Comités de árboles IGP

Gonzalo Martínez-Muñoz
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Campus Cantoblanco
28049 Madrid
gonzalo.martinez@uam.es

Alberto Suárez
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Campus Cantoblanco
28049 Madrid
alberto.suarez@uam.es

Resumen

El presente trabajo desarrolla un nuevo método de generación de conjuntos de clasificadores que combina la estabilidad frente al ruido de algoritmos basados en *bagging* con la agresividad para la reducción del error de generalización de algoritmos de *boosting*. Para ello se han utilizado comités generados con *boosting* como clasificadores base de un conjunto de clasificadores similar a *bagging* pero específico para trabajar con árboles de decisión IGP. La decisión final del conjunto se hace en dos etapas: primero cada comité seleccionará una clase mediante voto y posteriormente las decisiones de cada comité se combinarán de nuevo mediante voto para obtener la decisión final. La capacidad de generalización de este método de construcción de conjuntos de clasificadores es comparable o mejor que *boosting* y significativamente mejor que *bagging*.

1. Introducción

Dentro del campo de aprendizaje supervisado, el desarrollo de conjuntos de clasificadores es un tema de gran actividad ([1], [3], [4], [5], [6], [8], [10], [12], [13], [14], [15], [16]). Esta gran actividad se debe sobre todo a las significativas mejoras en la precisión de clasificación que se obtienen con esta técnica de sencilla implementación. Un conjunto de clasificadores clasifica nuevos ejemplos combinando la acción sus componentes mediante voto ponderado o sin ponderar, para obtener una clasificación final. Normalmente, de esta combinación resulta que el conjunto de clasificadores tiene más precisión que cada uno de los clasificadores en los que esta compuesto. En un problema de clasificación con dos clases, basta que los errores de los clasificadores individuales estén por debajo de 0,5 para que la combinación de clasificadores sea generalmente más precisa que la de cada uno de los componentes por separado. Si, además, los errores de los clasificadores no están correlacionados, entonces el error del conjunto tiende a 0 a medida que crece el número de clasificadores. Por tanto, lo importante es diseñar un procedimiento para generar una diversidad de clasificadores cuyos errores no estén correlacionados. Se han desarrollado gran cantidad de algoritmos para la generación de conjuntos de clasificadores. En [9] se hace una agrupación de los distintos métodos en 4 categorías.

Dentro de estas categorías el grupo de técnicas más difundidas consiste en el muestreo de los datos de entrenamiento. Básicamente, estas técnicas producen tantas modificaciones de la distribución de los datos de entrenamiento como clasificadores individuales se quieran obtener. Cada perturbación en la distribución de entrenamiento (que se supone debe parecerse a la distribución real) produce variaciones en los clasificadores individuales y así se obtiene la variabilidad requerida dentro del conjunto. Tenemos por tanto que, y para que se cree la suficiente variabilidad entre clasificadores, el algoritmo de construcción de clasificadores debe tener una cierta inestabilidad ante los cambios de muestreo (p.e. árboles de decisión, redes neuronales). Algunos de los métodos de generación de conjuntos de clasificadores más difundidos caen en esta categoría, como son: *boosting* [10], *wagging* [1] y *bagging* [6].

Boosting y *wagging* modifican el muestreo original mediante la asignación de pesos a los ejemplos. *Boosting* lo hace de forma adaptativa, aumentando los pesos de los ejemplos mal clasificados por el clasificador anterior y reduciendo los pesos de los bien clasificados. *Wagging* asigna pesos aleatoriamente a los

ejemplos para construir cada clasificador individual. Tanto *boosting* como *wagging* utilizan todos los ejemplos para construir cada clasificador, sin embargo, los ejemplos, dado que están ponderados, no tienen todos la misma importancia para el algoritmo de clasificación. *Boosting* ha demostrado ser uno de los métodos más eficientes para la construcción de conjuntos de clasificadores, a pesar de tener dificultades de generalización cuando los datos tienen ruido [8].

Otra técnica ampliamente utilizada es *bagging* (*Bootstrap sampling and aggregation*) [6]. *Bagging* no utiliza ningún tipo de ponderación de los datos. Cada clasificador del conjunto se crea utilizando una muestra aleatoria con repetición del mismo número de ejemplos que el conjunto de datos de entrenamiento. *Bagging* es generalmente más robusto que *boosting* frente a ruido o fallos en las etiquetas de clase de los ejemplos. En la ref. [8] se hace un estudio de cómo afecta el ruido de clase a distintos conjuntos de clasificadores. De este estudio *Bagging* se muestra claramente como el algoritmo más eficiente para afrontar problemas con ruido.

El presente artículo presenta un nuevo método que combina la estabilidad frente al ruido de los algoritmos basados en *bagging* y a capacidad de *boosting* para reducir el error de generalización y conseguir así un algoritmo aplicable a un mayor número de problemas. Para ello se ha desarrollado un conjunto de clasificadores basado en *bagging* que tiene como algoritmo base a *boosting*. Sin embargo no hemos usado *bagging* directamente sino un algoritmo similar a *bagging* [13] diseñado para funcionar de forma específica con el algoritmo de construcción de árboles denominado *Iterative Growing and Pruning Algorithm* (IGPA) [11]. Este algoritmo permite, aprovechando la variabilidad intrínseca de IGPA, utilizar todos los datos de entrenamiento para crear cada árbol y así conseguir resultados ligeramente mejores que *bagging* [13].

El artículo está estructurado del siguiente modo: En la sección 2 se hace una pequeña introducción a la notación y a los algoritmos de clasificación *bagging* y *boosting*, en la sección 3 se describe el algoritmo de construcción de árboles IGP y conjuntos de clasificadores basados en IGP, en la sección 4 se presenta el nuevo algoritmo desarrollado, en la sección 5 se describen los resultados experimentales y finalmente en la sección 6 se exponen las conclusiones de este trabajo.

2. Notación y conjuntos de clasificadores

Un problema de clasificación supervisada consiste en una colección L de N ejemplos etiquetados que podemos definir como:

$$L = \{(\boldsymbol{x}_n, y_n),\ n=1,2,...,N,\ y \in \{y^1, y^2, ..., y^Y\}\} \qquad (1)$$

donde cada ejemplo $(\boldsymbol{x}_n, y_n)$ viene determinado por un vector de atributos $\boldsymbol{x}_n$ y una etiqueta de clase y_n perteneciente a alguna de las Y clases del problema $[y^1, y^2, ..., y^Y]$. El objetivo del algoritmo de clasificación será el de construir un clasificador que prediga, dado un nuevo vector de características $\boldsymbol{x}$ (no incluido en L), la clase 'y' a la que pertenece usando el conocimiento contenido en el conjunto de datos inicial L. En los algoritmos basados en conjuntos de clasificadores en vez de tener un único clasificador se generan muchos clasificadores "complementarios". La decisión final se toma combinando las decisiones individuales de cada clasificador generalmente mediante voto directo o voto ponderado. El punto crítico es la generación de un conjunto de clasificadores que se complementen para así aumentar la capacidad de generalización.

En *bagging* (*Bootstrap sampling and aggregation*), desarrollado por Breiman [6], cada clasificador base se crea a partir de una muestra aleatoria con repetición obtenida del conjunto de datos de entrenamiento y con el mismo número de ejemplos que éste (*bootstrap sampling*). Tenemos por tanto que en *bagging* cada clasificador se construye con un subconjunto de los datos originales (en media es 63.2%) y el resto son ejemplos repetidos.

Bagging es considerado como un algoritmo que produce una mejora en la mayoría de los conjuntos de datos con respecto al algoritmo base ([1], [6], [8], [16]). Además es un algoritmo robusto frente a ruido o fallos en las etiquetas de clase de los ejemplos [8].

En *boosting*, desarrollado por Freud y Shapire [10] se hace uso de la clasificación de los clasificadores ya construidos para construir los siguientes. Este proceso adaptativo se consigue asignando pesos a los ejemplos y modificándolos de acorde a los resultados del último clasificador

generado. La modificación de pesos se hace de forma que los ejemplos mal clasificados por un clasificador aumenten en importancia para construir el siguiente clasificador y viceversa.

Boosting ha demostrado ser uno de los métodos más eficientes para la construcción de conjuntos de clasificadores ([1], [8], [14], [16]). Sin embargo, hay una serie de problemas de clasificación donde su rendimiento es inferior a *bagging* ([8], [16]). Donde mayores dificultades encuentra *boosting* es en conjuntos de datos donde los datos tienen ruido, bien porque hay datos mal tomados o bien porque hay ejemplos con la clase mal asignada.

3. Conjunto de clasificadores basados en árboles IGP

En esta sección nos centraremos en una variante del *bagging* [13] que utiliza todos los datos para construir cada clasificador base y donde el clasificador base es el algoritmo de generación de árboles de decisión *Iterative Growing and Pruning Algorithm* (IGPA), desarrollado por Gelfant et. al. [11].

3.1. Algoritmo base, *Iterative Growing and Pruning Algorithm*

Iterative Growing and Pruning Algorithm es un algoritmo desarrollado por Gelfand et al. [11] para la construcción de árboles de decisión. Este algoritmo tiene la propiedad, al igual que CART [7], que usa todos los datos para crecer y para podar el árbol.

Previamente a la construcción de un árbol con IGPA hay que dividir el conjunto de datos L en dos subconjuntos, L_1 y L_2, de aproximadamente igual tamaño y misma distribución de clases. Una vez divididos los datos, el algoritmo IGP utiliza uno de los subconjuntos para hacer crecer el árbol y el otro para podarlo. La secuencia de crecimiento y poda es repetida intercambiando los papeles de los subconjuntos en cada iteración, es decir, primero se crece un árbol T_0 usando L_1, una vez crecido el árbol T_0, éste se poda hasta el tamaño óptimo T^*_0 con respecto al subconjunto L_2. Una vez que la primera poda se ha completado, los papeles de los subconjuntos de datos son intercambiados y se hace crecer un nuevo árbol T_1 a partir de los nodos terminales de T^*_0 utilizando L_2. A continuación T_1 se poda a su tamaño óptimo con respecto a L_1. Los papeles de los subconjuntos de crecimiento y de poda se van intercambiando hasta que dos árboles podados consecutivos sean de igual tamaño. Ha sido demostrado que esta secuencia converge [11] siempre que la clase de cada nodo t se elija, al igual que en CART, por mayoría, pero, en caso de empate, se asigne la clase del nodo padre de t. La idea detrás de la demostración es que después de cada iteración el árbol podado resultante T^*_k siempre es mayor o igual al árbol podado previo T^*_{k-1}.

IGPA utiliza el mismo procedimiento que CART para hacer crecer el árbol basado en el criterio de Gini [7]. Para podarlo aplica un algoritmo de poda que devuelve el árbol más preciso con respecto a un conjunto de datos sin tener en cuenta la complejidad. Este método de poda contrasta por su velocidad con la poda basada en validación cruzada que utiliza CART, en la cual es preciso construir árboles auxiliares para determinar los parámetros de poda.

3.2. Conjuntos con árboles IGP

Este método presentado en [13] (mostrado en Algoritmo 1) se aprovecha de la variabilidad intrínseca del algoritmo IPG con respecto a la división del conjunto de entrenamiento. De esta forma se pueden generar tantos árboles IGP como se deseen a partir de otras tantas particiones aleatorias del conjunto de datos L y obtener así la diversidad necesaria para su combinación dentro de un conjunto. Por tanto se tiene que todos los árboles generados usan todos los datos de entrenamiento y no un muestreo del conjunto original como en *bagging*. Este método obtiene resultados equivalentes o mejores que *bagging* [13].

```
Conjunto IGP
   Entrada:
        Conjunto de datos L.
        Tamaño del conjunto T.

1. Para t=1 hasta T:
2.      Partir L en L1 y L2
3.      Ct = ArbolIGP(L1, L2)

   Salida:
        Conjunto de árboles IGP
```

Algoritmo 1. Conjunto de árboles IGP

4. Algoritmo propuesto. Comités de *boosting* IGP

El algoritmo propuesto fusiona el algoritmo de generación de conjuntos de árboles IGP (descrito en la sección 3.2) con un *boosting* que utiliza como clasificador base el algoritmo IGP. Para ello se ha sustituido el clasificador base por un conjunto de clasificadores generados con *boosting*. De esta forma se pueden complementar dos algoritmos de conjuntos de clasificadores. Por ejemplo, en el multiboosting, la idea de Webb era unir la capacidad de reducir la variabilidad del *wagging* con la capacidad de reducir el sesgo que tiene *boosting* [16].

El algoritmo propuesto consiste en substituir para el paso 3 del Algoritmo 1 por un *boosting* adaptado para funcionar con árboles IGP. A cada uno de estos *boosting*s de clasificadores dentro del conjunto principal se le denomina comité. El algoritmo completo se muestra en el Algoritmo 2. Este algoritmo tiene, a parte del conjunto de datos, otros dos parámetros de entrada. El parámetro $\mathbf{T_1}$ indica el número de clasificadores base que se van a construir, en este caso el número comités que se van a realizar mediante *boosting* con árboles IGP. El parámetro $\mathbf{T_2}$ identifica el número de clasificadores que se quieren construir dentro del *boosting* IGP, es decir, el número de miembros que tendrá cada comité. Como salida este algoritmo devuelve $\mathbf{T_1}$ comités que son los que votan para obtener la clasificación. Cada uno de los comités tiene $\mathbf{T_2}$ árboles IGP. En total se generan $\mathbf{T = T_1 \times T_2}$ árboles.

```
ConjuntoBoostingsIGP
   Entrada:
       Conjunto de datos L
       Comités a construir T1
       No. clasif. por comité T2

1. Para t=1 hasta T1:
2.     Partir L en L1 y L2
3.     Ct = BoostingIGP(L1, L2, T2)

   Salida:
       Conjunto de comités
```

Algoritmo 2. Algoritmo propuesto. Comités de *boosting* IGP

El algoritmo *boosting* implementado para trabajar con árboles IGP (paso 3 del Algoritmo 2) funciona igual que Adaboost [10] con la salvedad de que las actualizaciones de pesos de los datos de entrenamiento se hacen separadamente para L_1 y L_2. De esta forma la suma de los pesos de cada uno de los conjuntos se mantiene a la mitad del peso total del conjunto y se evita que los subconjuntos se descompensen lo que produce efectos indeseables en la construcción de los árboles IGP.

La decisión final del conjunto se toma en dos etapas. Primero, cada comité toma una decisión consultando a sus miembros mediante voto y, posteriormente, las decisiones de los comités se combinan nuevamente mediante voto para tomar la decisión final.

Este método se puede considerar como un algoritmo intermedio entre el algoritmo descrito en 3.2 y *boosting*. De hecho, si ejecutamos este método con un comité ($\mathbf{T_1}$=1) obtenemos el método *boosting*. Y, si lo ejecutamos con varios comités pero cada comité sólo con un miembro ($\mathbf{T_2}$=1), obtenemos el conjunto de árboles IGP.

5. Resultados

Para probar el funcionamiento del método propuesto, se han utilizado los siguientes conjuntos de datos del repositorio de la UCI [2]: *Breast Cancer Wisconsin*, *Pima Indian Diabetes*, *German Credit, Sonar* y *Waveform*. La Tabla 1 sintetiza las características de los conjuntos de datos seleccionados. La columna 2 de la tabla muestra el número de ejemplos empleados para construir los clasificadores (datos de entrenamiento), mientras que la 3ª columna indica el número de ejemplos utilizados para estimar el error de los clasificadores (datos de test). En la columna 4 se puede ver el número de clases y en la columna 5 el número de atributos para cada uno de los conjuntos de datos.

	Entr.	Test	Clases	Atribs.
Breast W.	500	199	2	9
Diabetes	500	268	2	8
German	600	400	2	24
Sonar	120	88	2	60
Waveform	300	5000	3	21

Tabla 1. Conjuntos de datos utilizados

Se ha ejecutado el algoritmo propuesto 50 veces para cada uno de los conjuntos arriba descritos. En cada ejecución se ha utilizado una

división aleatoria de los datos entre entrenamiento y test. En cada ejecución del algoritmo propuesto se ha construido $\mathbf{T}$=99 árboles divididos en 11 comités, esto es $\mathbf{T}_1$=11 y $\mathbf{T}_2$=9. En todos los casos se han utilizado elementos impares para reducir los casos de empate en las votaciones. Además se han ejecutado los casos extremos de este algoritmo, esto es, ($\mathbf{T}_1$=99 y $\mathbf{T}_2$=1) y ($\mathbf{T}_1$=1 y $\mathbf{T}_2$=99) que corresponden al conjunto de árboles IGP y *boosting* de árboles IGP respectivamente.

Asimismo se han realizado 50 ejecuciones con los algoritmos *bagging* y *boosting* (de tamaño $\mathbf{T}$=99), usando CART como algoritmo base, que nos servirán como referencia para evaluar la calidad de los resultados obtenidos. También se han ejecutado los algoritmos CART e IGP individualmente. En todos los experimentos con CART se ha utilizado validación cruzada de 10 particiones para podar los árboles.

En la Tabla 2 se presentan los errores de clasificación obtenidos para cada uno de los conjuntos de datos descritos en la Tabla 1 promediados sobre 50 ejecuciones (la desviación estándar se muestra entre paréntesis). La tabla se divide en tres secciones: la primera muestra los resultados para un clasificador individual sin utilizar conjuntos de clasificadores, la segunda sección muestra los datos obtenidos para conjuntos de clasificadores de 9 árboles y la tercera para 99 árboles. En la primera sección se muestran, en dos filas, los resultados de CART e IGPA ejecutados individualmente. En las secciones segunda y tercera se muestran, en cinco filas, los resultados obtenidos para: *bagging* usando CART (*Bagging* CART), conjunto de árboles IGP (Conjunto IGP), *boosting* basado en CART (*Boosting* CART), *boosting* de árboles IGP (*Boosting* IGP) y comités de *boostings* de árboles IGP (Conj. *Boost*. IGP). Hemos marcado en negrita, para cada conjunto de datos y sección, la celda correspondiente al clasificador que menor error medio ha obtenido. Para calcular los errores de Conj. *Boost*. IGP de $\mathbf{T}$=9 se han utilizado los tres primeros árboles de los tres primeros comités generados. Lo que es equivalente a $\mathbf{T}_1$=3 y $\mathbf{T}_2$=3.

De forma general en la Tabla 2 podemos ver que los conjuntos de clasificadores normalmente mejoran la clasificación con respecto al clasificador base. Como excepción a remarcar tenemos los algoritmos *boosting* CART y *boosting* IGP que empeoran el resultado obtenido por un solo clasificador para el conjunto de datos *Pima Indian Diabetes*. Este comportamiento coincide con el obtenido en estudios previos ([8], [16]) y pone de manifiesto los problemas de generalización de *boosting* para ciertos conjuntos de datos. Los algoritmos *bagging* CART y conjunto IGP no presentan este problema en ninguno de los conjuntos estudiados. Por otra parte, se puede observar como la convergencia del algoritmo propuesto con respecto al número de clasificadores parece ser más lenta que el resto de algoritmos probados. Para $\mathbf{T}$=9 nuestro algoritmo no obtiene el mejor resultado para ninguna de las bases de datos estudiadas mientras que para $\mathbf{T}$=99 obtiene el mejor resultado en 4 de los 5 conjuntos de datos.

En la Tabla 2 se puede observar como el algoritmo propuesto obtiene generalmente y para los conjuntos estudiados resultados mejores que *bagging* CART. Además, para $\mathbf{T}$=99, las diferencias entre estos dos conjuntos son estadísticamente significativas para un valor-p=0,01 (usando la prueba t de Student pareada con dos colas), exceptuando en el conjunto *Pima Indian Diabetes* donde obtienen resultados equivalentes. Con respecto a *boosting* CART y para $\mathbf{T}$=99 el algoritmo propuesto presenta mejores resultados en 4 de los 5 conjuntos estudiados aunque las diferencias son estadísticamente significativas (valor-p=0,01) sólo para *Sonar*, a favor de *boosting* CART y *Pima Indian Diabetes* a favor de nuestro algoritmo. En la Tabla 3 se muestra la significancia de los resultados del algoritmo propuesto con respecto a los otros 4 algoritmos probados para cada conjunto de datos y para la configuración de $\mathbf{T}$=99 usando la prueba t de Student. En negrita se han resaltado las diferencias estadísticamente significativas para un valor-p=0,01. Se puede observar que de las 20 comparaciones nuestro algoritmo es más eficaz en 11, en 8 no se obtienen resultados estadísticamente significativos y en 1 es menos eficiente (*Boosting* CART y el problema *Sonar*).

6. Conclusión

Este trabajo muestra un nuevo método de creación de conjuntos de clasificadores que combina *bagging* y *boosting*. De este modo se

		Breast	*Diabetes*	*German*	*Sonar*	*Waveform*
Individual	CART	0,059 (0,018)	**0,259 (0,025)**	**0,270 (0,020)**	**0,301 (0,040)**	**0,297 (0,022)**
	IGPA	**0,056 (0,016)**	0,263 (0,025)	0,283 (0,021)	0,305 (0,052)	0,306 (0,017)
T=9	*Bagging* CART	0,0528 (0,016)	0,250 (0,023)	0,265 (0,019)	0,278 (0,043)	0,243 (0,021)
	Conjunto IGP	0,0464 (0,017)	**0,244 (0,019)**	**0,252 (0,018)**	0,261 (0,047)	0,236 (0,016)
	Boosting CART	0,0447 (0,011)	0,269 (0,026)	0,269 (0,021)	**0,227 (0,047)**	**0,214 (0,012)**
	Boosting IGP	**0,0421 (0,014)**	0,274 (0,023)	0,276 (0,021)	0,248 (0,052)	0,215 (0.010)
	Conj. *Boost.* IGP[(1)]	0,0437 (0,013)	0,252 (0,020)	0,261 (0,017)	0,257 (0,052)	0,222 (0,0091)
T=99	*Bagging* CART	0,0467 (0,015)	0,249 (0,019)	0,259 (0,017)	0,261 (0,042)	0,222 (0,022)
	Conjunto IGP	0,0423 (0,013)	0,247 (0,023)	0,243 (0,017)	0,252 (0,043)	0,214 (0,019)
	Boosting CART	0,0364 (0,011)	0,261 (0,018)	0,241 (0,016)	**0,174 (0,039)**	0,176 (0,0064)
	Boosting IGP	0,0378 (0,013)	0,264 (0,022)	0,256 (0,021)	0,208 (0,050)	0,182 (0,010)
	Conj. *Boost.* IGP[(2)]	**0,0343 (0,011)**	**0,242 (0,020)**	**0,236 (0,014)**	0,206 (0,043)	**0,175 (0,0052)**

[(1)] 3 comités de 3 clasificadores cada uno (9 clasificadores en total)
[(2)] 11 comités de 9 clasificadores cada uno (99 clasificadores en total)

Tabla 2. Error medio sobre 50 ejecuciones para los clasificadores individuales y para conjuntos de 9 y 99 clasificadores. Entre paréntesis se muestra la desviación estándar.

	Bagging CART	Conjunto IGP	*Boosting* CART	*Boosting* IGP
Breast W.	**3,5e-5**	**0,002**	0,33	0,11
Diabetes	0,14	0,37	**3,5e-6**	**3,1e-6**
German	**2e-10**	0,022	0,085	**3,6e-8**
Sonar	**9,6e-8**	**2,5e-6**	**2,4e-4**	0,82
Waveform	**6e-19**	**1e-18**	0,66	**1,2e-5**

Tabla 3. Prueba-t de conjunto de *boosting*s IGP con respecto al resto de algoritmos probados para T=99

logra obtener resultados más eficaces para un rango mayor de problemas de clasificación. Concretamente se consigue obtener un clasificador competitivo con *boosting* a la vez que estable en conjuntos de datos donde *boosting* encuentra dificultades de generalización. Asimismo el algoritmo presentado obtiene resultados significativamente mejores que *bagging* para todos los conjuntos de datos estudiados.

Referencias

[1] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning 36(1-2) 1999

[2] Blake C. L., Merz C. J. UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. 1998

[3] Breiman L., Random forests, Machine Learning 45(1) 2001

[4] Breiman L., Randomizing outputs to increase prediction accuracy, Machine Learning 40(3) 2000

[5] Breiman L., Bias, variance, and arcing classifiers, Tech. Rep. 460, Statistics Department, University of California 1996

[6] Breiman L., Bagging predictors, Machine Learning 24(2) 1996

[7] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. New York: Chapman & Hall 1984

[8] Dietterich T. G., An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Machine Learning 40(2) 2000

[9] Dietterich T.G., Machine learning research: four current directions. Artificial Intelligence Magazine 18(4) 1997.

[10] Freund Y., Schapire R. E., A decision-theoretic generalization of on-line learning and an application to boosting, in: Proc. 2nd European Conference on Computational Learning Theory, 1995

[11] Gelfand, S.B., Ravishankar, C.S., Delp, E.J.: An iterative growing and pruning algorithm for classification tree design. IEEE Trans. Pattern Analysis and Machine Intelligence. 13(2) 1991

[12] Martínez-Muñoz G, Suárez A., Switching class labels to generate classification ensembles, Pattern Recognition (en prensa)

[13] Martínez-Muñoz, G., Suárez, A., Using all data to generate decision tree ensembles. IEEE Transactions on Systems, Man and Cybernetics C. 34(4) 2004

[14] Raetsch G., Onoda T., Mueller K.-R., Soft margins for AdaBoost, Machine Learning 42(3) 2001

[15] R. Schapire, Y. Freund, P. Bartlett, W. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, The Annals of Statistics 12(5) 1998

[16] Webb, G.I.: MultiBoosting: A technique for combining boosting and wagging. Machine Learning. 40(2) 2000