COMS 4771 Lecture 14

1. Boosting

Boosting

What is boosting?

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

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

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

Basic idea:

Input: training data S, "weak" learning algorithm $\mathcal{A}$

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

1. Choose subset of examples $\boxed{S_t \subseteq S}$ (or a distribution over S).
2. Call weak learning algorithm to get classifier: $f_t := \mathcal{A}(S_t)$.

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

Boosting: history

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

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.

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).

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.

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.

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".

Input Training data S from $\mathcal{X} \times \{\pm 1\}$.
Weak learning algorithm $\mathcal{A}$ (for importance-weighted classification).

1: **initialize** $D_1(x, y) := 1/|S|$ for each $(x, y) \in S$ (a probability distribution).
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\boxed{\text{Give } D_t\text{-weighted examples to } \mathcal{A}}$; get back $f_t \colon \mathcal{X} \to \{\pm 1\}$.

Dt is the weight for all samples

4: Update weights:

the bigger the Zt, the accurate the

$$z_t := \sum_{(x,y) \in S} D_t(x, y) \cdot y f_t(x) \in [-1, +1]$$

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

$$D_{t+1}(x, y) \propto D_t(x, y) \exp(-\alpha_t \cdot y f_t(x)) \quad \text{for each } (x, y) \in S.$$

5: **end for**
6: **return** Final classifier $f_{\text{final}}(x) := \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot f_t(x)\right)$.

when f(x) is wrong, the larger the exp(-a y f(x)) the probability for a single point !

the larger, the better

Interpreting z_t

$$\text{If} \quad \Pr_{(X,Y)\sim D_t}[f_t(X) = Y] \;=\; \frac{1}{2} + \gamma_t \quad \text{for some } \gamma_t \in [-1/2, +1/2],$$

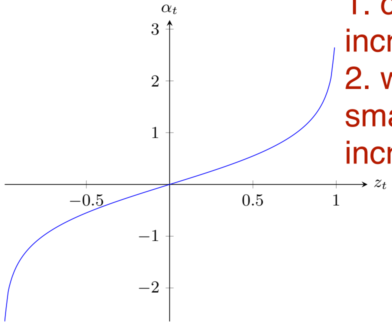
$$\text{then} \quad z_t \;=\; \sum_{(x,y)\in S} D_t(x,y) \cdot y f_t(x) \;=\; 2\gamma_t \in [-1, +1].$$

Zt is for classifier

$z_t = 0 \iff$ 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_t's predictions.

Interpretation

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

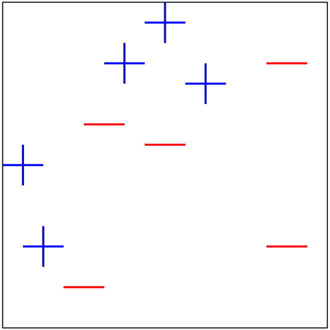


as zt increase:
1. classifier weight increase
2. wrongly labeled smaple's weight increase

Example weights $D_{t+1}(x, y)$

$$D_{t+1}(x, y) \propto D_t(x, y) \cdot \exp(-\alpha_t \cdot y f_t(x))$$

Example: AdaBoost with decision stumps

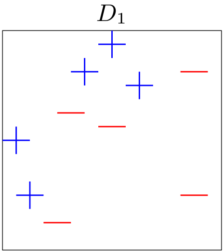


one-level
decision tree

Weak learning algorithm $\mathcal{A}$: 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.

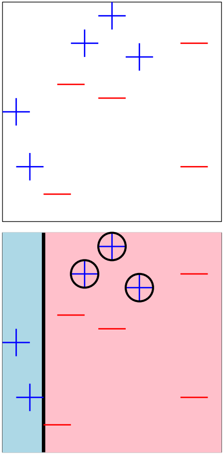
(Example from Figures 1.1 and 1.2 of Schapire & Freund text.)

Example: execution of AdaBoost



Example: execution of AdaBoost

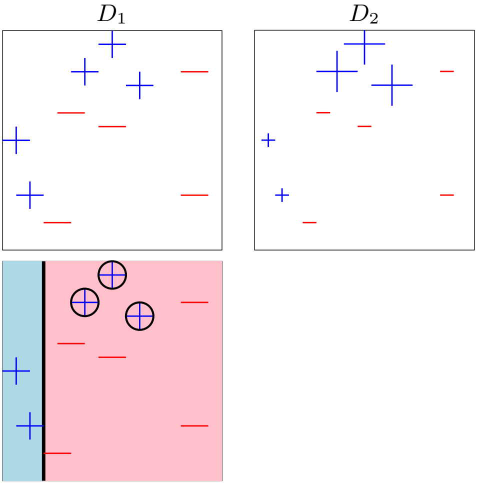
D_1



f_1

$z_1 = 0.40,\ \alpha_1 = 0.42$

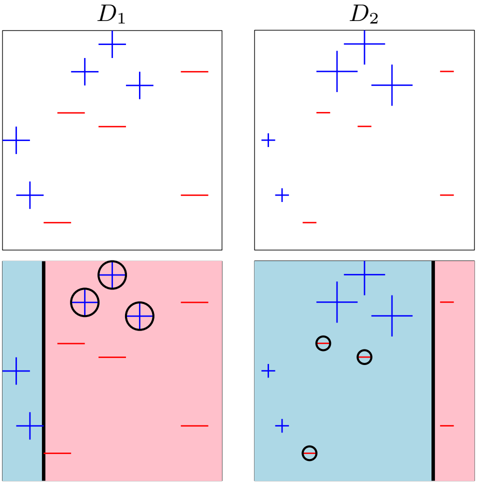
Example: execution of AdaBoost



f_1

$z_1 = 0.40, \ \alpha_1 = 0.42$

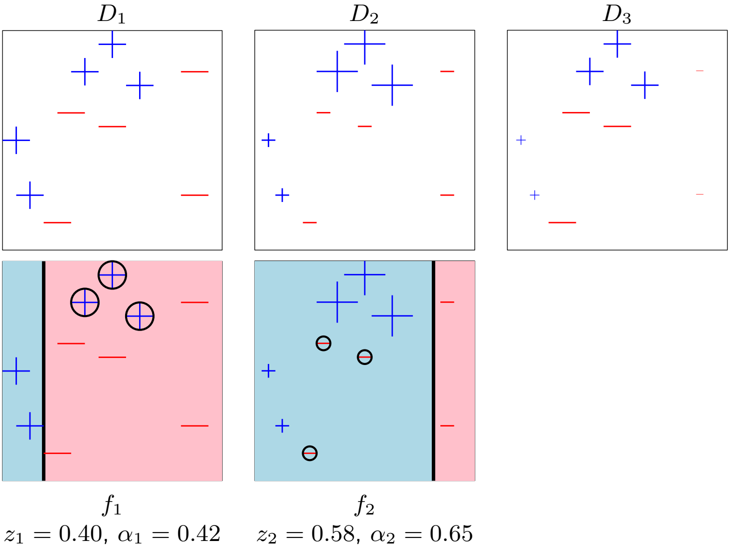
Example: execution of AdaBoost



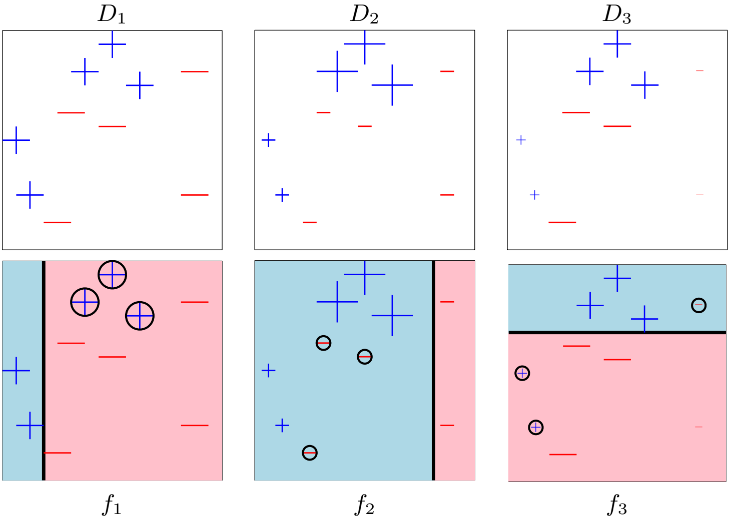
f_1 $\quad$ f_2

$z_1 = 0.40,\ \alpha_1 = 0.42$ $\quad$ $z_2 = 0.58,\ \alpha_2 = 0.65$

Example: execution of AdaBoost

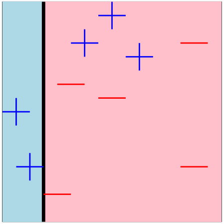


Example: execution of AdaBoost



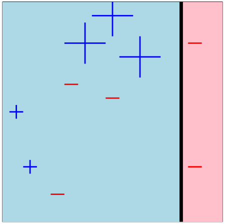
$z_1 = 0.40,\ \alpha_1 = 0.42$ $z_2 = 0.58,\ \alpha_2 = 0.65$ $z_3 = 0.72,\ \alpha_3 = 0.92$

Example: final classifier from AdaBoost



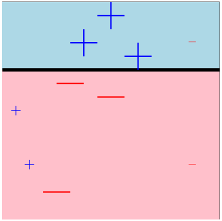
f_1

$z_1 = 0.40,\ \alpha_1 = 0.42$



f_2

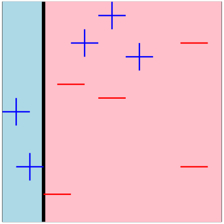
$z_2 = 0.58,\ \alpha_2 = 0.65$



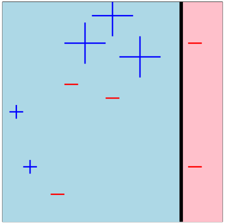
f_3

$z_3 = 0.72,\ \alpha_3 = 0.92$

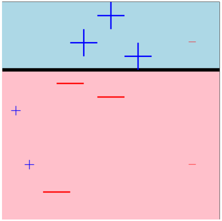
Example: final classifier from AdaBoost



f_1
$z_1 = 0.40,\ \alpha_1 = 0.42$



f_2
$z_2 = 0.58,\ \alpha_2 = 0.65$

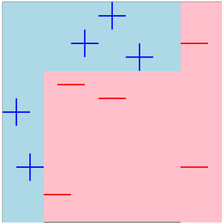


f_3
$z_3 = 0.72,\ \alpha_3 = 0.92$

Final classifier

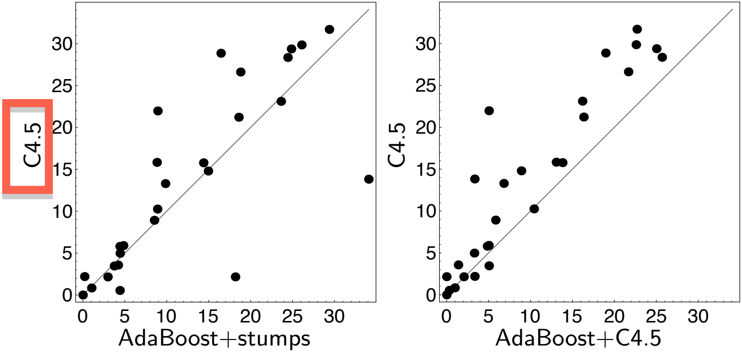
$f_{\mathsf{final}}(x) = \operatorname{sign}(0.42 f_1(x) + 0.65 f_2(x) + 0.92 f_3(x))$

(Zero training error!)



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 of final classifier

Recall $\gamma_t := \Pr_{(X,Y)\sim D_t}[f_t(X) = Y] - 1/2 = z_t/2$.

Training error of final classifier from AdaBoost:

$$\operatorname{err}(f_{\mathsf{final}}, S) \;\leq\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

Training error of final classifier

Recall $\gamma_t := \Pr_{(X,Y)\sim D_t}[f_t(X) = Y] - 1/2 = z_t/2$.

Training error of final classifier from AdaBoost:

$$\operatorname{err}(f_{\mathsf{final}}, S) \;\leq\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

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

Training error of final classifier

Recall $\gamma_t := \Pr_{(X,Y)\sim D_t}[f_t(X) = Y] - 1/2 = z_t/2$.

Training error of final classifier from AdaBoost:

$$\operatorname{err}(f_{\mathsf{final}}, S) \;\leq\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

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

"AdaBoost" = "<u>Ada</u>ptive <u>Boost</u>ing"
Some γ_t could be small (or even negative!)—only care about overall average $\bar{\gamma}^2$.

Training error of final classifier

Recall $\gamma_t := \Pr_{(X,Y)\sim D_t}[f_t(X) = Y] - 1/2 = z_t/2$.

Training error of final classifier from AdaBoost:

$$\operatorname{err}(f_{\mathsf{final}}, S) \;\leq\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

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

"AdaBoost" = "<u>Ada</u>ptive <u>Boost</u>ing"
Some γ_t could be small (or even negative!)—only care about overall average $\bar{\gamma}^2$.

What about true error?

Combining classifiers

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

Combining classifiers

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

The function class used by AdaBoost is

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

i.e., "linear combinations of T functions from $\mathcal{F}$".
Complexity of $\mathcal{F}_T$ grows *linearly* with T.

Combining classifiers

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

The function class used by AdaBoost is

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

i.e., "linear combinations of T functions from $\mathcal{F}$".
Complexity of $\mathcal{F}_T$ grows *linearly* with T.

Theoretical guarantee: with high probability over choice of i.i.d. sample S,

$$\operatorname{err}(f) \;\le\; \operatorname{err}(f, S) + O\left(\sqrt{\frac{T \log |\mathcal{F}_{|S}|}{|S|}} \right) \quad \forall f \in \mathcal{F}_T.$$

Combining classifiers

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

The function class used by AdaBoost is

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

i.e., "linear combinations of T functions from $\mathcal{F}$".
Complexity of $\mathcal{F}_T$ grows *linearly* with T.

Theoretical guarantee: with high probability over choice of i.i.d. sample S,

$$\operatorname{err}(f) \ \leq \ \operatorname{err}(f, S) + O\left(\sqrt{\frac{T \log |\mathcal{F}_{|S}|}{|S|}} \right) \quad \forall f \in \mathcal{F}_T.$$

Theory suggests danger of over-fitting when T is very large.

Combining classifiers

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

The function class used by AdaBoost is

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

i.e., "linear combinations of T functions from $\mathcal{F}$".
Complexity of $\mathcal{F}_T$ grows *linearly with* T.

this part get bigger

Theoretical guarantee: with high probability over choice of i.i.d. sample S,

$$\operatorname{err}(f) \ \leq \ \operatorname{err}(f, S) + O\left(\sqrt{\frac{\boxed{T} \log |\mathcal{F}_{|S}|}{|S|}} \right) \quad \forall f \in \mathcal{F}_T.$$

can lead to overfitting

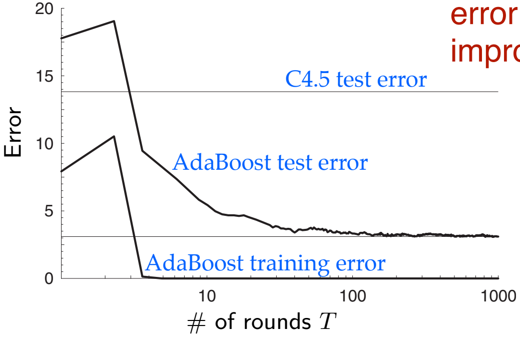
Theory suggests danger of over-fitting when T is very large.
Indeed, this does happen sometimes . . . **but often not**!

As the T increases, the training error goes down!

A typical run of boosting

AdaBoost+C4.5 on "letters" dataset.

even when training error is zero, keeps improving



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

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

(Figure 1.7 from Schapire & Freund text)

Boosting the margin

Final classifier from AdaBoost:

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

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

Boosting the margin

Final classifier from AdaBoost:

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

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 generalization error**, independent of T.
- AdaBoost tends to increase margins on training examples.

(Similar but not the same as SVM margins.)

Boosting the margin

Final classifier from AdaBoost:

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

the normlizer has no effect over sign!

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 generalization error**, independent of T.
- AdaBoost tends to increase margins on training examples.

actually increase large margin a different kind of margin

(Similar but not the same as SVM margins.)

On "letters" dataset:

	$T = 5$	$T = 100$	$T = 1000$
training error	0.0%	0.0%	0.0%
test error	8.4%	3.3%	3.1%
% margins ≤ 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":

$$x \mapsto \boldsymbol{\phi}(x) := (f(x) : f \in \mathcal{F}) \in \{\pm 1\}^{\mathcal{F}}$$

(possibly infinite dimensional!).

|F| is the size of feature functions' colletion

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

$$x \mapsto \boldsymbol{\phi}(x) := \boxed{(f(x) : f \in \mathcal{F}) \in \{\pm 1\}^{\mathcal{F}}}$$

(possibly infinite dimensional!).

AdaBoost's final classifier is a *linear classifier* in $\boxed{\{\pm 1\}^{\mathcal{F}}:}$

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

where

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

it's interesting!

Exponential loss

AdaBoost is a particular "coordinate descent" algorithm for

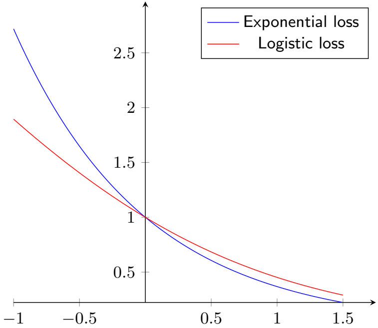
$$\min_{\boldsymbol{w} \in \mathbb{R}^{\mathcal{F}}} \qquad \frac{1}{|S|} \sum_{(\boldsymbol{x}, y) \in S} \ell_{\exp}(y \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle)$$

actually an algorithm for solving
loss_exp target!!!

AdaBoost is a particular "coordinate descent" algorithm for

great!!!!

$$\min_{\boldsymbol{w}\in\mathbb{R}^{\mathcal{F}}} \quad \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S} \ell_{\exp}(y\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x})\rangle)$$



Exponential loss:

$$\ell_{\exp}(z) = \exp(-z).$$

$\mathbb{E}[\ell_{\exp}(Y \cdot g(x))|X = x]$ is minimized by

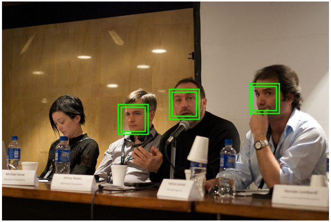
$$g(x) = \frac{1}{2}\ln\left(\frac{\eta(x)}{1-\eta(x)}\right)$$

where $\eta(x) = \Pr[Y = +1|X = x]$.

Application: face detection

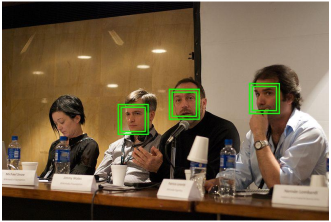
Face detection

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



Face detection

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

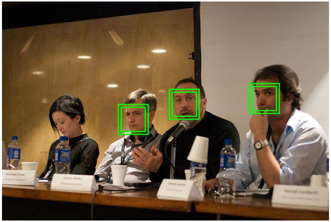


As a classification problem:

- Divide up images into patches (at varying scales, e.g., 24×24, 48×48).
- Classify each patch as "face" or "not face".

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., 24×24, 48×48).
- Classify each patch as "face" or "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.

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**.

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 $\boldsymbol{x} \in [0, 1]^{d^2}$.

Face detectors via AdaBoost [Viola & Jones, 2001]

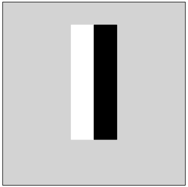
Face detector architecture by Viola & Jones (2001): major achievement in computer vision; **detector actually usable in real-time**.

- 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 $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$, where $\boldsymbol{w}$ has a very particular form:

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 $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$, where $\boldsymbol{w}$ has a very particular form:



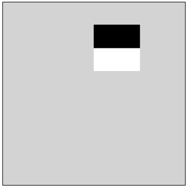
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle =$ average pixel value in black box

$-$ average pixel value in white box

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 $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$, where $\boldsymbol{w}$ has a very particular form:



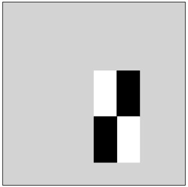
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle =$ average pixel value in black box

$-$ average pixel value in white box

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 $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$, where $\boldsymbol{w}$ has a very particular form:

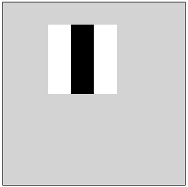


$\langle \boldsymbol{w}, \boldsymbol{x} \rangle =$ average pixel value in black box
$-$ average pixel value in white box

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

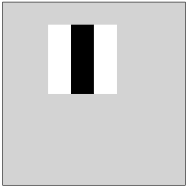


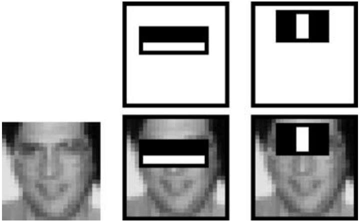
$$\boxed{\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{x} \rangle = &\ \text{average pixel value in black box} \\ &- \text{average pixel value in white box} \end{aligned}}$$

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 $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$, where $\boldsymbol{w}$ has a very particular form:

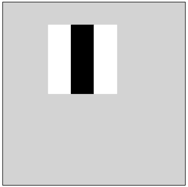




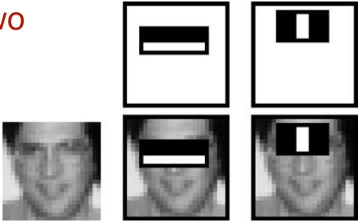
Face detectors via AdaBoost [Viola & Jones, 2001]

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 $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$, where $\boldsymbol{w}$ has a very particular form:



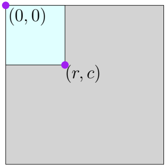
note the two different patches



- **Many possible "rules-of-thumb" of this form**.
  AdaBoost combines several of them to build an accurate classifier.

Viola & Jones "integral image" trick

"Integral image" trick:



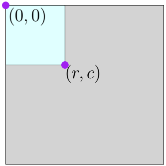
For every image, pre-compute

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

(single pass through image).

Viola & Jones "integral image" trick

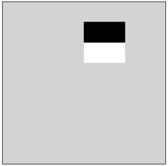
"Integral image" trick:



For every image, pre-compute

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

(single pass through image).



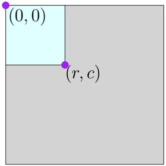
To compute inner product

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

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

Viola & Jones "integral image" trick

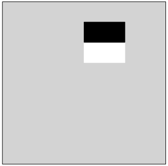
"Integral image" trick:



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**.

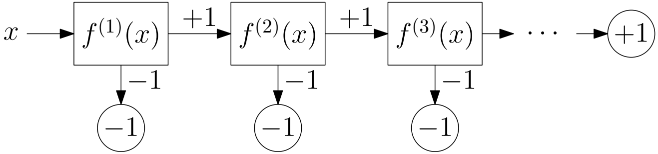
Viola & Jones cascade architecture

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

Viola & Jones cascade architecture

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

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

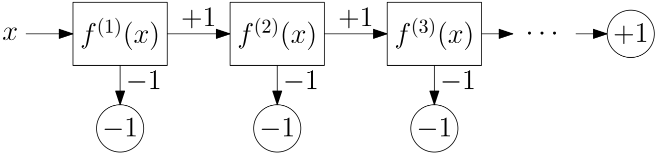


- Each $f^{(\ell)}$ is trained (using AdaBoost), adjust threshold (before passing through sign) to minimize false negative rate.
- Can make $f^{(\ell)}$ in later stages more complex than in earlier stages, since most examples don't make it to the end.

Viola & Jones cascade architecture

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

Solution: Train several classifiers (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 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

More on boosting

Many variants of boosting:

- AdaBoost.L and LogitBoost (replaces ℓ_{exp} with ℓ_{log}).
- Forward-{step,stage}wise regression (replaces ℓ_{exp} with ℓ_{sq}).
- Boosted decision trees = boosting + decision trees, often with ℓ_{sq}.

(See ESL Chapter 10.)

- Boosting algorithms for *ranking* and *multi-class*.
- Boosting algorithms that are robust to certain kinds of noise.
- ...

Many connections between boosting and other subjects:

- Game theory, online learning
- Information geometry
- Computational complexity
- ...