SL : BOOSTING

Boosting is a powerful technique, which can produce very accurate learners. Boosting combines the prediction of many (weak) learners to produce a powerful 'committee' / aka a strong learner.

→ A weak learner f does slightly better than a random guess. The formal definition of a weak learner was introduced by Vaillant (1984) in the context of PAC learning (Probably Approximately Correct).

→ A strong learner f does arbitrarily well.

For us, it is enough to understand that boosting converts a set of weak learners into a strong one by taking a weighted combination of the weak learner's decisions. Formally, consider

$$\mathcal{F} = \Big\{ \underbrace{\sum_{j=1}^{M} \beta_j f_j(\cdot)}_{\substack{\text{strong learner} \\ \text{"good prediction accuracy"}}} \;\Big|\; \|\beta\|_1 \leq 1, \quad \underbrace{f_j}_{\substack{\text{weak learner} \\ \text{"prediction accuracy close to a random guess"}}} : X \to [-1,1] = \text{soft classifiers} \Big\}$$

I. RISK BOUND FOR BOOSTING

In this section, we derive an oracle inequality for boosting. In the context of convex relaxation for binary classification, the risk of an element $f \in \mathcal{F}$ is $R_\varphi(f) = \mathbb{E}\,\varphi(-Y f(X))$, where $Y \in \{-1, 1\}$, and φ = convex surrogate.

The empirical φ-risk, based on a random sample $\mathcal{L}_n$, is

$$\hat{R}_{n,\varphi} = \frac{1}{n} \sum_{i=1}^{n} \varphi(-Y_i f(X_i)).$$

Put
- $\hat{f}_n \in \underset{f \in \mathcal{F}}{\operatorname{argmin}}\ \hat{R}_{n,\varphi}(f)$ = empirical risk minimizer
- $\bar{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}}\ R(f)$ = best element in the class

Step I. Bound for $R_\varphi(\hat{f}_n) - R_\varphi(\bar{f})$

the risk of $\hat{f}_n$ is computed conditionally on $\mathcal{F}$.

$$R_\varphi(\hat{f}_n) = R_\varphi(\hat{f}_n) + \underbrace{\hat{R}_{n,\varphi}(\hat{f}_n)}_{\leq \hat{R}_{n,\varphi}(\bar{f})} - \hat{R}_{n,\varphi}(\hat{f}_n) + R_\varphi(\bar{f}) - R_\varphi(\bar{f})$$

$$\leq R_\varphi(\hat{f}_n) + \hat{R}_{n,\varphi}(\bar{f}) - \hat{R}_{n,\varphi}(\hat{f}_n) + R_\varphi(\bar{f}) - R_\varphi(\bar{f})$$

$$\leq R_\varphi(\bar{f}) + 2 \underbrace{\sup_{f \in \mathcal{F}} \left| \hat{R}_{n,\varphi}(f) - R_\varphi(f) \right|}_{= \sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \{\varphi(-Y_i f(X_i)) - \mathbb{E}\varphi(-Y f(X))\} \right|}$$

We assume now that φ is an L-Lipschitz convex surrogate:
$\varphi : [-1, 1] \to \mathbb{R}_+$; $\forall u, v \in [-1,1]$, $|\varphi(u) - \varphi(v)| \leq L\,|u - v|$.

Ex:
- Hinge $\varphi(x) = \max(x+1, 0)$; $L = 1$
- Exp $\varphi(x) = e^x$; $L = e$
- Logistic $\varphi(x) = \log_2(1 + e^x)$; $L = \frac{e}{1+e} \log_2 e \simeq 2.43$

Recall that if $\forall x \in [-1,1]$ $\varphi'(x) \leq L$, then $\forall u, v \in [-1,1]$

$$\varphi(u) - \varphi(v) = \int_v^u \varphi'(x)\,dx \leq \int_v^u L\,dx = L(u - v) \qquad (u > v)$$

Consider the function (3)

$$(x_1,y_1)\cdots(x_n,y_n) \xrightarrow{g} \sup_{f\in\mathcal{F}} \left| \frac{1}{n}\sum_{i=1}^n \varphi(-y_i f(x_i)) - R_\varphi(f)\right|$$

Then

$$g\Big((x_1,y_1),\dots,(x_i,y_i),\dots,(x_n,y_n)\Big)$$

Change only the i-th entry

$$- g\Big((x_1,y_1),\dots,(x_i',y_i'),\dots,(x_n,y_n)\Big)$$

$$\leq \frac{1}{n}\big(\varphi(1)-\varphi(-1)\big) \leq \frac{2L}{n}.$$

φ is non-decreasing φ is L-Lipschitz

$\Rightarrow$ We can apply the bounded difference inequality (p.16 in SL: VAPNIK CHERVONENKIS THEORY)

$$\mathbb{E}\left(\left| \sup_{f\in\mathcal{F}} |\hat{R}_{n,\varphi}(f) - R(f)| - \mathbb{E}(\dots)\right| > \varepsilon\right) \leq 2\exp\left(-\frac{2\varepsilon^2}{\sum\left(\frac{2L}{n}\right)^2}\right)$$

$\Leftrightarrow$

$$\sup_{f\in\mathcal{F}} |\hat{R}_{n,\varphi}(f) - R(f)| \leq \mathbb{E}(\dots) + 2L\sqrt{\frac{\log(2/\delta)}{2n}}$$

w.p. $\geq 1-\delta$.

$\Rightarrow$ We can now focus on bounding $\mathbb{E}\left(\sup_{f\in\mathcal{F}} |\hat{R}_{n,\varphi}(f) - R(f)|\right)$

Using the symmetrization trick, $\mathbb{E}(\dots)$ is upper-bounded by twice the Rademacher complexity

$$\mathcal{R}_n(\varphi\circ\mathcal{F}) := \sup_{(x_1,y_1)\cdots(x_n,y_n)} \mathbb{E}\left\{\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{i=1}^n \sigma_i \varphi(-y_i f(x_i))\right|\right\}$$

compare with the expression on page 21 in SL: VC THEORY.

$\sigma_i = \pm 1$ w.p. $1/2$.

The convex surrogate φ is such that $\varphi(0)=1$. To apply a contraction inequality on $\mathcal{R}_n(\varphi\circ\mathcal{F})$, we need $\psi(0)=0$ $\Rightarrow$ put $\psi(x) = \varphi(x)-1$. Then (4)

$$\mathbb{E}(\dots) \leq 2\,\mathcal{R}_n(\psi\circ\mathcal{F})$$

and

$$\mathcal{R}_n(\psi\circ\mathcal{F}) \leq 2L\,\mathcal{R}_n(\mathcal{F}),$$

contraction inequality (not trivial) [details omitted]

where $$\mathcal{R}_n(\mathcal{F}) = \sup_{(x_1,y_1)\dots(x_n,y_n)} \mathbb{E}\left\{\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{i=1}^n \sigma_i y_i f(x_i)\right|\right\}.$$

Summary: So far, we have established that

$$\boxed{R_\varphi(\hat{f}_n) - R_\varphi(\bar{f}) \leq 8L\,\mathcal{R}_n(\mathcal{F}) + 4L\sqrt{\frac{\log(2/\delta)}{2n}} \quad \text{w.p.} \geq 1-\delta}$$

Step II. Bound for $\mathcal{R}_n(\mathcal{F})$.

Recall that $\mathcal{F} = \left\{\sum_{j=1}^M \beta_j f_j(\cdot) \;\middle|\; \|\beta\|_1 \leq 1;\ f_j: X \to [-1,1]\right\}$

fixed in advance (and so is M)

Then

$$\mathcal{R}_n(\mathcal{F}) = \sup \mathbb{E}\left\{\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{j=1}^M \beta_j \sum_{i=1}^n \sigma_i y_i f_j(x_i)\right|\right\}$$

Put $$|Z_\beta| := \left|\sum_{j=1}^M \beta_j \sum_{i=1}^n \sigma_i y_i f_j(x_i)\right|.$$

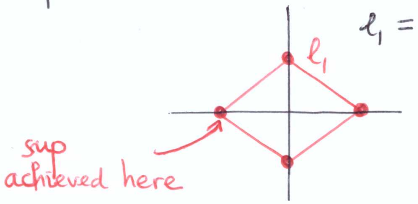
±1 ±1

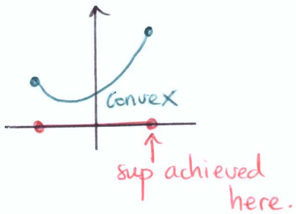
Note that $\beta \mapsto |Z_\beta|$ is a convex function of β

$$\left|\sum_{j=1}^{M} \left(\lambda \beta_{1,j} + (1-\lambda)\beta_{2,j}\right) \sum_{i=1}^{n} \sigma_i y_i f_j(x_i)\right|$$

$$\leq \lambda \left|\sum_j \beta_{1,j} \sum_i \sigma_i y_i f_j(x_i)\right| + (1-\lambda)\left|\sum_j \beta_{2,j} \sum_i \sigma_i y_i f_j(x_i)\right|$$

$\Rightarrow$ $\sup_{\|\beta\|_1 \leq 1} |Z_\beta|$ is achieved at a vertex of the unit ball $\ell_1 = \{\beta \mid \|\beta\|_1 \leq 1\}$





$\Rightarrow$ Consider the set

$$B_M := \left\{ \begin{pmatrix}1\\0\\ \vdots\\0\end{pmatrix} \begin{pmatrix}0\\1\\0\\ \vdots\end{pmatrix} \cdots \begin{pmatrix}0\\ \vdots\\0\\1\end{pmatrix} \begin{pmatrix}-1\\0\\ \vdots\\0\end{pmatrix} \begin{pmatrix}0\\-1\\0\\ \vdots\end{pmatrix} \cdots \begin{pmatrix}0\\ \vdots\\0\\-1\end{pmatrix} \right\}$$

$|B_M| = 2M$,

so that $\mathbb{E} \sup_{f \in \mathcal{F}} |Z_\beta| = \mathbb{E} \max_{\beta \in B_M} |Z_\beta|$

Proceed as usual:

$$\mathbb{E} \max_{\beta \in B_M} |Z_\beta| = \frac{1}{s} \log \exp s\, \mathbb{E} \max |Z_\beta|$$

$$\leq \frac{1}{s} \log \mathbb{E}\, e^{s \max |Z_\beta|}$$

$$\leq \frac{1}{s} \log \sum_{\beta \in B_M} e^{s Z_\beta}$$

Hoeffding lemma

$$\leq \frac{1}{s} \log \left\{ |B_M| \left(e^{\frac{s^2 2^2}{8}}\right)^n \right\}$$

$|\sigma_i y_i \sum \beta_j f_j(x_i)| < \sum |\beta_j| |f_j| = 1$

$$\mathbb{E} \max_{\beta \in B_M} |Z_\beta| \leq \frac{\log |B_M|}{s} + \frac{sn}{2}$$

Take $s = \sqrt{(2 \log |B_M|)/n}$

$$\leq 2\sqrt{\frac{n}{2} \log |B_M|}, \quad \text{with } |B_M| = 2M.$$

$$\Rightarrow \mathcal{R}_n(\mathcal{F}) \leq \sqrt{\frac{2 \log 2M}{n}}$$

Summary:

$$\boxed{R_\varphi(\hat{f}_n) - R_\varphi(\bar{f}) \leq 8L\sqrt{\frac{2\log 2M}{n}} + 4L\sqrt{\frac{\log(2/\delta)}{2n}} \quad \text{w.p.} \geq 1-\delta}$$

Step III. Bound for the excess risk $R(\text{sign}\, \hat{f}_n) - R^*$

risk under 0/1 loss

$$R(\text{sign}\, \hat{f}_n) - R^* \leq 2c\left(R_\varphi(\hat{f}_n) - R_\varphi^*\right)^\gamma$$

Zhang lemma (p.7 in SL: CONVEX RELAXATION)

$$= 2c\left(R_\varphi(\hat{f}_n) - R_\varphi(\bar{f}) + R_\varphi(\bar{f}) - R_\varphi^*\right)^\gamma$$

$$\leq 2c\left(8L\sqrt{\frac{2\log 2M}{n}} + 4L\sqrt{\frac{\log(\frac{2}{\delta})}{2n}} + R_\varphi(\bar{f}) - R_\varphi^*\right)^\gamma$$

since for $\alpha_i \geq 0$ and $\gamma \in [0,1]$,

$(\alpha_1 + \alpha_2 + \alpha_3)^\gamma \leq \alpha_1^\gamma + \alpha_2^\gamma + \alpha_3^\gamma$

$$\leq 2c\left(R_\varphi(\bar{f}) - R_\varphi^*\right)^\gamma + 2c\left(8L\sqrt{\frac{2\log 2M}{n}}\right)^\gamma + 2c\left(4L\sqrt{\frac{\log(2/\delta)}{2n}}\right)^\gamma$$

Theorem: Risk Bound for Boosting (7)

Consider. $\mathcal{F} = \left\{ \sum_{j=1}^{M} \beta_j f_j(\cdot) \mid \|\beta\|_1 \leq 1,\ f_j : \mathcal{X} \to [-1,1] \right\}$

- φ = an L-Lipschitz convex surrogate.
- $\hat{f}_n = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{R}_{n,\varphi}(f)$.

Then

$$\underbrace{R(\operatorname{sign} \hat{f}_n) - R^*}_{\text{excess risk of the hard classifier}} \leq 2c\left(R_\varphi(\bar{f}) - R_\varphi^*\right)^\gamma + 2c\left(8L\sqrt{\frac{2\log 2M}{n}}\right)^\gamma + 2c\left(4L\sqrt{\frac{\log(2/\delta)}{2n}}\right)^\gamma$$

w.p $\geq 1-\delta$

II - AdaBoost

II.1. Derivation of the algorithm

AdaBoost (introduced by Freund & Shapire (1995)) uses an exponential loss:

- $\hat{f}_n \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{R}_{n,\varphi}(f)$
- $\hat{R}_{n,\varphi}(f) = \frac{1}{n} \sum_{i=1}^{n} \varphi(-Y_i f(X_i)) = \frac{1}{n}\sum_{i=1}^{n} e^{-Y_i f(X_i)}$
- $\mathcal{F} = \left\{ \sum_{j=1}^{M} \beta_j f_j(\cdot) \mid \|\beta\|_1 \leq 1 \quad f_j : \mathcal{X} \to [-1,1] \right\}$

AdaBoost is an algorithm that approximately computes $\hat{f}_n$.
Several assumptions are relaxed:

→ M is not fixed in advance // coefficients are not constrained in ℓ_1.

→ $f_j(\cdot)$ are not fixed in advance: the dictionary adapts to the data. We only require that f_j take a particular form

---

(8)

→ AdaBoost does not minimize $\frac{1}{n}\sum_{i=1}^{n} e^{-Y_i f(X_i)}$ directly, but performs this recursively.

Suppose that at iteration $(m-1)$, the current estimate is

$$f^{(m-1)}(x) = \sum_{k=1}^{m-1} \beta_k \overset{\text{taking values in } \{-1,1\}}{f_k(x)}$$

Iteration m is searching for the 'best' value of β_m and f_m such that:

$$\boxed{(\beta_m, f_m) \in \underset{\beta, f}{\operatorname{argmin}} \sum_{i=1}^{n} \varphi\left(-y_i\left[f^{(m-1)}(x_i) + \beta f(x_i)\right]\right)}$$

OPTIMIZATION PROBLEM SOLVED BY AdaBoost.

Put $w_i^{(m)} := \exp\left(-y_i f^{(m-1)}(x_i)\right)$. Then

$$(\beta_m, f_m) \in \underset{\beta, f}{\operatorname{argmin}} \underbrace{\left\{ \sum_{i=1}^{n} w_i^{(m)} e^{-\beta f(x_i) y_i} \right\}}_{\text{Call this } W^{(m)}}$$

$$W^{(m)} = e^{-\beta} \sum_{\substack{i \\ y_i = f(x_i)}} w_i^{(m)} + e^{\beta} \sum_{\substack{i \\ y_i \neq f(x_i)}} w_i^{(m)}$$

$$= e^{-\beta} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i = f(x_i)) + e^{\beta} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f(x_i))$$

$$= e^{-\beta} \sum_{i=1}^{n} w_i^{(m)} \left(1 - \mathbb{1}(y_i \neq f(x_i))\right) + e^{\beta} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f(x_i))$$

$$= \left(e^{\beta} - e^{-\beta}\right) \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f(x_i)) + \underbrace{e^{-\beta} \sum_{i=1}^{n} w_i^{(m)}}_{\text{indpt of } f}$$

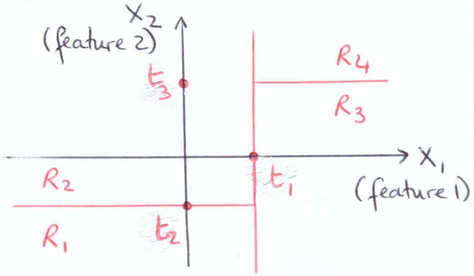
⇒ At step m, the weak classifier f_m is selected according to:

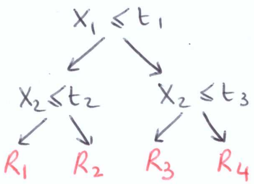
$$\boxed{f_m \in \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f(x_i))}$$

Weighted version of the empirical 0/1 loss

Example: Fit a decision tree to the weighted version of the learning sample

CART procedure.





4 terminal nodes = 4 regions.

Q: How to choose the split variable + split point?

Q: How to classify point in each region?

↳ Answer to this question is relatively easy: For each class $\in \{-1, 1\}$, compute the weighted proportion of points falling into region R_ℓ:

$$\hat{p}_{\ell k}^{(m)} = \frac{\sum_{x_i \in R_\ell} w_i^{(m)} \mathbb{1}(y_i = k)}{\sum_{x_i \in R_\ell} w_i^{(m)}}$$

Then classify according to $\hat{k} = \underset{k \in \{-1,1\}}{\operatorname{argmax}} \hat{p}_{\ell k}^{(m)}$

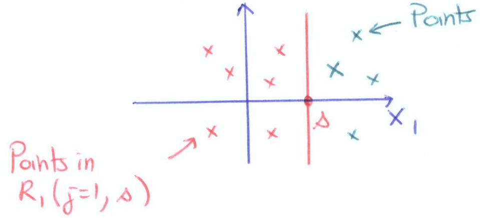
↳ growing a tree; ie finding the terminal regions, is more tricky = use a GREEDY ALGORITHM + the prune it (using e.g. cost-complexity pruning).

More precisely, for a splitting variable j and a splitting point s, consider

$$R_1(j,s) = \{x_1, \dots, x_n \mid x_{kj} \leq s\}$$

$$x_k = (x_{k1}, \dots, x_{kd}) \in \mathbb{R}^d$$

$$R_2(j,s) = \{x_1, \dots, x_n \mid x_{kj} > s\}$$



Perform an exhaustive search: for each j and s (there are finitely many - why?), compute

$$\sum_{x_i \in R_1(j,s)} w_i^{(m)} \mathbb{1}(y_i \neq \hat{y}_1^{(m)}) + \sum_{x_i \in R_2(j,s)} w_i^{(m)} \mathbb{1}(y_i \neq \hat{y}_2^{(m)})$$

where $$\hat{y}_k^{(m)} = \underset{y \in \{-1,1\}}{\operatorname{argmax}} \left(\frac{\sum_{x_i \in R_k(j,s)} w_i^{(m)} \mathbb{1}(y_i = y)}{\sum_{x_i \in R_k(j,s)} w_i^{(m)}} \right)$$

And return the value of (j,s) that minimizes this.

⇒ Grow a large ~~tree~~ T_j and then prune it to avoid overfitting.

- Complexity criterion used is

$$C_\lambda(T) = \sum_{\ell=1}^{|T|} |R_\ell| Q_\ell(T) + \lambda |T|$$

$Q_\ell(T)$: goodness of fit measure; λ: Tuning Parameter; $|T|$: # of terminal leaves

↳ $Q_\ell(T) = \frac{1}{|R_\ell|} \sum_{x_i \in R_\ell} w_i^{(m)} \mathbb{1}(y_i \neq y_\ell^{(m)})$ [weighted misclass]

where $y_\ell^{(m)} = \underset{j \in \{-1,1\}}{\operatorname{argmax}} \hat{p}_{\ell j}^{(m)}$

↳ $Q_\ell(T) = \sum_{j \in \{-1,1\}} \hat{p}_{\ell j}^{(m)} (1 - \hat{p}_{\ell j}^{(m)})$ [Gini index] ↳ CART

↳ $Q_\ell(T) = - \sum_{j \in \{-1,1\}} \hat{p}_{\ell j}^{(m)} \log \hat{p}_{\ell j}^{(m)}$

For each λ, find a subtree $T_\lambda \subset T$ minimizing $C_\lambda(T)$. (11)

How? Using weakest link pruning = successively collapse internal nodes that result in the smallest increase of $\sum |R_\ell| \, Q_\ell(T)$, until you reach the root. This gives a finite sequence of trees, and it can be shown that this sequence must contain T_λ. The value of λ is usually selected using cross-validation.

Remarks: (i) In boosting, growing large trees + pruning them is not needed. Typically, a rule of thumb is that up to 5/6 terminal nodes is OK. You can also consider a tree with a single split / two terminal nodes (aka STUMPS).

(ii) In the approach above, the weak (tree) classifiers f_m are restricted to taking values in $\{1, \dots, K\}$, and in the context of binary classification, in $\{-1, 1\}$. AdaBoost then combines scaled classification trees $\beta_m f_m$.

"Discrete AdaBoost"

Alternatively, you may boost classification trees without this restriction:

$$\{R_{jm}, \gamma_{jm}\} = \arg\min \sum_{i=1}^{n} \ell\left(y_i, f^{(m-1)}(x_i) + T(x_i)\right)$$

terminal regions

prediction in each terminal region

min taken over the parameters of the tree T, ie the split variables, split points, and predicted value.

For example, with a square loss, the problem reduces to: (11a)

$$\boxed{\arg\min \sum_{i=1}^{n} \Big(\underbrace{y_i - f^{(m-1)}(x_i)}_{\text{current residuals}} - T(x_i) \Big)^2}$$

$\equiv$ fit a regression tree that best fits the current residuals.

make the problem (p.9) $\sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(\dots)$ convex

With an exponential loss, we need to solve

$$\boxed{\arg\min \sum_{i=1}^{n} w_i^{(m)} \exp\left(-y_i T(x_i)\right)}.$$

Once the terminal regions are computed, the predicted value in R_{jm} is

$$\gamma_{jm} = \frac{1}{2} \log\left(\frac{\sum_{x_i \in R_{jm}} w_i^{(m)} \mathbb{1}(y_i = 1)}{\sum_{x_i \in R_{jm}} w_i^{(m)} \mathbb{1}(y_i = -1)} \right)$$

"Real AdaBoost"

Not restricted to $\{-1, 1\}$

Back on track: once f_m is constructed, the next step is to compute β_m:

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{n} w_i^{(m)} e^{-\beta y_i f_m(x_i)}$$

$$W^{(m)} = e^{-\beta} \Big\{ \sum_{y_i = f_m(x_i)} w_i^{(m)} + e^{2\beta} \sum_{y_i \neq f_m(x_i)} w_i^{(m)} \Big\}$$

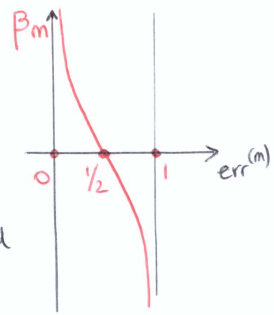
$$= e^{-\beta} \Big\{ \sum_{i=1}^{n} w_i^{(m)} + (e^{2\beta} - 1) \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f_m(x_i)) \Big\}$$

$$= e^{-\beta} \Big(\sum_{i=1}^{n} w_i^{(m)} \Big) \Big\{ 1 + (e^{2\beta} - 1) \underbrace{\frac{\sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f_m(x_i))}{\sum_{i=1}^{n} w_i^{(m)}}}_{\text{err}^{(m)} \in (0,1)} \Big\}$$

$$= e^{-\beta} \Big(\sum_{i=1}^{n} w_i^{(m)} \Big) \Big\{ 1 + (e^{2\beta} - 1) \, \text{err}^{(m)} \Big\}.$$

Thus $\beta_m = \arg\min_{\beta} \; e^{-\beta}\left\{1 + (e^{2\beta}-1)\, err^{(m)}\right\}$

$$\boxed{\beta_m = \frac{1}{2}\log\left(\frac{1-err^{(m)}}{err^{(m)}}\right)}$$



Meaning: if classifier f_m (weak) performs well; so that $err^{(m)}$ is close to 0, the weight associated to f_m is large positively $\Rightarrow f_m$ has a strong contribution to the final classification $\sum_{k=1}^{M} \beta_k f_k$.

→ Once β_m and f_m are computed, update

- $f^{(m)}(x) = f^{(m-1)}(x) + \beta_m f_m(x)$
- $w_i^{(m+1)} = \exp\left(-y_i f^{(m)}(x_i)\right)$

$$= \exp\left(-y_i f^{(m-1)}(x_i)\right)\exp\left(-y_i \beta_m f_m(x_i)\right)$$

$$= w_i^{(m)} \exp\left(-y_i \beta_m f_m(x_i)\right)$$

Remark:

If $y_i = f_m(x_i)$, then $-y_i f_m(x_i) = -1 = 2\,\mathbb{1}(y_i \neq f_m(x_i)) - 1$

If $y_i \neq f_m(x_i)$, then $-y_i f_m(x_i) = 1 = 2\,\mathbb{1}(y_i \neq f_m(x_i)) - 1$

Thus

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{\beta_m\left(2\,\mathbb{1}(y_i \neq f_m(x_i)) - 1\right)\right\}$$

$$= w_i^{(m)} \exp\left\{\underset{2\beta_m}{\overset{\alpha_m}{\shortparallel}}\; \mathbb{1}(y_i \neq f_m(x_i))\right\} e^{-\beta_m}$$

independent of i $\Rightarrow$ can be dropped (cancel out in the def of $\hat{\beta}_{ke}^{(m+1)}$)

AdaBoost Algorithm.

(i) Initialize observation weights to $w_i^{(1)} = 1/n$.

(ii) For $m = 1$ to M

→ Fit a classifier f_m to the learning sample using weights $w_i^{(m)}$

→ Compute $err^{(m)} = \dfrac{\sum_{i=1}^{n} w_i^{(m)} \mathbb{1}(y_i \neq f_m(x_i))}{\sum_{i=1}^{n} w_i^{(m)}}$

→ Compute $\alpha_m = \log\left(\dfrac{1-err^{(m)}}{err^{(m)}}\right)$

→ Update $w_i^{(m+1)} \leftarrow w_i^{(m)}\left(\exp\left(\alpha_m \mathbb{1}(y_i \neq f_m(x_i))\right)\right)$

provided $\alpha_m > 0$, a misclassified observation has its weight increased at the next iteration

(iii) Output $f^{(M)}(x) = \sum_{m=1}^{M} \beta_m f_m(x)$ and classify according to sign $f^{(M)}$.

Remark: The weights $w_i^{(m+1)}$ are sometimes renormalized so that they sum up to 1 ($\equiv$ probability distribution on observations).

Update is $w_i^{(m+1)} = \dfrac{w_i^{(m)}}{Z_m} e^{-\beta_m y_i f_m(x_i)}$

so that $\left[\sum_{i=1}^{n} w_i^{(m+1)} = 1\right]$

II.2. Performance bound

- $\boxed{\text{We show that provided } err^{(\ell)} = \tfrac{1}{2} - \gamma_\ell \text{; for } \gamma_\ell \geq \gamma > 0 \;\; \forall \ell, \text{ then } \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i \neq f^{(M)}(x_i)) \leq e^{-2\gamma^2 M}}$

$\Rightarrow$ Training error can be made arbitrarilly small, provided M is sufficiently large.

→ **Step I** Cascading the weights,

$$w_i^{(M+1)} = \frac{w_i^{(M)}}{Z_M} \exp\left(-y_i \beta_M f_M(x_i)\right)$$

$$w_i^{(M)} = \frac{w_i^{(M-1)}}{Z_{M-1}} \exp\left(-y_i \beta_{M-1} f_{M-1}(x_i)\right)$$

$$\vdots$$

$$w_i^{(2)} = \frac{w_i^{(1)}}{Z_1} \exp\left(-y_i \beta_1 f_1(x_i)\right)$$

$$w_i^{(1)} = \frac{1}{n}$$

$$w_i^{(M+1)} = \frac{1}{n} \frac{1}{\prod_{\ell=1}^{M} Z_\ell} \exp\left(-y_i \sum_{\ell=1}^{M} \beta_\ell f_\ell(x_i)\right)$$

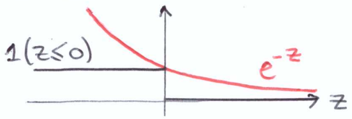
$$w_i^{(M+1)} = \frac{1}{n \prod_{\ell=1}^{M} Z_\ell} \exp\left(-y_i f^{(M)}(x_i)\right)$$

→ **Step II**

$$\frac{1}{n}\sum_{i=1}^{n} 1\left(y_i \neq f^{(M)}(x_i)\right) = \frac{1}{n}\sum_{i=1}^{n} 1\left(y_i f^{(M)}(x_i) \leq 0\right)$$

training error

$$\leq \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i f^{(M)}(x_i)\right)$$



$$= \frac{\cancel{1}}{\cancel{n}} \sum_{i=1}^{n} \left(\cancel{n}\left[\prod_{\ell=1}^{M} Z_\ell\right] w_i^{(M+1)}\right)$$

$$= \prod_{\ell=1}^{M} Z_\ell \underbrace{\left(\sum_{i=1}^{n} w_i^{(M+1)}\right)}_{=1}$$

$$= \prod_{\ell=1}^{M} Z_\ell$$

Next, $Z_\ell = \sum_{i=1}^{n} w_i^{(\ell)} \exp\left(-\beta_\ell y_i f_\ell(x_i)\right)$

$$= \sum_{y_i = f_\ell(x_i)} w_i^{(\ell)} e^{-\beta_\ell} + \sum_{y_i \neq f_\ell(x_i)} w_i^{(\ell)} e^{\beta_\ell}$$

$$= e^{-\beta_\ell} + err^{(\ell)}\left(e^{\beta_\ell} - e^{-\beta_\ell}\right)$$

$$err^{(\ell)} = \sum_{i=1}^{n} w_i^{(\ell)} 1\left(y_i \neq f_\ell(x_i)\right)$$

[compare with page 11 / weights sum to 1]

$$= 2\sqrt{err^{(\ell)}\left(1 - err^{(\ell)}\right)}$$

since $\beta_\ell = \frac{1}{2}\log\left(\frac{1-err^{(\ell)}}{err^{(\ell)}}\right)$

Thus $\boxed{\frac{1}{n}\sum_{i=1}^{n} 1\left(y_i \neq f^{(M)}(x_i)\right) \leq 2^M \prod_{\ell=1}^{M} \sqrt{err^{(\ell)}\left(1 - err^{(\ell)}\right)}}$

→ **Step III**. Conclusion.

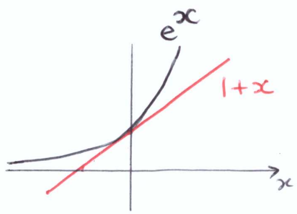
Under the assumption that $err^{(\ell)} = \frac{1}{2} - \gamma_\ell$,

$$2\sqrt{err^{(\ell)}\left(1-err^{(\ell)}\right)} = 2\sqrt{\frac{1}{4} - \gamma_\ell^2}$$

$$= \sqrt{1 - 4\gamma_\ell^2}$$

$$\leq \sqrt{e^{-4\gamma_\ell^2}}$$

$$= e^{-2\gamma_\ell^2}$$



$$\Rightarrow \frac{1}{n}\sum_{i=1}^{n} 1\left(y_i \neq f^{(M)}(x_i)\right) \leq \prod_{\ell=1}^{M} \exp\left(-2\gamma_\ell^2\right)$$

$$= \exp\left(-2\sum_{\ell=1}^{M} \gamma_\ell^2\right)$$

$$\leq \exp\left(-2M\gamma^2\right)$$

provided $\gamma_\ell \geq \gamma > 0$

■

⇒ Provided weak classifiers perform slightly better than a random guess, AdaBoost returns a proportion of the training samples misclassified arbitrarily small. Note that if $\mathrm{err}^{(\ell)} > \frac{1}{2}$, the corresponding coefficient β_ℓ is negative, and AdaBoost does not update the weights in the right direction.

⇒ The bound suggests that we can achieve any degree of accuracy, provided M is large enough. However, AdaBoost is prone to overfitting, and M should be limited in size. In practice, one approach is to hold out a part of the training sample, and to test the performance of $f^{(M)}$ as a function of M. In fact, we can show that $VC(\mathcal{F}_M) = O(VC(\mathcal{F}_\ell)\, M \log M)$, for $\mathcal{F}_M = \{ \mathrm{sign}(\sum_{m=1}^{M} \beta_m f_m),\ \beta_m \in \mathbb{R},\ f_m \in \mathcal{F}_\ell \}$.

II. 3. AdaBoost for K-class classification

* Coding-scheme adopted here for K-class classification is $Y = (Y_1, \dots, Y_K)^t$, where $Y_k = \begin{cases} 1 & \text{if } X \text{ belongs to class } k \\ -\frac{1}{K-1} & \text{otherwise} \end{cases}$

$(K \geq 3)$

* Goal: Construct $f(x) = (f_1(x), \text{—}, f_K(x))^t$, under the constraint $\sum f_k(x) = 0$, and classify x according to the largest value of $f_k(x)$, $k = 1, \text{—}, K$.

* Is this a good idea? It is, under a generalized version of the exponential loss

$$\boxed{\ell(Y, f(X)) = \exp\left(-\frac{1}{K} Y^t f(X)\right)}$$

Exp loss for K-class classification.

⇒ Population minimizer $f^*(x) = \underset{f}{\mathrm{argmin}}\ \mathbb{E}(\ell(Y, f(X)) \mid X = x)$

$$f^*(x) = (f_1^*(x), \text{—}, f_K^*(x))$$

is such that

$$f_k^*(x) = (K-1)\left\{ \log \mathbb{P}(Y_k = 1 \mid X = x) - \frac{1}{K}\sum_{j=1}^{K} \log \mathbb{P}(Y_j = 1 \mid X = x) \right\}$$

Proba dos ∈ class k

And thus, classifying x according to the largest value of $f_k(x)$ makes sense.

proof: $f^*(x) = \underset{f}{\mathrm{argmin}}\ \mathbb{E}\left\{ \exp\left(-\frac{1}{K} Y^t f(X)\right) \mid X = x \right\}$

Constrained optimization problem

The Lagrangian is

$$\mathcal{L}(f, \lambda) = \underbrace{\mathbb{E}\left\{ \exp\left(-\frac{1}{K} Y^t f(X)\right) \mid X = x \right\}} - \lambda \sum_{k=1}^{K} f_k(x)$$

$$= \mathbb{E}\left\{ \exp\left(-\frac{1}{K}[Y_1 f_1 + \dots + Y_K f_K]\right) \mid X = x \right\}$$

$$= \exp\left\{-\frac{1}{K}\left[f_1 - \frac{1}{K-1} f_2 - \dots - \frac{1}{K-1} f_K \right]\right\} \mathbb{P}(Y_1 = 1 \mid X = x)$$

$$+ \exp\left\{-\frac{1}{K}\left[-\frac{1}{K-1} f_1 + f_2 - \dots - \frac{1}{K-1} f_K \right]\right\} \mathbb{P}(Y_2 = 1 \mid X = x)$$

$$\dots$$

$$+ \exp\left\{-\frac{1}{K}\left[-\frac{1}{K-1} f_1 - \frac{1}{K-1} f_2 - \dots + f_K \right]\right\} \mathbb{P}(Y_K = 1 \mid X = x)$$

Making use of $-\frac{1}{K-1} = 1 - \frac{K}{K-1}$,

$$
\begin{aligned}
&= \exp\left\{-\frac{1}{K}\Big[\overbrace{f_1+\cdots+f_K}^{=0} - \frac{K}{K-1}\big(\overbrace{f_2+\cdots+f_K}^{=-f_1}\big)\Big]\right\} \mathbb{P}(Y_1=1 \mid X=x) \\
&+ \exp\left\{-\frac{1}{K}\Big[\overbrace{f_1+\cdots+f_K}^{=0} - \frac{K}{K-1}\big(\overbrace{f_1+f_3+\cdots+f_K}^{=-f_2}\big)\Big]\right\} \mathbb{P}(Y_2=1 \mid X=x) \\
&\;\;\vdots \\
&+ \exp\left\{-\frac{1}{K}\Big[\overbrace{f_1+\cdots+f_K}^{=0} - \frac{K}{K-1}\big(\overbrace{f_1+\cdots+f_{K-1}}^{=-f_K}\big)\Big]\right\} \mathbb{P}(Y_K=1 \mid X=x) \\
&= \exp\left\{\frac{-f_1}{K-1}\right\} \mathbb{P}(Y_1=1 \mid X=x) + \cdots + \exp\left\{-\frac{f_K}{K-1}\right\} \mathbb{P}(Y_K=1 \mid X=x)
\end{aligned}
$$

$\Rightarrow$ Lagrangian becomes

$$
\mathcal{L}(f,\lambda) = \sum_{k=1}^{K} \exp\left\{-\frac{f_k}{K-1}\right\} \mathbb{P}(Y_k=1 \mid X=x) - \lambda \sum_{k=1}^{K} f_k
$$

Taking derivatives with respect to f_k and λ yields

$$
\begin{cases}
\dfrac{\partial \mathcal{L}(f,\lambda)}{\partial f_k} = -\dfrac{1}{K-1} \exp\left\{-\dfrac{f_k^*}{K-1}\right\} \mathbb{P}(Y_k=1 \mid X=x) - \lambda = 0 \\[2ex]
\dfrac{\partial \mathcal{L}(f,\lambda)}{\partial \lambda} = -\displaystyle\sum_{k=1}^{K} f_k^* = 0
\end{cases}
$$

We get from the first relation that $\exp\left\{-\frac{f_k^*}{K-1}\right\} \times \mathbb{P}(\ldots) = -\lambda(K-1)$

$$\Updownarrow$$

$$
f_k^* = (K-1)\log \mathbb{P}(Y_k=1 \mid X=x) - (K-1)\log[-\lambda(K-1)]
$$

Enforcing that the f_k^* sum to zero, we get

$$
\frac{1}{K-1}\sum_k f_k^* = \sum_k \log \mathbb{P}(Y_k=1 \mid X=x) - \sum_k \log[-\lambda(K-1)] = 0
$$

$$\Leftrightarrow$$

$$
\lambda = -\frac{1}{K-1}\exp\left\{\frac{1}{K}\sum_k \log \mathbb{P}(Y_k=1 \mid X=x)\right\}
$$

Plugging this value of λ back into the expression of f_k^* yields the result. ■

<u>Remark</u>. Once the f_k^* are (somehow) estimated, the posterior class probabilities can be computed:

$$
\underset{\substack{\uparrow \\ \text{sum to one:}}}{\mathbb{P}(Y_k=1 \mid X=x)} = \exp\left\{\frac{f_k^*}{K-1}\right\}\left(\prod_{j=1}^{K} \mathbb{P}(Y_j=1 \mid X=x)\right)^{\frac{1}{K}}
$$

$$
1 = \underset{\substack{\uparrow}}{\left(\prod_{j=1}^{K} \mathbb{P}(Y_j=1 \mid X=x)\right)^{\frac{1}{K}}}\left(\sum_{k=1}^{K}\exp\left\{\frac{f_k^*}{K-1}\right\}\right)
$$

Plugging this expression of $\left(\prod_{j=1}^{K}\mathbb{P}(\ldots)\right)^{\frac{1}{K}}$ back into the posterior proba gives:

$$
\boxed{\mathbb{P}(Y_k=1 \mid X=x) = \frac{\exp\left\{\frac{f_k^*}{K-1}\right\}}{\sum_{j=1}^{K}\exp\left\{\frac{f_j^*}{K-1}\right\}}}
$$

- <u>AdaBoost algorithm</u>.

↘ Classical AdaBoost (K=2) : $\underset{\substack{\uparrow \\ \text{soft classifier}}}{f^{(M)}}(x) = \sum_{\ell=1}^{M} \beta_\ell \underset{\substack{\uparrow \\ \in\{-1,1\} \\ \text{(or, more generally, in } [-1,1])}}{f_\ell}(x)$

↘ K-class AdaBoost : $f^{(M)}(x) = \sum_{\ell=1}^{M} \beta_\ell \, f_\ell(x)$ (20)

Each weak classifier take values in

$$\left\{ \left(1, -\frac{1}{K-1}, \ldots, -\frac{1}{K-1}\right), \left(-\frac{1}{K-1}, 1, \ldots, -\frac{1}{K-1}\right), \ldots, \left(-\frac{1}{K-1}, -\frac{1}{K-1}, \ldots, 1\right) \right\}$$

The (β_ℓ, f_ℓ) are fit in a stagewise manner:

$$(\beta_\ell, f_\ell) \in \underset{(\beta, f)}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(y_i, f^{(\ell-1)}(x_i) + \beta f(x_i)\right)$$

vectors in $\mathbb{R}^K$

$$= \underset{(\beta, f)}{\operatorname{argmin}} \sum_{i=1}^{n} \exp\left\{-\frac{1}{K} y_i^t \left(f^{(\ell-1)}(x_i) + \beta f(x_i)\right)\right\}$$

$$\boxed{(\beta_\ell, f_\ell) = \underset{(\beta, f)}{\operatorname{argmin}} \sum_{i=1}^{n} w_i^{(\ell)} \exp\left\{-\frac{1}{K} \beta y_i^t f(x_i)\right\} \quad \text{where } w_i^{(\ell)} = \exp\left\{-\frac{1}{K} y_i^t f^{(\ell-1)}(x_i)\right\}}$$

We proceed as for binary classification:

$$\sum_{i=1}^{n} w_i^{(\ell)} \exp\left\{-\frac{1}{K} \beta y_i^t f(x_i)\right\}$$

$$= \sum_{\substack{\text{correctly} \\ \text{classified} \\ \text{observations}}} w_i^{(\ell)} \exp\left\{-\frac{1}{K} \beta y_i^t f(x_i)\right\} + \sum_{\substack{\text{incorrectly} \\ \text{classified} \\ \text{obs}}} w_i^{(\ell)} \exp\{\ldots\}$$

More formally, with $y_i = (y_{i,1}, \ldots, y_{i,K})$
correctly classified obs $= \{(x_i, y_i) \mid y_i = f(x_i)\}$

↳ For a correctly classified observation, (21)

$$y_i^t f(x_i) = 1 + \underbrace{\left(\frac{1}{K-1}\right)^2 + \cdots + \left(\frac{1}{K-1}\right)^2}_{K-1} = \frac{K}{K-1}$$

↳ For an incorrectly classified observation,

$$y_i^t f(x_i) = -\frac{2}{K-1} + \underbrace{\left(\frac{1}{K-1}\right)^2 + \cdots + \left(\frac{1}{K-1}\right)^2}_{K-2} = -\frac{K}{(K-1)^2}$$

Thus,

$$\sum_{i=1}^{n} w_i^{(\ell)} \exp\left\{-\frac{1}{K} \beta y_i^t f(x_i)\right\}$$

$$= \sum_{\text{correct}} w_i^{(\ell)} \exp\left(-\frac{\beta}{K-1}\right) + \sum_{\text{incorrect}} w_i^{(\ell)} \exp\left(\frac{\beta}{(K-1)^2}\right).$$

$$= e^{-\frac{\beta}{K-1}} \sum_{i=1}^{n} w_i^{(\ell)} + \left(e^{\frac{\beta}{(K-1)^2}} - e^{-\frac{\beta}{K-1}}\right) \underline{\sum w_i^{(\ell)} \mathbb{1}(y_i \neq f(x_i))}$$

Call this term $W^{(\ell)}$

To find f_ℓ, construct a classifier minimizing a weighted 0/1 loss
Ex: grow a tree.

- It remains to find β_ℓ.

$$W^{(\ell)} = e^{-\frac{\beta}{K-1}} \left(\sum_{i=1}^{n} w_i^{(\ell)}\right)\left[1 + \left(\frac{e^{\frac{\beta}{(K-1)^2}} - e^{-\frac{\beta}{K-1}}}{e^{-\frac{\beta}{K-1}}}\right) \underbrace{\frac{\sum w_i^{(\ell)} \mathbb{1}(y_i \neq f(x_i))}{\sum w_i^{(\ell)}}}_{\text{call this term } err^{(\ell)}}\right]$$

$$= e^{-\frac{\beta}{K-1}} \left(\sum_{i=1}^{n} w_i^{(\ell)}\right)\left[1 + \left\{\exp\left(\frac{\beta}{(K-1)^2} + \frac{\beta}{K-1}\right) - 1\right\} err^{(\ell)}\right]$$

Thus

$$\beta_\ell = \underset{\beta}{\operatorname{argmin}} \; e^{-\frac{\beta}{K-1}} \left[1 + \left\{\exp\left(\frac{\beta K}{(K-1)^2}\right) - 1\right\} err^{(\ell)}\right]$$

Put $u = \exp\left(\frac{\beta}{K-1}\right)$, and consider

$$\varphi(u) = u^{-1}\left(1 + err^{(\ell)}\left[u^{K/(K-1)} - 1\right]\right)$$

Solve $\varphi'(u^*) = 0 \Rightarrow \frac{K}{K-1}\log u^* = \log(K-1) + \log\left(\frac{1-err^{(\ell)}}{err^{(\ell)}}\right)$

Thus

$$\boxed{\begin{aligned}\beta_\ell &= \frac{(K-1)^2}{K}\left\{\log(K-1) + \log\left(\frac{1-err^{(\ell)}}{err^{(\ell)}}\right)\right\} \\ &= \frac{(K-1)^2}{K}\,\alpha_\ell\end{aligned}}$$

- Update: * $f^{(\ell)}(x) = f^{(\ell-1)}(x) + \beta_\ell f_\ell(x)$

  * $w_i^{(\ell+1)} = \exp\left\{-\frac{1}{K} y_i^t f^{(\ell)}(x_i)\right\}$

$$= w_i^{(\ell)} \exp\left\{-\frac{1}{K}\beta_\ell\, y_i^t f_\ell(x_i)\right\}$$

$$= \begin{cases} w_i^{(\ell)} \exp\left\{-\frac{K-1}{K}\alpha_\ell\right\} & \text{if } y_i = f_\ell(x_i) \\ w_i^{(\ell)} \exp\left\{\frac{\alpha_\ell}{K}\right\} & \text{if } y_i \neq f_\ell(x_i) \end{cases}$$

Multiply weights by $\exp\left(\frac{\alpha_\ell}{K}\right)$ = indpt of i

$\Leftrightarrow$ update

$$\begin{cases} w_i^{(\ell)} \exp(-\alpha_\ell) & \text{if } y_i = f_\ell(x_i) \\ w_i^{(\ell)} & \text{if } y_i \neq f_\ell(x_i) \end{cases}$$

$\Leftrightarrow$ update

$$\begin{cases} w_i^{(\ell)} & \text{if } y_i = f_\ell(x_i) \\ w_i^{(\ell)} \exp(\alpha_\ell) & \text{if } y_i \neq f_\ell(x_i) \end{cases}$$

$$\Rightarrow w_i^{(\ell+1)} = w_i^{(\ell)} \exp\left[\alpha_\ell\, \mathbb{1}(y_i \neq f_\ell(x_i))\right]$$

For the weights to be updated in the right direction, we need $\alpha_\ell > 0$; that is $\log(K-1) + \log\left(\frac{1-err^{(\ell)}}{err^{(\ell)}}\right) > 0$: in K-class classif, we do not longer require that the base classifier perform better than a random guess.

AdaBoost (K-class; $K \geq 3$)

① Initialize weights $w_i^{(1)} = \frac{1}{n}$

② For $\ell = 1$ to M,

→ Fit a classifier f_ℓ to the training sample with weights $w_i^{(\ell)}$.

→ Compute its error $err^{(\ell)} = \frac{\sum_{i=1}^{n} w_i^{(\ell)}\, \mathbb{1}(y_i \neq f_\ell(x_i))}{\sum_{i=1}^{n} w_i^{(\ell)}}$

→ $\alpha_\ell = \log(K-1) + \log\left(\frac{1-err^{(\ell)}}{err^{(\ell)}}\right)$

→ Update $w_i^{(\ell+1)} = w_i^{(\ell)} \exp\left[\alpha_\ell\, \mathbb{1}(y_i \neq f_\ell(x_i))\right]$

③ Output $f^{(M+1)}(x) = \sum_{\ell=1}^{M} \alpha_\ell f_\ell(x)$

SAMME algorithm (Zhu et al 2009) Plugging in β_ℓ makes no difference.

III_ GRADIENT BOOSTING

III.1. Gradient Boosting for regression

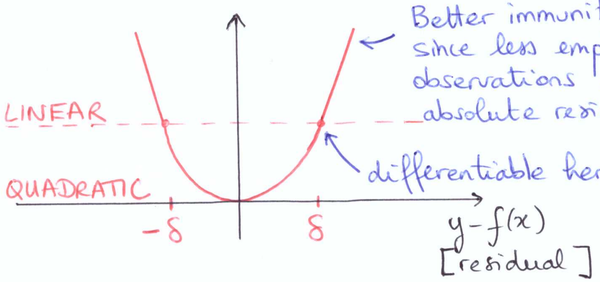
Gradient boosting algorithms are procedures minimizing general differentiable loss functions (other than the exponential loss in the context of classification).

Gradient → the gradient of the loss function is computed & implemented using a gradient descent-type algorithm

Boosting → the final (soft) classifier is expressed using a stagewise additive expansion.

Remark: In regression problems, the HUBER loss is a good alternative to the square loss (sensitive to outliers) and to the absolute loss (non-differentiable).

$$\ell(y, f(x)) = \begin{cases} (y-f(x))^2 & \text{if } |y-f(x)|<\delta \\ 2\delta|y-f(x)| - \delta^2 & \text{otherwise} \end{cases}$$



Better immunity to outliers since less emphasis on observations with large absolute residuals

differentiable here (and everywhere)

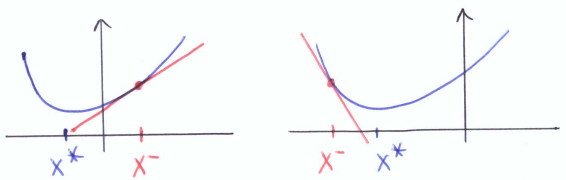
- **Gradient descent**: relatively cheap & it works (even if better descent algorithms exist).

  Goal: find the minimizer x^* of a function $\phi(x)$; $x \in \mathbb{R}^n$.

  Idea: move along the opposite direction of the gradient.

  $\Rightarrow$ Recursive procedure.

Indeed,



gradient ≥ 0
$x^+ = x^- - \varrho\,(\text{positive number})$
$\Rightarrow$ decrease the value of your current approximation towards x^*

gradient ≤ 0
$x^+ = x^- - \varrho\,(\text{neg } \#)$
$\Rightarrow$ increase the value of x^- towards x^*

$\varrho > 0$

$\Rightarrow$ Update is $x^+ = x^- - \varrho \nabla_x \phi(x)\big|_{x=x^-}$

learning rate

$\rightarrow$ Convergence towards a local minimum.

$\rightarrow$ ϱ is updated at each iteration via a line search:

$$\varrho^- = \underset{\varrho}{\text{argmin}}\ \phi\left(x^- - \varrho \nabla_x \phi(x)\big|_{x=x^-}\right)$$

Search in a particular direction $\Rightarrow$ it should be OK in many cases to perform the minimization.

- **Gradient boosting for regression.**

  Consider a differentiable loss function ℓ. Empirical Risk Minimization is the task of selecting a function f in a class of candidates $\mathcal{F}$ minimizing the empirical risk

$$\boxed{\sum_{i=1}^{n} \ell(y_i, f(x_i))}.$$

Forget for now that f belongs to some $\mathcal{F}$. We incorporate this constraint later

Consider the vector $\begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} \in \mathbb{R}^n$. Gradient boosting algorithms compute a vector in $\mathbb{R}^n$ minimizing the function $\phi(z) := \sum_{i=1}^{n} \ell(y_i, z_i)$, $z = (z_1, \dots, z_n)^t \in \mathbb{R}^n$

$\hookrightarrow$ Use Gradient Descent to minimize ϕ:

Given the current estimate $z^{(\ell-1)}$, compute the gradient of ϕ evaluated at $z^{(\ell-1)}$.

$$z^{(\ell-1)} = \begin{pmatrix} z_1^{(\ell-1)} \\ \vdots \\ z_n^{(\ell-1)} \end{pmatrix}$$

$$\nabla\phi(z) = \left(\frac{\partial \ell(y_1, z_1)}{\partial z_1}, \dots, \frac{\partial \ell(y_n, z_n)}{\partial z_n}\right)$$

$$\Rightarrow z^{(\ell)} = z^{(\ell-1)} - \varrho_{\ell-1} \nabla\phi(z)\big|_{z=z^{(\ell-1)}}$$

where $\rho_{\ell-1} = \underset{\rho}{\operatorname{argmin}}\ \phi\left(z^{(\ell-1)} - \rho \nabla\phi(z)\big|_{z=z^{(\ell-1)}}\right)$.

$z^{(\ell-1)}$ represents our current estimate of $(y_1, \dots, y_n)^t \in \mathbb{R}^n$

Denote it $z^{(\ell-1)} = \left(f^{(\ell-1)}(x_1), \dots, f^{(\ell-1)}(x_n)\right)^t$

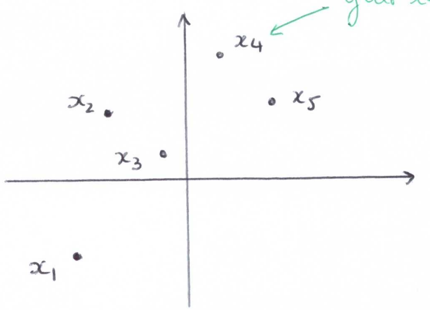
$\left(= (z_1^{(\ell-1)}, \dots, z_n^{(\ell-1)})^t\right)$

This approach is however not satisfying; since we cannot generalize the prediction to unseen points x. The main idea of gradient boosting is that before updating the current estimate, we generalize the value of the gradient for any $x \in \mathbb{R}^d$:

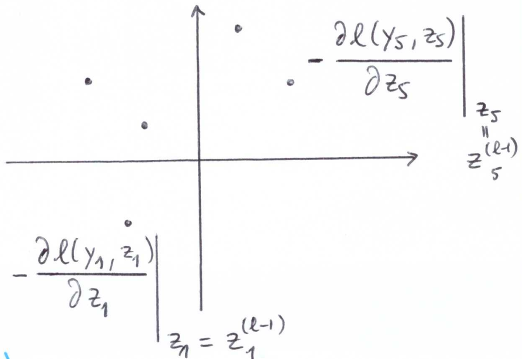
and prevents us from overfitting.

Ex: in $\mathbb{R}^2$

your learning sample



At iteration ℓ, the gradient of ϕ is computed at the training point $x_1, \dots, x_5$



generalize the value of $-\frac{\partial \ell(y_i, z_i)}{\partial z_i}\Big|_{z_i = z_i^{(\ell-1)}}$ to all x by viewing the task as a regression problem on $\left(x_i, -\frac{\partial \ell(y_i, z_i)}{\partial z_i}\right)_{z_i = z_i^{(\ell-1)}}$

Let $h(x;\theta)$ = regression function. The parameter θ is estimated using the training sample

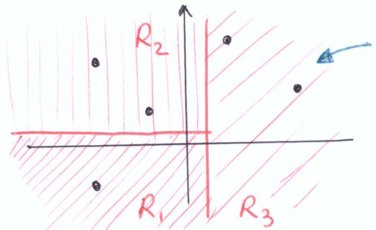
$$\left\{\left(x_i, -\frac{\partial \ell(y_i, z_i)}{\partial z_i}\Big|_{z_i = z_i^{(\ell-1)}}\right)\right\}_{i=1,\dots,n}.$$

Ex: for a regression tree, θ includes the split variables, the split points and the predicted value in each terminal region.

Typically, this regression problem is considered with a square loss function:

$$\boxed{\theta_\ell = \underset{\theta}{\operatorname{argmin}} \left(-\frac{\partial \ell(y_i, z_i)}{\partial z_i}\Big|_{z_i = z_i^{(\ell-1)}} - h(x_i, \theta)\right)^2}$$

Back to our example in $\mathbb{R}^2$, you may want to fit a decision tree with, say, three terminal nodes:



You have generalized the value of the gradient at the points $x_1, \dots, x_n$, to all points in $\mathbb{R}^2$.

Once the gradient is estimated everywhere, it remains to perform a line search:

$$\rho_\ell = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^n \ell\left(y_i, f^{(\ell-1)}(x_i) + \rho h(x_i; \theta_\ell)\right)$$

and update

$$f^{(\ell)}(x) = f^{(\ell-1)}(x) + \rho_\ell h(x; \theta_\ell).$$

Continue until convergence.

additive expansion

$\Rightarrow$ After M steps: $f^{(M)}(x) = \sum_{\ell=1}^M \rho_\ell h(x; \theta_\ell)$ BOOSTING!

weak learners

Gradient-Boost algorithm (Friedman 99)

1. Put $f^{(0)}(x) = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(y_i, \rho)$
2. For $\ell = 1$ to M, do
3. • Compute the pseudo-response $\tilde{y}_{i\ell}$:
$$\tilde{y}_{i\ell} = -\left[\frac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i) = f^{(\ell-1)}(x_i)}$$
4. • $\theta_\ell = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\tilde{y}_{i\ell} - h(x_i;\theta)\right)^2$
5. • $\rho_\ell = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(y_i, f^{(\ell-1)}(x_i) + \rho\, h(x_i;\theta_\ell)\right)$
6. • $f^{(\ell)}(x) = f^{(\ell-1)}(x) + \rho_\ell\, h(x;\theta_\ell)$
7. End for
8. End Algorithm

Examples: (i) Square loss $\ell(y, f(x)) = \frac{1}{2}(y - f(x))^2$

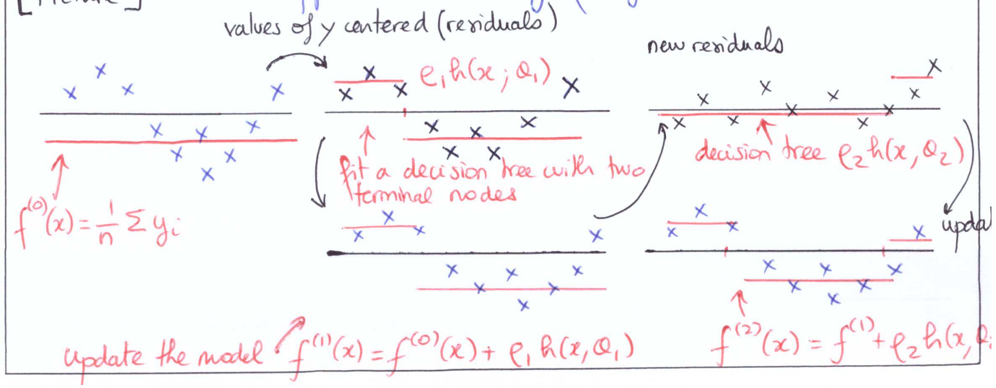
The pseudo response is $\tilde{y}_{i\ell} = y_i - f^{(\ell-1)}(x_i)$

$= \text{residuals}$

$= \text{negative gradient}$

Gradient boosting with a square loss $\equiv$ stagewise approach iteratively fitting the current residuals.

[Picture]



(ii) Absolute loss $\ell(y, f(x)) = |y - f(x)|$

The pseudo response is $\tilde{y}_{i\ell} = \operatorname{sign}\left(y_i - f^{(\ell-1)}(x_i)\right)$.

$\Rightarrow$ The regression function is fit to the sign of the residuals in line 4 of the algorithm.

Then, line 5 becomes

$$\rho_\ell = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} \left| y_i - f^{(\ell-1)}(x_i) - \rho\, h(x_i,\theta_\ell)\right|$$

$$= \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} |h(x_i,\theta_\ell)| \left| \frac{y_i - f^{(\ell-1)}(x_i)}{h(x_i,\theta_\ell)} - \rho \right|$$

$$= \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \left| z_i - \rho \right|$$

$\rho_\ell = \text{median}_W \{z_1, \dots, z_n\}$

$=$ weighted median of $\{z_1, \dots, z_n\}$ with weights $w_1, \dots, w_n$.

(iii) Huber loss $\ell(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & |y-f| \leq \delta \\ \delta\left(|y-f| - \frac{\delta}{2}\right) & \text{———} > \delta \end{cases}$

The pseudo response is

$$\tilde{y}_{i\ell} = \begin{cases} y_i - f^{(\ell-1)}(x_i) & \text{if } |y_i - f^{(\ell-1)}(x_i)| \leq \delta \\ \delta \operatorname{sign}\left(y_i - f^{(\ell-1)}(x_i)\right) & \text{if } \text{———} > \delta \end{cases}$$

RK = δ may be allowed to vary with the iteration number:

$\delta_\ell = \alpha\text{-quantile of } \{y_i - f^{(\ell-1)}(x_i)\}_{i=1,\dots}$

$=$ break-down point

and

$$\rho_\ell = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(y_i, f^{(\ell-1)}(x_i) + \rho\, h(x;\theta_\ell)\right)$$

solve numerically; see Huber (1964)

III. 2. Gradient boosting for trees

In the case where the (weak) learners $h(x;\theta)$ are decision trees, the Gradient-Boost algorithm takes a simple form for specific loss functions.

Line 4 → parameter θ_ℓ corresponds to the terminal regions

Line 5 → parameter ρ_ℓ corresponds to the predicted value in each terminal region.

Specifically,

Gradient Tree-Boosting Algorithm

1. Put $f^{(0)}(x) = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(y_i, \rho)$
2. For $\ell = 1$ to M, do
3. $\cdot\ \tilde{y}_{i\ell} = -\left[\dfrac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f^{(\ell-1)}(x_i)}$
4. $\cdot$ Compute the J terminal regions $\{R_{j\ell}\}_{j=1}^{J}$ = J-terminal tree fitted to $(x_i, \tilde{y}_{i\ell})$.
5. $\cdot$ Compute $\gamma_{j\ell} = \operatorname{argmin} \sum_{x_i \in R_{j\ell}} \ell(y_i, f^{(\ell-1)}(x_i) + \gamma)$
6. $\cdot$ Update $f^{(\ell)}(x) = f^{(\ell-1)}(x) + \sum_{j=1}^{J} \gamma_{j\ell}\, \mathbb{1}(x \in R_{j\ell})$
7. End For
8. End Algorithm

Ex: With an absolute loss, line 5 becomes:

$$\gamma_{j\ell} = \underset{x_i \in R_{j\ell}}{\operatorname{median}} \left\{ y_i - f^{(\ell-1)}(x_i) \right\}.$$

III. 3. Gradient boosting for classification.

- Context: K-class classification
- Coding scheme: $Y_i = (Y_{i1}, \dots, Y_{iK})$, where $Y_{ik} = 1$ if observation X_i belongs to class k, and $Y_{ik} = 0$ otherwise.

Put
$$p_k(x_i) = \mathbb{P}(Y_{ik} = 1 \mid X = x_i) = \frac{\exp f_k(x_i)}{\sum_{j=1}^{K} \exp f_j(x_i)}, \quad k = 1, \dots, K$$

softmax representation

adding a constant value to all the f_j does not change the value of p_k ⇒ impose that $\sum_{j=1}^{K} f_j = 0$

- Goal: Estimation of $(f_1, \dots, f_K) =: f$
- loss: the multinomial deviance (for n obs) [minus the log lik]

$$\ell(y, f(x)) = -\sum_{i=1}^{n}\sum_{k=1}^{K} Y_{ik} \log p_k(X_i)$$
$$= -\sum_{i,k} Y_{ik} f_k(X_i) - \sum_{i=1}^{n} \underbrace{\sum_{k=1}^{K} Y_{ik}}_{=1} \log\left[\sum_{j=1}^{K} f_j(X_i)\right]$$
$$= -\sum_{i,k} Y_{ik} f_k(X_i) - \sum_{i} \log\left(\sum_{j} f_j(X_i)\right)$$

- Pseudo-residuals

$$\tilde{y}_{ik\ell} = \frac{\partial}{\partial f_k(x_i)}\left[-\ell(y_i, f_k(x_i))\right]\Big|_{f_k(x_i) = f_k^{(\ell-1)}(x_i)}$$
$$= y_{ik} - p_{k,\ell-1}(x_i),$$

obs i, class k, iteration ℓ

where $p_{k,\ell-1}(x_i) := \dfrac{\exp f_k^{(\ell-1)}(x_i)}{\sum_{j=1}^{K} \exp f_j^{(\ell-1)}(x_i)}$.

Gradient Boosting for K-class classification

1. Initialise $f_k^{(0)}(x)$ for $k=1,\dots,K$
2. For $\ell=1$ to M, repeat
3. For $k=1$ to K, repeat
4. • Compute the pseudo response

$$\tilde{y}_{ik\ell} = y_{ik} - p_{k,\ell-1}(x_i), \quad i=1,\dots,n$$

where $p_{k,\ell-1}(x_i) = \dfrac{\exp f_k^{(\ell-1)}(x_i)}{\sum_{j=1}^{K} \exp f_j^{(\ell-1)}(x_i)}$

5. • Fit a decision tree $T_{k\ell}$ to $\{(x_1, \tilde{y}_{1k\ell}), \dots, (x_n, \tilde{y}_{nk\ell})\}$, producing regions $\{R_{jk\ell}\}_{j=1}^{J}$
6. • Compute

$$\{\gamma_{jk\ell}\} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jk\ell}} \ell(y_i, f_k^{(\ell-1)}(x_i) + \gamma)$$

7. • Update

$$f_k^{(\ell)}(x) = f_k^{(\ell-1)}(x) + \sum_{j=1}^{J} \gamma_{jk\ell} \, \mathbb{1}(x \in R_{jk\ell}).$$

8. End For
9. End For
10. Compute the posterior probabilities $\hat{p}_k(x) = \dfrac{\exp f_k^{(M)}(x)}{\sum_{j=1}^{K} \exp f_j^{(M)}(x)}$,

and classify a new observation x according to $\underset{1\leq k\leq K}{\operatorname{argmax}} \, \hat{p}_k(x)$.

Remark: Computing the values $\gamma_{jk\ell}$ in line 6 is not trivial, and can be approximately done using a one-step Newton-Raphson update using only the diagonal of the Hessian matrix of ℓ, see (Friedman 99)

References

Freund Y and Schapire R (1996). Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference, 148-156.

Zhu J, Zou H, Rosset S, and Hastie T (2009). Multi-class AdaBoost. Statistics and its Interface. Vol 2, p. 349-360.

Friedman J (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, Vol 29, No 5, 1189-1232.