Combinación de Clasificadores

Parte 1: Métodos de comité, Bagging, Boosting

Reconocimiento de Patrones

Departamento de Procesamiento de Señales
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería, UdelaR

2018

Bishop, cap. 14 – Hastie, Tibshirani & Friedman, sec. 16.3, sec. 8.8, cap. 10

Objetivo

La combinación de clasificadores consta de dos tareas:

1. Diseñar un conjunto de clasificadores de base a partir de los datos de entrenamiento
2. Diseñar una estrategia de combinación para combinarlos en un único clasificador.

Ejemplo 1:

1. Entrenar varios clasificadores
2. Nueva muestra: cada clasificador vota por su salida
3. Se clasifica según la mayoría.

Ejemplo 2:

- Supongamos que todo clasificador puede cuantificar la confianza en su decisión (e.g: si en 10-NN, 9 vecinos de $\mathbf{x}$ son de la clase $\mathcal{C}_k$, confianza alta). clasificadores
- Confianza del clasificador y_m en clasificar $\mathbf{x}$: $C(y_m|\mathbf{x}) = \max_{j=1,\ldots,K} \hat{P}(\mathcal{C}_k|\mathbf{x}, y_m)$.
- Se clasifica según mayoría ponderada por $C(y_m|\mathbf{x})$.

Motivación

Motivación:

- *No free lunch theorem.* En machine learning: ningún clasificador es universalmente mejor que todos los demás.
- En media, una estrategia de combinación adecuada generará mejores predicciones que cualquiera de los clasificadores individualmente.

Otro paradigma:

- Hasta ahora: búsqueda o diseño de mejores características, diseño de mejores clasificadores
- Ahora: foco en la estrategia de clasificación. Aún partiendo de clasificadores débiles (acierto de $0.5 + \varepsilon$), una buena estrategia de combinación lleva muy buenos desempeños.

Posibles escenarios

Varios escenarios posibles:

- Distintos tipos de clasificadores operando sobre distintas características (identificación de personas combinando un clasificador de huellas dactilares, de caras, de firmas, de iris, etc).
- Múltiples clasificadores operando sobre todo el conjunto de características
- Múltiples clasificadores operando sobre todo el conjunto de datos
- Múltiples instancias de un mismo clasificador, cada uno operando sobre un subconjunto distinto de datos
- Múltiples instancias de un mismo tipo de clasificador, con distintos parámetros (e.g. clasificadores lineales; mismo algoritmo con distintas inicializaciones).

Características deseables del conjunto de clasificadores base

- **Diversidad:**
  - Clasificadores especializados en distintas características;
  - Clasificadores entrenados sobre distintos datos (e.g. sobre distintos subconjuntos elegidos al azar)
  - Distintos tipos de clasificador: lineal, k-NN, redes, SVM, ...
  - Distintos parámetros: cantidad de capas y de neuronas en redes, distintos k en k-NN, ...
- **Independencia:** máxima reducción de error, máxima reducción de varianza.
- **Complementariedad:** variedad de especialización.

Métodos y estrategias de combinación

- **Entrenamiento en paralelo, con distintos conjuntos de entrenamiento:**
  - Bagging (boostrap aggregation): entrenar distintos modelos sobre distintos muestreos del training set, luego promediar o votar predicciones (hoy).
  - Random forest: combina bagging con selección aleatoria de atributos (la próxima).
- **Entrenamiento secuencial o en cascada:**
  - Boosting: al pasar por la cascada de clasificadores, a las muestras difíciles de clasificar se les va aumentando el peso en el costo total, para que los clasificadores cuesta arriba se focalicen en su clasificación (hoy).
- **Entrenamiento paralelo sobre un mismo conjunto de entrenamiento:**
  - Mezcla de expertos (la próxima).

Por qué funciona (1)

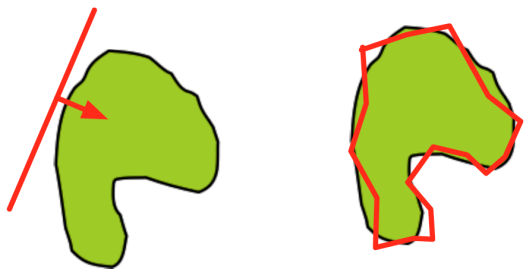
Recordemos: Compromiso sesgo-varianza (análisis del error de generalización):

$$\textbf{Error cuadrático medio} = (\textbf{Sesgo})^2 + \textbf{Varianza} + \textbf{Ruido}.$$

- **Ruido:** independiente del clasificador; error irreducible, intrínseco al problema.
- **Varianza:** sensibilidad del clasificador ante fluctuaciones en el conjunto de entrenamiento.
- **Sesgo:** error medio de predicción (sobre todos los conjuntos de datos) del clasificador.

Dos observaciones fundamentales

- **Reducción de varianza:** si los conjuntos de entrenamiento son independientes, la combinación reduce la varianza sin afectar el sesgo (e.g. bagging).
- **Reducción del sesgo:** la combinación o el promedio de modelos simples tiene mayor capacidad que cada modelo por separado (e.g. boosting, mezcla de expertos).



Sesgo: un clasificador lineal vs. mezcla de clasificadores lineales

Por qué funciona (2)

M clasificadores independientes, todos con probabilidad de error p:

$$P(m \text{ errores}) = \binom{M}{m} p^m (1-p)^{M-m}.$$

Votación por mayoría:

$$P(\text{clasificación incorrecta}) = \sum_{m=\lceil M/2 \rceil}^{M} \binom{M}{m} p^m (1-p)^{M-m}.$$

	$p = 0.3$
M	$P(\text{clasificación incorrecta})$
11	0.078225
21	0.026390
121	0.000002

	$p = 0.49$
M	$P(\text{clasificación incorrecta})$
11	0.472948
121	0.412750
10001	0.022731

Por qué funciona (3)

Efecto de la especialización:

Supongamos:

- Muestras etiquetadas $(\mathbf{x}_n, t_n)$, $n = 1, \ldots, N$.
- $y_m(\mathbf{x})$ clasificadores entrenados, $m = 1, \ldots, M$.
- $\{R_1, R_2, \ldots, R_L\}$ partición del espacio de características $\mathcal{R}$.
- $y^*(\mathbf{x})$ clasificador con menor tasa de error.
- En R_j elijo $y_{i(j)}(\mathbf{x})$ tal que $\forall\, m = 1, \ldots, M$,

$$P(y_{i(j)}(\mathbf{x}_n) = t_n | xv_n \in R_j) \geq P(y^*(\mathbf{x}_n) = t_n | xv_n \in R_j).$$

Entonces,

$$P(\text{clasificación correcta}) = \sum_{j=1}^{L} P(\mathbf{x}_n \in R_j) P(y_{i(j)}(\mathbf{x}_n) = t_n | \mathbf{x}_n \in R_j) \geq P(y^*(\mathbf{x}_n) = t_n, \mathbf{x}_n \in \mathcal{R}).$$

Estimación Bootstrap (Efron, 1979)

- Muestrear con reposición N muestras del conjunto de entrenamiento $\mathcal{D}$
- Para cada conjunto de muestras, estimar el estadístico
- El estimado bootstrap es el promedio de los estimados individuales
- En general interesa estimar el estadístico y su varianza.

Predictores Bagging (Breiman, 1994)

Bagging: **B**ootstrap **Agg**rega**ting**

- M subconjuntos de entrenamiento: muestras con reposición (bootstrap) del conjunto de entrenamiento completo.
- M clasificadores, cada uno entrenado con uno de los subconjuntos anteriores.
- Las predicciones se combinan por mayoría.

- *Idealmente:* subconjuntos independientes, muestreados de los datos.
- *En la práctica:* un solo conjunto de entrenamiento

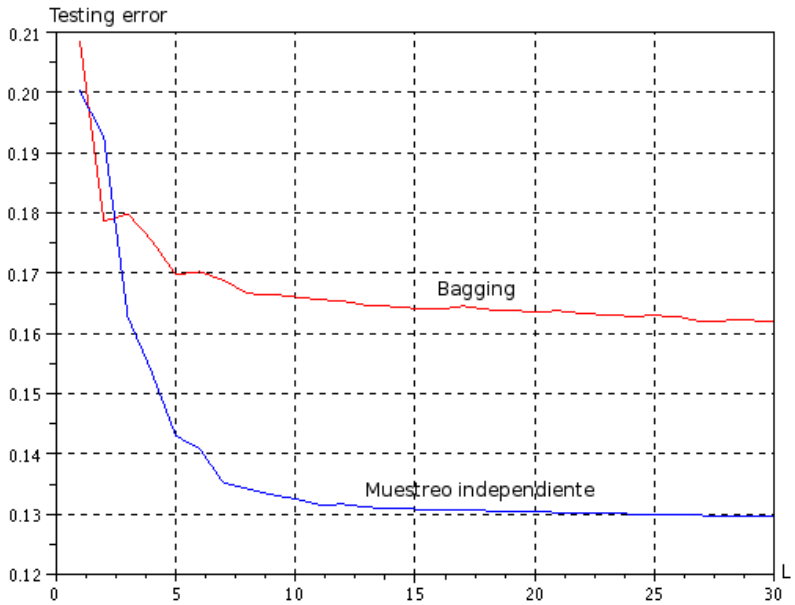
$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \rightarrow \text{bootstrap} \rightarrow \mathcal{S}_m = \{\mathbf{x}_1^{(m)}, \ldots, \mathbf{x}_{N_m}^{(m)}\}.$$

Clasificador base inestable (sesgo pequeño, varianza grande): redes neuronales, árboles de decisión, ...

Bagging aproxima la media de la distribución a posteriori. Cuanto más muestreos mejor.

¿Por qué funciona?

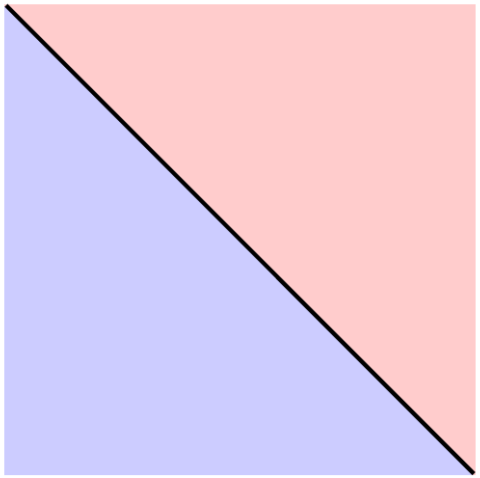
- Para M clasificadores independientes con misma probabilidad de error p, el voto por mayoría garantiza mejora.
- Bagging intenta generar clasificadores independientes usando bootstrap en el conjunto de entrenamiento. del conjunto de entrenamiento completo.
- En realidad son pseudo-independientes pues se toman muestras del mismo conjunto $\mathcal{D}$.
- Esto implica un deterioro en el desempeño, pero es mejor que nada.



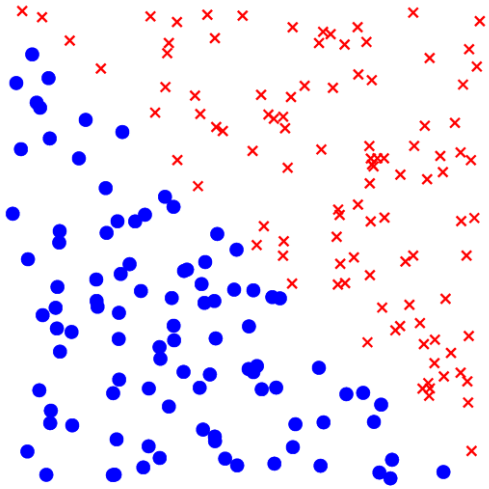
Evolución del error con M.

Predictores Bagging

Ejemplo: dos clases, los y_m árboles de decisión



Frontera verdadera

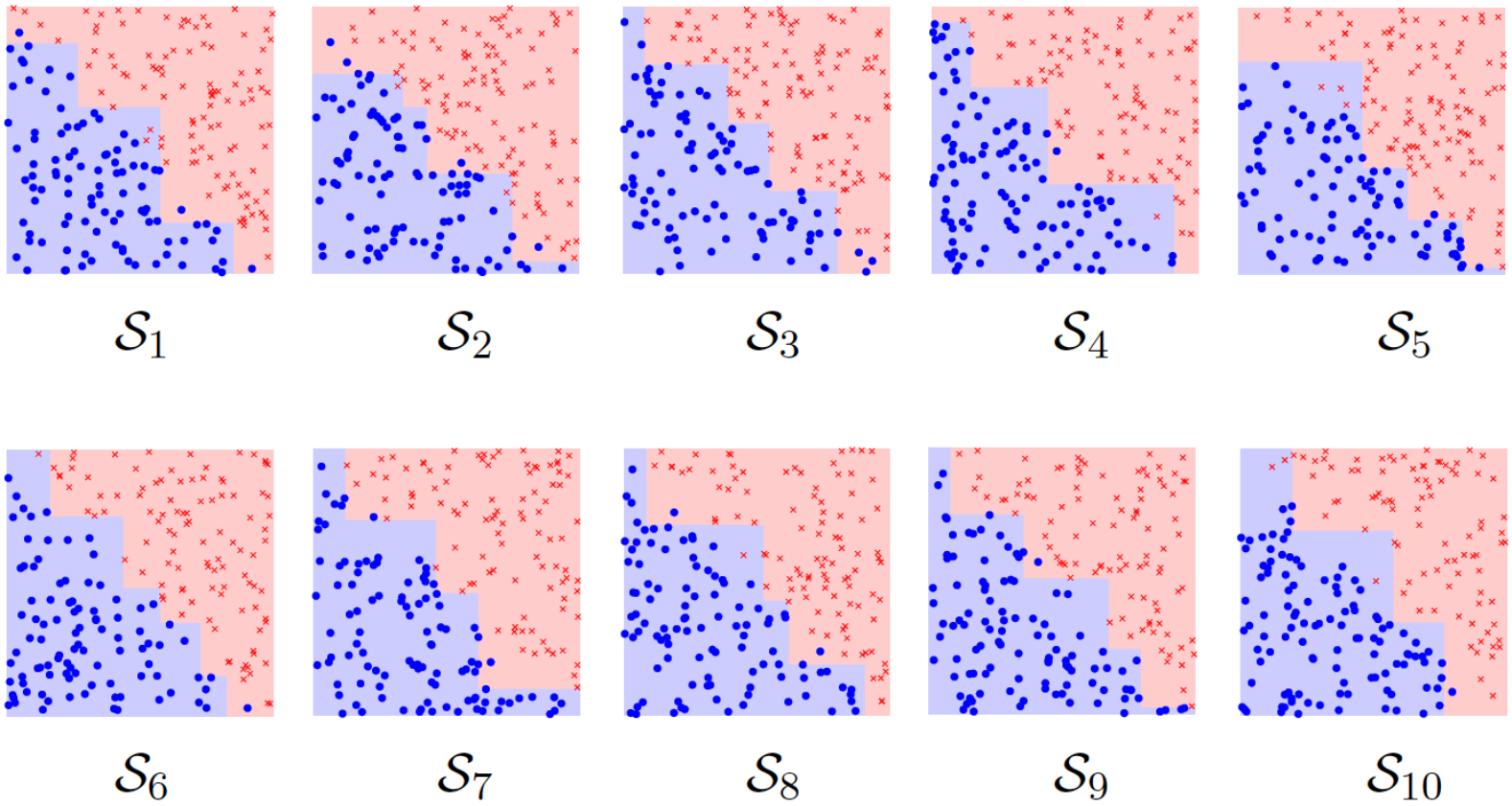


Un muestreo $\mathcal{S}_m$

Arboles de decisión:

- **Sesgo bajo:** en promedio, frontera de decisión cercana a las verdadera.
- **Varianza grande:** frontera de decisión muy sensible a la muestra específica.

Ejemplo: dos clases, los y_m árboles de decisión

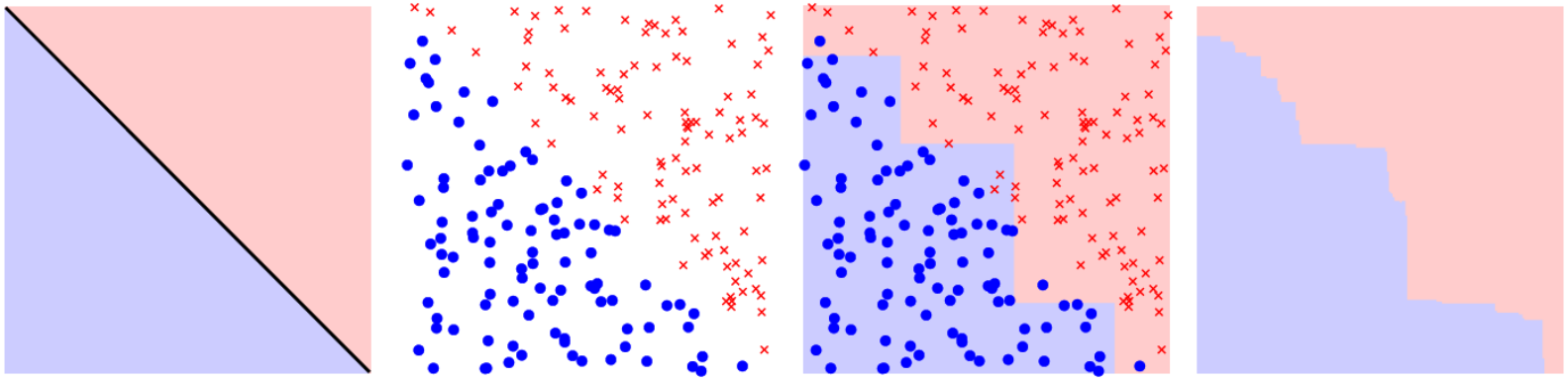


Predictores Bagging

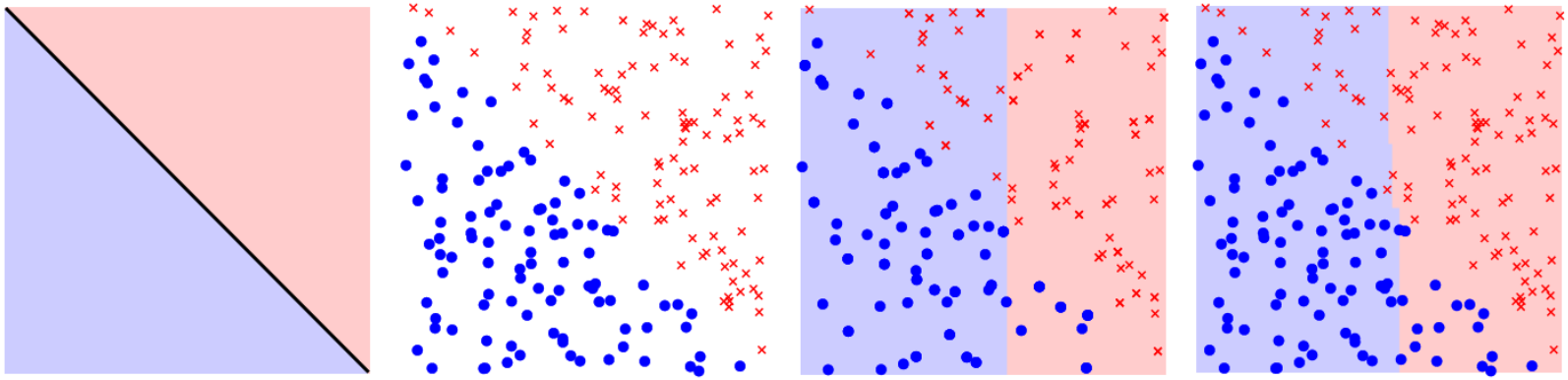
Ejemplo: dos clases, los y_m árboles de decisión

Ejemplo: [Sesgo bajo / Varianza grande] vs. [Sesgo grande / Varianza baja]

Ground truth $\mathcal{S}_1$ y_1 Bagging, $M = 100$



Arboles de decisión



Clasificadores lineales verticales y horizontales

Boosting

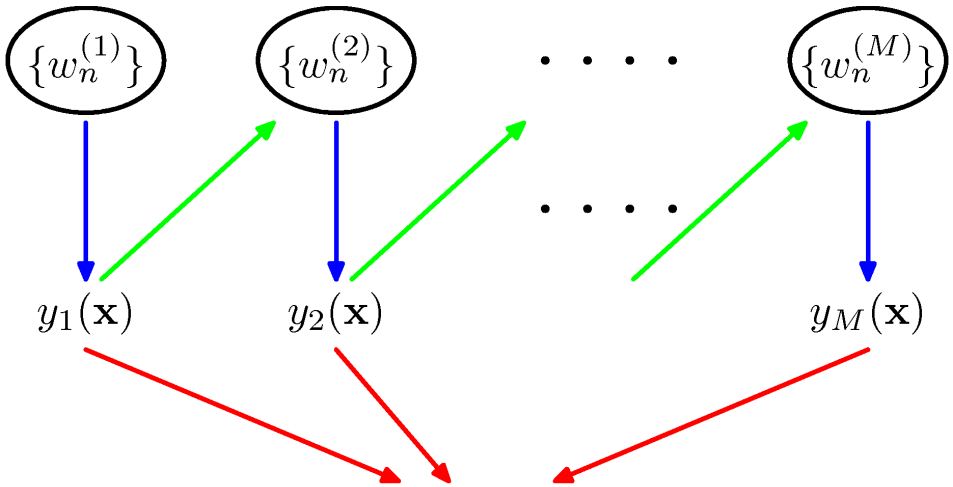
- Una de las ideas más fundamentales en learning, de los últimos 20 años.
- Motivación: proponer procedimiento que combine la salida de múltiples clasificadores débiles para producir un comité performante.
- Diferencia principal con los métodos de comité como bagging: los clasificadores base se entrenan secuencialmente o en cascada.
- Cada clasificador se entrena condicionalmente a la performance de los clasificadores ya entrenados: fuerza a focalizarse en las muestras difíciles.

Describiremos el algoritmo de boosting más popular llamado Adaboost.M1.

Adaboost.M1 (Freund & Shapire, 1997)

- Cada clasificador base se entrena usando una forma ponderada de las muestras: los pesos dependen de la performance de los clasificadores previos en clasificar las muestras.
- A las muestras mal clasificadas por un clasificador base se le asignan un peso mayor a la hora de entrenar el clasificador base siguiente.
- Una vez entrenados todos los clasificadores de la cascada, sus predicciones se combinan mediante mayoría ponderada.

Adaboost.M1



- Cada $y_m(\mathbf{x})$ se entrena en base a una forma ponderada del training set, $\left\{w_n^{(m)}, n = 1, \ldots, N\right\}$.
- Los pesos $w_n^{(m)}$ dependen de la performance de $y_{m-1}(\mathbf{x})$ (y de los anteriores) al clasificar a $\mathbf{x}_n$.
- Una vez que toda la cascada fue entrenada, el meta-clasificador se define como el promedio ponderado de todos los y_m: $Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x})\right)$.
- Los α_m ponderan la performance general de los y_m.

Adaboost.M1: algoritmo

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.15}$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{14.16}$$

   and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}. \tag{14.17}$$

   (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \tag{14.18}$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) \right). \tag{14.19}$$

Adaboost.M1: interpretación

- El primer clasificador $y_1(\mathbf{x})$ es entrenado con pesos uniformes.
- En las iteraciones siguientes los pesos de los datos crecen para los datos mal clasificados y permanecen fijos para los bien clasificados.
- Por consiguiente, los clasificadores sucesivos deben poner mayor énfasis en las muestras mal clasificadas por los clasificadores previos.
- Los ϵ_m representan la tase de error de cada clasificador base par el training set.
- Los pesos de los clasificadores, α_m, son el logaritmo del cociente [tasa de acierto / tasa de error].

Objetivo: minimización del costo exponencial

Se considera la función de error exponencial definida como

$$E_m = \sum_{n=1}^{N} \exp\left(-t_n f_m(\mathbf{x}_n)\right),$$

donde

$$f_m(\mathbf{x}_n) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(\mathbf{x}), \quad t_n \in \{-1, +1\} \text{ son los targets del training set.}$$

Objetivo: minimizar E con respecto a los α_l y a los clasificadores base $y_l(\mathbf{x})$.

Adaboost.M1: justificación (cont.)

- Suponemos que los clasificadores $y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{m-1}(\mathbf{x})$, así como sus coeficientes $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ están fijos, y minimizamos con respecto a $y_m(\mathbf{x})$ y α_m.
- La función de error se escribe

$$E_m = \sum_{n=1}^{N} \exp\left(-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right)$$
$$= \sum_{n=1}^{N} w_n^{(m)} \exp\left(-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right),$$

  donde los $w_n^{(m)}$ son constantes con respecto a $y_m(\mathbf{x})$ y α_m.
- Llamamos $\mathcal{T}_m$ y $\mathcal{M}_m$ a los conjuntos de puntos bien y mal clasificados por $y_m(\mathbf{x})$, respectivamente.

Adaboost.M1: justificación (cont.)

- Operando,

$$E_m = \mathrm{e}^{-\alpha_m/2} \sum_{\mathbf{x}_n \in \mathcal{T}_m} w_n^{(m)} + \mathrm{e}^{\alpha_m/2} \sum_{\mathbf{x}_n \in \mathcal{M}_m} w_n^{(m)}$$

$$= (\mathrm{e}^{\alpha_m/2} - \mathrm{e}^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} \mathbf{1}\{y_m(\mathbf{x}_n) \neq t_n\} + \mathrm{e}^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}.$$

- El segundo término así como el factor multiplicativo del primer término son constantes con respecto a $y_m(\mathbf{x})$, lo cual es equivalente a minimizar la Eq. (14.15) del algoritmo.
- De la misma forma, minimizando con respecto a α_m se obtiene (14.17) (ejercicio).

Adaboost.M1: ejemplo

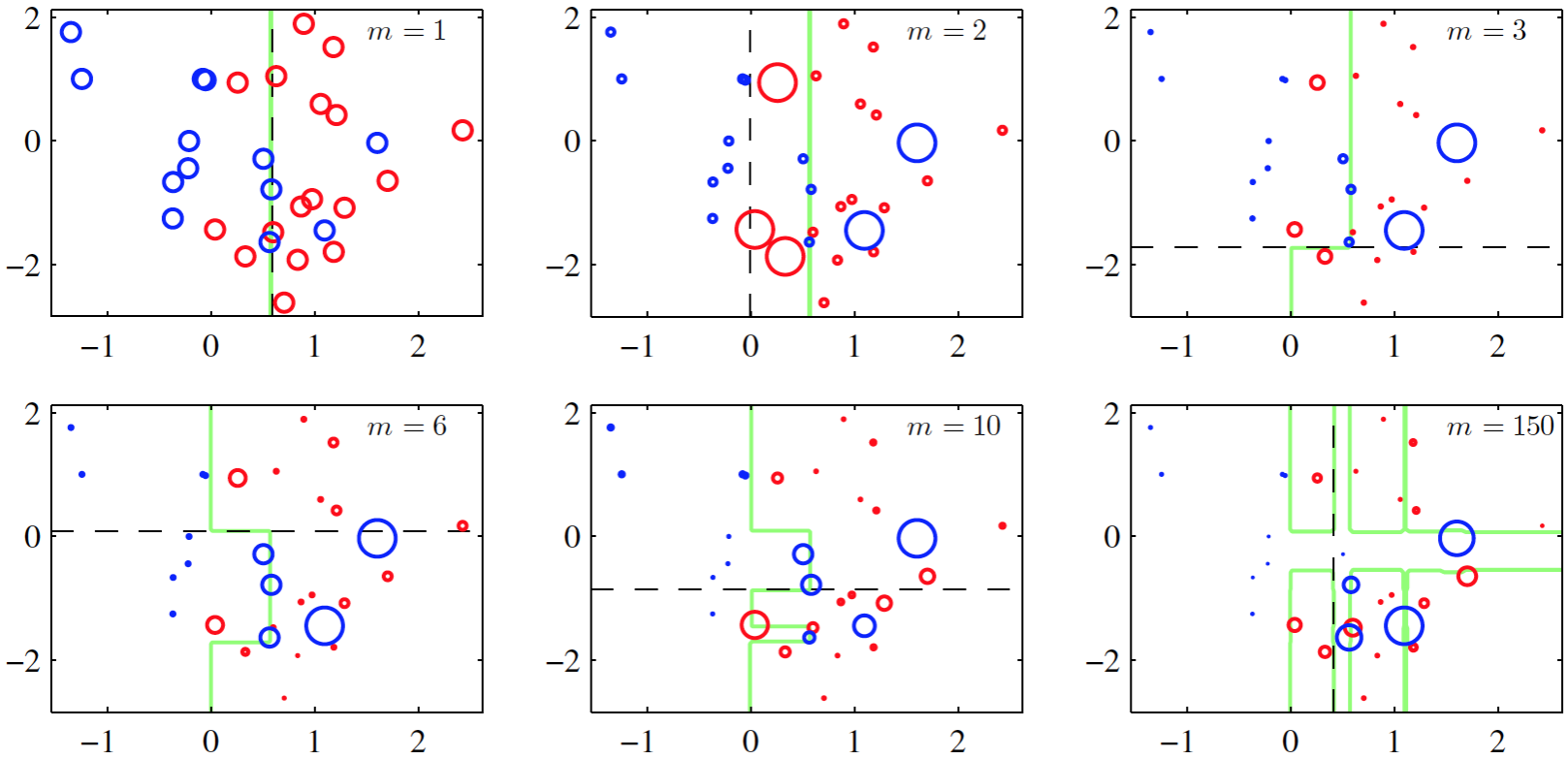


Figure 14.2 Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

Adaboost.M1: aplicación

Detector de objetos (caras) de Viola & Jones (2001)

- Propuesto en 2001, fue la primer aplicación de detección de objetos con resultados competitivos en tiempo real.
- Principal motivación: detección de caras.
- La mayoría de los detectores de caras en cámaras digitales y celulares son implementaciones de este algoritmo.

Detector de Viola & Jones (2001)

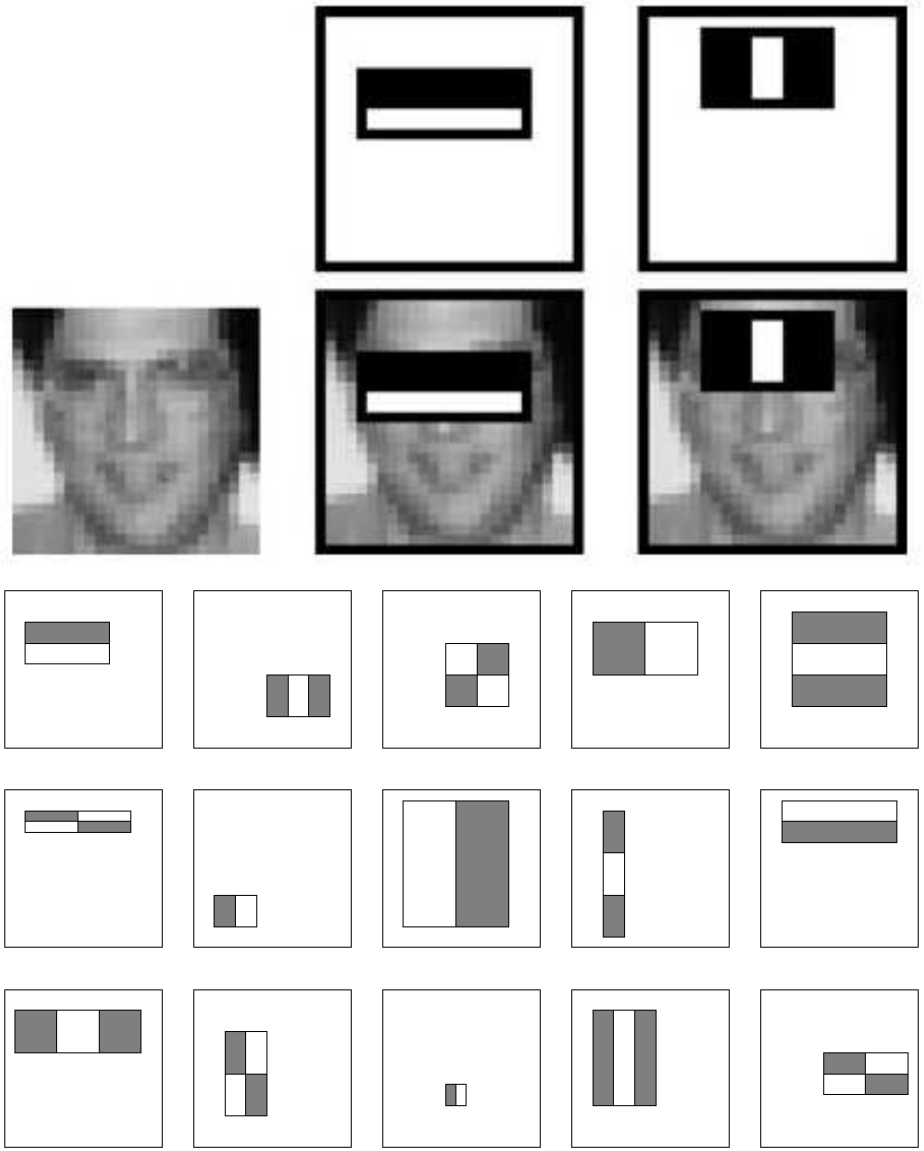
Conjunto de entrenamiento:

- Muestras positivas (caras): $t_n = +1$.
- Muestras negativas (patches sin caras elegidos al azar): $t_n = -1$.
- Todos los patches se re-escalan al mismo tamaño (invarianza a escala).

Training set: muestras positivas

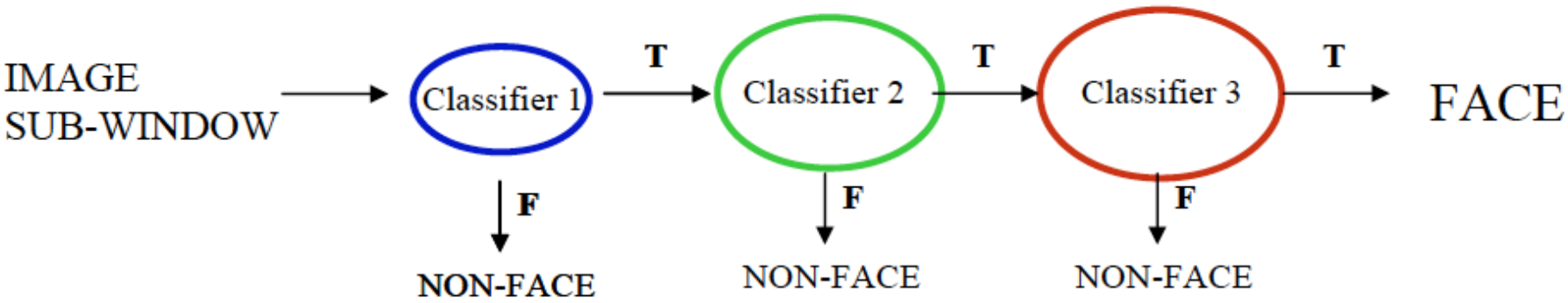
Detector de Viola & Jones (2001)

Clasificadores débiles: correlaciones con funciones de Haar:



Detector de Viola & Jones (2001)

Muy rápido en testing: la mayoría de los patches no tienen caras, y son descartados rápidamente.



Detector de Viola & Jones (2001)