Resumen teórico - Aprendizaje Automatizado

Los problemas que se resuelven mediante aprendizaje automatiazdo suelen ser problemas simples de entender o reconocer pero difíciles de definir. Puede ser no trivial diseñar algoritmos sobre ellos.

El AA introduce métodos que resuelven problemas **aprendiendo** la solución a partir de ejemplos. Un programa **aprende** si mejora su desempeño en una cierta tarea al incorporar experiencia. Al aprender se **generaliza** (no se memoriza).

Para generalizar se detectan tendencias (*bias*) en los datos de entrada. En general se usa el principio de navaja de Occam: la explicación más sencilla suele ser la más probable.

1. Clasificación de problemas

- Aprendizaje supervisado: la entrada es un conjunto de instancias etiquetadas o clasificadas
  - Problemas de clasificación: dado un conjunto de datos descripto por un conjunto de características se asignan una (o varias) etiquetas(s) de un conjunto finito de etiquetas.
    Ejemplos: asignar un símbolo alfanumérico a una secuencia de movimientos del lápiz en una pantalla táctil.
  - Regresión: dado un conjunto de datos se asigna a cada elemento un número real. Suelen ser modelos predictivos
    Ejemplos: predecir la relación euro/dolar a futuro, predecir el clima.
- Aprendizaje no supervisado: la entrada es un conjunto de instancias sin clasificar. Se aprende mediante exploración autónoma.
  - Ranking-retrieval / Búsqueda-recomendación: dada una query, se asignan y ordenan respuestas dentro de una base de datos.
    Ejemplos: buscadores en internet, sistemas de recomendación
  - Detección de novedades: se detectan *outliers*, es decir, objetos que son sustancialmente difeferente a los otros mediante alguna característica en particular.
    Ejemplos: comportamiento de compras con tarjeta, detección de fallas en sistemas críticos.
  - Clustering: el equivalente a problemas de clasificación pero con aprendizaje no supervisado. Los resultados de aprender un modelo en este tipo de problemas suelen ser funciones *frontera* que separan los datos según una característica en particular.
  - Aprendizaje por refuerzo: el sistema aprende por prueba y error. Realiza una exploración autónoma para inferir reglas de comportamiento por lo cual se requieren muchas repeticiones de una misma tarea para adquirir experiencia. Una desventaja de esta técnica es que requiere un elevado número de repeticiones para ganar experiencia.
    Ejemplos: robot que navega por una oficina, software que aprende a jugar al ajedrez
  - Aprendizaje deductivo (EBL)
  - Razonamiento basado en casos (CBR)

2. Aprendizaje inductivo

Se genera un modelo a partir de ejemplos específicos.
Elementos fundamentales:

- Modelo resultante (hipótesis): resultado del proceso de aprendizaje
  Ejemplos de modelos: árboles de decisión, lista de reglas, redes neuronales, modelos bayesianos / probabilísticos, etc.
- Instancias: cada ejemplo particular
- Atributos: propiedades o características que se miden u observan de las instancias.
  Tipos de atributo: real, discreto, categórico (discreto no ordenado).
- Clases: atributo que debe ser deducido a partir de los demás

3. Entrenamiento y validación de un modelo

En general el conjunto de datos de entrada de un algoritmo de AA se debe separar en al menos 3 partes

- Conjunto de entrenamiento: datos que van a alimentar al algoritmo.
- Conjunto de validación: conjunto de datos que se utiliza para determinar si se debe seguir entranando.
- Conjunto de testeo: conjunto de datos que se utiliza para estimar el error con el que infiere/predice el modelo. Trata de estimar qué tan bueno va a ser el modelo para nuevas entradas.

4. Árboles de decisión

Son árboles en donde cada nodo interno está etiquetado con pares atributo-valor y cada hoja corresponde a una clase. Cada par de ramas de un nodo se corresponde con una regla lógica if-then-else en el caso de que se clasifique en 2 clases.

El objetivo al generar un modelo de este tipo es que el árbol sea mínimo. Como este problema es NP-completo, se utilizan algoritmos de búsqueda local. El aprendizaje a veces se realiza sobre ensambles de árboles de decisión (bosques).

4.1. Algoritmo para generar árboles de decisión

Se seleccionan atributos comenzando desde el nodo raíz eligiendo el que maximice la **ganancia de información**, siguiendo el **método de Gini** o el que **reduzca más la varianza**. Esta selección se realiza recursivamente para los subárboles hasta que:

- todos los datos se corresponden a una sola clase o
- la partición no agrega valor a la predicciones (condición de parada) o
- no quedan más datos que etiquetar o
- no quedan más clases para etiquetar

ID3 vs C4.5 El algoritmo ID3 (*interactive dichotomizer v3*) funciona como el algoritmo descripto, realiza búsqueda hill-climbing de lo simple a lo complejo y no incorpora backtracking. El algoritmo C4.5 se puede aplicar también a atributos continuos (se discretizan): en cada nodo queda seleccionado un atributo y un umbral o valor de corte. Este último algoritmo requiere que se settee un mínimo de ganancia para cada nodo.

ID3 nunca produce árboles demasiado grandes en contraposición con C4.5 que puede repetir atributos con diferentes umbrales.

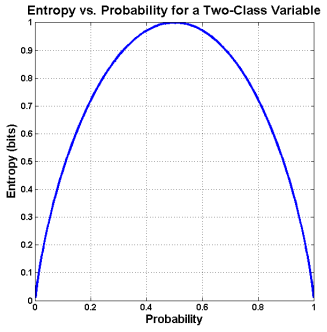
4.2. Medidas que se utilizan en el argoritmo de generación de árboles

Entropía Medida de incertidumbre en un sistema, es decir, la probabilidad de que ocurra cada uno de los posibles resultados. Si la incertidumbre es máxima, 1, todos los resultados tienen la misma probabilidad de ocurrir.

La entropía se calcula como

$$E(S) = -p\oplus \log_2(p\oplus) - p\Theta \log_2(\Theta)$$

donde $p\oplus$ es la proporción de ejemplos positivos y $p\Theta$ es la proporción de ejemplos negativos.



Ganancia de información Al elegir un atributo como siguiente nodo del árbol se está **ganando información** cuando la clasificación envía instancias con clases distintas a los distintos subárboles. La ganancia de información mide la reducción esperada de entropía sabiendo el valor del atributo A.

$$G(S, A) = E(S) - \sum_{v \in valores(A)} \frac{|Sv|}{|S|} E(Sv)$$

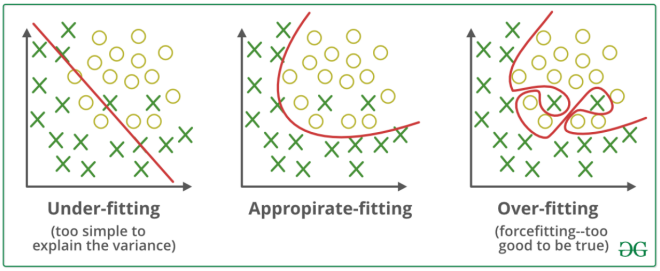
donde $valores(A)$ es el conjunto de posibles valores del atributo A y Sv es el subconjunto de S en el cual el atributo de A tiene el valor v.

Método de Gini La impureza de Gini se puede calcular sumando la probailidad de elegir cada elemento por la probabilidad de error de categorización de ese elemento. Alcanza el mínimo 0 cuando todos los casos del nodo corresponden a una sola categoría.

4.3. Sobreajuste - *Overfitting*

En una hipótesis (modelo) se dice que existe sobreajuste o sobreentrenamiento si existe otra hipótesis que tiene mayor error sobre los datos de entrenamiento pero menor error sobre todos los datos.

Un árbol de decisión muy grande puede producir sobreajuste, se ajusta demasiado a los datos de entrenamiento. El sobreajuste se produce también cuando se trata de satisfacer datos ruidosos como por ejemplo una lectura ruidosa o una muestra chica (no es lo suficientemente representativa). Cuando hay sobreajuste el programa no aprende (generaliza) sino que memoriza.



Estrategias para mitigar el sobreajuste en un árbol sobreentrenado:

- establecer una cota para para el crecimiento del árbol
- incorporar una etapa de post-procesamiento (*prunning*)

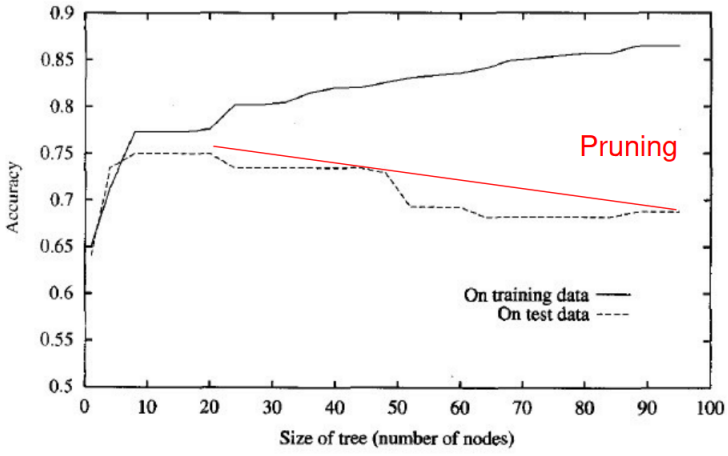


Figura 1: Poda de un árbol en base a la métrica Accuracy

Ambas estrategias se apoyan en **estadísticas** o el verificaciones con el conjunto de datos de **validación**.

4.4. Problemas apropiados

- Las instancias se pueden representar como pares atributo-valor
- La función objetivo tiene valores discretos o discretizables
- Los datos de entrenamiento pueden contener errores
- Los datos de entrenamiento pueden estar incompletos

5. Evaluación de hipótesis / modelo

Es posible estimar medidas de error sobre muestras finitas sin sesgos y con varianza que decrece con el tamaño de la muestra.

RMSE (*Root-Mean Squared Error*)

$$RMSE(f) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2}$$

donde m es el número de instancia en el test, $f(x_i)$ es el output que predice el modelo e y_i son los varores reales.

Tiene la ventaja (y desventaja) de no tener unidad: se deben conocer los rangos de valores en los que varían las variables para determinar si el error es alto o bajo. Por esta razón a veces se suele normalizar para no tener que depender de la variabilidad de la entrada.

En general, el error disminuye a medida que aumenta el tamaño de la muestra

5.1. Matriz de confusión

Sintetiza gráficamente los resultados de un predicción. Distingue los falsos positivos y los falsos negativos de las predicciones acertadas.

↓ predicción / clase real →	T	F
T	TP	FP
F	FN	TN
	P = TP + FN	N = FP + TN

Tabla 1: Matriz de confusión para 2 clases

Algunas métricas que se calculan sobre los valores de la matriz de confusión son

- Multi-clase
  - Accuracy = (TP + TN) / (P + N) proporción de elementos correctamente clasificados
- Máximo 2 clases
  - Precision = TP / (TP + FP) cantidad de positivos reales que se predijeron respecto a los que arrojó el modelo
  - Recall = TP / P cantidad de positivos que clasificó el modelo del total de positivos reales
  - Fallout = FP / N
  - Sensitivity = TP / P
  - Specificity = TN / N

Recall y Sensitivity son equivalentes, ver

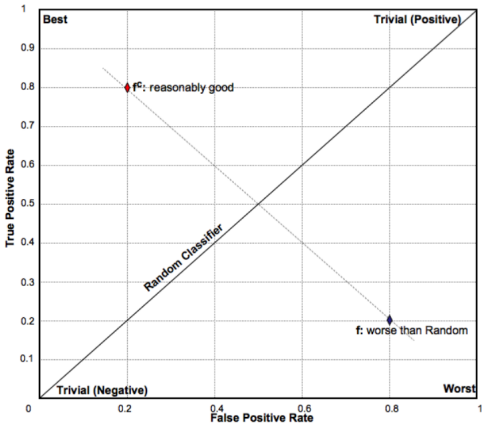
Además

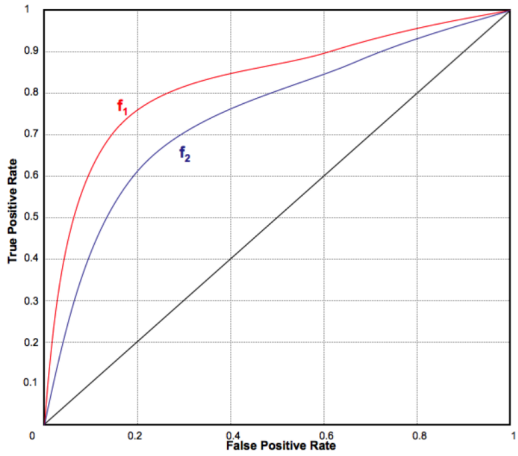
- FPR (false positive rate) = FP / (FP + TN) = 1 - Specificity
- TPR (true positive rate) = TP / (TP + FN) = Sensitivity

Consideraciones sobre las métricas

- Si 2 modelos tienen igual Accuracy, se debe elegir el que mejor Recall tenga si se quiere priorizar la clasificación de los positivos o el que mejor TN/N tenga si se quiere priorizar la clasificación de los negativos.
- Mayor Accuracy no significa necesariaamente que el clasificador sea más sofisticado (un clasificador que es preciso en el 100 % de los positivos pero le erra a todos los negativos no es sofisticado). Esto ocurre cuando los datasets están muy desbalanceados.
- A misma Precision y Recall se pueden esperar comportamientos muy diferentes respecto a la predicción de negativos. Es necesario usar Accuracy para desempatar.

Curvas ROC Se utilizan para ver el desempeño de clasificadores sobre distintos rangos. Cada punto en la curva representa un umbral distinto de corte para varibales con 2 clases. El área bajo las curvas ROC (AUC) permite comparar clasificadores.





5.2. Testear el modelo

- Usar los mismos datos para entrenar y testear
  - Ventajas: no disminuye el tamaño del conjunto de entrenamiento
  - Desventajas: el resultado es optimista. El desempeño del modelo es un sobreajuste completo.
- Separar un conjunto para test antes de entrenar los datos y evaluar las métricas sobre el conjunto de testeo.
  - Ventajas: 1) los conjuntos de entrenamiento y testeo son independientes 2) es una buena estimación para cualquier clasificador.
  - Desventajas: disminuye el tamaño del conjunto de entrenamiento
- *Re-sampling*: se separa el conjunto de entramiento y testeo n veces
  - Elegir cada vez un conjunto random. Hay superposición en los conjuntos de entrenamiento y en los conjuntos de testeo. Además por cada resampleo los conjuntos de entrenamiento y testeo no son independientes.
  - *K-fold cross validation*: Elegir cada conjunto de forma de que los conjuntos de testeo no se superpongan. Método más utilizado. También se puede utilizar este método para ajustar parámetros a partir del conjunto de datos de validación.
    - Ventajas: 1) se utilizan todos los datos para entrenar, 2) Los conjuntos de testeo son independientes entre sí 3) al promediar los errores se consigue una buena estimación del error para datasets de gran tamaño.
    - Desventajas: en cada fold los conjuntos de entrenamiento y testeo se superponen levemente.

En general, tener pocos datos para entrenar hace que el modelo no sea general y tener pocos datos para testear puede arrojar medidas no confiables sobre el modelo.

Observaciones Las muestras tienen que ser representativas de la variedad de clases. En particular, en el caso de los k-fold, cada uno de los sampleos debe ser representativo. Cuando esto no se puede lograr, se debe tratar de maximizar qué tan representativo es cada muestro.

6. Redes neuronales artificiales

Es un sistema de tratamiento de la información cuya unidad básica está inspirada en la neurona. El potencial del sistema está en las conexiones entre neuronas.

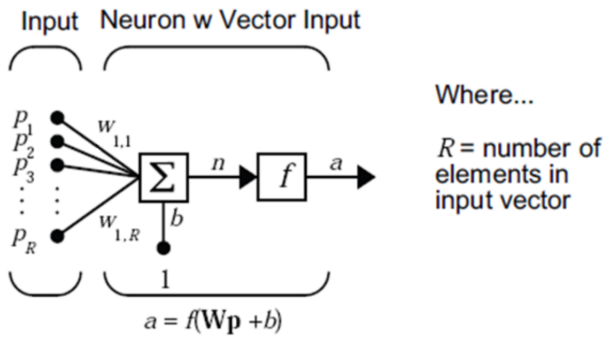
6.1. Problemas apropiados

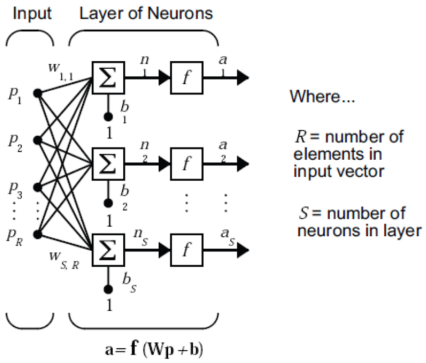
- regresión y clasificación multiclase
- entradas de alta dimensión
- entradas con errores
- la función objetivo aprendida debe poder evaluarse rápidamente
- la comprensión de la función objetivo no es importante (modelo caja negra)

Estructura artificial

- capa de entrada: recibe estímulos externos
- capa(s) oculta(s): elementos internos de procesamiento
- capa de salida: reciben la información procesada y retornan la respuesta del sistema el exterior

Neuronas





Funciones de transferencia Las funciones de transferencia más utilizadas son *hardlim*, *purelim*, *logsig* y *tansig* (escalón perceptrón, lineal adaline, sigmoide, y tangente hiperbólica).

Arquitectura

- Cantidad de capas, cantidad de neuronas por capa, función de transferencia por capa
- Tipos de conexiones permitidas: solo entre capa n y n+1? todas con todas?

Mecanismo de aprendizaje Una vez determinada la arquitectura de la red neuronal se deben ir **ajustando los pesos** y los **umbrales** (*biases*) de cada neurona en cada capa para que la salida ajuste el objetivo. El aprendizaje puede ser supervisado o no supervisado.

6.2. Modelos de redes artificiales - Redes feedforward

Las neuronas de una capa se conectan con las neuronas de la capa siguiente. Son útiles en aplicaciones de reconocimiento o clasificación de patrones.

Red	Arquitectura	Función de transferencia	Tipo de frontera	Aprendizaje
Perceptrón	1 capa oculta de S neuronas	Escalón	Recta*	Supervisado
Adaline	n capas de s_i neuronas	Lineal	Recta*	Supervisado
Backpropagation	n capas de s_i neuronass	Sigmoidea + Lineal	Cualquier función**	Supervisado

* problemas linealmente separables ** con un número finito de discontinuidades.

6.2.1. Perceptrón

Aprendizaje El valor t es el valor deseado, a el resultante y e su diferencia. Se quiere disminuir e a partir de la modificación del vector de pesos w.

- a = t entonces el vector no se modifica
- a = 0, t = 1. El vector de entrada p se suma a w
- a = 1, t = 0. El vector de entrada p se resta a w

entonces la regla de aprendizaje es

$$W_i \leftarrow W_i + \Delta W_i, \Delta W_i = \eta(t-a)p = \eta ep$$

donde η es un rate pequeño para dar pasos chicos. Si b = 1 entonces b' = b + e.

El problema del XOR El perceptrón no puede modelar la operación lógica XOR ya que la frontera no se puede modelar con una recta. El XOR no es un problema linealmente separable.

6.2.2. Adaline

Aprendizaje Similar al Perceptrón pero busca minimizar el error cuadrático medio RMSE. Para esto se utiliza la noción de gradiente: se trata de ir en contra de la dirección de máximo crecimiento.

$$W_i \leftarrow W_i - \alpha \nabla E, \nabla E = \frac{1}{Q}\sum_{k=1}^{Q} \nabla e^2(w), e^2(w) = (t-a(w))^2$$

donde α es un rate pequeño para dar pasos chicos.

6.2.3. Backpropagation

Admiten múltiples capas y funciones derivables no lineales. Los pesos se ajustan hacia atrás (desde la capa de salida hacia la capa de entrada): el nuevo peso deseado en la capa s es el target de la capa $s-1$.

La función sigmoidea se elige porque es derivable en todo punto, tiene crecimiento suave y tien derivada finita. Como se utiliza descenso al gradiente la derivada debe ser sencilla de calcular.

Se puede probar que una red de 2 capas donde la primera capa es sigmoidea y la segunda es lineal se puede entrenar para aproximar cualquier función con una cantidad finita de discontinuidaes.

Aprendizaje Se puede llegar a minimos locales → se utilizan heurísticas para evitar este problema (momentum term, stochastic gradient descent, training multiple networks, etc).

Poder expresivo Este tipo de redes puede representar funciones booleanas, funciones continuas y funciones arbitrarias, entre ellas, funciones provenientes de problemas no linealmente separables.

6.3. Validación y testing

Se suele dividir el dataset en 3 conjuntos: conjunto de entrenamiento (ajuste de pesos y umbrales), conjunto de validación (ajuste de learning rate y otros parámetros) y conjunto de test (evaluación del modelo).

Se puede producir **overfitting**: es muy fácil sobreajustar por el poder de expresividad que tienen las redes. Generalmente, las fronteras de decisión muy complejas producen sobreajuste. Los conjuntos de validación suelen detener el proceso de entrenamiento según la cantidad de **épocas** (*epochs*).

7. Comparación entre Árboles de decisión y Redes neuronales

	Árboles de decisión	Redes neuronales
Capacidad de representación	No muy alta: superficies de decisión perpendiculares a los ejes	Alta: fronteras no lineales
Legibilidad	Muy alta	Nula
Tiempo de cómputo on-line	Rápido: recorrer árbol hasta una hoja	Rápido: sumas y multiplicaciones
Tiempo de cómputo off-line	Rápido: algoritmos simples	Muy lento
Parámetros a ajustar	Tamaño del árbol	Muchos: estructura, learning rate, condiciones de terminación
Robustez	Alta	Alta
Sobreajuste	Puede ocurrir: podar o limitar crecimiento	Puede ocurrir: limitar mediante conjunto de validación