Boosting

MATH-412 - Statistical Machine Learning

Boosting

"Weak classifiers"
Assume that, in the context of binary classification, we have a collection of weak classifiers

$$\{\, \mathbf{x} \mapsto G(\mathbf{x}, \gamma) \;\mid\; \gamma \in \Gamma \,\}$$

achieving classification error < 0.5 are available.

General principle :
Construct iteratively an additive model of the form

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})$$

by adding one term G_m at a time.

Example of weak classifier : the **stump** $G(x, \gamma) = 1_{\{\langle x, \gamma \rangle \geq 1\}}$, or **trees**.

Adaboost *(adaptive boosting)*

Given a training set

$$D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

and a collection of weights

$$(w_1, \ldots, w_n)$$

associated with the training points, we define the

Reweighted misclassification error :

$$\widehat{\mathcal{R}}^{\text{0-1}}_{w,n}(G) = \frac{1}{W} \sum_{i=1}^{n} w_i 1_{\{y_i \neq G(\mathbf{x}_i)\}} \qquad \text{with} \qquad W = \textstyle\sum_{i=1}^{n} w_i.$$

Adaboost algorithm

$\forall i, \quad \text{set} \quad w_i^{(0)} = \frac{1}{n}.$

for $m = 1, \ldots, M$

$$\begin{aligned} \mathsf{err}_m(G) \quad &:= \quad \frac{1}{W^{(m-1)}} \sum_{i=1}^{n} w_i^{(m-1)} \, 1_{\{y_i \neq G(\mathbf{x}_i)\}} \\ G_m \quad &\leftarrow \quad \mathsf{arg}\min_G \mathsf{err}_m(G) \\ \alpha_m \quad &\leftarrow \quad \log\left(\frac{1 - \mathsf{err}_m(G_m)}{\mathsf{err}_m(G_m)}\right) \\ w_i^{(m)} \quad &\leftarrow \quad w_i^{(m-1)} \, . \, \exp\left(\alpha_m \, 1_{\{y_i \neq G_m(\mathbf{x}_i)\}}\right) \end{aligned}$$

end for

$$f(\mathbf{x}) = \mathsf{sign}\Big[\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})\Big].$$

Interpretations of Adaboost

- Solves a sequence of reweighted misclassification error minimization
- At each iteration concentrates on the harder examples
- Includes each weak classifier with a coefficient which is a decreasing function of the current reweighted error.

Forward stagewise additive modelling

Set $f_0(x) = 0$

for $m = 1$ to M,

- Solve $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} \ell\big(y_i, f_{m-1}(x_i) + \beta G(x, \gamma)\big)$
- Set $f_m(x) = f_{m-1}(x) + \beta_m G(x_i, \gamma_m)$.

end for

Particular case of the square loss (then aka Matching Pursuit)

$$\min_{\beta,\gamma} \quad \sum_{i=1}^{n} \Big(y_i - f_m(x_i) - \beta G(x_i, \gamma)\Big)^2$$

reduces to
$$\min_{\beta,\gamma} \quad \sum_{i=1}^{n} \Big(res_{im} - \beta G(x_i, \gamma)\Big)^2$$

for $res_{im} = y_i - f_m(x_i)$ the previous residuals.

Deriving boosting from the exponential loss

Assuming $y \in \{-1, 1\}$ and binary weak predictors with $G(x) \in \{-1, 1\}$, Adaboost arises as an instance of forward additive modeling for the exponential loss $\ell : (a, y) \mapsto \exp(-ay)$.

$$\begin{aligned}
(\beta_m, G_m) &= \operatorname*{arg\,min}_{\beta, G} \sum_{i=1}^{n} \ell\big(y_i, f_{m-1}(\mathbf{x}_i) + \beta G(\mathbf{x}_i)\big) \\
&= \operatorname*{arg\,min}_{\beta, G} \sum_{i=1}^{n} \exp\Big(-y_i f_{m-1}(\mathbf{x}_i) - y_i \beta G(\mathbf{x}_i)\Big) \\
&= \operatorname*{arg\,min}_{\beta, G} \sum_{i=1}^{n} w_i^{(m-1)} \exp\Big(-y_i \beta G(\mathbf{x}_i)\Big)
\end{aligned}$$

with $w_i^{(m-1)} := \exp\big(-y_i f_{m-1}(\mathbf{x}_i)\big)$.

From exponential loss to weighted misclassification error

For $G : \mathcal{X} \to \{-1, 1\}$, we have

$$\begin{aligned}
\mathcal{R}_{\exp}(\beta, G) &= \sum_{i=1}^{n} w_i \, e^{-y_i \beta G(\mathbf{x}_i)} \\
&= \sum_{i: y_i = G(\mathbf{x}_i)} w_i \, e^{-\beta} + \sum_{i: y_i \neq G(\mathbf{x}_i)} w_i \, e^{\beta} \\
&= \sum_{i=1}^{n} w_i \, e^{-\beta} + \sum_{i: y_i \neq G(\mathbf{x}_i)} w_i \left(e^{\beta} - e^{-\beta}\right) \\
&= W \, e^{-\beta} + W \left(e^{\beta} - e^{-\beta}\right) \sum_{i=1}^{n} \frac{w_i}{W} 1_{\{y_i \neq G(\mathbf{x}_i)\}} \\
&= W \left[e^{-\beta} + \left(e^{\beta} - e^{-\beta}\right) \widehat{\mathcal{R}}^{\text{0-1}}_{w,n}(G) \right],
\end{aligned}$$

with $W = \sum_{i=1}^{n} w_i$.

Solutions for β and G

$$\mathcal{R}_{\text{exp}}(\beta, G) = W\left[\, e^{-\beta} + \left(e^{\beta} - e^{-\beta}\right)\, \widehat{\mathcal{R}}^{\textsf{0-1}}_{w,n}(G)\,\right].$$

Minimizing $\mathcal{R}_{\text{exp}}(\beta, G)$

- w.r.t. G

$$G_m \quad \leftarrow \quad \text{arg}\min_{G} \widehat{\mathcal{R}}^{\textsf{0-1}}_{w,n}(G)$$

- w.r.t. β :

$$\beta_m \quad \leftarrow \quad \frac{1}{2}\log\left(\frac{1 - \widehat{\mathcal{R}}^{\textsf{0-1}}_{w,n}(G_m)}{\widehat{\mathcal{R}}^{\textsf{0-1}}_{w,n}(G_m)}\right) = \frac{1}{2}\alpha_m$$

- The weights are updated to

$$\begin{aligned} w_i^{(m)} &= w_i^{(m-1)} \exp(-y_i \beta_m G_m(\mathbf{x}_i)) \\ &= w_i^{(m-1)} \exp(-\beta_m) \exp(2\beta_m 1_{\{y_i \neq G_m(\mathbf{x}_i)\}}) \end{aligned}$$

We retrieve the updates of Adaboost with $\alpha_m = 2\beta_m$ (since weights are normalised).

Comparing with Adaboost

$\forall i, \quad \text{set} \quad w_i^{(0)} = \frac{1}{n}.$

for $m = 1, \ldots, M$

$$\text{err}_m(G) \quad := \quad \sum_{i=1}^{n} w_i^{(m-1)} \, 1_{\{y_i \neq G(\mathbf{x}_i)\}}$$

$$G_m \quad \leftarrow \quad \arg\min_G \text{err}_m(G)$$

$$\alpha_m \quad \leftarrow \quad \log\left(\frac{1-\text{err}_m(G_m)}{\text{err}_m(G_m)}\right)$$

$$w_i^{(m)} \quad \leftarrow \quad w_i^{(m-1)} \, . \, \exp\left(\alpha_m \, 1_{\{y_i \neq G_m(\mathbf{x}_i)\}}\right)$$

end for

$$f(\mathbf{x}) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})\right].$$

Boosting of Trees

General gradient tree boosting

- For classification, it makes sense to use the logit loss.
- For a general loss ℓ, an optimal choice of tree is difficult.
- The direction that decreases most the empirical risk is the one opposite to the gradient whose components are

$$r_{im} = -\frac{\partial \ell(y_i, a)}{\partial a}\Big|_{a=f_{(m-1)}(\mathbf{x}_i)}.$$

$\rightarrow$ Find the tree which approximates best r_{im} in the least-square sense.

General gradient tree boosting

1. $f_0(x) = \text{argmin}_\gamma \sum_{i=1}^n \ell(y_i, \gamma)$. Set $m = 1$.
2. For $i = 1, \ldots, n$, $\quad r_{im} = -\frac{\partial \ell(y_i, a)}{\partial a}\Big|_{a=f_{(m-1)}(\mathbf{x}_i)}$.
3. Fit a regression tree with maximal depth d for the least-square loss
$$\min_{\gamma,R} \sum_{i=1}^n (r_{im} - G_m(\mathbf{x}_i))^2 \quad \text{with} \quad G_m(\mathbf{x}) = \sum_{j=1}^d \gamma_{jm} 1_{\{\mathbf{x}\in R_{jm}\}}.$$
4. Throw away γ and only keep $\{R_{jm}\}_{1\le j\le d}$.
5. For $j = 1, \ldots, d$, $\quad \gamma_{jm} = \text{argmin}_{\gamma'} \sum_{i:\mathbf{x}_i\in R_{jm}} \ell(y_i, f_{m-1}(\mathbf{x}_i) + \gamma')$.
6. For some $\nu \in [0, 1]$, set $\quad f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu \sum_{j=1}^d \gamma_{jm} 1_{\{\mathbf{x}\in R_{jm}\}}$.
7. Increase m and go back to (2).

d and ν can be chosen by cross-validation.

Gradient tree boosting

Easy to use and efficient implementations :

- XGBoost
- LightGBM