    - -Modifica los pesos de los arcos que llegan al nuevo nodo, maximizando la correlación entre la salida del nodo y el error residual.
  - -Entrenamos los pesos de salida nuevamente
- -Repetir

REDES DE CONVOLUCIÓN:

Una red de convolución es una red neuronal clásica en la que se aplican previamente, a los datos de entrada, una serie de transformaciones basadas en filtros de convolución.
Se aplica a imágenes, ya que los filtros de convolución (kernel) son matrices que combinan valores de píxeles cercanos y generan, a su vez, varias "imágenes"nuevas (una por filtro) donde, dependiendo del filtro, se resaltan (o atenúan) ciertas características de la imagen original. Cada filtro de convolución genera una imagen, a la que se le aplica un filtro de threshold, y luego se "compacta" mediante subsampleo, para reducir el aumento del tamaño de la entrada. Este proceso puede repetirse varias veces en serie para, finalmente, ser la entrada de una red neuronal clásica. La parte de red clásica se entrena con los algoritmos tradicionales y las **neuronas están fuertemente conectadas (**los valores de los pesos que conectan las neuronas son relativamente grandes, lo que indica que la información que se transmite a través de estas conexiones tiene un gran impacto en el comportamiento de la red**)**.
En la parte de los filtros se entrenan los pesos de los componentes del filtro. Los filtros básicamente actúan como detectores de características. Por ejemplo, la primera convolución suele ser capaz de detectar líneas y curvas. Las siguientes convoluciones combinaciones de estas (es decir, formas más complejas).

El número de parámetros adicionales a ajustar es reducido (las celdas de los kernels)
El sistema permite que, en el proceso de aprendizaje, no solo se "aprenda" a distinguir entre casos sino a realizar, a la vez, un aprendizaje de extracción de características.

Mapas autoorganizativos (SOM):

Un SOM está formado por un array de neuronas La entrada está totalmente conectada al array (arco de cada entrada a toda neurona)
Nodos cercanos se ajustan a determinadas señales de entrada (i.e. patrones) de manera ordenada **Aprendizaje competitivo**: para cada entrada, solo una neurona se parece más que el resto a esa entrada (i.e. best matching unit)
Aprendizaje básico no supervisado.
Aplicación por excelencia: organización de documentos por temática.

La versión estocástica realiza una iteración por ejemplar.
El ratio de vecindad decrece lentamente: una organización global de los nodos se consigue en fases tempranas del aprendizaje. Los ajustes locales se realizan después.

Los SOM consisten en una representación basada en m vectores n dimensionales en el espacio de los reales de unos datos mucho más grandes. Por lo tanto, es equivalente a una **compresión con pérdida**: clustering Partiendo de esa base SOM tiene aplicabilidad en

1. Preparación de datos para el aprendizaje
    a. Reducción de datos Podríamos transformar un cjto. de datos excesivamente grande en uno más pequeños usando los modelos almacenados en los nodos y algún nodo más (diagramas de Voronoi)
    b. Discretización Asociamos a cada modelo una etiqueta discreta y cada valor de entrada continuo lo sustituimos por la etiqueta de su BMU
    c. Valores nulos Al calcular el BMU de un ejemplar con valores nulos, solamente usamos valores conocidos y asumimos que los nulos son iguales a los que sí están presentes en los modelos.
    d. Visualización SOM es una representación 1D o 2D de los datos de entrenamiento. Si se ordena adecuadamente puede utilizarse para visualizar propiedades en los datos (mediante matrices de distancias o mapas coloreados)
2. Visualización de datos: Si segregamos la visualización de las diferentes variables de entrada podemos responder a las preguntas ¿Qué tipos de valores tienes las variables de entrada? ¿Qué tipos de valores o combinaciones de ellos son típicos en los clusters? ¿Hay dependencias significativas entre variables? Cada dimensión puede verse como una sección del mapa y cada componente muestra dispersión de valores en su dimensión (similares a histogramas) Si usamos varios mapas podemos detectar correlaciones en variables.

Tema 5: Reducción de la dimensionalidad

Reducir mediante algún método de selección o extracción de características el número de características de la base de datos antes de utilizar el algoritmo de aprendizaje.

1. Para cjtos. grandes de datos puede ser interesante reducir la complejidad espacial y temporal de trabajo del algoritmo.
2. Si ciertas entradas no son necesarias, podemos ahorrar el coste de extraerlas
3. Los modelos más simples son más robustos (menos varianza) en conjuntos de datos más pequeños
4. Con menos características los modelos son más sencillos de entender
5. Con pocas dimensiones podemos visualizar los datos para validar su estructura y detectar outliers

- **Selección de características**: Buscamos k de las d dimensiones de los X en D y descartamos las otras d-k dimensiones. Buscamos el subconjunto de dimensiones que mejor contribuye a la precisión del modelo.
    - **Selección hacia delante:** iterativamente añadimos la característica que más decrementa el error hasta que añadir cualquiera restante no lo decrementa significativamente.
    - **Selección hacia atrás:** comenzamos con las d características, eliminando aquellas que hacen que el resto decremente el error más hasta que cualquier eliminación incrementa el error significativamente.
- **Extracción de características:** Generamos un cjto. de k nuevas dimensiones y nuevos datos, k < d, combinando las d dimensiones originales.
    - **Análisis de componentes principales (PCA):** Es un método basado en la proyección: buscamos una selección desde el espacio d-dimensional original a un nuevo espacio k-dimensional, con una pérdida mínima de información.
      Dado un vector x, si queremos proyectarlo en la dirección de otro vector w, lo hacemos con su producto escalar z = w^T·x
      El método PCA, centra la muestra en todas sus dimensiones y luego transforma los datos al rotar los ejes para alinearlos con las direcciones de mayor varianza. No supervisado.
    - **Análisis de discriminante línea:** es un método de reducción dimensional supervisado. Consiste en encontrar la dirección de un vector en el cual al proyectar los datos a clasificar los objetos de una clase se encuentren lo más alejados posibles de objetos de otras clases.

Tema 6: Preprocesado de datos (2º)

Posibles daños en los datos:

- **Resultados espurios:** los conjuntos de datos contienen información producida de manera artificial, que no forma parte de los datos genuinos del problema. Y esta información llega a estar tan presente que puede convertirse en un patrón.
- **Visión de los algoritmos de análisis como cajas negras**: entender los entresijos de cada uno de los paradigmas y técnicas concretas es muy importante.
- **Limitaciones de las técnicas**: determinadas técnicas se usan simplemente porque son bien conocidas o disponemos de SW que las implementa, pero se deben considerar sus limitaciones.
- **Garantías en los resultados de los procesos de minería**

EDA: Análisis exploratorio de datos: Es el descubrimiento de estructura en los datos por medio de métodos simples como parámetros de estadística descriptiva o técnicas de visualización.

La **muestra** es el cjto. de datos a partir de la cual vamos a extraer características regulares de un fenómeno que estamos estudiando.

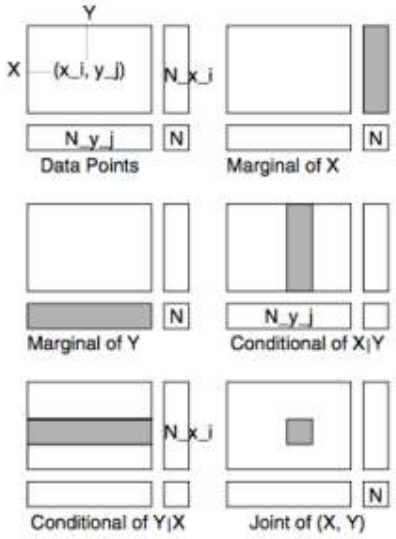
Modelamos cada atributo de la muestra como una **variable aleatoria**

- El conjunto de valores que puede tomar se denomina **dominio**
- Una variable aleatoria puede venir definida por su **función de distribución de probabilidad *f***
- Si consideramos varias variables aleatorias simultáneamente, estudiamos como se interrelacionan las ocurrencias de las variables consideradas entre sí
- Una di**stribución de probabilidad multivariable** representa la probabilidad, para un conjunto de atributos, de que tomen un cjto. dato de valores de entre los de sus dominios respectivos

Estimación de f: tarea compleja, realizaremos incrementalmente, análisis cada vez más complejos

- **Estimación paramétrica:** asumimos que la distribución subyacente a los datos tiene forma de función paramétrica que podemos representar.
  - Valores típicos
  - Mediana
  - Modo
  - Distribución de probabilidad
  - Varianza
  - Desviación absoluta
  - Rangos
  - Relación entre atributos
  - Curvas Q-Q
- **Estimación no paramétrica**

- Tablas de frecuencias



- Probabilidad acumulada
- Histogramas

PREPROCESADO DE DATOS:

Data Cleanning:

- **eliminando datos que faltan**
  - Ignorar la tupla (i.e. el dato que falta es el de la clase o existen nulos en varios de los atributos)
  - Rellenar los datos de manera manual (inabordable en el data mining)
  - Usar una constante global para rellenar los datos (i.e. cambiar cada ausente por unknown)
  - Usar la media del atributo para sustitución
  - Usar la media del atributo obtenida con todos los ejemplares que pertenecen a la misma clase que el ejemplar a modificar
  - Utilizar técnicas de minería de datos para predecir el valor más probable en cada caso (e.g. un árbol de decisión)
- **suavizando el efecto del ruido**: un error en las variables que se están midiendo.
  - **Binning**: Una serie de valores ordenados se agrupan en porciones y luego se suaviza cada porción. De esta forma, lo que se hace es un tratamiento local del ruido ya que se actúa de manera individual en cada porción.
  - **Clustering**: Aquellos valores que caen fuera de los clusters pueden considerarse como valores fuera de rango
  - **Regresión lineal**: Dado un conjunto de tuplas representado por dos variables, hallamos la línea recta que mejor se ajusta a esos datos
- **eliminando datos fuera de rango**
- **corrigiendo inconsistencias**

Data Transformation: Transformamos unos datos en otros equivalentes y así dejarlos listos para la minería

- **Suavizado o eliminación del ruido**
- **Agregación:** agregamos valores de atributos. (Combinar)
- **Generalización:** mediante jerarquías de conceptos, sustituimos valores categóricos o numéricos por otros valores más abstractos. (Sustituir)
- **Normalización:** escalamos el atributo a un cjto. de valores apropiado según el caso
    - Min/max
    - Media cero
    - Escalado decimal
- **Construccion de atributos**

Data Reducctión:

Cuando el cjto. de datos es realmente grande, es posible que las técnicas de minería de datos disponibles no sean adecuadas, hasta tal punto que sea totalmente imposible realizar un proceso de minería.

- Agregación en data cubes
- Reducción de dimensiones
- Compresión de datos
- Reducción de la numerosidad (i.e. del número de tuplas)
- Discretización de atributos y generación de jerarquías de conceptos.

Tema 7: Aprendizaje de Arboles de decisión.

Construcción de árboles: Se hace al hacer crecer el árbol desde la raíz: se **disgrega** de manera sucesiva cada una de las hojas hasta alcanzar un nivel de profundidad determinado. Cuando existe una partición exacta de los datos, la construcción del árbol es muy simple Si no existe, tenemos solapamiento -> **Sobreaprendizaje** (soluciones: parar de crecer el árbol antes de terminarlo o podarlo una vez construido).

Algoritmo básico de aprendizaje:
Estrategia de búsqueda top-down, del tipo greedy.
El espacio de búsqueda es el formado por todos los árboles de decisión posibles. El algoritmo por excelencia es el **ID3**.
Se comienza buscando cual será el nodo raíz (resuelve aplicando un test estadístico).
Se desarrolla una rama para cada posible valor.
En los nuevos nodos se vuelve a hacer la misma pregunta Así hasta desarrollar un árbol completo (en la versión básica)

ID3 -> para problemas de clasificación, relación de casos formados por pares <atributo, valor>. Una vez aprendido el árbol, se usa para la clasificación.

ID3(Ejemplos, Etiquetas, Atributos)

- Paso 0: Definición
  - Sea `Ejemplos` el conjunto de ejemplos,
  - `Etiquetas` es el conjunto de posibles clases.
  - `Atributos` es el cjto. de atributos en los datos.
- Paso 1: Crear un nodo raíz para el árbol.
- Paso 2: Si todos los ejemplos son positivos, devolver el nodo raíz, con etiqueta $+$.
- Paso 3: si todos los ejemplos son negativos, devolver el nodo raíz, con etiqueta $-$.
- Paso 4: Si Atributos está vacío, devolver el nodo raíz, con el valor de Etiquetas más probable en Ejemplos.
- Si no
  - Inicializar $A \leftarrow$ atributo que *mejor* clasifica Ejemplos.
  - Hacer que el nodo root tenga como atributo de decisión al atributo A.
  - Ahora $\forall v_i \in A$
    - ★ Añadir arco bajo raíz, con test $A = v_i$
    - ★ Sea $Ejemplos_{v_i}$ el subconjunto de Ejemplos con valor v_i en el atributo A.
    - ★ Si $Ejemplos_{v_i} = \emptyset$ añadir un nodo hoja al arco que acabamos de añadir con la etiqueta de Etiquetas más probable en Ejemplos.
    - ★ Sino añadir al nuevo arco el subárbol generado por $ID3(Ejemplos_{v_i}, Etiquetas, Atributos - \{A\})$
- Paso 5: Devolver el nodo raíz

Disgregación de nodo hoja:

Dependiendo del número de valores por atributo:

- Si los atributos son binarios, disgregación binaria.
- Si los atributos son categóricos con L (>2) valores, considerar disgregaciones binarias hasta haber realizado tests para los L valores.
- Si los atributos son ordinales, los tests en las disgregaciones serán del tipo x ≤ xc
- Combinaciones lineales de atributos continuos y expresiones lógicas.

En cada hoja, se va a disponer de un conjunto de atributos candidatos para usar en la siguiente disgregación. Sea D × C el **espacio cartesiano** formado por los datos de entrada (**D**) y sus clases (**C**). Y sea una hoja A (con valores a1, …, am) del árbol a disgregar, La distribución de probabilidad sobre esos valores y las clases y la hoja hija correspondiente al test A = ai viene dada por **p(k|ai) = pik /pi** (a través de las diferentes k clases, siendo pi sobre D y pik sobre D × C).

Mediadas de impureza

La elección del siguiente atributo a disgregar podría basarse en comprobar si el conjunto de los nodos hijos generados a partir de la combinación hoja+atributo son más puros que el padre. Un nodo será más puro en la medida en que más ejemplos que contiene pertenecen a una sola clase.
La siguiente generación es más pura si la media de la pureza de los hijos ponderada por el tamaño de cada hijo es mayor que la del padre.

Descansito &Coins

Gemini + 

¿Sabrías identificar en qué te puede ayudar Google Gemini para estudiar?

REGLAS

1. Observa las opciones disponibles
2. Responde como Gemini te ayuda a estudiar.
3. Gana Wuolah coins para descargas sin publi.

Fácil | 10

A. Sintetiza horas de investigación en minutos.

B. Convierte tus apuntes en un esquema visual.

C. Prepara un examen para autoevaluarte.

D. Convierte tus apuntes en podcasts.

E. Sube hasta 1.500 páginas y analiza textos largos.

F. Todas las anteriores.

Gemini

Para calcular el valor de la disminución de impureza al escoger un atributo para disgregar tenemos que agregar los valores de la función de (im)pureza escogida (p.e. **Entropía o Gini**) para cada uno de los nodos hijos (por el ”tamaño” de cada uno).

Dado un atributo A, la disminución en impureza promedio (a maximizar) al usar ese atributo en la disgregación vendrá dada por

$$i(p_c) - \sum_{i=1}^{m} p_i \times i(p(c|a_i)),$$

siendo

- $i(p_c)$ la impureza del nodo padre
- $i(p(c|a_i))$ la impureza en el correspondiente nodo hijo generado a partir de un test para el valor a_i de A
- $\sum_{i=1}^{m} p_i \times i(p(c|a_i))$ la impureza generada en los hijos

La disgregación en ID3: Medida básica → **ganancia de información**. La medida de error de clasificación no es diferenciable Entropía y Gini son más precisas al reflejar cambios en probabilidades de nodos

BUSQUEDAD Y OVERFITTING

El espacio de búsqueda es completo.
El tipo de busqueda es top-down con estrategia hill-climbing (o greedy) por lo que no hay backtracking (mínimos locales)
Búsqueda basada en probabilidades (robustez al ruido)
Tendencia en la búsqueda: preferencia por los árboles con caminos a las hojas más cortos desde la raíz

ID3 puede adolecer de overfitting.

- El conjunto de ejemplos no es lo suficientemente representativo
- Los ejemplos tienen errores

Def **Overfitting**: Dado un espacio de hipótesis H, se dice que una hipótesis particular h ∈ H sobreajuste los datos de entrenamiento si existe una hipótesis alternativa h ' ∈ H, tal que h presenta un error menor que h ' sobre los ejemplos de entrenamiento, pero h ' presenta un error menor que h sobre el conjunto total de observaciones.
Para evitarlo: Parar de construir el árbol, antes de que este clasifique perfectamente todos los ejemplos (**pre-prunning**). Se puede construir el árbol y después intentar una poda (p**ost-prunning**).

Incorporación de **valores continuos**: transformamos el dominio de los continuos en una serie de intervalos ▶ Sea A un atributo continuo, el nuevo Ac será true si A < c y false si A ≥ c.
Valores nulos: Al calcular Ganancia(S, A), siendo < x, c(x) > un ejemplo de S para el cual el valor A(x) no se conoce -> Calcular valor más probable, calcular valor más probable de entre los ejemplos de c(x), asignar probabilidades a los distintos valores de un atributo.

Atributos con relevancias diferentes: Podemos necesitar diferenciar atributos en términos de su coste

CONTROL DEL TAMAÑO:
Un árbol balanceado, y con un número pequeño de nodos es más fácil de analizar. "pequeños disjuntos" → clasifican pocos casos.

- **Control de tamaño**. En este enfoque se intenta controlar el tamaño, bien intentando construir un árbol limitado, podándolo ulteriormente o ajustándolo on-line.
    - **Pre-prunning:** imposición de un criterio, no trivial, con el que parar de expandir el árbol.
    - **Post-prunning:** eliminación de subárboles después de la construcción del árbol total.
    - **Reajuste incremental:** si se mantienen los casos de entrenamiento, y se van acumulando los subsiguientes casos que vayan apareciendo el árbol puede ir actualizándose on-line.
- Modificación del espacio de los tests
    - La construcción de test dirigida por datos se basa en construir nuevos atributos mediante combinación de atributos base mediante operadores numéricos, por combinación de estos por operadores lógicos
    - En la construcción de tests dirigida por hipótesis se almacenan los tests construidos según la forma anterior, (antiguos ´arboles) ▶ los ´arboles que van construyéndose (lo buenos que han sido) influyen en la decisión de si aplicar esos test más adelante o no.
- Modificación de la búsqueda de tests. se puede modificar la búsqueda usando una diferente a la de tipo greedy
    - Nuevas medidas de selección: podemos usar medidas alternativas a la ganancia como por ejemplo la basada en el principio MDL (Minimun Description Length), la medida de Kolmogorov-Smirnoff, separabilidad de clases, etc.
    - Uso de atributos continuos: la discretización de atributos continuous se puede realizar de tal forma que se reduzca el sesgo y la selección de tests en atributos continuos.

- Búsqueda anticipada: en esta, se construyen en forma tentativa subárboles usando determinados tests, y se evalúan a una profundidad suficiente. Así, se puede escoger el tests que mejores resultados ha dado con más información que si se realizara una simple búsqueda greedy.

- Restricciones en los datos
  - Podemos eliminar casos o atributos
- Estructuras de datos alternativas
  - Una vez se ha construido el árbol, podemos convertir este en una estructura diferente, más compacta
  - grafos y
  - reglas de decisión

Tema 8: Aprendizaje no supervisado.

El aprendizaje no supervisado trata de buscar alguna estructura en el conjunto D.
Fases del aprendizaje no supervisado:
1. Encontrar una k-partición en D, de k conjuntos exhaustivos y mutuamente exclusivos (cada partición es un **clúster**)
2. Diseñar un clasificador basado en las etiquetas asignadas a las particiones.
Buscaremos una solución de compromiso entre una varianza pequeña y un número de clústeres bajo (1pto solo -> var 0, pero no nos sirve).

El escalado (normalización) en estos casos es importante, para que todos los componentes/dimensiones en los datos tengan la misma importancia.

1. **Clustering**
   a. **Distancias:** basa su funcionamiento en una medida de similitud entre puntos del espacio.
   Algoritmo:
   1. Sean c1 , c2 , ..., ck puntos aleatoriamente seleccionados en un espacio ndimensional
   2. Para cada xi en D, hacer
      1. Escoger el cj más cercano a xi
      2. Modificar cj
   Cada uno de los ci se denomina prototipo o centroide del clúster. No tiene porqué ser uno de los xi.
      i. **Algoritmo K-medidas:** El problema general es mapear de un espacio continuo a uno discreto: **vector quantization**.
      Supongamos una muestra X={x^1 ,x^2 ,...x^N} y k vectores de referencia m1 ,...,mk. Una vez que tengamos los mj representaremos cada x^t con su entrada en el mapa más similar. Debemos encontrar unos valores del alfabeto que minimicen dicho error. Se calculan los b, y luego se ajustan los m, que se

usan a su vez para calcular nuevos b.
Eficacia depende de la inicialización aleatoria de los mi. Los siguientes son métodos posibles para inicializar los m's
1. Seleccionar aleatoriamente k instancias que pasan a ser los mi . Método más simple.
2. Se puede usar el vector de medias de todos los datos, y para calcular los mi añadimos vectores aleatorios al vector de medias
3. Hacer un análisis de componentes principales, dividir su rango en k intervalos iguales, particionar los datos en k grupos y la media de dichos grupos como los centros iniciales
4. Podemos hacer que el propio algoritmo calcule sus clústeres (e.g. "clustering" jerárquico)

b. **Distribuciones de probabilidad:** Aquí el sesgo es asumir que los clústeres tienen una "forma" que sigue alguna distribución de probabilidad y el algoritmo busca los parámetros de esas distribuciones.
   i. **Algoritmo basado en Bayes.**
   Supongamos una partición P, del conjunto de datos D en k clústeres exhaustivos y mutuamente exclusivos C1 , C2 , ..., Ck
   Podríamos decidir a qué clúster pertenece cada patrón x mediante la selección de aquel Ci que hace máximo p(Ci |x).
   Mediante Bayes podemos hacerlo mediante p(x|Ci)p(Ci).
   Denotamos con S(x,Ci) a la similaridad del ejemplar x al clúster Ci de patrones. Seguimos calculando la media del clúster de la misma forma, con la media aritmética.
   ii. **Algoritmo EM:** Trata el problema en el que hay variables ocultas, pero se conoce la forma general de la distribución de probabilidad que siguen.
   Sea un D cuyas x se han generado mediante una distribución de probabilidad que es una mezcla de k distribuciones normales distintas.
   La tarea de aprendizaje es generar una hipótesis h=(μ1 , μ2 , ..., μk) que describa la media de las k distribuciones. Siguiendo el enfoque Bayesiano, queremos encontrar una hipótesis de mayor verosimilitud para estas medias, es decir aquella que maximice P(D|h).
   Podemos describir (para k=2) cada instancia del problema con (xi ,zi1,zi2) en donde
   • xi es el valor observado de la instancia i-ésima
   • zi1: 1 si el valor i-ésimo fue generado con la distribución 1, 0 en otro caso
   • zi2: 1 si el valor i-ésimo fue generado con la distribución 2, 0 en

otro caso.
Se busca la h de mayor verosimilitud mediante la estimación iterativa de los valores esperados para las zij, dada la hipótesis actual, i.e. (μ1 , μ2 , ..., μk). Luego se usan esos valores de zij para recalcular (μ1 , μ2 , ..., μk) de nuevo.
Existe la posibilidad de caer en máximos locales.

c. **Densidad:** tratan de delimitar zonas del hiperespacio donde existan una alta frecuencia de ejemplos y crea ”fronteras” en donde dejan de aparecer ejemplos. Dependiendo del algoritmo dejan (o no) “fuera” de los clústeres a los “outliers”

2. **Clustering difuso:** todos los puntos pertenecen a “todos” los clústeres, pero con cierto grado de pertenencia. Los algoritmos de agrupamiento “tradicionales” se suelen adaptan para permitir eso.
3. **Clustering jerarquico:** solamente se necesita como requisito una medida de distancia. El parámetro k no tiene sentido ya que se crea una jerarquía. Podemos usar distancias como Minkowksi (la mas general), o la distancia de Manhattan (más barata computacionalmente).
   a. **Aglutinador:** Comienza con N grupos, cada uno formado por uno de los ejemplares en D, en donde |D|=N mezclando grupos similares, según distancia, hasta llegar a un grupo único formado por todos los ejemplares en D.
   b. **Divisor:** comienza con un único grupo hasta llegar a grupos de un único ejemplar.

Cálculo de las distancias entre grupos:

”Clustering" de **enlace simple** (i.e. single-link): la distancia entre dos grupos es la más pequeña entre pares cualesquiera de ambos grupos (los puntos más cercanos de diferente grupo).
”Clustering" de **enlace completo** (i.e. complete-link): la distancia entre dos grupos es la más grande entre pares de ambos grupos (los dos puntos más lejanos de diferente grupo).
Otras posibilidades incluyen: enlace promedio y distancia entre centroides.
Si consideramos un grafo con costos conectado completamente, el método de enlace simple equivale al cálculo del spanning tree mínimo.

El parametro K.

Determinar el número de clústeres equivale a resolver el compromiso (i.e. trade off) entre complejidad del modelo y precisión alcanzada. Los siguientes son métodos para trabajar con el valor de k apropiadamente

• En determinadas aplicaciones, k vendrá dado
• Utilizar una distancia máxima para observaciones en un mismo clúster. Variarla

progresivamente.
• La validación de los clústeres puede hacerse por un experto en el dominio: se trata de comprobar si tienen sentido en dicho dominio.
• Representar el error de reconstrucción en función de k y buscar un cambio de tendencia en la curva (después de un cierto k el error no variará significativamente)

Agrupando en sub-espacios

La utilización de **esembles** es una forma de mejorar rendimientos los algoritmos de aprendizaje.
Se basa en combinar varios algoritmos, o varios modelos obtenidos con un mismo algoritmo (variando hiper-parámetros, o sub-sampleando los datos).
Esemble interesante para el clustering: se pueden hacer diversos agrupamientos diferentes seleccionando subconjuntos de las variables de entrada (una forma de selección de variables) Esto genera agrupamientos en sub-espacios que luego el esemble puede combinar en una decisión final. Se podrían incluso combinar diferentes algoritmos en esos modelos (model buckets) del esemble.

TEMA 9: Introducción al aprendizaje Bayesiano

En esta asignatura estamos interesados en encontrar la mejor h de H, dados unos datos D. En este tema la h será la decisión a tomar entre un conjunto H de posibles decisiones. P.e. en clasificación, cada h sería cada posible clase, y mediante Bayes calcularemos las probabilidades (o una medida relacionada) de cada h dada una instancia (nueva).
En el contexto de toma de decisión basada en Bayes, la mejor h es la más probable (dado D y junto a cualquier conocimiento de las probabilidades previas de las distintas h): el Teorema de Bayes proporciona un mecanismo para calcular esas probabilidades para cada h en H, a partir de tres elementos:
• los datos **D** de la muestra que definen nuestro problema.
• la **probabilidad previa de la h en H**. (En clasificación es la proporción de cada clase en la población general... que no necesariamente es la muestral)
• la **probabilidad condicionada** de observar determinados datos dado que se observa la hipótesis h en H. (Frecuencia de determinados valores de entrada por cada h).

Notación a usar de ahora en adelante
• P(h) es la **probabilidad previa** (antes de tener disponible D) de que se cumpla h (Si se desconoce se asume que todas las h son equiprobables)
• P(D) es la probabilidad previa (sin saber nada de las P(h)) de que vamos a observar D a partir de un muestreo (también se le llama "**evidencia**")
• P(D|h) es la probabilidad de observar D, dado que h se cumple (se denomina **verosimilitud** o likelihood)

• P(h|D) es la probabilidad de que se cumpla h, dada la muestra D, denominada **probabilidad posterior** de h. (la confianza de que sea h, cuando se ha observado D)

Las cantidades de interés siguen una distribución de probabilidad. Se pueden realizar decisiones óptimas mediante el razonamiento sobre esas probabilidades, si se combina con los datos observados.

Aspectos positivos del aprendizaje bayesiano:

• Cada ejemplar de aprendizaje observado puede variar la probabilidad estimada de que una hipótesis es correcta, de manera incremental (nuevos ejemplos de aprendizaje se pueden "añadir" al modelo sin problema)

• Es posible combinar algunos conocimientos previos, a saber:

(1) probabilidad previa de cada hipótesis posible (proporciones de las clases en la población general. y

(2) distribución de probabilidad de cada hipótesis en la muestra con muestras de datos (conocimiento posterior) para determinar la probabilidad final de una hipótesis

• El aprendizaje bayesiano trabaja bien con hipótesis que realizan predicciones probabilistas (p.e. la probabilidad de cura de este paciente es de 92%)

• Es posible clasificar nuevos ejemplares combinando predicciones de varias hipótesis ponderadas por sus probabilidades respectivas

Aspectos negativos: la estimación de las probabilidades previas no es algo trivial.

Probabilidad posterior = Verosimilitud * Probabilidad previa/ Evidencia:

$P(h \mid D) = P(D \mid h) \cdot P(h) / P(D)$

Estamos interesados en encontrar la h en H más probable, dado D.

Bayes Naive

El algoritmo Bayes naive nace de la relajación del problema: la relajación ingenua consiste en considerar los valores de los atributos como condicionalmente independientes

$$P(a_1, a_2, ..., a_n \mid v_j) = \subseteq\!\!\!\subseteq_i P(a_i \mid v_j)$$

Con lo que obtenemos, a partir de la expresión anterior, el clasificador Bayes Naive, según

$$v_{BN} = \arg\max_{v_j \llcorner V} P(v_j) \subseteq\!\!\!\subseteq_i P(a_i \mid v_j)$$

Ahora el número de terminos $P(a_i|v_j)$ distintos es mucho menor (número de distintos valores de cada a_i por distintos v_j) y se pueden estimar a partir de D.

El algoritmo se reduce a estimar los $P(v_j)$ y los $P(a_i|v_j)$ contando apariciones en el cjto. D. Cada nueva instancia se clasifica mediante la expresión de arriba.

Clasificador Gaussian Bayes Naive:

Para atributos continuos se asume una distribución normal de los valores de los atributos por cada variable y para cada clase. Se segmentan los datos por clase vj , y para cada variable continua de la entrada (por ejemplo, la í-esima) se calcula la media y la varianza de los valores de los atributos de esa variable usando solo los ejemplos de dicha clase vj.

Aprendizaje con variables no observadas

Existen muchos problemas en los que se debe aprender en ausencia de valores de algunas de las variables (e incluso con variables ocultas que nunca se observarán)
Solución: es mejor aun usar probabilidades y, o bien generar un dato, o bien "distribuirlo". Ejemplo de problema (hidden markov chain), múltiples bolas en una urna. La variable observada se corresponde con la bola obtenida, la variable no observada representa de qué urna se obtuvo.

El objetivo es modelar estadísticamente las variables ocultas y luego darle utilidad. Un algoritmo muy útil para este tipo de problemas es alguna versión del E-M que vimos en clustering.

Los métodos bayesianos permiten incluir en el aprendizaje conocimiento a priori sobre:
• las posibles hipótesis con que clasificar nuevas instancias
• las probabilidades de observar determinados datos, dadas las hipótesis
• El método Bayes naive es sencillo y está a la altura de redes neuronales y árboles de decisión. Se denomina naive porque asume independencia condicional en los datos de las observaciones
• El algoritmo EM se puede usar en problemas en los que parte de los datos no se conocen, pero sí su distribución de probabilidad. Así se pueden estimar los valores de dichas variables ocultas.