(1) # AdaBoost. Spring 2014

Note Title 5/18/2008

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

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

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

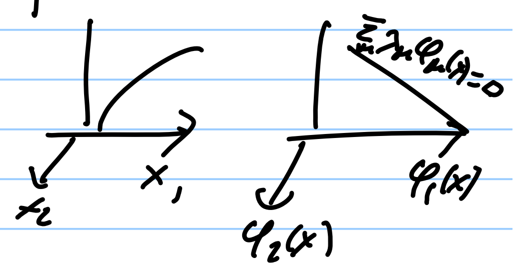
$y^i \in \{-1, 1\}$

<u>Output</u>: strong classifier

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

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

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



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.

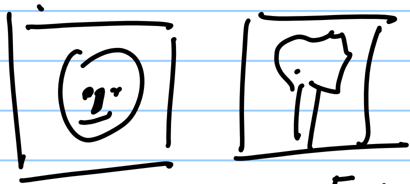
The combination is by weighted summation $\sum_\mu \lambda_\mu \phi_\mu(x)$

Why linear weighted combination?

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

Spring 2013

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

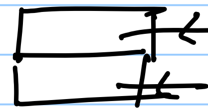


Face   Non Face.

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

threshold

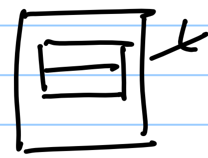
Weak classifier:



average intensity within window

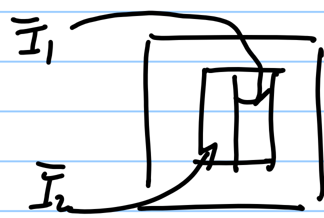
Face: if $\bar{I}_1(x) - \bar{I}_2(x) > T$

Intensity in forehead is bigger than intensity in eye region.



window position

Forehead Detector.



Symmetry Detection
→ Faces symmetric.

Face: if $\bar{I}_1 - \bar{I}_2 < \epsilon_1$
$\bar{I}_2 - \bar{I}_1 < \epsilon_2$

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 Descript.

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

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

Error Rate: $E[\{\lambda_i\}] = \sum_{i=1}^{N} \{1 - I(S(x_i), y_i)\}$ ← Identity Function ← Empirical Risk.

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

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

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

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

so $y^i \sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(x^i) = A > 0$ (Defines A)

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

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

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

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

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

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 $\lambda_1^l, \ldots \lambda_N^l$.

For each $i \rightarrow$ minimize Z w.r.t. λ_μ
with λ_ν^l fixed for $\nu \neq \mu$

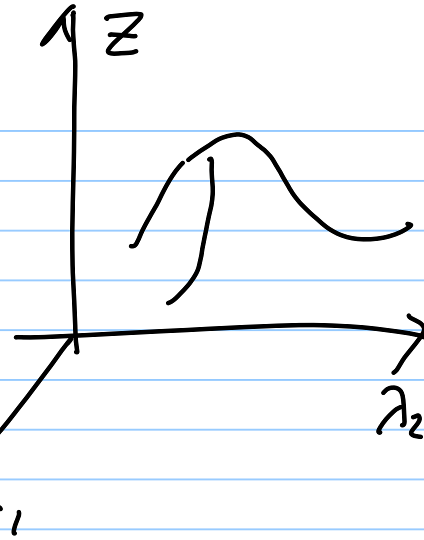
solve $\frac{\partial Z}{\partial \lambda_\mu} = 0$ to solve for $\hat{\lambda}_\mu$ for each μ

Compute $Z[\lambda_1^l, \ldots, \lambda_{\mu-1}^l, \hat{\lambda}_\mu, \lambda_{\mu+1}^l, \ldots \lambda^l]$ for each μ

select $\hat{\mu} = \arg\min_\mu Z[\lambda_1^l, \ldots, \lambda_{\mu-1}^l, \hat{\lambda}_\mu, \lambda_{\mu+1}^l, \ldots \lambda_M^l]$

set $\lambda_\nu^{l+1} = \lambda_\nu^l, \ \nu \neq \hat{\mu}, \ \lambda_{\hat{\mu}}^{l+1} = \hat{\lambda}_{\hat{\mu}}$.

(6) Intuition: at each time step l.



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

select the $\lambda_{\hat{\mu}}$ 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?

Practical — we can compute $\frac{\partial Z}{\partial \lambda_\mu}$ and the minimum of Z easily.

Why not steepest descent? We want to keep most of the $\lambda_\mu = 0$. use as few non-zero λ's as possible

(7) # AdaBoost Algorithm.

Data $\{(x^i, y^i) : i = 1, .., N\}$

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

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

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

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

$$D_i^t = \frac{e^{-y^i \sum_{\mu=1}^{M} \lambda_\mu^t \phi_\mu(x^i)}}{\sum_{i=1}^{N} e^{-y^i \sum_{\mu=1}^{M} \lambda_\mu^t \phi_\mu(x^i)}} \qquad \sum_i D_i^t = 1 \qquad D_i^t \geq 0.$$

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

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

implies $\text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu^t \phi_\mu(x^i)\right) \neq y^i$.

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

implies $\text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu^t \phi_\mu(x^i)\right) = y^i$

(8)

Adaboost Algorithm.

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

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

For each μ, calculate $\Delta_\mu^t = \frac{1}{2} \log \left\{ \frac{\sum_{i \in W_\mu^+} D_i^t}{\sum_{i \in W_\mu^-} D_i^t} \right\}$

(change in Δ_μ^t due to solving $\frac{\partial Z}{\partial \lambda_\mu} = 0$, see later)

calculate $\sqrt{\sum_{i \in W_\mu^+} D_i^t} \sqrt{\sum_{i \in W_\mu^-} D_i^t}$

(change in Z due to setting λ_μ to $\hat{\lambda}_\mu$, see later)

<u>select</u> $\hat{\mu} = \arg\min_\mu \sqrt{\sum_{i \in W_\mu^+} D_i^t} \sqrt{\sum_{i \in W_\mu^-} D_i^t}$

set $\lambda_\nu^{t+1} = \lambda_\nu^t \quad , \nu \neq \hat{\mu}$

$\lambda_{\hat{\mu}}^{t+1} = \lambda_{\hat{\mu}}^t + \Delta_{\hat{\mu}}^t$.

repeat, until convergence.

(9) Intuition for the $\sum_{i\in w_\mu^+} D_i^t$ and $\sum_{i\in w_\mu^-} D_i^t$ terms.

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

$\sum_{i\in w_\mu^+} D_i^{t=0}$ is the proportion of data that is correctly classified by $\varphi_\mu(.)$

$\sum_{i\in w_\mu^-} D_i^{t=0}$ is proportion incorrectly classified

i.e. $\sum_{i\in w_\mu^-} D_i^{t=0}$ is the normalized error rate if we just use classifier $\varphi_\mu(.)$ – normalized by $1/N$ //

For $t\neq 0$, $\sum_{i\in w_\mu^+} D_i^t$ is data correctly classified by $\varphi_\mu(x)$ taking into account the previously selected classifiers (those for which $\lambda_\mu^t \neq 0$).

$\varphi_\mu(.)$ is a useless classifier if $\sum_{i\in w_\mu^+} D_i^t = \sum_{i\in w_\mu^-} D_i^t$
i.e. weighted error = ½.
corresponds to $\Delta_\mu^t = 0$ (i.e. no change in weight λ_μ)

(10) <u>Term</u> $\sqrt{\sum_{i\in\omega_\mu^+} D_i^t}\sqrt{\sum_{i\in\omega_\nu^-} D_i^t}$ is a non-linear function of the weighted error rate of $\varphi_\mu(.)$.

It can be rewritten as

$$\sqrt{\left(\sum_{i\in\omega_\mu^-} D_i^t\right)\left(1-\sum_{i\in\omega_\mu^+} D_i^t\right)}$$

because $\sum_{i\in\omega_\mu^+} D_i^t+\sum_{i\in\omega_\mu^-} D_i^t=1$

It's smallest values are if

$\sum_{i\in\omega_\mu^-} D_i^t=0$, φ_μ has optimal weighted classifier

$\sum_{i\in\omega_\mu^-} D_i^t=1$, φ_μ worst classifier implies $-\varphi_\mu$ best classifier

its largest values are when

$\sum_{i\in\omega_\mu^-} D_i^t=\frac{1}{2}$, i.e. when weighted error is $\frac{1}{2}$, i.e. $\varphi_\mu(\,)$ useless

(11)

When does AdaBoost converge?

It stops when all weak classifiers are useless – ie when $\sum_{i \in W_\mu^-} D_i^t = 1/2$, for all μ

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

The weight update $\Delta_\mu^t = \frac{1}{2} \log \left\{ \frac{\sum_{i \in W_\mu^+} D_i^t}{\sum_{i \in W_\mu^-} D_i^t} \right\}$ is 0 for all $\phi_\mu(.)$ (since $\log 1 = 0$).

In general, at time step t select the classifier $\phi_{\hat{\mu}}(.)$ with smallest "weighted error rate" $\sqrt{\sum_{i \in W_\mu^+} D_i^t} \sqrt{\sum_{i \in W_\mu^-} D_i^t}$ $\quad \Delta_{\hat{\mu}}^t = \frac{1}{2} \log \left\{ \frac{\sum_{i \in W_\mu^+} D_i^t}{\sum_{i \in W_\mu^-} D_i^t} \right\}$

Update $\lambda_{\hat{\mu}}^t \to \lambda_{\hat{\mu}}^t + \Delta_{\hat{\mu}}$

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

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

AdaBoost mathematics requires:

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

(ii) efficient computation of Z.

(1) $\frac{\partial Z}{\partial \lambda_\mu} = 0$

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

set. $\lambda_\mu \Rightarrow \lambda_\mu + \Delta_\mu$ solve for Δ_μ

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

$$\sum_{i=1}^{N} \{y^i \varphi_\mu(x^i)\} D_i \, e^{-y^i \Delta_\mu \varphi_\mu(x^i)} = 0.$$

Recall $D_i = \frac{e^{-y^i \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x^i)}}{\sum_i e^{-y^i \sum_\mu \lambda_\mu \varphi_\mu(x^i)}}$

Divide $\sum_{i=1}^{N} = \sum_{i \in W_\mu^+} + \sum_{i \in W_\mu^-}$

$$\sum_{i \in W^+} D_i e^{-\Delta_\mu} - \sum_{i \in W^-} D_i e^{\Delta_\mu} = 0.$$

$$e^{2\Delta_\mu} = \left(\sum_{i \in W_\mu^+} D_i\right) / \left(\sum_{i \in W_\mu^-} D_i\right).$$

$$\Delta_\mu = \frac{1}{2} \log\left(\sum_{i \in W_\mu^+} D_i / \sum_{i \in W_\mu^-} D_i\right). //$$

(13) (2) Computation of Z.

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

$$Z[\lambda_1 \ldots \lambda_\mu + \Delta_\mu, \ldots \lambda_N]$$
$$= K \left\{ \sum_{i \in W_\mu^+} D_i e^{-\Delta_\mu} + \sum_{i \in W_\mu^-} D_i e^{\Delta_\mu} \right\}$$
$$= 2K \sqrt{\sum_{i \in W_\mu^+} D_i} \sqrt{\sum_{i \in W_\mu^-} D_i}$$
using e^{Δ_μ} from previous page.

Hence, coordinate descent reduces to computing $\Delta_\mu = \frac{1}{2} \log \left(\frac{\sum_{i \in W_\mu^+} D_i}{\sum_{i \in W_\mu^-} D_i} \right)$ → how much to change λ_μ

Solving $\hat{\mu} = \arg\min_\mu \sum_{i \in W_\mu^+} D_i e^{-\Delta_\mu} + \sum_{i \in W_\mu^-} D_i e^{\Delta_\mu}$ to find best λ_μ to change.

Error to Avoid in AdaBoost.

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

Probabilistic Interpretation.

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

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

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 $\phi_\mu(x)$ 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_μ.

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

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

Multilayer perception 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.