(1) Note Title 5/18/2008

AdaBoost.

AdaBoost is a method for combining a number of weak classifiers to make a strong classifier.

<u>Input</u>: set of weak classifiers $\{\varphi_\mu(x) : \mu = 1 \text{ to } M\}$

labelled data $\{(\underline{x}^t, y^t) : t = 1 \text{ to } N\}$

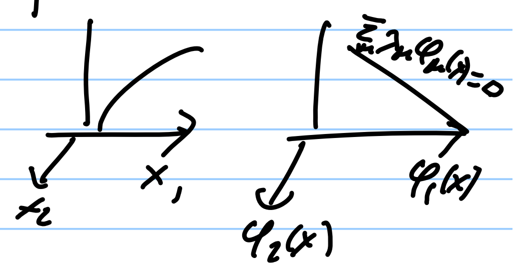
$y^t \in \{-1, 1\}$

<u>Output</u>: strong classifier

$$\mathrm{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x)\right)$$

$\{\lambda_\mu\}$ weights/coefficients.

. The strong classifier is a plane in feature space $\{\varphi_\mu(x)\}$.



In practice, most of the $\lambda_\mu = 0$.

The "selected" weak classifiers are those with $\lambda_\mu \neq 0$.

(2) The task of AdaBoost is to select weights $\{\lambda_\mu\}$ to make the strong classifier as effective as possible.

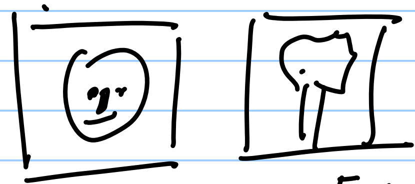
The motivation is that it is often possible to obtain weak classifier for classification tasks – i.e. a weak classifier that is effective 60% of the time. Want to build a strong classifier – effective 99% of the time – that is built by combining weak classifiers.

The combination is by weighted summation $\sum_\mu \lambda_\mu \varphi_\mu(x)$

Why linear weighted combination?

Because we can use an efficient algorithm – AdaBoost – to estimate them.

(3) Example: Face Detection (Viola & Jones).

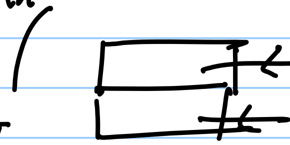


Face   Non-Face.

Training examples are a set of images x^t labelled by $y^t = 1$ if image contains a face, $y^t = -1$ if not.

Weak classifier:

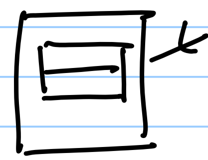
threshold



$\bar{I}_1$, $\bar{I}_2$ — average intensity within window

Face: if $\bar{I}_1(x) - \bar{I}_2(x) > T$

Intensity in forehead is bigger than intensity in eye region.



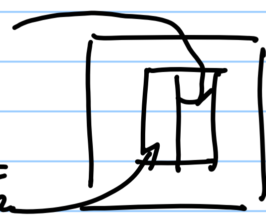
window position

Forehead Detector.

$\bar{I}_1$

Face: if $\bar{I}_1 - \bar{I}_2 < \epsilon_1$

$\bar{I}_2 - \bar{I}_1 < \epsilon_2$



$\bar{I}_2$

Symmetry Detection

→ Faces symmetric.

where ϵ_1 & ϵ_2 are small constants.

Easy to get weak classifiers of this type — weak classifier is features $\bar{I}_1(x), \bar{I}_2(x)$ + threshold T — but each weak classifier is only partially success. AdaBoost gives a way to select weak classifiers and combine them to make a strong classifier.

(Algorithm later)

(4) **AdaBoost: Mathematical Description.**

Define $Z[\lambda_1, \dots \lambda_M] = \sum_{t=1}^{N} e^{-y^t \sum_{\mu=1} \lambda_\mu \varphi_\mu(x^t)}$.

This is an upper bound of the error rate of the strong classifier $S(x) = \text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x)\right)$.

Error Rate: $E[\{\lambda_i\}] = \sum_{t=1}^{N} \{1 - I(S(x^t), y^t)\}$

where $I(S(x^t), y^t) = 1$, if $S(x^t) = y^t$ (correct answer)
$= 0$, otherwise.

Claim: $E[\lambda_1 \dots \lambda_M] \leq Z[\lambda_1 \dots \lambda_M]$.

compare each term in the summation $\sum_{t=1}^{N}$

Case (i) If $S(x^t) = y^t$, then $\text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(\vec{x})\right) = \text{sign}\, y^t$

so $y^t \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^t) = A > 0$ (Defining A)

Error Term $\{1 - I(S(x^t), y^t)\} = 0$

Z Term $e^{-y^t \sum_\mu \lambda_\mu \varphi_\mu(x^t)} = e^{-A} > 0$ ✓

Case (ii) If $S(x^t) \neq y^t$, then $y^t \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^t) = -B < 0$ $(B > 0)$

Error Term $\{1 - I(S(x^t), y^t)\} = 1$

Z term $e^{-y^t (\sum_\mu \lambda_\mu \varphi_\mu(x^t))} = e^{B} > 1$ ✓

So Z term is bigger than error term in both cases.

Goal: AdaBoost minimizes

(5) $Z[\lambda_1 \ldots \lambda_N]$. This will guarantee that the error rate is small (but not necessarily the minimum error rate).

Strategy to minimize $Z[\lambda_1 \ldots \lambda_N]$.

Initialize by $\lambda_1 = \ldots = \lambda_N = 0$.
(i.e. $H_0(x) = 0 \Rightarrow$ no weak classifier selected).

Minimize $Z[\lambda_1 \ldots \lambda_N]$ by coordinate descent.

At time step ℓ.
state $\lambda_1^\ell, \ldots \lambda_N^\ell$.

For each $i \rightarrow$ minimize Z w.r.t. λ_i
with λ_j^ℓ fixed for $j \neq i$.

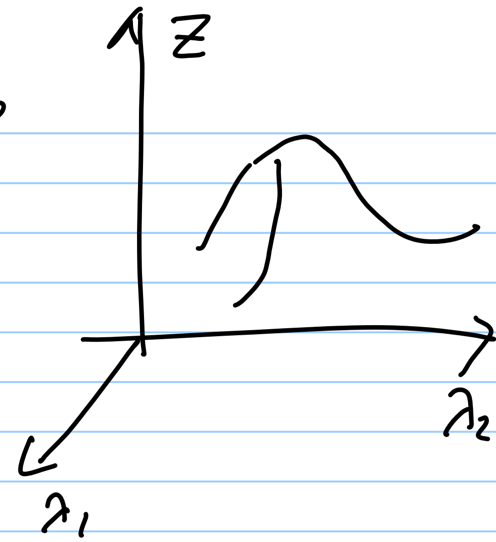
Solve $\frac{\partial Z}{\partial \lambda_i} = 0$ to solve for $\hat{\lambda}_i$ for each i.

Compute $Z[\lambda_1^\ell, \ldots, \lambda_{i-1}^\ell, \hat{\lambda}_i, \lambda_{i+1}^\ell, \ldots \lambda^\ell]$ for each.

select $\hat{i} = \arg\min_i Z[\lambda_1^\ell, \ldots, \lambda_{i-1}^\ell, \hat{\lambda}_i, \lambda_{i+1}^\ell, \ldots \lambda_N^\ell]$

set $\lambda_j^{\ell+1} = \lambda_j^\ell, \; j \neq \hat{i}, \quad \lambda_{\hat{i}}^{\ell+1} = \hat{\lambda}_{\hat{i}}$.

(6) <u>Intuition</u>: at each time step l.



Calculate how much you can decrease Z by changing only one of the $\{\lambda_i\}$.

select the $\hat{\lambda_i}$ which maximally decreases Z,

Each step of this algorithm is guaranteed to decrease Z.

So algorithm will converge to a minimum of Z. (But Z is convex in λ, so the algorithm converges to global minimum).

Why Z? Why this algorithm?

<u>Practical</u> — we can compute $\frac{\partial Z}{\partial \lambda_i}$ and the minimum of Z easily.

(7) <u>AdaBoost Algorithm.</u>

<u>Data</u> $\{(x^t, y^t) : t = 1, .., N\}$

set of weak classifiers. $\{\phi_\mu(x), \mu = 1, .. M\}$.

For each weak classifier, divide the training data into two classes

$$W_\mu^+ = \{t : y^t \phi_\mu(x^t) = 1\} \quad \phi_\mu \text{ correct}$$
$$W_\mu^- = \{t : y^t \phi_\mu(x^t) = -1\} \quad \phi_\mu \text{ wrong.}$$
$$W_\mu^+ \cup W_\mu^- = \{1 \dots N\}$$

At each time step ℓ, define a set of "weights" for the training examples.

$$D_t^\ell = \frac{e^{-y^t \sum_{\mu=1}^{M} \lambda_\mu^\ell \phi_\mu(x^t)}}{\sum_t e^{-y^t \sum_{\mu=1}^{M} \lambda_\mu^\ell \phi_\mu(x^t)}} \qquad \sum_t D_t^\ell = 1, \quad D_t^\ell \geq 0.$$

Gives bigger weights to data incorrectly classified by current "strong classifier".

i.e. D_t^ℓ is large if $y^t \sum_{\mu=1}^{M} \lambda_\mu^\ell \phi_\mu(x^t) < 0$

implies $\operatorname{sign}(\sum_{\mu=1}^{M} \lambda_\mu^\ell \phi_\mu(x^t)) \neq y^t$.

D_t^ℓ is small if $y^t \sum_{\mu=1}^{M} \lambda_\mu^\ell \phi_\mu(x^t) > 0$.

implies $\operatorname{sign}(\sum_{\mu=1}^{M} \lambda_\mu^\ell \phi_\mu(x^t)) = y^t$

(8)

Adaboost Algorithm.

Initialize $\lambda_1 = \lambda_2 = \ldots = \lambda_N = 0$

At time step $\lambda_1^\ell, \lambda_2^\ell \ldots \lambda_N^\ell$

For each i, calculate $\Delta_i^\ell = \frac{1}{2} \log \left(\frac{\sum_{t \in W_i^+} D_t^\ell}{\sum_{t \in W_i^-} D_t^\ell} \right)$

(change in Δ_i^ℓ due to solving $\frac{\partial Z}{\partial \lambda_i} = 0$, see later)

calculate $\sqrt{\sum_{t \in W_i^+} D_t^\ell} \sqrt{\sum_{t \in W_i^-} D_t^\ell}$

(change in Z due to setting λ_i to $\hat{\lambda}_i$, see later)

select $\hat{i} = \underset{i}{\mathrm{ARG\,MIN}} \sqrt{\sum_{t \in W_i^+} D_t^\ell} \sqrt{\sum_{t \in W_i^-} D_t^\ell}$

set $\lambda_j^{\ell+1} = \lambda_j^\ell \quad , j \neq \hat{i}$

$\lambda_{\hat{i}}^{\ell+1} = \lambda_{\hat{i}}^\ell + \Delta_{\hat{i}}^\ell$.

repeat, until convergence.

(9) Intuition for the $\sum_{t \in w_i^+} D_t^l$ and $\sum_{t \in w_i^-} D_t^l$ terms.

Initially, $D_t^{l=0} = \frac{1}{N}$ the data is equally weighted.

$\sum_{t \in w_i^+} D_t^{l=0}$ is the proportion of data that is correctly classified by $\varphi_i(x)$

$\sum_{t \in w_i^-} D_t^{l=0}$ is proportion incorrectly classified

. i.e. $\sum_{t \in w_i^-} D_t^{l=0}$ is the normalized error rate if we just use classifier $\varphi_i(x)$ – normalized by $1/N$ //

For $l \neq 0$, $\sum_{t \in w_i^+} D_t^l$ is data correctly classified by $\varphi_i(x)$ taking into account the previously selected classifiers (those for which $\lambda_j^l \neq 0$).

$\varphi_i(x)$ is a useless classifier if $\sum_{t \in w_i^+} D_t^l = \sum_{t \in w_i^-} D_t^l$
i.e. weighted error = $1/2$.

corresponds to $\Delta_i^l = 0$ (i.e. no change in weight λ_i)

(10) Term $\sqrt{\sum_{t\in w_i^+} D_t^\ell}\sqrt{\sum_{t\in w_i^-} D_t^\ell}$ is a non-linear function of the weighted error rate of $\varphi_i(x)$.

It can be rewritten as

$$\sqrt{\left(\sum_{t\in w_i^-} D_t^\ell\right)\left(1-\sum_{t\in w_i^-} D_t^\ell\right)}$$

because $\sum_{t\in w_i^+} D_t^\ell + \sum_{t\in w_i^-} D_t^\ell = 1$

It's smallest values are if

$\sum_{t\in w_i^-} D_t^\ell = 0$, $\varphi_i(x)$ has optimal weighted classifier

$\sum_{t\in w_i^-} D_t^\ell = 1$, $\varphi_i(x)$ worst possible classifier $\rightarrow$ so implies $-\varphi_i(x)$ best possible classifier

its largest values are when

$\sum_{t\in w_i^-} D_t^\ell = \frac{1}{2}$, i.e. when weighted error is $\frac{1}{2}$, $\varphi_i(x)$ useless

(11)

When does AdaBoost converge?

It stops when all weak classifiers are useless – ie when $\sum_{t \in W_i^-} D_t^\ell = 1/2$, for all i.

In this case $\sqrt{\sum_{t \in W_i^+} D_t^\ell} \sqrt{\sum_{t \in W_i^-} D_t^\ell}$ takes its biggest (i.e. worst) value of $1/2$, for all i.

The weight update $\Delta_i^\ell = \frac{1}{2} \log \left\{ \frac{\sum_{t \in W_i^+} D_t^\ell}{\sum_{t \in W_i^-} D_t^\ell} \right\}$ is 0 for all $\varphi_i(x)$ (since $\log 1 = 0$).

In general, at time step ℓ select the classifier $\varphi_{\hat{i}}(x)$ with smallest "weighted error rate" $\sqrt{\sum_{t \in W_{\hat{i}}^+} D_t^\ell} \sqrt{\sum_{t \in W_{\hat{i}}^-} D_t^\ell}$

$$\Delta_{\hat{i}}^\ell = \frac{1}{2} \log \left\{ \frac{\sum_{t \in W_{\hat{i}}^+} D_t^\ell}{\sum_{t \in W_{\hat{i}}^-} D_t^\ell} \right\}$$

Update $\lambda_{\hat{i}}^\ell \to \lambda_{\hat{i}}^\ell + \Delta_{\hat{i}}^\ell$

the smaller the weighted error rate – ie smaller $\sum_{t \in W_{\hat{i}}^-} D_t^\ell$ or $\sum_{t \in W_{\hat{i}}^+} D_t^\ell$ – then the bigger the change $\Delta_{\hat{i}}^\ell$.

(12) How does AdaBoost algorithm relate to AdaBoost mathematics?

AdaBoost mathematics requires:

(i) efficient solution of $\frac{\partial Z}{\partial \lambda_i} = 0$.

(ii) efficient computation of Z.

(i) $\frac{\partial Z}{\partial \lambda_i} = 0$

$$\frac{\partial Z}{\partial \lambda_i} = \sum_{t=1}^{N} \{-y^t \varphi_i(x^t)\} e^{-y^t \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^t)}.$$

set. $\lambda_i \Rightarrow \lambda_i + \Delta_i$ solve for Δ_i.

$$\frac{\partial Z}{\partial \lambda_i} = 0 \Rightarrow \sum_{t=1}^{N} \{y^t \varphi_i(x^t)\} e^{-y^t \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^t)} e^{-y^t \Delta_i \varphi_i(x^t)} = 0$$

$$\sum_{t=1}^{N} \{y^t \varphi_i(x^t)\} D_t e^{-y^t \Delta_i \varphi_i(x^t)} = 0.$$

Recall $D_t = \frac{e^{-y^t \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^t)}}{\sum_t e^{-y^t \sum_\mu \lambda_\mu \varphi_\mu(x^t)}}$

Divide $\sum_{t=1}^{N} = \sum_{t \in W_i^+} + \sum_{t \in W_i^-}$

$$\sum_{t \in W_i^+} D_t e^{-\Delta_i} - \sum_{t \in W_i^-} D_t e^{\Delta_i} = 0.$$

$$e^{2\Delta_i} = \left(\sum_{t \in W_i^+} D_t\right) / \left(\sum_{t \in W_i^-} D_t\right).$$

$$\Delta_i = \frac{1}{2} \log\left(\sum_{t \in W_i^+} D_t / \sum_{t \in W_i^-} D_t \right). //$$

(13) (2) Computation of Z.

$$Z[\lambda_1, \ldots, \lambda_i, \lambda_i+\Delta_i, \ldots \lambda_N]$$
$$= \sum_{t=1}^{N} e^{-y^t\{\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^t) + \Delta_i \varphi_i(x^t)\}}$$
$$= K \sum_{t=1}^{N} D_t e^{-y^t \Delta_i \varphi_i(x^t)}$$
where $K = \sum_{t=1}^{N} D_t e^{-y^t \Delta_i \varphi_i(x^t)}$
is independent of i.

$$Z[\lambda_1 \ldots, \lambda_i, \lambda_i+\Delta_i, \ldots \lambda_N]$$
$$= K \left\{ \sum_{t\in W_i^+} D_t e^{-\Delta_i} + \sum_{t \in W_i^-} D_t e^{\Delta_i} \right\}$$
$$= 2K \sqrt{\sum_{t\in W_i^+} D_t} \sqrt{\sum_{t\in W_i^-} D_t}$$
using $e^{\Delta i}$ from previous page.

Hence, coordinate descent reduces to computing $\Delta_i = \frac{1}{2} \log \left(\frac{\sum_{t\in W_i^+} D_t}{\sum_{t\in W_i^-} D_t} \right)$ → how much to change λ_i

solving $\hat{i} = \arg\min_i \left\{ \sum_{t\in W_i^+} D_t e^{-\Delta_i} + \sum_{t\in W_i^-} D_t e^{\Delta_i} \right\}$
to find best λ_i to change.

(14)

Error to Avoid in AdaBoost.

Once a weak classifier $\varphi_i(x)$ has been selected, it can be selected again.
This should be obvious from the coordinate descent formulation — if you decide to update λ_i at time step l, then you can also update λ_i at a later time step.

Probabilistic Interpretation.

It can be shown (Friedman, Hastie, Tibshirani) that AdaBoost relates to logistic regression.

$$P(y|x) = \frac{e^{y\sum_\mu \lambda_\mu \varphi_\mu(x)}}{e^{\sum_\mu \lambda_\mu \varphi_\mu(x)} + e^{-\sum_\mu \lambda_\mu \varphi_\mu(x)}}$$

This result is asymptotic — only true in the limit as the number of samples N becomes infinitely large.

Note: standard sigmoid regression means that you specify a small number of features $\varphi_\mu(x)$ that are not necessarily binary-valued.

(15) The main advantage of AdaBoost is that you can specify a large set of weak classifiers and the algorithm decides which weak classifiers to use — by assigning them non-zero λ_i.

Standard logistic regression only uses a small set of features (like weak classifiers).

SVM uses the kernel trick $K(\underline{x}, \underline{x}') = \underline{\varphi}(\underline{x}) \cdot \underline{\varphi}(\underline{x}')$ to simplify the dependence on $\underline{\varphi}(\underline{x})$, but doesn't say how to select $K(\cdot,\cdot)$ or $\underline{\varphi}(\cdot)$.

Multilayer perceptron can be interpreted as selecting weak classifiers → but in a non-optimal manner.

Recent work, suggests that AdaBoost can be improved by making it more similar to logistic regression.

AdaBoost can be extended to multiclasses.

(8) Second Reason:

Adaboost converges to the posteriors.

$$p(\omega=1|x) = \frac{e^{\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}}{e^{\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)} + e^{-\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}}$$

$$p(\omega=-1|x) = \frac{e^{-\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}}{e^{\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)} + e^{-\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}}$$

$$\frac{\partial Z}{\partial \lambda_\nu} = \sum_{i=1}^{N} \omega_i \varphi_\nu(x_i) e^{-\omega_i \sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x_i)} = 0$$

hence, $\sum_{i=1}^{N} \delta_{\omega_i,1} \varphi_\nu(x_i) e^{-\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x_i)} = \sum_{i=1}^{N} \delta_{\omega_i,-1} \varphi_\nu(x_i) e^{+\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x_i)}$ $\forall \nu$.

$$\frac{1}{N}\sum_{i=1}^{N} \delta_{\omega_i,1}\delta_{x_i,x} \to p(\omega=1,x)$$

$$\frac{1}{N}\sum_{i=1}^{N} \delta_{\omega_i,-1}\delta_{x_i,x} \to p(\omega=-1|x).$$

$$\frac{1}{N}\sum_{i=1}^{N} \delta_{\omega_i,1} f(x_i) = \sum_x \left\{\frac{1}{N}\sum_{i=1}^{N}\delta_{\omega_i,1}\delta_{x_i,x} f(x)\right\},$$

$$\frac{1}{N}\sum_{i=1}^{N} \delta_{\omega_i,1} f(x_i) \to \sum_x p(\omega=1,x) f(x) \quad \forall \nu.$$

So,

$$\sum_x p(\omega=1,x)\varphi_\nu(x) e^{\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)} = \sum_x p(\omega=-1,x)\varphi_\nu(x) e^{-\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}$$

$$p(\omega=1|x) = \frac{e^{+\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}}{Z}, \quad p(\omega=-1|x) = \frac{e^{-\sum_{\mu=1}^{M}\lambda_\mu \varphi_\mu(x)}}{Z}$$