# Ensemble methods



## Learning theory

- Probability distribution $P$ over $\mathcal{X} \times \{0,1\}$; let $(X,Y) \sim P$.
- We get $S := \{(x_i, y_i)\}_{i=1}^n$, an iid sample from $P$.
- **Goal**: Fix $\epsilon, \delta \in (0,1)$. With probability at least $1-\delta$ (over random choice of $S$), learn a classifier $\hat{f} \colon \mathcal{X} \to \{-1,+1\}$ with low error rate

$$\mathrm{err}(\hat{f}) \; = \; P(\hat{f}(X) \neq Y) \; \leq \; \epsilon \, .$$

- **Basic question**: When is this possible?
  - Suppose I even promise you that there is a perfect classifier from a particular *function class* $\mathcal{F}$.
    (E.g., $\mathcal{F}$ = linear classifiers or $\mathcal{F}$ = decision trees.)
  - **Default**: Empirical Risk Minimization (i.e., pick classifier from $\mathcal{F}$ with lowest training error rate), but this might be computationally difficult (e.g., for decision trees).
- **Another question**: Is it easier to learn just non-trivial classifiers in $\mathcal{F}$ (i.e., better than random guessing)?



## Boosting

**Boosting**: Using a learning algorithm that provides "rough rules-of-thumb" to construct a very accurate predictor.

**Motivation**:
Easy to construct classification rules that are correct more-often-than-not (e.g., "If $\geq 5\%$ of the e-mail characters are dollar signs, then it's spam."),

but seems hard to find a single rule that is almost always correct.

**Basic idea**:

**Input**: training data $S$

For $t = 1, 2, \ldots, T$:

1. Choose subset of examples $S_t \subseteq S$ (or a distribution over $S$).
2. Use "**weak learning**" algorithm to get classifier: $f_t := \mathrm{WL}(S_t)$.

**Return** an "ensemble classifier" based on $f_1, f_2, \ldots, f_T$.



## Boosting: history

1984 Valiant and Kearns ask whether "boosting" is theoretically possible (formalized in the PAC learning model).

1989 Schapire creates first boosting algorithm, solving the open problem of Valiant and Kearns.

1990 Freund creates an optimal boosting algorithm (Boost-by-majority).

1992 Drucker, Schapire, and Simard empirically observe practical limitations of early boosting algorithms.

1995 **Freund and Schapire** create **AdaBoost**—a boosting algorithm with practical advantages over early boosting algorithms.

**Winner of 2004 ACM Paris Kanellakis Award**:

*For their "seminal work and distinguished contributions [...] to the development of the theory and practice of boosting, a general and provably effective method of producing arbitrarily accurate prediction rules by combining weak learning rules"; specifically, for AdaBoost, which "can be used to significantly reduce the error of algorithms used in statistical analysis, spam filtering, fraud detection, optical character recognition, and market segmentation, among other applications".*

# AdaBoost

**input** Training data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{X} \times \{-1, +1\}$.

**initialize** $D_1(i) := 1/n$ for each $i = 1, 2, \dots, n$ (a probability distribution).

**for** $t = 1, 2, \dots, T$ **do**

Give $D_t$-weighted examples to WL; get back $f_t \colon \mathcal{X} \to \{-1, +1\}$.

Update weights:

$$z_t := \sum_{i=1}^n D_t(i) \cdot y_i f_t(x_i) \in [-1, +1]$$

$$\alpha_t := \frac{1}{2} \ln \frac{1 + z_t}{1 - z_t} \in \mathbb{R} \quad \text{(weight of } f_t\text{)}$$

$$D_{t+1}(i) := D_t(i) \exp\big(-\alpha_t \cdot y_i f_t(x_i)\big) / Z_t \quad \text{for each } i = 1, 2, \dots, n,$$

where $Z_t > 0$ is normalizer that makes $D_{t+1}$ a probability distribution.

**end for**

**return** Final classifier $\hat{f}(x) := \operatorname{sign}\left(\sum_{t=1}^T \alpha_t \cdot f_t(x)\right)$.

(Let $\operatorname{sign}(z) := 1$ if $z > 0$ and $\operatorname{sign}(z) := -1$ if $z \le 0$.)



# Interpretation

## Interpreting $z_t$

Suppose $(X, Y) \sim D_t$. If

$$P(f(X) = Y) = \frac{1}{2} + \gamma_t,$$

then

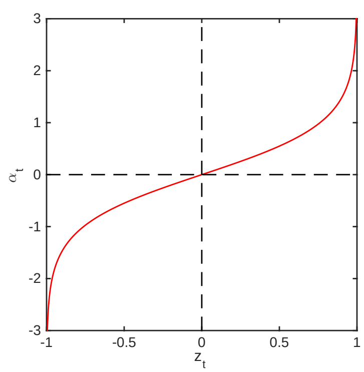
$$z_t = \sum_{i=1}^n D_t(i) \cdot y_i f(x_i) = 2\gamma_t \in [-1, +1].$$

- $z_t = 0 \iff$ random guessing w.r.t. $D_t$.
- $z_t > 0 \iff$ better than random guessing w.r.t. $D_t$.
- $z_t < 0 \iff$ better off using the opposite of $f$'s predictions.



# Interpretation

**Classifier weights** $\alpha_t = \frac{1}{2} \ln \frac{1+z_t}{1-z_t}$

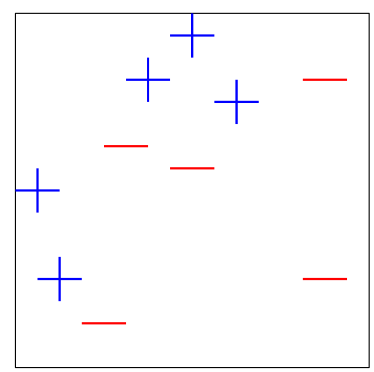



**Example weights** $D_{t+1}(i)$

$$D_{t+1}(i) \propto D_t(i) \cdot \exp(-\alpha_t \cdot y_i f_t(x_i)).$$



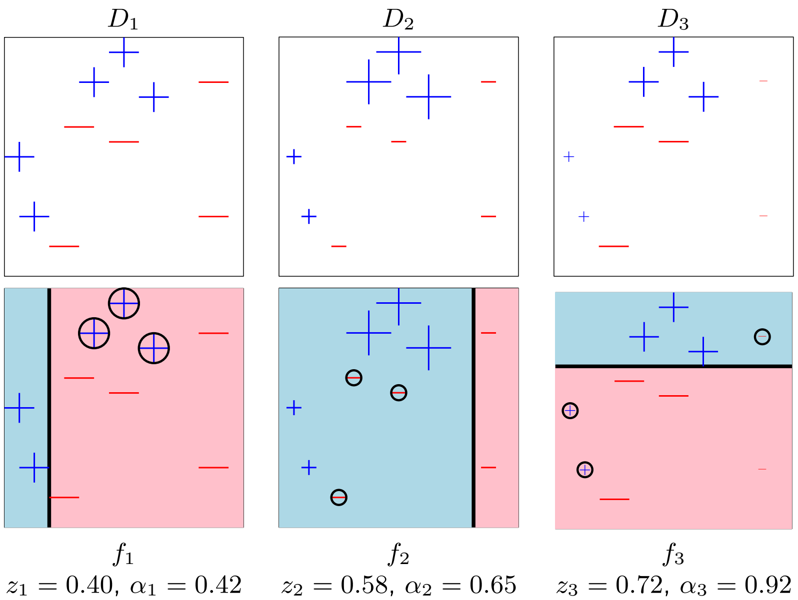
# Example: AdaBoost with decision stumps



**Weak learning algorithm** WL: ERM with $\mathcal{F} =$ "decision stumps" on $\mathbb{R}^2$ (i.e., axis-aligned threshold functions $\boldsymbol{x} \mapsto \operatorname{sign}(vx_i - t)$). Straightforward to handle importance weights in ERM.

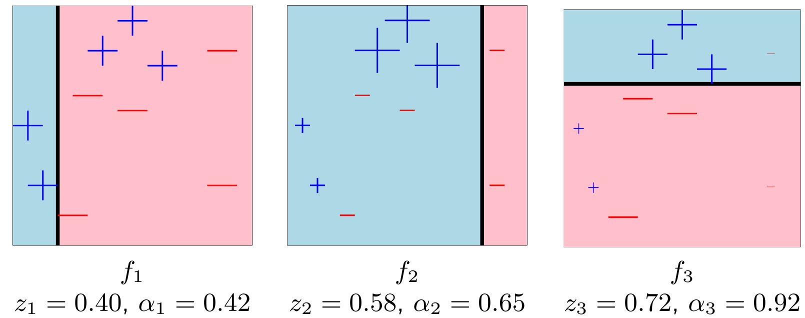
(Example from Figures 1.1 and 1.2 of Schapire & Freund text.)

# Example: execution of AdaBoost



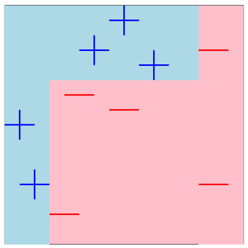



# Example: final classifier from AdaBoost




**Final classifier**
$\hat{f}(x) = \mathrm{sign}(0.42 f_1(x) + 0.65 f_2(x) + 0.92 f_3(x))$

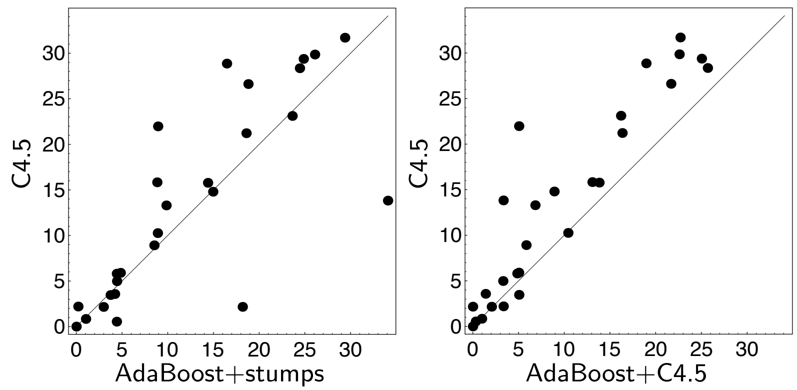
(Zero training error rate!)





# Empirical results

**Test error rates of C4.5 and AdaBoost on several classification problems.**
Each point represents a single classification problem/dataset from UCI repository.




C4.5 = popular algorithm for learning decision trees.

(Figure 1.3 from Schapire & Freund text.)



# Training error rate of final classifier

Recall $\gamma_t := P(f_t(X) = Y) - 1/2 = z_t/2$ when $(X, Y) \sim D_t$.

**Training error rate of final classifier from AdaBoost**:

$$\mathrm{err}(\hat{f}, \{(x_i, y_i)\}_{i=1}^{n}) \ \leq \ \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2\right).$$

If average $\bar{\gamma}^2 := \frac{1}{T} \sum_{t=1}^{T} \gamma_t^2 > 0$, then training error rate is $\leq \exp\left(-2\bar{\gamma}^2 T\right)$.

**"AdaBoost" = "Adaptive Boosting"**
Some $\gamma_t$ could be small, even negative—only care about overall average $\bar{\gamma}^2$.

**What about true error rate?**

## Combining classifiers

Let $\mathcal{F}$ be the function class used by the **weak learning algorithm** WL.

The function class used by AdaBoost is

$$\mathcal{F}_T \;:=\; \left\{ x \mapsto \operatorname{sign}\left( \sum_{t=1}^{T} \alpha_t f_t(x) \right) : f_1, f_2, \dots, f_T \in \mathcal{F}, \alpha_1, \alpha_2, \dots, \alpha_T \in \mathbb{R} \right\}$$

i.e., linear combinations of $T$ functions from $\mathcal{F}$.

Complexity of $\mathcal{F}_T$ grows *linearly* with $T$.

**Theoretical guarantee** (e.g., when $\mathcal{F}$ = decision stumps in $\mathbb{R}^d$):
With high probability (over random choice of training sample),

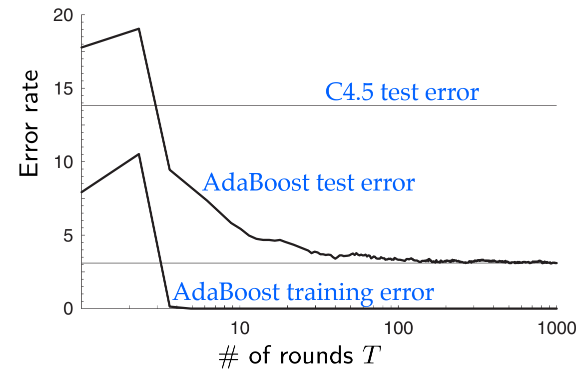
$$\operatorname{err}(\hat{f}) \;\le\; \underbrace{\exp\left(-2\bar{\gamma}^2 T\right)}_{\text{Training error rate}} \;+\; \underbrace{O\left(\sqrt{\frac{T \log d}{n}}\right)}_{\text{Error due to finite sample}} .$$

**Theory suggests danger of overfitting when $T$ is very large.**
Indeed, this does happen sometimes ... **but often not**!



## A typical run of boosting

AdaBoost+C4.5 on "letters" dataset.




(# nodes across all decision trees in $\hat{f}$ is $>2 \times 10^6$)

*Training error rate is zero after just five rounds,*
but test error rate continues to decrease, even up to 1000 rounds!

(Figure 1.7 from Schapire & Freund text)



## Boosting the margin

Final classifier from AdaBoost:

$$\hat{f}(x) \;=\; \operatorname{sign}\underbrace{\left( \frac{\sum_{t=1}^{T} \alpha_t f_t(x)}{\sum_{t=1}^{T} |\alpha_t|} \right)}_{g(x) \in [-1,+1]} .$$

Call $y \cdot g(x) \in [-1,+1]$ the **margin** achieved on example $(x, y)$.

**New theory** [Schapire, Freund, Bartlett, and Lee, 1998]:

- **Larger margins** $\Rightarrow$ **better resistance to overfitting**, independent of $T$.
- AdaBoost tends to increase margins on training examples.

(Similar but not the same as SVM margins.)

**On "letters" dataset**:

| | $T = 5$ | $T = 100$ | $T = 1000$ |
|---|---|---|---|
| training error rate | 0.0% | 0.0% | 0.0% |
| test error rate | 8.4% | 3.3% | 3.1% |
| % margins $\le 0.5$ | 7.7% | 0.0% | 0.0% |
| min. margin | 0.14 | 0.52 | 0.55 |



## Linear classifiers

Regard function class $\mathcal{F}$ used by weak learning algorithm as "feature functions":

$$x \;\mapsto\; \boldsymbol{\phi}(x) \;:=\; \big(f(x) : f \in \mathcal{F}\big) \;\in\; \{-1,+1\}^{\mathcal{F}}$$

(possibly infinite dimensional!).

AdaBoost's final classifier is a *linear classifier* in $\{-1,+1\}^{\mathcal{F}}$:

$$\hat{f}(x) \;=\; \operatorname{sign}\left( \sum_{t=1}^{T} \alpha_t f_t(x) \right) \;=\; \operatorname{sign}\left( \sum_{f \in \mathcal{F}} w_f f(x) \right) \;=\; \operatorname{sign}\big(\langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle\big)$$

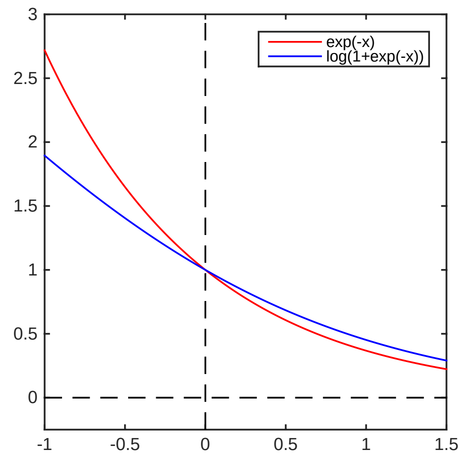
where

$$w_f \;:=\; \sum_{t=1}^{T} \alpha_t \cdot \mathbb{1}\{f_t = f\} \quad \forall f \in \mathcal{F} .$$

# Exponential loss

AdaBoost is a particular "coordinate descent" algorithm (similar to but not the same as gradient descent) for

$$\min_{\boldsymbol{w}\in\mathbb{R}^{\mathcal{F}}} \quad \frac{1}{n}\sum_{i=1}^{n} \exp\big(-y_i\langle \boldsymbol{w}, \boldsymbol{\phi}(x_i)\rangle\big) .$$






# More on boosting

**Many variants of boosting**:

- AdaBoost with different loss functions.
- Boosted decision trees = boosting + decision trees.
- Boosting algorithms for *ranking* and *multi-class*.
- Boosting algorithms that are robust to certain kinds of noise.
- Boosting for online learning algorithms (very new!).
- . . .

**Many connections between boosting and other subjects**:

- Game theory, online learning
- "Geometry" of information (replace $\|\cdot\|_2^2$ with relative entropy divergence)
- Computational complexity
- . . .

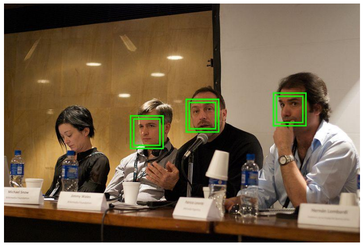


# An application of AdaBoost

# Application: face detection

## Face detection

**Problem**: Given an image, locate all of the faces.



**As a classification problem**:

- Divide up images into patches (at varying scales, e.g., $24\times24$, $48\times48$).
- Learn classifier $f\colon \mathcal{X}\to\mathcal{Y}$, where $\mathcal{Y} = \{\text{face}, \text{not face}\}$.

Many other things built on top of face detectors (e.g., face tracking, face recognizers); now in every digital camera and iPhoto/Picasa-like software.

**Main problem**: how to make this **very fast**.

## Face detectors via AdaBoost [Viola & Jones, 2001]

**Face detector architecture by Viola & Jones (2001)**: major achievement in computer vision; **detector actually usable in real-time**.

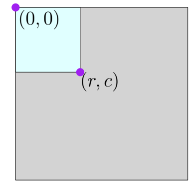
- Think of each image patch ($d \times d$-pixel gray-scale) as a vector $\boldsymbol{x} \in [0,1]^{d^2}$.
- Used weak learning algorithm that picks linear classifiers $f_{\boldsymbol{w},t}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - t)$, where $\boldsymbol{w}$ has a very particular form:

- AdaBoost combines many "rules-of-thumb" of this form.
  - Very simple.
  - **Extremely fast to evaluate** via pre-computation.



## Viola & Jones "integral image" trick

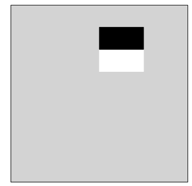
**"Integral image" trick**:




For every image, pre-compute

$s(r,c)$ = sum of pixel values in rectangle from $(0,0)$ to $(r,c)$

(single pass through image).



To compute inner product

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \text{average pixel value in black box} - \text{average pixel value in white box}$$

just need to add and subtract a few $s(r,c)$ values.

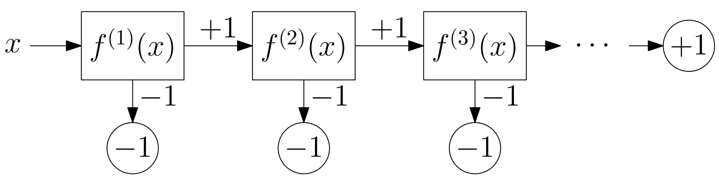
$\Rightarrow$ Evaluating "rules-of-thumb" classifiers is **extremely fast**.



## Viola & Jones cascade architecture

**Problem**: severe class imbalance (most patches don't contain a face).

**Solution**: Train several classifiers (each using AdaBoost), and arrange in a special kind of **decision list** called a **cascade**:




- Each $f^{(\ell)}$ is trained (using AdaBoost), adjust threshold (before passing through $\operatorname{sign}$) to *minimize false negative rate*.
- Can make $f^{(\ell)}$ in later stages more complex than in earlier stages, since most examples don't make it to the end.

$\Rightarrow$ (Cascade) classifier evaluation **extremely fast**.



## Viola & Jones detector: example results

## Summary

Two key points:

- AdaBoost effectively combines many fast-to-evaluate "weak classifiers".
- Cascade structure optimizes speed for common case.



# Bagging

## Bagging

**Bagging** = **B**ootstrap **agg**regat**ing** (Leo Breiman, 1994).

**Input**: training data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{X} \times \{-1,+1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick $n$ examples *with replacement* from training data $\longrightarrow$ $\{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ (a *bootstrap* sample).
2. Run learning algorithm on $\{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ $\longrightarrow$ classifier $f_t$.

**Return** a majority vote classifier over $f_1, f_2, \ldots, f_T$.



## Aside: sampling with replacement

**Question**: if $n$ individuals are picked from a population of size $n$ *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

**Answer**:

$$\left(1 - \frac{1}{n}\right)^n$$

For large $n$:

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679\,.$$

**Implications for bagging**:

- Each bootstrap sample contains about $63\%$ of the data set.
- Remaining $37\%$ can be used to estimate error rate of classifier trained on the bootstrap sample.
- Can average across bootstrap samples to get estimate of *bagged classifier*'s error rate (sort of).

## Random Forests

**Random Forests** (Leo Breiman, 2001).

**Input**: training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick $n$ examples *with replacement* from training data
   $\longrightarrow \quad \{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ (a *bootstrap* sample).
2. Run variant of decision tree learning algorithm on $\{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^n$, where each split is chosen by only considering a random subset of $\sqrt{d}$ features (rather than all $d$ features)
   $\longrightarrow \quad$ decision tree classifier $f_t$.

**Return** a majority vote classifier over $f_1, f_2, \ldots, f_T$.



## Key takeaways

1. Theoretical concept of weak and strong learning.
2. AdaBoost algorithm; concept of margins in boosting.
3. Interpreting AdaBoost's final classifier as a linear classifier, and interpreting AdaBoost as a coordinate descent algorithm.
4. Structure of decision lists / cascades.
5. Concept of bootstrap samples; bagging and random forests.