Boosting Algorithms

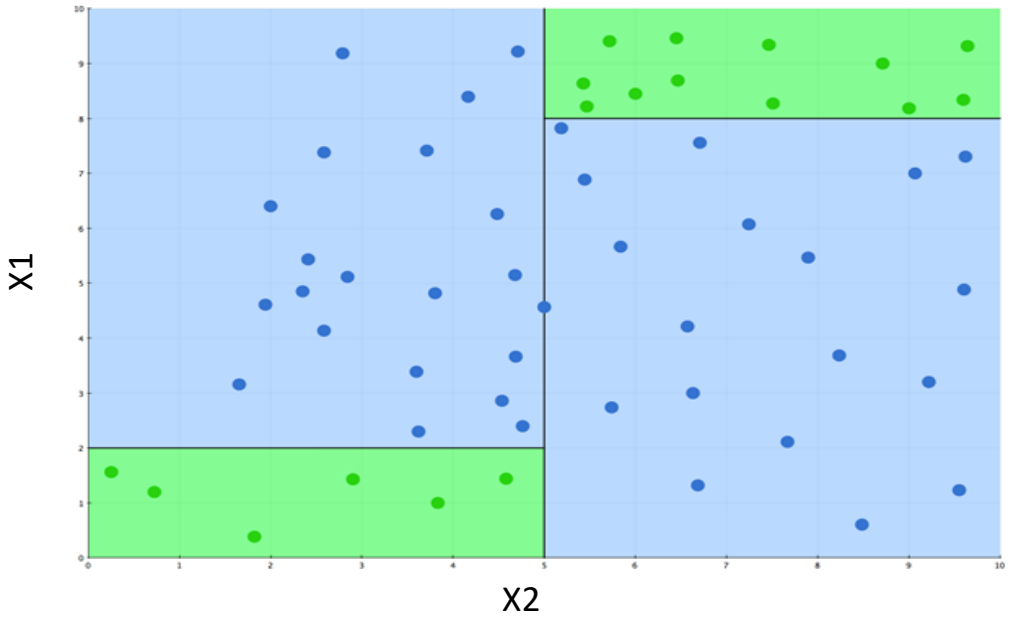
Gilmer Valdes, PhD DABR
Assistant Professor, Department of Radiation Oncology
University of California, San Francisco.

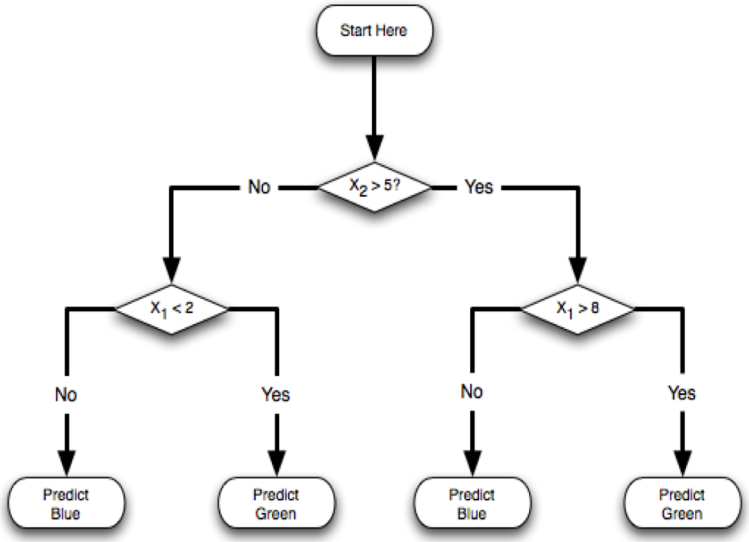
Outline

- The origin of Boosting
- Adaboost
- Adaboost as a exponential loss minimizer
- Gradient Boosting
- Practical considerations.

Recap

- Decision Trees





Advantages and Disadvantages of Decision Trees

Advantages

1. Consistent estimators
2. Fast
3. *Natural way to handle missing data*
4. *Independent of monotonic transformations of the input variables.*
5. *Interpretable*

Disadvantages

1. Each rule is obtained with exponentially less data.
2. Rules are *memory-less.*
3. *They are unstable procedures.*
4. *Do not model linear data very well.*

The origin of Boosting

Hypothesis: *Does the existence of a weak algorithm that outputs a hypothesis that is just slightly better than random implies that there is a strong algorithm which outputs a hypothesis of arbitrary accuracy? (1988)*

http://www.cis.upenn.edu/~mkearns/papers/boostnote.pdf

Adaboost

Discrete AdaBoost [Freund and Schapire (1996b)]

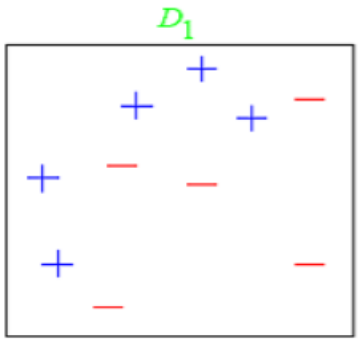
1. Start with weights $w_i = 1/N, i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$.

"Experiments with a new boosting algorithm." Yoav Freund and Robert E. Schapire (1996)
http://web.eecs.utk.edu/~leparker/Courses/CS425-528-fall12/Handouts/AdaBoost.M1.pdf

https://alliance.seas.upenn.edu/~cis520/dynamic/2018/wiki/index.php?n=Lectures.Boosting

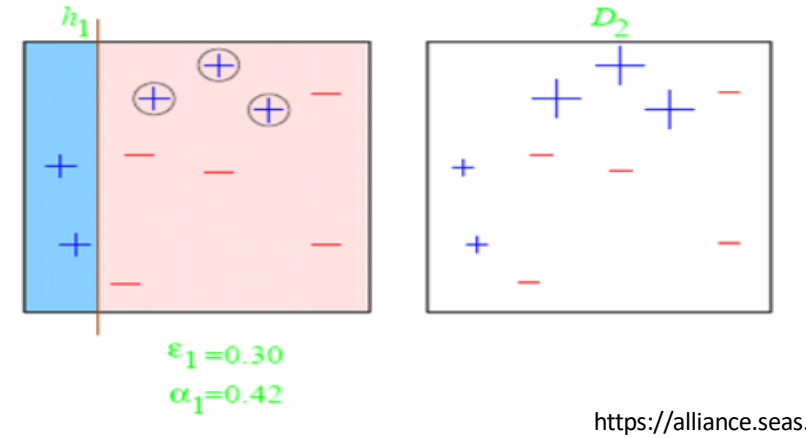
Boosting Decision Stumps

$$h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$



$h_1(x)$ = 

First round:



https://alliance.seas.upenn.edu/~cis520/wiki/

Adaboost

Discrete AdaBoost [Freund and Schapire (1996b)]

1. Start with weights $w_i = 1/N, i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$.

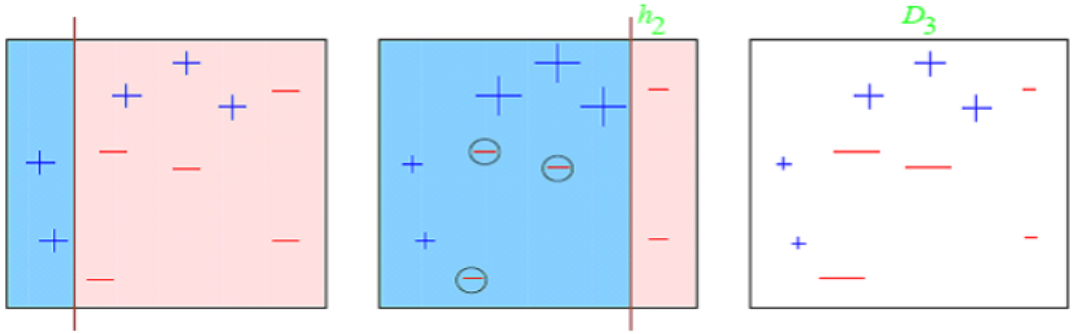
"Experiments with a new boosting algorithm." Yoav Freund and Robert E. Schapire (1996)
http://web.eecs.utk.edu/~leparker/Courses/CS425-528-fall12/Handouts/AdaBoost.M1.pdf

https://alliance.seas.upenn.edu/~cis520/dynamic/2018/wiki/index.php?n=Lectures.Boosting

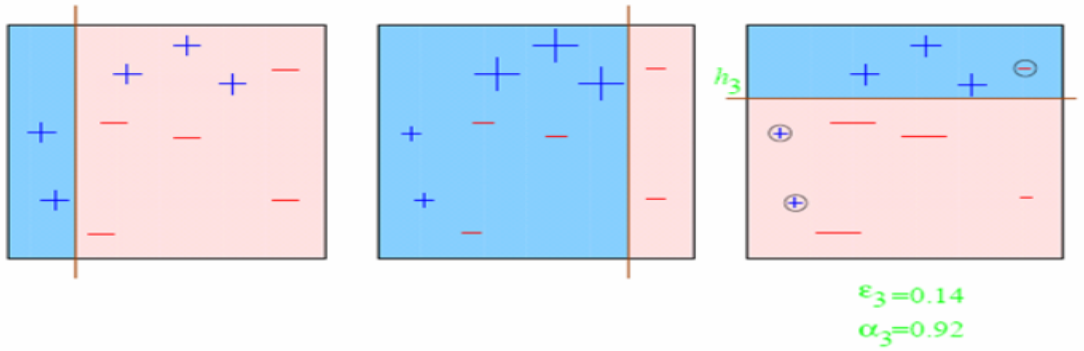
Boosting Decision Stumps

$$h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

Second round:



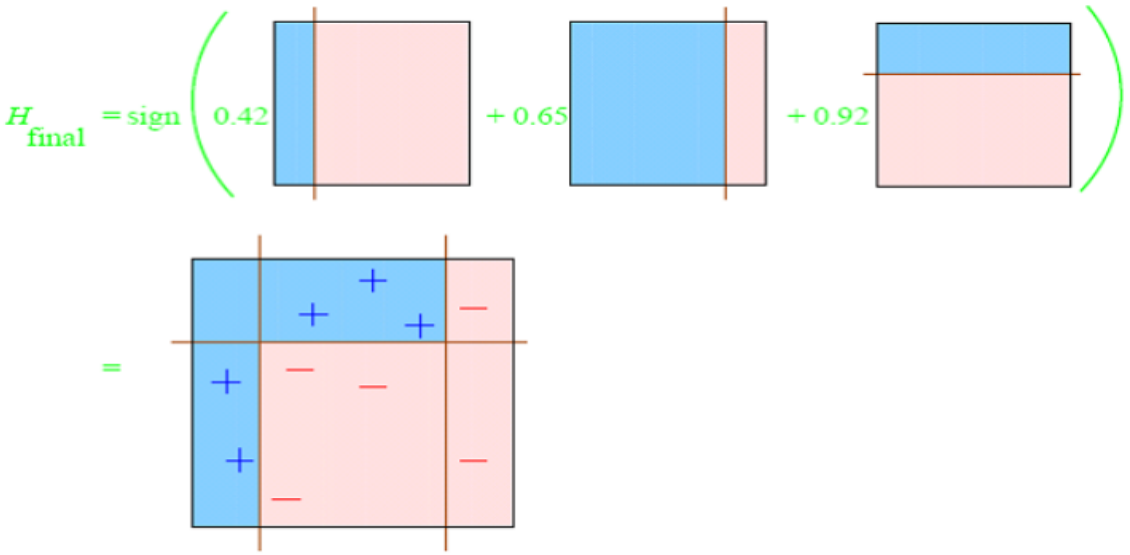
Third round:



https://alliance.seas.upenn.edu/~cis520/wiki/

Boosting Decision Stumps

The final classifier: $h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$



https://alliance.seas.upenn.edu/~cis520/wiki/

Why boosting works?

- All machine Learning algorithms consist of:

1. Choose a function class f(x) (ex trees, ensemble)

2. Choose a loss function given the data $L = \sum_{i=1}^{N} L(y_i|F(x))$

3. Choose an optimization method to find parameters (ex gradient descent).

Why boosting works?

- Minimizes exponential loss

$$L(y, f(x)) = exp(-yf(x))$$

- Using Forward Stagewise Additive Modeling

"Additive Logistic Regression: Statistical View of Boosting" (2000). Friedman et al

https://projecteuclid.org/download/pdf_1/euclid.aos/1016218223

Why boosting works?

1. Type of function: Additive modeling is just the fact that the solution to adaboost is a weighted sum of “basis” functions

$$\text{sign}[\sum_{m=1}^{M} f_m(x)].$$

$$f_m(x) = \beta_m b(x; \gamma_m).$$

2. Loss function: Average over the training data.

$$L = argmin_{\{\beta_m, \gamma_m\}_{m=1}^{M}} \sum_{i=1}^{N} exp(-y_i * \sum_{i=1}^{N} \beta_m \, b(x; \, \gamma_m))$$

$$L(y, f(x)) = exp(-yf(x))$$

Why boosting works?

3. Stage forward fitting

- It is a greedy optimization algorithm in which each function is added one at a time.
- Previously added terms are not modified.

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\,\gamma} E\left[y - F_{m-1}(x) - \beta b(x;\gamma)\right]^2$$

Gradient Boosting

Algorithm 1: Gradient_Boost

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$
4. $\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\quad \rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. endFor

end Algorithm

Gradient Boosting vs Adaboost

Similar

•Computing trees in a sequential way

•Final model is a sum of weighted trees

Different

•Trees in Adaboost are fitted on the same data but changing weights

•Trees in boosting are fitted by modifying the target variable

Gradient Boosting: Regression with Least Squared Loss

$$L = \sum_{i=1}^{N} (y_i - F(x_i))^2$$

Algorithm 2: LS_Boost

$F_0(\mathbf{x}) = \bar{y}$

For $m = 1$ to M do:

$\tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i), \quad i = 1, N$

$(\rho_m, \mathbf{a}_m) = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(\mathbf{x}_i; \mathbf{a})]^2$

$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

Least Squared Loss is a bad loss for classification

- Exponential

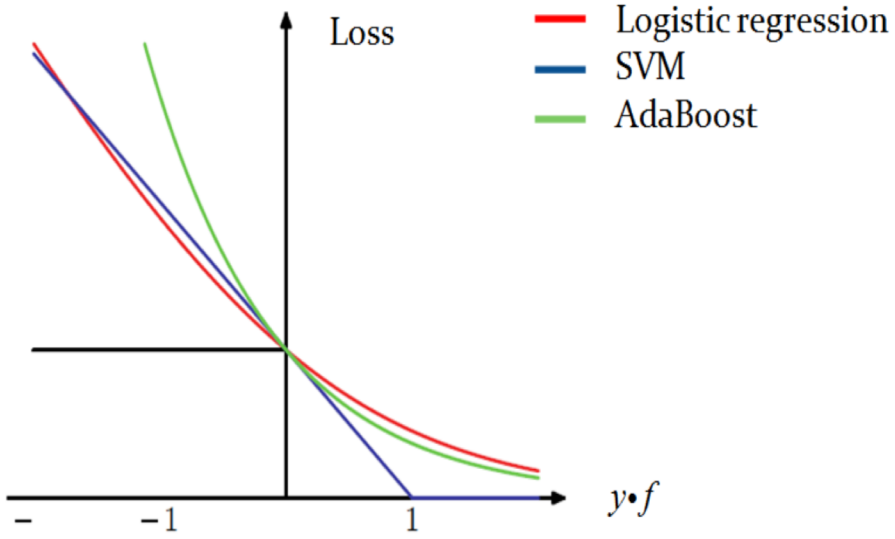
$$L(y, f(x)) = e^{-yf(x)}$$

- Log-loss

$$L(y, f(x)) = \log(1 + e^{-yf(x)})$$

- Hinge loss (SVM)

$$L(y, f(x)) = \max(0, 1 - yf(x))$$



Gradient Boosting for Classification

$$L(y, F) = \log\left(1 + \exp(-2yF)\right), \quad y \in \{-1, 1\}.$$

Algorithm 5: $\mathbf{L_2}$_TreeBoost

$F_0(\mathbf{x}) = \frac{1}{2}\log\frac{1+\bar{y}}{1-\bar{y}}$

For $m = 1$ to M do:

$\tilde{y}_i = 2y_i / (1 + \exp(2y_i F_{m-1}(\mathbf{x}_i))),\ i = 1, N$

$\{R_{jm}\}_1^J = J$–terminal node $tree(\{\tilde{y}_i, \mathbf{x}_i\}_1^N)$

$\gamma_{jm} = \sum_{\mathbf{x}_i \in R_{jm}} \tilde{y}_i \Big/ \sum_{\mathbf{x}_i \in R_{jm}} |\tilde{y}_i|\,(2 - |\tilde{y}_i|),\ j = 1, J$

$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^J \gamma_{jm} 1(\mathbf{x} \in R_{jm})$

endFor

end Algorithm

Regularization in Gradient Boosting

$$f_m(x) = f_{m-1}(x) + \nu T(x; \theta_m)$$

1. Learning rate (also called shrinkage)

2. If decrease learning rate need to increase the number of three (M)
   a) First fix a learning rate (say 0.1).
   b) Then find the optimal M.
   c) Then you can divide learning rate by 2 and set M = 2M.

3. Very important parameter in gradient boosting

Regularization in Gradient Boosting

eta: shrinkage	0.1
num_rounds: number of trees	
max_depth : maximum depth of tree	7
sub_sample: subsample ratio of the training instance	0.5
colsample_bytree: fraction of columns to sample	

http://xgboost.readthedocs.io/en/latest/parameter.html

Gradient Boosting vs Random Forests

•Gradient Boosting generally perform better than a random forest.

•Gradient Boosting have a few hyper-parameters to tune.

•Random forest is almost tuning-free:

- You may want to control how big a tree can grow
  - max_depth or min sample per leaf
- max_features

Gradient Boosting vs Neural Nets

- Random Forest and Gradient Boosting models work well on tabular data.
  - Each column of a tabular data has high information content on its own
  - Demographic features, such as the age and gender of a person
- Neural Nets useful when applied to “raw” sensory data
  - Speech, Images, NLP
  - Individual feature contains very little information
  - Some success on training embeddings for categorical features on tabular data

Empirical Comparison of Algorithms

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	—	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	—	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	—	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	—	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	—	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	—	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	—	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	—	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	—	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	—	.496	.562	.781	.825	.738	.735	.347	-.633	.481	.489

"An Empirical Comparison of Supervised Learning Algorithms." Rich Caruana, Alexandru Niculescu-Mizil. Cornell University. https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

Gradient Boosting in R

https://cran.r-project.org/web/packages/gbm/gbm.pdf

Package **GBM: hyper-parameters**

`n.trees`	Integer specifying the total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion. Default is 100.
`interaction.depth`	Integer specifying the maximum depth of each tree (i.e., the highest level of variable interactions allowed). A value of 1 implies an additive model, a value of 2 implies a model with up to 2-way interactions, etc. Default is 1.
`n.minobsinnode`	Integer specifying the minimum number of observations in the terminal nodes of the trees. Note that this is the actual number of observations, not the total weight.

Gradient Boosting in R

https://cran.r-project.org/web/packages/gbm/gbm.pdf

Package **GBM: hyper-parameters**

`shrinkage`	a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually work, but a smaller learning rate typically requires more trees. Default is 0.1.
`bag.fraction`	the fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomnesses into the model fit. If `bag.fraction` < 1 then running the same model twice will result in similar but different fits. `gbm` uses the R random number generator so `set.seed` can ensure that the model can be reconstructed. Preferably, the user can save the returned `gbm.object` using `save`. Default is 0.5.
`train.fraction`	The first `train.fraction * nrows(data)` observations are used to fit the `gbm` and the remainder are used for computing out-of-sample estimates of the loss function.
`cv.folds`	Number of cross-validation folds to perform. If `cv.folds`>1 then `gbm`, in addition to the usual fit, will perform a cross-validation, calculate an estimate of generalization error returned in `cv.error`.

Questions?

Email: gilmer.valdes@ucsf.edu