Aprendizaje Semisupervisado

Eduardo Morales

INAOE

Contenido

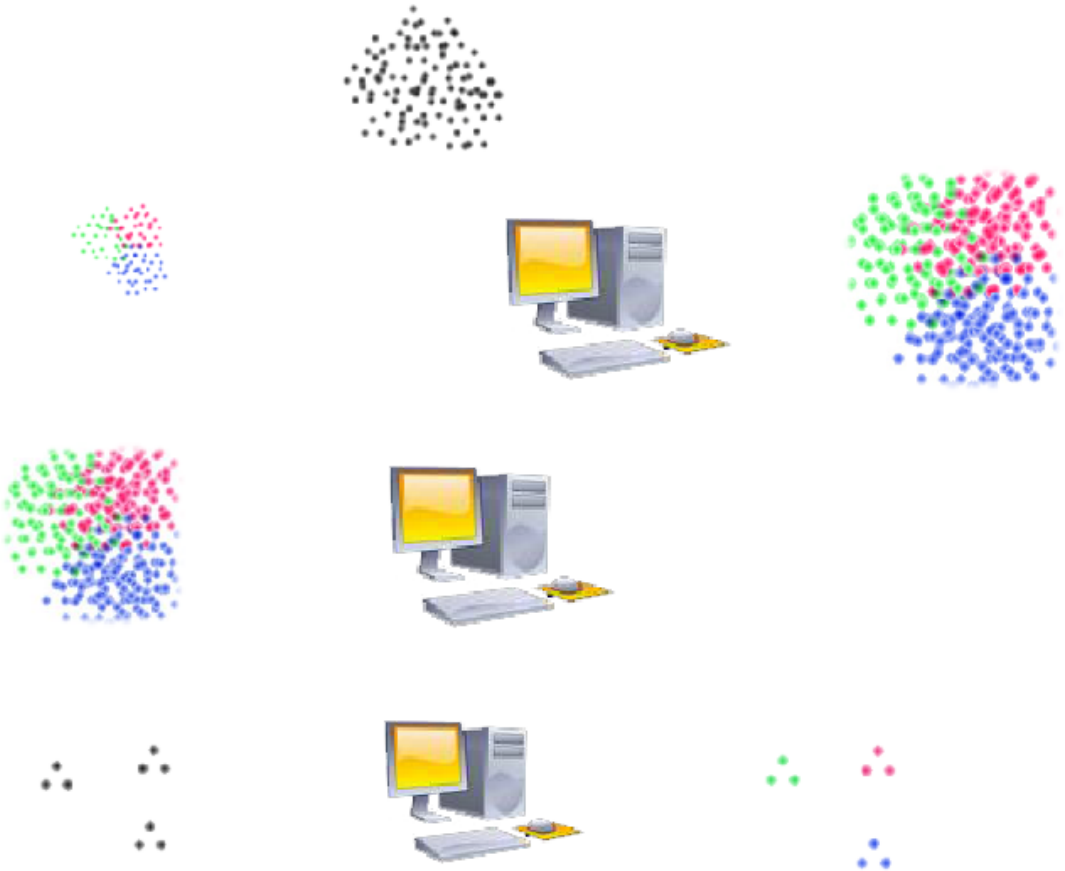
Aprendizaje Semisupervisado

- Actualmente existe una gran cantidad de datos disponibles, sin embargo, no todos tienen asignada una clase o etiqueta, con la cual crear un clasificador
- Los ejemplos más cláros de esto son en texto e imágenes, en donde existe una gran cantidad de texto e imágenes no etiguetadas
- El reto es ver si se pueden combinar los datos no etiquetados con los etiquetados para construir un mejor clasificador debido a: (i) la cantidad de datos no etiquetados dispobibles y (ii) el alto costo asociado a asignarles etiquetas

Aprendizaje Semisupervisado

- El añadir datos no etiquetados no quiere decir que siempre sea útil
- Normalmente se supone que los datos etiquetados y no etiquetados vienen de la misma distribución
- Por otro lado, los valores de los atributos pueden ser diferente para los datos etiquetados y los no etiquetados
- Finalmente, puede existir un sesgo en la selección de los datos no etiquetados

Esquema General de Aprendizaje Semisupervisado



Aprendizaje Semisupervisado

Se han propuesto varios métodos para resolver problemas de aprendizaje semi-supervisado, entre los que se encuentran:

- *Self-Training*
- *Co-training*
- Usando ensambles (ASSEMBLE)
- *Re-weighting*
- Basados en EM
- *transductive* SVM
- Modelos basados en grafos
- ...

Aquí vamos a ver sólo algunos de ellos.

Self-training

Es posiblemente el método más ampliamente utilizado y también el más simple:

1. Aprender un clasificador con los ejemplos etiquetados
2. Etiquetar todos los ejemplos no etiquetados con el clasificador
3. Añadir los nuevos ejemplos etiquetados con altos índices de confianza a los ejemplos etiquetados originales
4. Repetir hasta que no queden no etiquetados o no se pueda añadir ningún nuevo dato

Este enfoque también se conoce como *self-teaching* o *bootstrapping*.

Algoritmo de Self-training

L = ($\mathbf{x}_i$, $\mathbf{y}_i$); *ejemplos etiquetados*
U = ($\mathbf{x}_i$, ?); *ejemplos no etiquetados*
T; *umbral de confianza*
while U $\neq \emptyset$ or **U'** $\neq \emptyset$ **do**
  Entrena un clasificador **C** con **L**
  Clasifica **U** con **C**
  Encuentra un subconjunto **U'** de **U** con alto nivel de confianza (confianza $>$ **T**)
  L + U' $\Rightarrow$ **L**
  U - U' $\Rightarrow$ **U**
end while

Co-Training

- Co-entrenamiento (o *co-training*) supone que existen dos conjuntos de atributos independientes y compatibles para los datos (dos vistas de los datos)
- Cada conjunto (o vista) de atributos es suficiente para aprender un clasificador adecuado
- Se aprende un clasificador con cada subconjunto de atributos (redundantes) y se usa para clasificar datos para el otro clasificador y aumentar su conjunto de entrenamiento

Co-Training

- Encontrar dos conjuntos independientes y redundantes de atributos puede no ser factible en aplicaciones reales
- Otra opción (la más usada) es aprender dos clasificadores diferentes para el mismo conjunto de entrenamiento
- La hipótesis es que como cada clasificador construye su modelo usando diferentes formas para representar sus modelos, estos dos modelos diversos pueden complementarse
- Los dos clasificadores se usan tanto para clasificar ejemplos nuevos combinando los clasificadores, como para etiquetar ejemplos del otro clasificador

Co-Training

- Supongamos que tenemos dos clasificadores A y B. Sea U el conjunto de ejemplos no-etiquetados y L el conjunto de etiquetados, sean H_A y H_B los clasificadores o hipótesis construidas con cada clasificador y l_A y h_A los intervalos de confianza bajo y alto del clasificador H_A (lo mismo para H_B), y sea w_A un estimado del número de ejemplos en L_A mal etiquetados
- La idea es clasificar ejemplos no vistos con el modelo más confiable para ese ejemplo en particular

Co-Training

Se calcula para L usando 10-*fold cross validation* intervalos de confianza (con 95%) para $H_A, [l_A, h_A]$ y $H_B, ([l_B, h_B])$.

for cada ejemplo x a clasificar **do**

Sea z_A la *clase de equivalencia* de H_A que contiene x (esto es, la rama o regla o ..., que clasifica a x).

Calcula los intervalos de confianza (con 95%) para los ejemplos que están en $z_A, ([l_{zA}, h_{zA}]$ y lo mismo para la clase de equivalencia de $H_B, z_B([l_{zB}, h_{zB}])$

Llama a Condiciones

end for

Condiciones

if $(l_A + h_A)/2 > (l_B + h_B)/2$ {esto es, que H_A en promedio es más precisa que H_B} **then**
  if $z_A = \emptyset$ **then**
    if $(l_A + h_A)/2 - \max\{(l_B + h_B)/2, (l_{zB} + h_{zB})/2\} > 0.1$ **then**
      predice con $H_A(x)$
    else
      predice con $H_B(x)$
    end if
  else if
  $((l_A + h_A)/2 > (l_{zB} + h_{zB})/2) \vee ((l_{zA} + h_{zA})/2 > (l_{zB} + h_{zB})/2)$
  then
    predice con $H_A(x)$
  else
    predice con $H_B(x)$
  end if
else
  ... (lo mismo para B con H_B más precisa que H_A)
end if

Co-Training

- Lo anterior supone que ya se construyeron los clasificadores usando los ejemplos etiquetados y no etiquetados
- La otra parte tiene que ver con cómo etiquetar un ejemplo con uno de los modelos construidos hasta ese momento, digamos H_A, y ponerlo en L_B para mejorar el modelo de H_B
- Además de que la confianza en la etiqueta del ejemplo tiene que ser mayor para A que B, tenemos que compensar por el posible error en la etiquetada que pueda hacer H_A

Co-Training

- Para esto se utiliza una fórmula que relaciona el error ϵ, el tamaño de la muestra m y la razón de ruido en la clasificación η (*classification noise rate*) como sigue:

$$m = \frac{c}{\epsilon^2(1-2\eta)^2}$$

  m es el número de ejemplos de entrenamiento, por ejemplo para B, $m = |L \cup L_B|$ y el estimador de η también para B, es $\eta = w_B/|L \cup L_B|$.

- De aquí, un estimador para el inverso del error al cuadrado de B:

$$1/\epsilon_B^2 = q_B = |L \cup L_B| \left(1 - \frac{2w_B}{|L \cup L_B|}\right)^2$$

  donde $w_B = (l_B + h_B)/2$ (lo mismo para w_A).

Algoritmo de co-training

Corre A en $L \cup L_A$ para obtener H_A
Encuentra un estimador para $1/\epsilon_A^2$:
$q_A = |L \cup L_A| \left(1 - \left(\frac{2w_A}{|L \cup L_A|}\right)^2\right.$ (lo mismo para B)
for cada clase equivalente z definida por H_A **do**
  Sea U_z los ejemplos en U (no etiquetados) que mapean a z
  Usa L y 10-*fold cross-validation* para calcular con 95% de confianza los intervalos para z, $[l_z, h_z]$
  if $h_z > l_B$ **then**
    llama Subrutina
  end if
  Se hace lo mismo para B
end for

Subrutina co-training

$w_z = (1 - l_z)|U_z|$ {estimador conservador de ejemplos en U_z mal clasificados por H_A}

$q_z = |L \cup L_B \cup U_z| \left(1 - \frac{2(w_B + w_z)}{|L \cup L_B \cup U_z|}\right)^2$

if $q_z > q_B$ {la razón de error estimado para B decrece si los ejemplos en U_z son etiquetas por A} **then**

  Sea L_z ejemplos en U_z etiquetados por H_A

  $L_B = L_B \cup L_z$

  $w_B = w_B + w_z$

end if

Democratic Co-Training

- Recientemente se hizo una extensión a este algoritmo (*Democratic Co-training*) para utilizar un conjunto de clasificadores (≥ 3).
- La idea es usar varios clasificadores para etiquetar datos de otro clasificador usando voto mayoritario del resto de los clasificadores.
- Además de usar las pruebas anteriores, ahora se verifica que el promedio de los valores de confianza de los clasificadores del voto mayoritario sea mayor que el promedio de los valores de confianza de los clasificadores del voto minoritario

Assemble

- La idea de ASSEMBLE (*Adaptive Semi-Supervised enSEMBLE*) es constuir un ensamble de clasificadores que trabaje en forma consistente tanto con ejemplos etiquetados como no etiquetados.
- La idea es maximizar el margen (*margin*) tanto con ejemplos etiquetados como no etiquetados. Para esto, se introduce el concepto de pseudo-clase para los ejemplos no etiquetados.
- Recordando un poco de AdaBoost. Sean $f_j(x)$ los clasificadores base y supongamos que se tienen dos clases $+1, -1$. El ensamble $F(x)$ se forma por la combinación lineal de N clasificadores: $F(x) = \sum_{i=1}^{N} \alpha_i f_i(x)$, donde α_i es el peso asociado al i-ésimo clasificador.

Assemble

- La pseudo-clase de un ejemplo no etiquetado es la clase predicha por el ensamble hasta ese momento: $y_{x_i} = F(x_i)$ para $x_i \in U$.
- Las pseudo-clases para los datos iniciales se pueden obtener usando vecinos más cercanos o clase mayoritaria en los ejemplos etiquetados.
- Inicialmente se les da más peso a los ejemplos etiquetados.
- La idea es, al igual que AdaBoost, construir una secuencia de clasificadores, en donde se modifican (aumentan) los pesos de los ejemplos mal clasificados, para que el siguiente clasificador los pueda cubrir. La diferencia es que los ejemplos etiquetados reciben más peso que los no etiquetados.

Algoritmo Assemble

Sea $l = |L|$ y $u = |U|$ y

$$D_1(i) = \begin{cases} \beta/l & \text{si } i \in L \\ (1-\beta)/u & \text{si } i \in U \end{cases}$$

Sea $y_i = c$ {c es la clase del vecino más cercano en L a $i \in U$}

Sea $f_1 = \mathcal{L}(L+U, Y, D_1)$ {$\mathcal{L}$ es el algoritmo de aprendizaje}

for $t = 1$ to T **do**

Sea $\hat{y}_i = f_t(x_i), i = 1, \ldots, l+u$ {clasificación con el ensamble actual}

$\epsilon = \sum_i D_t[y_i \neq \hat{y}_i], i = 1, \ldots, l+u$ {errores pesados}

if $\epsilon > 0.5$ **then**

Stop

end if

$w_t = 0.5 * log(\frac{1-\epsilon}{\epsilon})$ {peso para el clasificador}

end for

Algoritmo Assemble (continuación)

Sea $F_t = F_{t-1} + w_t f_t$ {nuevo ensamble}
Sea $y_i = F_t(x_i)$ si $i \in U$ {posiblemente nueva clasificación para los ejemplos no etiquetados con el nuevo ensamble}
{cálculo de nuevos pesos a los ejemplos}

$$D_{t+1} = \frac{\alpha e^{-y_i F_{t+1}(x_i)}}{\sum_j \alpha_j e^{-y_j F_{t+1}(x_j)}} \text{ para toda } i$$

S = Muestrea($L + U, l, D_{t+1}$) {muestreo nuevo de datos}

Assemble

- Se usó $\beta = 0.9, \alpha = 1$ en todos los experimentos. También se mantiene constante el número de ejemplos (igual al tamaño total de los ejemplos etiquetados), aunque el muestreo es sobre todos los ejemplos ($L + U$).
- Una alternativa a ASSEMBLE es usar un clasificador naïve Bayes de base y cambiar los pesos de los ejemplos no etiquietados de acuerdo a su probabilidad de clase.

Algoritmo

Algoritmo 4.1 Semi-Supervised AdaBoost (SA).

Entrada: L: Instancias Etiquetadas, U: Instancias No Etiquetadas, T: Iteraciones

Salida: Hipótesis final y probabilidades: $H_f = argmax_{y \in Y} \sum_{t=1}^{T} log\frac{1}{B_t} \cdot h_t(x, y)$,

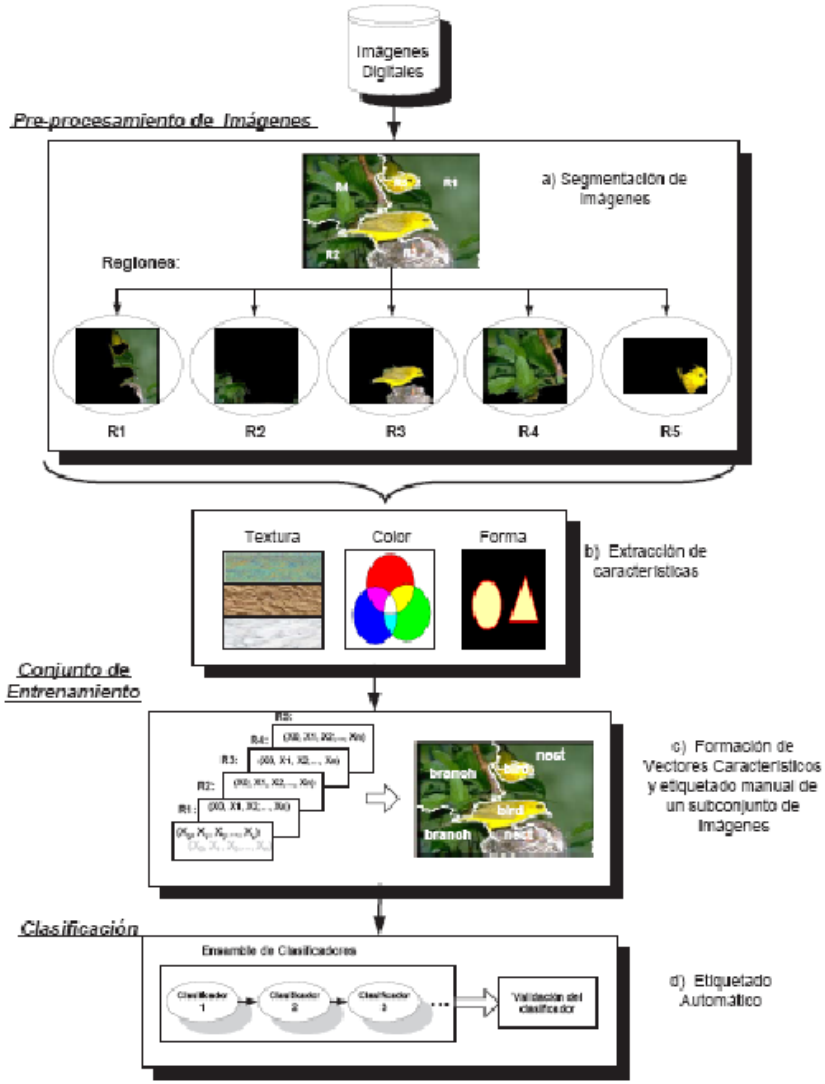
1: $W_0(x_i) = \frac{1}{|L|}, \quad \forall x_i \in L$ {Pesos iniciales para L}
2: $h_1 = C(L, W_0(x_i))$ {Clasificador inicial}
3: Porcentaje de error sobre L: $e_1 = \sum_{i=1}^{N} W_0(x_i) \quad \text{if } h_1(x_i) \neq y_i, \forall x_i \in L$
4: $B_1 = \frac{e_1}{(1-e_1)}$
5: $W_1(x_i) = W_0(x_i) \cdot B_1 \quad \text{if } h_1(x_i) = y_i, \forall x_i \in L$
6: $W_1(x_i) = \frac{1}{|U|} \quad \forall x_i \in U,$
7: **for** t = 2 hasta T **do**
8: $\quad W_t(x_i) = \frac{W(x_i)}{\sum_{i=1}^{N} W(x_i)} \forall x_i \in L \cup U$ {Pesos Normalizados}
9: $\quad h_t = C(L \cup U, W_t(x_i))$
10: $\quad e_t = \sum_{i=1}^{N} W_t(x_i) \quad \text{if } h_t(x_i) \neq y_i$
11: $\quad$ **if** $e_t \geq 0.5$ **then**
12: $\quad\quad$ Salir del ciclo
13: $\quad$ **end if**
14: $\quad B_t = \frac{e_t}{(1-e_t)}$
15: $\quad W_{t+1}(x_i) = W_t(x_i)^t \cdot B_t \quad \text{if } h_t(x_i) = y_i \quad \forall x_i \in L$
16: **end for**

Assemble

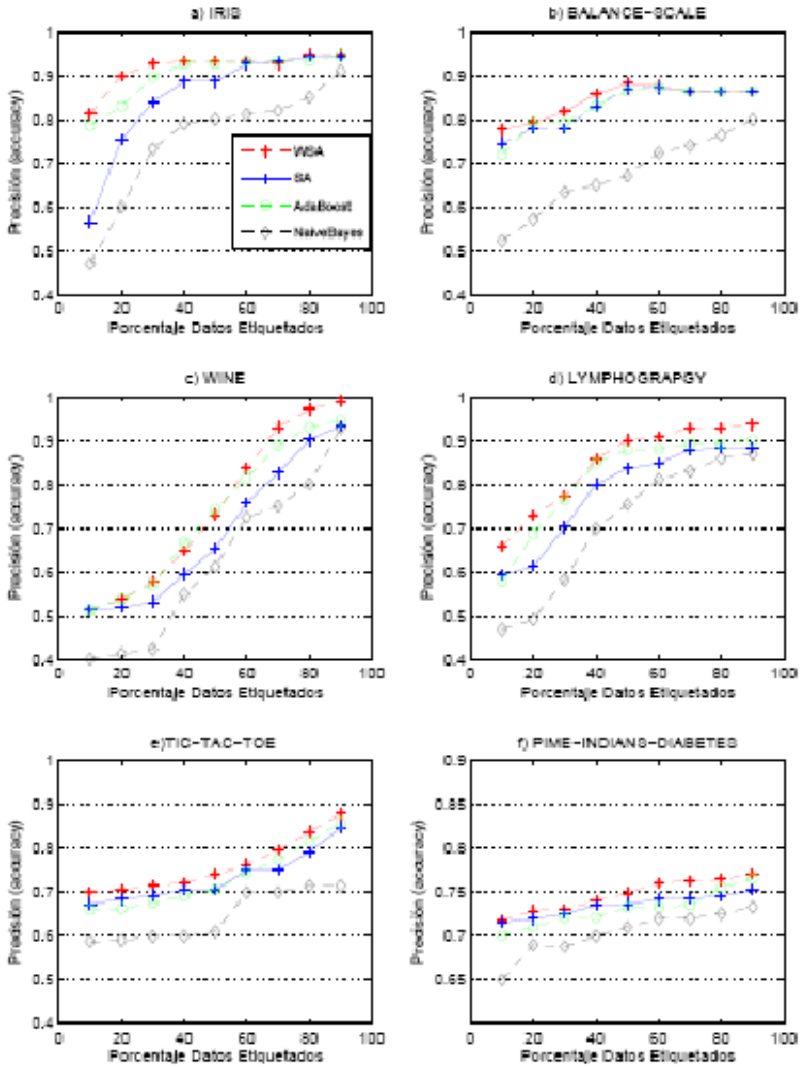
El algoritmo considera los siguientes aspectos:

- Los pesos iniciales de los ejemplos dependen del inverso del número de datos (suponiendo más datos no etiquetados que etiquetados)
- Primero se usan todos los ejemplos para construir un nuevo clasificador
- Los errores en los ejemplos se usan para pesar los ejemplos etiquetados
- Los pesos de los ejemplos no etiquetados son proporcionales a su probabilidad de clase
- Los no etiquetados se re-etiquetan en cada ciclo
- Los pesos de los no etiquetados del ciclo anterior no se consideran en el ciclo actual

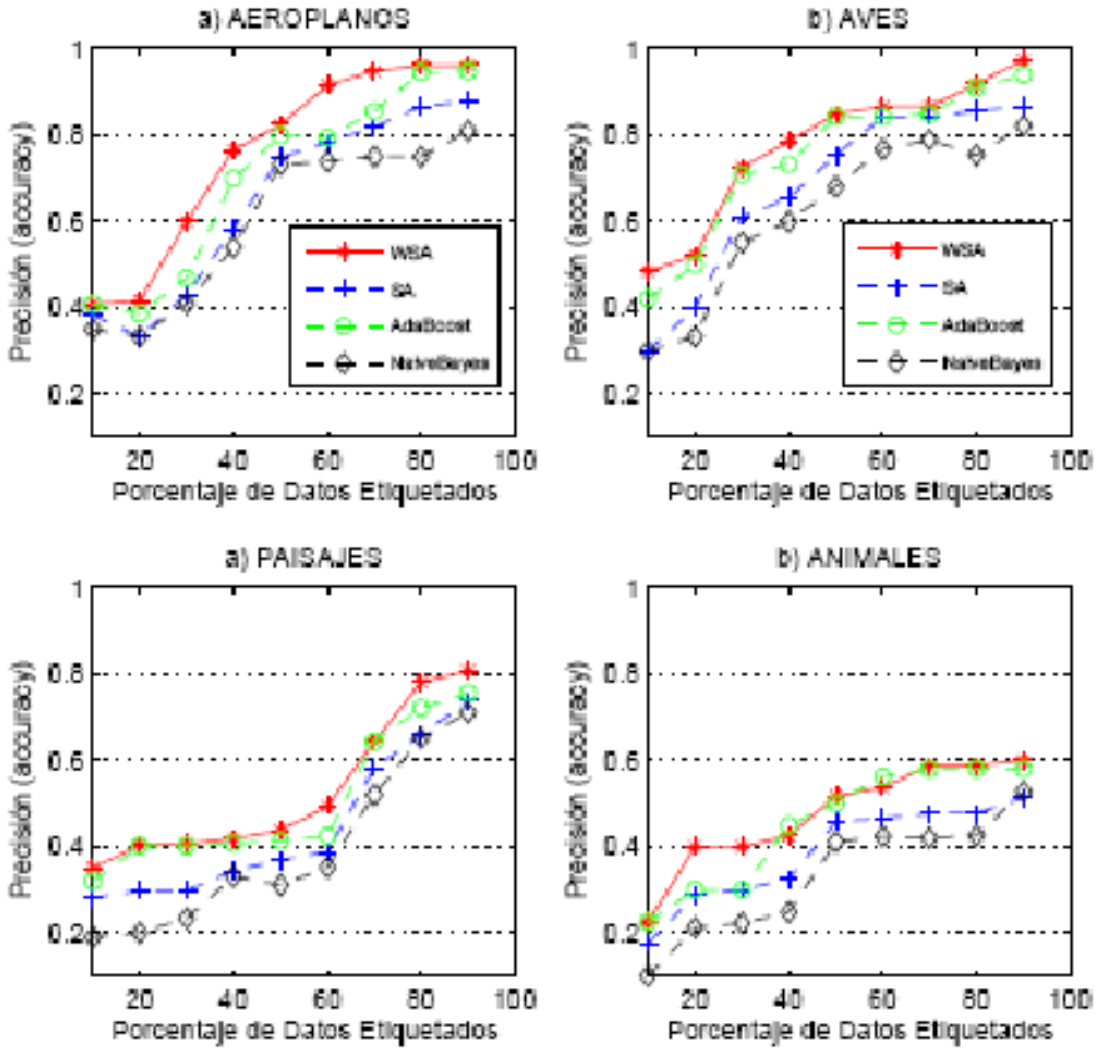
Esquema para etiquetar imagenes



Resultados



Resultados



Re-Weighting

- Supone que la distribución de clases en los ejemplos etiquetados es la misma que en los no etiquetados
- Existen diferentes enfoques, el que vamos a ver se llama extrapolación (*extrapolation*)
- Lo primero que se hace es que estima la distribución de las clases tomando en cuenta los ejemplos etiquetados

Re-Weighting

- Se aprende un modelo con los ejemplos etiquetados y se aplica a los ejemplos no etiquetados
- Los ejemplos etiquetados y no etiquetados se agrupan de acuerdo al valor de probabilidad de que se tenga esa clase
- Los ejemplos no etiquetados en cada grupo se etiquetan de acuerdo a la distribución de las clases de los ejemplos etiquetados en ese grupo

Re-Weighting

- Por ejemplo, supongamos que tenemos la siguiente tabla:

Grupo	No Etiq.	Etiq.	Clase 0	Clase 1
0.6 - 0.7	20	100	10	90

- Lo que nos dice es que en el grupo que tienen probabilidad entre 0.6 y 0.7 se encuentran 20 ejemplos no etiquetados y 100 etiquetados, de los cuales 10 son de clase 0 y 90 de clase 1
- El peso del grupo se define como: $|L + U|/|L| = 120/100 = 1.2$.
- Se aplica este peso a cada clase: NClase0 $= 1.2 * 10 = 12$ y NClase1 $= 1.2 * 90 = 108$

Re-Weighting

- Luego se etiquetan de forma aleatoria los ejemplos no etiquetados siguiendo esta distribución, esto es, se seleccionan 2 ejemplos $(12 - 10)$ de los no etiquetados en forma aleatoria y se les asigna la clase 0 y a 18 $(108 - 90)$ se les asigna la clase 1
- Una vez que los ejemplos se etiquetaron, se aprende un nuevo modelo con todos los ejemplos $(U + L)$

Basados en EM y Componentes Comunes

- La idea básica es entrenar un clasificador usando sólo ejemplos etiquetados, y usar ese clasificador para asignar probabilidades-pesos de etiquetas a los ejemplos no etiquetados. Después entrenar un nuevo clasificador usando esas estimaciones y volver a asignar pesos.
- Aunque los datos no etiquetados por sí solos no nos dan más información que una aleatoria si no sabemos la clase, la información de los valores de los atributos nos proporcionan información útil sobre la distribución de sus valores.
- En un enfoque probabilístico, muchos de los parámetros de los modelos se pueden estimar sin necesidad de conocer la clase.

Basados en EM y Componentes Comunes

- Usando un clasificador Bayesiano naïve.

$$c_{MAP} = argmax_{c_j \in C} \left(P(c_j \mid a_1, \ldots, a_n) \right)$$

- Usando Bayes:

$$c_{MAP} = argmax_{c_j \in C} \left(\frac{P(a_1,\ldots,a_n|c_j)P(c_j)}{P(a_1,\ldots,a_n)} \right)$$

$$= argmax_{c_j \in C} \left(P(a_1, \ldots, a_n \mid c_j) P(c_j) \right)$$

- $P(c_j)$ se puede estimar con la frecuencia de las clases, pero para $P(a_1, \ldots, a_n \mid c_j)$ tenemos muy pocos elementos

Basados en EM y Componentes Comunes

- Suponiendo independencia de atributos dada la clase (naïve Bayes):

$$P(a_1, \ldots, a_n \mid c_j) = \prod_i P(a_i \mid c_j)$$

- Por lo que:

$$c_{NB} = argmax_{c_j \in C} \left(P(c_j) \prod_i P(a_i \mid v_c) \right)$$

Basados en EM y Componentes Comunes

Los valores $P(a_i \mid c_j)$ se pueden estimar usando:

- La frecuencia de los datos observados: $P = \frac{Datos\_Valor}{Total\_Datos}$
- Estimador Laplaciano (distribución uniforme de las k clases):

$$P = \frac{Datos\_Valor + 1}{Total\_Datos + k}$$

- Estimador m: considera que las distribuciones *a priori* de las clases ($P_a(C)$), son independientes del número de clases y m es dependiente del dominio (entre más ruido, se selecciona una m mayor).

$$P = \frac{Datos\_Valor + m \cdot P_a(C)}{Total\_Datos + m}$$

Algoritmo básico EM usando naïve Bayes.

Dados D^L (datos etiquetados) y D^U (datos no etiquetados)
Creo un clasificador Bayesiano naïve inicial con D^L
while NOT criterio de paro **do**
(Paso-E): Usa el clasificador ($\hat{\Theta}$) - parámetros del clasificador - para estimar la clase de D^U
(Paso-M): Re-etima los parámetros ($\hat{\Theta}$) usando todos los datos
($D^L \cup D^U$ etiquetados)
end while

Algoritmo básico de EM usando naïve Bayes

- El criterio de paro se establece cuando la estimación de Θ no cambia
- El algoritmo está haciendo una búsqueda *hill-climbing* partiendo de un punto inicial (tomando en cuenta el modelo generado por los datos etiquetados) y por lo mismo puede caer en un mínimo local
- Uno de los problemas en aprendizaje semi-supervisado, es que los datos no etiquetados pueden ser mucho más que los etiquetados y si el modelo no es adecuado nos pueden hacer más daño
- Esto es, podemos obtener mejores resultados sólo con los ejemplos etiquetados

Algoritmo mejorado de EM usando naïve Bayes

- Una alternativa es introducir un peso ($\lambda < 1$) a los datos no etiquetados.
- A las estimaciones de probabilidad se le incorpora un nuevo componente:

$$K(i) = \begin{cases} \lambda & \text{si } d_i \in D^U \\ 1 & \text{si } d_i \in D^L \end{cases}$$

- El resto del algoritmo se queda igual

Algoritmo mejorado de EM usando naïve Bayes

- Una mejora es variar dinámicamente los pesos de los no etiquetados de acuerdo a la probabilidad de su clase predicha por el clasificador. Para naïve Bayes:

$$p(C) = \frac{1 + \sum_{i=1}^{|D_l|} g(y_i, c)}{|C| + |D_l|} + \frac{1 + \sum_{i=1}^{|D_u|} \lambda_i g(y_i, c)}{|C| + |D_u|}$$

$$p(A_{vk}|C) = \frac{1 + \sum_{i=1}^{|D_l|} f(y_i, c, A_{vk})}{|V| + \sum_{i=1}^{|D_l|} g(y_i, C)} + \frac{1 + \sum_{i=1}^{|D_u|} \lambda_i f(y_i, c, A_{vk})}{|V| + \sum_{i=1}^{|D_u|} g(y_i, C)}$$

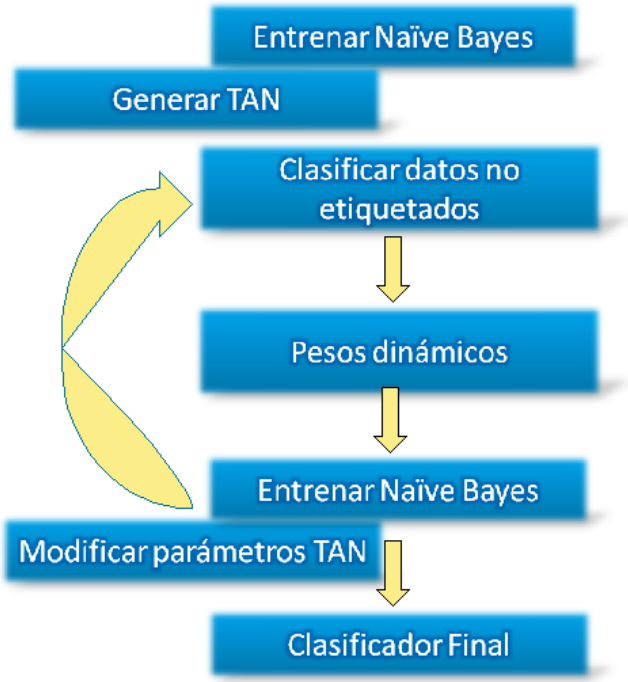
Algoritmo mejorado de EM usando naïve Bayes

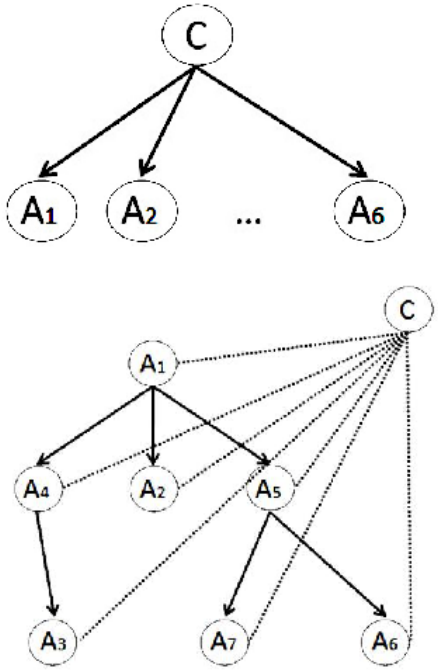
Y para TAN:

$$p(A_{vk}|A_{ul},C) = \frac{1+\sum_{i=1}^{|D_l|} h(y_i,c,A_{vk},A_{ul})}{|V|+\sum_{i=1}^{|D_l|} f(y_i,C,A_{ul})} + \frac{1+\sum_{i=1}^{|D_u|} \lambda_i h(y_i,c,A_{vk},A_{ul})}{|V|+\sum_{i=1}^{|D_u|} f(y_i,C,A_{ul})}$$

- Los primeros términos son de los datos etiquetados, los segundos de los etiquetados y las λ's son los pesos proporcionales a la probabilidad de la clase

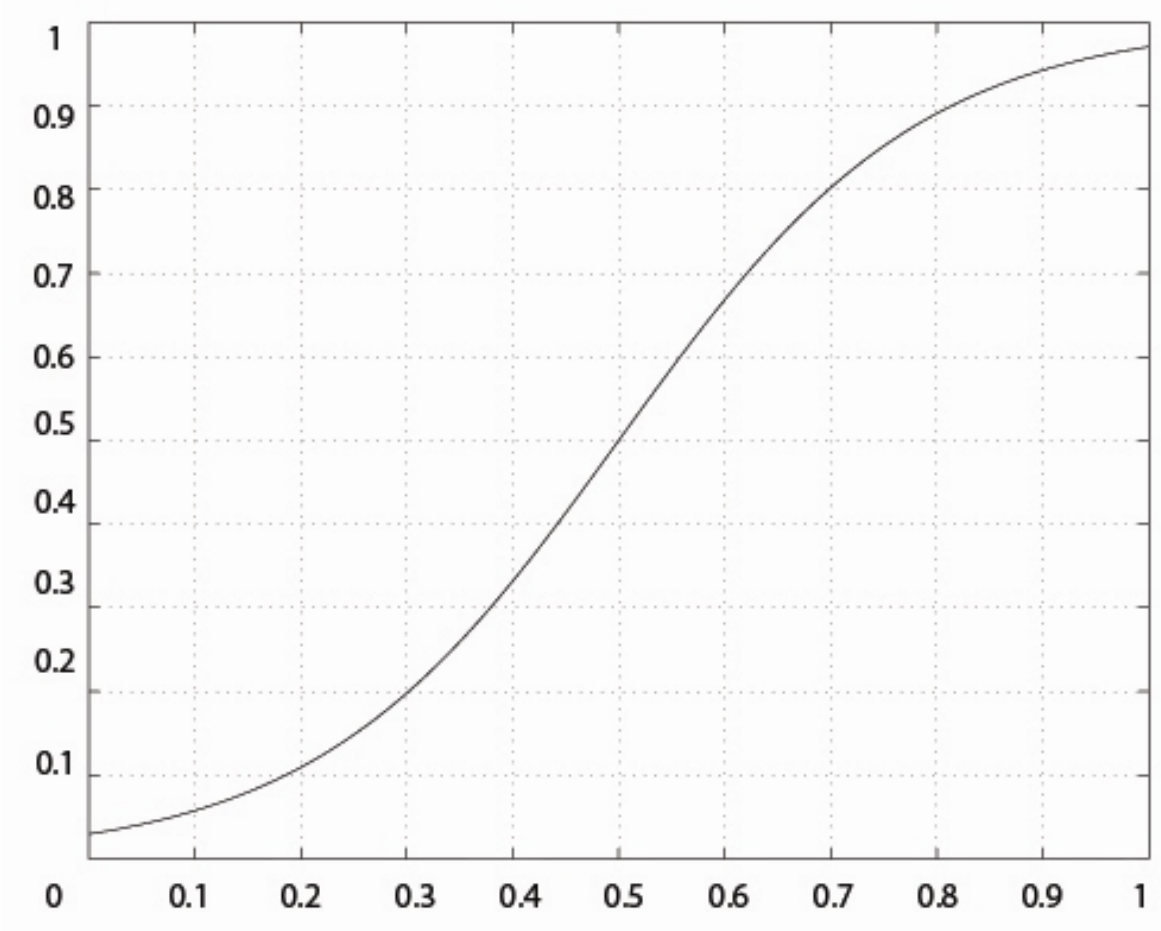
Algoritmo mejorado de EM usando naïve Bayes



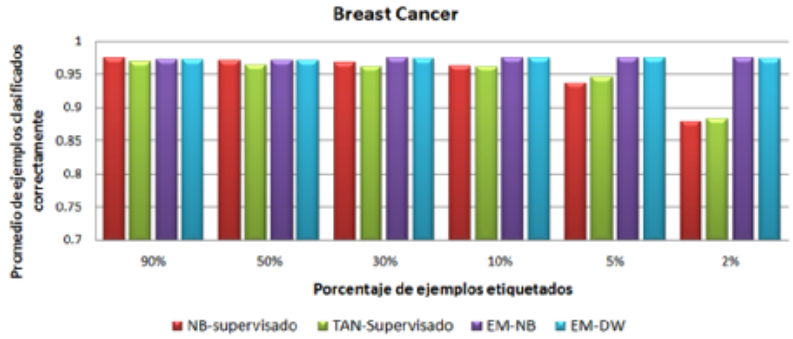


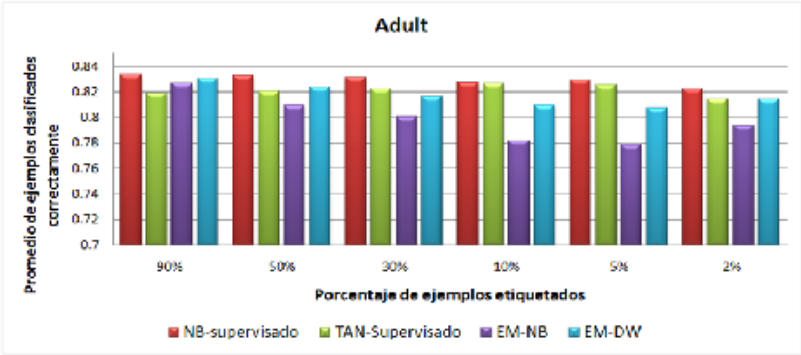
Función para asignar pesos que se usó

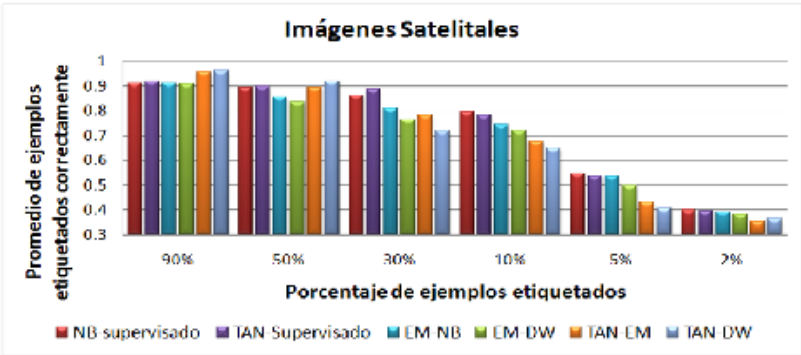
Usar directamente la probabilidad como peso, hace que algunos pesos sean muy parecidos, por lo que se amplia la diferencia entre probabilidades altas y bajas



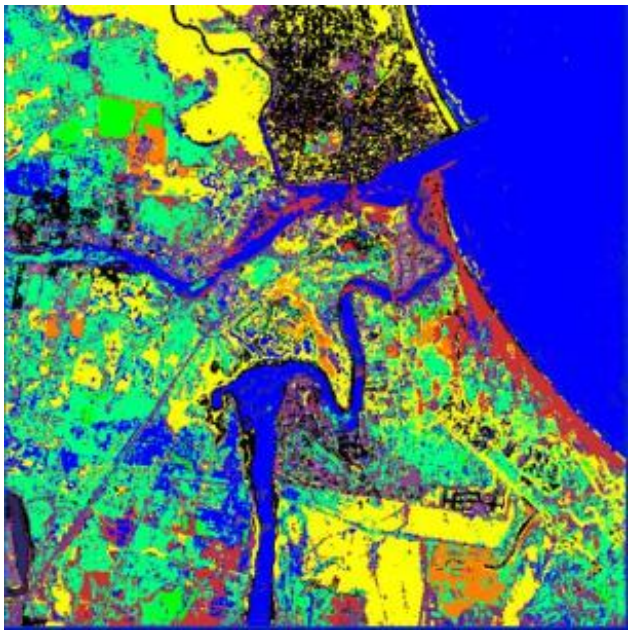
Resultados







Resultados



Otras Extensiones a Aprendizaje Semisupervisado

- Otra posible extensión es no suponer que cada componente del modelo corresponde a una clase, sino que las clases están determinados por un conjunto de componentes
- En clasificación de textos es como si se tuvieran sub-temas, e.g., la clase *deportes* consiste de documentos de *futbol* y de *beisbol*
- En estos documentos, *portería* y *balón* son más probable que co-ocurran en unos documentos, mientras que *bat* y *base*, co-ocurriran en otros
- Para los datos no etiquetados las cosas quedan igual, pero ahora para los etiquetados tenemos que estimar la mezcla de componentes que determinan la clase

Otras Extensiones a Aprendizaje Semisupervisado

- La mezcla de componentes para cada clase se inicializa aleatoriamente
- La probabilidad de un cierto tópico (sub-tema) dados los datos y el modelo es:

$$P(t_a|d_i,\hat{\Theta}) = \sum_{c_j} P(t_a|c_j,\hat{\Theta})P(c_j|d_i,\hat{\Theta})$$

- Después se clasifica sumando estas probabilidades hacia las clases
- El número de componentes se puede estimar usando una validación cruzada

Otros Enfoques

Entre otros esquemas, podemos mencionar *transductive* SVM y los modelos basados en grafos.

Transductive SVM

- En SVM la idea es encontrar un hiperplano que mejor divida a los ejemplos de entrenamiento, en el que existe el margen de separación más grande
- En TSVM se usan también los ejemplos no etiquetados. La meta es encontrar un etiquetado a los ejemplos no etiquetados para encontrar un frontera con el margen máximo en todos los ejemplos
- En forma intuitiva, los ejemplos no etiquetados guían la definición de la frontera en las regiones poco densas (de ejemplos etiquetados)

Transductive SVM

- El aprendizaje transductivo sólo sirve con los ejemplos etiquetados y no etiquetados proporcionados, pero no puede servir para clasificar ejemplos no vistos
- Recientemente se desarrolló un algoritmo llamado *Semi-Supervised Support Vector Machines* o S^3VM que es capaz de predecir la clasificación de nuevos ejemplos

Modelos basados en Grafos

- Estos modelos definen un grafo en donde los nodos son los ejemplos etiquetados y los no etiquetados y los arcos (que pueden estar pesados), reflejan la similaridad entre los ejemplos
- Tratan de estimar una función en el grafo que satisfaga al mismo tiempo: (i) ser cercana a las etiquetas de los nodos etiquetados y (ii) "suave" sobre todo el grafo
- Son generalmente transductivos, i.e., no pueden extenderse a ejemplos no contemplados en el conjunto de etiquetados y no etiquetados, aunque se han realizado extensiones para que cada ejemplo nuevo no altere el grafo y se clasifique por vecinos más cercanos
- Recientemente también se ha trabajado en *clustering*, regresión y aprendizaje activo

Comentarios

En este tipo de modelos se ha mostrado que:

- Los datos etiquetados y no etiquetados reducen la varianza en el clasificador
- Cuando el modelo es correcto el estimador no está sesgado y se reduce el error al reducir la varianza
- Cuando el modelo es incorrecto, el aumento en ejemplos no etiquetados reduce la varianza, pero al estar sesgado por el modelo incorrecto, aumenta el error

Comentarios

- Si las suposiciones iniciales para construir los modelos son incorrectas, el aprendizaje semi-supervisado puede perjudicar en forma importante el desempeño del clasificador
- Existen dos factores principales:
  1. El modelo es inadecuado
  2. La distribución de los datos no es adecuada

Comentarios

Para la primera suposición incorrecta lo que se puede hacer es cambiar el modelo, una posible estrategia es la siguiente:

1. Aprende un clasificador sólo con los datos etiquetados
2. Aprende un clasificador con los datos etiquetados y no etiquetados
3. Compara su desempeño, si es peor el clasificador con datos no etiquetados, entonces cambia el modelo (e.g., pasar de naïve Bayes a TAN)

Otra posibilidad es usar pesos dinámicos como se mencionó en las extensiones a ASSEMBLE y a EM.