BOOSTING

Nicolás Pérez de la Blanca, DECSAI

¿Como funciona boosting?

- Boosting construye un único clasificador fuerte con **adición** de múltiples clasificadores débiles.
  - Ahora el peso de los datos originales, cambia después de cada iteración de la secuencia, es decir antes de ajustar cada nuevo clasificador débil.
- En **Bagging** cada árbol se ajusta a una muestra distinta y se promedian: Disminución de Varianza
- En Boosting se construye un único clasificador de forma secuencial a partir de todas las muestras, mejorando en cada paso: Disminución de Sesgo

Boosting

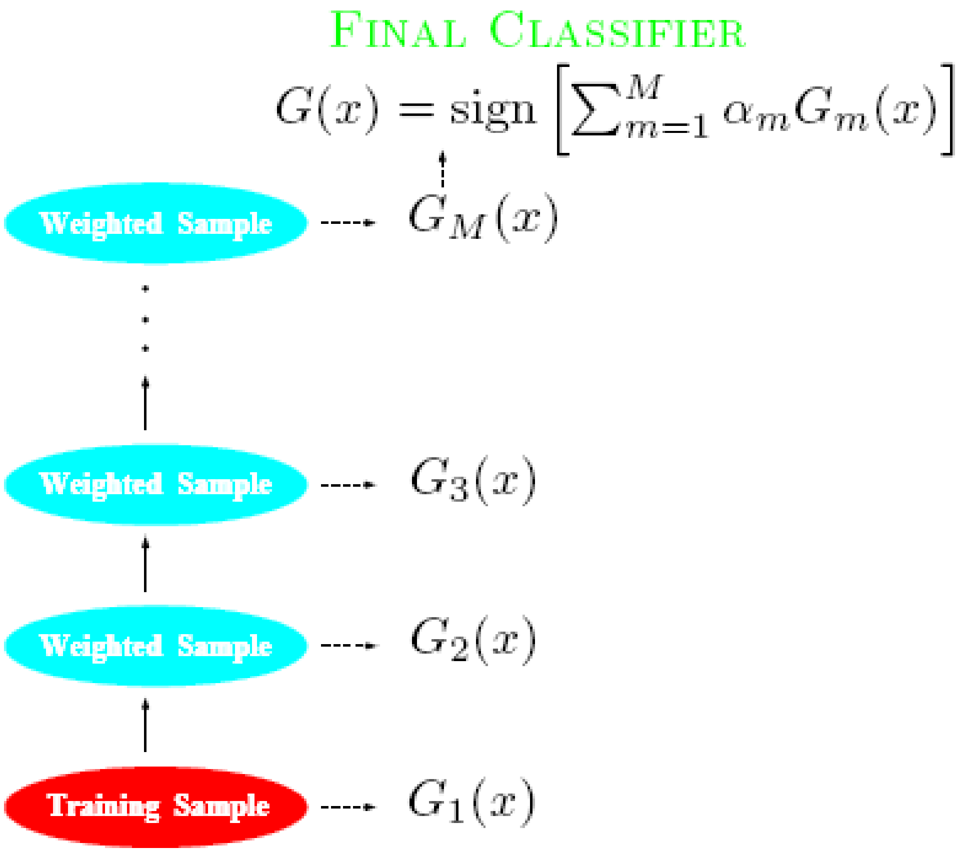


Figure 10.1: *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Ada Boost.M1

- The most popular boosting algorithm – Fruend and Schapire (1997)
- Consider a two-class problem, output variable coded as Y $\in$\{-1,+1\}
- For a predictor variable X, a weak classifier G(X) produces predictions that are in \{-1,+1\}
- The error rate on the training sample is

$$\overline{\mathrm{err}} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$$

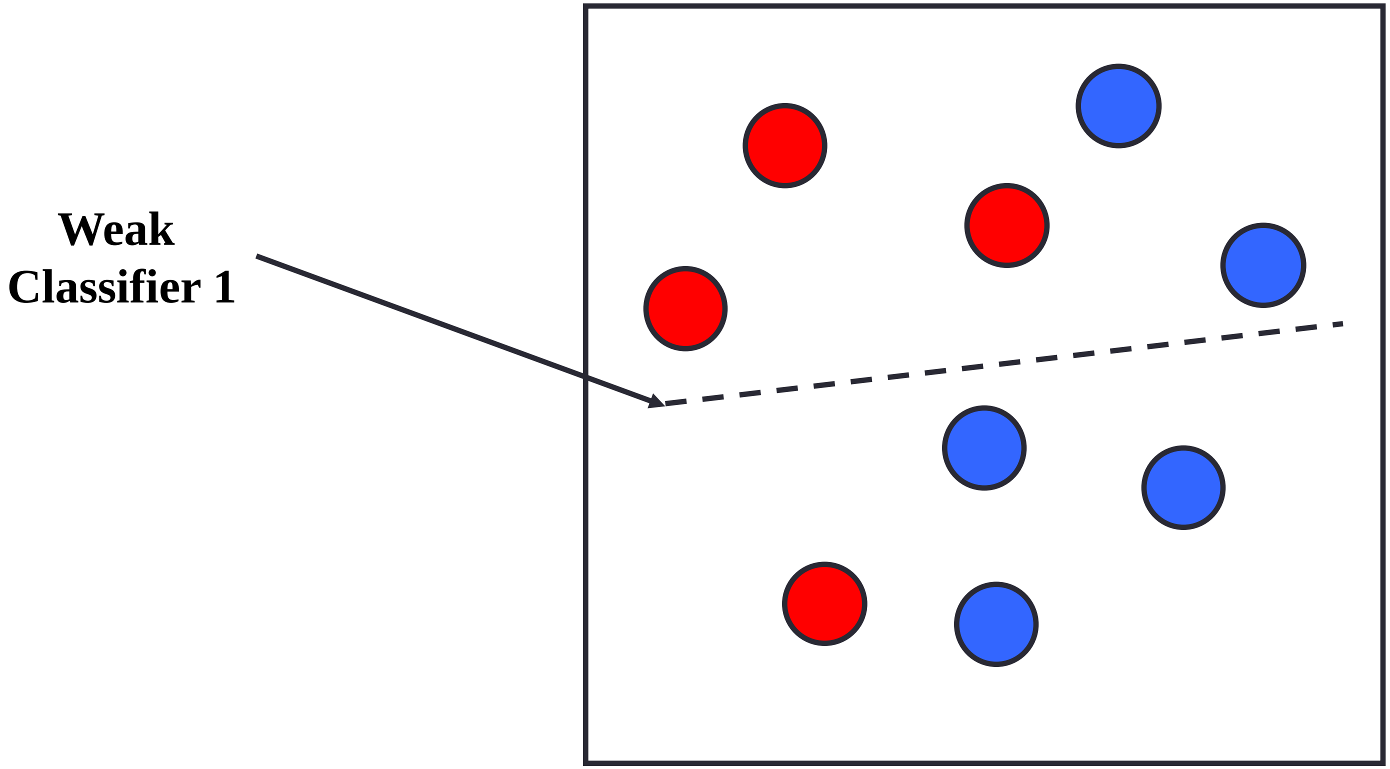
Ada Boost.M1 (Cont'd)

- Sequentially apply the weak classification to repeatedly modified versions of data
- → produce a sequence of weak classifiers $G_m(x)$ m=1,2,..,M
- The predictions from all classifiers are combined via majority vote to produce the final prediction

FINAL CLASSIFIER

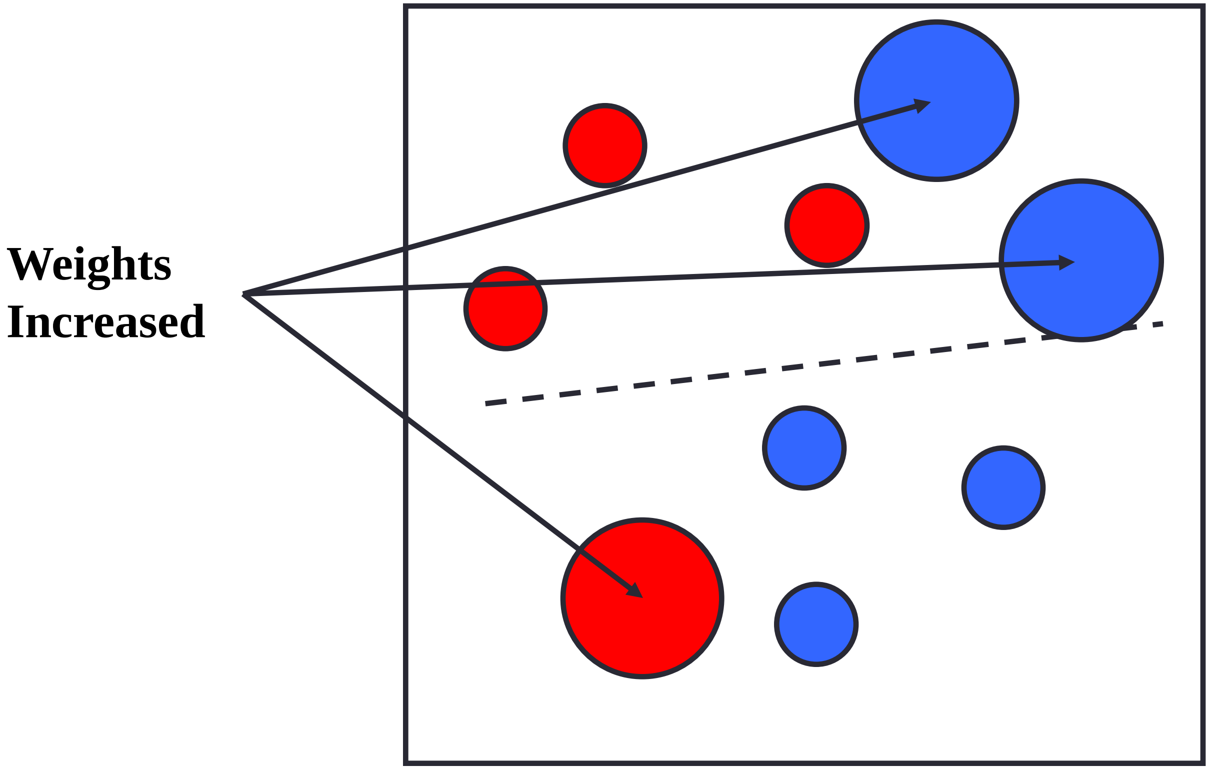
$$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

Boosting intuition

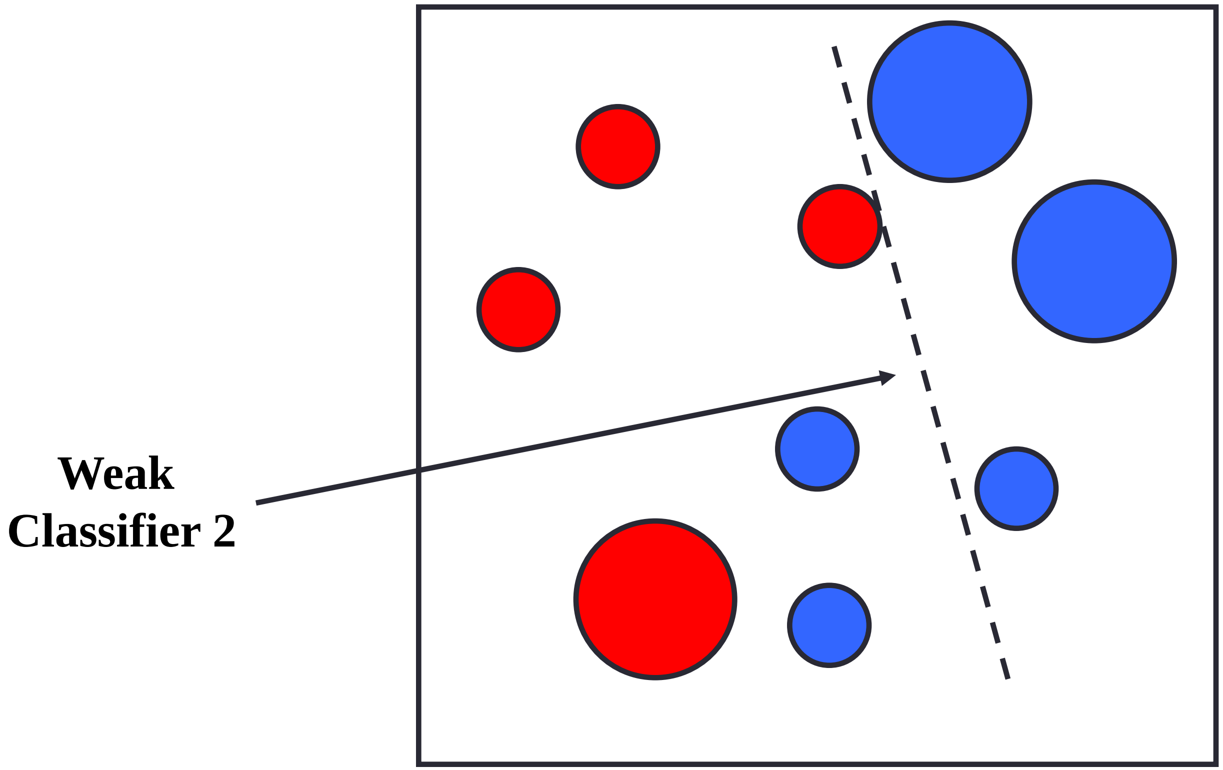


Slide credit: Paul Viola

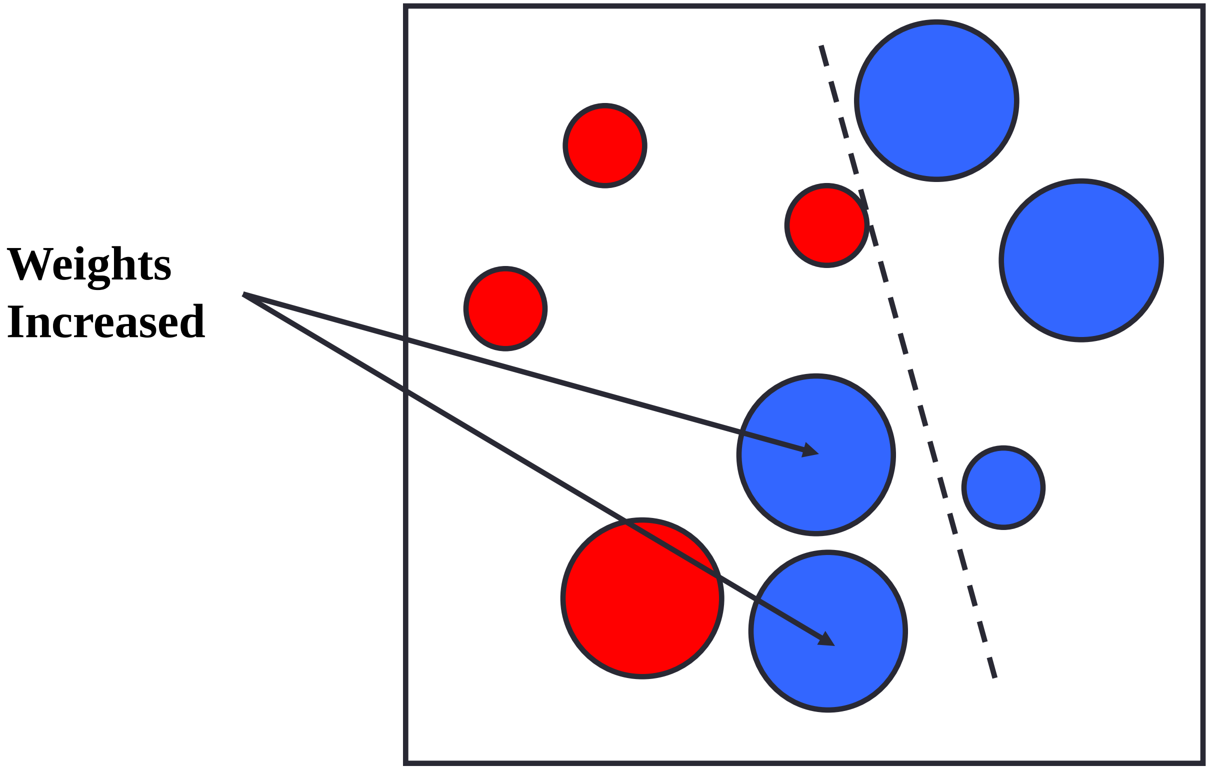
Boosting illustration



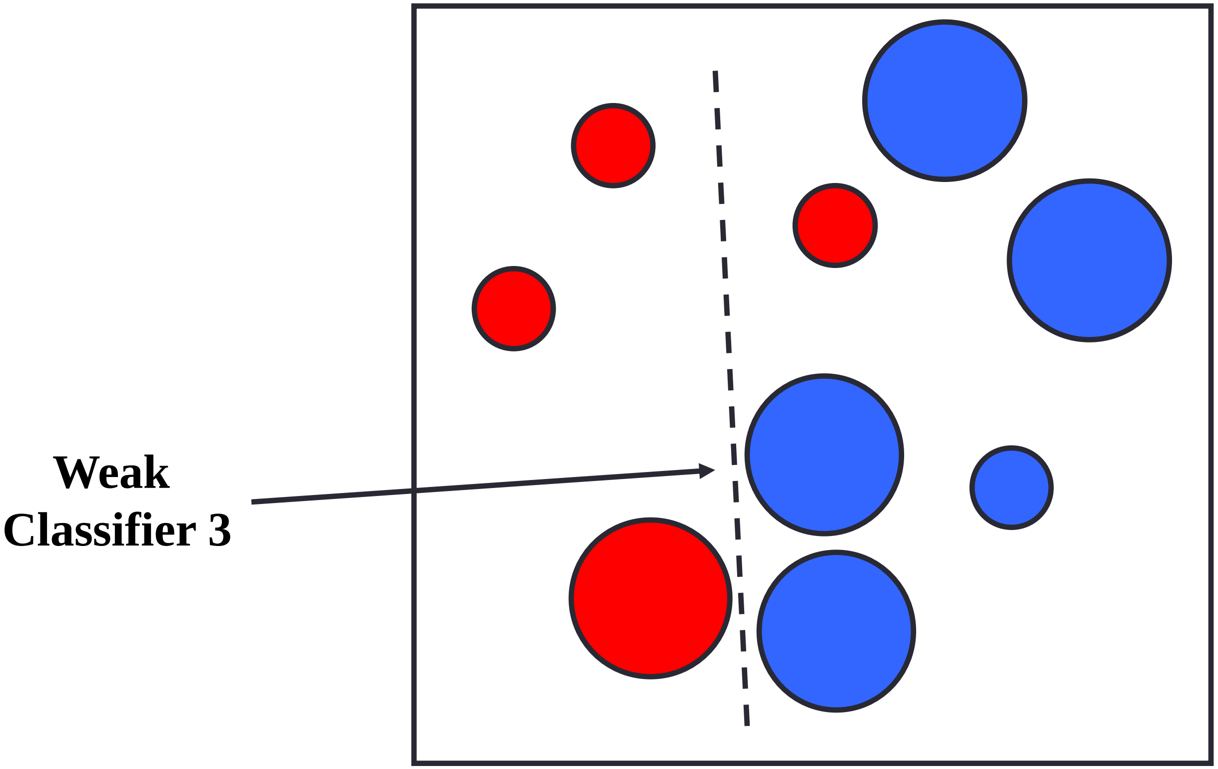
Boosting illustration



Boosting illustration

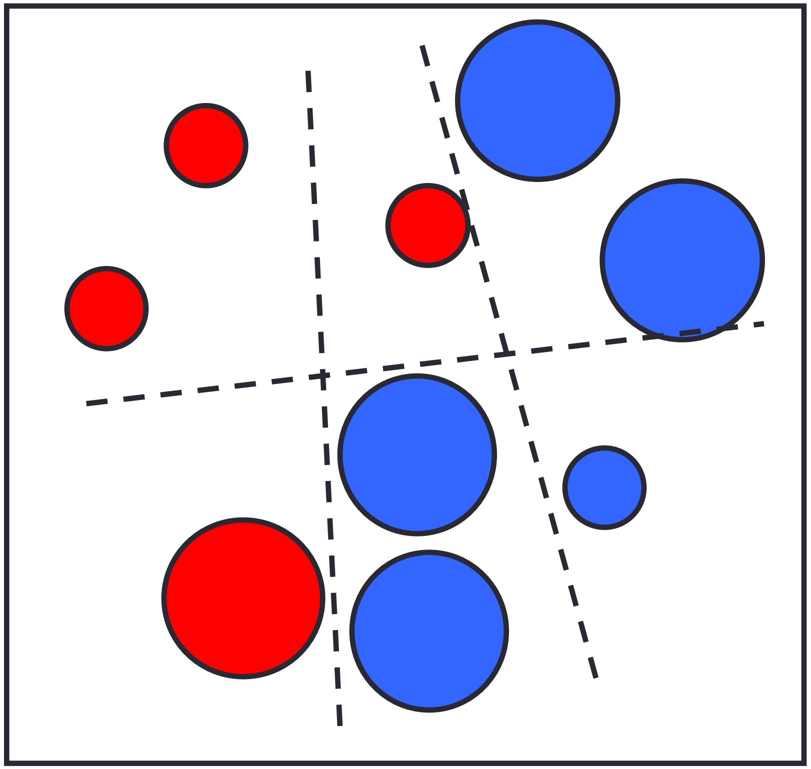


Boosting illustration



Boosting illustration

Final classifier is a combination of weak classifiers



Algorithm AdaBoost.M1

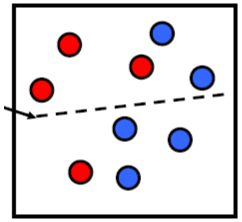
1. Initialize the observ. weights $w_i^{(1)} = 1/N, i = 1, \ldots, N$.
2. For $m = 1$ to M
   - Fit classifier $G_m(x)$ to the training data using weights $w_i^{(m)}$.
   - Compute $err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$.
   - Compute $\alpha_m = \log((1 - err_m)/err_m)$.
   - $w_i^{(m+1)} = w_i^{(m)} \exp[\alpha_m I(y_i \neq G_m(x_i))], i = 1, \ldots, N$.
3. Compute $G(x) = sign(\sum_{i=1}^{M} \alpha_m G_m(x))$.

Some slides borrowed from http://www.stat.ucl.ac.be/

AdaBoost Algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

Example: Adaboost.M1 (Cont'd)

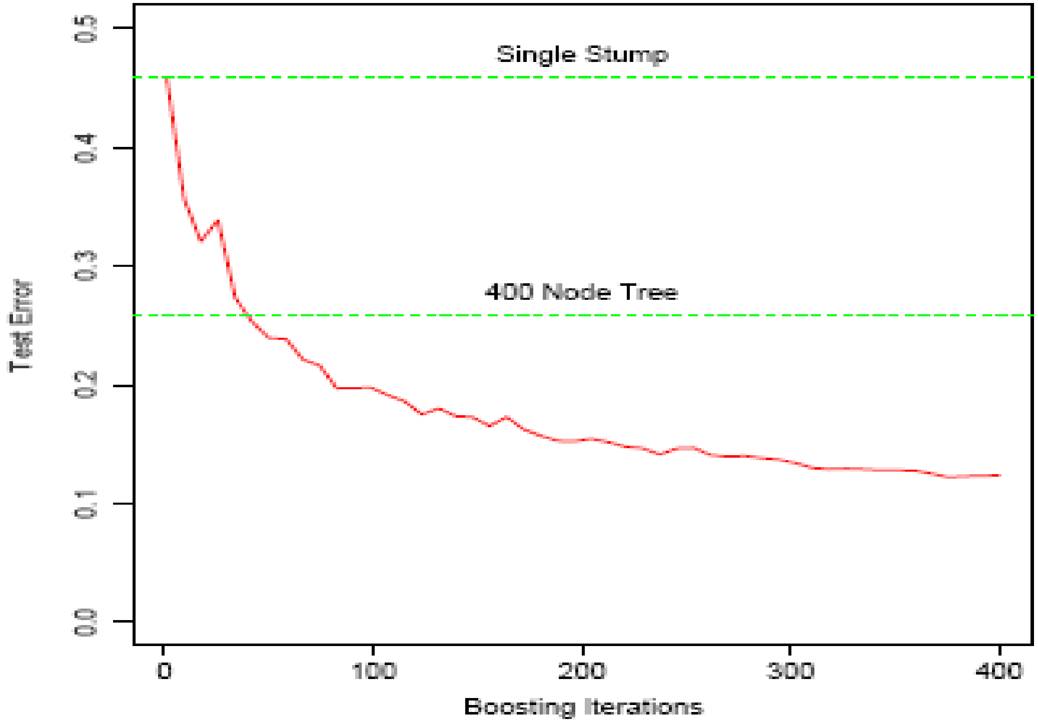


Figure 10.2: *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 400 node classification tree.*

Boosting Fits an Additive Model

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

where $b(x; \gamma_m) = G_m(x) \in \{-1, 1\}$ (for Adaboost) is like a set of elementary "basis functions"

This model is fit by minimizing a loss function L averaged over the training data set:

$$\min_{\{\beta_m, \gamma_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m)\right)$$

Source http://www.stat.ucl.ac.be/

Forward Stagewise Additive Modeling

- An approximate solution to the minimization problem is obtained via forward stagewise additive modeling (greedy algorithm)

1. Initialize $f_0(x) = 0$
2. For $m = 1$ to M
   - Compute
     $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
   - Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Source http://www.stat.ucl.ac.be/

Why adaBoost Works?

- Adaboost is a forward stagewise additive algorithm using the loss function

$$L(y; f(x)) = \exp(-yf(x))$$

with

$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^N \exp\left(-y_i(f_{m-1}(x_i) + \beta G(x_i)\right)$

or $(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^N w_i^{(m)} \exp\left(-\beta y_i G(x_i)\right)$

where $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$.

Source http://www.stat.ucl.ac.be/

Why Boosting Works? (Cont'd)

The solution is :

$$
\begin{aligned}
G_m &= \arg\min_G \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)), \\
\beta_m &= 1/2 \log \frac{1 - err_m}{err_m},
\end{aligned}
$$

where $err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$.

Source http://www.stat.ucl.ac.be/

¿Que minimiza AdaBoost?

- Exponential Loss:

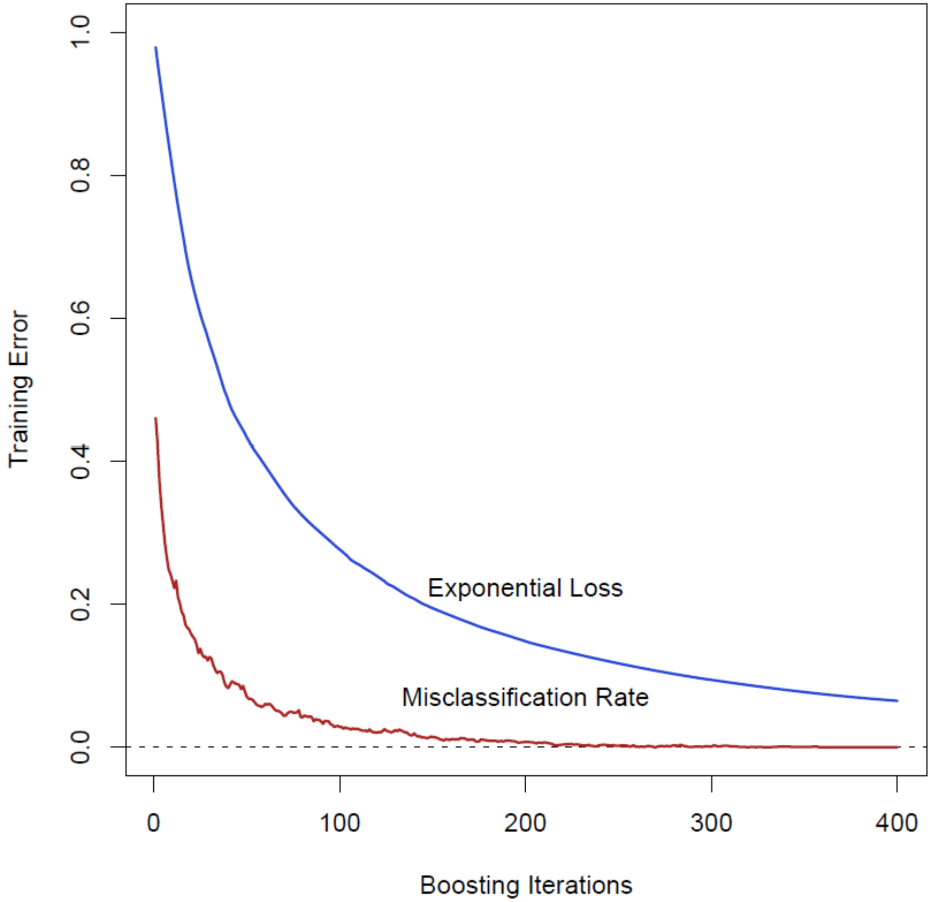
$$L(y, f(x)) = \exp(-yf(x))$$

- La imagen adjunta muestra como Adaboost NO minimiza el criterio "Error de Training" global

- La función que minimiza "Exponential Loss" sigue decreciendo después de que el error de training sea cero.

- El error de test también sigue decreciendo.

- AdaBoost tiene un modelo de RL como equivalente probabilistico



Loss Function

- $yf(x)$ is called the Margin
- The classification rule implies that observations with positive margin $y_i f(x_i) > 0$ were classified correctly, but the negative margin ones are incorrect
- The decision boundary is given by the $f(x) = 0$
- The loss criterion should penalize the negative margins more heavily than the positive ones

- **ADABOOST is a margin maximizing classifier**

Practical Advantages of AdaBoost

- fast
- simple and easy to program
- no parameters to tune (except T)
- flexible — can combine with any learning algorithm
- no prior knowledge needed about weak learner
- provably effective, provided can consistently find rough rules of thumb

  → shift in mind set — goal now is merely to find classifiers barely better than random guessing
- versatile
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification

Caveats

- performance of AdaBoost depends on data and weak learner
- consistent with theory, AdaBoost can fail if
  - weak classifiers too complex → overfitting
- weak classifiers too weak ($\gamma_t \to 0$ too quickly)

  → underfitting

  → low margins → overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise

Boosting en regresión

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Boosting en regresión: Parámetros

- Hay que fijar tres parámetros:
  - El número de árboles B (si es muy grande podría sobre-ajustar). Se puede estimar por validación-cruzada.
  - El valor del parámetro de amortiguación λ. Valores entre 0.01 y 0.001 son típicos. Si el valor de λ es muy pequeño podemos necesitar un valor de B muy grande.
  - El número de particiones del árbol d que controla la complejidad de cada uno de los árboles individuales. Normalmente d=1 y tenemos árboles con una sola partición (funciones stump)
    - En el caso d=1 tenemos un modelo que solo usa 1 variable en cada paso y por tanto ajusta un modelo aditivo (fácilmente interpretable!)
    - El parámetro d se denomina profundidad-de-interacción ya que d particiones podrían hacer participar a d variables distintas.
    - Aunque d=1 funciona bien en muchas aplicaciones, en general 2<= d <=3 funciona bien en el contexto de boosting.

Gradient Boosting

- Let's consider the problem as $\min_{f} L(f) = \sum_i L(y_i, f(x_i))$
- Numerical optimization could be considered to find *f* directly,

$$\mathbf{f} = \sum_{i=0}^{M} \mathbf{h}_i \text{ , } \mathbf{h}_i \in R^N$$

Where $\mathbf{h}_i$ is obtained adaptively from some numerical optimization criteria.

For any differentiable *L(f)* the gradient can be computed and the steepest descent criteria applied to compute h_i

$$\mathbf{h}_m = -\rho_m \mathrm{g}_m, \;\; \mathrm{g}_{mi} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$

$$\rho_m = arg\min_{\rho} L(\mathbf{f}_{m-1} - \rho \mathrm{g}_m), \;\; \mathbf{f}_m = \mathbf{f}_{m-1} + \mathbf{h}_m$$

Gradient Boosting

1. Initialize $f_0(x) = 0$
2. For $m = 1$ to M
   - Compute $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
   - Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Forward Stagewise Additive Model

- FSAM:
  - Cons: the vector component are not independent since they belong to a fitted tree model.
  - Pro: it allows generalization

- Good compromise: Use trees to fit by least squares the negative gradients on each iteration.

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

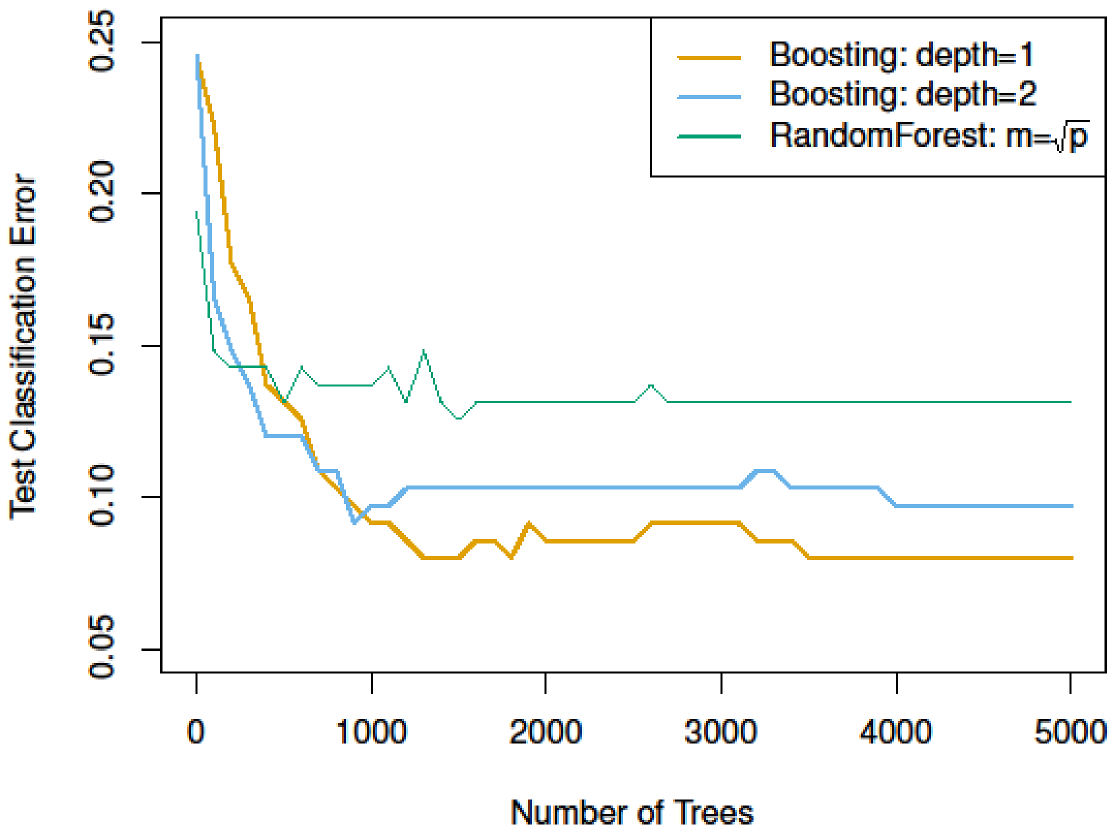
   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Comparativa RF vs Boosting



Conjunto de datos de expresiones genéticas de 15 -clases

Questions ?