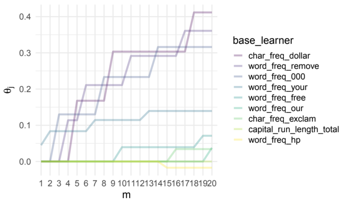
Introduction to Machine Learning

Componentwise Gradient Boosting



First 20 boosting iterations on Spam example

Learning goals

- Learn the concept of componentwise boosting and its relation to GLM
- Understand the built-in feature selection process
- Understand the problem of fair base learner selection

COMPONENTWISE GRADIENT BOOSTING

Gradient boosting, especially when using trees, has strong predictive performance but is difficult to interpret unless the base learners are stumps.

The aim of componentwise gradient boosting is to find a model that:

- has strong predictive performance,
- has components that are still interpretable,
- does automatic selection of components,
- is sparser than a model fitted with maximum-likelihood estimation.

This is achieved by using “nice” base learners which yield familiar statistical models in the end.

Because of this, componentwise gradient boosting is also often referred to as **model-based boosting**.

BASE LEARNERS

In classical gradient boosting only one kind of base learner is used, e.g., regression trees.

For componentwise gradient boosting we generalize this to multiple base learners

$$\{b_j^{[m]}(\mathbf{x}, \theta^{[m]}) : j = 1, 2, \dots, J\},$$

where j indexes the type of base learner.

Again, in each iteration, only the best base learner $b_{j^*}^{[m]}(\mathbf{x}, \theta^{[m]})$ is selected and updated.

BASE LEARNERS

We restrict these base learners to additive models, i.e.,

$$b_j^{[m]}(\mathbf{x}, \boldsymbol{\theta}^{[m]}) + b_j^{[m']}(\mathbf{x}, \boldsymbol{\theta}^{[m']}) = b_j(\mathbf{x}, \boldsymbol{\theta}^{[m]} + \boldsymbol{\theta}^{[m']}).$$

Often base learners are not defined on the entire feature vector **x** but on a single feature x_j:

$$b_j^{[m]}(x_j, \boldsymbol{\theta}^{[m]}) \quad \text{for } j = 1, 2, \ldots, p.$$

This directly incorporates a variable selection mechanism into the fitting process, since in each iteration only the best base learner is selected in combination with the associated feature, and each base learner can be (substantially) more complex than a stump (e.g., univariate linear effects or splines).

COMPONENTWISE BOOSTING ALGORITHM

Algorithm Componentwise Gradient Boosting.

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

RELATION TO GLM

In the simplest case we use linear models (without intercept) on single features as base learners:

$$b(x_j, \boldsymbol{\theta}^{[m]}) = x_j \theta^{[m]} \quad \text{for } j = 1, 2, \ldots, p.$$

This definition will result in an ordinary **linear regression** model (note that linear base learners without intercept only make sense for covariates that have been centered before).

- If we let the boosting algorithm converge, i.e., let it run for a really long time, the parameters estimated by the boosting model will converge to the **same solution** as the ML estimate.
- This means that, by specifying a loss function according to the negative likelihood of a distribution from an exponential family and defining a link function accordingly, this kind of boosting is equivalent to a (regularized) **generalized linear model (GLM)**.

RELATION TO GLM

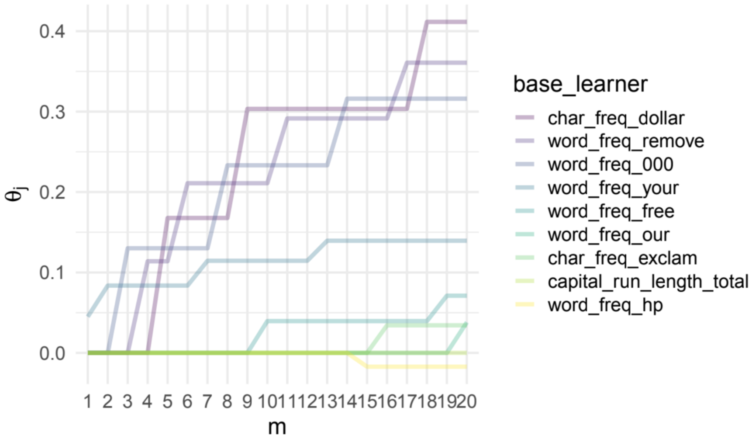
But: We do not *need* an exponential family and thus are able to fit models to all kinds of other distributions and losses, as long as we can calculate (or approximate) a derivative of the loss.

Usually we do not let the boosting model converge fully, but **stop early** for the sake of regularization and feature selection.

Even though the resulting model looks like a GLM, we do not have valid standard errors for our coefficients, so cannot provide confidence or prediction intervals or perform tests etc. $\rightarrow$ post-selection inference.

FEATURE SELECTION

We can visualize the updates of the θ_j together with the number of iterations to see which base learner was selected when.



First 20 boosting iterations on Spam example

FEATURE SELECTION

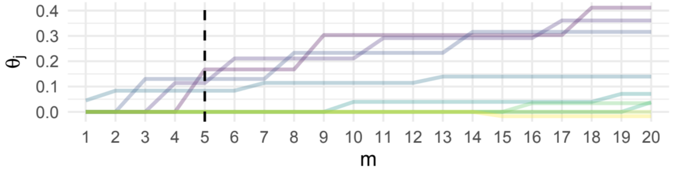
- We can further exploit the additive structure of the boosted ensemble to compute measures of **variable importance**.
- To this end, we simply sum for each feature x_j the improvements in empirical risk achieved over all iterations until $1 < m_{\text{stop}} \leq M$:

$$VI_j = \sum_{m=1}^{m_{\text{stop}}} \left(\mathcal{R}_{\text{emp}}\left(f^{[m-1]}(\mathbf{x})\right) - \mathcal{R}_{\text{emp}}\left(f^{[m]}(\mathbf{x})\right)\right) \cdot \mathbb{I}_{[j \in sel(m)]},$$

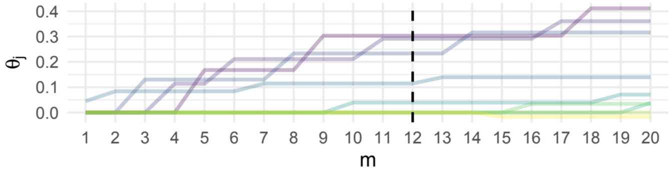
  where $sel(m)$ denotes the index set of features selected in the m-th iteration.

FEATURE SELECTION

The number of features effectively included in the final model depends on the value of M.
$\rightarrow$ A sparse logistic regression is fitted.



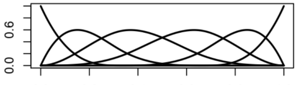
mstop = 5: Four selected base learners



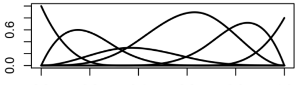
mstop = 12: Five selected base learners

NONLINEAR BASE LEARNERS

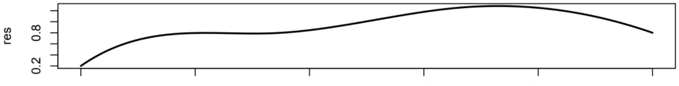
Nonlinear base learners like *P*- or *B*-splines make the model equivalent to a **generalized additive model (GAM)**, as long as the base learners keep their additive structure (which is the case for splines).



Cubic B−spline basis



Cubic B−spline basis weighted with coefficients



Sum of cubic B−splines weighted with coefficients

NONLINEAR BASE LEARNERS

Even when allowing for more complexity we typically want to keep solutions as simple as possible.

Kneib et al. (2009) proposed a decomposition of each base learner into a constant, a linear and a nonlinear part. The boosting algorithm will automatically decide which feature to include – linear, nonlinear, or none at all:

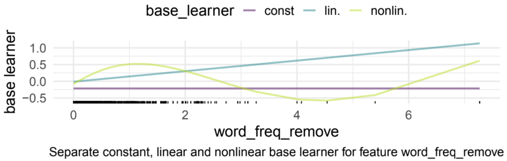
$$\begin{aligned} b_j(x_j, \boldsymbol{\theta}^{[m]}) &= b_{j,\text{const}}(x_j, \boldsymbol{\theta}^{[m]}) + b_{j,\text{lin}}(x_j, \boldsymbol{\theta}^{[m]}) + b_{j,\text{nonlin}}(x_j, \boldsymbol{\theta}^{[m]}) \\ &= \theta_{\text{const}}^{[m]} + x_j \cdot \theta_{j,\text{lin}}^{[m]} + s_j(x_j, \boldsymbol{\theta}_{j,\text{nonlin}}^{[m]}), \end{aligned}$$

where

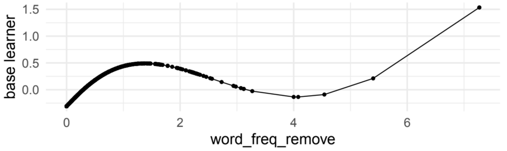
- θ_{const} is a constant term over all base learners,
- $x_j \cdot \theta_{j,\text{lin}}^{[m]}$ is a feature-specific linear base learner, and
- $s_j(x_j, \boldsymbol{\theta}_{j,\text{nonlin}}^{[m]})$ is a (centered) nonlinear base learner capturing deviation from the linear effect.

SPAM EXAMPLE: CONTINUED

- This time we fit a componentwise gradient boosting model to the spam dataset using the R package mboost with $M = 100$.
- We specify a linear and nonlinear (P-spline) base learner for each feature and let the boosting model decide which to select.
- The model selects 17 different base learners (out of 114):



Separate constant, linear and nonlinear base learner for feature word_freq_remove



Overall estimate for feature word_freq_remove

FAIR BASE LEARNER SELECTION

- Using splines and linear base learners in componentwise boosting will favor the more complex spline base learner over the linear base learner.
- This makes it harder to achieve the desired behavior of the base learner decomposition as explained previously.
- To conduct a fair base learner selection we set the degrees of freedom of all base learners equal.
- The idea is to set the regularization/penalty term of a single learner in a manner that their complexity is treated equally.

FAIR BASE LEARNER SELECTION

Especially for linear models and GAMs it is possible to transform the degrees of freedom into a corresponding penalty term.

- Parameters of the base learners are estimated via:

$$\theta_j^{[m]} = \left(\mathbf{Z}_j^T\mathbf{Z}_j + \lambda_j\mathbf{K}_j\right)^{-1}\mathbf{Z}_j^T\tilde{r}^{[m]} ,$$

  with $\mathbf{Z}_j$ the design matrix of the j-th base learner, λ_j the penalty term, and $\mathbf{K}_j$ the penalty matrix.

- Having that kind of model, we use the hat matrix $\mathbf{S}_j = \mathbf{Z}_j\left(\mathbf{Z}_j^T\mathbf{Z}_j + \lambda_j\mathbf{K}_j\right)^{-1}\mathbf{Z}_j^T$ to define the degrees of freedom:

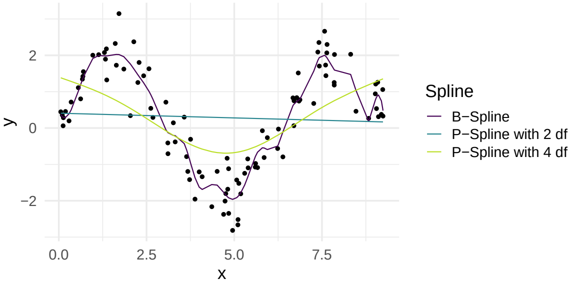
$$\text{df}(\lambda_j) = \text{tr}\left(2\mathbf{S}_j - \mathbf{S}_j^T\mathbf{S}_j\right) .$$

  Note: With $\lambda_j = 0$, $\mathbf{S}_j$ is the projection matrix into the target space with $\text{tr}(\mathbf{S}_j) = \text{rank}(\mathbf{X})$, which corresponds to the number of parameters in the model.

FAIR BASE LEARNER SELECTION

It is possible to calculate λ_j by applying the Demmler-Reinsch orthogonalization (see Hofer et al. (2011)).
Consider the following example of a GAM using splines with 24 parameters:

- Setting df $= 24$ gives *B*-splines with $\lambda_j = 0$.
- Setting df $= 4$ gives *P*-splines with $\lambda_j = 418$.
- Setting df $= 2$ gives *P*-splines with $\lambda_j = 42751174892$.



AVAILABLE BASE LEARNERS

There is a large amount of possible base learners, e.g.:

- Linear effects and interactions (with or without intercept)
- Uni- or multivariate splines and tensor product splines
- Trees
- Random effects and Markov random fields
- Effects of functional covariates
- ...

In combination with the flexible choice of loss functions, this allows boosting to fit a huge number of different statistical models with the same algorithm. Recent extensions include GAMLSS-models, where multiple additive predictors are boosted to model different distribution parameters (e.g., conditional mean and variance for a Gaussian model).

TAKE-HOME MESSAGE

- Componentwise gradient boosting is the statistical re-interpretation of gradient boosting.
- We can fit a large number of statistical models, even in high dimensions ($p \gg n$).
- A drawback compared to statistical models is that we do not get valid inference for coefficients $\rightarrow$ post-selection inference.
- In most cases, gradient boosting with tree will dominate componentwise boosting in terms of performance due to its inherent ability to include higher-order interaction terms.