

# Aprendizaje Supervisado

Inteligencia Artificial

Profesor: Alejandro Figueroa

# Introducción

- Aprendizaje supervisado es también conocido como **clasificación** o **aprendizaje inductivo**.
- La idea principal es aprender de la experiencia pasada con el objetivo de mejorar nuestra habilidad de llevar a cabo tareas del mundo real.
- Esas “experiencias” se refieren a datos.

# Conceptos Básicos

- Un conjunto de datos consiste en un conjunto de registros que se describen mediante un conjunto de atributos  $A=\{A_1, A_2, \dots, A_{|A|}\}$ .
- Donde  $|A|$  denota el número de atributos en  $A$ .
- Los datos también tienen un atributo especial llamado la clase que se considera aparte de  $A$ .
- El atributo clase puede tomar un valor dentro de un conjunto de valores  $C=\{C_1, C_2, \dots, C_{|C|}\}$ .

# Conceptos Básicos

- Donde  $|C|$  es el número de clases que es mayor o igual a dos.
- El valor del atributo clase también es llamado etiqueta o anotación.
- En el lenguaje de minería de datos, un evento de la “experiencia” es llamado un ejemplo, una instancia, un caso o un vector.
- Un conjunto de datos es un conjunto de vectores.
- Cada conjunto de atributos es una vista de la colección.

# Conceptos Básicos

- El objetivo del aprendizaje es producir una función de predicción/clasificación.
- Esta función puede ser utilizada para predecir etiquetas de ejemplos futuros.
- La función es llamada también modelos de clasificación, función discriminante, modelo predictivo o simplemente clasificador o etiquetador.

# Conceptos Básicos

- La función discriminante inferida desde los datos, depende largamente de la vista generada de los datos.
- Es decir, tendremos diferentes funciones discriminante de acuerdo al conjunto de atributos escogidos para modelar los datos.
- Ergo, hay vectores o conjuntos de atributos más eficientes que otros.
  - ¿Cuál es el más eficiente?

# Conceptos Básicos

- Los más eficientes son los atributos que permiten etiquetar ejemplos no-vistos (principalmente) de manera más efectiva.
- ¿Qué es más efectivo? Eso depende de las métricas utilizadas.
- Nótese que a los atributos, en la jerga de aprendizaje supervisado, también se les llama: características, features y propiedades.

Age	Has_job	Own_house	Credit_rating	Class
young	false	false	fair	NO
young	false	false	good	NO
young	true	false	good	YES
young	true	true	fair	YES
young	false	false	fair	NO
middle	false	false	fair	NO
middle	false	false	good	NO
middle	true	true	good	YES
middle	false	true	excellent	YES
middle	false	true	excellent	YES
old	false	true	excellent	YES
old	false	true	good	YES
old	true	false	good	YES
old	true	false	excellent	YES
old	false	false	fair	NO

# Conceptos Básicos

- En la tabla vemos un caso típico relacionado con la postulaciones a préstamos.
- Vemos cuatro atributos:
  - El primer atributo es “*edad*” y puede tomar tres valores: joven, mediana y viejo.
  - El segundo atributo es “*tiene\_trabajo*” que indica si el postulante tiene un trabajo.
  - El tercer atributo es “*es propietario de una casa*” que muestra si el postulante tiene una casa o no.
  - El cuarto atributo es el “*ranking crediticio*” que tiene tres posibles valores: aceptable, bueno y excelente

# Conceptos Básicos

- La etiqueta/clase es “si” o “no”.
- Entonces la idea sería aprender una función que pueda clasificar aplicaciones a créditos futuras. Es decir, si es aprobada o no.
  - Dado una combinación de valores de los cuatro features escogidos para modelar los datos.
- La tarea se llama aprendizaje supervisado porque las etiquetas son dadas (“si” y “no”).

# Conceptos Básicos

- Cuando no tenemos las etiquetas hablamos de “aprendizaje no-supervisado”.
- Alguna jerga:
  - Los datos utilizados para entrenamiento (experiencia) se llama “**datos de entrenamiento**”.
  - Lo que algoritmo aprende es un **modelo**.
  - El modelo es evaluado en un set de prueba o “**datos no-vistos**”. Datos no utilizados para entrenar.

# Conceptos Básicos

- Los datos de prueba no son utilizados para aprender el modelo. Por lo general, también tienen etiqueta.
- Estas etiquetas sirven para evaluar los modelos aprendidos.
- La precisión de un modelo de clasificación está definida por:

$$Accuracy = \frac{Number\ of\ correct\ classifications}{Number\ of\ test\ cases}$$

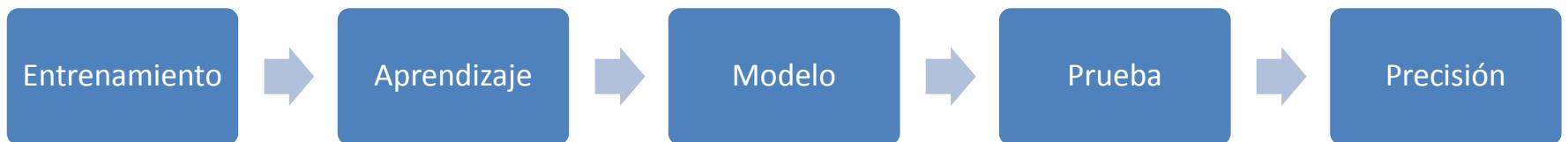
# Conceptos Básicos

- Para exemplificar, imaginemos que tenemos que decidir aleatoriamente o de acuerdo a la clase mayoritaria.
- La clase mayoritaria en el ejemplo es “si”.
- Si los datos de prueba tienen la misma distribución que los datos de entrenamiento, tendríamos  $9/15 = 0.6$ , si asignamos a todos “si”.
- ¿Se puede hacer mejor?

# Conceptos Básicos

- La hipótesis fundamental del aprendizaje automático es que los datos de entrenamiento tienen una distribución similar a los datos de prueba, incluyendo los datos futuros.
- En la práctica este fundamento no se cumple del todo, y la calidad de los resultados empeora a medida que se distancian las distribuciones.

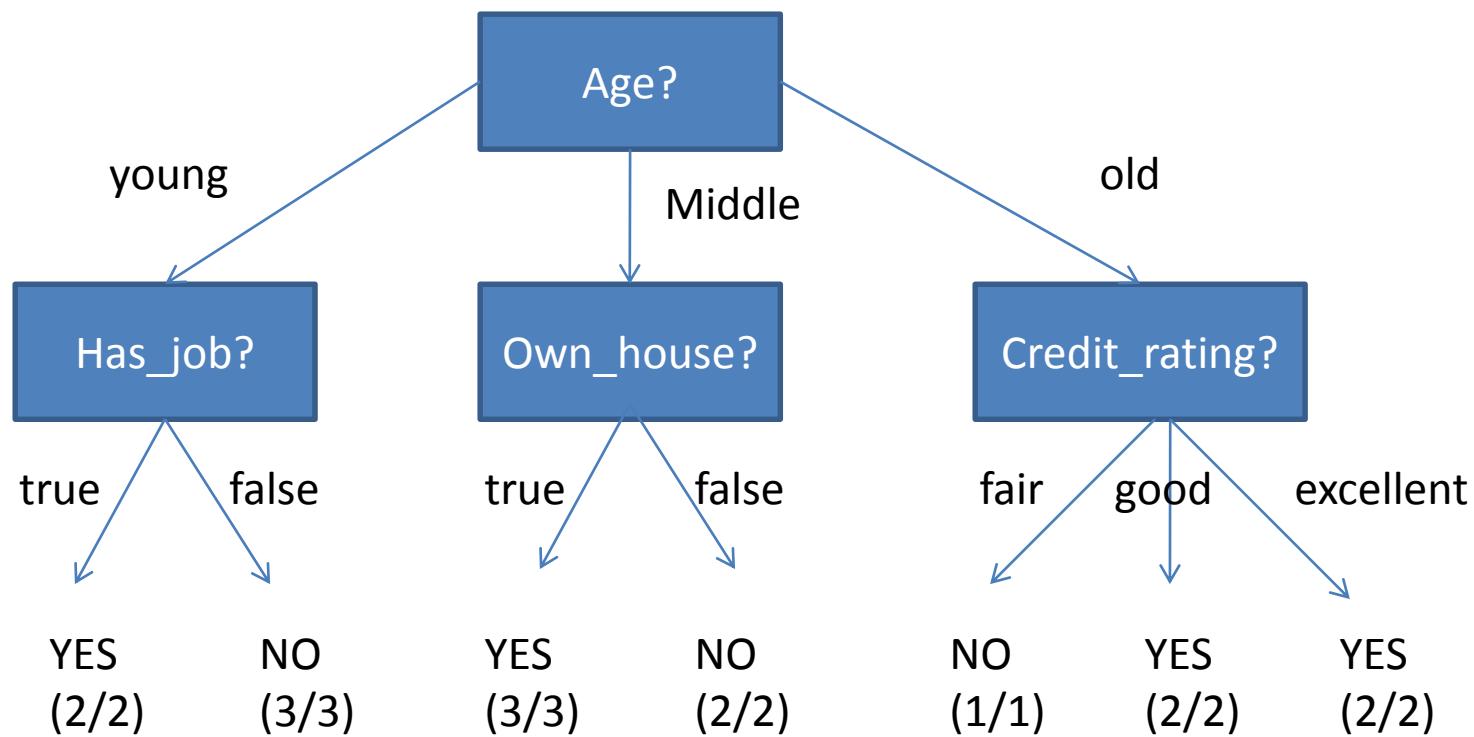
# Conceptos Básicos





# Inducción: Árboles de Decisión

- Árboles de decisión (decision trees) es una de las técnicas más usadas para clasificación.



# Inducción: Árboles de Decisión

- Un árbol de decisión tiene dos tipos de nodos: decisión y hojas. Los primeros son internos y especifican una pregunta respecto a un atributo, los segundos indican una clase.
- $(x/y)$  indica que x de y ejemplos que llegan a esa hoja tienen la clase que esta indica.
- Para usar el árbol de decisión se atraviesa de manera top-down de acuerdo a los valores de los atributos del caso de prueba.

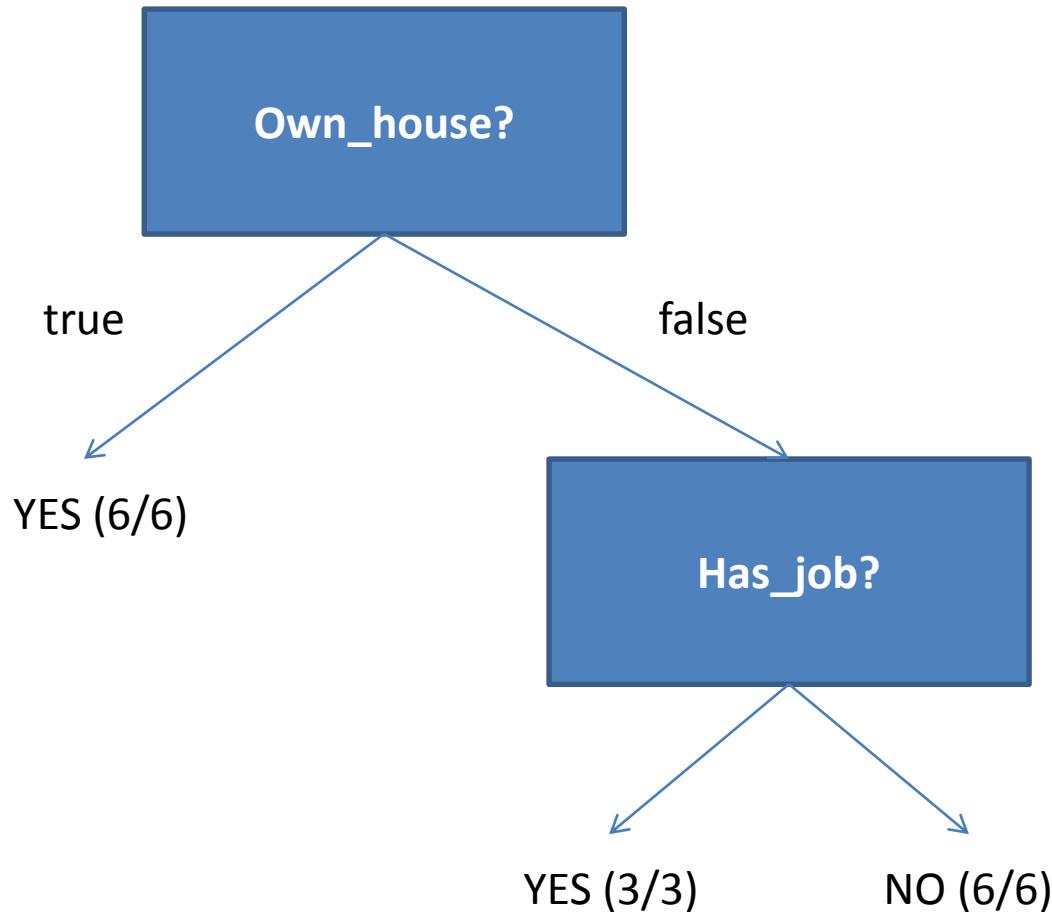
# Inducción: Árboles de Decisión

- Supongamos,
  - Age: young
  - Has\_job: false
  - Own\_house: false
  - Credit-rating: good
  - Class: ?
- La respuesta es: No.

# Inducción: Árboles de Decisión

- Un árbol de decisión es construido mediante la partición de los datos de tal forma que los subconjuntos resultantes son lo más puro posible.
- Un conjunto puro contiene ejemplos de una sola etiqueta.
- ¿Hay un único árbol?

# Inducción: Árboles de Decisión



# Inducción: Árboles de Decisión

- En la práctica, uno busca un árbol pequeño pero preciso.
- En general, árboles pequeños tienden a tener una mejor performance y son entendibles por los usuarios.
- En el ejemplo, vemos que los ratios  $x/y$  son todos 1. Pero en la vida real, esto no es siempre así. En realidad, este ratio es un valor de confiabilidad.

# Inducción: Árboles de Decisión

- Nótese que existe la posibilidad de que un árbol de decisión se pueda transformar en un conjunto de reglas if-then.
- Cada ruta desde la raíz a la hoja es una regla.
- Todos los nodos de decisión a lo largo del camino forman las condiciones de la regla y el nodo hoja la clase.
- Para cada regla se asocia un valor de confiabilidad y el número de ejemplos de soporte.

# Inducción: Árboles de Decisión

- En el último ejemplo tenemos tres reglas:
  - Own\_house = true -> Class = YES
    - Soporte = 6/15, confianza = 6/6
  - Own\_house=false, Has\_job = true -> Class = Yes
    - Soporte = 3/15, confianza = 3/3
  - Own\_house=true, Has\_job=false -> Class = No
    - Soporte = 6/15, confianza = 6/6
- En el ejemplo anterior, tenemos:
  - Age= young, Has\_job = false -> Class = No
    - Soporte = 3/15, confianza = 3/3

# Inducción: Árboles de Decisión

- Con los árboles de decisión encontramos sólo un subconjunto de las reglas que están “ocultas” en los datos.
- Un conjunto de reglas suficientes para la clasificación.
- Hay otra tarea que se llama “*minería de reglas de asociación*” que intenta buscar todas las reglas que están en cierto intervalo de confiabilidad.

# Inducción: Árboles de Decisión

- Una propiedad importante de los árboles de decisión es que el conjunto de reglas resultante es mutuamente exclusivo y exhaustivo.
- Quiere decir que cada caso está cubierto por sólo una regla, es decir, que satisface las condiciones de la regla.
- También se dice que es una representación más compacta de los datos.

# Inducción: Árboles de Decisión

- Construir el mejor árbol en precisión y tamaño es un problema NP-completo.
- Muchas heurísticas existen. Es posibles utilizar algoritmos genéticos, ACS, BCO, PSO, etc.

# Inducción: Árboles de Decisión

- El algoritmo de aprendizaje divide los datos de entrenamiento recursivamente.
- Al comienzo están todos en la raíz, a medida que el árbol crece los ejemplos son divididos recursivamente.
- El algoritmo para cuando todos los ejemplos tienen la misma etiqueta, o cuando, se han utilizado todos los atributos en una ruta.

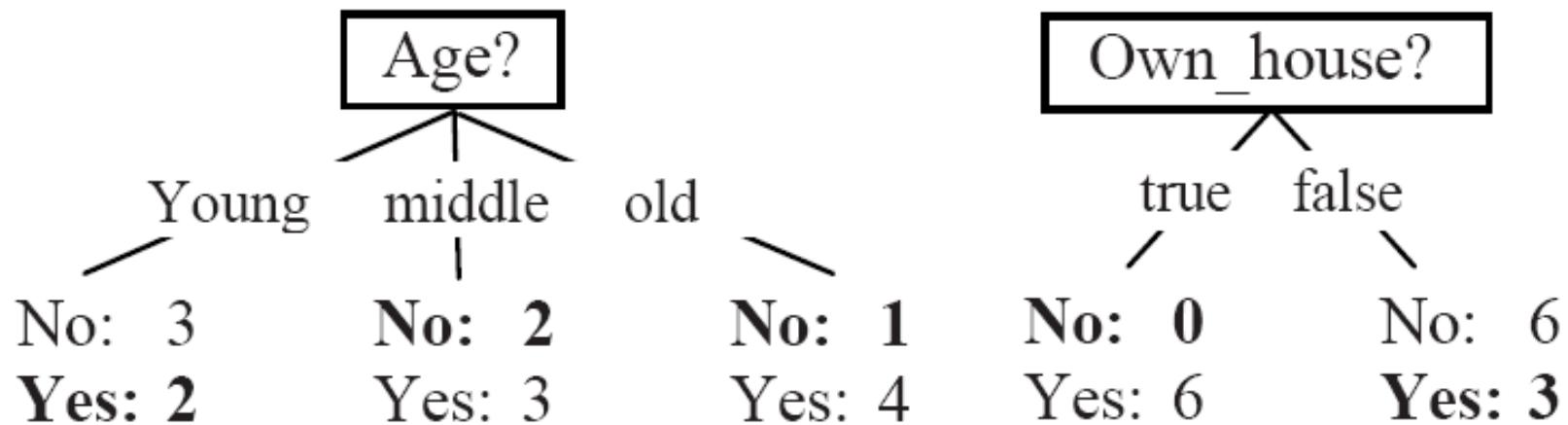
# Inducción: Árboles de Decisión

- En cada paso de la recursión utiliza el mejor atributo para particionar los datos y lo hace de acuerdo a los valores del atributo.
- El mejor atributo es seleccionado en tandem con pureza después del particionamiento.
- Lo clave de los árboles de decisión son las funciones de pureza (impurity functions).
- Es un algoritmo greedy sin backtracking: una vez que se crea un nodo no será “revisado”.



# Inducción: Árboles de Decisión

- Impurity Function



# Inducción: Árboles de Decisión

- En cada rama están el número de ejemplos de cada clase que llega ahí.
- En el árbol B, primero revisamos si Own\_house?, si eso es verdadero lo etiquetamos como “YES” (tenemos seis aciertos), si es falso tomamos la clase más grande que es “No”, lo cual conduce a tres errores.
- En el árbol A, si tomamos la clase más grande para cada rama, el escenario es aún peor, tenemos cinco errores.

# Inducción: Árboles de Decisión

- Entonces, decimos que la “impureza” del árbol A es mayor que la del B.
- En general, uno tendería a decir que Own\_house es una mejor alternativa para comenzar el árbol.
- Hay mejores formas de hacerlo. En especial, dos funciones de impureza: information gain y information gain ratio.



# Entropía

- $\Pr(c_j)$  es la probabilidad de la clase  $c_j$  en el data set D. Lo que se define como el número de ejemplos de clase  $c_j$  en D dividido por el número total de ejemplos.
- La suma de los  $\Pr(c_j)$  es, entonces, uno.
- Se define  $0\log_0 = 0$  y la unidad es bit.

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

# Entropía

- Si ambas clases tienen 50% de los datos: -  
 $0.5 * \log_2 0.5 - 0.5 * \log_2 0.5 = 1.$
- Si una clase tiene el 80% y la otra el 20%: -  
 $0.2 * \log_2 0.2 - 0.8 * \log_2 0.8 = 0.722.$
- Si una clase tiene el 100% de los datos:  $-0 * \log_2 0 - 1 * \log_2 1 = 0.$
- Cuando la entropía se torna más pequeña, el conjunto de datos es más puro.
- El máximo es con 50%, y necesita 1 bit.

# Information Gain

- La idea es primero calcular la entropía del conjunto de datos enteros, es decir,  $\text{entropy}(D)$ .
- Ahora, hay que buscar el atributo que reduce más la impureza si se utiliza para particionar los datos.
- Supongamos que  $v$  es el número posibles que puede tomar el atributo  $A_i$ .

# Information Gain

- Si vamos a utilizar  $A_i$  para particionar  $D$ , tenemos que dividir  $D$  en  $v$  conjuntos disjuntos  $D_1, D_2, \dots, D_v$ . La entropía después la partición es:

$$entropy_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} * entropy(D_j)$$

$$gain(A_i) = entropy(D) - entropy_{A_i}(D)$$

# Information Gain

- Se escoge el que reduce más la entropía.
- Normalmente, se utiliza una cota inferior, para las reducciones muy pequeñas. Ahí es donde el algoritmo para.
- El algoritmo es recursivo.
- Para los datos de ejemplo, tenemos 6 y 9 “NO” y “YES” casos.

$$\text{entropy}(D) = -\frac{6}{15} * \log_2 \frac{6}{15} - \frac{9}{15} * \log_2 \frac{9}{15} = 0.971$$

# Information Gain

- El atributo Age tiene tres subconjuntos (young, middle and old). Cada subconjunto tiene cinco ejemplos.
- El atributo Own\_house parte los datos en dos grupos (true and false).

$$\begin{aligned} \text{entropy}_{age}(D) &= \frac{5}{15} \text{entropy}(D_1) + \frac{5}{15} \text{entropy}(D_2) + \frac{5}{15} \text{entropy}(D_3) \\ &= \frac{5}{15} 0.971 + \frac{5}{15} 0.971 + \frac{5}{15} 0.722 = 0.888 \end{aligned}$$

# Information Gain

- En detalle el calculo de las entropías de acuerdo al valor de los atributos:
  - Entropy ( $D_1$ ) = $-1*(2/5*LOG(2/5;2) + 3/5*LOG(3/5;2)) = 0,97095059$  [3 No y 2 Sí]
  - Entropy ( $D_2$ ) = $-1*(3/5*LOG(3/5;2) + 2/5*LOG(2/5;2)) = 0,97095059$  [2 No y 3 Sí]
  - Entropy ( $D_3$ ) = $-1*(1/5*LOG(1/5;2) + 4/5*LOG(4/5;2)) = 0,72192809$  [1 No y 4 Sí]

# Information Gain

$$entropy_{Own\_house}(D) = \frac{6}{15} entropy(D_1)$$

$$+ \frac{9}{15} entropy(D_2)$$

$$= \frac{6}{15} 0 + \frac{9}{15} 0.918 = 0.551.$$

$$entropy(D_1) = -\left(\frac{0}{6} * \log_2 \frac{0}{6} + \frac{6}{6} * \log_2 \frac{6}{6}\right) = 0$$

$$entropy(D_2) = -\left(\frac{3}{9} * \log_2 \frac{3}{9} + \frac{6}{9} * \log_2 \frac{6}{9}\right) = 0,91829583$$

D<sub>1</sub> tiene sólo seis “si”, y D<sub>2</sub> tiene seis “no” y tres “si”.

# Information Gain

- Asumiendo que  $entropy_{Has\_job}(D)=0.647$  y  $entropy_{Credit\_rating}(D)=0.608$ . Los valores de las ganancias están dado por:
  - $gain(D, Age) = 0.971 - 0.888 = 0.083$
  - **gain(D, Own\_house) = 0.971 - 0.551 = 0.420** 
  - $gain(D, Has\_job) = 0.971 - 0.647 = 0.324$
  - $gain(D, Credit\_rating) = 0.971 - 0.608 = 0.363$
- Como  $Own\_house = true$  tiene sólo una clase, resulta en una hoja. En el caso falso, se hace el mismo procedimiento pero tomando en cuenta  $D_2$ .

# Information Gain

- El problema de information gain es que pueden haber atributos como las Ids, donde hay una única por cada ejemplo.
- En este caso se particionaría los datos en una clase por ejemplo, ya que daría  $entropy_{ID}(D)=0$ .
- En este caso es una métrica inútil.
- Nótese que este es un caso extremo, pero una situación similar se da con atributos con muchos valores potenciales.

# Gain Ratio

- Esta métrica intenta solucionar este sesgo mediante la normalización de las ganancias previamente vistas, utilizando la entropía de los datos con respecto a los valores de cada atributo.

$$gainRatio(D, A_i) = \frac{gain(D, A_i)}{- \sum_{j=1}^s \frac{|D_j|}{|D|} \log_2 \frac{|D_j|}{|D|}}$$

# Gain Ratio

- Donde  $s$  es el número de valores posibles de  $A_i$ , y  $D_j$  es el subconjunto de los datos que tienen el  $j$ -ésimo valor de  $A_i$ .
- Si  $A_i$  tiene muchos valores, entonces el denominador va a ser grande.
- Si el denominador es muy pequeño, se pueden utilizar heurísticas para parcharlo.

# Propiedades de los Árboles de Decisión

Es uno de los métodos más utilizados debido a:

- La habilidad de lidiar con datos irrelevantes.
- Pueden lidiar con atributos de distintos tipos de manera natural: numéricos, binarios, y categóricos. A los contiguos hay que aplicarles un pre-procesamiento.
- Son muy rápidos de construir y ejecutar comparado con otras técnicas del mismo tipo.
- Son tolerantes a los valores perdidos, ya que son tratados como un valor separado.
- Tiene poco parámetros que deben ajustarse.
- El modelo tiene una representación interpretable.

# Propiedades de los Árboles de Decisión

## Algunas limitaciones:

- Es difícil encontrar la línea de tendencia en los datos (margen separador). Dado que cada split o nivel del árbol es una partición, para encontrar la tendencia de los márgenes se necesitan muchos niveles de división.
- La fragmentación de los datos. Cada nivel del árbol reduce el set de entrenamiento para los niveles inferiores. Esto es problemático cuando hay muchos atributos, ya que los vectores son “sparse” y es fácil que el modelo haga overfitting.
- El proceso de búsqueda del árbol puede caer en un óptimo local, por ende tener un gran sesgo, y pequeños cambios en los datos de entrenamiento pueden conducir a grandes cambios en el árbol final.
- Si hay errores en los niveles altos del árbol, éstos se propagan a los niveles inferiores.
- No es bueno cuando la función discriminante que se quiere aprender tiene dependencia en muchas variables, ya que los árboles resultantes son muy profundos.

# Atributos Continuos

- Para aplicar los árboles de decisión a datos continuos, los valores del atributo  $A_i$  pueden dividirse en intervalos.
- De ahí se puede considerar como el caso discreto.
- ¿Cómo segmentar los intervalos?
- En la práctica, dos intervalos es suficiente, pero hay que encontrar la cota.

# Atributos Continuos

- La idea es escoger una cota que maximice la ganancia (o el gain ratio).
- Aunque en teoría debería investigarse todos los valores posibles, que son infinitos, en la práctica, los valores que aparecen en los datos son finitos.
- Sin pérdida de generalidad podemos decir que si un atributo  $A_i$  tiene los valores  $\{v_1, v_2, \dots, v_r\}$  en los datos, cualquier cota entre  $v_i$  y  $v_{i+1}$  va a dar igual. (asumiendo orden ascendente)

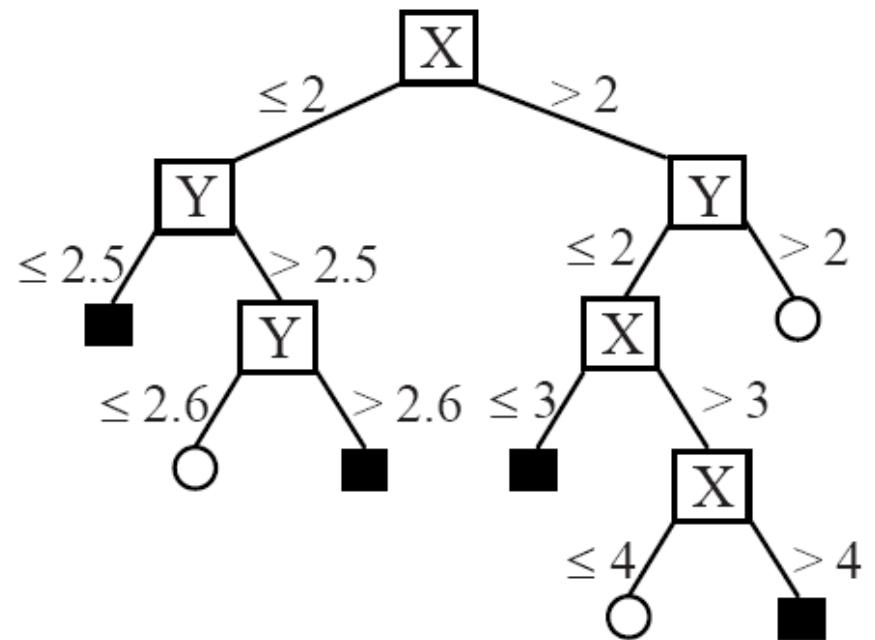
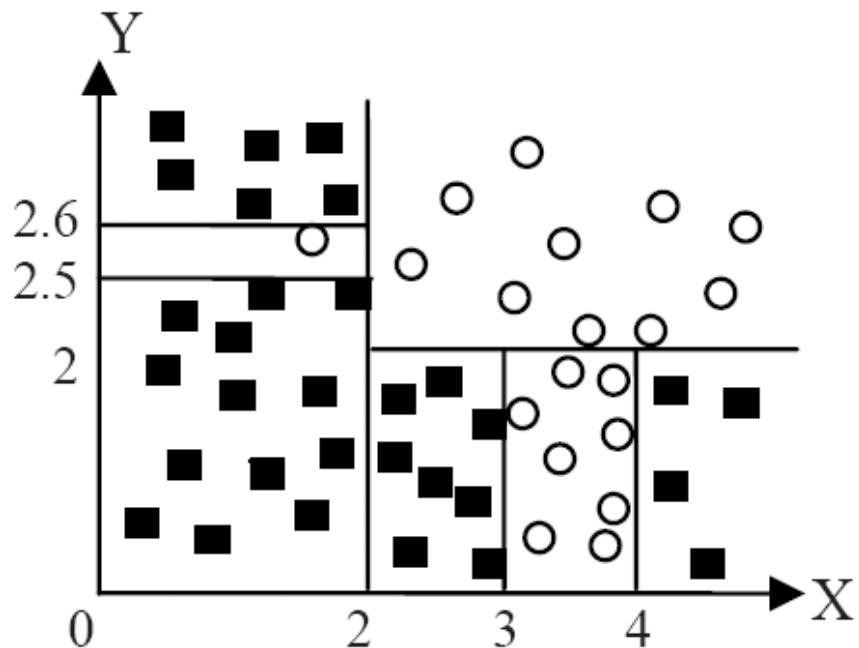
# Atributos Continuos

- En la práctica, hay  $r-1$  “splits” posibles.
- La cota puede ser escogida como el punto medio, o bien, al lado derecho de  $v_i$ .
- En el ultimo caso, resultan los intervalos  $A_i \leq v_i$  y  $A_i > v_i$ .
- Se escoge el valor que maximice la ganancia, y escoger el valor del lado derecho tiene la ventaja que los valores realmente aparecen en los datos.

# Atributos Continuos

- Desde un punto de vista geométrico, un árbol de decisión construido sólo a base de atributos continuos representa una partición del espacio generado por los datos.
- Una serie de “splits” que comienzan desde la raíz hasta un nodo hijo representa un hiperrectángulo en un hiperplano paralelo al eje.

# Atributos Continuos



# Atributos Continuos

- Con sólo atributos discretos, el algoritmo crece linealmente con respecto al tamaño del set de entrenamiento.
- En cambio, ordenar los datos continuos toma  $|D| \log |D|$ , lo que comienza a dominar el proceso de aprendizaje.
- El ordenamiento es importante ya que asegura que las ganancias pueden ser calculadas en una pasada por los datos.

# Overfitting

- Un árbol de decisión particiona recursivamente los datos hasta que no hay impurezas o no quedan atributos.
- Esto puede dar lugar a árboles muy profundos donde las hojas cubren muy pocos ejemplos.
- Si utilizamos esos árboles para predecir los datos de entrenamiento, tendremos muy buenos resultados, pero no en datos no vistos.

# Overfitting

- Eso indica que el aprendizaje no fue efectivo, y que los modelos no generalizan los datos de buena forma.
- Este fenómeno se llama **overfitting**.
- En general, se dice que existe overfitting cuando existe un clasificador  $c_1$  que obtiene una mejor performance que otro  $c_2$  en los datos de entrenamiento, pero en los datos de prueba pasa lo contrario. 

# Overfitting

- Es producto, muchas veces, de ruido en los datos:
  - Mal etiquetamiento de los datos.
  - Valores equivocados en los atributos.
  - El dominio es complejo y tiene un alto grado de “aleatoriedad”.
- Estos factores hacen que el árbol de decisión sea profundo, es decir, que use muchos atributos. 

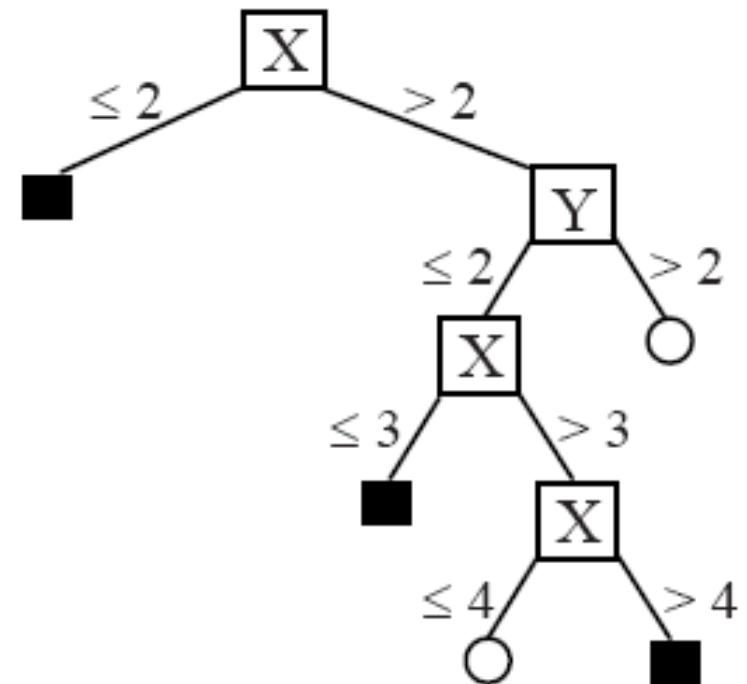
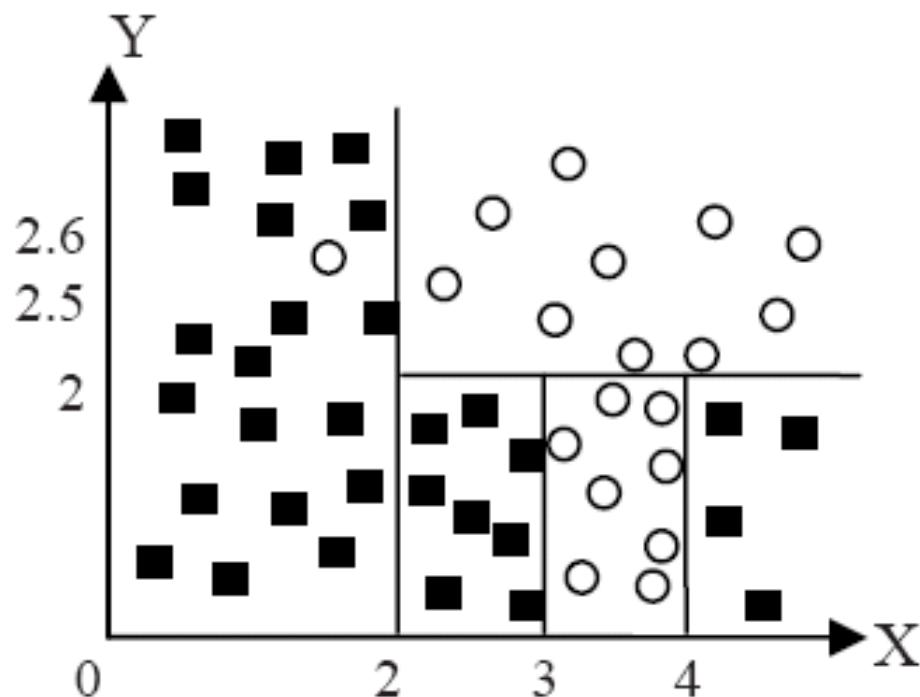
# Overfitting

- Para solucionar este problema se eliminan algunas ramas o sub-árboles, y se reemplazan con hojas de la mayor clase.
- Existe el pre-pruning y el post-pruning: el primero detiene la exploración del árbol tempranamente, y el segundo después que está construido.
- Post-pruning ha demostrado ser más eficiente en la práctica. Pre-pruning no sabe lo que pasaría si el árbol se sigue extendiendo.

# Post-pruning

- La idea general es estimar el error de cada nodo en el árbol.
- Si el error estimado para un nodo es menor que el de su sub-árbol, entonces ese sub-árbol es cortado.
- Otra técnica consiste en utilizar un **conjunto de validación**, el cual no es utilizado ni para entrenar, ni para evaluar. Este conjunto es clasificado, permitiendo encontrar errores y después podado.

# Post-pruning



# Rule Pruning

- La idea es convertir el árbol en un set de reglas.
- El objetivo es eliminar algunas condiciones que hacen las reglas más cortas.
- Las que quedan redundantes ahora se eliminan.
- Por lo general, este nuevo set es mejor, ya que tiene menos chances de hacer overfitting a los datos.

# Rule Pruning

- Pruning también es conocido como generalización ya que hace las reglas más generales.
- Una regla con más condiciones es más específica que una con menos condiciones.
- En el ejemplo, en el sub-árbol  $X \leq 2$  produce las siguientes reglas:
  - $X \leq 2, Y > 2.5, Y > 2.6 \rightarrow 1$
  - $X \leq 2, Y > 2.5, Y \leq 2.6 \rightarrow 0$
  - $X \leq 2, Y > 2.5, Y > 2.6 \rightarrow 1$

# Rule Pruning

- Las reglas después de la poda, no necesariamente se excluyen mutuamente, y tampoco son exhaustivas.
- Es decir, van a haber casos que van a satisfacer muchas reglas.
- Por eso se establece algún ordenamiento.
- También puede suceder que ninguna regla se satisfaga, para esto normalmente se aplica la clase mayoritaria.

# Atributos de Valores “perdidos”

- En aplicaciones del mundo real, valores nuevos pueden aparecer.
  - Utilizar un valor por defecto “unknown”.
  - Utilizar el valor más frecuente.
- En caso de valores continuos, se puede utilizar la media.

# Clases desbalanceadas

- Los árboles de decisión no se comportan bien cuando uno tiene muchas más instancias de una clase que de otra.
  - Clases desbalanceadas pueden generar un sesgo, ya que las probabilidades se distorsionan.
  - ¿Misma distribución en los datos de prueba? ¿En nuevos datos? 
- Se puede hacer oversampling/undersampling de las clases menores.
  - La idea es hacer ajustes de acuerdo a cuan probable van a ser las clases menores.

# Oversampling vs. Undersampling

- Ambas buscan ajustar el ratio entre las diferentes categorías en un conjunto de datos.
  - Puede ser para balancearlos o para que se ajuste a la realidad.
  - La idea es corregir el sesgo en el entrenamiento. Ya que normalmente viene de fuentes que no representan fielmente el problema.
- Por ejemplo, podríamos tener 1,000 ejemplos de personas, donde 66% son hombres sacadas de alguna colección específica (deportes). Por otro lado, sabemos que el ratio entre hombres y mujeres es cercano al 50%.
  - **Oversampling** copiaría cada dato de una mujer dos veces, así alcanzaría 667 ejemplos, igualando el número de datos provenientes de hombres. 
  - **Undersampling** descartaría aleatoriamente de los 667 ejemplos de hombres de manera que cada uno quedaría con 333. 

# Siempre Recordar .....

“Lo visto en la fase de entrenamiento nunca debe ser utilizado para probar/evaluar un sistema”

# Evaluación de Clasificadores

- Métricas:
  - **Accuracy:** El ratio de casos correctamente clasificados versus el número total de casos de prueba. 
  - **Error rate:** es  $1 - \text{accuracy}$ .
- Evaluación:
  - **Holdout set:** el set de datos es dividido en dos sub-conjuntos disjuntos uno de entrenamiento y el otro de evaluación.

# Evaluación

- El conjunto de prueba también es llamado **holdout set**. Este tipo de evaluación se utiliza cuando el conjunto de datos es grande y además, cuando está etiquetado.
- Es importante que los datos de prueba no sean utilizados para entrenar.
- Los porcentajes varían de acuerdo al número de datos: 50%-50%, 33%-66%, 20%-80%, etc.

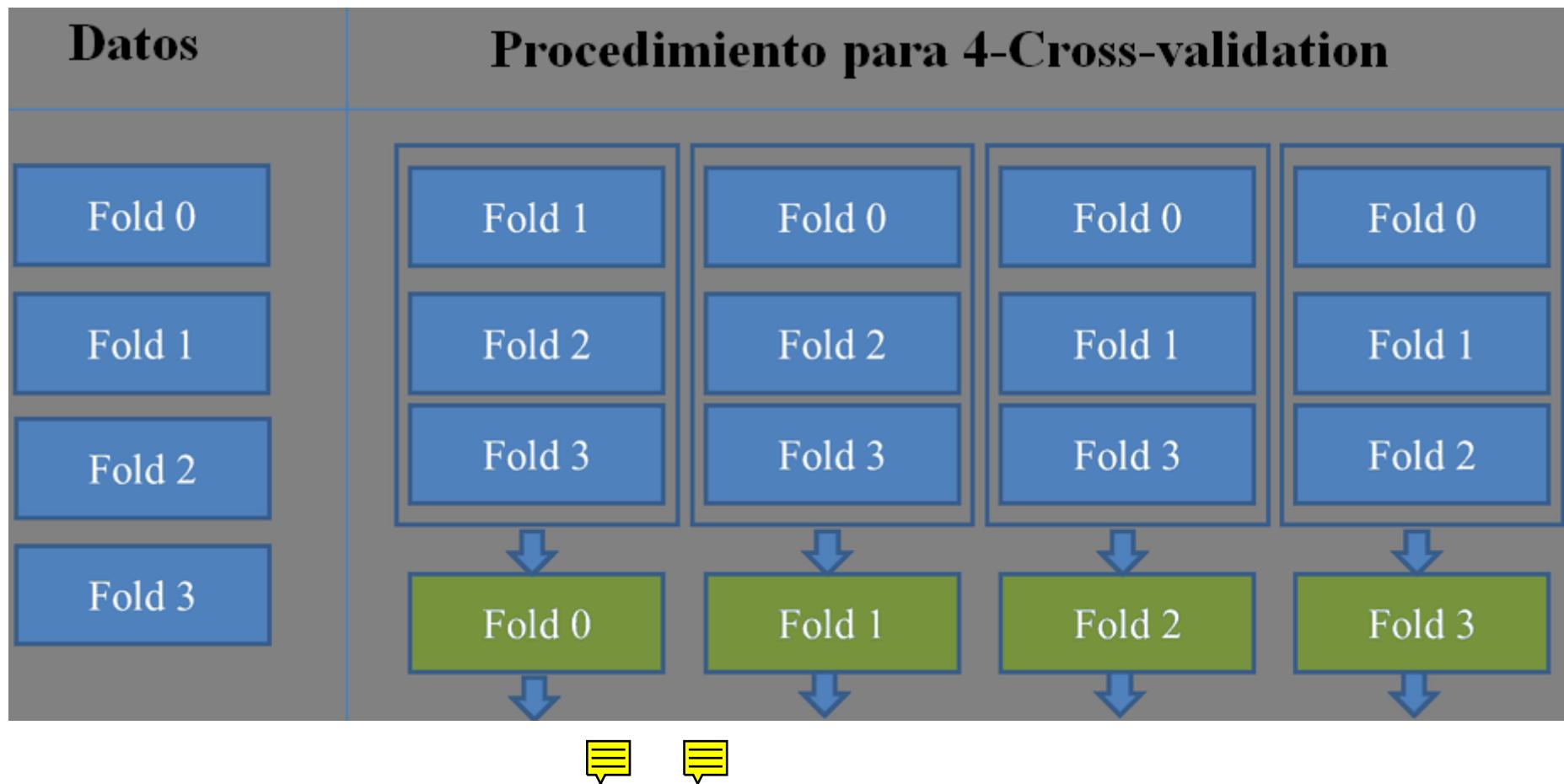
# Evaluación

- Para particionar los datos, se puede:
  - Utilizar random.
  - Lo que es mejor es dejar la data más reciente para probar y la más antigua para entrenar. Normalmente, esta forma refleja mejor la dinámica de la aplicación ya que tendrá que etiquetar datos futuros.
- **Multiple random sampling:** sirve cuando hay pocos datos y consiste en hacer holdout con random sampling muchas veces y calcular una accuracy promedio.

# Evaluación

- **Cross-validation:** se utiliza cuando el conjunto de datos es pequeño. Es el método más común. Los datos se dividen en  $n$  conjuntos disjuntos de igual tamaño. 
- Entonces,  $n-1$  conjuntos se utilizan para entrenar y el restante para probar.
- Este procedimiento se ejecuta  $n$  veces con cada  $n$  conjunto en modo prueba.
- La accuracy final es el “promedio” de las  $n$  accuracies , o el overall calculado sobre los  $n$  conjuntos.
  - 5-fold y 10-fold validations son las más usadas.

# Ejemplo: Cross-Validation



# Evaluación

- Un caso especial de **cross-validation** es el **leave-one out cross-validation (LOOCV)**:
  - Aquí se deja sólo un ejemplo de prueba y todo el resto de entrenamiento.
  - Sólo sirve para conjuntos de datos pequeños.
  - Es decir, es  $m$ -fold cross validations, donde  $m$  es el número de datos.
  - Si el conjunto de datos es muy grande, se hace inviable efectuar este proceso iterativo.

# Evaluación

- Por convención, la clase que uno le interesa identificar es la positiva, y todo el resto se le llama negativo.
- Accuracy tiene el problema de que si una de las clases es pequeña, bastaría un sistema que etiquetara todos los ejemplos como la clase mayoritaria y alcanzaría una valor alto en accuracy.
  - Si hay más de una clase, no diferencia la magnitud del error, es decir, en qué posición quedo del ranking.
  - A ésto se le llama “majority class baseline”.

# Evaluación

- Accuracy es una métrica buena cuando tenemos problemas de dos clases.
- Pero, ¿Qué problemas tenemos cuando aumentamos a más clases?
  - Diferencia si la etiqueta otorgada fue correcta o no, sin embargo, no te dice cuán cerca estuvo de ser elegido.
  - Es decir, si quedo en 2do o en último lugar da lo mismo.

# Evaluación

- Supongamos que tenemos un problema multi-clase, es decir que en vez de tener +/-, tenemos más clases.
  - Por ejemplo, deportes, música y ciencia.
- Un clasificador  $c_1$  etiqueta un ejemplo de música X en el siguiente orden: deportes, música y ciencia.
- Otro clasificador  $c_2$ , lo hace en el siguiente orden: deportes, ciencia y música.
- ¿Cuál es la accuracy de ambos clasificadores?

# Matriz de confusión

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Negativo Positivo

- TP = clasificación correcta de ejemplos positivos (True positive)
  - FN = clasificación incorrecta de ejemplos positivos (False negative)
  - FP = clasificación incorrecta de ejemplos negativos (False positive)
  - TN = clasificación correcta de ejemplos negativos (True negative)

La precisión y el recall con respecto a la clase positiva:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

# $F_1$ -score

- Es difícil comprar clasificadores basados en estas dos métricas que pueden dar resultados muy dispares.
- Por ejemplo,  $TP = 1$ ,  $TN= 1000$ ,  $FN=0$ , y  $FP=99$ :
  - El recall es 1% y la precisión 100%, porque clasificamos sólo ejemplo positivo correctamente, y no se clasificó ningún ejemplo negativo erróneamente.
- Normalmente hay un trade-off entre precision y recall.

# $F_1$ -score

- $F_1$ -score es la media harmónica entre precision y recall.
- La media harmónica tiene a estar cerca del menor de los dos valores. 
- Por ende, para ser alta, esta métrica debe contar con una precisión y un recall altos.
- Hay otra métrica: **precision and recall  
break-even point.**

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

# F<sub>1</sub>-score

- En palabras: “*mide la efectividad de la recuperación de información con respecto a un usuario que le da veces más importancia al recall que a la precisión*”.

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

# F<sub>1</sub>-score

$\beta$	Precision	Recall	$F(\beta)$	Precision	Recall	$F(\beta)$	Precision	Recall	$F(\beta)$
1	0,3	0,6	0,4	0,4	0,6	0,48	0,5	0,6	0,5455
2	0,3	0,6	0,5	0,4	0,6	0,545	0,5	0,6	0,5769
3	0,3	0,6	0,545	0,4	0,6	0,571	0,5	0,6	0,5882
4	0,3	0,6	0,567	0,4	0,6	0,583	0,5	0,6	0,593
5	0,3	0,6	0,578	0,4	0,6	0,589	0,5	0,6	0,5954
6	0,3	0,6	0,584	0,4	0,6	0,592	0,5	0,6	0,5968
7	0,3	0,6	0,588	0,4	0,6	0,594	0,5	0,6	0,5976
8	0,3	0,6	0,591	0,4	0,6	0,595	0,5	0,6	0,5982
9	0,3	0,6	0,593	0,4	0,6	0,596	0,5	0,6	0,5985
10	0,3	0,6	0,594	0,4	0,6	0,597	0,5	0,6	0,5988

$\beta$	Precision	Recall	$F(\beta)$	Precision	Recall	$F(\beta)$	Precision	Recall	$F(\beta)$
1	0,6	0,3	0,4	0,6	0,4	0,48	0,6	0,5	0,5455
2	0,6	0,3	0,333	0,6	0,4	0,429	0,6	0,5	0,5172
3	0,6	0,3	0,316	0,6	0,4	0,414	0,6	0,5	0,5085
4	0,6	0,3	0,309	0,6	0,4	0,408	0,6	0,5	0,505
5	0,6	0,3	0,306	0,6	0,4	0,405	0,6	0,5	0,5032
6	0,6	0,3	0,304	0,6	0,4	0,404	0,6	0,5	0,5023
7	0,6	0,3	0,303	0,6	0,4	0,403	0,6	0,5	0,5017
8	0,6	0,3	0,302	0,6	0,4	0,402	0,6	0,5	0,5013
9	0,6	0,3	0,302	0,6	0,4	0,402	0,6	0,5	0,501
10	0,6	0,3	0,301	0,6	0,4	0,401	0,6	0,5	0,5008

# Breakeven Point

- Es el punto donde la precisión y el recall son iguales.
- Asume que los casos de prueba pueden ser jerarquizados por el clasificador de acuerdo a su probabilidad de que sean positivos.
- En el caso de un árbol de decisión podemos utilizar los valores de confiabilidad.

# Breakeven Point

- Supongamos que tenemos 20 documentos, y 1 representa el ranking más alto y 20 el menor. Además, “+” (“-”) denotan un documento positivo (negativo).
- Asumamos que el conjunto tiene 10 ejemplos positivos.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
+	+	+	-	+	-	+	-	+	+	-	-	+	-	-	-	+	-	-	+

# Breakeven Point

- **Pos 1:**  $p = 1/1= 100\%$ ,  $r=1/10=10\%$
- **Pos 2:**  $p = 2/2= 100\%$ ,  $r=2/10=20\%$
- ....
- **Pos 9:**  $p = 6/9= 66.7\%$ ,  $r=6/10=60\%$
- **Pos 10:**  $p = 7/10= 70\%$ ,  $r=7/10=70\%$
- El breakeven point es 70%. Si el punto no se encuentra directamente, hay que interpolar.

# Receiver Operating Characteristic Curve

- Es la gráfica del ratio de los verdaderos positivos versus el ratio de los falsos negativos.
  - Gráfica bi-dimensional.
- Se utilizan en la clasificación de dos clases.
  - Grafica el trade-off entre los beneficios (TP) y los costos (FP). 
- El clasificador necesita rankear los ejemplos de prueba de acuerdo a su probabilidad de pertenecer a la clase, dejando el más probable en el lugar más alto.

# Receiver Operating Characteristic Curve

- El TPR (true positive rate) es definido como la fracción de casos genuinos verdaderos que son correctamente clasificados.

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

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

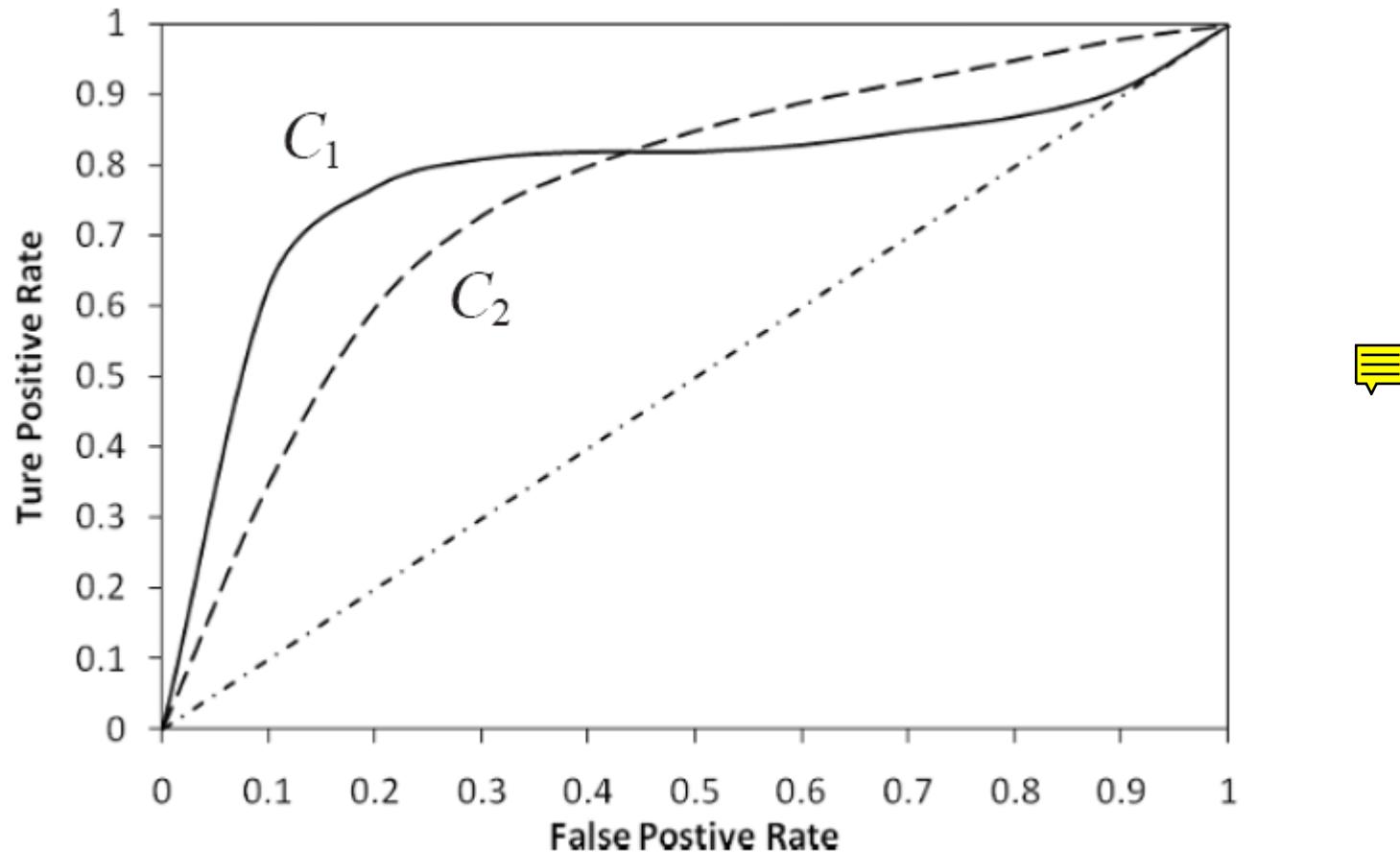
# Receiver Operating Characteristic Curve

- TPR es básicamente el recall de la clase positiva y también es llamado “sensitividad” en estadística.
- Hay otra métrica que se llama “especificidad”, la cual es el TNR, o el recall de la clase negativa:

$$TNR = \frac{TN}{TN + FP}$$

# Receiver Operating Characteristic Curve

- Se puede apreciar que  $FPR=1/\text{especificidad}$

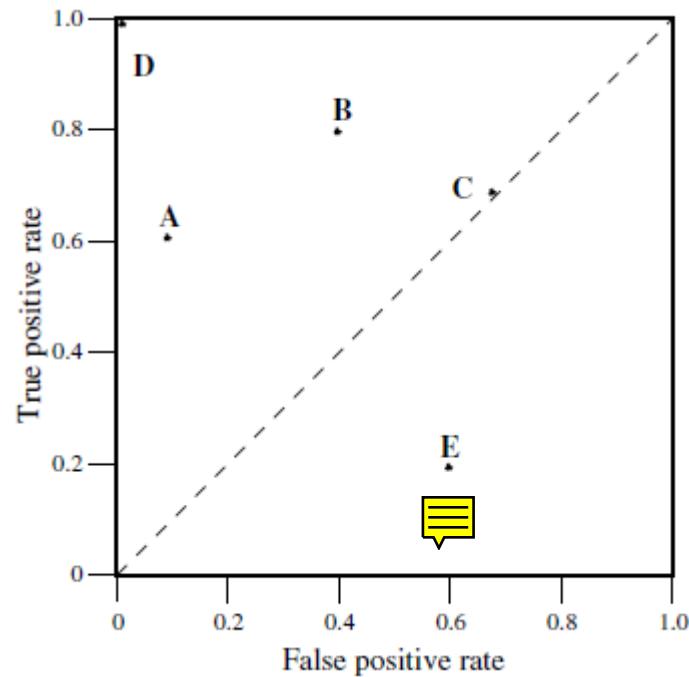


# Receiver Operating Characteristic Curve

- La curva ROC comienza en (0,0) y termina en (1,1).
- El punto (0,0) representa cuando todos los casos de prueba son etiquetados como negativo, y (1,1) positivos.
- El punto (0,1) representa la clasificación perfecta.
- La línea diagonal representa un “random guessing” con una probabilidad fija.
- En el gráfico anterior podemos ver que 0.43 es un punto de inflexión donde  $C_2$  pasa a ser mejor que  $C_1$ .

# Receiver Operating Characteristic Curve

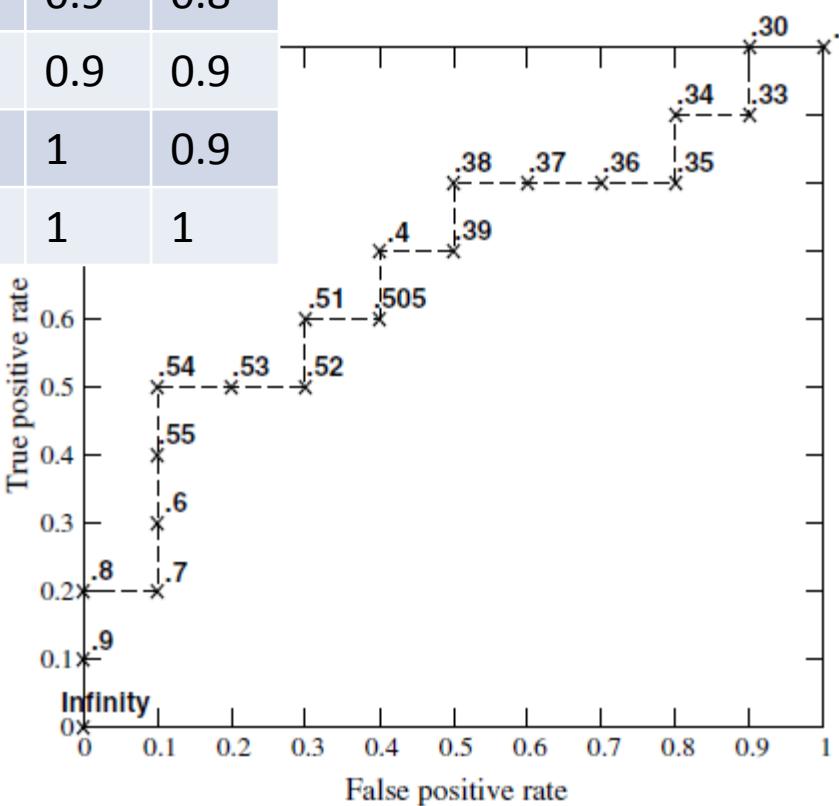
- Clasificadores cercanos al eje “TPR” son vistos como “conservadores”:
  - Etiquetan algo como positivo siempre cuando haya fuerte evidencia, ergo tienen pocos errores del tipo FPR y pocos aciertos TPR.
- Clasificadores cercanos al eje “FPR” (superior) son vistos como “liberales”:
  - Etiquetan algo como positivo bajo evidencia débil, ergo tienen altas tasas de errores del tipo FPR y altos aciertos TPR.



# Receiver Operating Characteristic Curve

- Normalmente, en las aplicaciones del mundo real, tenemos más datos negativos, ya que queremos sacar un subconjunto pequeño de datos (los que tienen la etiqueta/tipo que buscamos).
- Nótese que toda curva ROC generada por datos finitos nos va a dar una función discreta (función escalón).

#	Class	Score	TPR	FPR	#	Class	Score	TPR	FPR
1	+	0.9	0.1	0	11	+	0.4	0.7	0.4
2	+	0.8	0.2	0	12	-	0.39	0.7	0.5
3	-	0.7	0.2	0.1	13	+	0.38	0.8	0.5
4	+	0.6	0.3	0.1	14	-	0.37	0.8	0.6
5	+	0.55	0.4	0.1	15	-	0.36	0.8	0.7
6	+	0.54	0.5	0.1	16	-	0.35	0.8	0.8
7	-	0.53	0.5	0.2	17	+	0.34	0.9	0.8
8	-	0.52	0.5	0.3	18	-	0.33	0.9	0.9
9	+	0.51	0.6	0.3	19	+	0.30	1	0.9
10	-	0.505	0.6	0.4	20	-	0.1	1	1



# Receiver Operating Characteristic Curve

- La diagonal  $y=x$  representa una estimación aleatoria, porque uno esperaría que la mitad de los positivos y la mitad de los negativos estén correctamente etiquetados.
  - Esto nos da el punto  $(0.5;0.5)$  en el gráfico.
- Si asigna con 90% de probabilidad la clase positiva, es esperable que le descubra el 90% de los genuinos positivos, pero también esperable encontrar un 90% de falsos positivos.
  - Nos da el punto  $(0.9; 0.9)$
- Nos da un clasificador que se mueve en la diagonal.

# Receiver Operating Characteristic Curve

- ¿Qué pasa con los clasificadores que se mueven en la parte inferior del gráfico?
  - Son los que se desempeñan peor que un clasificador aleatorio.
  - Por eso, normalmente esa zona queda vacía.
  - Si un clasificador cae en esa zona, simplemente tenemos que invertir su salida y caerá en la zona de arriba.
    - Se cambia las etiquetas asignadas por la contraria.+
    - Es un clasificador que tiene información útil, pero la usa incorrectamente.

# Receiver Operating Characteristic Curve

- Una de las ventajas que tiene esta curva es que no es sensible a cambios en la distribución de las clases en el conjunto de datos.
  - Es decir, si la proporción de instancias negativas/positivas cambian, la curva no va a cambiar.
- Al contrario de otra métricas como accuracy, precision, y F-Score.
- Grandes sesgos son comunes en datos provenientes de aplicaciones del mundo real.

# Receiver Operating Characteristic Curve

- Esta curva directamente no nos puede dar, en general, una respuesta a qué clasificador es mejor.
- Para una comparación más general, los investigadores utilizan el área bajo la curva ROC. Esta área se denomina AUC.
- Si el área bajo la curva es mayor para un clasificador que para otro, se dice que es mejor que el otro.
- Si es perfecto tendrá un AUC de uno.
- Si hace sólo “random guesses” obtendrá 0.5.

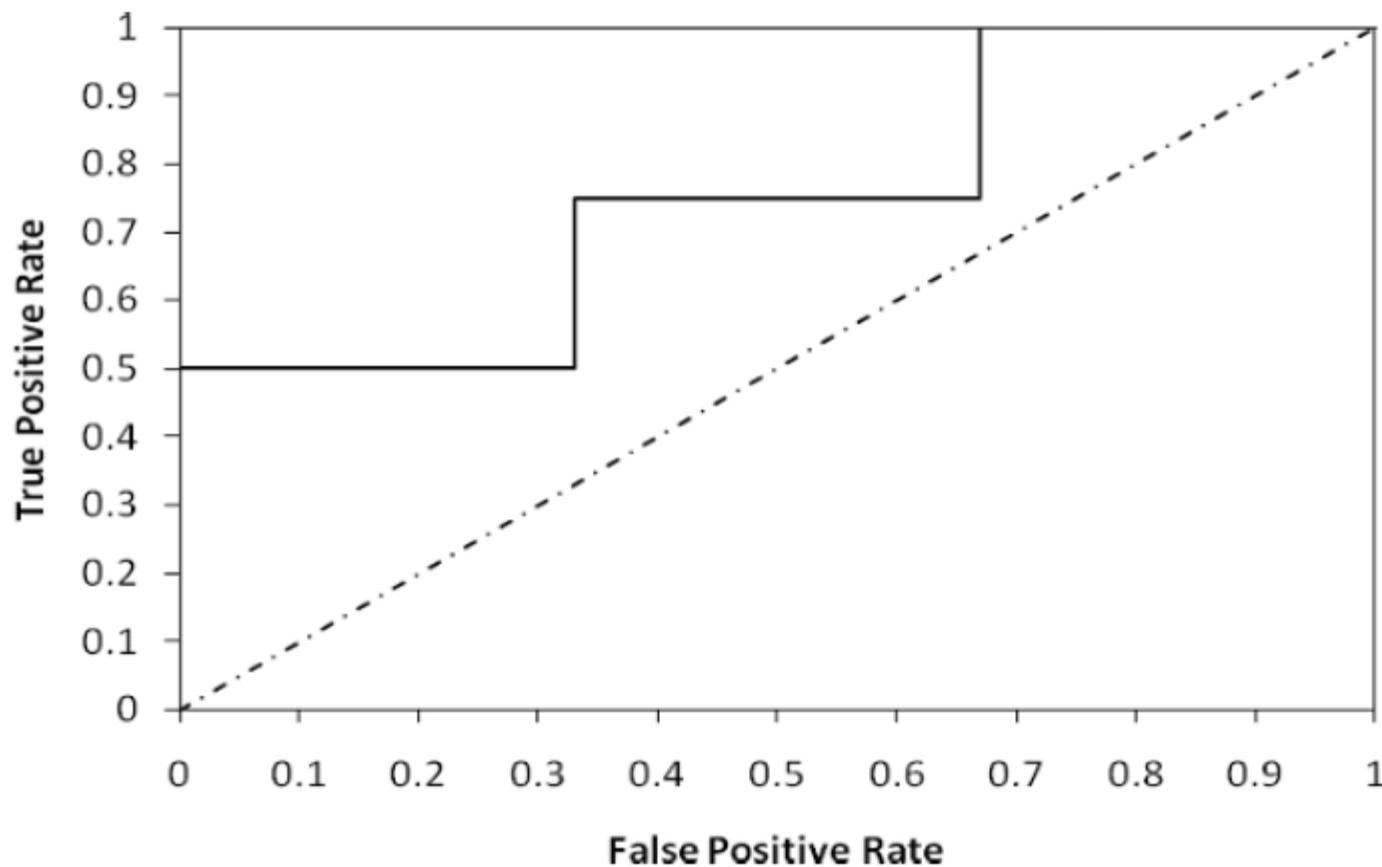
# Receiver Operating Characteristic Curve

- El ranking es obtenido vía ordenando los casos de prueba en orden decreciente de acuerdo a lo que asignó el clasificador.
- Se divide el resultado en dos partes en cada caso y se considera la parte superior positiva y la inferior negativa.
- Para cada partición se calcula el TPR y FPR.
- (0,0) y (1,1) son los puntos extremos.

# Receiver Operating Characteristic Curve

Rank		1	2	3	4	5	6	7	8	9	10
Actual class		+	+	-	-	+	-	-	+	-	-
TP	0	1	2	2	2	3	3	3	4	4	4
FP	0	0	0	1	2	2	3	4	4	5	6
TN	6	6	6	5	4	4	3	2	2	1	0
FN	4	3	2	2	2	1	1	1	0	0	0
TPR	0	0.25	0.5	0.5	0.5	0.75	0.75	0.75	1	1	1
FPR	0	0	0	0.17	0.33	0.33	0.50	0.67	0.67	0.83	1

# Receiver Operating Characteristic Curve



# Receiver Operating Characteristic Curve

- El estimador AUC tiene una propiedad estadística importante:
  - “*El AUC de un clasificador es equivalente a la probabilidad que el clasificador jerarquice un ejemplo positivo escogido aleatoriamente por sobre uno negativo escogido también aleatoriamente*”.
- Es posible que un clasificador obtenga un valor AUC muy alto, y aún así, en una zona específica de la curva ROC, tenga un desempeño mucho más malo que un clasificador con AUC muy bajo.

# Receiver Operating Characteristic Curve

- ¿Qué sucede cuando tenemos más de una clase?
  - La matriz de confusión crece significativamente, por ejemplo para tres clases tenemos 9 casillas.
  - De las cuales  $n^2-n$  son errores.
- Un método para manejar las n clases es producir n gráficos ROC diferentes.
  - Cada uno considera una clase como positiva, y todo el resto como negativo.
  - Sin embargo, este análisis se hace sensible a “class skew” (clases desbalanceadas).
  - A pesar de esto tiende a funcionar bien en la práctica.

# Receiver Operating Characteristic Curve

- Para el caso de múltiples clases, el AUC total se puede calcular:

$$AUC_{total} = \frac{2}{|C|(|C|-1)} \sum_{\{c_i, c_j\} \in C} AUC(c_i, c_j)$$

- AUC( $c_i, c_j$ ) es el área bajo la curva ROC construida con las clases  $c_i$  y  $c_j$ .
- La suma se hace mediante el cálculo del AUC para todos los pares posibles.
- Hay  $|C|(|C|-1)/2$  pares posibles.

# Lift Curve

- Es un método para la etiquetación de dos clases.
- Al igual que la curva ROC, el clasificador debe producir un ranking de las dos clases.
- Los casos de prueba se dividen en N paquetes de igual tamaño. N es regularmente 10 a 20.
- Los ejemplos positivos genuinos en cada paquetes son contados.
- Se hace un gráfico donde en el eje “x” son los porcentaje de los datos y en el “y” los porcentajes acumulados desde el primer al ultimo paquete.

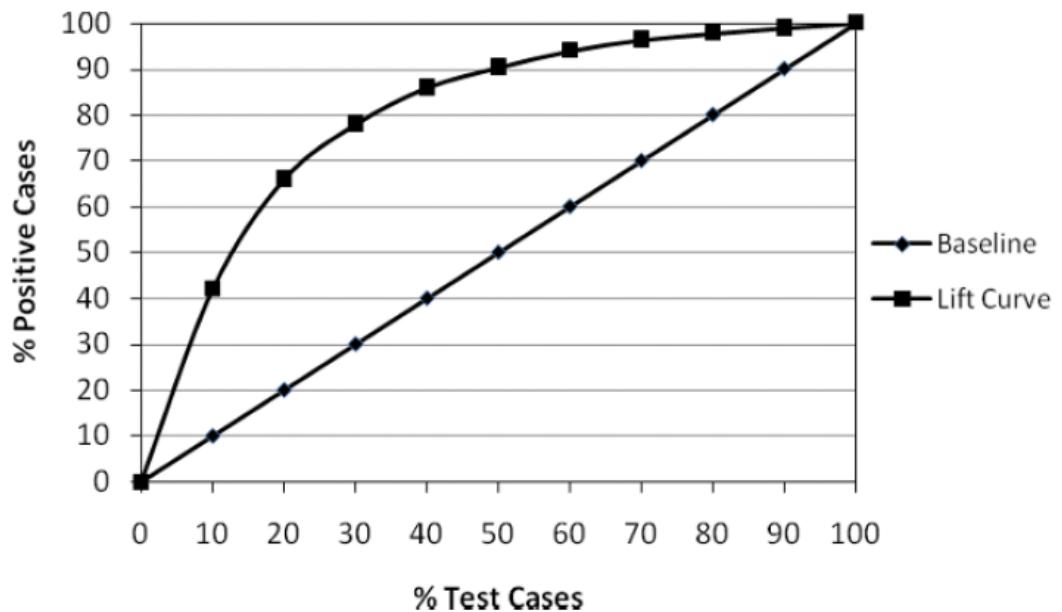
# Lift Curve

- Normalmente, también se incluye una línea base que consiste en una diagonal que va desde (0,0) a (100,100).
- Esta línea representa que no hubo aprendizaje, que los ejemplos positivos están distribuidos uniformemente en los paquetes.
- Entre más grande el área entre el clasificador y la línea base, mejor el clasificador.

# Lift Curve

Bin	1	2	3	4	5	6	7	8	9	10
# test cases	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
# positive cases	210	120	60	40	22	18	12	7	6	5
% of positive cases	42.0%	24.0%	12%	8%	4.4%	3.6%	2.4%	1.4%	1.2%	1.0%
% cumulative	42.0%	66.0%	78.0%	86.0%	90.4%	94.0%	96.4%	97.8%	99.0%	100.0%

Paquete	Nº ejemplos positivos	%
1	210	42
2	120	24
3	60	12
4	40	8
5	22	4,4
6	18	3,6
7	12	2,4
8	7	1,4
9	6	1,2
10	5	1
Total	500	100



# Repaso: Teorema de Bayes

- Expresa la probabilidad condicional de un evento A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal A.
- Vincula la probabilidad de A dado B con la de B dado A.
- Es decir, si se tiene la probabilidad de que si un gato vive en el 7<sup>mo</sup> piso se tire, se puede inferir con algún dato más la probabilidad de que un gato se tire si vive en el 7mo piso.

# Repaso: Teorema de Bayes

- Sea  $A=\{A_1, A_2, \dots, A_n\}$  un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero.
- Sea  $B$  un suceso cualquiera del que se conocen las probabilidades condicionales  $P(B|A_i)$ . Entonces la probabilidad  $P(A_i|B)$  viene dada por la expresión:

$$P(A_i | B) = \frac{P(B | A_i)P(A_i)}{P(B)}$$

# Repaso: Teorema de Bayes

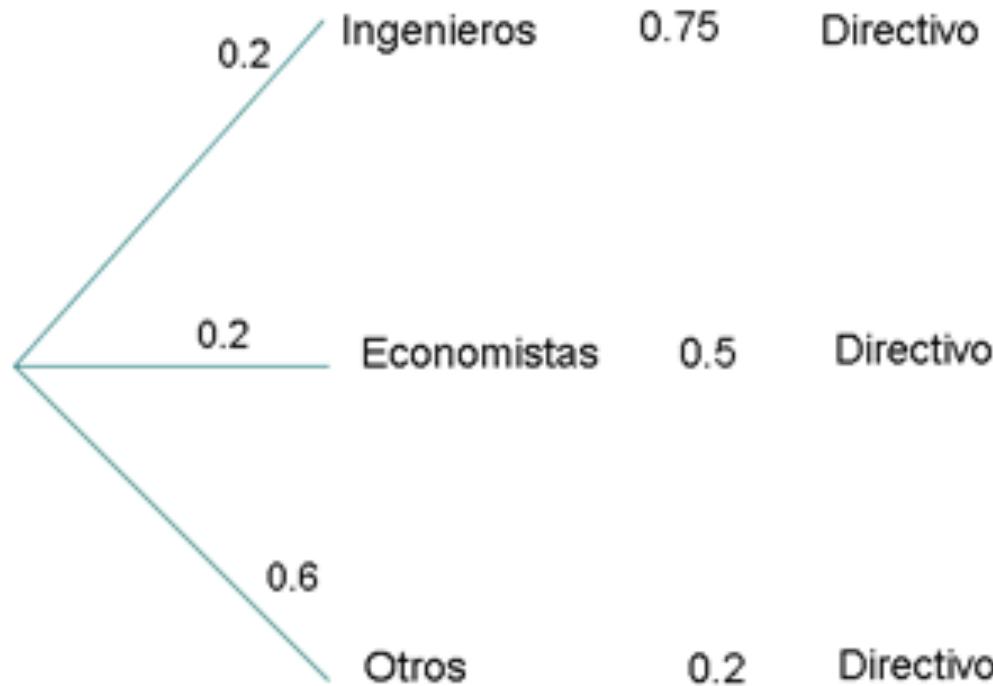
- $P(A_i)$  son las probabilidades a priori.
- $P(B | A_i)$  es la probabilidad de B en la hipótesis  $A_i$ .
- $P(A_i | B)$  son las probabilidades a posteriori.
- Si utilizamos la definición de probabilidad condicionada, tenemos:

$$P(A_i | B) = \frac{P(B | A_i)P(A_i)}{\sum_{k=1}^n P(B | A_k)P(A_k)}$$

# Repaso: Teorema de Bayes

- El 20% de los empleados de una empresa son ingenieros y otro 20% son economistas. El 75% de los ingenieros ocupan un puesto directivo y el 50% de los economistas también, mientras que los no ingenieros y los no economistas solamente el 20% ocupa un puesto directivo. ¿Cuál es la probabilidad de que un empleado directivo elegido al azar sea ingeniero?

# Repaso: Teorema de Bayes

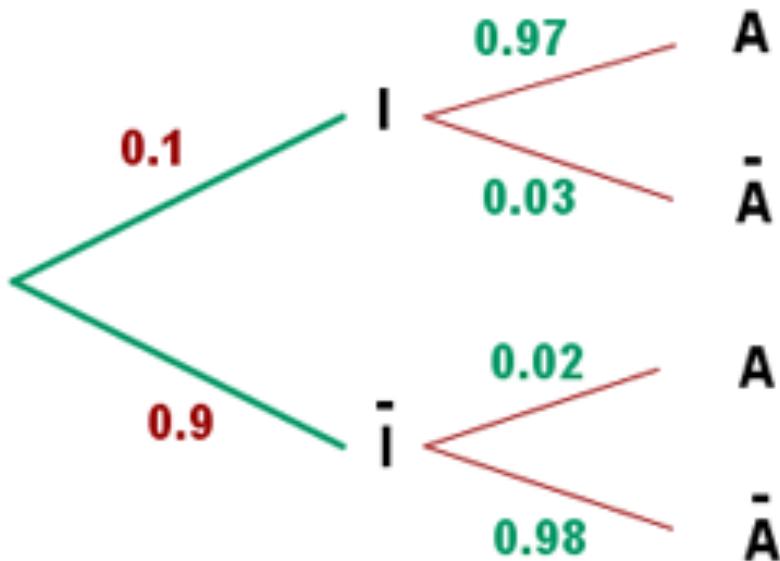


$$P(\text{ingeniero} \mid \text{directivo}) = \frac{0.2 * 0.75}{0.2 * 0.75 + 0.2 * 0.5 + 0.6 * 0.2} = 0.405$$

# Repaso: Teorema de Bayes

- La probabilidad de que haya un accidente en una fábrica que dispone de alarma es 0.1. La probabilidad de que suene esta sí se ha producido algún incidente es de 0.97 y la probabilidad de que suene si no ha sucedido ningún incidente es 0.02.
- En el supuesto de que haya funcionado la alarma, ¿cuál es la probabilidad de que no haya habido ningún incidente?
- Sean los sucesos:
  - $I$  = Producirse incidente.
  - $A$  = Sonar la alarma.

# Repaso: Teorema de Bayes



$$P(\bar{I} | A) = \frac{0.9 * 0.02}{0.1 * 0.97 + 0.9 * 0.02} = 0.157$$

# Clasificación Bayesiana

- La tarea de clasificar puede ser estimada como la probabilidad posterior de la clase dado un ejemplo de prueba  $d$ :

$$\Pr(C = c_j \mid d)$$

Vemos que clase  $C_j$  es más probable, y esa etiqueta asignamos.

# Clasificación Bayesiana

- Supongamos que  $A_1, A_2, \dots, A_{|A|}$  es el conjunto de atributos discretos para un set de datos D.
- Supongamos, además, que C es el atributo de clase con valores  $c_1, c_2, \dots, C_{|C|}$ .
- Dado un ejemplo de prueba  $d$  con valores observados para sus atributos  $a_1, \dots, a_{|A|}$ .
- Entonces,  $d$  es  $\langle A_1=a_1, \dots, A_{|A|}=a_{|A|} \rangle$

# Clasificación Bayesiana

- La predicción es la clase  $c_j$  tal que  $\Pr(C_j | A_1=a_1, \dots, A_{|A|}=a_{|A|})$  es máximo.
- Esta hipótesis es llamada **maximum a posteriori**.
- Entonces, podemos expresar:

$$\begin{aligned}\Pr(C = c_j | d) &= \Pr(C = c_j | A_1 = a_1, \dots, A_{|A|} = a_{|A|}) \\ &= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} | C = c_j) \Pr(C = c_j)}{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|})} \\ &= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} | C = c_j) \Pr(C = c_j)}{\sum_{k=1}^{|C|} \Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} | C = c_k) \Pr(C = c_k)}\end{aligned}$$

# Clasificación Bayesiana

- En resumen,
  - $P(C=c)$  es la probabilidad “prior” de la clase, es decir la que le asignaríamos si no viéramos nada de evidencia.
  - $P(A_1 | C=c)$  es la probabilidad de ver “ $A_1$ ” después de haber visto  $C=c$ .
  - $P(A_1)$  es la probabilidad de la evidencia ¿Cuán común es la representación de este feature dentro de los ejemplos?

# Clasificación Bayesiana

- El denominador es el mismo para todas las clases. Por ende, estamos interesados en calcular:

$$\begin{aligned} & \Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} | C = c_j) \\ &= \Pr(A_1 = a_1 | A_2 = a_2, \dots, A_{|A|} = a_{|A|}, C = c_j)^* \\ & \Pr(A_2 = a_2, \dots, A_{|A|} = a_{|A|} | C = c_j) \end{aligned}$$

# Clasificación Bayesiana

- Para poder trabajar con las probabilidades de manera “fácil” es necesario asumir que todos los atributos son “independiente” dado la clase  $C = C_j$ .
- Lo cual es pocas veces cierto.
- Lo que se traduce en:

$$\Pr(A_1 = a_1 \mid A_2 = a_2, \dots, A_{|A|} = a_{|A|}, C = c_j) = \Pr(A_1 = a_1 \mid C = c_j)$$

# Clasificación Bayesiana

$$\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) = \prod \Pr(A_i = a_i \mid C = c_j)$$

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{k=1}^{|C|} \Pr(C = c_k) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_k)}$$

# Clasificación Bayesiana

- Finalmente, lo que necesitamos estimar son las probabilidades “prior”  $\Pr(C=c_j)$  y las probabilidades condicionales  $\Pr(A_i=a_i|C_j)$ :

$$\Pr(C = c_j) = \frac{\text{Numero de ejemplos de clase } c_j}{\text{Numero de ejemplos en el set de datos}}$$

$$\Pr(A_i = a_i | C = c_j) = \frac{\text{Numero de ejemplos con } A_i = a_i \text{ en clase } c_j}{\text{Numero de ejemplos de clase } c_j}$$

# Clasificación Bayesiana

- Si sólo necesitamos una decisión sobre la clase más probable, podemos reducir la expresión a:

$$c = \arg \max_{c_j} \Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i | C = c_j)$$

# Clasificación Bayesiana

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

# Clasificación Bayesiana

- $\Pr(C=t) = \frac{1}{2}$
- $\Pr(c=f) = \frac{1}{2}$
- $\Pr(A=m | C=t) = 2/5 \quad \Pr(A=m | C=f) = 1/5$
- $\Pr(A=g | C=t) = 2/5 \quad \Pr(A=g | C=f) = 2/5$
- $\Pr(A=h | C=t) = 1/5 \quad \Pr(A=h | C=f) = 2/5$
- $\Pr(B=b | C=t) = 1/5 \quad \Pr(B=b | C=f) = 2/5$
- $\Pr(B=s | C=t) = 2/5 \quad \Pr(B=s | C=f) = 1/5$
- $\Pr(B=q | C=t) = 2/5 \quad \Pr(B=q | C=f) = 2/5$

# Clasificación Bayesiana

- Si queremos clasificar un nuevo caso:
  - A=m y B=q
- Tenemos,

$$\Pr(C = t) \prod_{j=1}^2 \Pr(A_j = q_j | C = t) = \frac{1}{2} * \frac{2}{5} * \frac{2}{5} = \frac{2}{25}$$

$$\Pr(C = f) \prod_{j=1}^2 \Pr(A_j = q_j | C = \text{MSG}) = \frac{1}{2} * \frac{1}{5} * \frac{2}{5} = \frac{1}{25}$$

# Clasificación Bayesiana

- Una de las ventajas de este método de aprendizaje es que es rápido de entrenar, ya que se puede construir haciendo una revisión lineal a los datos.
- A pesar de asumir independencia, los resultados son bastante competitivos.
- Pero hay que estudiar algunas otras cosas como: ¿qué pasa con atributos no discretos? ¿Valores que no existen? y ¿Zero counts?

# Clasificación Bayesiana

- Los atributos numéricos contiguos hay que discretizarlos.
- Los valores que no existen son normalmente ignorados.
- El problemas de las “zero counts”: se puede dar que un feature que en el conjunto de prueba nunca aparece en el conjunto de entrenamiento.
  - ¿Puedes pensar por qué ésto es problemático?

# Clasificación Bayesiana

- Es problemático porque una probabilidad cero, hace automáticamente cero todo el producto.
- La manera de corregir esto es introducir una corrección en todas las probabilidades, lo que se conoce como suavizamiento (en inglés smoothing).

# Clasificación Bayesiana

- Sea  $n_{ij}$  el número de ejemplos que están sobre  $A_i = a_i$  y  $C = c_j$ . Sea  $n_j$  el número total de ejemplos con  $C=c_j$  en el conjunto de entrenamiento.
- La estimación no corregida para  $\Pr(A_i=a_i | C=c_j)$  es  $n_{ij}/n_j$ . La estimación corregida está dada por:

$$\Pr(A_i = a_i | C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda m_i}$$

# Clasificación Bayesiana

- Donde  $m_i$  es el número de valores para el atributo  $A_i$  (e.g., dos para el caso booleano), y  $\lambda$  es un valor multiplicativo generalmente igualado a  $1/n$ . Donde  $n$  es el número de ejemplos en el conjunto de prueba.
- Cuando  $\lambda=1$  se tiene lo que se conoce como la ley de Laplace de sucesión.
- La fórmula de corrección antes vista es conocida como la ley de sucesión de Lidstone.

# Clasificación Bayesiana

- Repasando las probabilidades del ejemplo tenemos:
  - $\Pr(A=m|C=t) = (2+1/10)/(5+3/10) = 2.1/5.3 = 0.396$
  - $\Pr(B=b|C=t) = (1+1/10)/(5+3/10) = 1.1/5.3 = 0.208$

# Repaso: Distribución Multinomial

- Propiedades del experimento multinomial:
  - El experimento consiste en n pruebas idénticas.
  - Hay k posibles resultados de cada prueba
  - Las probabilidades de los k resultados, denotadas por  $p_1, p_2, p_3, \dots, p_k$  se mantienen constantes a lo largo de todas las pruebas, donde  $p_1 + p_2 + \dots + p_k = 1$
  - Las pruebas son independientes
  - Las variables aleatorias de interés son las cuentas  $y_1, y_2, \dots, y_k$  en cada una de las k categorías de clasificación
- $P(y_1, y_2, \dots, y_k) = (p_1)^{y_1} (p_2)^{y_2} \dots (p_k)^{y_k}$

# Repaso: Distribución Multinomial

- Donde
  - $p_i$ =probabilidad del resultado i en una sola prueba
  - $p_1 + p_2 + \dots + p_k = 1$
  - $n= y_1, y_2, \dots, y_k$  = numero de pruebas
  - $y_i$ =número de ocurrencias del resultado i en n pruebas
  - La media y la varianza de las variables multinomial  $y_i$  son respectivamente
    - $\mu=np_i$
    - $\sigma^2=np_i(1-p_i)$

# Repaso: Distribución Multinomial

- **Relaciones entre Multinomial y Binomial**
  - Para el caso en que  $k = 2$ , uno se puede convencer que la distribución multinomial coincide con la binomial: interpretando que si no se está en la categoría  $y_i$ , se está fuera de la categoría  $y_i$ . Como  $p_1 + p_2 = 1$ ,  $q - p_2 = 1 - p_1$  y decimos  $p = p_1$ .
  - De igual forma,  $n_2 = n - n_1$ . Reemplazando en la distribución multinomial los valores anteriores, se obtiene que  $P(N_1 = n_1, N_2 = n_2) = P(N_1 = n_1)$ , donde  $N_1$  se distribuye como un binomio de parámetros  $n$  y  $p = p_1$ .

# Repaso: Distribución Multinomial

- Ejemplo:
  - Una empresa desea conocer la opinión que se tiene sobre tres productos, A, B, C. Sabiendo que el producto A es preferido por el 10 % de los consumidores, el B, por el 30% y el C, por el 40%.
  - ¿Cuál es la probabilidad de que en una muestra aleatoria de 10 personas, dos prefieran A, tres prefieran B y dos prefieran el producto C?
  - Se realizan 10 experimentos consistentes en preguntar a 10 personas sus preferencias por unos productos determinados. Las opciones son cuatro:
    - Preferir A con probabilidad 0,1
    - Preferir B con probabilidad 0,3
    - Preferir C con probabilidad 0,4
    - No preferir ninguno con probabilidad 0,2

# Repaso: Distribución Multinomial

- Sea:
  - $X_1$ =número de personas que prefieren A
  - $X_2$ =número de personas que prefieren B
  - $X_3$ =número de personas que prefieren C
  - $X_4$ =número de personas que no prefieren ninguno
- Calculamos:

$$P[X_1 = 2, X_2 = 3, X_3 = 2, X_4 = 3] = 0.1^2 \cdot 0.3^3 \cdot 0.4^2 \cdot 0.2^3 = 0.0087$$



# Clasificación Bayesiana en Textos

- Clasificación de textos es el problema de aprendizaje de modelos de etiquetado dado un conjunto de documentos anotados de acuerdo a un conjunto de clases pre-definidas.
- Por ejemplo, tenemos tres clases de documentos deportes, tecnología, farándula.
- En general, se pueden aplicar las técnicas anteriores para clasificar documentos, pero hay versiones modificadas que mejoran los resultados.

# Máquinas de Soporte Vectorial

- Es un sistema de aprendizaje que tiene muchas propiedades que son deseables.
- No sólo tiene fuertes fundamentos matemáticos y estadísticos, sino que también en la práctica suele ser uno de los métodos más efectivos para muchas de las aplicaciones.
- En general, las máquinas de soporte vectorial (SVM por su nombre en inglés) son un sistema de aprendizaje lineal que construye clasificadores de dos clases.

# Máquinas de Soporte Vectorial

- Si consideramos un set de ejemplos D de la siguiente forma:
  - $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
- Donde  $x_i = (x_{i1}, x_{i2}, \dots, x_{ir})$  es un vector de entrada de r dimensiones en un espacio real  $X \subseteq \mathbb{R}^r$ .
- Por otro lado,  $y_i$  son las etiquetas de clases o valores de salida,  $y_i \in \{-1, 1\}$ . Donde 1 denota la clase positiva y -1 negativa.
- Los cambios en la notación son para reflejar lo común en la comunidad de la gente de SVM.

# Máquinas de Soporte Vectorial

- Para construir un clasificador, SVM busca una función lineal de la forma:

$$f(x) = \langle w \cdot x_i \rangle + b$$

- Donde un vector de entrada  $x_i$  es asignado a la clase positiva si  $f(x_i) \geq 0$ , en el caso contrario a la negativa:

$$y_i = \begin{cases} 1 & \text{si } \langle w \cdot x_i \rangle + b \geq 0 \\ -1 & \text{si } \langle w \cdot x_i \rangle + b < 0 \end{cases}$$

# Máquinas de Soporte Vectorial

- Entonces,
  - $f(x)$  es una función en los reales  $X \subseteq \mathbb{R}^r \rightarrow \mathbb{R}$ .
  - $w(w_1, w_2, \dots, w_r) \in \mathbb{R}^r$  es el vector de pesos.
  - $b \in \mathbb{R}$  es un sesgo.
  - $\langle w \cdot x \rangle$  es el producto interno euclíadiano entre  $w$  y  $x$ .
- Sin utilizar notación vectorial, podemos escribir:
  - $f(x_1, x_2, \dots, x_r) = w_1 x_1 + w_2 x_2 + \dots + w_r x_r + b$

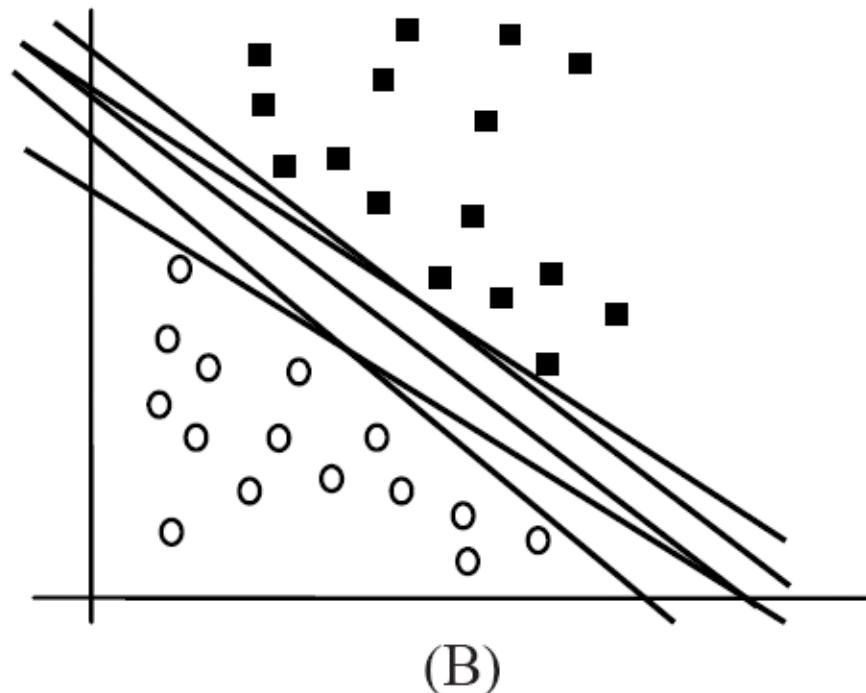
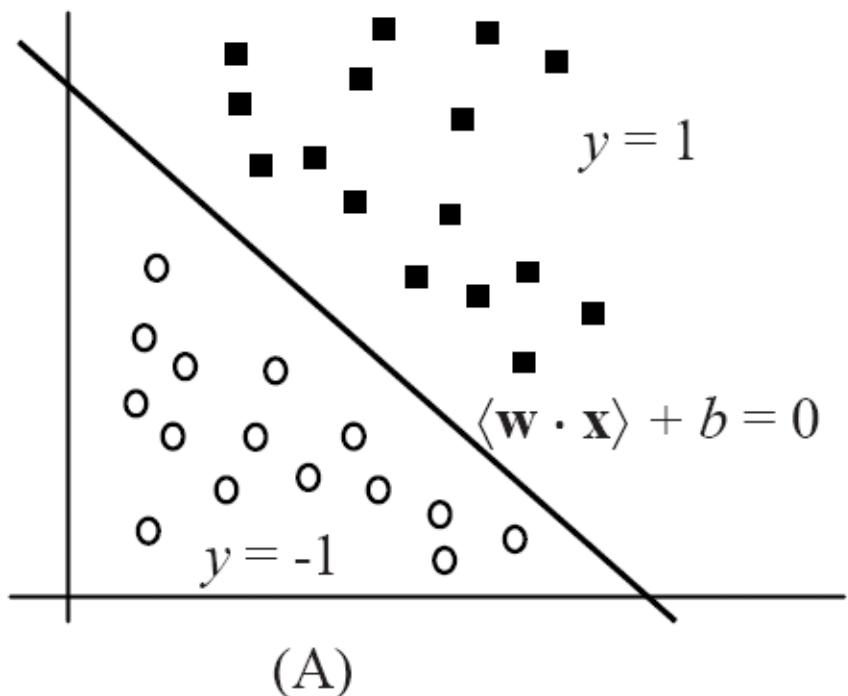
# Máquinas de Soporte Vectorial

- SVM encuentra el hiperplano que mejor separa los ejemplos positivos y negativos:

$$\langle w \cdot x \rangle + b = 0$$

- Este hiperplano separa el espacio de entrada (entrenamiento) en dos mitades: positiva y negativa.
- Recuerde que un hiperplano es llamado una línea en el espacio 2-dimensional, y un plano en el 3-dimensional.

# Máquinas de Soporte Vectorial



# Máquinas de Soporte Vectorial

- La figura A invita a hacerse algunas preguntas:
  - Hay una infinidad de hiperplanos que pueden separar los ejemplos positivos y negativos. ¿Cuál debemos escoger?
  - Un hiperplano puede separar los datos si y sólo si éstos son separables linealmente. ¿Qué pasa cuando no lo son?
- SVM da respuesta a ambas preguntas:

# Máquinas de Soporte Vectorial

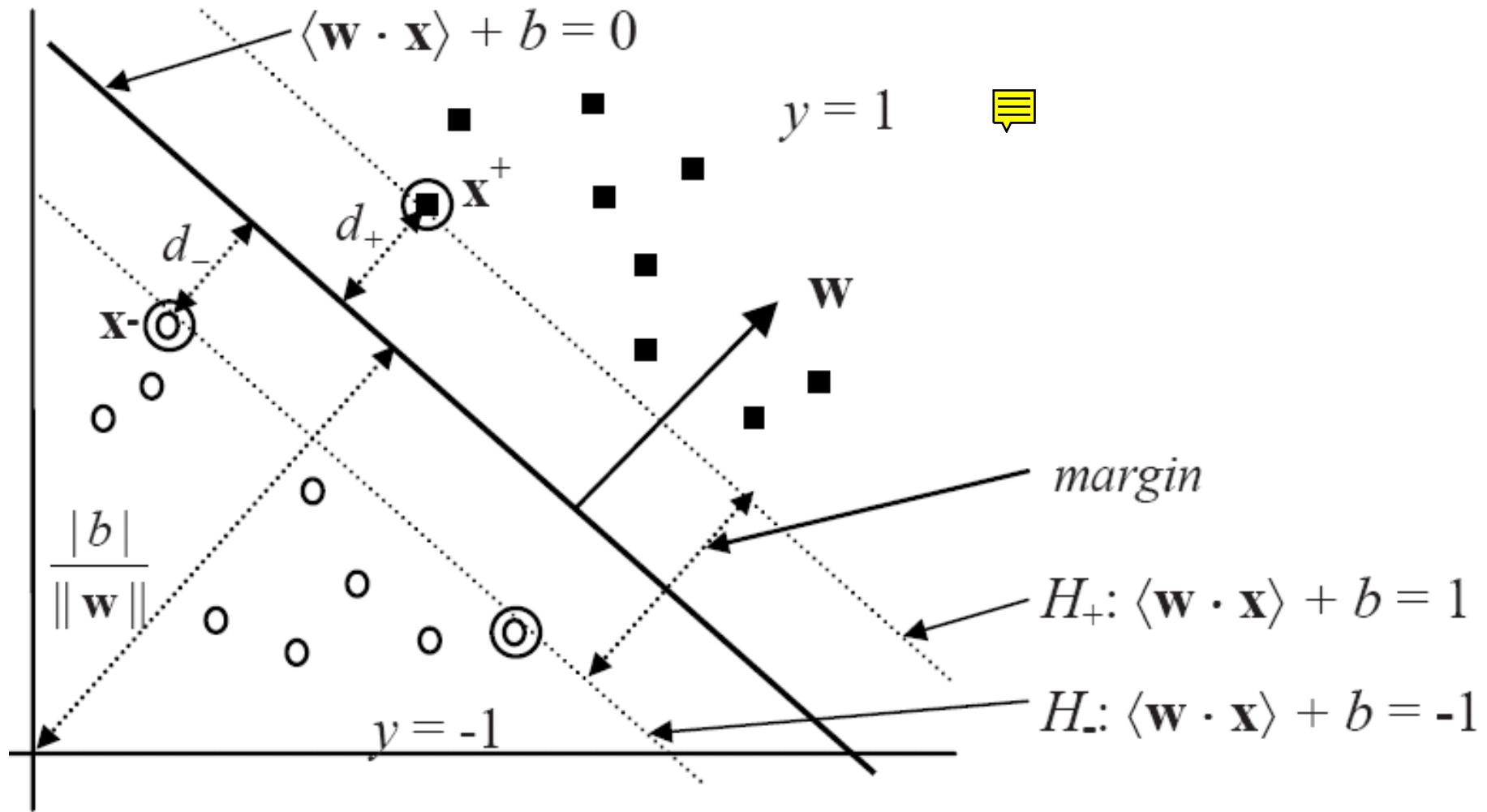
- SVM da respuesta a ambas preguntas:
  - Busca el hiperplano que maximiza el gap entre los datos positivos y negativos.
  - Usa funciones de kernel.
- SVM requiere:
  - Datos numéricos.
  - Y etiqueta de acuerdo a dos clases.

# SVM Lineal: Caso Separable

- En este caso,  $w$  define una dirección perpendicular al hiperplano. En álgebra lineal,  $w$  es llamado vector normal o simplemente normal.
- Si no se cambia  $w$ , entonces sabemos que un cambio de  $b$  significa un desplazamiento paralelo del hiperplano.
- Uno puede escalar el plano por un factor  $\lambda$ :

$$\langle w \cdot x \rangle + b = \langle \lambda w \cdot x \rangle + \lambda b = 0, \quad \lambda \in \mathbb{R}^+$$

# SVM Lineal: Caso Separable



# SVM Lineal: Caso Separable

- Hay que encontrar  $d_+$  y  $d_-$ , las distancias del hiperplano a los ejemplos positivo y negativos más cercanos.
- La idea es maximizar el margen  $d_+ + d_-$ . Este hiperplano es llamado de máximo margen y la función o límite de decisión.
- Resultados teóricos indican que maximizar este margen minimiza los errores de clasificación.

# SVM Lineal: Caso Separable

- Consideremos un punto  $(x_+, 1)$  y otro  $(x_-, -1)$  como los más cercanos al hiperplano  $\langle w \cdot x \rangle + b = 0$ . 
- Definamos como  $H_+$  y  $H_-$  los planos paralelos que pasan por  $x_+$  y  $x_-$ , respectivamente.
- Recordar que si son paralelos tienen la misma norma.

# SVM Lineal: Caso Separable

- Se puede re-escalar  $w$  y  $b$  para obtener:

$$H+ : \langle w \cdot x+ \rangle + b = 1$$

$$H- : \langle w \cdot x- \rangle + b = -1$$

- Tal que,

$$\langle w \cdot x_i \rangle + b \geq 1 \quad \text{si } y_i = 1$$

$$\langle w \cdot x_i \rangle + b \leq -1 \quad \text{si } y_i = -1$$

- Indicando que no hay datos de entrenamientos entre  $H+$  y  $H-$ .

# SVM Lineal: Caso Separable

- La distancia perpendicular euclíadiana desde un punto  $x_i$  a un hiperplano está dada por:

$$\frac{|\langle w \cdot x_i \rangle + b|}{\|w\|}$$

- Donde la norma euclíadiana  $\|w\|$  de  $w$  está dada por:

$$\|w\| = \sqrt{\langle w \cdot w \rangle} = \sqrt{{w_1}^2 + {w_2}^2 + \dots + {w_r}^2}$$

# SVM Lineal: Caso Separable

- Para calcular  $d_+$ :
  - En vez de calcular la distancia de  $x_+$  al hiperplano, debemos escoger cualquier punto  $x_s$  en el hiperplano  $\langle w \cdot x \rangle + b = 0$  y calcular la distancia al hiperplano dado por  $\langle w \cdot x^+ \rangle + b = 1$ .
  - Recordar que  $\langle w \cdot x_s \rangle + b = 0$
- Entonces, la distancia de  $x_s$  a  $\langle w \cdot x^+ \rangle + b = 1$ :

$$d_+ = \frac{|\langle w \cdot x_s \rangle + b - 1|}{\|w\|} = \frac{1}{\|w\|}$$

# SVM Lineal: Caso Separable

- Haciendo lo mismo con  $H_-$ , obtenemos  $d_- = 1/\|w\|$ , entonces sabemos que el discriminante está en la mitad de  $H_+$  y  $H_-$ .
- Entonces, el margen  $d_+ + d_-$  está dado por:  $2/\|w\|$ .
- El aprendizaje consiste en el siguiente problema de minimización:

$$\min \text{imizar} : \frac{\langle w \cdot w \rangle}{2}$$

*Sujeto a :*

$$\langle w \cdot x_i \rangle + b \geq 1 \quad y_i = 1$$

$$\langle w \cdot x_i \rangle + b \leq -1 \quad y_i = -1$$

# SVM Lineal: Caso Separable

- Dado que:
  - La función objetivo es cuadrática y convexa.
  - Las restricciones son lineales en términos de  $w$  y  $b$ .
- Por ende, podemos utilizar los multiplicadores de Lagrange estándar para resolver este problema.
- El problema se reduce a optimizar el lagragiano que considera las restricciones.

# SVM Lineal: Caso Separable

- Re-escribiendo las restricciones:

$$y_i(\langle w \cdot x_i \rangle + b) \geq 1, \quad i = 1, 2, 3, \dots, n$$

- Dado que tenemos restricciones que sólo usan el “ $\geq$ ”, el lagragiano es formado sólo por restricciones multiplicadas por multiplicadores positivos ( $\alpha_i$ ) de Lagrange que se restan de la función objetivo:

$$L_p = \frac{1}{2} \langle w \cdot w \rangle - \sum_{i=1}^n \alpha_i [y_i(\langle w \cdot x_i \rangle + b) - 1]$$

# SVM Lineal: Caso Separable

- La teoría de la optimización nos dice que la solución a esta ecuación debe cumplir las condiciones de Kuhn-Tucker:

$$\frac{\partial L_p}{\partial w_j} = w_j - \sum_{i=1}^n y_i \alpha_i x_{ij} = 0, \quad j = 1, 2, \dots, r \quad (1)$$

$$\frac{\partial L_p}{\partial b} = - \sum_{i=1}^n y_i \alpha_i = 0 \quad (2)$$

$$y_i (\langle w \cdot x_i \rangle + b) - 1 \geq 0, \quad i = 1, 2, \dots, n \quad (3)$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, n \quad (4)$$

$$\alpha_i (y_i (\langle w \cdot x_i \rangle + b) - 1) = 0, \quad i = 1, 2, \dots, n \quad (5)$$

# SVM Lineal: Caso Separable

- En particular podemos ver que (3) es el conjunto inicial de restricciones.
- Aunque hay un multiplicador de lagrange  $\alpha_i$  para cada caso de entrenamiento, sólo aquellos en los planos de los márgenes pueden tener  $\alpha_i > 0$ .
- Ésto lo vemos en la condición (5), ya que sólo para los puntos en  $H_+$  y  $H_-$  el factor  $y_i(\langle w \cdot x_i \rangle + b) - 1 = 0$ .

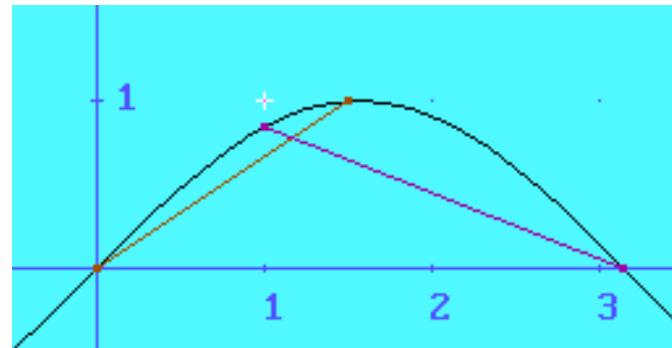
# SVM Lineal: Caso Separable



- Estos puntos son los llamados **soporte vectorial**.
- En general, las condiciones de Kuhn-Tucker son necesarias pero no suficientes.
- Pero dado que la función objetivo es convexa, entonces también son suficientes.

# Repaso: Funciones Cóncavas

- Una función es estrictamente cónica en un conjunto  $M$  convexo **si todo** segmento que une dos puntos de la gráfica está estrictamente por debajo de la gráfica.
- Una función es cónica (no estricta) **cuando no todas** las cuerdas que unen puntos de la gráfica en dicho intervalo quedan estrictamente por debajo.



# Repaso: Funciones Cóncavas

- Una función es cóncava si ( $\lambda \in [0,1]$ ):

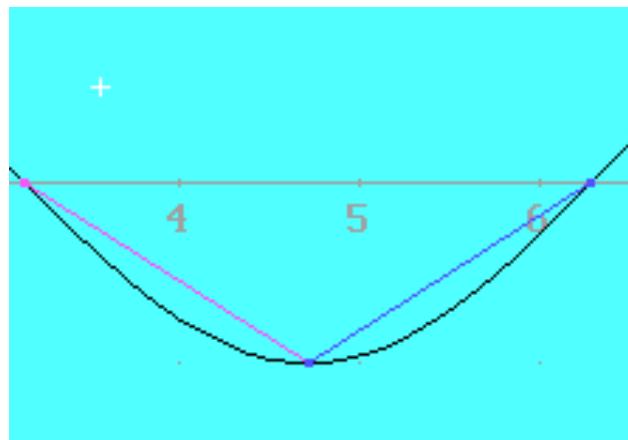
$$f((1-\lambda)x + \lambda y) \geq (1-\lambda)f(x) + \lambda f(y)$$

- Una función es estrictamente cóncava si ( $\lambda \in [0,1]$ ):

$$f((1-\lambda)x + \lambda y) > (1-\lambda)f(x) + \lambda f(y)$$

# Repaso: Funciones Convexas

- Una función es convexa en un intervalo si todo segmento que une dos puntos de la gráfica queda por encima de la gráfica. Si siempre queda estrictamente por encima decimos que la función es estrictamente convexa.



# Repaso: Funciones Convexas

- Una función es convexa si ( $\lambda \in [0,1]$ ):

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

- Una función es estrictamente convexa si ( $\lambda \in [0,1]$ ):

$$f((1-\lambda)x + \lambda y) < (1-\lambda)f(x) + \lambda f(y)$$

# Máquinas de Soporte Vectorial

- Otro concepto importante para manejar es el del **problema dual**:
  - La idea es reformular el problema de una manera alternativa talque sea más conveniente de resolver computacionalmente o tiene alguna relevancia teórica.
- En nuestro caso, el problema dual se puede obtener mediante la asignación de cero a las derivadas parciales del lagragiano con respecto a las variables  $w$  y  $b$ .

# Máquinas de Soporte Vectorial

- En términos prácticos, se traduce en reemplazar las siguientes dos ecuaciones en  $L_p$ , generando una nueva  $L_D$ :

$$w_j = \sum_{i=1}^n y_i \alpha_i x_{ij}, \quad j = 1, 2, \dots, r$$

$$\sum_{i=1}^n y_i \alpha_i = 0$$

$$L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j \langle x_i \cdot x_j \rangle$$

# Máquinas de Soporte Vectorial

- $L_D$  contiene sólo variables duales y debe ser maximizado con respecto a las restricciones (segundo conjunto de restricciones en la ponencia anterior), y  $\alpha_i \geq 0$ . 
- Esta formulación dual es conocida como el **dual de Wolfe**. Tiene la propiedad que para la función convexa original, los  $\alpha_i$ 's en el máximo de  $L_D$  dan  $w$  y  $b$  ocurriendo en el mínimo de  $L_p$ .

# Máquinas de Soporte Vectorial

- Resolviendo el problema dual, nos permite obtener los valores de  $\alpha_i$  que son utilizadas para calcular  $b$  y  $w$ .
- Dado que los valores de  $\alpha_i$  se calculan numéricamente, por ende tener errores, todos los vectores de soporte se utilizan para calcular  $b$ .
- Por ende, el hiperplano de máximo margen está dado por:

$$\langle w \cdot x \rangle + b = \sum_{i \in sv} y_i \alpha_i \langle x_i \cdot x \rangle + b = 0$$

# Máquinas de Soporte Vectorial

- En la fórmula anterior sv es índice los vectores de soporte originados por los datos de entrenamiento.
- Pronósticos:



$$\text{sign}(\langle w \cdot z \rangle + b) = \text{sign}\left(\sum_{i \in sv} y_i \alpha_i \langle x_i \cdot z \rangle + b\right)$$

Si retorna 1 es clasificado como positivo, sino negativo

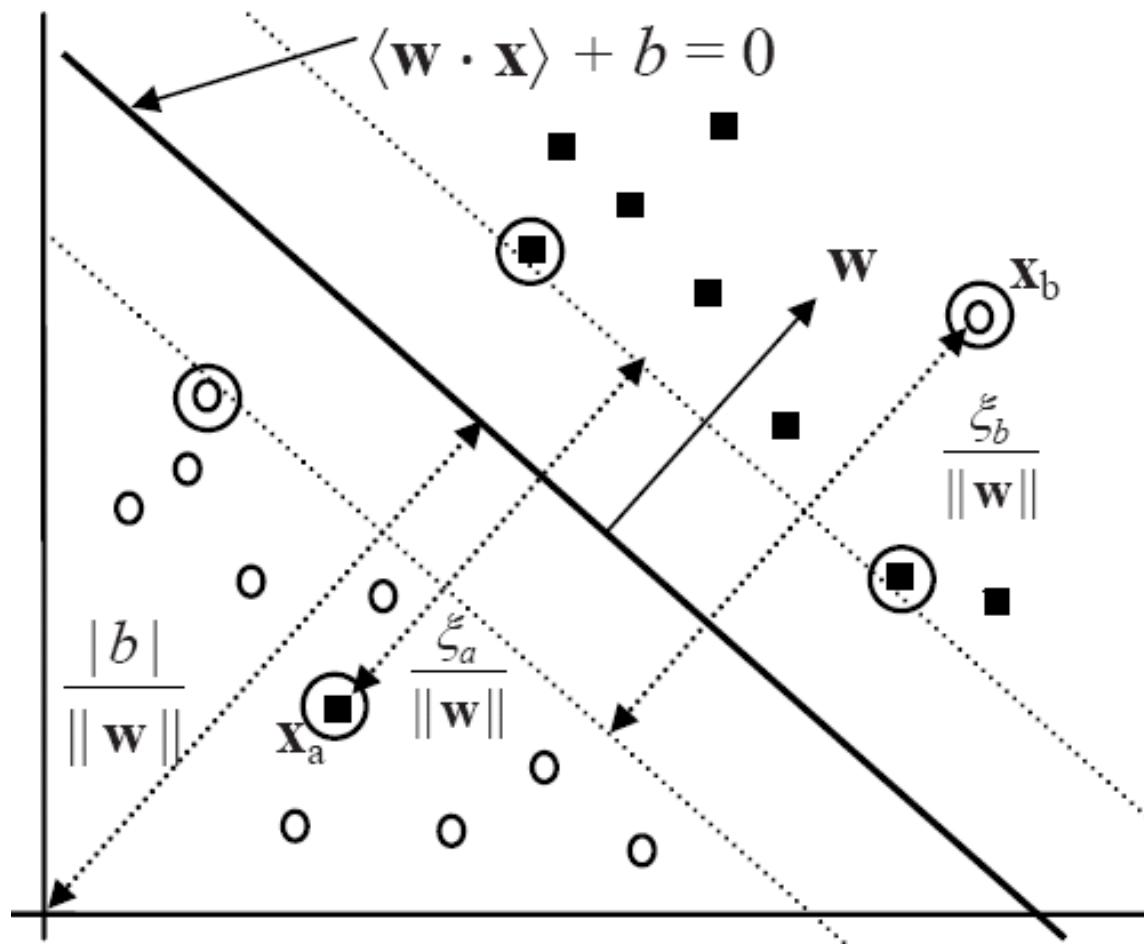
# SVM: Caso no Separable

- Muchas veces hay errores en los datos, otras hay cierta aleatoriedad.
- Por muchas razones, es necesario ir más allá del caso ideal.
- Para admitir errores en los datos, se utilizan las **variable de relajo**  $\xi_i$ :

$$y_i(\langle w \cdot x_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, n.$$

$$\xi_i \geq 0, \quad i = 1, 2, \dots, n.$$

# SVM: Caso no Separable



# SVM: Caso no Separable

- Los errores se penalizan en la función objetivo:

$$\begin{aligned} \text{Min} \quad & \frac{\langle w \cdot w \rangle}{2} + C \left( \sum_{i=1}^n \xi_i \right)^k \\ \text{Subject to:} \quad & y_i (\langle w \cdot x_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, n \\ & \xi_i \geq 0, \quad i = 1, 2, \dots, n \end{aligned}$$

- Donde  $C \geq 1$  y  $k=1$ .

# SVM: Caso no Separable

- Esta reformulación es llamada soft-margin SVM.
- Y siguiendo un procedimiento análogo al caso lineal, tenemos el siguiente problema dual:

$$\max L_D(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j < x_i \cdot x_j >$$

$$\sum_{i=1}^n y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n$$

# SVM: Caso no Separable

- Vemos que a diferencia del caso separable, tenemos sólo un nuevo conjunto de restricciones ( $0 \leq \alpha_i \leq C$ ).
- Las ecuaciones no dependen de  $\xi_i$ .
- El parámetro  $C$  se calcula empíricamente.

$$b = \frac{1}{y_i} - \sum_{i=1}^n y_i \alpha_i \langle x_i \cdot x_j \rangle$$

# SVM: Caso no Separable

- La función de prueba es la misma:

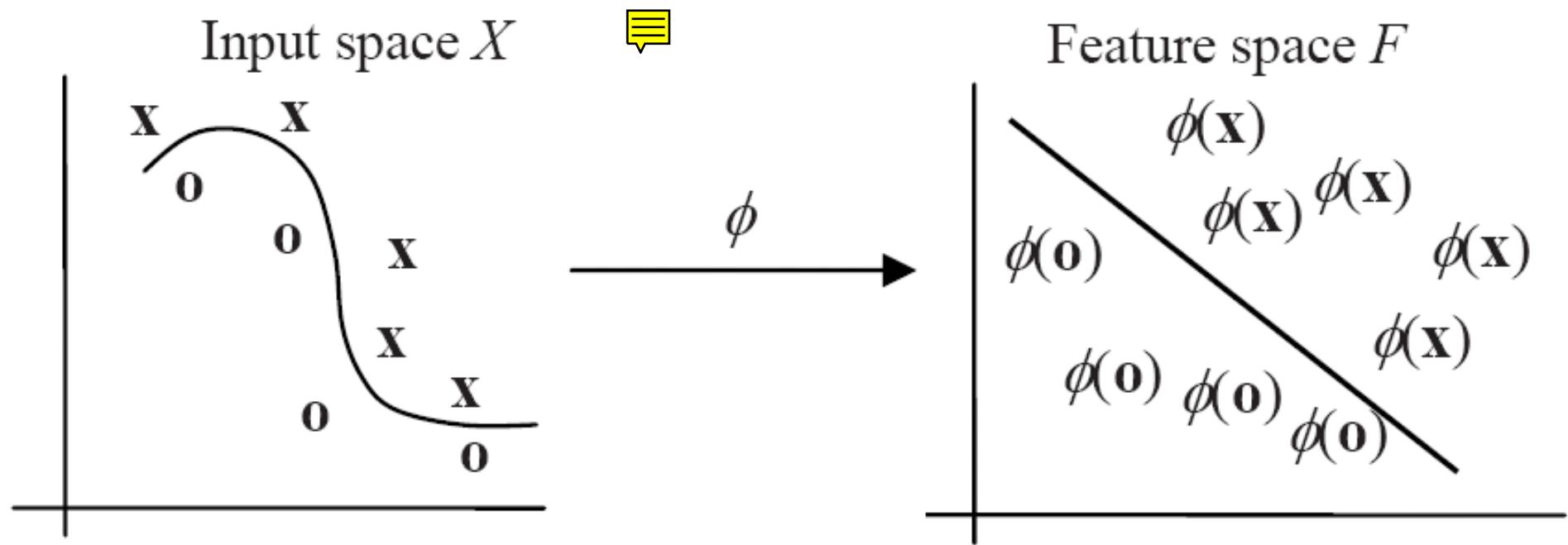
$$\text{sign}(\langle w \cdot x \rangle + b) = \text{sign}\left(\sum_{i \in sv} y_i \alpha_i \langle x_i \cdot x \rangle + b\right)$$

Si retorna 1 es clasificado  
como positivo, sino negativo

# SVM no-lineal: Funciones de Kernel

- Para muchas aplicaciones del mundo real, no es posible construir un hiperplano que separe linealmente los datos.
- La idea consiste en mapear los datos de entrenamiento a un espacio intermedio donde si son separables.
- El espacio de datos original  $X$  se llama de entrada (en inglés input space) y el intermedio feature space. Ésto se hace mediante una función de mapeo no-lineal  $\varphi:X\rightarrow F$ .

# SVM no-lineal: Funciones de Kernel



$$\sum_{i=1}^n y_i \alpha_i \langle \varphi(x_i) \cdot \varphi(x) \rangle + b$$

# SVM no-lineal: Funciones de Kernel

- Ejemplo de función de mapeo:

$$(x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$



- El ejemplo de entrenamiento  $((2,3), -1)$  es transformado en  $((4,9,8.5), -1)$ .
- Este ejemplo resalta uno de los problemas del feature space: el incremento de la dimensionalidad. Este efecto puede hacerlo computacionalmente intratable.

# SVM no-lineal: Funciones de Kernel

- Una de las ventajas es que si recordamos que resolvemos el problema dual, entonces podemos aprovechar de que no necesitamos el mapeo de los vectores, sino sólo calcular los productos punto  $\langle \varphi(x) \cdot \varphi(z) \rangle$ .
- Es posible calcular los productos puntos sin hacer el mapeo...muchas veces.
- Una función de kernel se define como  $K(x,z) = \langle \varphi(x) \cdot \varphi(z) \rangle$ . 

# SVM no-lineal: Funciones de Kernel

- Por ejemplo la función de kernel polinomial:

$$K(x, z) = \langle x \cdot z \rangle^d$$

- Ejemplo:  $d=2$ ,  $x=(x_1, x_2)$  y  $z=(z_1, z_2)$

$$\langle x \cdot z \rangle^2 = (x_1 z_1 + x_2 z_2)^2$$

$$= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2$$

$$= \langle (x_1^2, x_2^2, \sqrt{2}x_1 x_2) \cdot (z_1^2, z_2^2, \sqrt{2}z_1 z_2) \rangle$$

$$= \langle \varphi(x) \cdot \varphi(z) \rangle$$

# Máquinas de Soporte Vectorial

- Como el algoritmo de aprendizaje es independiente de los kernels, se pueden estudiar por separado. Es decir, variar los kernels sin variar el algoritmo de aprendizaje.
- Limitaciones:
  - Sólo funciona para valores de atributos reales. Para otros atributos hay que hacer una función de mapeo, normalmente binaria.
  - Sólo permite dos clases. Hay otras estrategias para múltiples clases.
  - Los hiperplanos son difíciles de graficar, más aún cuando utilizamos kernels.

# SVM no-lineal: Funciones de Kernel

- En general, el número de dimensiones en el feature space para el kernel polinomial es:

$$\binom{r+d-1}{d}$$

- En general, tenemos:

$$Polinomial : K(x, z) = (\langle x \cdot z \rangle + \theta)^d$$

$$Gausiano \quad RBF : K(x, z) = e^{-\|x-z\|^2/2\sigma}$$

Donde  $\theta \in \mathbb{R}$ ,  $d \in \mathbb{N}$ , y  $\sigma > 0$ .

# SVM: Recapitulación

- La idea de una separación máxima es buena, porque permite asegurar una mejor clasificación.
  - Menos probable que ejemplos de prueba caigan en el margen.
  - Dado que los ejemplos de prueba deberían seguir una distribución similar a la de entrenamiento.
- La otra idea importante con SVM es que no necesariamente todos los puntos del mismo tipo deben caer en el lado correcto del plano.
  - Penalización de los puntos que caen en el lado erróneo del margen.

# SVM: Recapitulación

- La penalización de un ejemplo mal clasificado es proporcional a la distancia al margen.
- La penalización se maneja con diversas funciones de pérdida. Se llaman “loss functions”.
  - La que usa SVM se llama “hinge loss.” 
- Está también la “Zero-one loss” que asigna cero a cada ejemplo bien clasificado y uno a los mal clasificados.
- La distancia cuadrada que especifica la penalización en conformidad al cuadrado de la distancia al margen.

# K-vecinos más Cercanos

- Al contrario de métodos vistos como árboles de decisión, clasificación bayesiana y hiperplanos, los k-vecinos más cercanos no aprenden un modelo.
- Los métodos anteriores son conocidos como “eager learning”, en cambio k-vecinos más cercanos es como conocido como “lazy learning”.
- En esta técnica, el aprendizaje ocurre sólo cuando un ejemplo necesita ser clasificado.

# K-vecinos más Cercanos

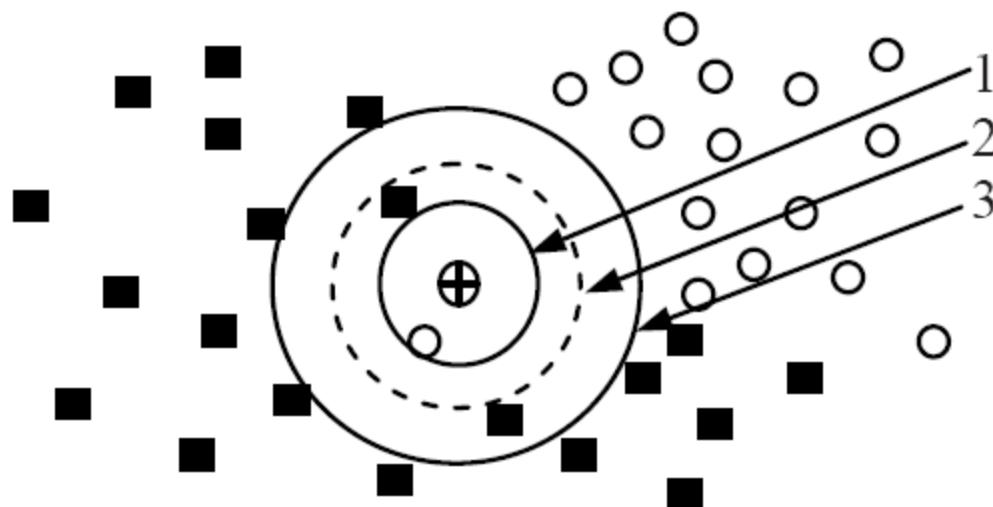
- En este método no se hace nada con los datos de entrenamiento.
- Cuando aparece una instancia de prueba, ésta se compara con todos los datos de entrenamiento.
- Esa comparación es basada en alguna métrica de similitud.
- Entonces los  $k$  vecinos más cercanos son escogidos.
- Se escoge la etiqueta más común dentro de estos vecinos.



# K-vecinos más Cercanos



- Hay muchas opciones para la métrica de distancia, por ejemplo euclidiana o coseno.
- El número de vecinos es normalmente determinado por un conjunto de validación, o bien, mediante cross-validation.



# K-vecinos más Cercanos

- Se ha mostrado empíricamente que para algunas tareas de clasificación de texto es tan competitivo como técnicas más sofisticadas, i.e. SVM.
- Es bastante flexible.
- Es lento al clasificar, porque se compara cada caso de prueba con todos los datos de entrenamiento.
- No produce un modelo entendible.

# K-vecinos más Cercanos

- Se utiliza para:
  - Encontrar compañías parecidas a las que prefieren nuestros compañías clientes.
  - Los consumidores en-línea que se parecen más a nuestros consumidores de retail.
  - Esto sirve, generalmente, para direccionar avisos.

# K-vecinos más Cercanos

- Ejemplo: Supongamos que a alguien le gusta el Whisky puro de malta escocés. Hay muchos tipos de este whiskies. Hay muchas variaciones de este tipo de whisky.
- Cuando uno encuentra uno, le gustaría saber cuales otros hay parecidos porque en un restaurant puede haber de un tipo, y en otro del otro.
- Supongamos que un día alguien prueba el “Bunnahabhain”.

# K-vecinos más Cercanos

- El consiste en representar cada tipo de Whiskey con un vector.
- Para ésto, definamos los siguientes cinco atributos:

Atributo	Valores	Total de valores
Color	amarillo, pálido, muy pálido...	14
Olor	aromático, dulce, fresco, ...	12
Cuerpo	suave, medio, firme, aceitoso,..	8
Paladar	seco, fructoso, ahumado, ...	15
Terminación	seco, limpio, suave, cálido, ...	19

# K-vecinos más Cercanos

- Bunnahabhain puede ser descrito como:
  - Color: dorado
  - Olor: fresco y mar
  - Cuerpo: firme, medio y liviano
  - Paladar: dulce, fructoso, y limpio
  - Terminación: completa.

Whiskey	Distancia	Descriptores
Bunnahabhain	---	dorado; firme; mediano; liviano; dulce; fruta; ...
Glenglassaugh	0.643	dorado; firme; liviano; suave; dulce; fruta; ...
Tullibardine	0.647	dorado; firme; medianol; liviano; suave; dulce; ...
Ardbeg	0.667	firme; mediano; completo; liviano; seco; ...
Bruichladdich	0.667	pálido; firme; liviano; suave; seco; dulce; ...
Glanmorangie	0.667	dorado; medio; aceitoso; liviano; dulce; picante; ...

# K-vecinos más Cercanos

- Con esa lista podríamos ir a cualquier tienda de Whiskey en Escocia y conseguir el más parecido al Bunnahabhain.
- Esto nos sirve para buscar elementos similares, pero veamos cómo lo podemos usar para clasificar. Consideremos el problema de predecir si un cliente va a responder de manera positiva a una oferta de tarjeta de crédito basado en cómo otros clientes similares han respondido.

# K-vecinos más Cercanos

Cliente	Edad	Ingreso	Tarjetas	Respuesta	Distancia desde David
David	37	50	2	?	0
John	35	35	3	Sí	$\sqrt{((35-37)^2 + (35-50)^2 + (3-2)^2)} = 15.16$
Rachael	22	50	2	No	$\sqrt{((22-37)^2 + (50-50)^2 + (2-2)^2)} = 15$
Ruth	63	200	1	No	$\sqrt{((63-37)^2 + (200-50)^2 + (1-2)^2)} = 152.23$
Jefferson	59	170	1	No	$\sqrt{((59-37)^2 + (170-50)^2 + (1-2)^2)} = 122$
Norah	25	40	4	Sí	$\sqrt{((27-37)^2 + (40-50)^2 + (4-2)^2)} = 15.74$

Hay tres clientes cercanos a David con una distancia de 15. Los otros están mucho más lejanos. Si consideramos los tres vecinos más cercanos tenemos un “no” y dos “si”. El voto de la mayoría indica “si”.

# K-vecinos más Cercanos

- ¿Cuántos vecinos deberíamos usar?
  - Números impares son buenos para romper empates en problemas de dos clases.
  - En general, entre más grande es el “k”, las estimaciones son más suaves (estables).
  - Podemos hacer crecer el “k” lo suficientemente grande hasta utilizar todo el conjunto de datos. Lo que da un promedio con respecto a todo el conjunto de datos (clase mayoritaria).

# K-vecinos más Cercanos

- ¿Deberían tener el mismo peso en la decisión final?
  - Aún cuando estamos confiados en el número de vecinos. Debemos saber si les damos a todos el mismo peso.
  - Recordemos que no todos los vecinos están a la misma distancia.
  - En el ejemplo, si usamos  $k=4$ , David recuperaría dos “si” y dos “no”, siendo la cuarta muy lejana a las otras tres.

# K-vecinos más Cercanos

- ¿Deberían tener el mismo peso en la decisión final?
  - Para incorporar ésto, se utiliza una “votación con pesos” o “votos moderados de acuerdo a la similitud”
  - El aporte de cada vecino es escalado de acuerdo a su similitud.
- Supongamos que utilizamos como similitud el recíproco de la distancia al cuadrado:



Cliente	Distancia	Peso similitud	Contribution	Class
John	15.16	0,00444444	0,344	No
Rachael	15	0,00432825	0,336	Sí
Ruth	152.23	0,00405696	0,312	Sí
Jefferson	122	0,00006718	0,005	No
Norah	15.74	0,00004316	0,003	No

# K-vecinos más Cercanos

- Considerando los aportes de cada uno, las probabilidades finales serían 0.65 “sí”, y 0.35 “no”.
- Utilizar pesos, generalmente, mitiga la importancia de cuántos vecinos utilizar.
- Nótese que k-NN no genera líneas como separaciones entre clases. El tipo de límite depende de las vecindades.
- 1-NN es muy sensible a los outliers. Tiende a hacer mucho overfitting.
- “k” se estima experimentalmente.

# K-vecinos más Cercanos

- Problemas con k-NN:
  - Para algunos dominios, el modelo puede ser interpretable, para otros no. Por un lado es fácil entender como una decisión local es tomada, pero no el modelo global.
    - Esto es útil para sistemas de recomendación como en Amazon, por ejemplo “Otros clientes también han comprado/visto X junto a Y”.
  - Si nos preguntamos ¿Qué tipo de conocimiento puede ser extraído desde los datos de mis clientes que me puedan ayudar a la toma de decisiones?, no tendríamos respuesta porque no hay un modelo explícito.

# K-vecinos más Cercanos

- Es difícil responder esta última pregunta porque el modelo consiste en el conjunto de datos completo.
- Otro problema es la dimensionalidad y el conocimiento del dominio:
  - Se toman en cuenta todos los features cuando se calcula la distancias. Hay que estandarizar, mapear atributos categóricos y contiguos, escalar, etc.
  - Cuando hay muchos atributos, muchos pueden ser irrelevantes para la métrica de similitud.

# K-vecinos más Cercanos

- Para el ejemplo de la tarjeta de crédito, podemos tener mucha información como el número de hijos, largo del período en el trabajo actual, tamaño de la casa, ingreso medio, modelo del auto, nivel educacional, etc. Pero sólo algunos son relevantes para decidir si el cliente va a aceptar o no una tarjeta de crédito.
- Este es el problema de la dimensionalidad de los vectores de features. Se pueden seleccionar atributos:
  - Manualmente.
  - Dado conocimiento previo del problema.

# K-vecinos más Cercanos

- Otra forma: se pueden ajustar pesos a los atributos en la métrica de distancia.
- El entrenamiento es muy rápido ya que involucra almacenar los vectores.
- El gran esfuerzo es al clasificar, ya que la base de datos debe ser consultada para recuperar los vecinos más cercanos y eso puede ser muy costoso.
- Lo cual lo hace indeseable para aplicaciones on-line.
  - Hay estructuras de datos especializadas que hacen el speed-up.

# ¿Se Pueden Mezclar los Clasificadores?

- La respuesta es sí.
- Principalmente, hay dos técnicas de “ensemble”: bagging y boosting.
- En ambos casos, varios clasificadores son construidos y la clasificación final de un caso de prueba es hecha a través de la votación de este “comité de clasificadores”.
  - La mayoría gana.

# Bagging (bootstrap aggregating)

- Entrenamiento:
  - Generar  $k$  muestras bootstrap  $S_1, S_2, \dots, S_k$ . Para ésto, hay que sacar aleatoriamente  $n$  ejemplos del conjunto de entrenamiento. A ésto se le llama replicado bootstrap del set original. En las réplicas, pueden haber algunos ejemplos repetidos muchas veces.
  - Construir un clasificador con cada  $S_k$ . Utilizando el mismo algoritmo base.
- Testing:
  - Clasificar un caso de prueba mediante la etiqueta mayoritaria otorgada por estos  $k$  clasificadores (pesos iguales).

# Bagging (bootstrap aggregating)

- Mejora los resultados sustancialmente cuando el algoritmo base es inestable, es decir, es sensible a pequeños cambios en los datos de entrenamiento.
  - Los árboles de decisión y las redes neuronales son considerados como algoritmos inestables.
  - El k-NN es considerado como estable.
- Algunas veces, para algoritmos estables, como clasificadores bayesianos y k-vecinos más cercanos, bagging puede dañar la performance.
- Normalmente, si los replicados bootstrap son del mismo tamaño que el conjunto original, van a tener cerca del 63% de los datos originales, el resto serán duplicados.

# Bagging (bootstrap aggregating)

- Claramente, la diversidad se logran creando diversos conjuntos de entrenamientos.
- En teoría cada clasificador es entrenado con el mismo número de datos “n”.
- Si los clasificadores fueren independientes, y las salidas también lo fueren, además, si todo tuvieran una precisión “p”, entonces el voto de la mayoría garantiza que mejorará el desempeño.

# Bagging (bootstrap aggregating)

- Pero en la realidad, los ejemplos son pseudo-independientes porque vienen del mismo  $Z$ .
- Podemos decir que los ensembles compuestos por clasificadores de menor correlación son los que tienen mayor diversidad.
- La diversidad no aumenta con el número de clasificadores, pero si el error tiende a disminuir.

# Bagging

- Entrenamiento:
  - Inicializar los parámetros
    - El ensemble  $S = \emptyset$ .
    - $k$  el número de clasificadores a entrenar
  - Para cada  $y=1, \dots, k$ 
    - Toma un sampling bootstrap  $S_y$  de  $Z$ .
    - Construir  $D_y$  utilizando  $S_y$  como entrenamiento.
    - Agregar  $S_y$  a  $S$ .
  - Retornar  $S$
- Clasificación
  - Ejecutar  $S_1, \dots, S_k$  en el ejemplo “ $x$ ”.
  - La clase con el mayor número de votos es escogida como la etiqueta para “ $x$ ”.

# Bagging

- Es una de las formas más simples de generar y combinar clasificadores.
- La respuesta es el voto de la mayoría en el caso de predecir una categoría, y un promedio cuando es un número (regresión).
  - Aunque para el caso de los clasificadores que retornan probabilidades para cada categoría, uno también podría promediar las salidas.
- Cada clasificador es ajustado independiente del otro, y no depende de los datos de entrenamiento.
- Asigna a cada clasificador el mismo peso o importancia.

# Bagging

- Es fácilmente paralelizable.
- ¿Se le puede hacer mejoras?
  - Se puede utilizar diferentes estrategias de sampling, ya sea algorítmicamente como en su tamaño.
  - En vez de asignar pesos uniformes a cada clasificador, es posible estimarlos utilizando regresión lineal.

# Bagging

- Pag. 49 .. Párrafo marcado como “key” y fig. 4.4 y 4.5.

# Boosting

- Boosting también manipula el set de entrenamiento para producir varios clasificadores.
- El más popular es AdaBoost, el cual asigna un peso a cada ejemplo.
- En general, Boosting tiende a funcionar mejor que bagging.
- También tiende a mejorar la performance cuando el algoritmo de aprendizaje base es inestable.

# Boosting

- La idea básica es ubicar **pesos** a un conjunto de técnicas para producir la salida a un evento.
- Esos pesos pueden ser interpretados como la probabilidad de que la estrategia respectiva sea la mejor del grupo.
- La distribución de los pesos es actualizada cada vez que se tiene información del éxito/fracaso de la categorización de uno nuevo.
  - Las estrategias con predicciones correctas reciben más peso, y las incorrectas se reducen.

# Boosting: AdaBoost

- Training:
  - Produce una secuencia de clasificadores (utilizando el mismo algoritmo de aprendizaje).
  - Cada uno es dependiente del anterior, y se enfoca en los errores del anterior.
  - Ejemplos que son mal clasificados por los clasificadores previos se les asignan un peso más alto.
  - Al comienzo el peso de cada elemento de entrenamiento es  $1/n$ .
  - Al final de cada iteración se actualizan y normalizan los pesos y se construye un nuevo clasificador.
- Testing:
  - Se combinan los resultados de los diferentes clasificadores.
  - Al contrario de bagging, acá se le asignan peso a los votos.

# Boosting: AdaBoost

**AdaBoost**( $D, Y$ , BaseLearner,  $k$ )

1. Initialize  $D_1(w_i) \leftarrow 1/n$  for all  $i$ ;  // initialize the weights
2. **for**  $t = 1$  to  $k$  **do**
3.      $f_t \leftarrow \text{BaseLearner}(D_t)$ ; // build a new classifier  $f_t$
4.      $e_t \leftarrow \sum_{i: f_t(D_t(\mathbf{x}_i)) \neq y_i} D_t(w_i)$ ; // compute the error of  $f_t$
5.     **if**  $e_t > 1/2$  **then** // if the error is too large,
6.          $k \leftarrow k - 1$ ;  // remove the iteration and
7.         **exit-loop** // exit
8.     **else**
9.          $\beta_t \leftarrow e_t / (1 - e_t)$ ;
10.          $D_{t+1}(w_i) \leftarrow D_t(w_i) \times \begin{cases} \beta_t & \text{if } f_t(D_t(\mathbf{x}_i)) = y_i \\ 1 & \text{otherwise} \end{cases}$ ; // update the weights
11.          $D_{t+1}(w_i) \leftarrow \frac{D_{t+1}(w_i)}{\sum_{i=1}^n D_{t+1}(w_i)}$  // normalize the weights
12.     **endif**
13. **endfor**
14.  $f_{final}(\mathbf{x}) \leftarrow \operatorname{argmax}_{y \in Y} \sum_{t: f_t(\mathbf{x}) = y} \log \frac{1}{\beta_t}$  // the final output classifier

# Referencias

- “*Web Data Mining: Exploring Hyperlinks, contents, and Usage Data*” by Bing Liu, Second Edition, chapter 3, 2011.
- Teorema de Bayes (visitado 27 Oct. 2012):
  - [http://es.wikipedia.org/wiki/Teorema\\_de\\_Bayes](http://es.wikipedia.org/wiki/Teorema_de_Bayes)
  - [http://www.vitutor.com/pro/2/a\\_17.html](http://www.vitutor.com/pro/2/a_17.html)
- Distribución Multinomial (visitado 27 Oct. 2012):
  - <http://alejandraestadistica.blogspot.com/2010/10/distribucion-multinomial.html>
- [http://en.wikipedia.org/wiki/Oversampling\\_and\\_undersampling\\_in\\_data\\_analysis](http://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis) (visitado 12 de julio 2014)

# Referencias

- “An introduction to ROC analysis”, by Tom Fawcett, In Pattern Recognition Letters 27 (2006) 861–874.
- “Data Science for Business”, Foster Provost and Tom Fawcett, ISBN: 978-1-449-36132-7, 2013.
- “*Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*”, Giovanni Seni and John F. Elder, Morgan and Claypool publishers, ISBN 9782608452842, 2010