Reducción de la Dimensionalidad

Eduardo Morales, Hugo Jair Escalante

INAOE

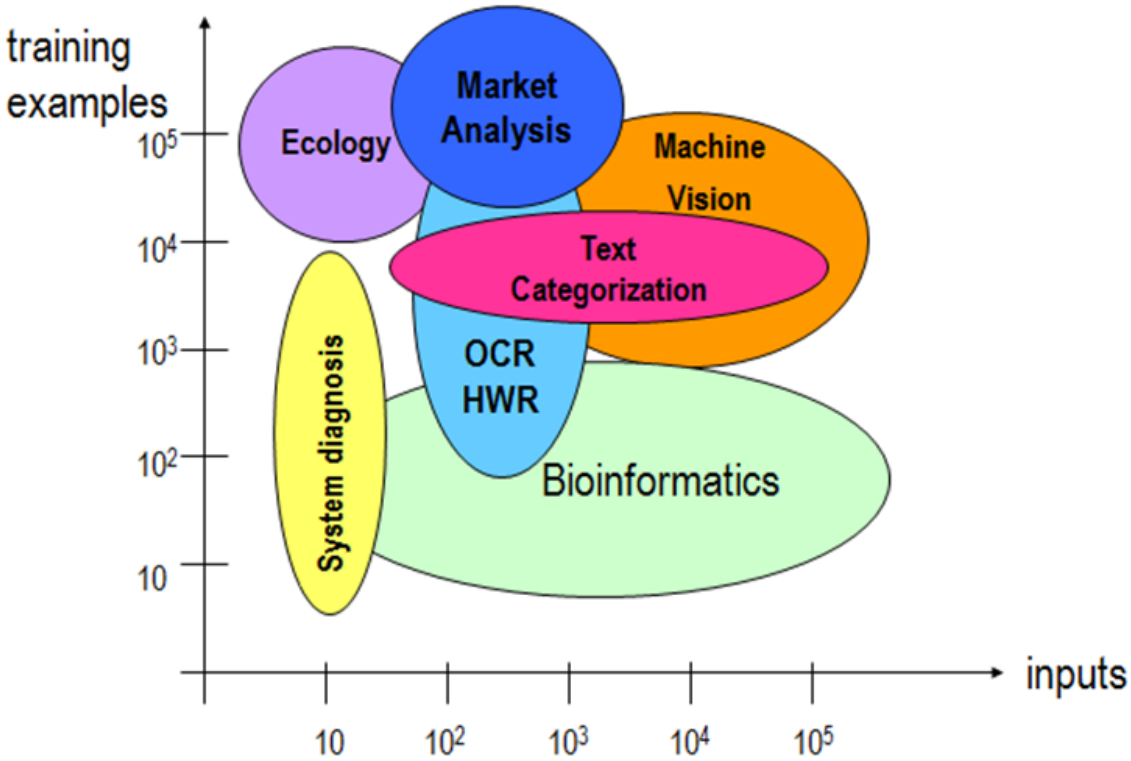
Contenido

Reducción de la Dimensionalidad

En algunos dominios donde aplicamos algoritmos de aprendizaje computacional, nos enfrentamos a conjuntos de datos de alta dimensionalidad
Por ejemplo, en clasificación de textos:

- Los documentos se representan por su BoW, esto es vectores con tantos elementos como palabras hay en el vocabulario
- Para bases de datos con 1,000-10,000 documentos se pueden llegar a tener vocabularios de varias decenas/cientos de miles de elementos

Reducción de la Dimensionalidad



I. Guyon, et al. **Feature Extraction: Foundations and Applications,** Springer 2006.

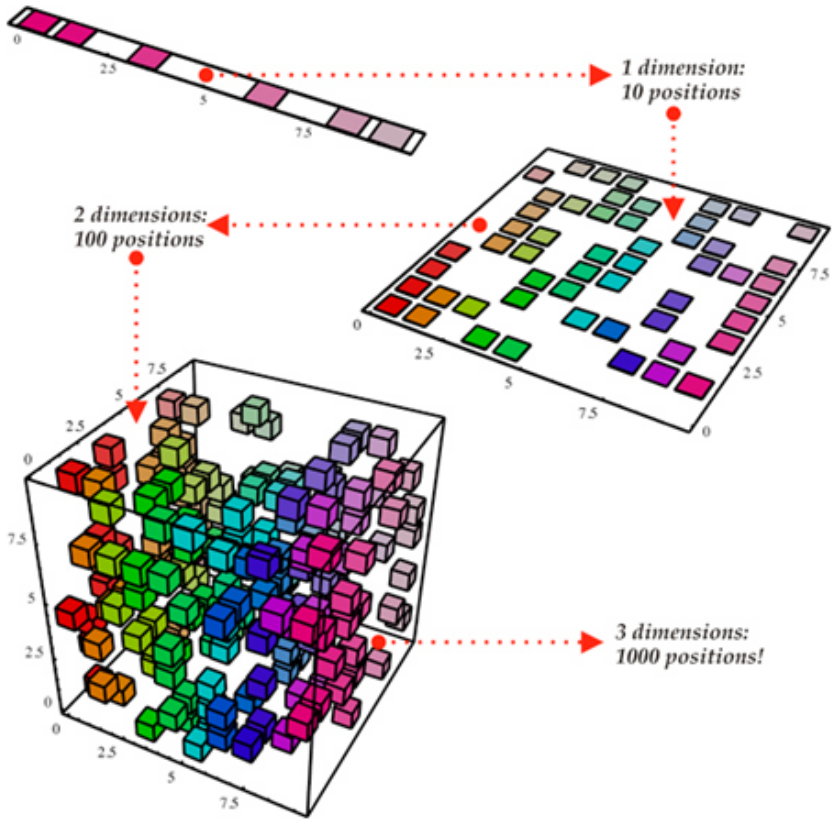
Reducción de la Dimensionalidad

Problemas de la alta dimensionalidad en los datos:

- Costo de procesamiento y almacenamiento
- Atributos relevantes e irrelevantes
- Dificultad para evaluar distancias
- *Data sparsity*
- La maldición de la dimensionalidad

La maldición de la dimensionalidad

- El número de posiciones escala de manera exponencial con la dimensionalidad del problema.
- Necesitamos un número exponencial de ejemplos de entrenamiento para cubrir todas las posiciones



Selección de Atributos

- A partir de los atributos originales selecciona un subconjunto de estos
- La meta es seleccionar el subconjunto S más pequeño de todos los atributos F, tal que $P(C|S) \approx P(C|F)$
- Ventajas esperadas:
  1. Mejorar el desempeño predictivo
  2. Construir modelos más eficientemente
  3. Mejorar entendimiento sobre los modelos generados

Selección de Atributos

- O sea, seleccionar el subconjunto más pequeño de atributos tal que no se afecte significativamente el porcentage de clasificación y que la distribución de clases resultante sea lo más parecido posible a la original
- Un atributo es *irrelevante* si no afecta de ninguna forma al concepto meta
- Un atributo es *redundante* si no añade nada nuevo al concepto meta
- Un atributo se considera *relevante* si no es irrelevante o redundante.

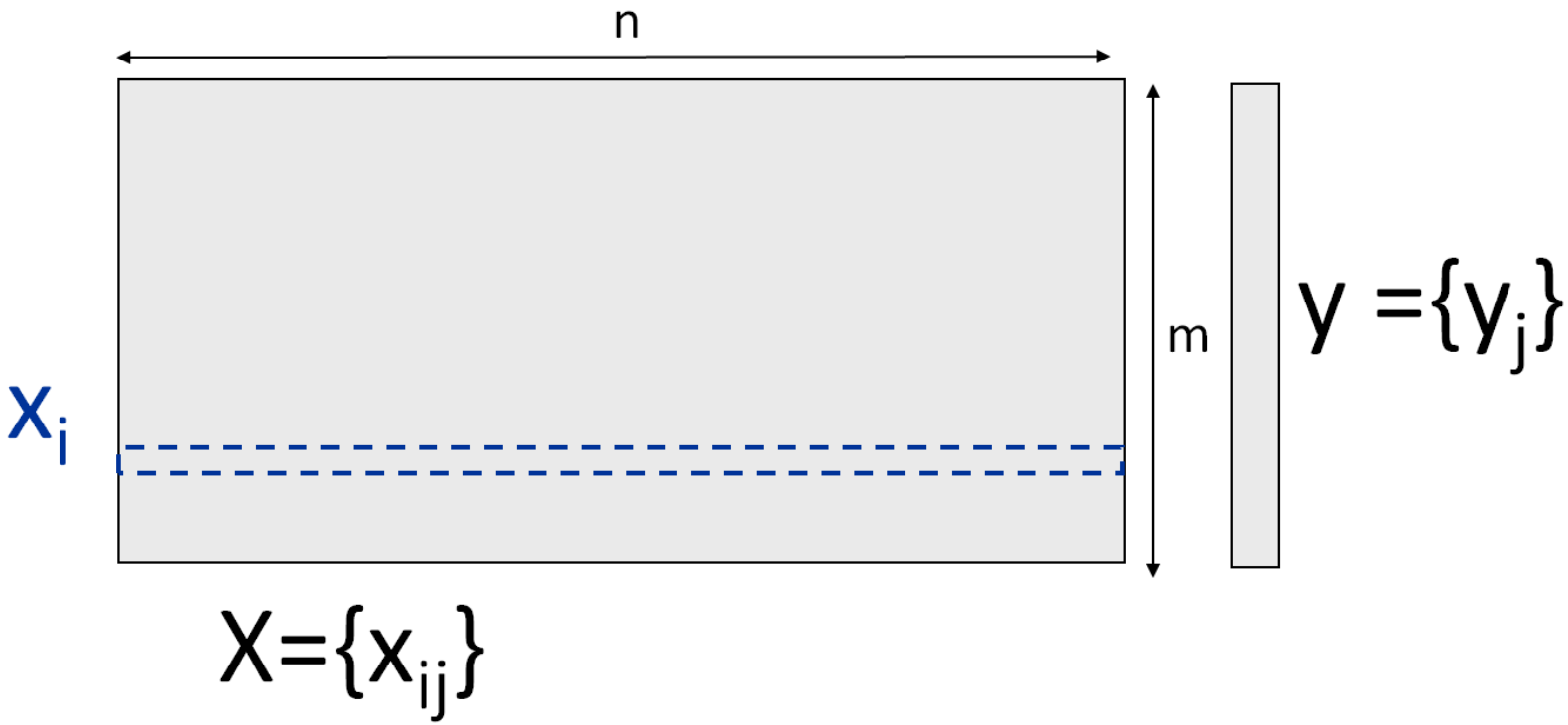
Reducción de la dimensionalidad

Dos soluciones principales:

- **Selección de atributos:** Seleccionar un subconjunto de atributos
- **Generación de atributos (re-parametrización/transformación):** Mapear los atributos originales a un nuevo espacio de menor dimensión

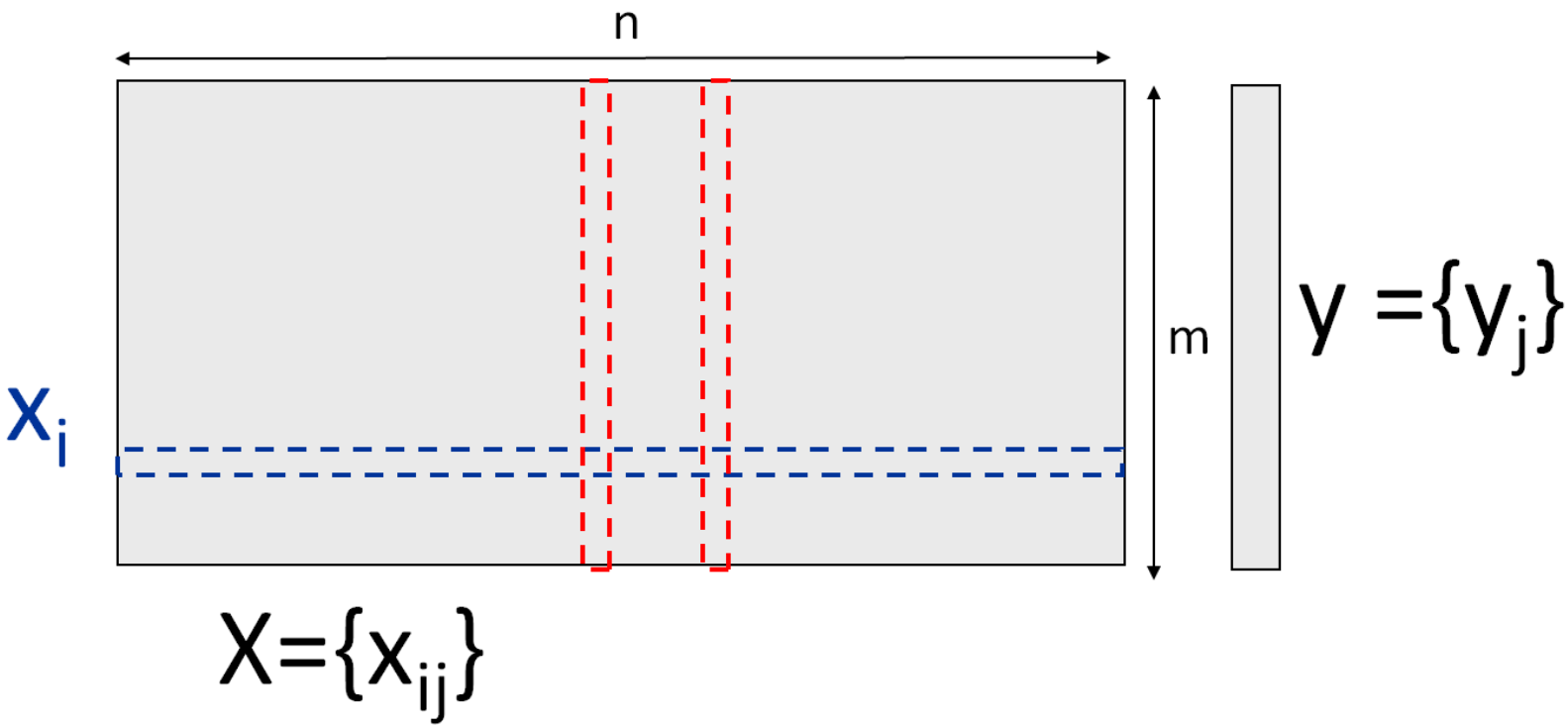
Selección de atributos

- **Problema:** Encontrar un subconjunto de atributos que sean más útiles para la clasificación
- **Objetivos:** Eliminar atributos irrelevantes/ruidosos; seleccionar/mantener atributos relevantes; reducir la dimensionalidad



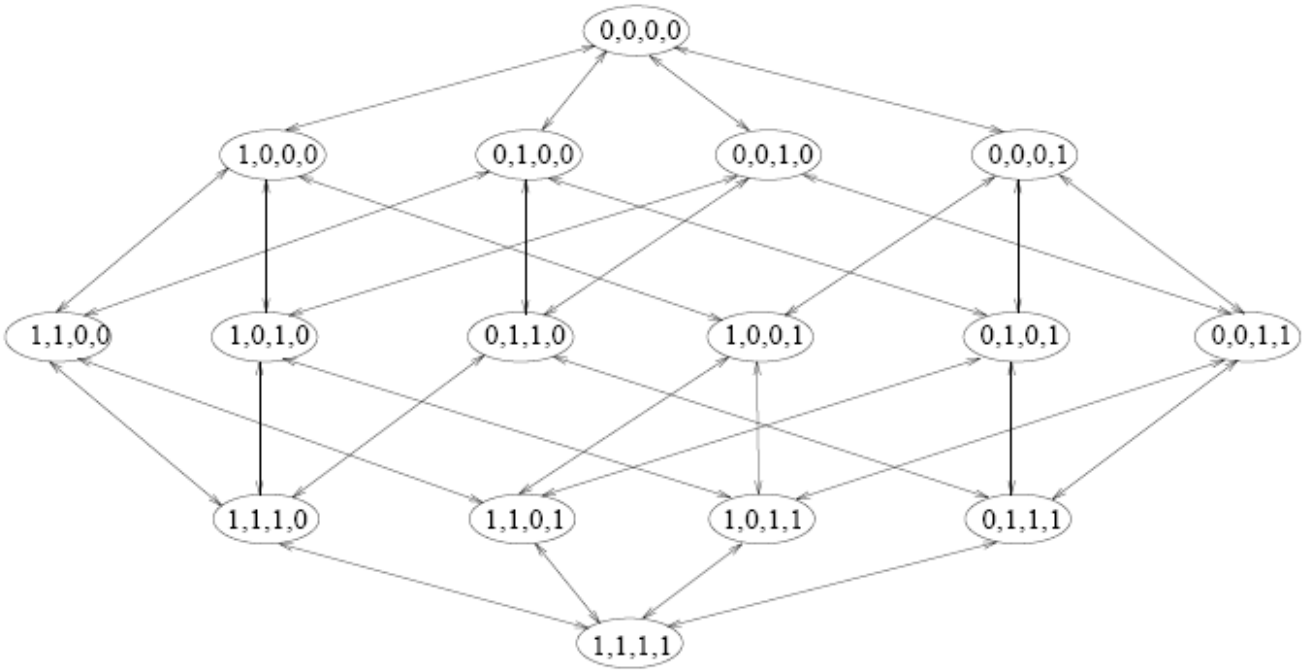
Selección de atributos

- **Problema:** Encontrar un subconjunto de atributos que son más útiles para clasificación.
- **Objetivos:** Eliminar atributos irrelevantes/ruidosos; seleccionar/mantener atributos relevantes; reducir la dimensionalidad.



Selección de atributos

Para un problema con $d-$atributos, cuántos subconjuntos existen?



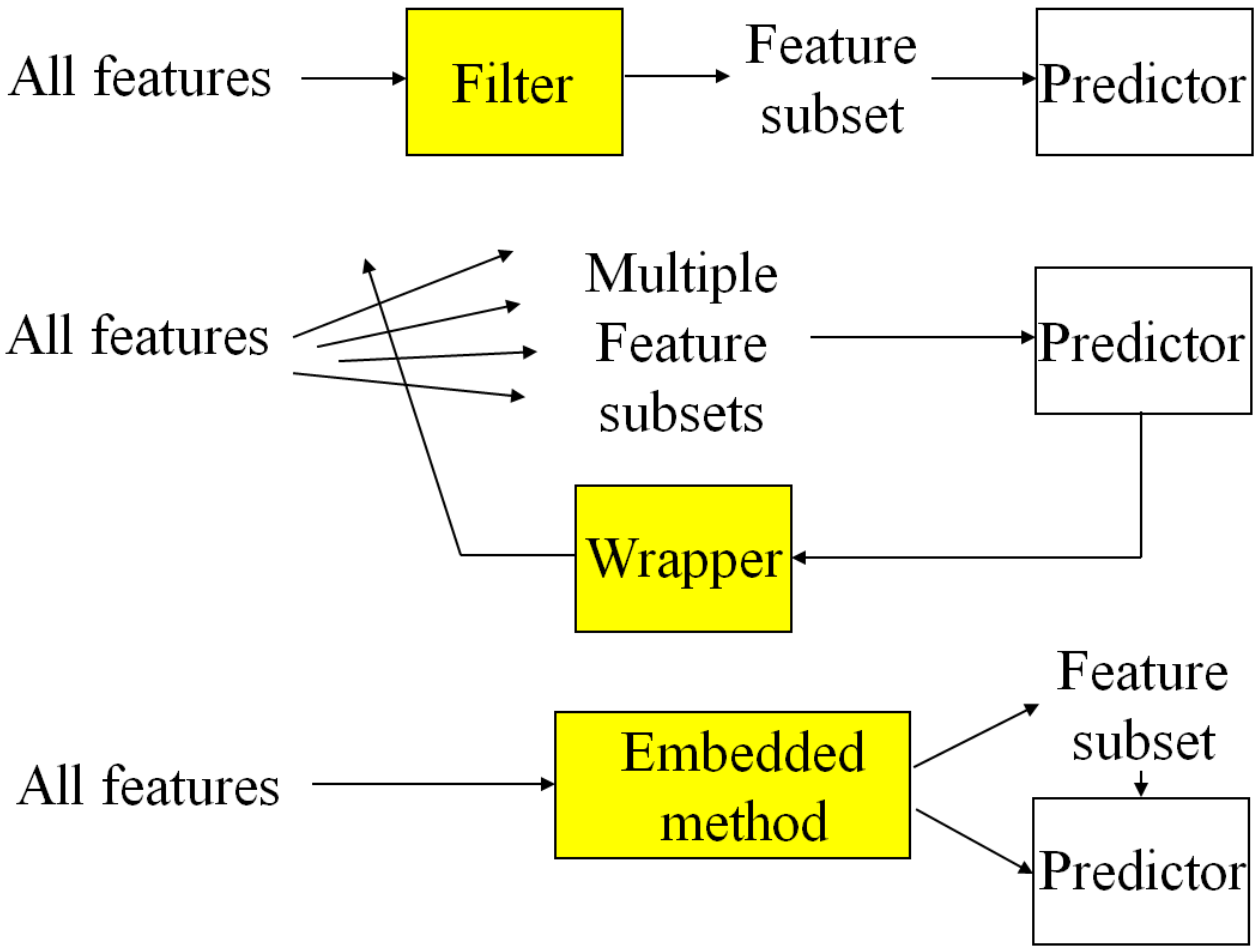
Problema computacionalmente costoso!

Selección de atributos

Tres enfoques principales:

1. **Filtros:** Evalúan la importancia de los atributos usando métodos que son independientes del modelo de clasificación
2. **Wrappers:** Evalúan la importancia de subconjuntos de atributos usando un modelo de clasificación (se adopta una estrategia de búsqueda)
3. **Embedded:** Toman ventaja de la naturaleza del modelo de clasificación considerado

Selección de atributos



Selección de atributos - Isabelle Guyon

Filters

Methods:

- Criterion: Measure feature/feature subset "relevance"
- Search: Usually order features (individual feature ranking or nested subsets of features)
- Assessment: Use statistical tests

Results:

- Are (relatively) robust against overfitting
- May fail to select the most "useful" features

Selección de atributos - Isabelle Guyon

Wrappers

Methods:

- Criterion: A risk functional
- Search: Search the space of feature subsets
- Assessment: Use cross-validation

Results:

- Can in principle find the most “useful” features, but
- Are prone to overfitting

Selección de atributos - Isabelle Guyon

Embedded Methods

Methods:

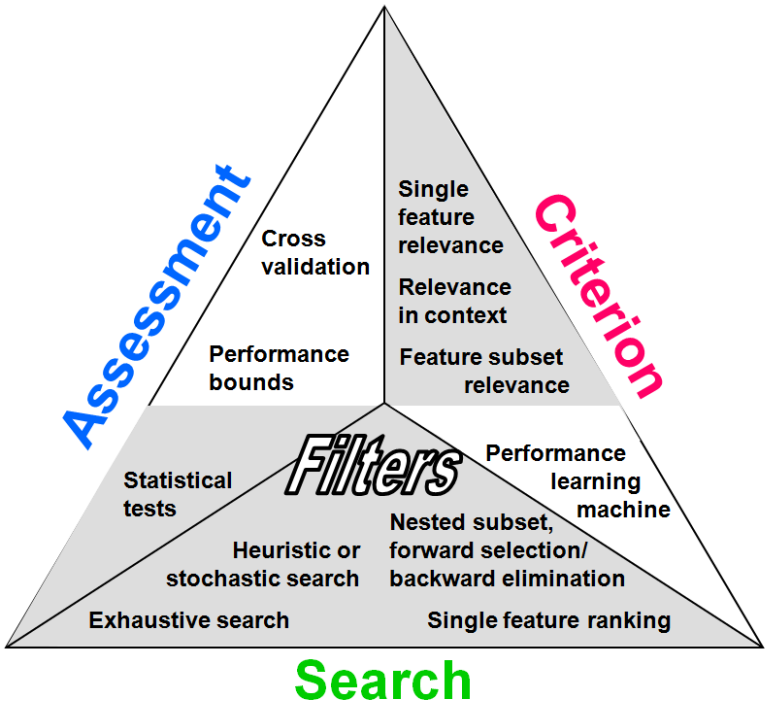
- Criterion: A risk functional
- Search: **Search guided by the learning process**
- Assessment: Use cross-validation

Results:

- Similar to wrappers, but
- Less computationally expensive
- Less prone to overfitting

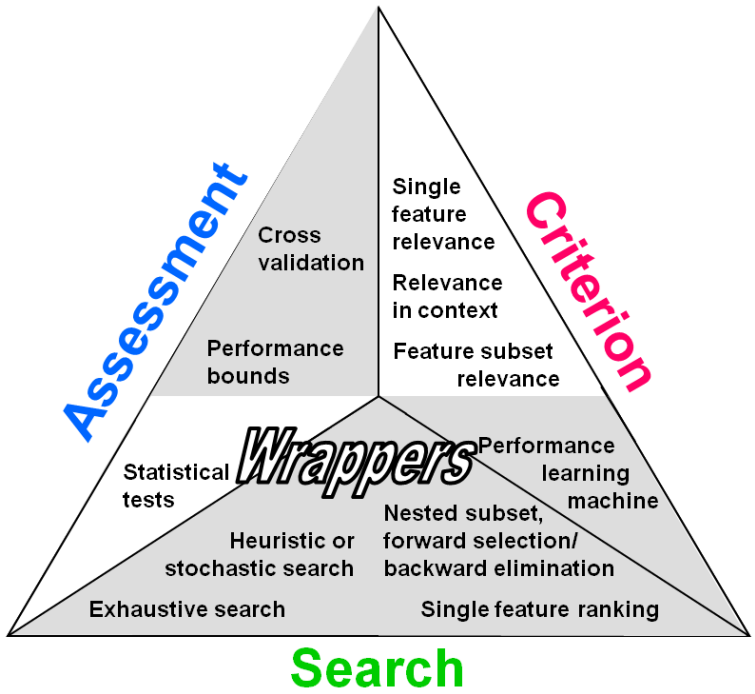
Selección de atributos - Isabelle Guyon

Three "Ingredients"



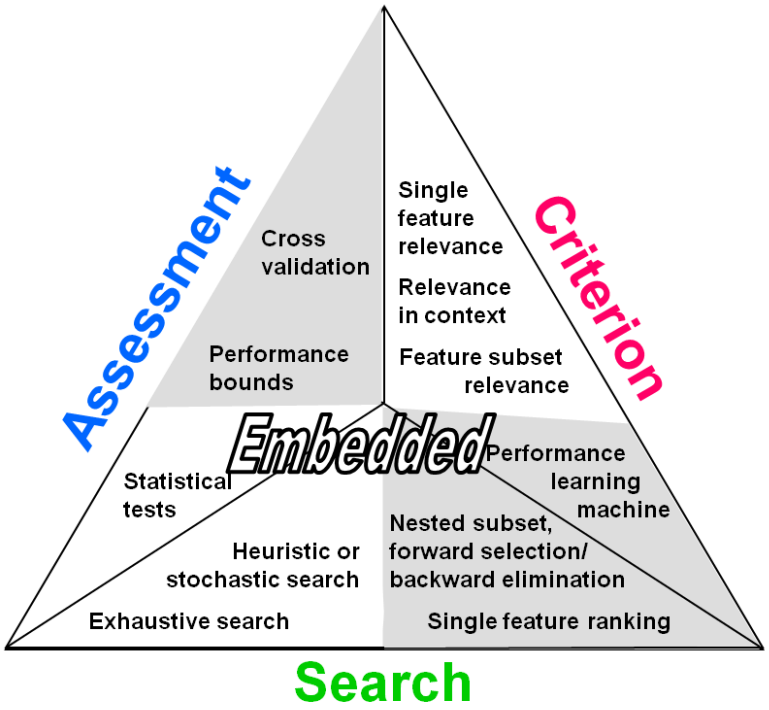
Selección de atributos - Isabelle Guyon

Three "Ingredients"



Selección de atributos - Isabelle Guyon

Three "Ingredients"



Algoritmo de Selección de Atributos

Sea: D datos de entrenamiento, S_0
$S_{mejor} := S_0$
$Ev_{mejor} = eval(S_{mejor}, D, M)$ {evalua S usando M}
while NOT criterio de paro **do**
  $S_{nvo} = genera(S_{mejor}, D)$
  $Ev_{nvo} = eval(S_{nvo}, D, M)$
  if (Ev_{nvo} es mejor que Ev_{mejor}) **then**
    $Ev_{mejor} := Ev_{nvo}$
    $S_{mejor} := S_{nvo}$
  end if
end while
Regresa S_{mejor}

Si M - evalua atributos independientemente del algoritmo de aprendizaje, entonces tenemos un *filter* y si M es un algoritmo de aprendizaje, tenemos un *wrapper*

Selección de Atributos

- Cambiando la forma de generar nuevos candidatos (estrategia de búsqueda) y el criterio de paro, podemos generar diferentes versiones
- Diferentes criterios de evaluación (independientes del algoritmo de aprendizaje) nos generan diferentes algoritmos *filter*
- Diferentes algoritmos de aprendizaje nos generan diferentes algoritmos *wrapper*

También hay algoritmos híbridos

La idea general del algoritmo híbrido es la siguiente:

- Empieza con un subconjunto S_0 (por ejemplo vacío si se usa *forward-selection*)
- Aumenta la cardinalidad (1 atributo) y evalúa todos los elementos de esa cardinalidad con una medida independiente del algoritmo de aprendizaje
- El mejor subconjunto se evalua con un algoritmo de aprendizaje y se queda con el mejor (de todas las cardinalidades vistas) hasta el momento
- Continua con la siguiente cardinalidad

La no mejora en la calidad del algoritmo de aprendizaje puede usarse como criterio de paro.

Estrategias de Búsqueda

- La generación de subconjuntos involucra una estrategia de búsqueda. Para N atributos existen 2^N subconjuntos, por lo que se requiere de una buena estrategia de búsqueda.
- Para esto se requiere especificar el punto inicial de búsqueda. Si se empieza con un conjunto de atributos vacío y se van añadiendo (*forward selection*), si se empiezan con todos los atributos y se van eliminando (*backward elimination*) o si se añaden y quitan atributos (*bi-direccional*).
- El punto inicial también puede ser un conjunto de atributos aleatorio.

Estrategias de Búsqueda

- Dado un criterio de evaluación, se pueden realizar una gran cantidad de estrategias de búsqueda. Por ejemplo, *best-first*, A^*, *beam-search*, *hill climbing*, etc.
- Dentro de estas estrategias existen métodos basados en *branch-&-bound* para no considerar todos los subconjuntos. Sin embargo, en general requiere que se especifique de entrada un número de atributos a seleccionar.
- En general, y dada la gran cantidad de posibles subconjuntos, se prefiere usar una estrategia *greedy* o *hill-climbing*.

Estrategias de Búsqueda

En general se tienen tres estrategias de búsqueda:

- *forward selection*: se van incorporando progresivamente las variables. Es computacionalmente más eficiente, pero tiende a producir peores subconjuntos, ya que se toman decisiones locales.
- *backward selection*: se van eliminando progresivamente las variables. Es computacionalmente más caro pero puede considerar variables débiles individualmente, pero fuertes cuando se consideran en conjunto.
- *Bi-directional selection*: Se pueden añadir o eliminar atributos partiendo de un subconjunto inicial. Una alternativa es añadir (o quitar) p atributos en cada paso y eliminar (añadir) q atributos en el siguiente paso ($p > q$).

Estrategias de Búsqueda

- También se puede realizar búsqueda aleatoria, con un punto inicial generado aleatoriamente y continuando desde ahí con una estrategia *greedy* repitiendo el proceso varias veces (*random restart hill-climbing*), se puede introducir aleatoriedad a la búsqueda (e.g., *simulated annealing*), etc.
- Se puede usar diversas variantes de búsqueda y optimización como *local search*, *tabu search*, *ant colony optimization*, algoritmos genéticos, *swarm optimization*, etc.
- En general, podemos dividir a las estrategias de generación de candidatos en: (i) completas (completa no necesariamente implica que sea exhaustiva), (ii) heurísticas, y (iii) aleatorias.

Evaluación de Subconjuntos

- A grandes rasgos depende de si se usa un criterio independiente o no de un algoritmo de aprendizaje.
- Los independientes son los filtros. Aquí se puede usar una medida de distancia, basada en información, en dependencia o en consistencia.
- Los dependientes son los *wrappers* y usan el desempeño de un algoritmo de aprendizaje para evaluar un subconjunto de atributos.
- En el caso de clustering, se trata de evaluar la calidad de los clusters, por ejemplo, con base en que tan compactos son los grupos o que tan separados están.

Evaluación de Subconjuntos

- Por consideraciones de eficiencia, escalabilidad, simplicidad y buenos resultados empíricos, muchas de las medidas utilizadas independientemente del algoritmo de aprendizaje se aplican a un solo atributo a la vez.
- Estos métodos no pueden capturar combinaciones que podrían dar buenos resultados. Por lo que una variable que aparentemente no sirve por sí sola, puede dar resultados muy buenos en combinación con otras.
- Inclusive dos variables que no aportan nada por separado pueden ser útiles juntas.

Evaluación de Subconjuntos

- Los que evaluan un atributo a la vez pueden eliminar atributos irrelevantes, pero no los redundantes, ya que tienen una evaluación parecida a otros
- Son rápidos y producen una lista ordenada de atributos de acuerdo a su medida de evaluación o *ranking*
- Una vez producida la lista, se tiene que especificar donde cortar (o hasta qué atributo considerar) y para esto existen diferentes opciones

Selección de Atributos *Rankeados*

- Especificar un número *N* y quedarse con los primeros *N* atributos
- Especificar un umbral y si la medida de desempeño es menor que el umbral, entonces descartar, por ejemplo, que los que están alejados dos varianzas de la media de la medida de desempeño.
- Introducir variables aleatorias (1-3) y eliminar los atributos que reciban valores por debajo de la(s) variable(s) aleatoria(s)

Efecto de Introducir variables Aleatorias

Distancia Euclidea	Distancia Matusita	Kullback-Leibler 1	Kullback-Leibler 2	Entropía Shannon	Bhatta-charyya	Relief	OneR	Chi-Square
fctura	fctura	fctura	fctura	kwh	kwEen	anio	factra	factra
mes	kwEen	status	mes	enrgia	fctura	mes	status	status
cIMcC	kwMen	kwEen	status	total	kwMen	factra	anio	mes
anio	RAND3	cCEto	cgInst	tarifa	RAND3	digito	tarifa	kwEen
RAND3	status	RAND3	cgCont	cgInst	toMkw	RAND3	digito	kwMcI
tarifa	cCEto	kwMen	cCEto	cgCont	toMcI	RAND2	mes	kwh
digito	toMkw	toMkw	cIMcC	kwEen	enrgia	RAND1	cIMcC	toMcI
status	enrgia	enrgia	anio	toMcI	cCEto	status	cgCont	toMcC
RAND2	toMcI	kwMcI	kwEen	kwMen	total	cgInst	cgInst	total
cIEen	total	toMcI	RAND2	toMen	RAND2	tarifa	RAND1	toMen
cgInst	kwMcI	RAND2	RAND3	toMkw	kwMcI	cgCont	toMkw	enrgia
cgCont	RAND2	kwh	toMkw	kwMcI	toMcC	cCEto	RAND2	kwMen
RAND1	kwh	total	kwh	toMcC	RAND1	cIEen	cCEto	toMkw
cCEto	toMcC	mes	kwMen	cCEto	kwh	cIMcC	kwh	cCEto
toMcC	RAND1	toMcC	cIEen	cIEen	toMen	kwEen	toMcI	cIEen
kwMcI	toMen	RAND1	enrgia	status	mes	toMen	RAND3	cgInst
toMkw	mes	cIEen	total	cIMcC	status	total	total	cgCont
toMen	cIEen	toMen	kwMcI	anio	cIEen	toMcC	toMen	anio
kwMen	cgInst	cgInst	RAND1	RAND2	cgInst	toMcI	kwEen	cIMcC
toMcI	cgCont	cgCont	tarifa	RAND1	cgCont	kwh	enrgia	tarifa
kwEen	cIMcC	cIMcC	digito	RAND3	cIMcC	kwMcI	kwMen	RAND2
total	anio	anio	toMcC	digito	anio	kwMen	cIEen	RAND3
enrgia	tarifa	tarifa	toMcI	mes	tarifa	toMkw	kwMcI	RAND1
kwh	digito	digito	toMen	fctura	digito	enrgia	toMcC	digito

Selección de Atributos por Subconjuntos

- En lugar de evaluar atributos individuales y ordenarlos, en general es deseable considerar subconjuntos de atributos. Esto elimina la suposición genérica en la evaluación individual de que los atributos son independientes entre si dada la clase.
- Estos algoritmos en general pueden eliminar tanto atributos irrelevantes como redundantes, sin embargo, son en general poco eficientes.

Medidas de Evaluación

- Existe una gran cantidad de medidas de evaluación de atributos (e.g., Euclideana, distancia Matusita, Kullback-Leibler 1 y 2, basada en entropía, Bhattacharyya, Relief, OneR, Chi-Square, etc.) pero sólo vamos a revisar algunas de ellas.
- Se usan para evaluar atributos individuales, aunque muchas de ellas se pueden extender para evaluar sobconjuntos, aunque implica más cálculos y a veces más datos para obtener mejores estimaciones
- Esto de nuevo puede crear una lista ordenada, pero ahora de subconjuntos. La idea es quedarse con el mejor subconjunto.

Medidas de Evaluación

Podemos clasificar las medidas en los siguientes tipos:

- *Medidas de distancia*: para un problema de dos clases, un atributo X es mejor que uno Y si X genera más diferencia entre las probabilidades condicionales de las clases que Y. Aquí podemos incluir muchas de las medidas propuestas, como la Euclideana, entre otras.
- Medidas de información: La ganancia de información del atributo X la podemos definir como la diferencia entre la incertidumbre previa y la posterior al usar X. El atributo X es mejor que el Y si X tiene más ganancia de información que Y.

Medidas de Evaluación

- Kullback-Leibler o entropía cruzada estima la información mutua entre cada variable y la clase

$$I(i) = \int_{x_i} \int_{y} p(x_i, y) log \frac{p(x_i, y)}{p(x_i)p(y)} dxdy$$

Que para variables discretas se escribe como:

$$I(i) = \sum_{x_i} \sum_{y} P(X = x_i, Y = y) log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)}$$

y estimar las probabilidades con base en frecuencia de los datos, lo cual se complica con muchas variables y clases.

Medidas de Evaluación

- Medidas de dependencia: miden dependencia o correlación entre variables. Si la correlación del atributo X con la clase C es mayor que la del atributo Y con la clase C, entonces X es mejor que Y
- Una variante es determinar que tan correlacionado está un atributo con otros para determinar redundancia. La idea es ver qué atributos están correlacionados fuertemente con la clase. El coeficiente de correlación está dado por:

$$R(i) = \frac{cov(X_i, Y)}{\sqrt{var(X_i)var(Y)}}$$

Medidas de Evaluación

- Esto se puede estimar como:

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \overline{x_i})(y_k - \overline{y})}{\sqrt{\sum_{k+1}^{m}(x_{k,i} - \overline{x_i})^2 \sum_{k+1}^{m}(y_k - \overline{y})^2}}$$

  Muchas veces se usa $R(i)^2$ porque representa qué tan bien se ajusta linealmente una variable individual con respecto a la clase (y).
- Esto se puede extender para cualquier número de clases.
- El coeficiente de correlación sólo detecta dependencias lineales entre una variable y la clase. Una alternativa es usar ajustes no lineales.

Medidas de Evaluación

- Medidas de consistencia: miden la consistencia de las hipótesis con respecto a un grupo de atributos, buscando el mínimo conjunto de atributos que genere hipótesis consistentes.
- Medidas basadas en error: usan un algoritmo de clasificación como medida de evaluación (*wrapper*).

Con estas medidas y diferentes estrategias para generar atribitutos (i.e., completos, heurísticos, y aleatorios), podemos formar una gran cantidad de posibles algoritmos.

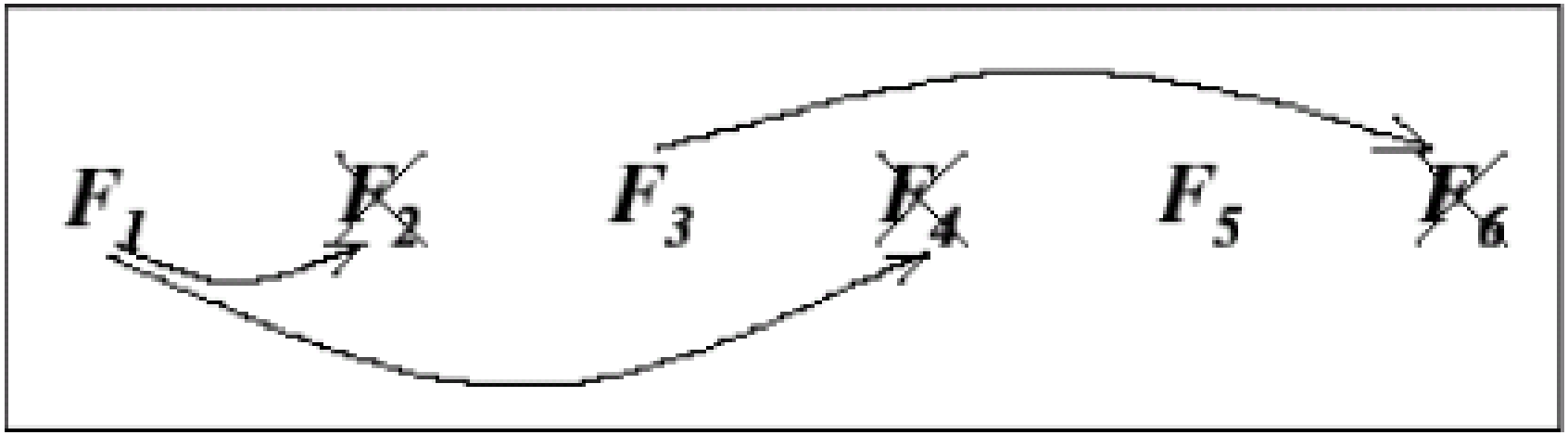
Algunos Algoritmos

- SOAP: opera sobre atributos numéricos. No emplea distancias ni cálculos estadísticos y tiene un bajo costo. Idea: contar cuantas veces el valor de la clase cambia con respecto al atributo ordenado en forma ascendente.
- SFS/SBS (*Sequential Forward/Backward Selection*): empieza con el cojunto vacío/completo y en cada iteración añade/quita un atributo seleccionado con una función de evaluación.

Algunos Algoritmos

- SFFS/SFBS (Sequential Floating Forward/Backward Selection): aplica, después de cada paso *forward*/*backward*, tantos pasos *backward*/*forward* como se pueda, mientras el subconjunto de atributos sea mejor que los evaluados previamente en ese nivel. Si empieza con el conjunto vacío añade más atributos y si empieza con el conjunto completo, elimina más atributos.
- Liu (2004) encuentra atributos fuertemente relevantes (si se eliminan siempre afectan la calidad predictiva), débilmente relevantes (pueden o no afectar la calidad al ser eliminados), los cuales pueden ser o no redundantes, y los irrelevantes. Los redundantes los eliminan calculando la cobija de Markov

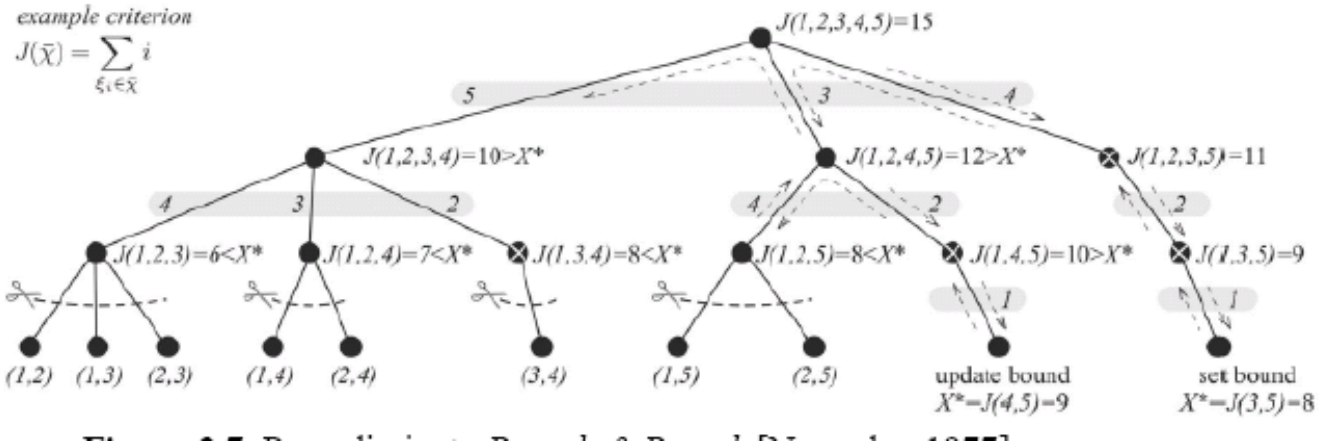
Un Esquema para Eliminar Atributos Redundantes



Algunos Algoritmos

- Branch & Bound (Narendra): Búsqueda completa y evaluación basada en distancia. Define un tamaño de profundidad (i.e., número de atributos) y usa una medida sobre los atributos para definir criterios de corte. Sigue un proceso de *backward elimination*
- Requiere que la medida de evaluación sea monotónica y en principio requiere saber el número de atributos meta.
- Se han hecho extensiones (e.g., Somol) usando estimaciones

Ejemplo de Branch & Bound



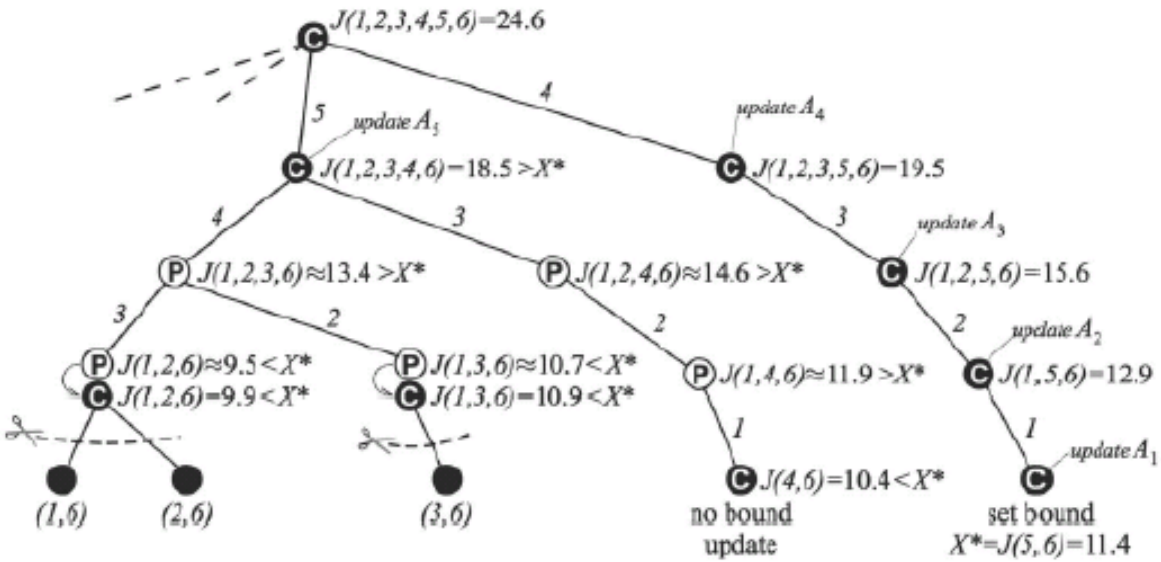
Ejemplo de Branch & Bound con Estimaciones

example criterion

$$J(\bar{\chi}) = \frac{|\bar{\chi}|^2}{10} + \sum_{\ell_i \in \bar{\chi}} i$$

Ⓒ - *computed value*
Ⓟ - *predicted value*

	estimated prediction info
A_1	1.5
A_2	2.7
A_3	3.9
A_4	5.1
A_5	6.1
A_6	n/a



Algunos Algoritmos

- Clasificadores con una sola variable: Una posibilidad es usar una sola variable para tratar de clasificar (*decision stump*) y ordenar las variables dependiendo de que tan bien clasifican. Se puede medir el error con varios criterios, basada en falsos positivos, falsos negativos, curvas ROC, etc.
- Basados en selección y estrategias de algoritmos existentes. Una posibilidad es usar un algoritmo de árboles (e.g., C4.5) y quedarse con los atributos que aparecen en un árbol podado.

Algunos Algoritmos

- Heurísticos basados en distancia, por ejemplo, *Relief/Relieff*: Dado un ejemplo, Relief busca a sus dos vecinos más cercanos, uno de la misma clase y otro de una clase diferente, y actualiza los pesos de los atributos involucrados dependiendo de si sus valores son iguales o no a estos ejemplos

Algotitmo de Relief

for $i = 1$ to n **do**
  Selecciona aleatoriamente un ejemplo E de una clase
  Encuentra el ejemplo de la misma clase más cercano P
  y el ejemplo de otra clase más cercano N
  for $A := 1$ to *Num. de atributos* **do**
    $W[A] := W[A] - \text{diff}(A, E, P)/n + \text{diff}(A, E, N)/n$
  end for
end for

Relief

- diff(Atr, $Inst1$, $Inst2$) calcula la diferencia entre los valores del atributo Atr de dos ejemplos. Para valores discretos la diferencia es 1 si son diferentes y 0 si son iguales. Para atributos continuos se puede normalizar y tomar un valor continuo entre 0 y 1
- La idea es favorecer atributos que tengan valores diferentes en ejemplos parecidos de diferente clase y valores iguales en ejemplos parecidos de la misma clase, que se puede interpretar como:

$$W[A] = P(\text{dif. valor de A} \mid \text{instancia de clase dif.}) - P(\text{dif. valor de A} \mid \text{instancia de misma clase})$$

Extensiones a Relief (Relieff)

1. En Relieff, el proceso se repite un número de veces igual al número total de ejemplos
2. En lugar de tomar el ejemplo positivo y negativo más cercano, se toman los k ejemplos positivos y negativos más cercanos y se promedia el resultado (e.g., $k = 10$).
3. Para N clases, busca los k ejemplos más cercanos para cada una de las clases multiplicado por la probabilidad de ocurrencia de cada clase.

Extensiones a Relief (Relieff)

- Para actualizar los pesos se usan probabilidades $P(X|Y)$ calculadas usando un estimador Laplaciano.

$$P(X|Y) = \frac{N(X \wedge Y) + mP_a(X)}{N(Y) + m}$$

Donde:

- $N(Z) =$ número de ejemplos con resultado Z,
- $P_a(X) = \frac{N(X)+1}{N+Num.posiblesresultados}$
- m es un parámetro relacionado a la cantidad de ruido ($m = 2$ en las publicaciones originales)

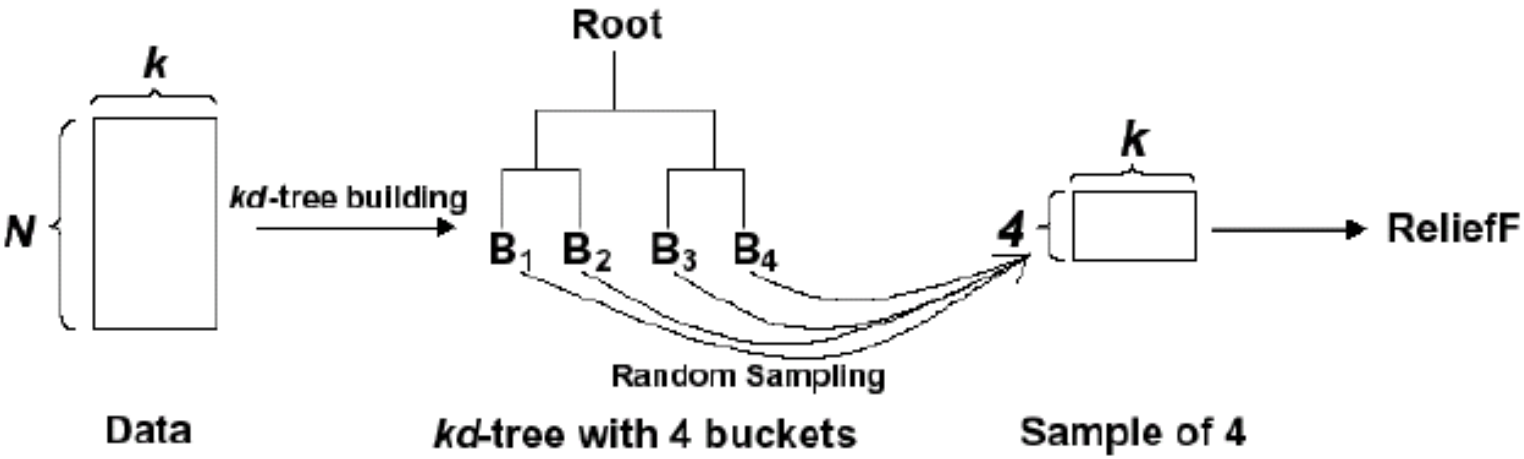
Algunos Algoritmos

- Basados en dependencias entre variables: Otra posibilidad es usar filtros basados en cobijas de Markov. Una cobija de Markov de una variable x_i es un conjunto de variables, sin incluir a x_i que hacen a la variable "innecesaria". Una vez que se encuentra la cobija de Markov se puede eliminar esa variable.
- Basados en información completos: usando el principio de descripción mínima (MDL), en donde se utiliza una expresión que se interpreta como el número de bits necesarios para transmitir la clase de las instancias, los parámetros óptimos, los atributos relevantes y los irrelevantes.

Algunos Algoritmos

- Combinando con *progressive sampling*: se usa un conjunto pequeño de datos y se corre C4.5. Se incremente el número de datos y se repite el proceso. Se comparan los atributos seleccionados y se evaluan si los atributos diferentes son relevantes o no.
- Combinado con *active learning*: la idea es seleccionar instancias que pueden ser relevantes para realizar la selección de atributos. Las instancias se agrupan usando *kd-trees* y luego se seleccionan *n* instancias de cada grupo y se alimenta a relieff

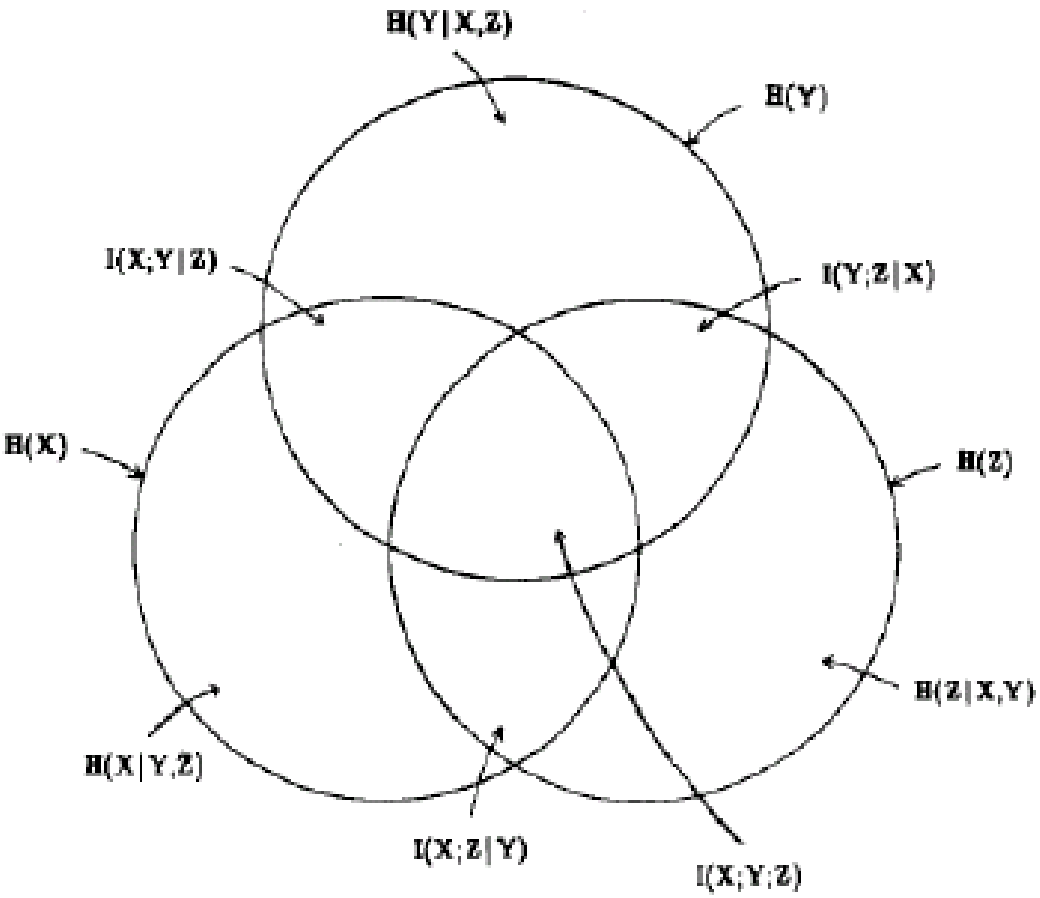
Combinando con *active sampling*



Interacción entre Atributos

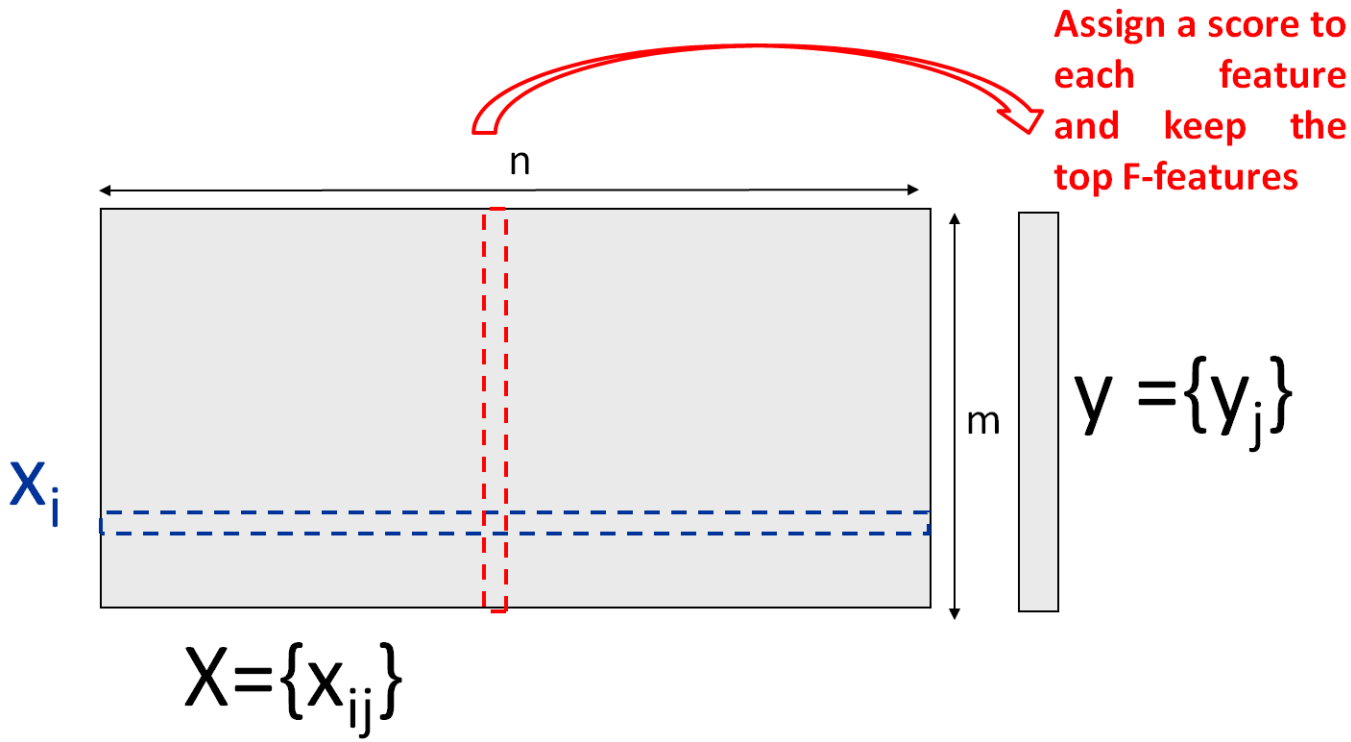
- Interacción entre atributos: Por ejemplo, si queremos ver la interacción entre 4 atributos usando medidas de información (Jakulin):
$I(X;Y;Z;C) = I(X,Y,Z;C) - I(X,Y;C) - I(Y,Z;C) - I(X,Z;C) + I(X;C) + I(Y;C) + I(Z;C)$ y variantes (nwG)

Esquema de dependencias de información entre atributos



Filtros para selección de atributos

Idea: Asignar un valor a cada uno de los atributos, y seleccionar los mejores (algunos métodos son multi-variables)



Slide taken from I. Guyon. **Feature and Model Selection.** Machine Learning Summer School, Ile de Re, France, 2008.

Filtros para selección de atributos

Correlación: Forma sencilla de calcular la correlación lineal entre dos variables. $\rho(X_i, Y)$ Nos indica la correlación entre el atributo X_i y la clase Y

$$\rho(X_i, Y) = \frac{cov(X_i, Y)}{\sigma_{X_i}, \sigma_Y}$$

con la covarianza (cov):

$$cov(X_i, Y) = E[(X_i - E[X])(Y - E[Y])]$$

y la desviación estándar (σ):

$$\sigma_W = \sqrt{E([W - E[W]]^2)}$$

Relación con información mutua.

Correlación

- Para estudiar la relación lineal entre dos variables contínuas

$$Cov_{muestral} = Cov(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{N - 1}$$

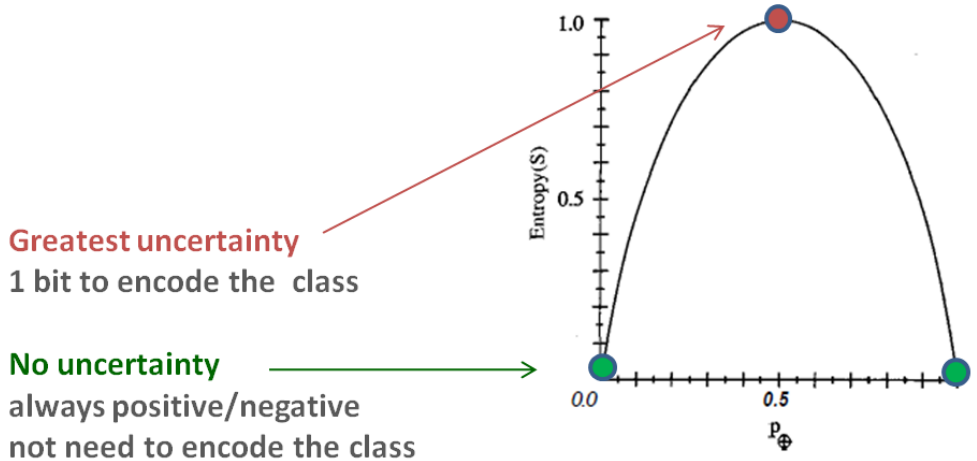
- Existen diferentes medidas
  - Pearson: Variables con distribución normal
  - Spearman: Datos ordinales o de intervalo
  - Kendall: Pocos datos

Filtros para selección de atributos

Ganancia de información (IG): Mide qué tan bien un atributo separa los ejemplos de entrenamiento en dos clases

Se basa en el concepto de **entropía**, que caracteriza la impureza de una colección arbitraria de ejemplos.

- La entropía específica el número mínimo de bits de información necesaria para codificar un elemento arbitrario del conjunto de datos.



Filtros para selección de atributos

Ganancia de información (IG). Mide qué tan bien un atributo separa los ejemplos de entrenamiento
Comunmente se mantienen aquellos atributos con IG mayor a 0.

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

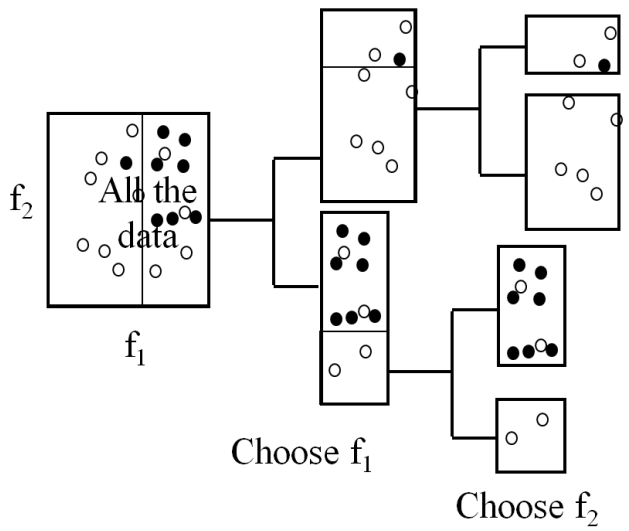
$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$$

$$Ganancia(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - E(A)$$

La IG de un atributo mide la reducción esperada de la entropía causada por la partición de ejemplos usando dicho atributo.

Filtros para selección de atributos

Ganancia de información (IG). Mide qué tan bien un atributo separa los ejemplos de entrenamiento en dos clases.

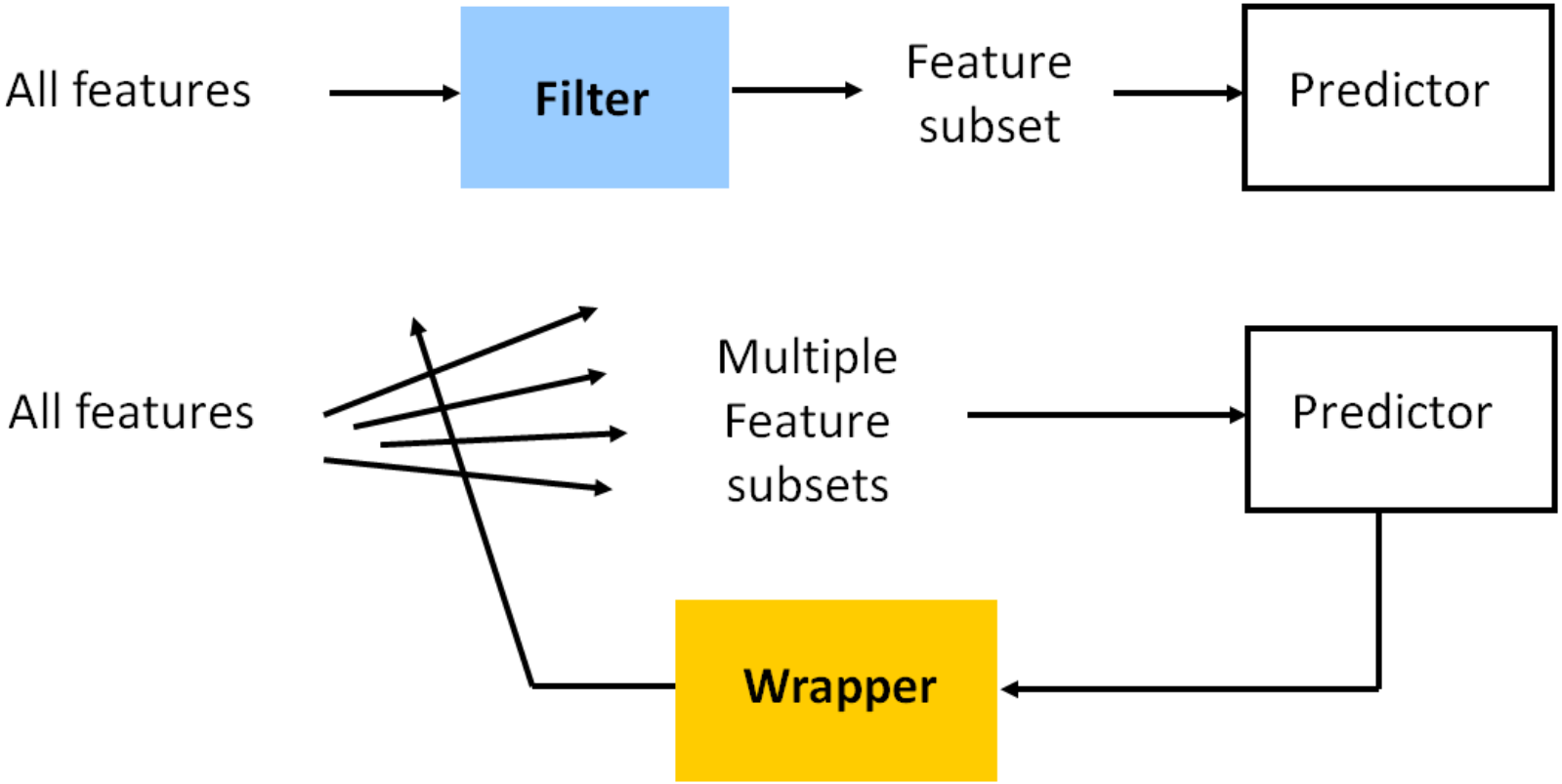


At each step, choose the feature that "reduces entropy" most. Work towards "node purity".

Otros criterios:

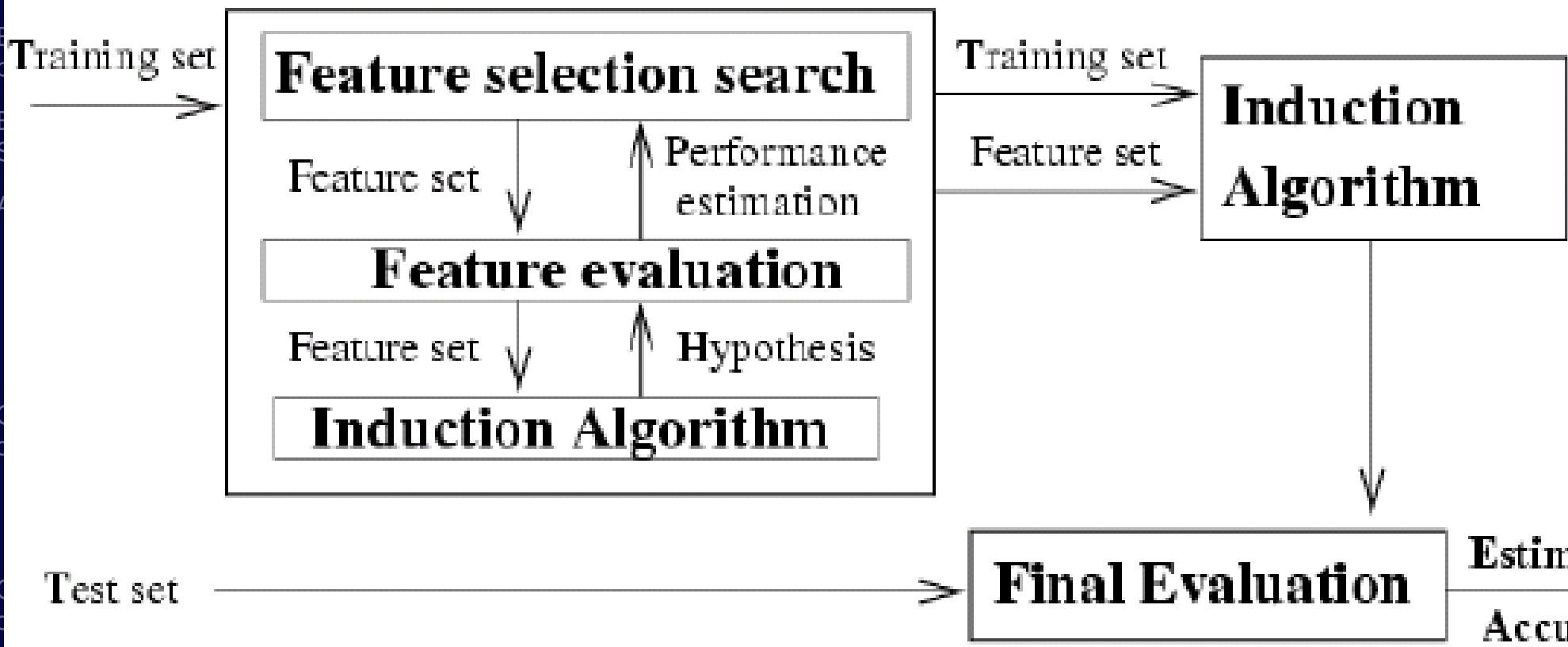
Method		X			Y			Comments
Name	Formula	B	M	C	B	M	C	
Bayesian accuracy	Eq. 3.1	+	s		+	s		Theoretically the golden standard, rescaled Bayesian relevance Eq. 3.2.
Balanced accuracy	Eq. 3.4	+	s		+	s		Average of sensitivity and specificity; used for unbalanced dataset, same as AUC for binary targets.
Bi-normal separation	Eq. 3.5	+	s		+	s		Used in information retrieval.
F-measure	Eq. 3.7	+	s		+	s		Harmonic of recall and precision, popular in information retrieval.
Odds ratio	Eq. 3.6	+	s		+	s		Popular in information retrieval.
Means separation	Eq. 3.10	+	i	+	+			Based on two class means, related to Fisher's criterion.
T-statistics	Eq. 3.11	+	i	+	+			Based also on the means separation.
Pearson correlation	Eq. 3.9	+	i	+	+	i	+	Linear correlation, significance test Eq. 3.12, or a permutation test.
Group correlation	Eq. 3.13	+	i	+	+	i	+	Pearson's coefficient for subset of features.
χ^2	Eq. 3.8	+	s		+	s		Results depend on the number of samples m.
Relief	Eq. 3.15	+	s	+	+	s	+	Family of methods, the formula is for a simplified version ReliefX, captures local correlations and feature interactions.
Separability Split Value	Eq. 3.41	+	s	+	+	s		Decision tree index.
Kolmogorov distance	Eq. 3.16	+	s	+	+	s	+	Difference between joint and product probabilities.
Bayesian measure	Eq. 3.16	+	s	+	+	s	+	Same as Vajda entropy Eq. 3.23 and Gini Eq. 3.39.
Kullback-Leibler divergence	Eq. 3.20	+	s	+	+	s	+	Equivalent to mutual information.
Jeffreys-Matusita distance	Eq. 3.22	+	s	+	+	s	+	Rarely used but worth trying.
Value Difference Metric	Eq. 3.22	+	s		+	s		Used for symbolic data in similarity-based methods, and symbolic feature-feature correlations.
Mutual Information	Eq. 3.29	+	s	+	+	s	+	Equivalent to information gain Eq. 3.30.
Information Gain Ratio	Eq. 3.32	+	s	+	+	s	+	Information gain divided by feature entropy, stable evaluation.
Symmetrical Uncertainty	Eq. 3.35	+	s	+	+	s	+	Low bias for multivalued features.
J-measure	Eq. 3.36	+	s	+	+	s	+	Measures information provided by a logical rule.
Weight of evidence	Eq. 3.37	+	s	+	+	s	+	So far rarely used.
MDL	Eq. 3.38	+	s		+	s		Low bias for multivalued features.

Filtros vs. Wrappers



Wrappers para selección de atributos

Diagrama general de un método tipo wrapper.



Wrappers para selección de atributos

Variantes de estrategias de búsqueda:

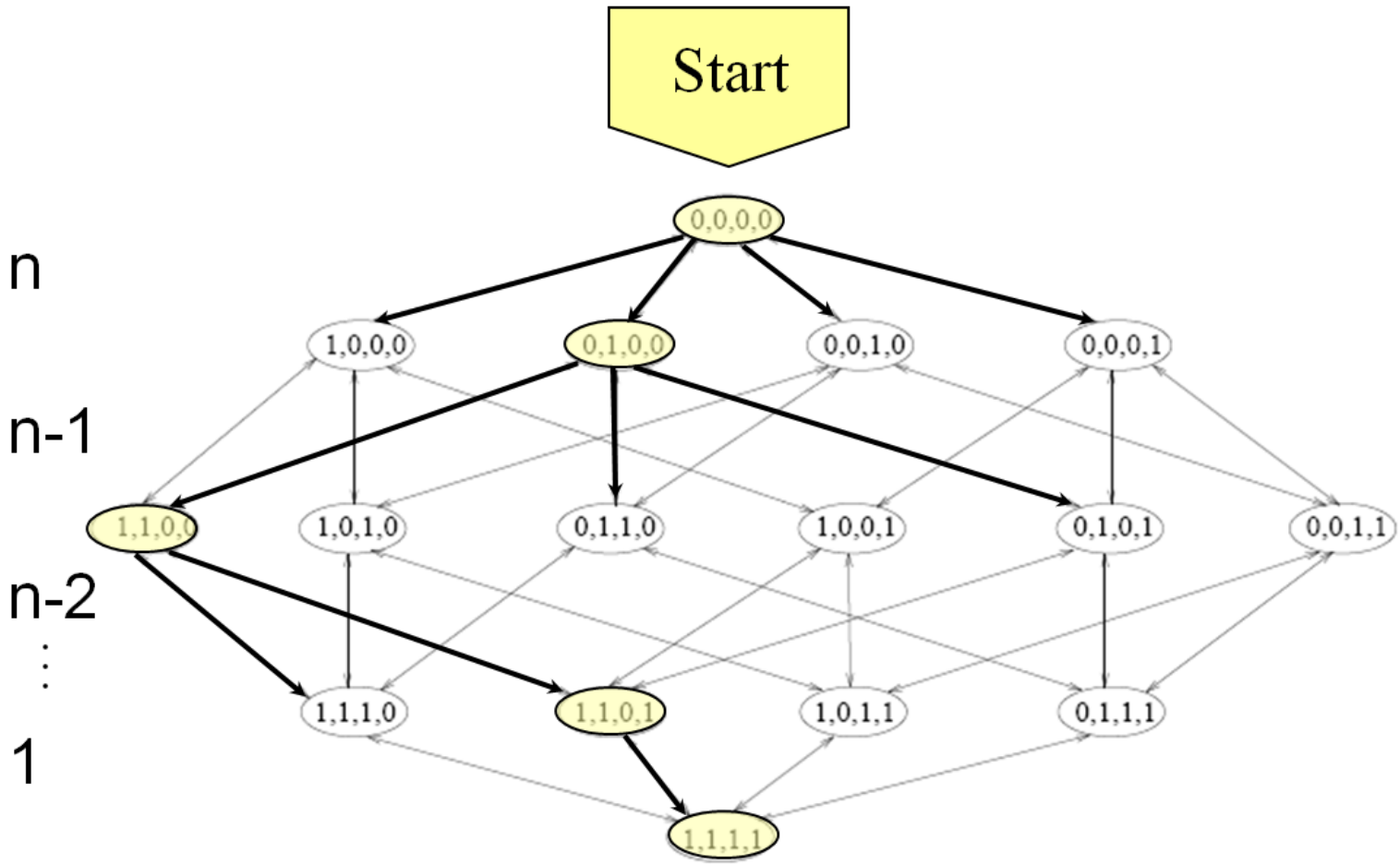
- Sequential Forward Selection (SFS)
- Sequential Backward Elimination (SBS)
- Beam search: keep k best path at each step
- Floating search (SFFS and SBFS): Alternate between SFS and SBS as long as we find better subsets than those of the same size obtained so far
- Extensive search (simulated annealing, genetic algorithms, exhaustive search).

Wrappers para selección de atributos

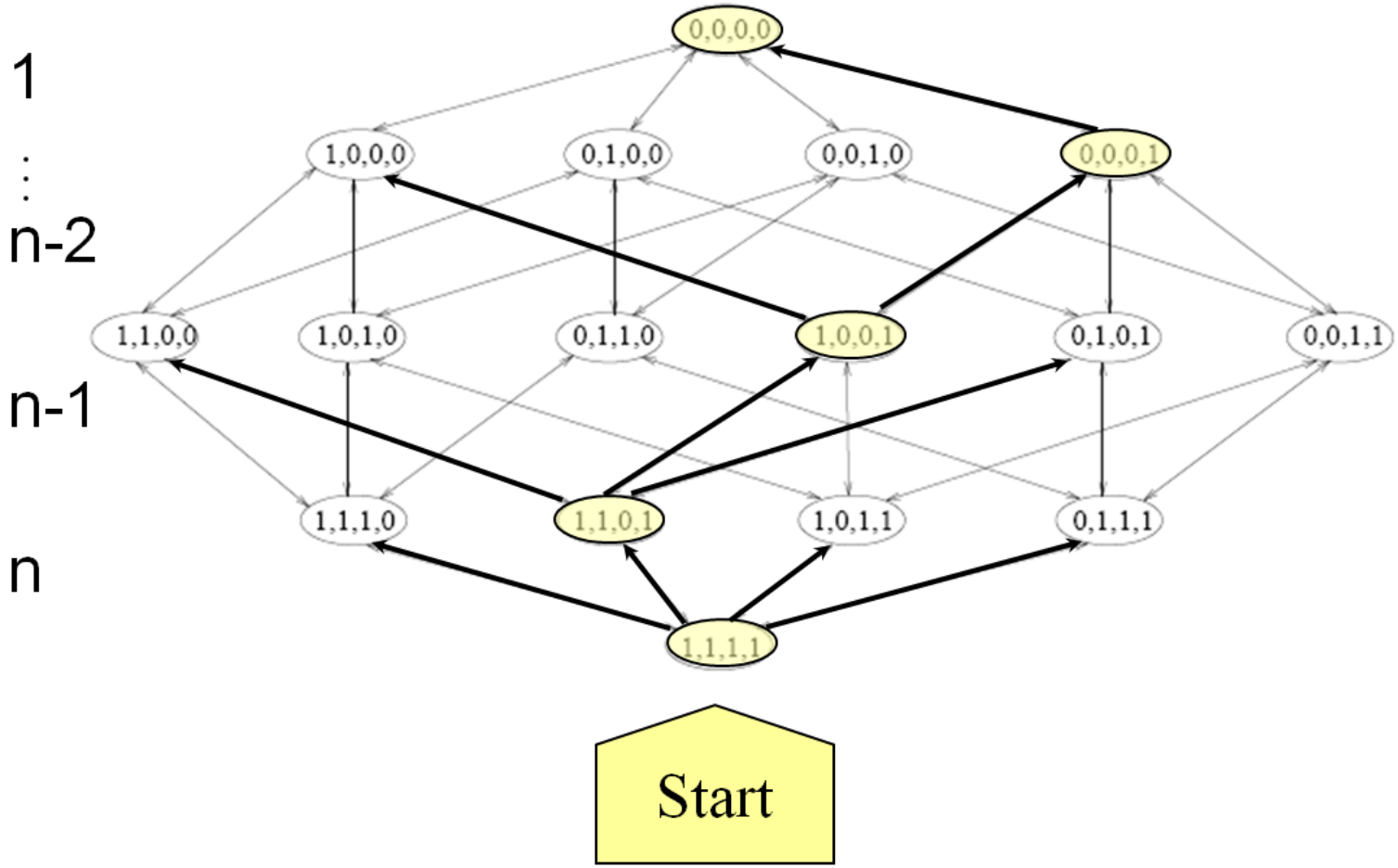
Variantes:

- **Criterio:** Generalmente son guiados por el desempeño de clasificación, medido por una función perdida.
- **Muestreo:** Se suele usar validación cruzada u otras estrategias de evaluación para evitar el sobre ajuste.
- **Criterio de paro:** Número máximo de iteraciones, *early stopping*, convergencia.

Forward selection (SFS)

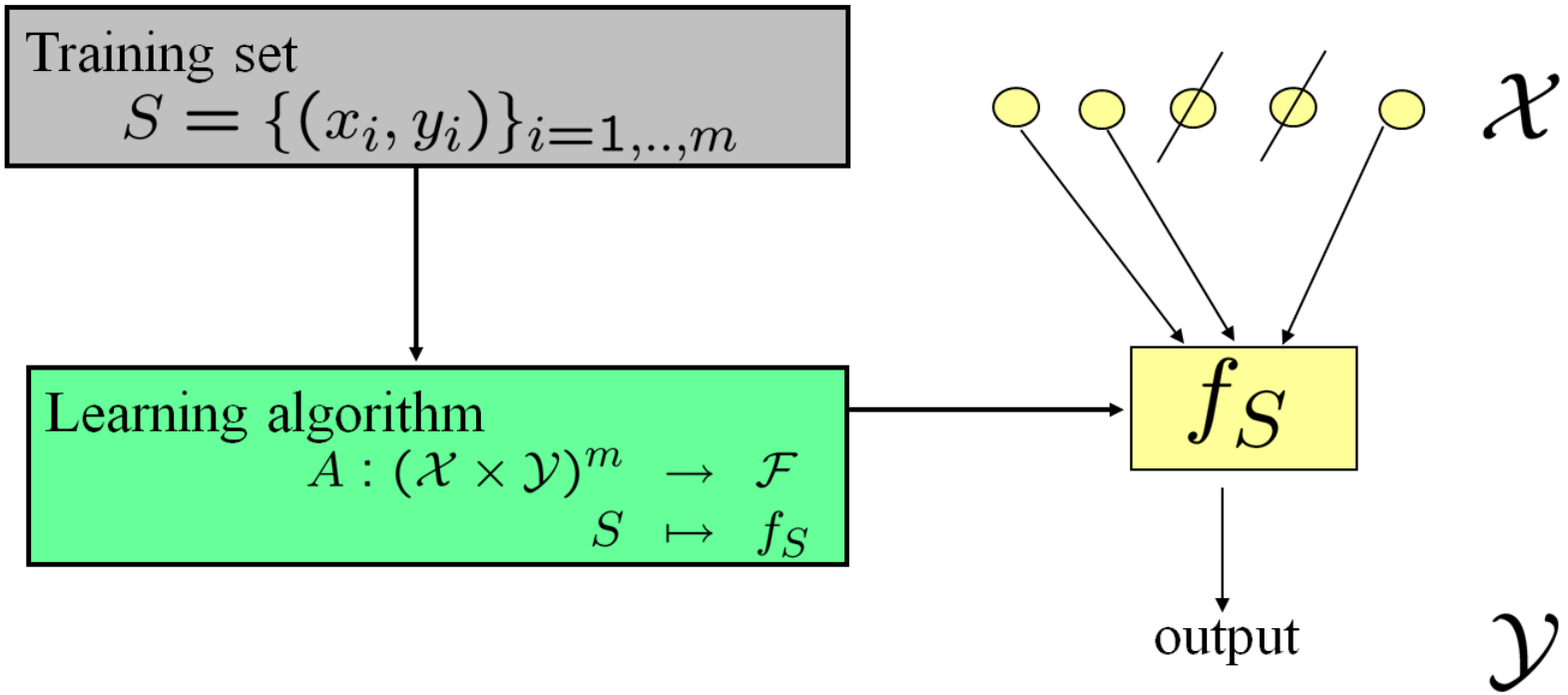


Backward elimination (SBS)



Embedded methods

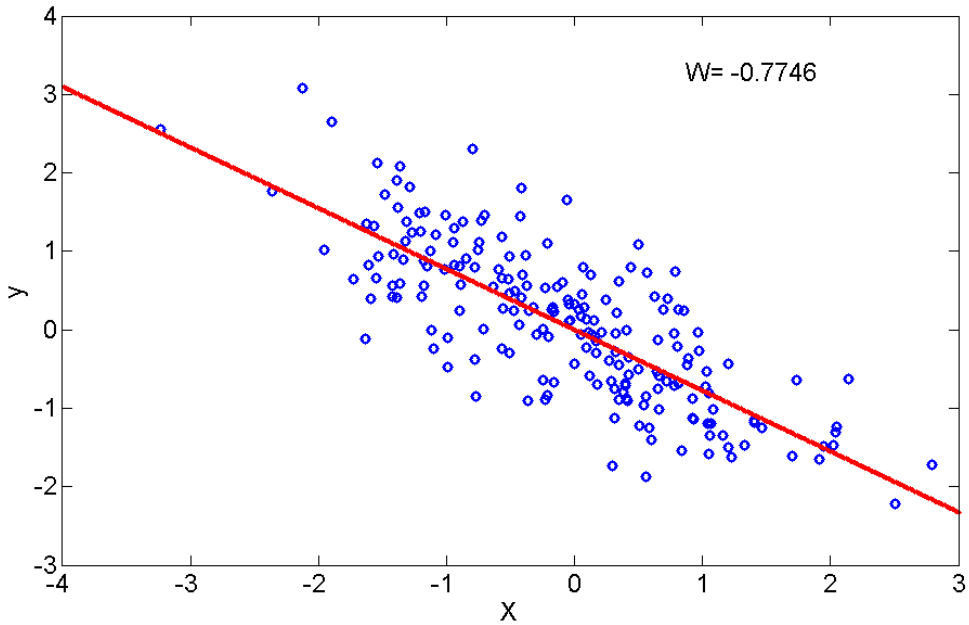
Definition: an embedded feature selection method is a *machine learning algorithm* that returns a model using a limited number of features.



Embedded methods

Idea: sacar provecho del modelo o proceso de aprendizaje para identificar variables relevantes.

- Regresión lineal.



Embedded methods

Idea: sacar provecho del modelo o proceso de aprendizaje para identificar variables relevantes.

- Ejemplo: SVM
- Sabiendo que:

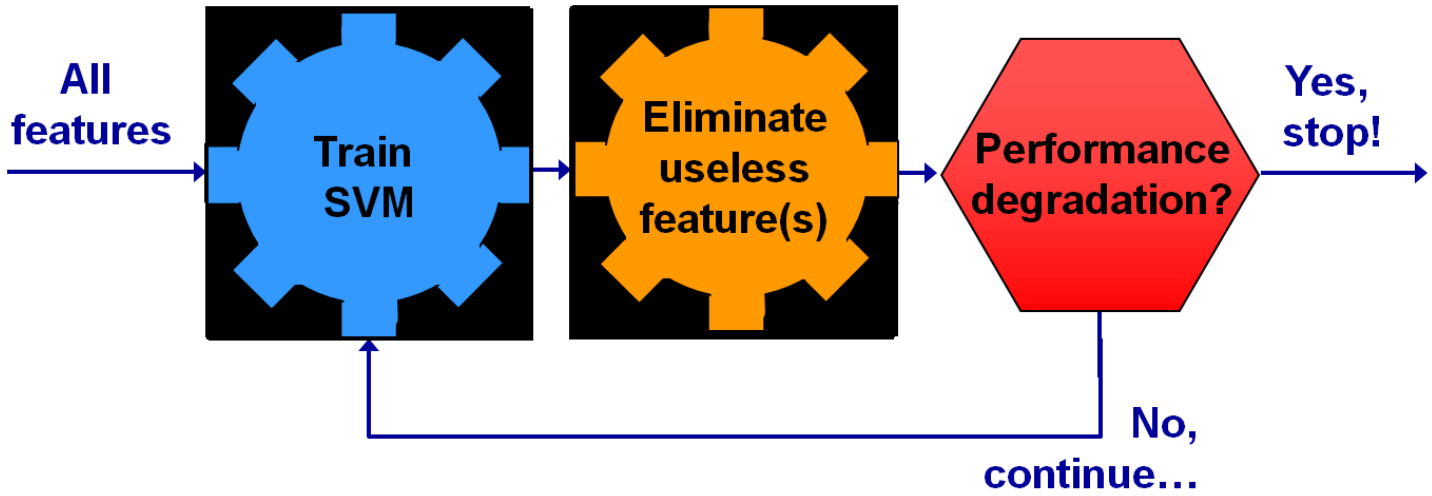
$$\mathbf{w} = \sum_k \alpha_k y_k \mathbf{x}_k \tag{1}$$

- Se usa a **w** como criterio para dar importancia a los atributos

Embedded methods

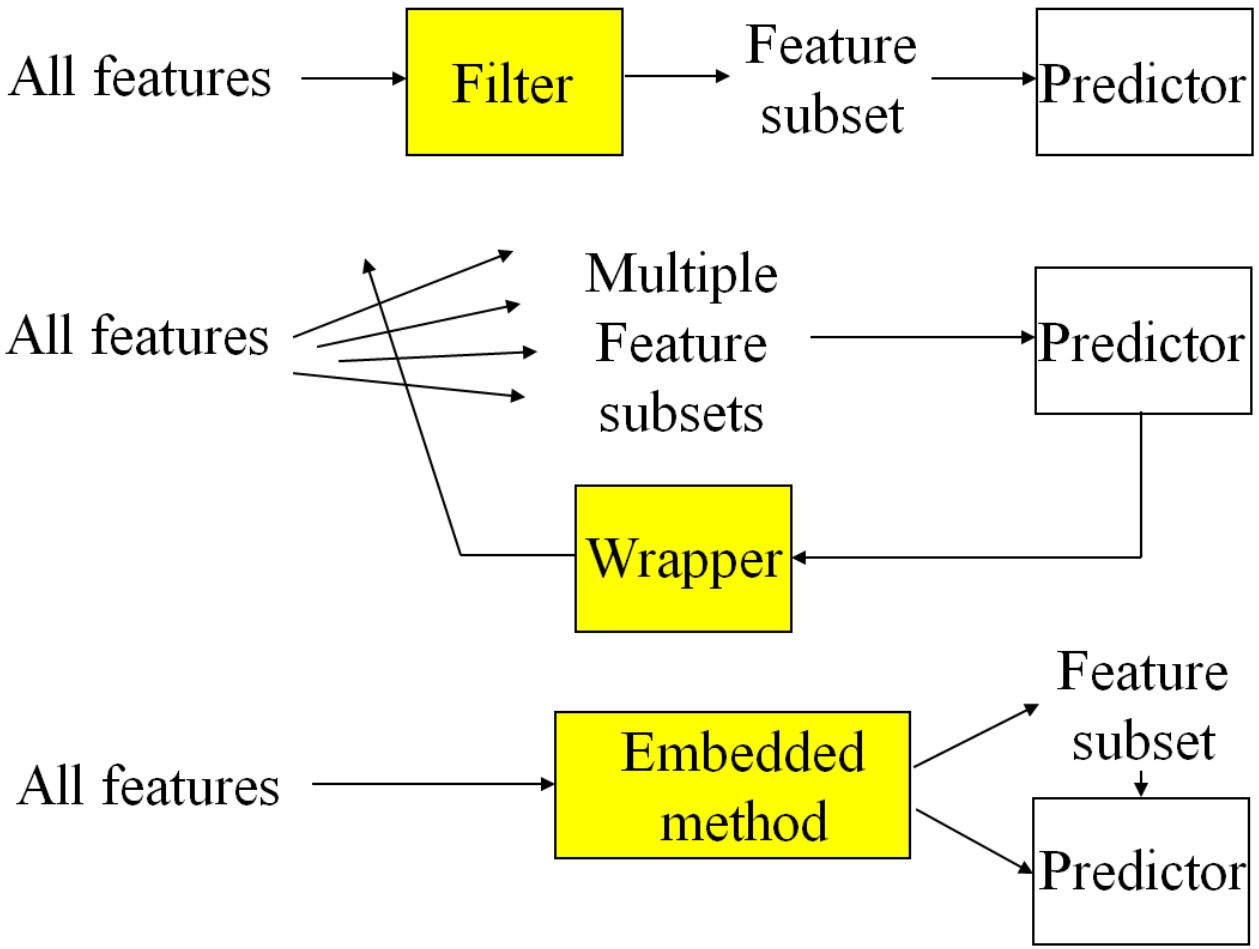
Idea: sacar provecho del modelo o proceso de aprendizaje para identificar variables relevantes.

- SVM.



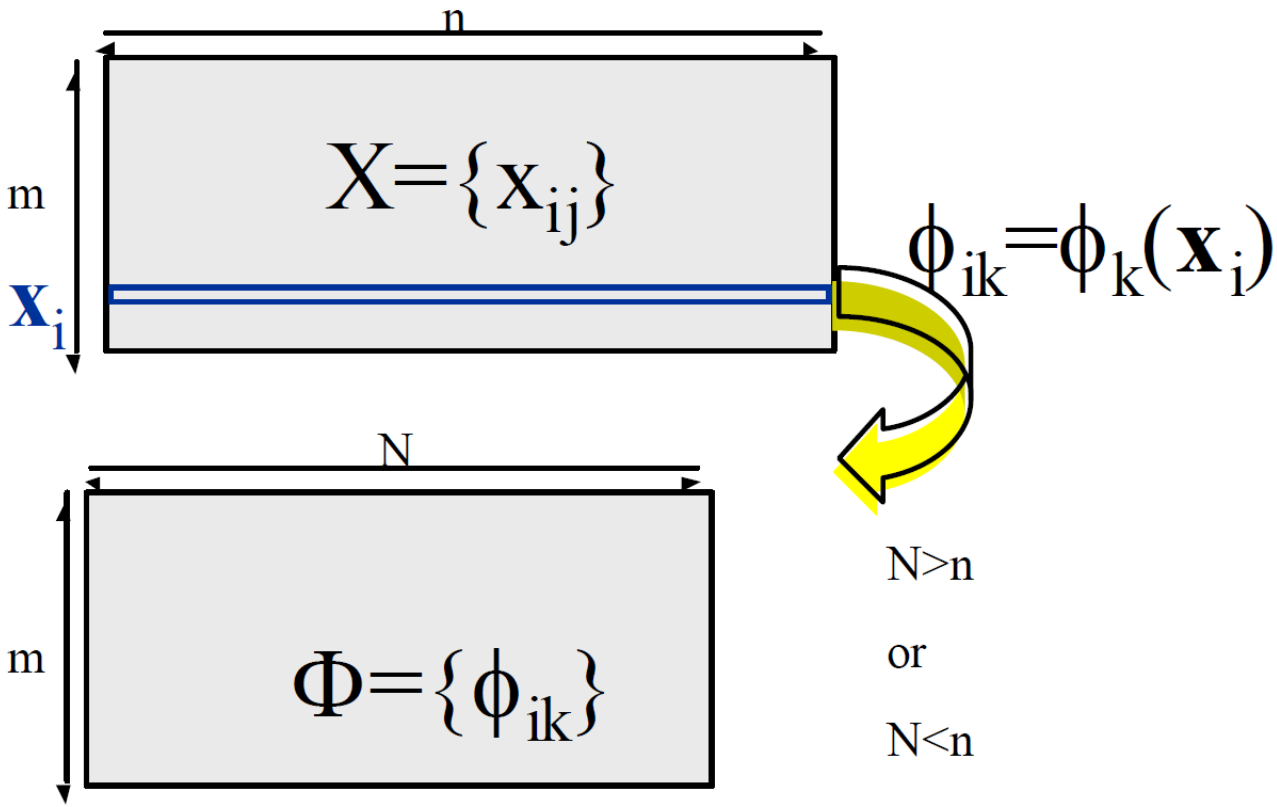
Recursive Feature Elimination (RFE) SVM. *Guyon-Weston, 2000. US patent 7,117,188*

Selección de atributos



Generación de atributos

Mapear los atributos originales a un nuevo espacio de menor dimensión.



Generación de atributos

- Cómo definir el mapeo $\Phi(\mathbf{x})$?
- Métodos más usados: PCA, LDA, AutoEncoders, Factorización de matrices, etc.

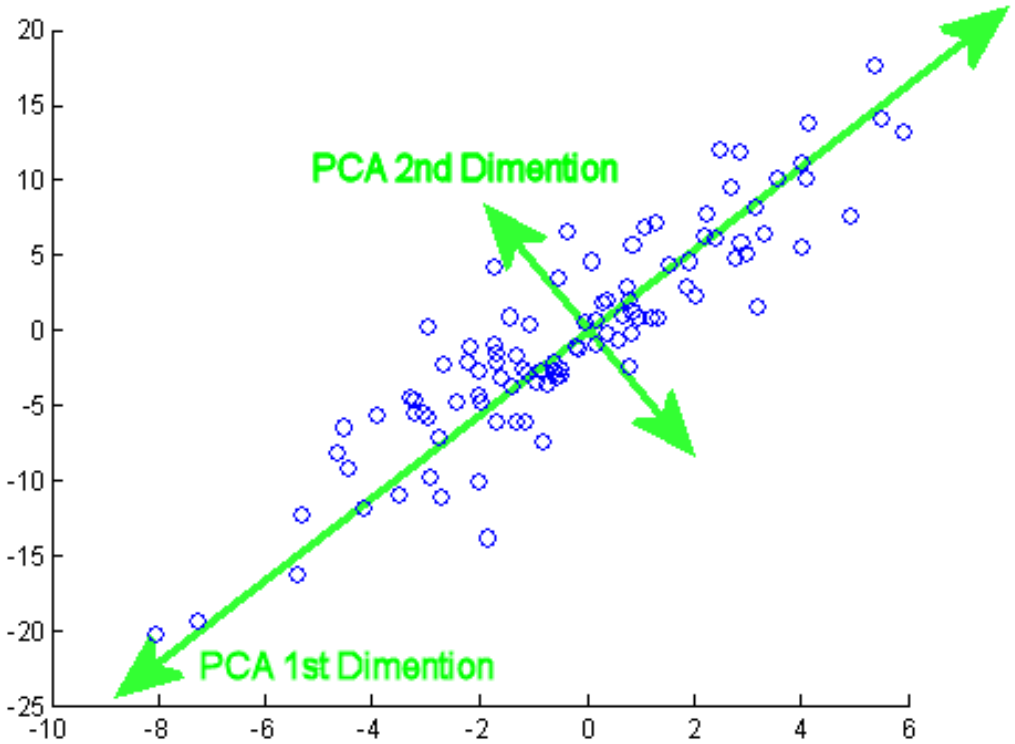
PCA: Análisis de Componentes Principales

PCA: Herramienta estándar en análisis de datos

- Simple
- No paramétrico
- Extrae información relevante a partir de conjuntos de datos confusos
- Provee forma de reducir un conjunto de datos complejo a otro con dimensión menor
- Revela estructuras simplificadas (algunas veces ocultas)
- Permite remover ruido, información no relevante.

PCA: Análisis de Componentes Principales

Idea: Conceptualmente, PCA es un método que encuentra un conjunto de bases que maximizan la varianza de los datos originales y que son ortogonales entre si. Encuentra las direcciones de mayor variación.



PCA: Análisis de Componentes Principales

Receta: Sea $\mathcal{X} = \{(\mathbf{x_i}\}_{1,\ldots,N}$ nuestro conjunto de datos de entrenamiento[1] con $\mathbf{x_i} \in \mathbb{R}^d$.

- Centrar los datos: A cada columna $\mathbf{X}_i$ de $\mathbf{X}$ restamos la media de $\mathbf{X}_i$.
- Calcular la matriz de covarianza:

$$\mathbf{C_X} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$$

- Calcula los eigenvectores $\mathbf{v}$ de $\mathbf{C_X}$:

$$\mathbf{C_X}\mathbf{v} = \lambda\mathbf{v}$$

- La matriz $\mathbf{v}$ es el conjunto de componentes principales.

[1] Ojo: PCA es un método no supervisado.

PCA: Análisis de Componentes Principales

- Se pueden proyectar nuevos datos $\mathbf{X}^*$ al espacio de componentes principales mediante:

$$\mathbf{P}^* = \mathbf{v}^T \mathbf{X}^*$$

- Se pueden reconstruir los datos originales:

$$\hat{\mathbf{X}}^* = \mathbf{P}^* \mathbf{v}^\mathbf{T}$$

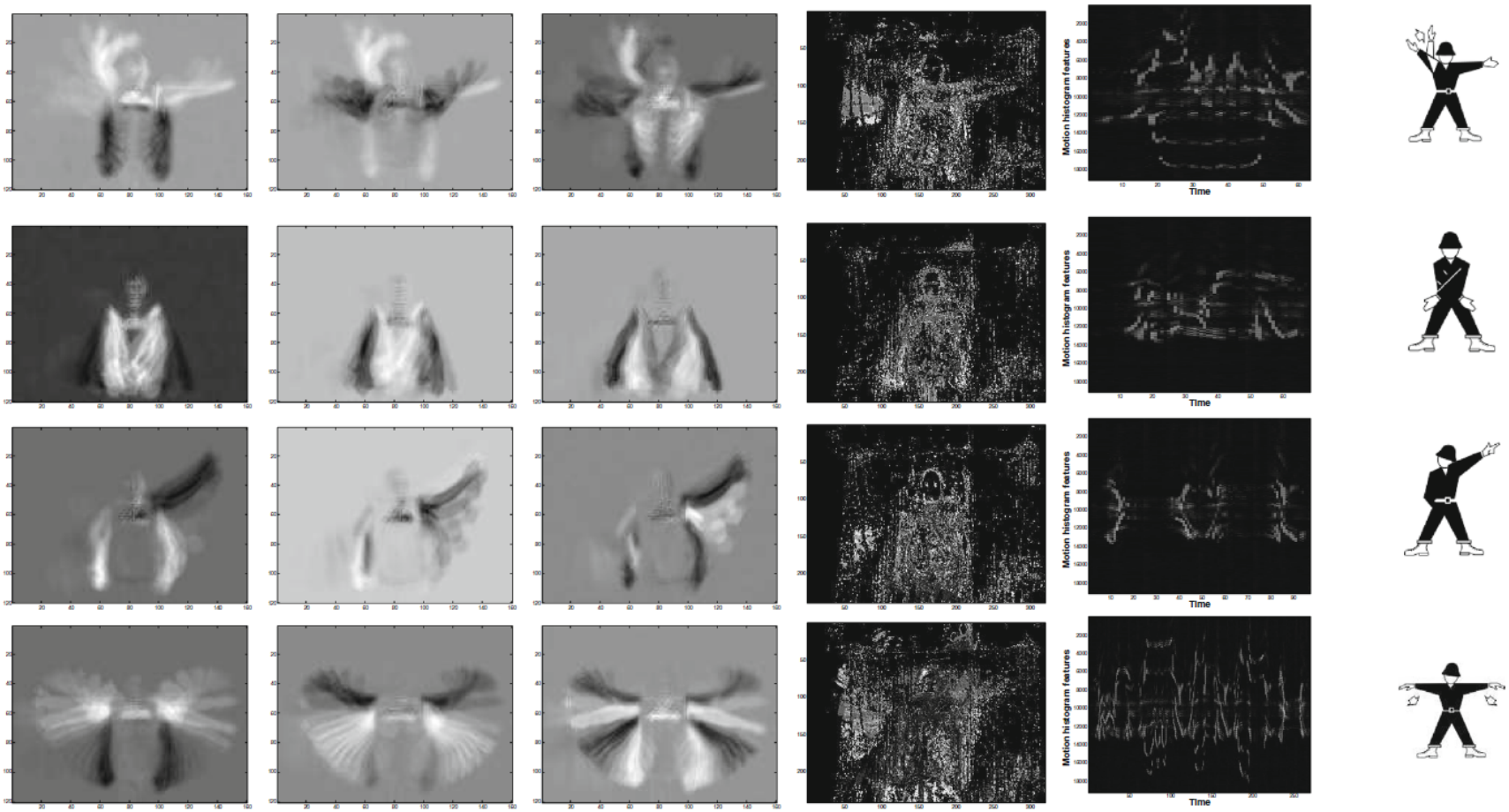
PCA: Para qué sirve?

Cómo se ven los componentes principales?

"eigenfaces"

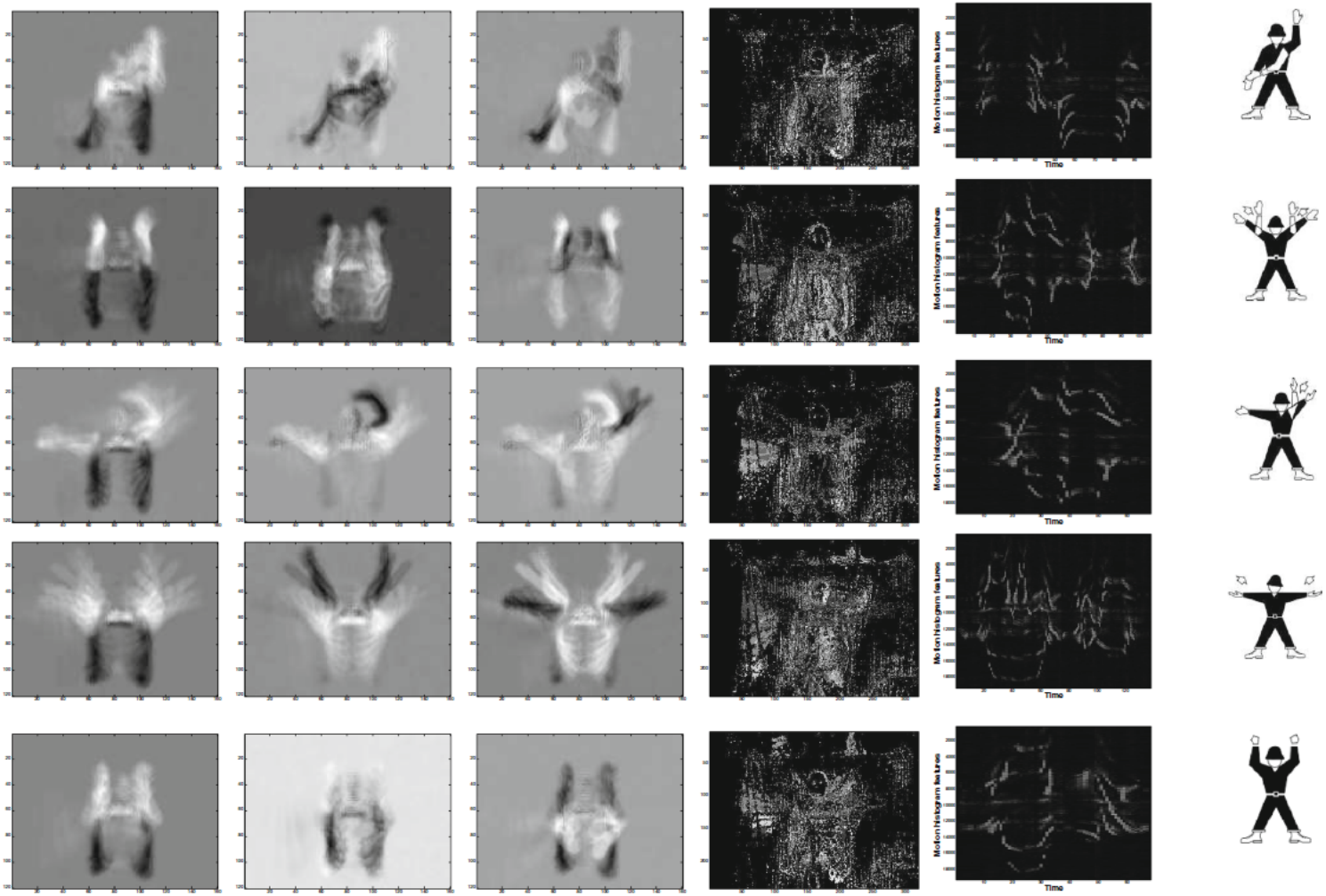
PCA: Para qué sirve?

Cómo se ven los componentes principales?



PCA: Para qué sirve?

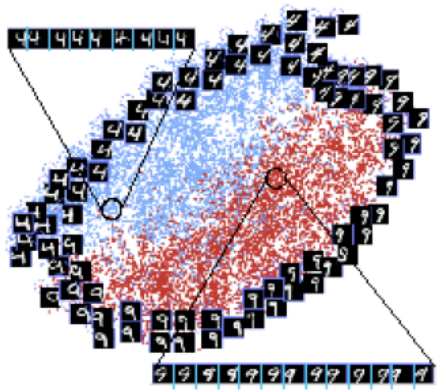
Cómo se ven los componentes principales?



PCA: Para qué sirve?

Aplicación 1: Reducción de la dimensionalidad de los datos.
Con PCA obtenemos un conjunto de componentes que nos permiten mapear los datos a otro espacio. Los componentes están ordenados de acuerdo a su importancia para explicar los datos, así, es posible proyectar los datos originales en solo unos cuantos componentes principales.

$$\mathbf{P}^* = \mathbf{v}_{1:k}^T \mathbf{X}^*$$

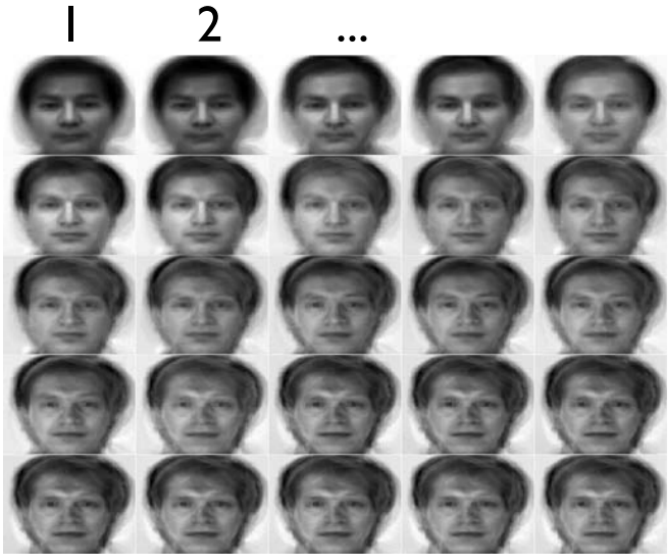


PCA: Para qué sirve?

Aplicación 2: Remover ruido, información no relevante de lo datos. La idea es obtener los componentes principales y posteriormente reconstruir los datos usando solo un número pequeño de componentes k ($k << d$).

$$\hat{\mathbf{X}}^* = \mathbf{P}^* \mathbf{v}^T_{1,\ldots,k}$$

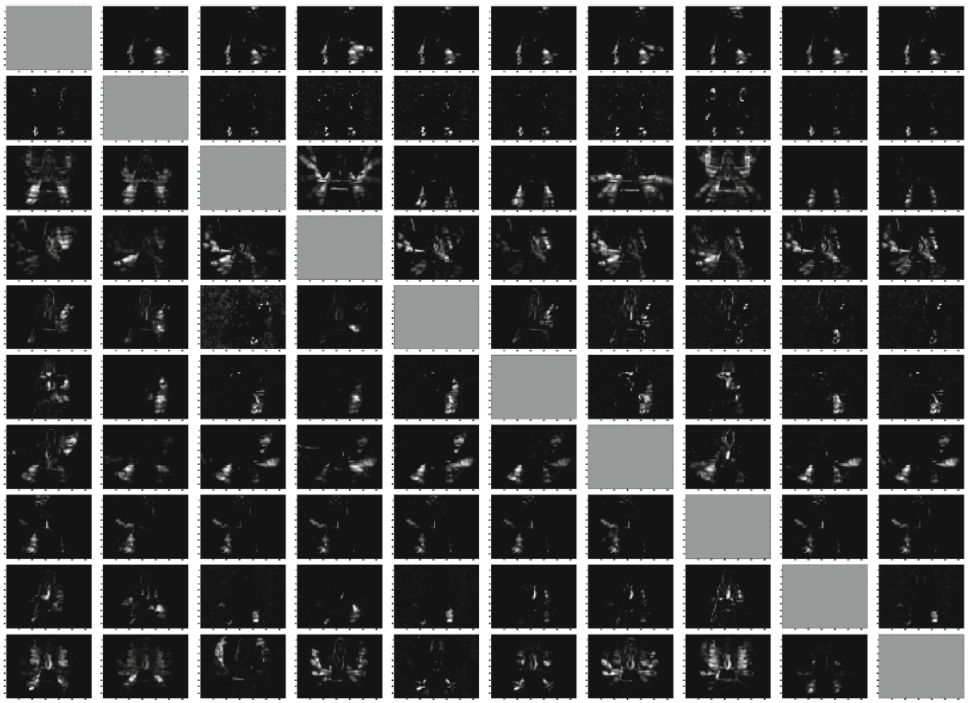
Reconstruction using the first 25 components (eigenfaces), one at a time



Same, but adding 8 PCA components at each step

PCA: Para qué sirve?

Aplicación 3: Como clasificador. Se sabe que los componentes principales minimizan el error de reconstrucción de los objetos originales. Entonces, se puede aprender un modelo PCA por cada clase y clasificar nuevas instancias asignandolas a la clase con el menor error de reconstrucción.

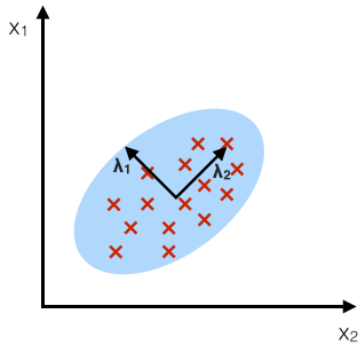


LDA: Análisis de discriminador lineal

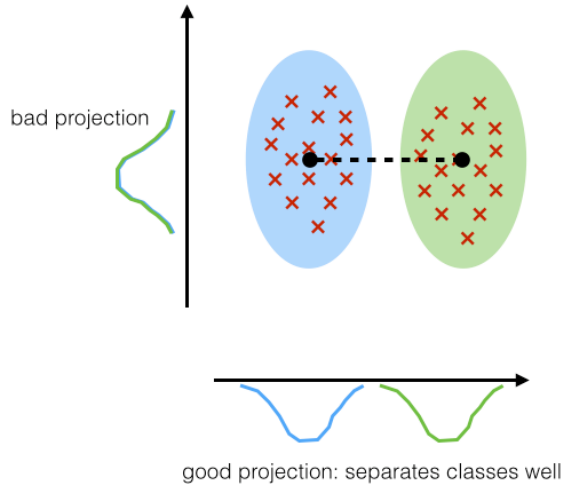
PCA es una herramienta que provee transformaciones lineales que capturan la varianza de los datos, sin embargo, es un método no supervisado.

LDA (*Linear discriminant analysis*) es una variante supervisada que nos permite generar atributos tratando de maximizar la discriminatividad.

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



LDA: Análisis de discriminador lineal

En LDA (para reducción de la dimensionalidad), buscamos la proyección de los datos que nos maximiza la separación entre clases.

$$\mathbf{y} = \mathbf{w}^T\mathbf{x}$$

Queremos encontrar $\mathbf{w}$ que maximiza dicha separación. Fisher planteó el siguiente criterio:

$$J(\mathbf{w}) = \frac{(c_- - c_+)^2}{s_+^2 + s_-^2}$$

con

$$c_+ = \mathbf{w}^T\mathbf{c}_+; c_- = \mathbf{w}^T\mathbf{c}_-$$

$$s_+^2 = \sum_{\{i|y_i=+1\}} (y_i - c_+)^2; s_-^2 = \sum_{\{i|y_i=-1\}} (y_i - c_-)^2;$$

LDA: Análisis de discriminador lineal

Haciendo explícita la dependencia con $\mathbf{w}$, tenemos:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

donde $\mathbf{S}_B$ es la matriz de covarianza entre clases:

$$\mathbf{S}_B = (\mathbf{c}_- \mathbf{c}_+)(\mathbf{c}_- \mathbf{c}_+)^T$$

y $\mathbf{S}_W$ es la matriz de covarianza intra clase:

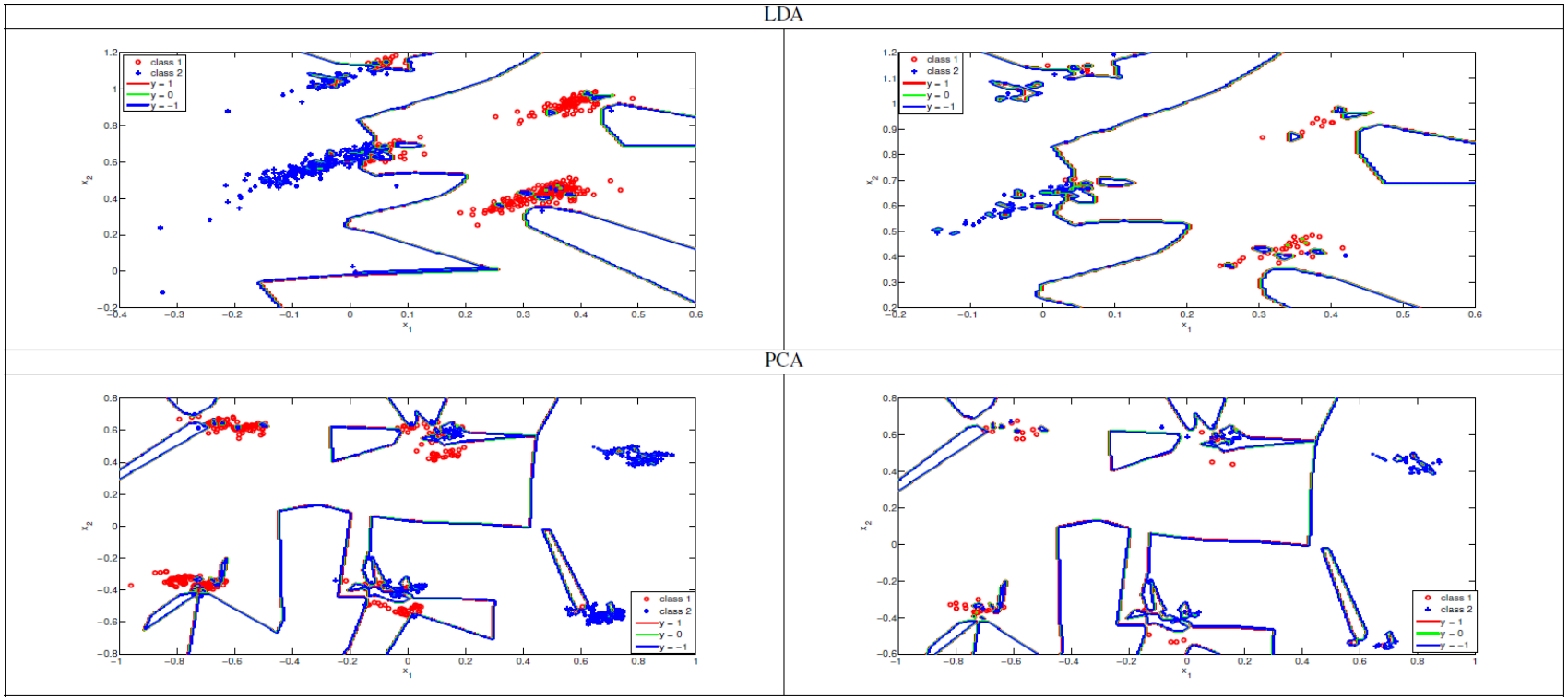
$$\mathbf{S}_W = \sum_{\{i|y_i=+1\}} (\mathbf{x}_i - \mathbf{c}_+)(\mathbf{x}_i - \mathbf{c}_+)^T + \sum_{\{i|y_i=-1\}} (\mathbf{x}_i - \mathbf{c}_-)(\mathbf{x}_i - \mathbf{c}_-)^T$$

La solución está dada por:

$$\mathbf{w} \approx \mathbf{S}_W^{-1}(\mathbf{c}_- - \mathbf{c}_+)$$

LDA: Análisis de discriminador lineal

Proyección de datos con LDA:



Programación genética como extractor de atributos

- Una alternativa a PCA y LDA que trabaja de forma intuitiva es la generación de atributos mediante programación genética.
- La idea para generar un atributo es combinar, mediante ciertas operaciones, atributos del espacio original, e.g.,:

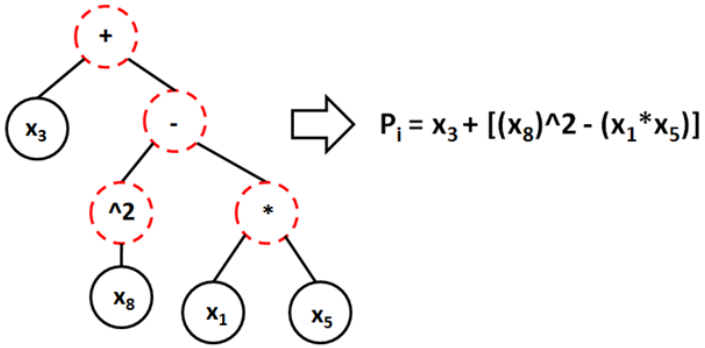
$$\mathbf{w}_i = \Phi_i(\mathbf{X}_S)$$

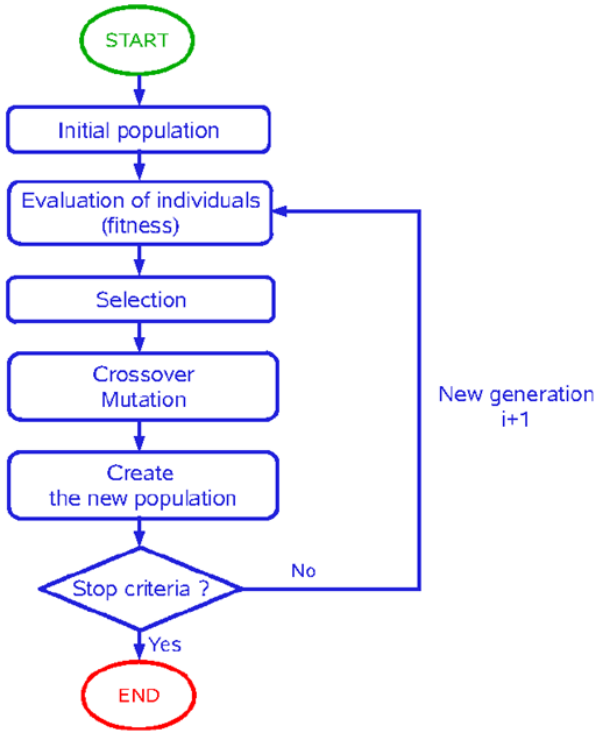
  Donde $\mathbf{X}_S$ es un subconjunto de las columas de $\mathbf{X}$ y Φ_i es una función que combina dichas columnas.
- El problema consiste en encontrar $\Phi_{1,\ldots,k}$

Programación genética

Qué es programación genética?

GP: standard EA with tree-based representation





Programación genética

Programación genética para generar atributos
La idea es muy simple: codificar atributos como arboles, y dejar al programa genético que determine los mejores atributos.

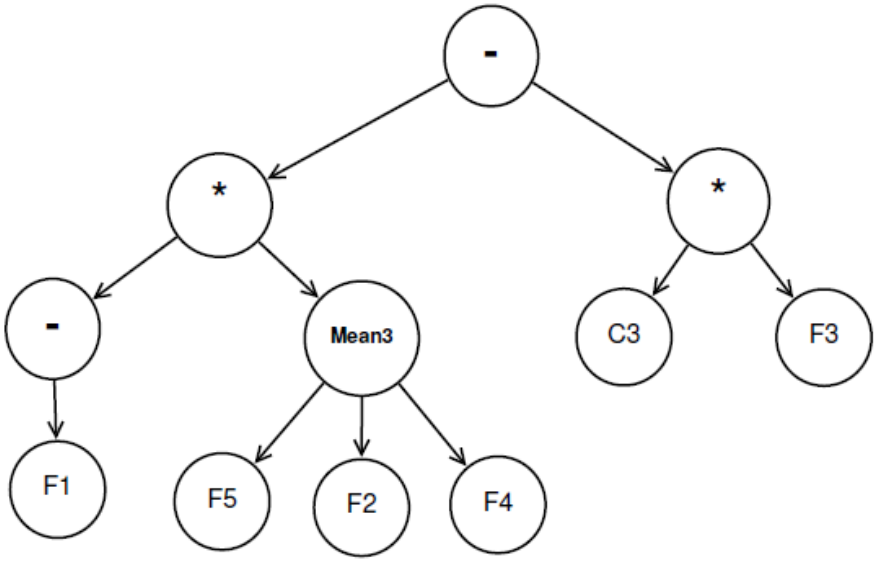
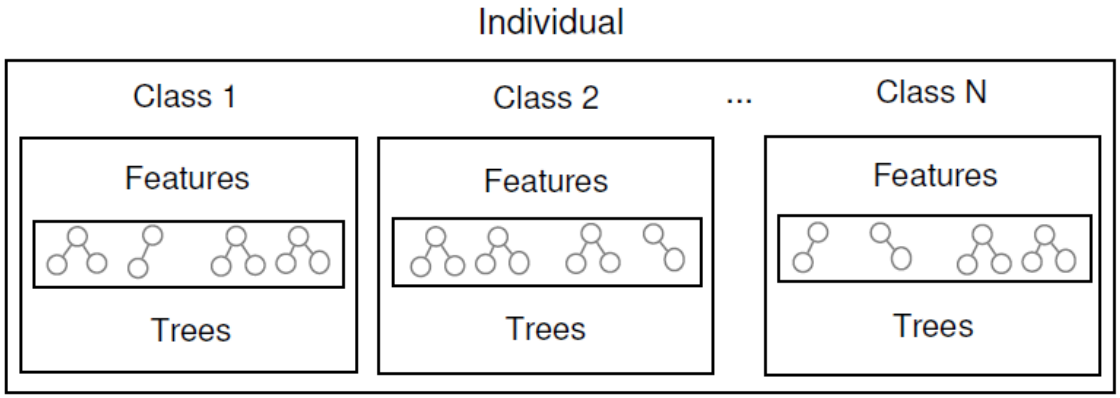


Fig. 2: A feature representation (tree-feature). $\mathcal{F}_y = (-F_1 * \frac{F_5+F_2+F_4}{3}) - (C_3 * F_3)$

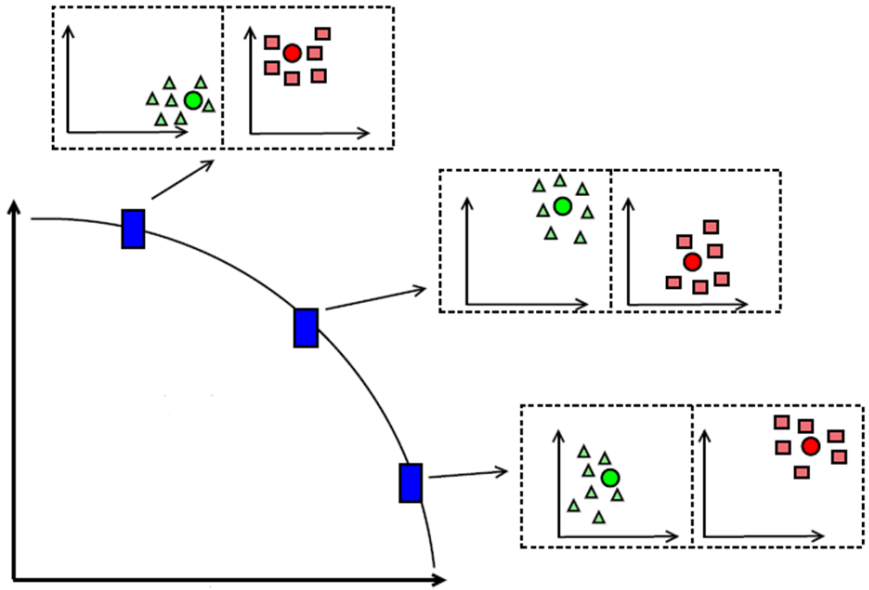
Programación genética

Programación genética para generar atributos
La idea es muy simple: codificar atributos como arboles, y dejar al programa genético que determine los mejores atributos.



Programación genética

Programación genética para generar atributos
La idea es muy simple: codificar atributos como arboles, y dejar al programa genético que determine los mejores atributos.



Construccón de Atributos

- Aunque hemos mencionado en general las ventajas de eliminar atributos, muchas veces también conviene añadir atributos.
- Esto es porque una representación diferente puede simplificar la tarea del algoritmo de aprendizaje (e.g., máquinas de soporte vectorial).
- Una forma de introducir nuevos atributos es creando atributos derivados de los atributos originales.
- Normalmente se usan combinaciones booleanas para atributos binarios y combinaciones aritméticas para atributos numéricos.

Construcción de Atributos

- También se pueden utilizar aproximaciones lineales de atributos que den una buena predicción de los datos originales (*singular value decomposition* o SVD).
- La idea de *Constructive Induction* es crear nuevos atributos en base a operadores de construcción predefinidos e ir seleccionando los mejores atributos (originales o derivados), manteniendo simpre fijo un número máximo de atributos, hasta llegar a un criterio de paro.

Algoritmo General de Construcción de Atributos

AtribActual = atributos originales
Operadores = conjunto de operadores de construcción
while NOT criterio de terminación **do**

- *AtribNvos* = *AtribActual* $\cup$ atributos nuevos construidos con *Operadores* sobre *AtribActual*
- Corre algoritmo de aprendizaje en *AtribNvos*
- *AtribActual* = Selecciona los mejores atributos de *AtribNvos*

end while

Ejemplo de Construcción de Atributos

Construcción de tres atributos, a partir de R,G,B para clasificar piel:

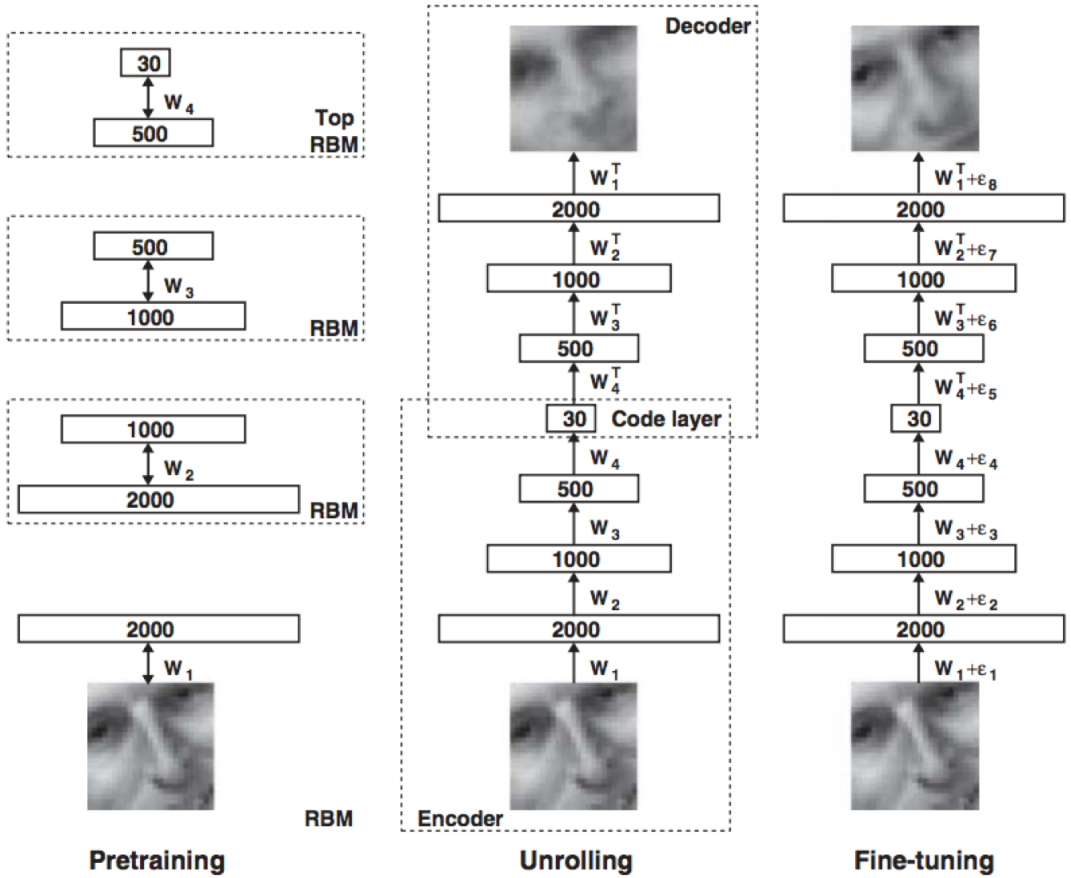
color model (three components)			recall (%)	precision (%)	success rate (%)
$\frac{r}{g}$	$\frac{rb}{(r+g+b)^2}$	$\frac{rg}{(r+g+b)^2}$	93.7	**91.7**	**92.6**
$\frac{r}{g}$	$\frac{rb}{(r+g+b)^2}$	$\frac{g}{b}$	94.2	88.6	91
$\frac{r}{g}$	$\frac{g}{b}$	$\frac{rb}{(r+g+b)^2}$	94.3	88.5	91
$\frac{b}{g}$	$\frac{r}{g}$	$\frac{rb}{(r+g+b)^2}$	95.1	87.5	91
$\frac{r}{g}$	$\frac{g}{b}$	$\frac{-2r+g+b}{3(r+g+b)}$	95.1	86	90
$\frac{b}{g}$	$\frac{r+g+b}{3r}$	$\frac{r-g}{r+g+b}$	96	85	89.2
$\frac{b}{g}$	$\frac{r}{g}$	$\frac{-2r+g+b}{3(r+g+b)}$	96	84.3	89.1
$\frac{b}{g}$	$\frac{r+g+b}{3r}$	$\frac{r+g-2b}{3(r+g+b)}$	**97.5**	67.1	75
SPM on raw RGB			95.8	77.3	91

Construcción de Atributos

- Una alternativa que se ha usado para la construcción de atributos está basada en *clustering*
- La idea es reemplazar un grupo de variables "parecidas" por el centroide del *cluster* que se vuelve un nuevo atributo.
- En general, el automatizar el cambio de representación sigue siendo un problema abierto con poco trabajo en ML

Autoencoders

Los auto encoders pueden verse/usarse como reductores de dimensionalidad:



Comentarios finales

- La dimensionalidad puede llegar a ser un problema serio en aprendizaje computacional, cuando ésta es “grande”.
- La maldición de la dimensionalidad afecta mayormente a cierto tipo de métodos.
- Existen dos alternativas: seleccionar o generar atributos.
- Ambas estrategias son útiles y se pueden combinar, además que se les puede dar otros usos.
- En general es un arte, seleccionar el mejor método para un problema en particular.

Comentarios

- En general es deseable hacer primeramente un análisis pare tratar de eliminar las variables cláramente irrelevantes e identificar las variables redundandantes.
- Al escoger un algoritmo de selección de atributos es importante tomar en cuenta:
  1. El proposito: visualización, entendimiento de datos, limpieza de datos, eliminación de redundancia y/o irrelevancia, desempeño
  2. El tiempo de procesamiento: si no es crítico se puede usar una estrategia de búsqueda cara
  3. La salida: lista ordenada o subconjunto

Comentarios

Cont...

4. La relación entre atributos relevantes y el total de atributos: Pocos relevantes - *forward selection*, pocos irrelevantes - *backward elimination*
5. Si es clasificación o *clustering* la tarea que se tiene
6. El tipo de atributos que se tienen
7. La relación atributos y el número de instancias
8. Si se puede usar conocimiento del dominio.

Algunos Retos

Dentro de los retos actuales:

- Poder lidear exitosamente con una alta dimensionalidad y/o alta dimensionalidad y pocas instancias
- Incorporar ideas de *active* y *progressive sampling* a selección de atributos
- Tomar en cuenta interdependencias entre varias variables a la vez
- Extender las ideas a otro tipo de atributos (e.g., secuencias, datos semi-estructurados, texto, etc.).
- Combinarlo con ideas de sobre/sub–muestreo