Supervised Learning :: CHEAT SHEET

AdaBoost

Boosting is a homogeneous ensemble method that takes a weak classifier and sequentially apply it to modified versions of the training data.

Algorithm 1 AdaBoost $\mathcal{Y} = \{-1, +1\}$

1: Initialize observation weights: $w^{[1](i)} = \frac{1}{n} \quad \forall i \in \{1, \ldots, n\}$
2: **for** $m = 1 \to M$ **do**
3: Fit classifier to training data with weights $w^{[m]}$ and get $\hat{b}^{[m]}$
4: Calculate weighted in-sample misclassification rate

$$\text{err}^{[m]} = \sum_{i=1}^{n} w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}}$$

5: Compute: $\hat{\beta}^{[m]} = \frac{1}{2} \log\left(\frac{1-\text{err}^{[m]}}{\text{err}^{[m]}}\right)$
6: Set: $w^{[m+1](i)} = w^{[m](i)} \cdot \exp\left(-\hat{\beta}^{[m]} \cdot y^{(i)} \cdot \hat{b}^{[m]}(\mathbf{x}^{(i)})\right)$
7: Normalize $w^{[m+1](i)}$ such that $\sum_{i=1}^{n} w^{[m+1](i)} = 1$
8: **end for**
9: Output: $\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} \hat{\beta}^{[m]} \hat{b}^{[m]}(\mathbf{x})$

	Random Forest	AdaBoost
Base Learners	typically deeper decision trees	weak learners, e.g. only stumps
Weights	equal	different, depending on predictive accuracy
Structure	independent BLs	sequential, order matter
Aim	variance reduction	bias and variance reduction
Overfit	tends not to	tends to

Gradient Boosting

We want to learn an additive model: $f(\mathbf{x}) = \sum_{m=1}^{M} \alpha^{[m]} b(\mathbf{x}, \boldsymbol{\theta}^{[m]})$
Hence, we minimize the empirical risk:

$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right) = \sum_{i=1}^{n} L\left(y^{(i)}, \sum_{m=1}^{M} \alpha^{[m]} b(\mathbf{x}^{(i)}, \boldsymbol{\theta}^{[m]})\right)$$

And add additive components in a greedy fashion by sequentially minimizing the risk only w.r.t. the next additive component:

$$\min_{\alpha, \boldsymbol{\theta}} \sum_{i=1}^{n} L\left(y^{(i)}, \hat{f}^{[m-1]}\left(\mathbf{x}^{(i)}\right) + \alpha b\left(\mathbf{x}^{(i)}, \boldsymbol{\theta}\right)\right)$$

Doing this iteratively is called **forward stagewise additive modeling**.

Algorithm 2 Gradient Boosting Algorithm

1: Initialize $\hat{f}^{[0]}(\mathbf{x}) = \arg\min_{\theta} \sum_{i=1}^{n} L(y^{(i)}, b(\mathbf{x}^{(i)}, \boldsymbol{\theta}))$
2: **for** $m = 1 \to M$ **do**
3: For all i: $\tilde{r}^{[m](i)} = -\left[\frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f(\mathbf{x}^{(i)})}\right]_{f=\hat{f}^{[m-1]}}$
4: Fit a regression base learner to the pseudo-residuals $\tilde{r}^{[m](i)}$:
5: $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (\tilde{r}^{[m](i)} - b(\mathbf{x}^{(i)}, \boldsymbol{\theta}))^2$
6: Set $\beta^{[m]}$ to β being a small constant value or via line search
7: Update $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \beta^{[m]} b(\mathbf{x}, \hat{\boldsymbol{\theta}}^{[m]})$
8: **end for**
9: Output $\hat{f}(\mathbf{x}) = \hat{f}^{[M]}(\mathbf{x})$

Gradient Boosting with Trees

Tree can be seen as additive model: $b(\mathbf{x}) = \sum_{t=1}^{T} c_t \mathbb{1}_{\{\mathbf{x} \in R_t\}}$, R_t are the terminal regions, c_t are terminal constants.
GB with trees is still additive: $f^{[m]}(\mathbf{x}) = f^{[m-1]}(\mathbf{x}) + \alpha^{[m]} b^{[m]}(\mathbf{x}) = f^{[m-1]}(\mathbf{x}) + \alpha^{[m]} \sum_{t=1}^{T^{[m]}} c_t^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_t^{[m]}\}} = f^{[m-1]}(\mathbf{x}) + \sum_{t=1}^{T^{[m]}} \tilde{c}_t^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_t^{[m]}\}}$
With $\tilde{c}_t^{[m]} = \alpha^{[m]} \cdot c_t^{[m]} = \arg\min_c \sum_{\mathbf{x}^{(i)} \in R_t^{[m]}} L(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + c)$.

Algorithm 3 Gradient Boosting for g-class Classification.

1: Initialize $f_k^{[0]}(\mathbf{x}) = 0, \quad k = 1, \ldots, g$
2: **for** $m = 1 \to M$ **do**
3: Set $\pi_k(\mathbf{x}) = \frac{\exp(f_k^{[m]}(\mathbf{x}))}{\sum_j \exp(f_j^{[m]}(\mathbf{x}))}, k = 1, \ldots, g$
4: **for** $k = 1 \to g$ **do**
5: For all i: Compute $\tilde{r}_k^{[m](i)} = \mathbb{1}_{\{y^{(i)}=k\}} - \pi_k(\mathbf{x}^{(i)})$
6: Fit regr. tree to the $\tilde{r}_k^{[m](i)}$ giving terminal regions $R_{tk}^{[m]}$
7: Compute
8: $\hat{c}_{tk}^{[m]} = \frac{g-1}{g} \frac{\sum_{\mathbf{x}^{(i)} \in R_{tk}^{[m]}} \tilde{r}_k^{[m](i)}}{\sum_{\mathbf{x}^{(i)} \in R_{tk}^{[m]}} |\tilde{r}_k^{[m](i)}| \left(1 - |\tilde{r}_k^{[m](i)}|\right)}$
9: Update $\hat{f}_k^{[m]}(\mathbf{x}) = \hat{f}_k^{[m-1]}(\mathbf{x}) + \sum_t \hat{c}_{tk}^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_{tk}^{[m]}\}}$
10: **end for**
11: **end for**
12: Output $\hat{f}_1^{[M]}, \ldots, \hat{f}_g^{[M]}$

XGBoost

XGBoost uses stochastic GB for data subsampling with regularization:

$$\mathcal{R}_{\text{reg}}^{[m]} = \sum_{i=1}^{n} L(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + b^{[m]}(\mathbf{x}^{(i)})) + \lambda_1 J_1(b^{[m]}) + \lambda_2 J_2(b^{[m]}) + \lambda_3 J_3(b^{[m]})$$

- $J_1(b^{[m]}) = T^{[m]}$: Nr of leaves to penalize tree depth
- $J_2(b^{[m]}) = \left\|\mathbf{c}^{[m]}\right\|_2^2$: L2 penalty over leaf values
- $J_3(b^{[m]}) = \left\|\mathbf{c}^{[m]}\right\|_1$: L1 penalty over leaf values

Component Wise Boosting

For CWB we generalize to multiple base learner sets $\{\mathcal{B}_1, \ldots \mathcal{B}_J\}$ with associated parameter spaces $\{\boldsymbol{\Theta}_1, \ldots \boldsymbol{\Theta}_J\}$.

Algorithm 4 Componentwise Gradient Boosting.

1: Initialize $f^{[0]}(\mathbf{x}) = \arg\min_{\theta_0 \in \mathbb{R}} \sum_{i=1}^{n} L(y^{(i)}, \theta_0)$
2: **for** $m = 1 \to M$ **do**
3: For all i: $\tilde{r}^{[m](i)} = -\left[\frac{\partial L(y,f)}{\partial f}\right]_{f=f^{[m-1]}(\mathbf{x}^{(i)}), y=y^{(i)}}$
4: **for** $j = 1 \to J$ **do**
5: Fit regression base learner $b_j \in \mathcal{B}_j$ to the vector of pseudo-residuals $\tilde{r}^{[m]}$:
6: $\hat{\boldsymbol{\theta}}_j^{[m]} = \arg\min_{\boldsymbol{\theta} \in \Theta_j} \sum_{i=1}^{n} (\tilde{r}^{[m](i)} - b_j(\mathbf{x}^{(i)}, \boldsymbol{\theta}))^2$
7: **end for**
8: $j^{[m]} = \arg\min_j \sum_{i=1}^{n} (\tilde{r}^{[m](i)} - \hat{b}_j(\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}_j^{[m]}))^2$
9: Update $f^{[m]}(\mathbf{x}) = f^{[m-1]}(\mathbf{x}) + \alpha \hat{b}_j(\mathbf{x}, \hat{\boldsymbol{\theta}}_{j^{[m]}}^{[m]})$
10: **end for**
11: Output $\hat{f}(\mathbf{x}) = f^{[M]}(\mathbf{x})$

Handling Categorical Features in CWB

- One base learner to simultaneously estimate all categories:

$$b_j(x_j|\boldsymbol{\theta}_j) = \sum_{g=1}^{G} \theta_{j,g} \mathbb{1}_{\{g=x_j\}} = (\mathbb{1}_{\{x_j=1\}}, \ldots, \mathbb{1}_{\{x_j=G\}})\boldsymbol{\theta}_j$$

Hence, b_j incorporates a one-hot encoded feature with group means $\boldsymbol{\theta} \in \mathbb{R}^G$ as estimators.
- One binary base learner per category: $b_{j,g}(x_j|\theta_{j,g}) = \theta_{j,g} \mathbb{1}_{\{g=x_j\}}$
Including all categories of the feature means adding G base learners $b_{j,1}, \ldots, b_{j,G}$

Handling Intercept in CWB

Loss-optimal constant $f^{[0]}(\mathbf{x})$ as an initial model intercept.
- Include an intercept BL
- Add BL $b_{\text{int}} = \theta$ as potential candidate considered in each iteration and remove intercept from all linear BLs, i.e., $b_j(\mathbf{x}) = \theta_j x_j$. Final intercept is given as $f^{[0]}(\mathbf{x}) + \hat{\theta}$.