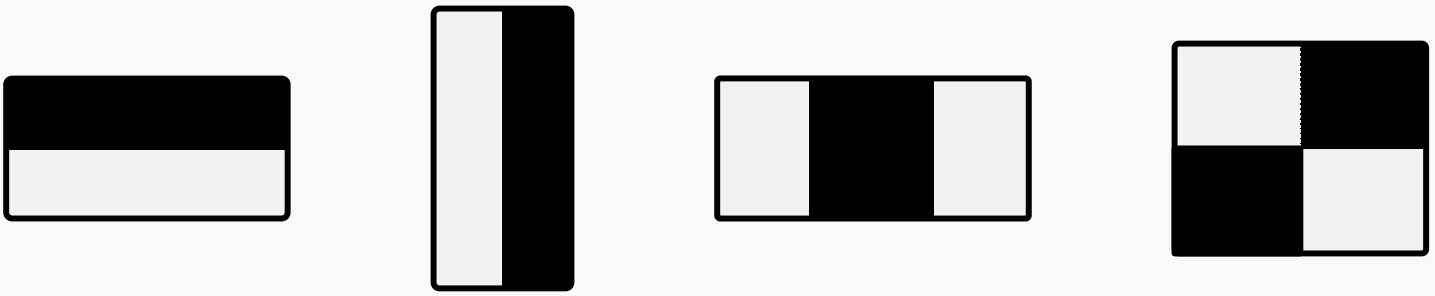
boosting

The motivation behind **Boosting** originated from a question asked by Michael Kearns: "**Can a 'weak' learning algorithm** (slightly better than random guessing) **be turned into a 'strong' learning algorithm** (error rate arbitrarily close to 0)**?**"

Schapire and French worked on the above problem, examine ways to make the algorithm create numerous classifiers and determine the best method of combination. The struggled existed with having a **single dataset** being able to create different classifiers. The answer was to **reweight the data in many ways** and feed them into the algorithm; creating a **weak classifier** for each (**reweighted**) dataset. Ultimately, a **weighted average** would be computed of the weak classifiers.

One of the most noted application of **combining weak classifiers** was that of Viola and Jones in their image detection experiments. The concept of weak classifiers for facial recognition is as follows:



The above are representative of matrix filters that scan images to determine contrasts in the pixel densities between the light and darker segments of each filter. The pixel densities are subtracted to determine features in an image. The filters are applied in all different shapes, sizes and orientations. The intuition behind these weak classifiers is covered in detail in the Applied Machine Learning Documentation. For this purpose, the motivation behind combined many weak classifiers to create a strong learning algorithm to classify new labels in a dataset is the intent of the illustration.

The following application of **weak classifiers** above will be in the context of the **AdaBoost** algorithm.

adaboost

AdaBoost functions by reweighting the dataset in many iterations to compute different results out of the weak learning algorithm each time around.

AdaBoost Pseudocode

Assign observation i the weight of $d_{1i} = \frac{1}{n}$ (equal weights)

For t=1:T

Train weak learning algorithm using data weighted by $d_{1i} = \frac{1}{n}$ ...

...producing weak classifier h_t.

Choose coefficient α_t.

Update weights:

$y_i h_t(x_i) = 1$ if correct, -1 if incorrect

$$d_{t+1,i} = \frac{d_{t,i}\exp(-\alpha_t(y_i h_t(x_i))}{Z_t}$$ with Z_t serving as a normalization factor

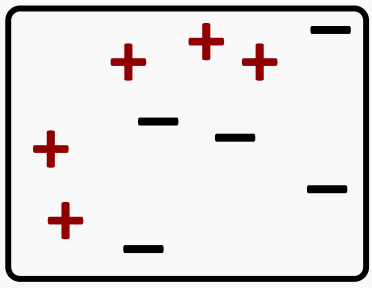
End

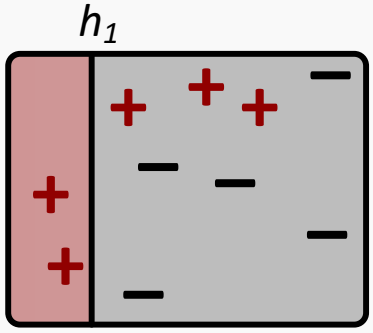
Output the final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right)$

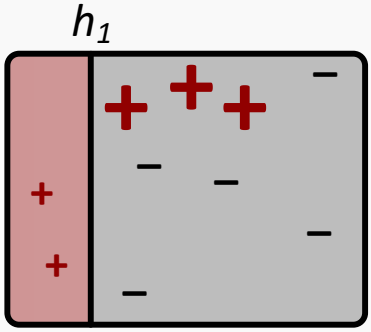
Princeton Boosting Survey

The following illustrative example assumes any decision boundaries are **horizontal** or **vertical** lines:

Each point begins the algorithm with equal weights:

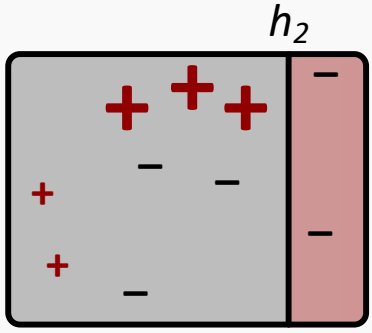


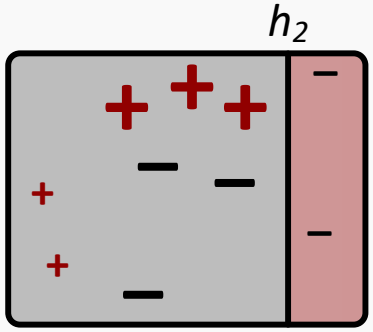




The **first Boosting** iteration (**weak classifier**) is run and the coefficient weight is chosen → $\boldsymbol{\alpha_1 = 0.42}$

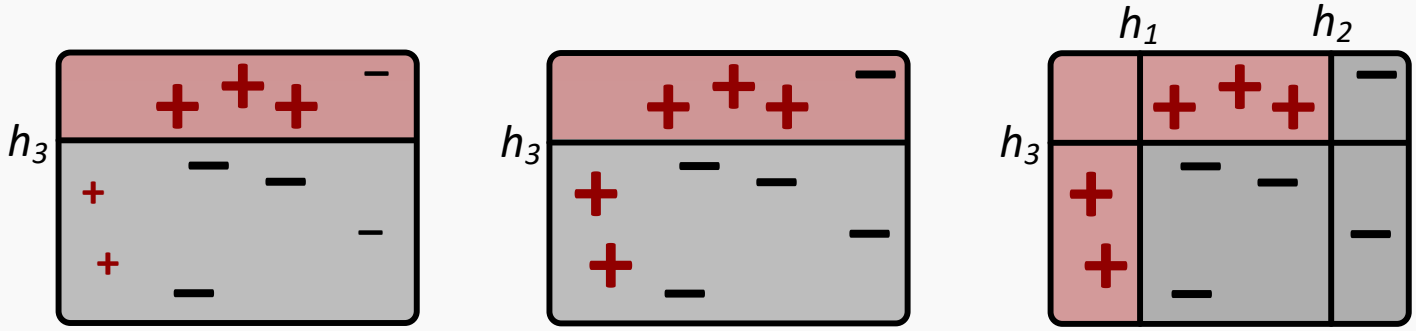
The weights of the **misclassified points** are **increased**; **decrease correctly classified points** weights.





The **second Boosting** iteration is run and the coefficient weight is chosen → $\boldsymbol{\alpha_1 = 0.66}$

The weights of the **misclassified points** are **increased**; **decrease correctly classified points** weights.



The **third Boosting** iteration is run and the coefficient weight is chosen → $\boldsymbol{\alpha_1 = 0.66}$

The weights of the **misclassified points** are **increased**; **decrease correctly classified points** weights.

Finally, output the **weighted average classifier** $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x_i))$:

$$H = \text{sign}\left(0.42 \; + 0.66 \; + 0.93 \; \right)$$

AdaBoost Coeffisicients Update

The alpha α examines how well the classifier performs at each iteration; the **classification error**:

$$\text{Error}_t = \sum_{i \,:\, h_t(x_i) \neq y_i} d_t = \text{sum of weights of misclassified points}$$

Intuitively, if the error rate is large, alpha α will conversely be assigned a small value to minimize the classifiers impact on the model. Plotting alpha α against the error illustrates the latter intuition:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - Error_t}{Error_t}\right)$$