(Introduction to Data Mining [Tan-2013])

(Chapter 5)

Ensemble Methods [5.6]

Data Scientist
Jhoan Esteban Ruiz Borja Msc

Las técnicas de clasificación que hemos visto hasta ahora en este capítulo, con la excepción del método del vecino más cercano (KNN), predicen las etiquetas de clase de ejemplos desconocidos utilizando un único clasificador inducido a partir de datos de entrenamiento. Esta sección presenta técnicas para mejorar la precisión de la clasificación al agregar las predicciones de clasificadores múltiples. Estas técnicas se conocen como los métodos de combinación de conjunto (Ensemble Methods) o clasificador. Un método ensemble construye un conjunto de clasificadores básicos a partir de los datos de entrenamiento y realiza una clasificación mediante la votación de las predicciones realizadas por cada clasificador básico. Esta sección explica por qué los métodos de conjunto tienden a funcionar mejor que cualquier clasificador único y presenta técnicas para construir el ensemble clasificador.

Rationale for Ensemble Method [5.6.1]

El siguiente ejemplo ilustra cómo un método de ensemble puede mejorar el rendimiento de un clasificador.

Ejemplo 5.7 Considere un conjunto de veinticinco clasificadores binarios, cada uno de los cuales tiene una tasa de error de $\boldsymbol{\epsilon = 0.35}$. El clasificador de conjunto predice la etiqueta de clase de un ejemplo de prueba tomando un voto mayoritario sobre las predicciones hechas por los clasificadores de base. Si los clasificadores básicos son idénticos, el conjunto clasificará incorrectamente los mismos ejemplos pronosticados incorrectamente por los clasificadores básicos. Por lo tanto, la tasa de error del conjunto sigue siendo 0.35. Por otro lado, si los clasificadores básicos son independientes, es decir, sus errores no están correlacionados, entonces el conjunto hace una predicción errónea solo si más de la mitad de los clasificadores básicos predicen incorrectamente. En este caso, la tasa de error del clasificador conjunto es

$$e_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06 \qquad (5.66)$$

que es considerablemente más bajo que la tasa de error de los clasificadores básicos. **Fin del ejemplo.**

La Figura 5.30 muestra la tasa de error de un conjunto de veinticinco clasificadores binarios ($\boldsymbol{e_{ensemble}}$) para diferentes tasas de error de clasificador base ($\boldsymbol{\epsilon}$). La línea diagonal representa el caso en el que los clasificadores básicos son idénticos, mientras que la línea continua representa el caso en el que los clasificadores básicos son independientes. Observe que el

clasificador de conjunto funciona peor que los clasificadores de base cuando ϵ es mayor que 0.5.

El ejemplo anterior ilustra dos condiciones necesarias para que un clasificador conjunto tenga un mejor desempeño que un solo clasificador: (1) los clasificadores básicos deben ser independientes entre sí, y (2) los clasificadores básicos deben funcionar mejor que un clasificador que realiza adivinanzas al azar. En la práctica, es difícil asegurar la independencia total entre los clasificadores de base. Sin embargo, se han observado mejoras en las precisiones de clasificación en los métodos de conjunto en los que los clasificadores de base están ligeramente correlacionados.

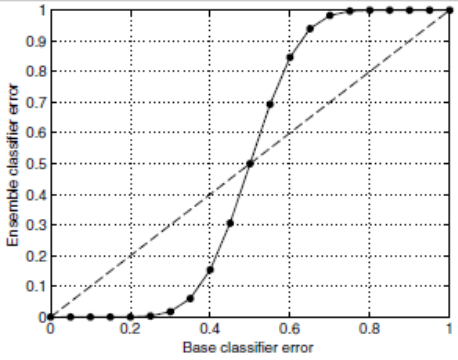


Figure 5.30. Comparison between errors of base classifiers and errors of the ensemble classifier.

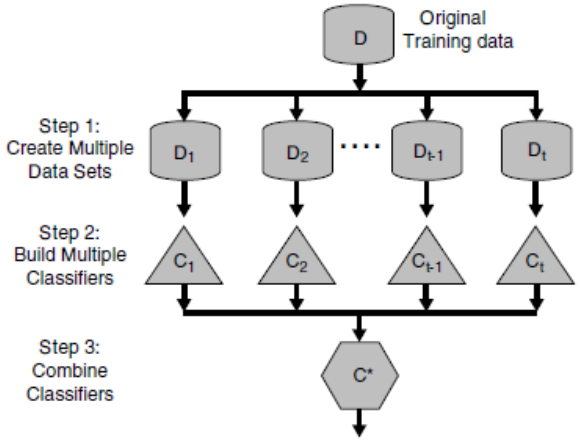


Figure 5.31. A logical view of the ensemble learning method.

Methods for Constructing an Ensemble Classifier [6.5.2]

En la Figura 5.31 se presenta una vista lógica del método de conjunto. La idea básica es construir múltiples clasificadores a partir de los datos originales y luego agregar sus predicciones cuando se clasifican ejemplos desconocidos. El conjunto de clasificadores se puede construir de muchas maneras:

1. **Manipulando el conjunto de entrenamiento**. En este enfoque, se crean conjuntos de entrenamiento múltiples al remuestrear los datos originales de acuerdo con alguna distribución de muestreo. La distribución muestral determina la probabilidad de que se

seleccione un ejemplo para la capacitación, y puede variar de una prueba a otra. Luego se construye un clasificador a partir de cada conjunto de entrenamiento utilizando un algoritmo de aprendizaje particular. Empaquetado y refuerzo son dos ejemplos de métodos conjuntos que manipulan sus conjuntos de entrenamiento. Estos métodos se describen con más detalle en las Secciones 5.6.4 y 5.6.5.

2. **Manipulando las características de entrada.** En este enfoque, se elige un subconjunto de características de entrada para formar cada conjunto de entrenamiento. El subconjunto puede elegirse al azar o en base a la recomendación de expertos en dominios. Algunos estudios han demostrado que este enfoque funciona muy bien con conjuntos de datos que contienen características altamente redundantes. El bosque aleatorio (**Random forest**), que se describe en la Sección 5.6.6, es un método conjunto que manipula sus características de entrada y utiliza árboles de decisión como sus clasificadores básicos.
3. **Manipulando las etiquetas de clase.** Este método se puede utilizar cuando el número de clases es suficientemente grande. Los datos de entrenamiento se transforman en un problema de clase binario al dividir aleatoriamente las etiquetas de clase en dos subconjuntos desunidos, $\boldsymbol{A_0}$ y $\boldsymbol{A_1}$. Los ejemplos de entrenamiento cuya etiqueta de clase pertenece al subconjunto $\boldsymbol{A_0}$ se asignan a la clase 0, mientras que los que pertenecen al subconjunto A1 se asignan a la clase 1. Los ejemplos que se vuelven a etiquetar se utilizan para entrenar un clasificador de base. $\boldsymbol{A_1}$ repetir los pasos de reetiquetado de clase y construcción de modelos varias veces, se obtiene un conjunto de clasificadores básicos. Cuando se presenta un ejemplo de prueba, cada clasificador base $\boldsymbol{C_i}$ se usa para predecir su etiqueta de clase. Si el ejemplo de prueba se predice como clase 0, todas las clases que pertenecen a $\boldsymbol{A_0}$ recibirán un voto. A la inversa, si se prevé que sea de clase 1, todas las clases que pertenecen a $\boldsymbol{A_1}$ recibirán un voto. Los votos se cuentan y la clase que recibe el voto más alto se asigna al ejemplo de prueba. Un ejemplo de este enfoque es el método de codificación de salida con corrección de errores descrito en la página 307.
4. **Manipulando el algoritmo de aprendizaje.** Muchos algoritmos de aprendizaje pueden manipularse de tal manera que la aplicación del algoritmo varias veces en los mismos datos de entrenamiento puede resultar en diferentes modelos. Por ejemplo, una red neuronal artificial puede producir diferentes modelos al cambiar su topología de red o los pesos iniciales de los enlaces entre las neuronas. De manera similar, se puede construir un conjunto de árboles de decisión inyectando aleatoriedad en el procedimiento de cultivo de árboles. Por ejemplo, en lugar de elegir el mejor atributo de división en cada nodo, podemos elegir al azar uno de los mejores $\boldsymbol{k}$ atributos para la división.

Los primeros tres enfoques son métodos genéricos que son aplicables a cualquier clasificador, mientras que el cuarto enfoque depende del tipo de clasificador utilizado. Los clasificadores básicos para la mayoría de estos enfoques se pueden generar secuencialmente (uno tras otro) o en paralelo (todos a la vez). El algoritmo 5.5 muestra los pasos necesarios para construir un clasificador de conjunto de una manera secuencial. El primer paso es crear un conjunto de entrenamiento a partir de los datos originales $\boldsymbol{D}$. Dependiendo del tipo de método de conjunto

usado, los conjuntos de entrenamiento son idénticos o leves modificaciones de $\boldsymbol{D}$. El tamaño del conjunto de entrenamiento a menudo se mantiene igual que el Datos originales, pero la distribución de ejemplos puede no ser idéntica; es decir, algunos ejemplos pueden aparecer varias veces en el conjunto de entrenamiento, mientras que otros pueden no aparecer incluso una vez. Luego se construye un clasificador de base $\boldsymbol{C_i}$ a partir de cada conjunto de entrenamiento $\boldsymbol{D_i}$. Los métodos de conjunto funcionan mejor con clasificadores inestables, es decir, clasificadores básicos que son sensibles a perturbaciones menores en el conjunto de entrenamiento. Los ejemplos de clasificadores inestables incluyen árboles de decisión, clasificadores basados en reglas y redes neuronales artificiales. Como se discutirá en la Sección 5.6.3, la variabilidad entre los ejemplos de entrenamiento es una de las fuentes principales de errores en un clasificador. Al agregar los clasificadores básicos creados a partir de diferentes conjuntos de entrenamiento, esto puede ayudar a reducir este tipo de errores.

Finalmente, un ejemplo de prueba $\boldsymbol{x}$ se clasifica combinando las predicciones hechas por los clasificadores base $\boldsymbol{C_i(x)}$:

$$\boldsymbol{C^*(x) = Vote\big(C_1(x), C_2(x), .., C_k(x)\big)}$$

La clase se puede obtener tomando un voto mayoritario sobre las predicciones individuales o ponderando cada predicción con la precisión del clasificador de base.

Algorithm 5.5 General procedure for ensemble method.

1: Let D denote the original training data, k denote the number of base classifiers, and T be the test data.
2: **for** $i = 1$ to k **do**
3: Create training set, D_i from D.
4: Build a base classifier C_i from D_i.
5: **end for**
6: **for** each test record $x \in T$ **do**
7: $C^*(x) = Vote(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x}))$
8: **end for**

La descomposición Sesgo-Varianza (Bias-Variance Decomposition) [6.5.3]

La descomposición de la desviación de sesgo es un método formal para analizar el error de predicción de un modelo predictivo. El siguiente ejemplo da una explicación intuitiva para este método.

La figura 5.32 muestra las trayectorias de un proyectil lanzado en un ángulo particular. Supongamos que el proyectil golpea la superficie del piso en alguna ubicación $\boldsymbol{x}$, a una distancia $\boldsymbol{d}$ alejada de la posición objetivo $\boldsymbol{t}$. Dependiendo de la fuerza aplicada al proyectil, la distancia observada puede variar de una prueba a otra. La distancia observada puede descomponerse en varios componentes. El primer componente, conocido como sesgo, mide la distancia promedio entre la posición de destino y la ubicación donde el proyectil golpea el piso. La cantidad de sesgo depende del ángulo del lanzador de proyectiles. El segundo componente, que se conoce como varianza, mide la desviación entre $\boldsymbol{x}$ y la posición promedio

$\bar{x}$ donde el proyectil golpea el piso. La variación puede explicarse como resultado de los cambios en la cantidad de fuerza aplicada al proyectil. Finalmente, si el objetivo no es estacionario, la distancia observada también se ve afectada por los cambios en la ubicación del objetivo. Esto se considera el componente de ruido asociado con la variabilidad en la posición de destino. Al juntar estos componentes, la distancia promedio se puede expresar como:

$$d_{f,\theta}(y,t) = Bias_{\theta} + Variance_{f} + Noise_{t} \qquad (5.67)$$

donde f se refiere a la cantidad de fuerza aplicada y θ es el ángulo del lanzador.

La tarea de predecir la etiqueta de clase de un ejemplo dado puede analizarse utilizando el mismo enfoque. Para un clasificador dado, algunas predicciones pueden resultar correctas, mientras que otras pueden estar completamente fuera de lugar. Podemos descomponer el error esperado de un clasificador como una suma de los tres términos dados en la Ecuación 5.67, donde el error esperado es la probabilidad de que el clasificador clasifique incorrectamente un ejemplo dado. El resto de esta sección examina el significado de sesgo, varianza y ruido en el contexto de la clasificación.

Un clasificador suele ser entrenado para minimizar su error de entrenamiento. Sin embargo, para que sea útil, el clasificador debe poder hacer una suposición informada sobre las etiquetas de clase de los ejemplos que nunca ha visto antes. Esto requiere que el clasificador generalice su límite de decisión a regiones donde no hay ejemplos de capacitación disponibles, una decisión que depende de la elección de diseño del clasificador. Por ejemplo, un problema de diseño clave en la inducción del árbol de decisión es la cantidad de poda necesaria para obtener un árbol con un bajo error esperado. La Figura 5.33 muestra dos árboles de decisión, T_1 y T_2, que se generan a partir de los mismos datos de entrenamiento, pero tienen diferentes complejidades. T_2 se obtiene recortando T_1 hasta que se obtiene un árbol con una profundidad máxima de dos. T_1, por otro lado, realiza muy poca poda en su árbol de decisión. Estas opciones de diseño introducirán un sesgo en el clasificador que es análogo al sesgo del lanzador de proyectiles descrito en el ejemplo anterior. En general, cuanto más fuertes sean los supuestos hechos por un clasificador sobre la naturaleza de su límite de decisión, mayor será el sesgo del clasificador. Por lo tanto, T_2 tiene un sesgo mayor porque hace suposiciones más firmes sobre su límite de decisión (que se refleja en el tamaño del árbol) en comparación con T_1. Otras opciones de diseño que pueden introducir un sesgo en un clasificador incluyen la topología de red de una red neuronal artificial y el número de vecinos considerados por un clasificador de vecinos más cercanos.

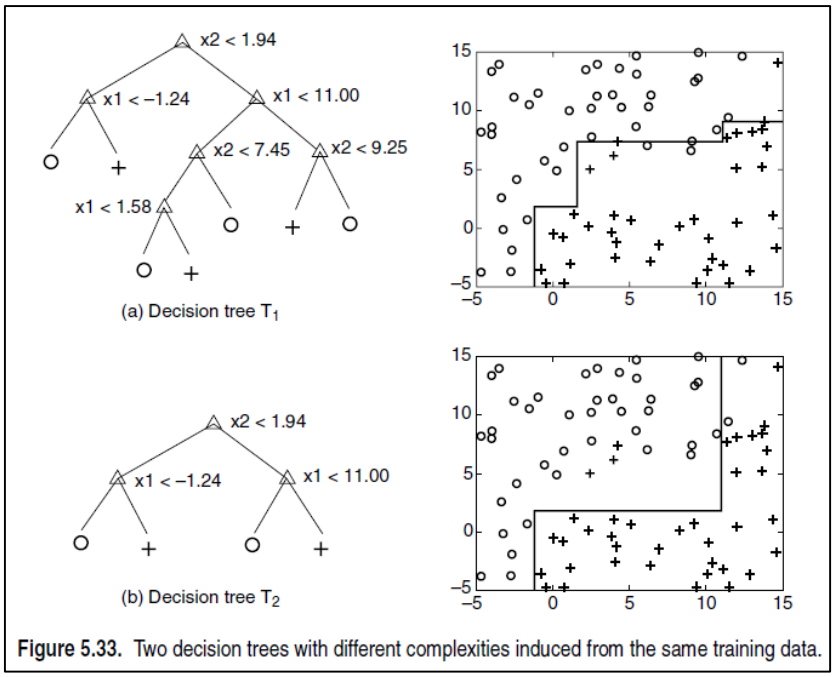
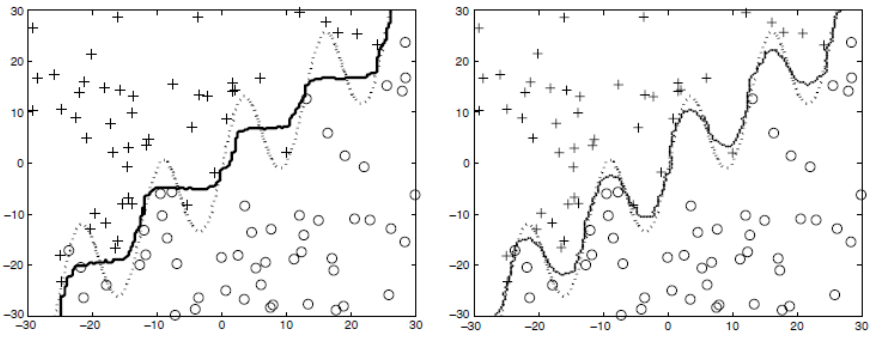


Figure 5.33. Two decision trees with different complexities induced from the same training data.

El error esperado de un clasificador también se ve afectado por la variabilidad en los datos de entrenamiento porque diferentes composiciones del conjunto de entrenamiento pueden llevar a diferentes límites de decisión. Esto es análogo a la varianza en $\boldsymbol{x}$ cuando se aplican diferentes cantidades de fuerza al proyectil. El último componente del error esperado está asociado con el ruido intrínseco en la clase objetivo. La clase objetivo para algunos dominios puede ser no determinista; es decir, las instancias con los mismos valores de atributo pueden tener diferentes etiquetas de clase. Tales errores son inevitables incluso cuando se conoce el verdadero límite de decisión.

La cantidad de sesgo y varianza que contribuye al error esperado depende del tipo de clasificador utilizado. La Figura 5.34 compara los límites de decisión producidos por un árbol de decisión y un clasificador vecino más cercano. Para cada clasificador, trazamos el límite de decisión obtenido al "promediar" los modelos inducidos a partir de 100 conjuntos de entrenamiento, cada uno con 100 ejemplos. El verdadero límite de decisión a partir del cual se generan los datos también se traza utilizando una línea discontinua. La diferencia entre el límite de decisión real y el límite de decisión "promediado" refleja el sesgo del clasificador. Después de promediar los modelos, observe que la diferencia entre el límite de decisión real y el límite de decisión producido por el clasificador vecino más cercano es menor que la diferencia observada para un clasificador de árbol de decisión. Este resultado sugiere que el sesgo de un clasificador vecino 1 más cercano es más bajo que el sesgo de un clasificador de árbol de decisión.

Por otro lado, el clasificador del vecino 1 más cercano es más sensible a la composición de sus ejemplos de entrenamiento. Si examinamos los modelos inducidos a partir de diferentes conjuntos de entrenamiento, hay más variabilidad en el límite de decisión de un clasificador vecino más cercano que un clasificador de árbol de decisión. Por lo tanto, el límite de decisión de un clasificador de árbol de decisión tiene una variación más baja que el clasificador vecino más cercano.



(a) Decision boundary for decision tree.

(b) Decision boundary for 1-nearest neighbor.

Figure 5.34. Bias of decision tree and 1-nearest neighbor classifiers.

Bagging [6.5.4]

El **bagging**, que también se conoce como agregación de arranque (**bootstrap aggregation**), es una técnica que muestrea repetidamente (con reemplazo) de un conjunto de datos de acuerdo con una distribución de probabilidad uniforme. Cada muestra de bootstrap tiene el mismo tamaño que los datos originales. Debido a que el muestreo se realiza con reemplazo, algunos casos pueden aparecer varias veces en el mismo conjunto de entrenamiento, mientras que otros pueden omitirse del conjunto de entrenamiento. En promedio, una muestra de bootstrap $\boldsymbol{D_i}$ contiene aproximadamente el **63%** de los datos de entrenamiento originales porque cada muestra tiene una probabilidad de $\mathbf{1-(1-1/N)^N}$ de ser seleccionada en cada $\boldsymbol{D_i}$. Si $\boldsymbol{N}$ es suficientemente grande, esta probabilidad converge a $\mathbf{1-1/e \cong 0.632}$. El procedimiento básico para el **bagging** se resume en el algoritmo 5.6. Después de entrenar a los $\boldsymbol{k}$ clasificadores, se asigna una instancia de prueba a la clase que recibe el mayor número de votos.

Algorithm 5.6 Bagging algorithm.

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size N, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \delta\big(C_i(x) = y\big)$.
$\{\delta(\cdot) = 1$ if its argument is true and 0 otherwise$\}$.

Para ilustrar cómo funciona el empaquetamiento, considere el conjunto de datos que se muestra en la Tabla 5.4.

Digamos que $\boldsymbol{x}$ denota un atributo unidimensional e $\boldsymbol{y}$ denota la etiqueta de clase. Supongamos que aplicamos un clasificador que induce solo árboles de decisión binarios de un nivel, con una condición de prueba $\boldsymbol{x \leq k}$, donde $\boldsymbol{k}$ es un punto de división elegido para

minimizar la entropía de los nodos de la hoja. Tal árbol también se conoce como un tocón de decisión.

Sin el **bagging**, el mejor tocón de decisión que podemos producir divide los registros en $\boldsymbol{x \leq 0.35}$ o $\boldsymbol{x \leq 0.75}$. De cualquier manera, la precisión del árbol es como máximo del 70%. Supongamos que aplicamos el procedimiento de **bagging** en el conjunto de datos utilizando diez muestras de bootstrap. Los ejemplos elegidos para el entrenamiento en cada ronda de **bagging** se muestran en la Figura 5.35. En el lado derecho de cada tabla, también ilustramos el límite de decisión producido por el clasificador.

Table 5.4. Example of data set used to construct an ensemble of bagging classifiers.

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	−1	−1	−1	−1	1	1	1

Clasificamos todo el conjunto de datos dado en la Tabla 5.4 tomando un voto mayoritario entre las predicciones hechas por cada clasificador de base. Los resultados de las predicciones se muestran en la Figura 5.36. Dado que las etiquetas de clase son −1 o +1, tomar el voto mayoritario es equivalente a resumir los valores pronosticados de $\boldsymbol{y}$ y examinar el signo de la suma resultante (consulte la segunda a la última fila en la Figura 5.36). Observe que el clasificador de conjunto clasifica perfectamente los diez ejemplos en los datos originales.

El ejemplo anterior ilustra otra ventaja de usar métodos ensemble para mejorar la representación de la función objetivo. Aunque cada clasificador de base es un muñón de decisión, la combinación de los clasificadores puede llevar a un árbol de decisión de profundidad 2.

El **bagging** mejora el error de generalización al reducir la varianza de los clasificadores básicos. El rendimiento del **bagging** depende de la estabilidad del clasificador de base. Si un clasificador de base es inestable, el **bagging** ayuda a reducir los errores asociados con las fluctuaciones aleatorias en los datos de entrenamiento. Si un clasificador de base es estable, es decir, robusto a pequeñas perturbaciones en el conjunto de entrenamiento, entonces el error del conjunto se debe principalmente a un sesgo en el clasificador de base. En esta situación, el **bagging** puede no ser capaz de mejorar significativamente el rendimiento de los clasificadores básicos. Incluso puede degradar el rendimiento del clasificador porque el tamaño efectivo de cada conjunto de entrenamiento es aproximadamente un 37% más pequeño que los datos originales.

Finalmente, dado que cada muestra tiene la misma probabilidad de ser seleccionada, el bagging no se enfoca en ninguna instancia particular de los datos de entrenamiento. Por lo tanto, es menos susceptible al sobreajuste de modelos cuando se aplica a datos ruidosos.

Boosting [6.5.5]

El **boosting** es un procedimiento iterativo que se utiliza para cambiar de forma adaptativa la distribución de los ejemplos de capacitación, de modo que los clasificadores básicos se

centren en los ejemplos que son difíciles de clasificar. A diferencia del **bagging**, el **boosting** asigna un peso a cada ejemplo de entrenamiento y puede cambiar el peso de manera adaptativa al final de cada ronda de **boosting**. Los pesos asignados a los ejemplos de entrenamiento se pueden utilizar de las siguientes maneras:

1. Se pueden usar como una distribución de muestreo para dibujar un conjunto de muestras de bootstrap de los datos originales.
2. El clasificador de base puede utilizarlos para aprender un modelo que está orientado hacia ejemplos de mayor peso.

Bagging Round 1:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9	x <= 0.35 ==> y = 1
y	1	1	1	1	-1	-1	-1	-1	1	1	x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1	x <= 0.65 ==> y = 1
y	1	1	1	-1	-1	1	1	1	1	1	x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9	x <= 0.35 ==> y = 1
y	1	1	1	-1	-1	-1	-1	-1	1	1	x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9	x <= 0.3 ==> y = 1
y	1	1	1	-1	-1	-1	-1	-1	1	1	x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1	x <= 0.35 ==> y = 1
y	1	1	1	-1	-1	-1	-1	1	1	1	x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1	x <= 0.75 ==> y = -1
y	1	-1	-1	-1	-1	-1	-1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1	x <= 0.75 ==> y = -1
y	1	-1	-1	-1	-1	1	1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1	x <= 0.75 ==> y = -1
y	1	1	-1	-1	-1	-1	-1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1	x <= 0.75 ==> y = -1
y	1	1	-1	-1	-1	-1	-1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9	x <= 0.05 ==> y = -1
y	1	1	1	1	1	1	1	1	1	1	x > 0.05 ==> y = 1

Figure 5.35. Example of bagging.

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

Esta sección describe un algoritmo que utiliza ponderaciones de ejemplos para determinar la distribución muestral de su conjunto de entrenamiento. Inicialmente, a los ejemplos se les

asigna pesos iguales, $\mathbf{1}/\boldsymbol{N}$, de modo que es probable que se elijan para el entrenamiento. Se toma una muestra de acuerdo con la distribución muestral de los ejemplos de capacitación para obtener un nuevo conjunto de capacitación. A continuación, se induce un clasificador del conjunto de entrenamiento y se usa para clasificar todos los ejemplos en los datos originales. Los pesos de los ejemplos de entrenamiento se actualizan al final de cada ronda de **boosting**. A los ejemplos que se clasifican incorrectamente se les aumentará el peso, mientras que a los clasificados correctamente se les disminuirá el peso. Esto obliga al clasificador a centrarse en ejemplos que son difíciles de clasificar en iteraciones posteriores.

La siguiente tabla muestra los ejemplos elegidos durante cada ronda de **boosting**.

Boosting (Round 1):	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2):	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3):	4	4	8	10	4	5	4	6	3	4

Inicialmente, todos los ejemplos tienen asignados los mismos pesos. Sin embargo, algunos ejemplos pueden elegirse más de una vez, por ejemplo, los ejemplos 3 y 7, porque el muestreo se realiza con reemplazo. Un clasificador construido a partir de los datos se utiliza para clasificar todos los ejemplos. Supongamos que el ejemplo 4 es difícil de clasificar. El peso de este ejemplo se incrementará en futuras iteraciones a medida que se clasifique erróneamente en varias ocasiones. Mientras tanto, los ejemplos que no se eligieron en la ronda anterior, por ejemplo, los ejemplos 1 y 5, también tienen una mejor oportunidad de ser seleccionados en la próxima ronda, ya que sus predicciones en la ronda anterior probablemente sean erróneas. A medida que avanzan las rondas de refuerzo, los ejemplos más difíciles de clasificar tienden a prevalecer aún más. El conjunto final se obtiene agregando los clasificadores básicos obtenidos de cada ronda de **boosting**.

A lo largo de los años, se han desarrollado varias implementaciones del algoritmo de **boosting**. Estos algoritmos difieren en términos de (1) cómo se actualizan los pesos de los ejemplos de entrenamiento al final de cada ronda de **boosting**, y (2) cómo se combinan las predicciones hechas por cada clasificador. Una implementación llamada **AdaBoost** se explora en la siguiente sección.

AdaBoost

Sea $\{(\boldsymbol{x_j}, \boldsymbol{y_j}) \mid \boldsymbol{j} = \mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{N}\}$ denota un conjunto de $\boldsymbol{N}$ ejemplos de entrenamiento. En el algoritmo de AdaBoost, la importancia de un clasificador base $\boldsymbol{C_i}$ depende de su tasa de error, que se define como

$$\epsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} w_j \, I\Big(C_i(\mathbf{x}_j) \neq y_j\Big)\right], \qquad (5.68)$$

donde $\boldsymbol{I}(\boldsymbol{p}) = \mathbf{1}$ si el predicado $\boldsymbol{p}$ es verdadero, y 0 en caso contrario. La importancia de un clasificador $\boldsymbol{C_i}$ viene dada por el siguiente parámetro,

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right).$$

Tenga en cuenta que $\boldsymbol{\alpha_i}$ tiene un valor positivo grande si la tasa de error es cercana a 0 y un valor negativo grande si la tasa de error es cercana a 1, como se muestra en la Figura 5.37.

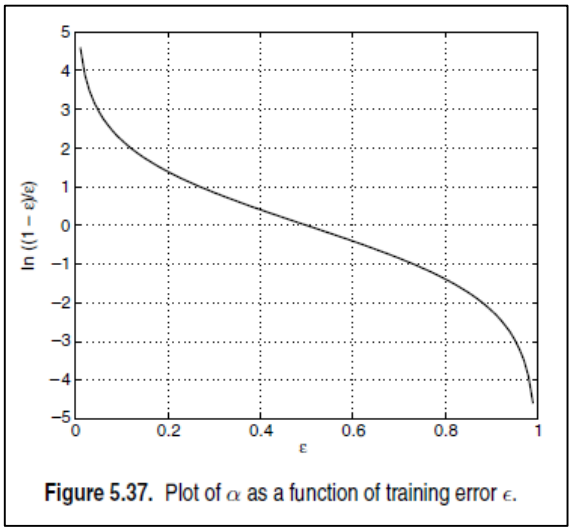


Figure 5.37. Plot of α as a function of training error ϵ.

El parámetro $\boldsymbol{\alpha_i}$ también se usa para actualizar el peso de los ejemplos de entrenamiento. Para ilustrar, digamos que $\boldsymbol{w_i^{(j)}}$ denota el peso asignado al ejemplo $(\boldsymbol{x_i, y_i})$ durante la ronda de $\boldsymbol{j}$-esimo refuerzo. El mecanismo de actualización de peso para AdaBoost viene dado por la ecuación:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(\mathbf{x_i}) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(\mathbf{x_i}) \neq y_i \end{cases}, \qquad (5.69)$$

donde $\boldsymbol{Z_j}$ es el factor de normalización utilizado para asegurar que $\sum_i w_i^{(j+1)} = 1$. La fórmula de actualización de peso dada en la Ecuación 5.69 aumenta los pesos de los ejemplos clasificados incorrectamente y disminuye los pesos de los clasificados correctamente.

En lugar de utilizar un esquema de votación por mayoría, la predicción hecha por cada clasificador $\boldsymbol{C_j}$ se ponderará de acuerdo con $\boldsymbol{\alpha_j}$. Este enfoque le permite a AdaBoost penalizar los modelos que tienen poca precisión, por ejemplo, los generados en las rondas de refuerzo anteriores. Además, si las rondas intermedias producen una tasa de error superior al 50%, las ponderaciones vuelven a sus valores uniformes originales, $\boldsymbol{w_i = \frac{1}{N}}$, y se repite el procedimiento de remuestreo. El algoritmo de AdaBoost se resume en el algoritmo 5.7.

Algorithm 5.7 AdaBoost algorithm.

1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Initialize the weights for all N examples.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all examples in the original training set, D.
7: $\epsilon_i = \frac{1}{N}\left[\sum_j w_j \, \delta(C_i(x_j) \neq y_j)\right]$ {Calculate the weighted error.}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Reset the weights for all N examples.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each example according to Equation 5.69.
14: **end for**
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.

Examinemos cómo funciona el enfoque de refuerzo en el conjunto de datos que se muestra en la Tabla 5.4. Inicialmente, todos los ejemplos tienen pesos idénticos. Después de tres rondas de refuerzo, los ejemplos elegidos para el entrenamiento se muestran en la Figura 5.38 (a). Las ponderaciones para cada ejemplo se actualizan al final de cada ronda de aumento con la Ecuación 5.69.

Sin aumentar, la precisión del grupo de decisión es, en el mejor de los casos, el 70%. Con AdaBoost, los resultados de las predicciones se dan en la Figura 5.39 (b). La predicción final del clasificador conjunto se obtiene al tomar un promedio ponderado de las predicciones hechas por cada clasificador base, que se muestra en la última fila de la Figura 5.39 (b). Tenga en cuenta que AdaBoost clasifica perfectamente todos los ejemplos en los datos de entrenamiento.

Un importante resultado analítico de la mejora muestra que el error de entrenamiento del conjunto está limitado por la siguiente expresión:

$$e_{\text{ensemble}} \leq \prod_i \left[\sqrt{\epsilon_i(1-\epsilon_i)}\right], \tag{5.70}$$

donde $\boldsymbol{\epsilon_i}$ es la tasa de error de cada clasificador base $\boldsymbol{i}$. Si la tasa de error del clasificador de base es inferior al 50%, podemos escribir $\boldsymbol{\epsilon_i = 0.5 - \gamma_i}$, donde $\boldsymbol{\gamma_i}$ mide cuánto mejor es el clasificador que el cálculo aleatorio. El límite en el error de entrenamiento del conjunto se convierte en

$$e_{\text{ensemble}} \leq \prod_i \sqrt{1 - 4\gamma_i^2} \leq \exp\left(-2\sum_i \gamma_i^2\right). \tag{5.71}$$

Si $\boldsymbol{\gamma_i < \gamma *}$ para todos los $\boldsymbol{i}$, entonces el error de entrenamiento del conjunto disminuye exponencialmente, lo que conduce a la rápida convergencia del algoritmo. Sin embargo, debido a su tendencia a centrarse en ejemplos de entrenamiento que se clasifican erróneamente, la técnica de refuerzo puede ser bastante susceptible de sobrealimentación.