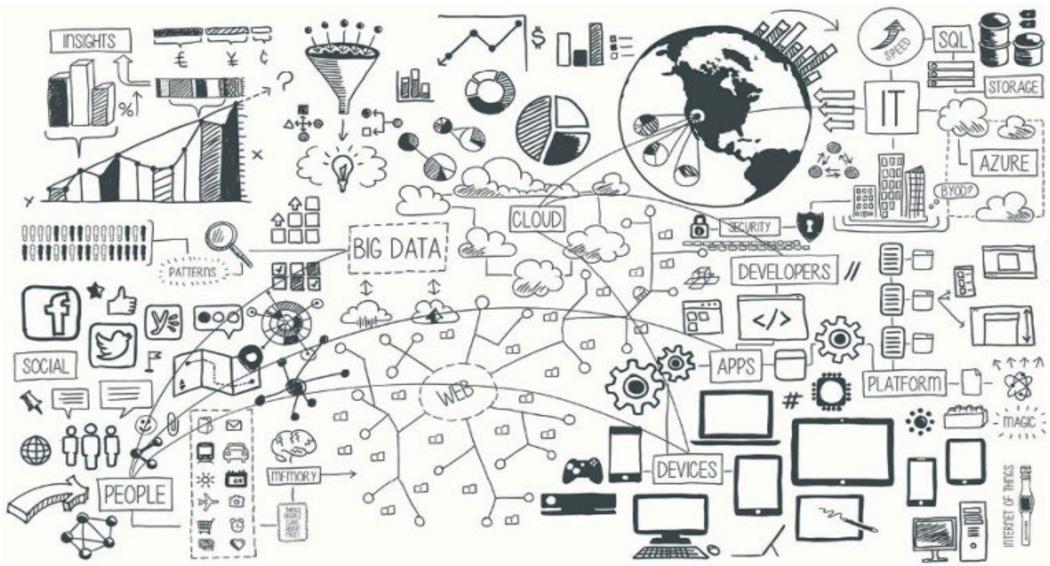
Data Mining (Minería de Datos)

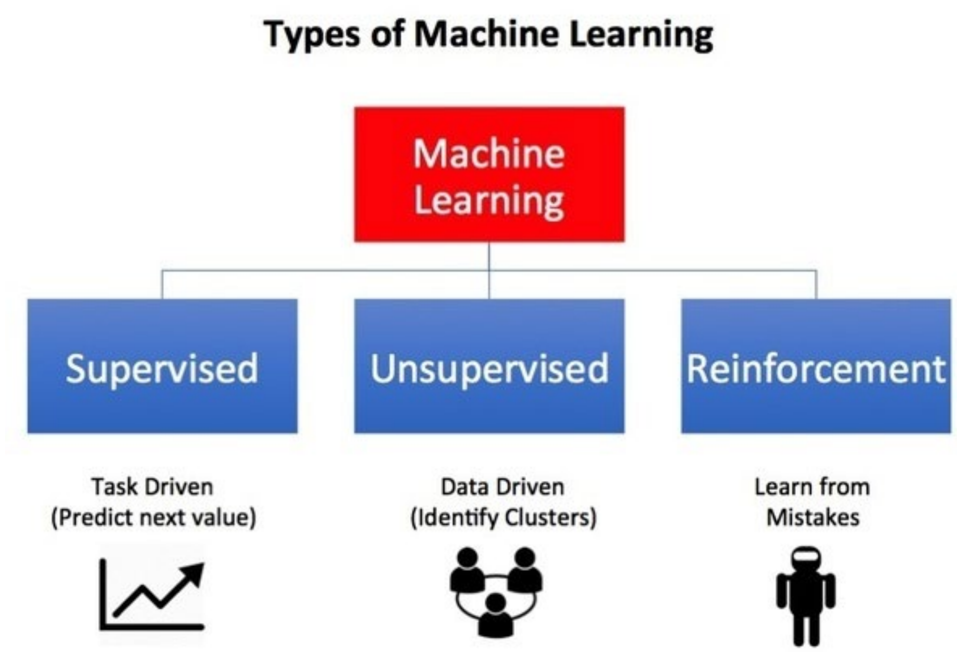
Ensembles: Gradient Boosting



Sixto Herrera

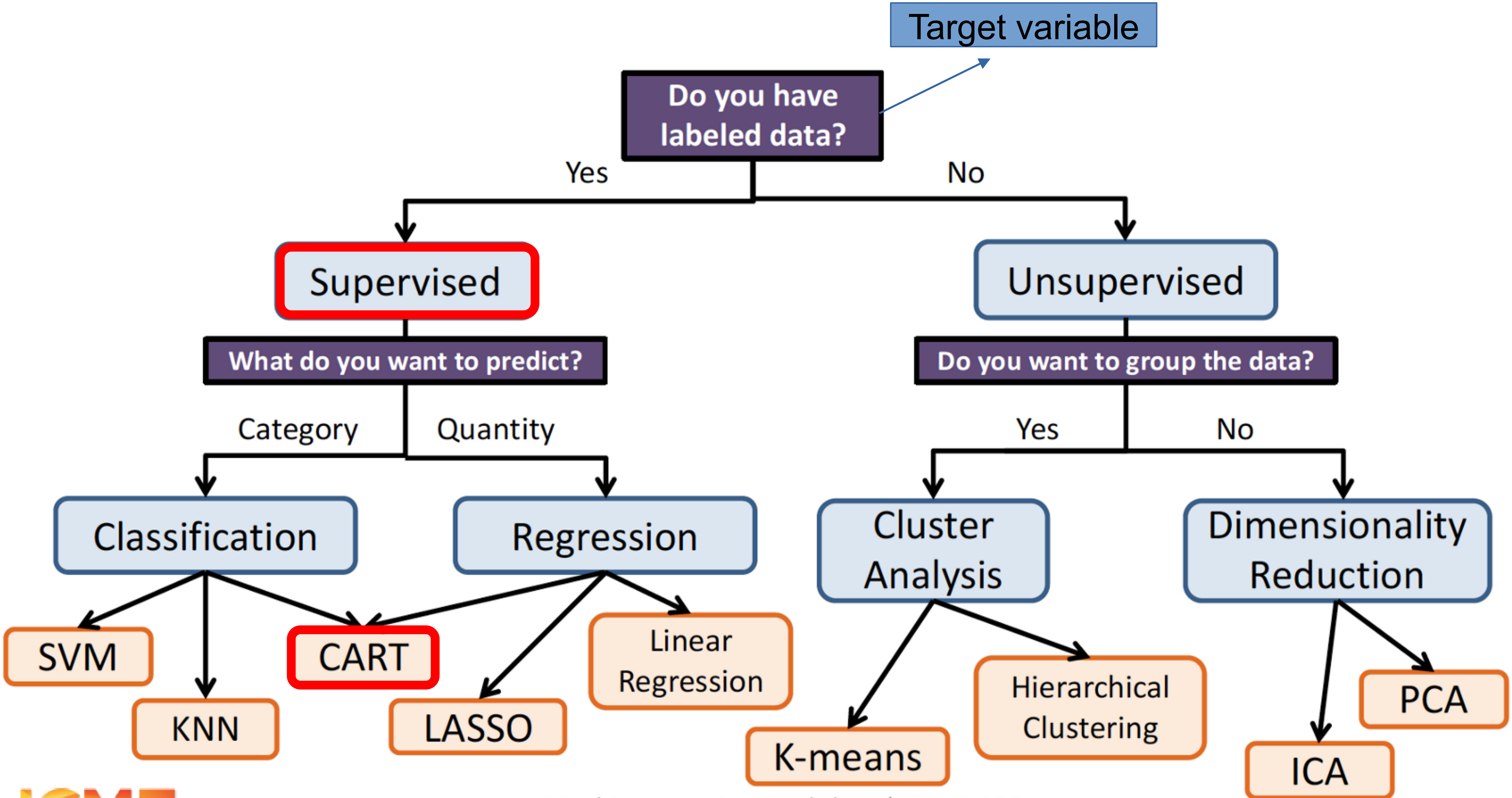
Grupo de Meteorología

Univ. de Cantabria – CSIC
MACC / IFCA



NOTA: Las líneas de código de R en esta presentación se muestran sobre un fondo gris.

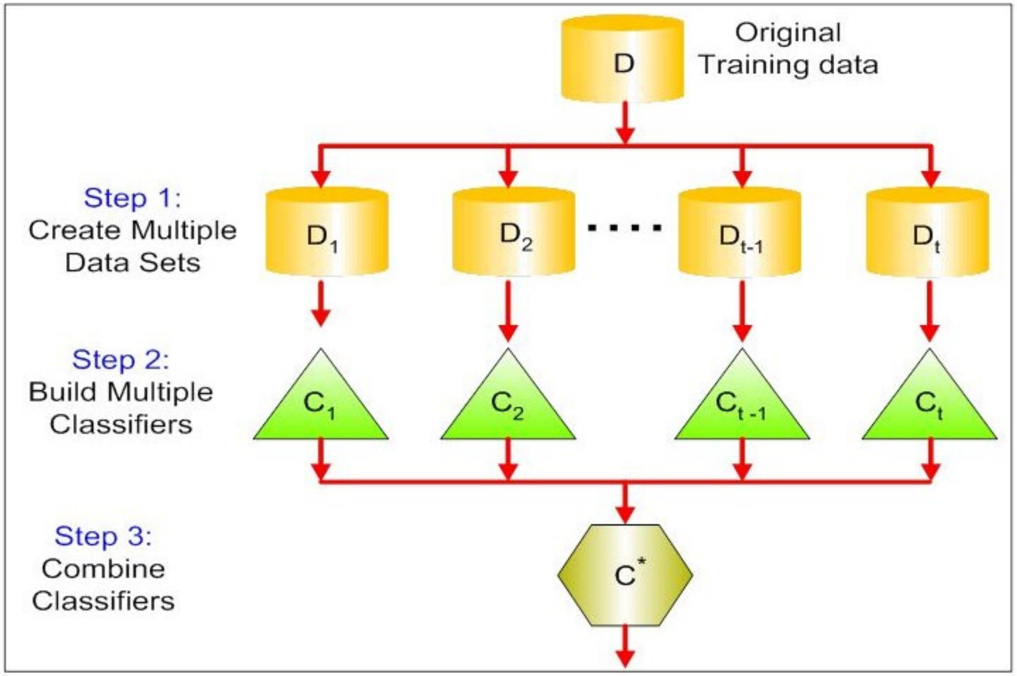
Oct	29	Presentación, introducción y perspectiva histórica
	30	Paradigmas, problemas canónicos y data challenges
	31	Reglas de asociación
Nov	4	Practica: Reglas de asociación
	6	Evaluación, sobreajuste y crossvalidación
	11	Practica: Crossvalidación
	13	Árboles de clasificación y decisión
	18	Practica: Árboles de clasificación
	20	Técnicas de vecinos cercano (k-NN)
	25	Práctica: Vecinos cercanos
	27	Comparación de Técnicas de Clasificación.
Dic	2	Árboles de clasificación y regresión (CART)
	4	Práctica: Árboles de clasificación y regresión (CART)
	9	Practica: El paquete CARET
	11	**Ensembles: Bagging and Boosting**
	13	**Random Forests**
	16	**Gradient boosting**
	18	Práctica: XAI-Explainable Artificial Intelligence
Ene	8	Reducción de dimensión no lineal
	13	Reducción de dimensión no lineal
	15	Técnicas de agrupamiento
	20	Práctica: Técnicas de agrupamiento
	22	Predicción Condicionada
	24	Sesión de refuerzo/repaso.
	29	**Examen**



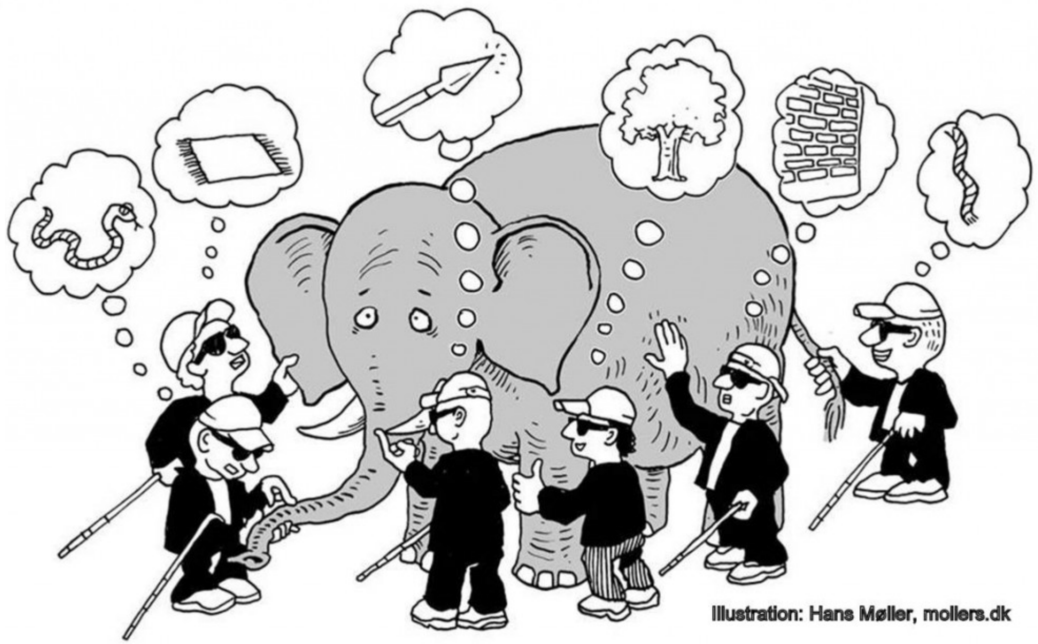
Target variable
Do you have labeled data?
Yes
No
Supervised
Unsupervised
What do you want to predict?
Do you want to group the data?
Category
Quantity
Yes
No
Classification
Regression
Cluster Analysis
Dimensionality Reduction
SVM
KNN
CART
LASSO
Linear Regression
K-means
Hierarchical Clustering
PCA
ICA

Ensemble learning

Ensemble learning is a supervised approach in which the basic idea is to generate multiple weak models on a training dataset and combining them to generate a strong model which improves the ***stability*** and the ***performance*** of the individual models.



The wisdom of the crowd



Fable of blind men and elephant
https://en.wikipedia.org/wiki/Blind_men_and_an_elephant

Ensemble learning

Ensemble approaches are typically used with CART.

Pros

Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons

Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the instability of the trees can be reduced and their predictive performance substantially improved.

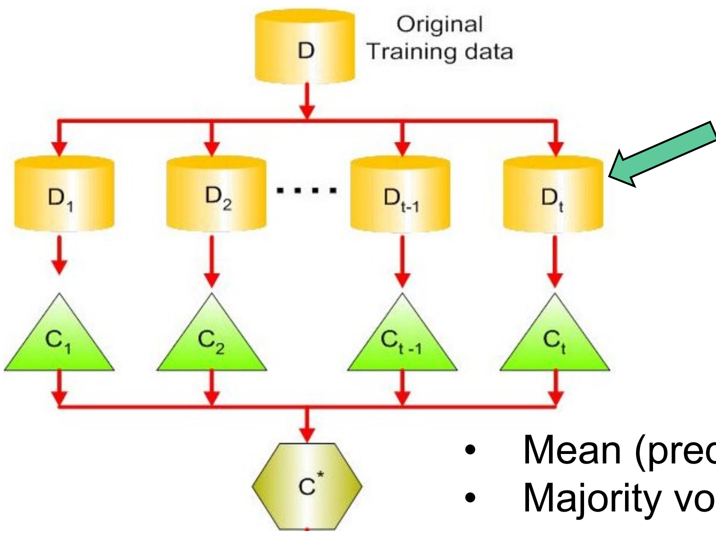
Ensemble learning

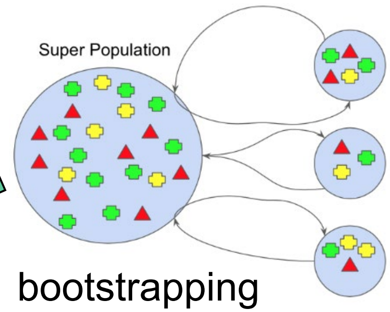
Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.





- Mean (prediction)
- Majority voting (classification)

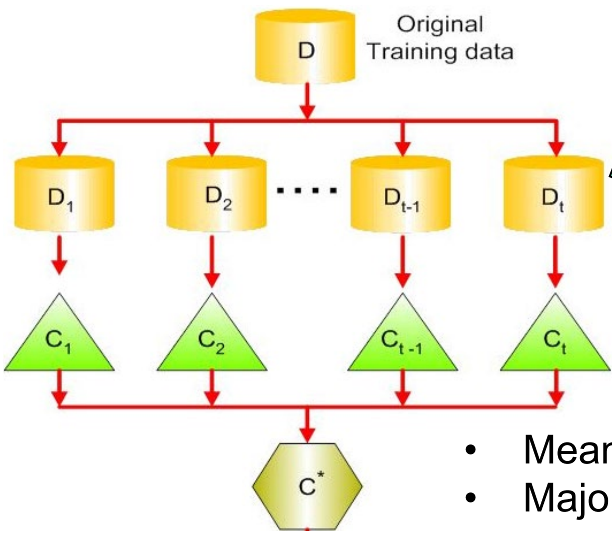
Ensemble learning

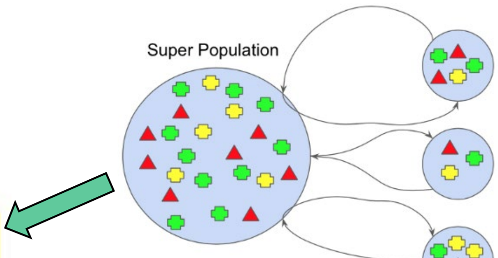
Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.





bootstrapping

- Mean (prediction)
- Majority voting (classification)

Weak learners

Low bias and high variance

High degree of freedom models
e.g. fully developed trees

Ensemble learning

Ensemble approaches are typically used with CART.

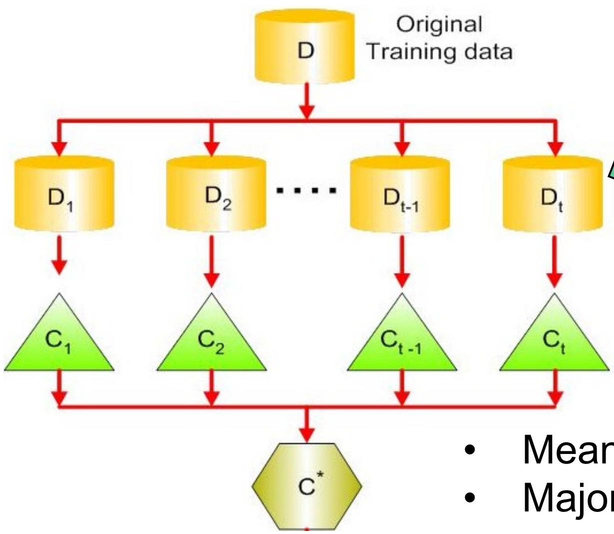
Pros

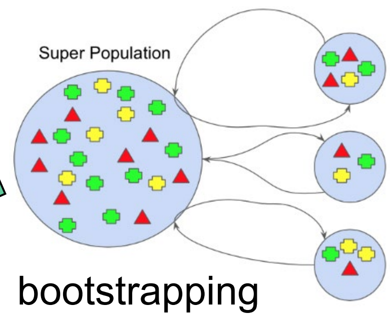
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons

Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.





bootstrapping

- Mean (prediction)
- Majority voting (classification)

Weak learners

High bias and low variance

Low degree if freedom models
e.g. low depth trees

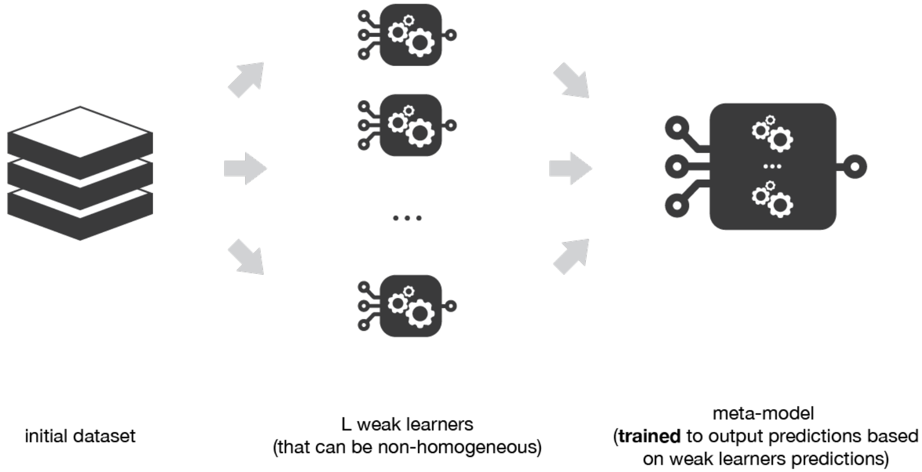
Ensemble learning

Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.



Heterogenous Weak Learners

Stacking considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions.

https://stats.stackexchange.com/questions/290701/how-to-stack-machine-learning-models-in-r

https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

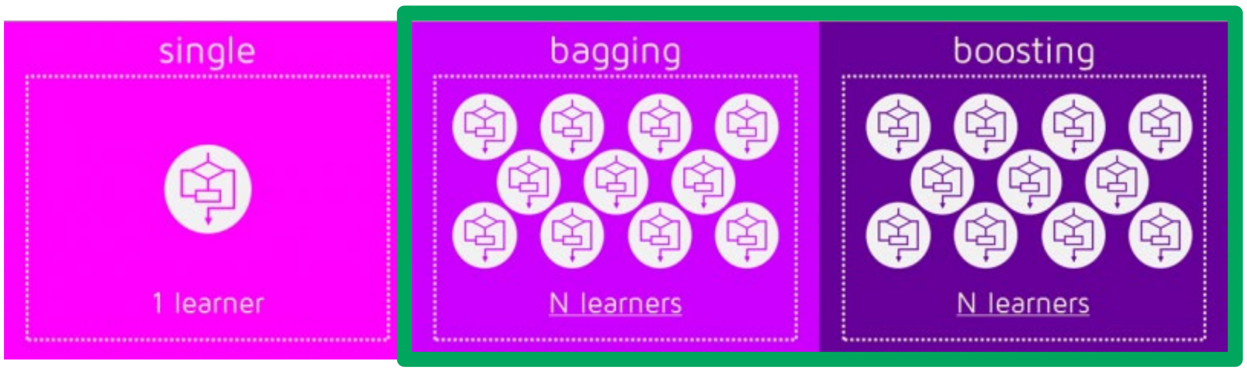
Ensemble learning

Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.



Homogenous Weak Learners

Bagging and Boosting

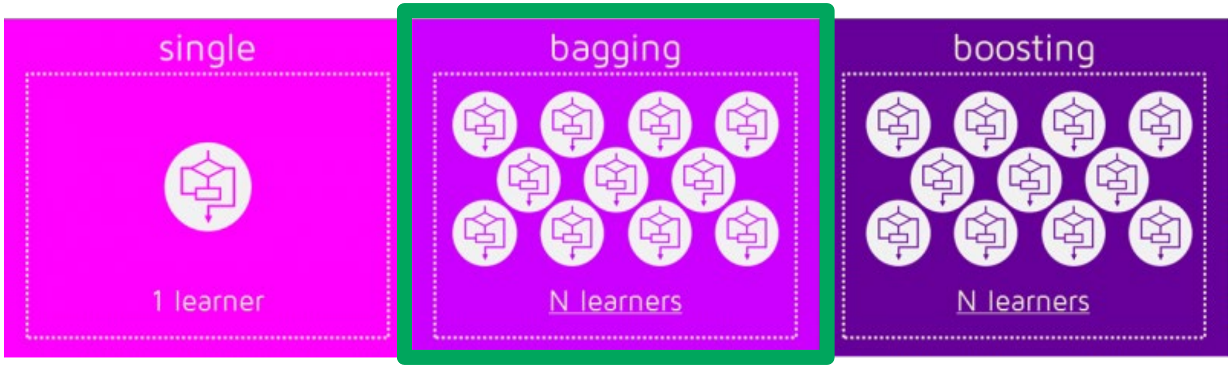
Ensemble learning

Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.



Homogenous Weak Learners

Low bias and high variance

High degree of freedom models
e.g. fully developed trees

Random Forests

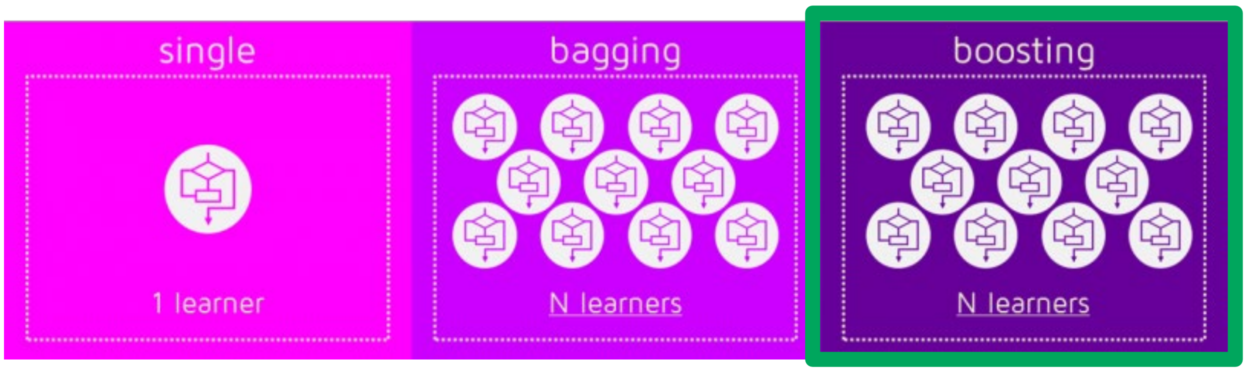
Ensemble learning

Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.



Homogenous Weak Learners

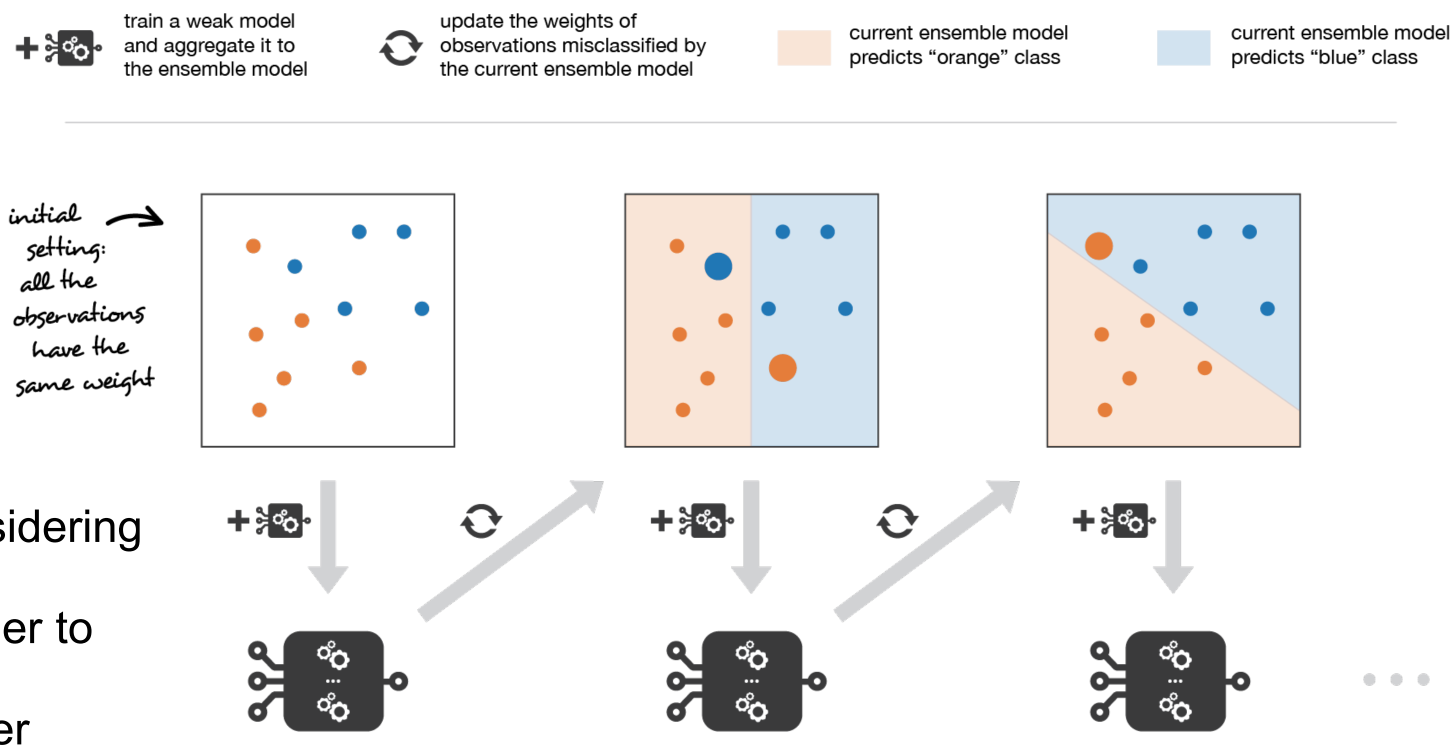
High bias and low variance

Low degree if freedom models
e.g. low depth trees

AdaBoost: Adaptive Boosting

Ensemble: Boosting Methods

Adaptative Boosting (AdaBoost)



Step 1: All the observations have the **same weights**

Repeat (1:L):
 a) Fit the weak model considering the observations weights.
 b) Evaluate the weak learner to obtain its coefficient.
 c) Update the strong learner adding the weak learner.
 d) Update the obervations weights

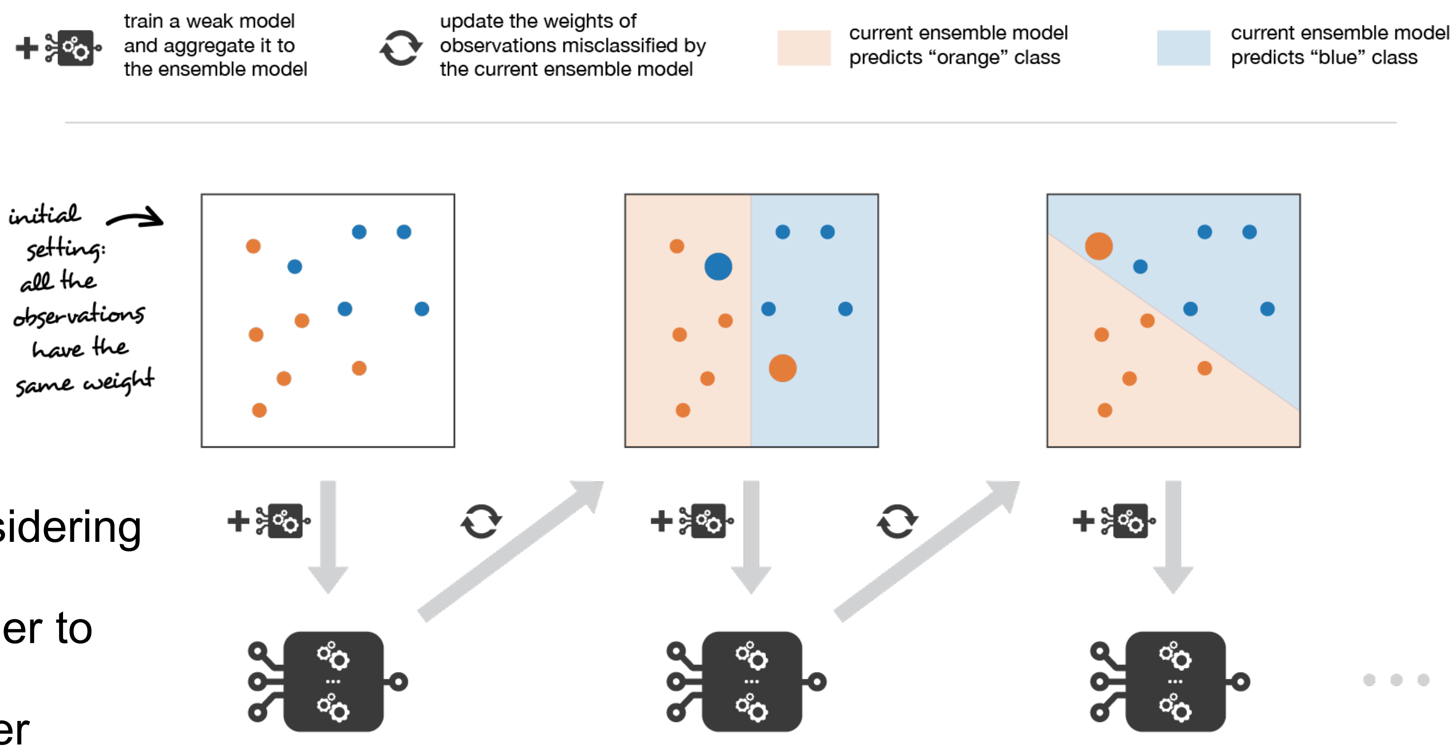
Result: A strong learner is obtained as a simple linear combination of weak learners weighted by coefficients expressing the performance of each learner. Variants of this algorithm could be obtained by modifying the loss function (e.g. logit for classification or L2 for regression).

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.) \qquad \text{where } c_l\text{'s are coefficients and } w_l\text{'s are weak learners}$$

$$(c_l, w_l(.)) = \arg\min_{c,w(.)} E(s_{l-1}(.) + c \times w(.)) = \arg\min_{c,w(.)} \sum_{n=1}^{N} e(y_n, s_{l-1}(x_n) + c \times w(x_n))$$

Ensemble: Boosting Methods

Adaptative Boosting (AdaBoost)



Step 1: All the observations have the **same weights**

Repeat (1:L):
a) Fit the weak model considering the observations weights.
b) Evaluate the weak learner to obtain its coefficient.
c) Update the strong learner adding the weak learner.
d) Update the obervations weights

Result: A strong learner is obtained as a simple linear combination of weak learners weighted by coefficients expressing the performance of each learner. Variants of this algorithm could be obtained by modifying the loss function (e.g. logit for classification or L2 for regression).

AdaBoost combines multiple single split decision trees. AdaBoost puts more emphasis on observations that are more difficult to classify by adding new weak learners where needed.

Ensemble: Boosting Methods

Adaptative Boosting (AdaBoost)

train a weak model and aggregate it to the ensemble model

update the weights of observations misclassified by the current ensemble model

current ensemble model predicts “orange” class

current ensemble model predicts “blue” class

Step 1: All the obs
have the **same w**

Repeat (1:L):
a) Fit the weak m
the observations w
b) Evaluate the w
obtain its coefficie
c) Update the str
adding the weak l
d) Update the ob
weights

Result: A strong l
coefficients expre
by modifying the l

AdaBoost comb
observations that

Explicación detallada de AdaBoost

1. Inicie con un conjunto de entrenamiento (X, Y) con m observaciones denotadas como $(x_1, y_1), \ldots, (x_m, y_m)$ de tal manera que $x_i \in R^p$. Los valores de y deben ser -1 o 1 para aplicar el método.
2. Inicie con la distribución discreta $D_1(i) = 1/m$ que indica el peso de la observación i en la iteración 1.
3. Para $t = 1, \ldots, T$.

- Construya un clasificador h_t definido así: $h_t : X \to \{-1, 1\}$.
- Calcule el error asociado ϵ_t al clasificador $\epsilon_t = \sum_{i=1}^{m} D_t(i) \times \delta_i$, donde $\delta_i = 0$ si $h_t(x_i) = y_i$, es decir, si fue correcta la clasificación; caso contrario es $\delta_i = 1$.
- Calcule la nueva distribución $D_{t+1}(i) = D_t(i) \times F_i / Z_t$, donde:
  - $F_i = \exp(-\alpha_t)$ si la clasificación fue correcta, es decir si $h_t(x_i) = y_i$.
  - $F_i = \exp(\alpha_t)$ si la clasificación fue incorrecta, es decir si $h_t(x_i) \neq y_i$.
  - $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
  - Z_t es una constante de normalización de tal manera que $\sum_{i=1}^{m} D_t(i) = 1$. Usualmente es $\sum D_t(i) \times F_i$.

3. Construya el clasificador final H_{final} como el promedio ponderado de los t clasificadores h_t, usando $H_{final} = sign(\sum_t \alpha_t h_t(x))$.

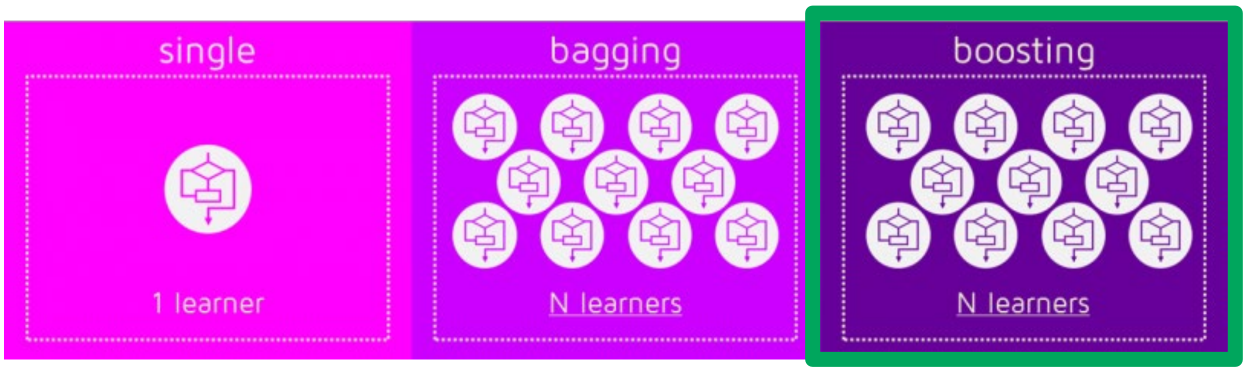
Ensemble learning

Ensemble approaches are typically used with CART.

Pros
Trees are very easy to explain (even easier than linear regression)
Trees can be plotted graphically, and are easily interpreted
Trees can easily handle qualitative predictors
They work fine on both classification and regression problems

Cons
Poor prediction accuracy (compared with other approaches)
Instability when changing the train/test partition (cross-validation is key)

By aggregating many trees, the **instability** of the trees can be reduced and their **performance** improved.



Homogenous Weak Learners

High bias and low variance

Low degree if freedom models
e.g. low depth trees

Gradient Descent Boosting

Ensemble: Boosting Methods

Gradient Boosting

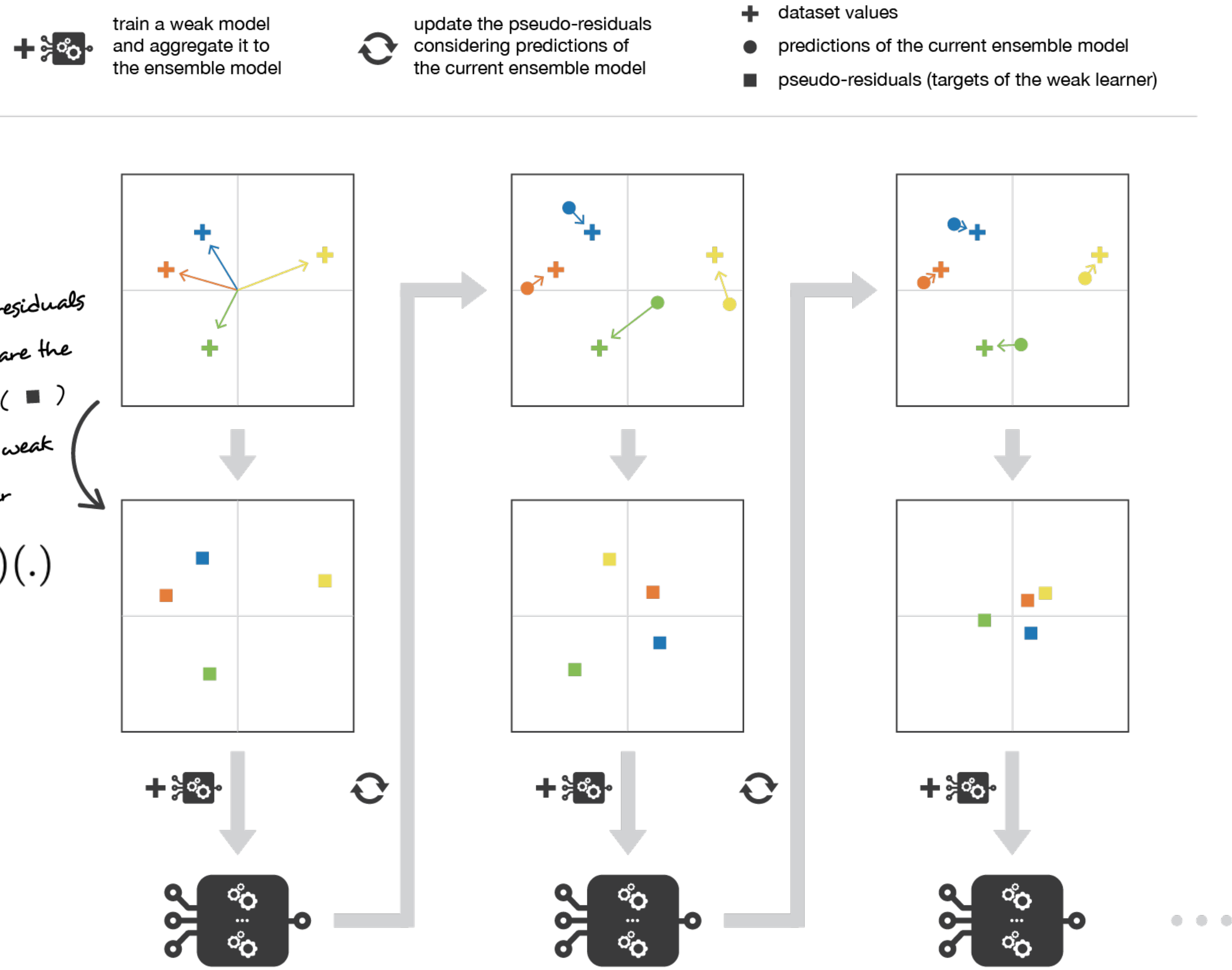
Gradient boosting casts the problem into a gradient descent one: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

pseudo-residuals (↗) are the targets (■) of the weak learner

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

Pseudo-residuals:

$$-\nabla_{s_{l-1}} E(s_{l-1})(.)$$



$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.)$$

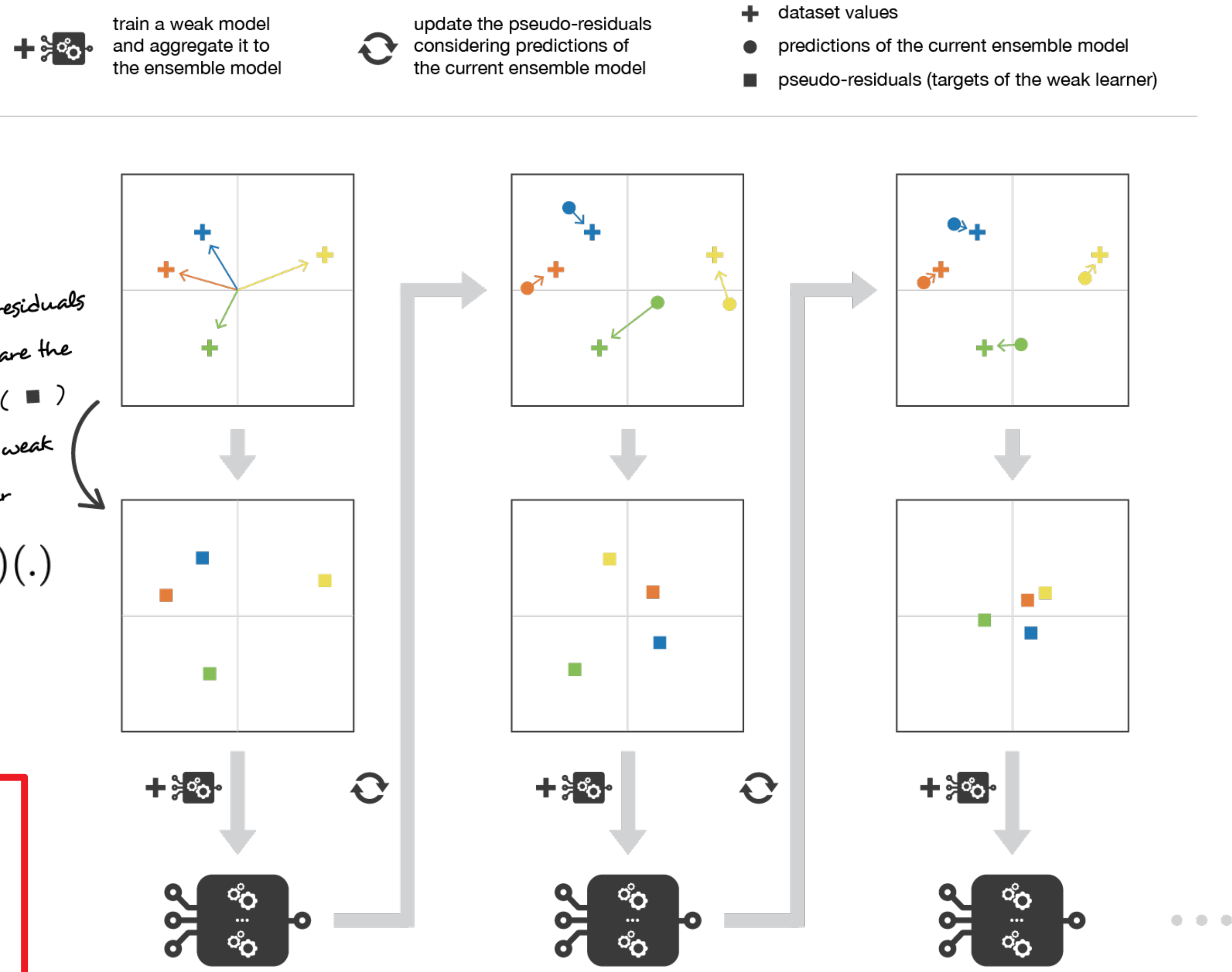
where c_l's are coefficients and w_l's are weak learners

Ensemble: Boosting Methods

Gradient Boosting

Gradient boosting casts the problem into a gradient descent one: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

Pseudo-residuals (↗) are the targets (■) of the weak learner



$$s_l(.) = s_{l-1}(.) - \boxed{c_l} \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

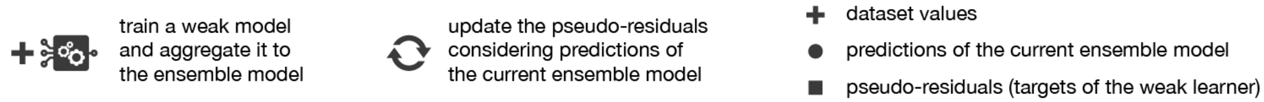
Pseudo-residuals:

$$-\nabla_{s_{l-1}} E(s_{l-1})(.)$$

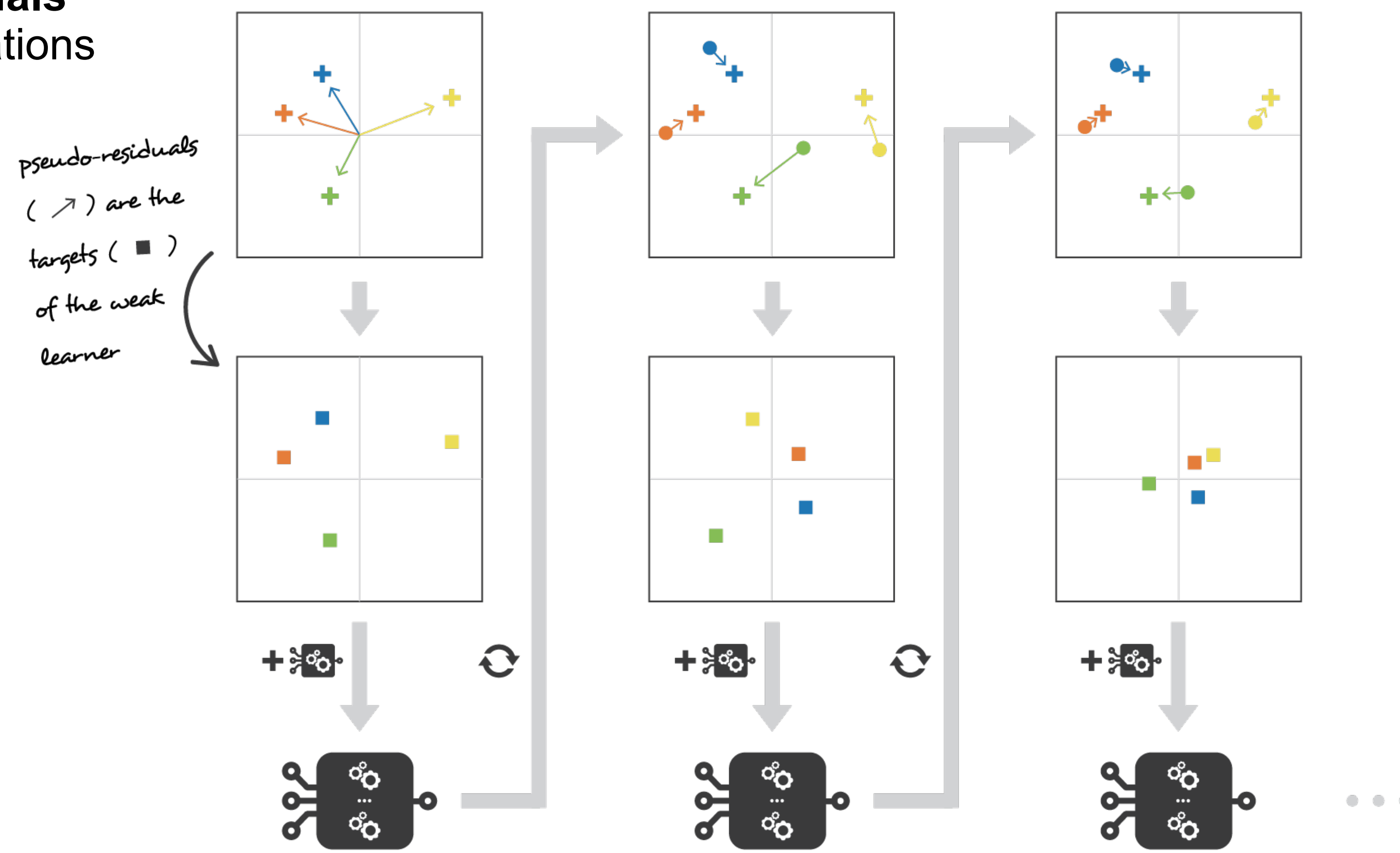
Step size: how much we update the ensemble model in the direction of the new weak learner

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.)$$ where c_l's are coefficients and w_l's are weak learners

Gradient Boosting



Step 1: The **pseudo-residuals** are set equal to the observations



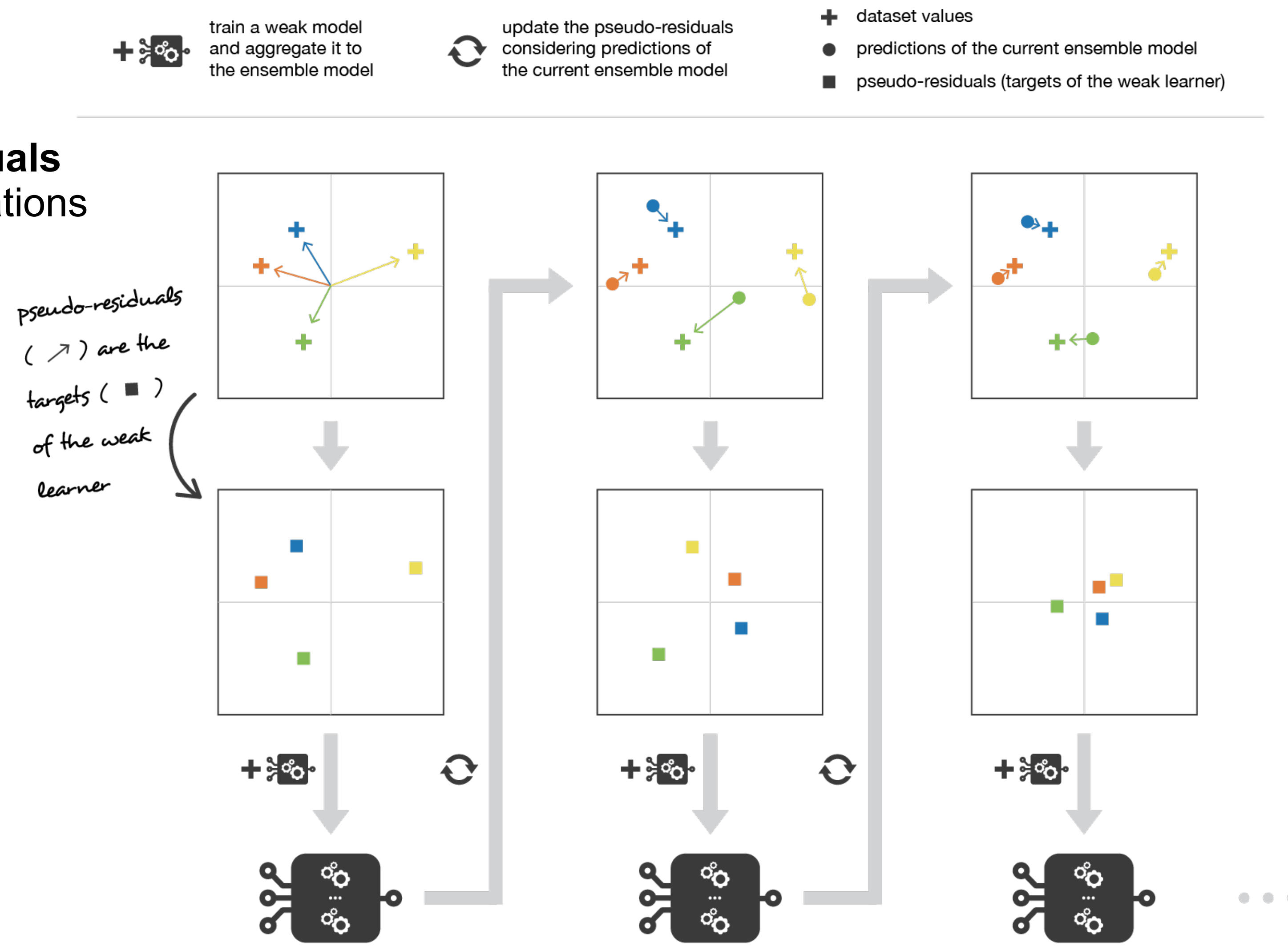
$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.)$$ where c_l's are coefficients and w_l's are weak learners

Ensemble: Boosting Methods

Gradient Boosting

Step 1: The **pseudo-residuals** are set equal to the observations

Repeat (1:L):
a) Fit the weak learner to pseudo-residuals.
b) Compute the optimal step size of the weak learner.
c) Update the strong learner adding the weak learner.
d) Update the pseudo-residuals

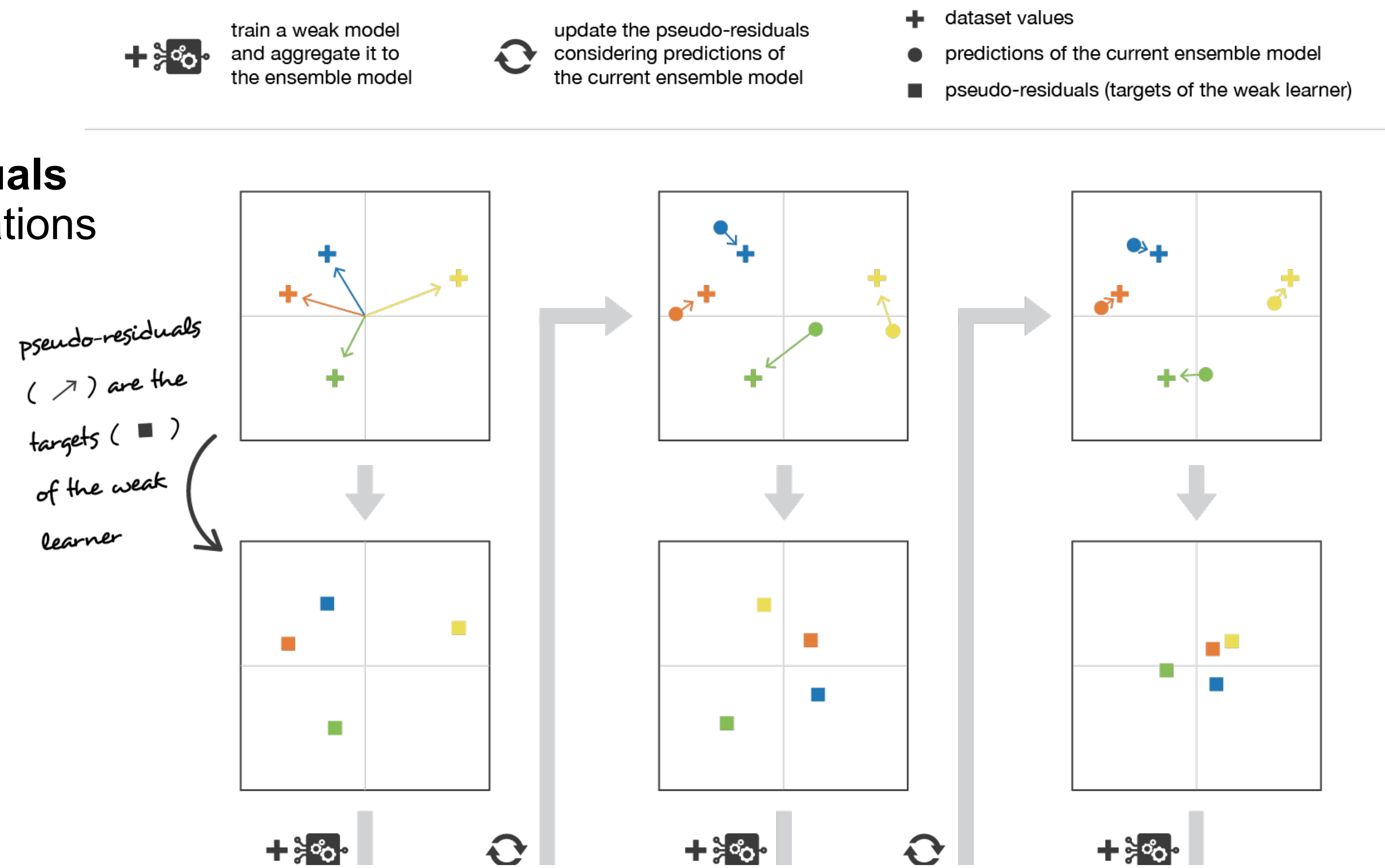


https://www.kdnuggets.com/2018/07/intuitive-ensemble-learning-guide-gradient-boosting.html

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.) \qquad \text{where } c_l\text{'s are coefficients and } w_l\text{'s are weak learners}$$

Ensemble: Boosting Methods

Gradient Boosting



Step 1: The **pseudo-residuals** are set equal to the observations

Repeat (1:L):
 a) Fit the weak learner to pseudo-residuals.
 b) Compute the optimal step size of the weak learner.
 c) Update the strong learner adding the weak learner.
 d) Update the pseudo-residuals

Gradient Boosting is comprised of only three elements:

Weak Learners – simple decision trees that are constructed based on purity scores (e.g., Gini).
Loss Function – a differentiable function you want to minimize. In regression, this could be a mean squared error, and in classification, it could be log loss.
Additive Models – additional trees are added where needed, and a functional gradient descent procedure is used to minimize the loss when adding trees.

Gradient Boosting

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*:

   $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

   $$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

   $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

https://en.wikipedia.org/wiki/Gradient_boosting#Algorithm

Weak Learners – simple decision trees that are constructed based on purity scores (e.g., Gini).
Loss Function – a differentiable function you want to minimize. In regression, this could be a mean squared error, and in classification, it could be log loss.
Additive Models – additional trees are added where needed, and a functional gradient descent procedure is used to minimize the loss when adding trees.

Ensemble: Boosting Methods

Adaptative vs Gradient Descent Boosting Approaches

Adaptative Boosting (AdaBoost)

Step 1: All the observations have the **same weights**

Repeat (1:L):
a) Fit the weak model considering the observations weights.
b) Evaluate the weak learner to obtain its coefficient.
c) Update the strong learner adding the weak learner.
d) Update the obervations weights

$$(c_l, w_l(.)) = \underset{c,w(.)}{\arg\min} E(s_{l-1}(.) + c \times w(.))$$

$$\underset{c,w(.)}{\arg\min} \sum_{n=1}^{N} e(y_n, s_{l-1}(x_n) + c \times w(x_n))$$

Gradient Boosting

Gradient boosting casts the problem into a gradient descent one: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

Pseudo-residuals: $-\nabla_{s_{l-1}} E(s_{l-1})(.)$

Step 1: The **pseudo-residuals** are set equal to the observations

Repeat (1:L):
a) Fit the weak learner to pseudo-residuals.
b) Compute the optimal step size of the weak learner.
c) Update the strong learner adding the weak learner.
d) Update the pseudo-residuals

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.) \qquad \text{where } c_l\text{'s are coefficients and } w_l\text{'s are weak learners}$$

Ensemble: Boosting Methods

Adaptative vs Gradient Descent Boosting Approaches

Adaptative Boosting (AdaBoost)

Step 1: All the observations have the **same weights**

Repeat (1:L):
a) Fit the weak model considering the observations weights.
b) Evaluate the weak learner to obtain its coefficient.
c) Update the strong learner adding the weak learner.
d) Update the obervations weights

$$(c_l, w_l(.)) = \arg\min E(s_{l-1}(.) + c \times w(.))$$

Gradient Boosting

Gradient boosting casts the problem into a gradient descent one: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

Pseudo-residuals: $-\nabla_{s_{l-1}} E(s_{l-1})(.)$

Step 1: The **pseudo-residuals** are set equal to the observations

Repeat (1:L):
a) Fit the weak learner to pseudo-residuals.
b) Compute the optimal step size of the weak learner.

Notice that, while adaptative boosting tries to solve at each iteration exactly the “local” optimisation problem (find the best weak learner and its coefficient to add to the strong model), gradient boosting uses instead a gradient descent approach and can more easily be adapted to large number of loss functions. Thus, gradient boosting can be considered as a generalization of adaboost to arbitrary differentiable loss functions.

Ensemble: Boosting Methods

Adaptative vs Gradient Descent Boosting Approaches

Adaptative Boosting

boosting {adabag} R Documentation

Applies the AdaBoost.M1 and SAMME algorithms to a data set

Description

Fits the AdaBoost.M1 (Freund and Schapire, 1996) and SAMME (Zhu et al., 2009) algorithms using classification trees as single classifiers.

Usage

```
boosting(formula, data, boos = TRUE, mfinal = 100, coeflearn = 'Breiman',
      control,...)
```

Arguments

`formula`	a formula, as in the `lm` function.
`data`	a data frame in which to interpret the variables named in `formula`.
`boos`	if TRUE (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If FALSE, every observation is used with its weights.
`mfinal`	an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to `mfinal=100` iterations.
`coeflearn`	if 'Breiman'(by default), `alpha=1/2ln((1-err)/err)` is used. If 'Freund' `alpha=ln((1-err)/err)` is used. In both cases the AdaBoost.M1 algorithm is used and `alpha` is the weight updating coefficient. On the other hand, if coeflearn is 'Zhu' the SAMME algorithm is implemented with `alpha=ln((1-err)/err)+ ln(nclasses-1)`.
`control`	options that control details of the rpart algorithm. See rpart.control for more details.
`...`	further arguments passed to or from other methods.

Details

Gradient Boosting

R: Generalized Boosted Regression Modeling (GBM) Find in Topic

gbm {gbm} R Documentation

Generalized Boosted Regression Modeling (GBM)

Description

Fits generalized boosted regression models. For technical details, see the vignette: `utils::browseVignettes("gbm")`.

Usage

```
gbm(formula = formula(data), distribution = "bernoulli",
  data = list(), weights, var.monotone = NULL, n.trees = 100,
  interaction.depth = 1, n.minobsinnode = 10, shrinkage = 0.1,
  bag.fraction = 0.5, train.fraction = 1, cv.folds = 0,
  keep.data = TRUE, verbose = FALSE, class.stratify.cv = NULL,
  n.cores = NULL)
```

Arguments

`formula`	A symbolic description of the model to be fit. The formula may include an offset term (e.g. y~offset(n)+x). If `keep.data = FALSE` in the initial call to gbm then it is the user's responsibility to resupply the offset to `gbm.more`.
`distribution`	Either a character string specifying the name of the distribution to use or a list with a component name specifying the distribution and any additional parameters needed. If not specified, gbm will try to guess: if the response has only 2 unique values, bernoulli is assumed; otherwise, if the response is a factor, multinomial is assumed; otherwise, if the response has class `"Surv"`, coxph is assumed; otherwise, gaussian is assumed. Currently available options are `"gaussian"` (squared error), `"laplace"` (absolute loss), `"tdist"` (t-distribution loss), `"bernoulli"` (logistic regression for 0-1 outcomes), `"huberized"` (huberized hinge loss for 0-1 outcomes), classes), `"adaboost"` (the AdaBoost exponential loss for 0-1 outcomes), `"poisson"` (count outcomes), `"coxph"` (right censored observations), `"quantile"`, or `"pairwise"` (ranking measure using the LambdaMart algorithm). If quantile regression is specified, `distribution` must be a list of the form `list(name = "quantile", alpha = 0.25)` where `alpha` is the quantile to estimate. The current version's quantile regression method does not handle non-constant weights and will stop.

Extensions of Gradient Descent Boosting Approach

dmlc
XGBoost

Extreme Gradient Boosting

XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

https://arxiv.org/pdf/1603.02754.pdf
https://github.com/dmlc/xgboost

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

Gradient Boosting

Gradient boosting casts the problem into a gradient descent one: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

Pseudo-residuals: $-\nabla_{s_{l-1}} E(s_{l-1})(.)$

Step 1: The **pseudo-residuals** are set equal to the observations

Repeat (1:L):
a) Fit the weak learner to pseudo-residuals.
b) Compute the optimal step size of the weak learner.
c) Update the strong learner adding the weak learner.
d) Update the pseudo-residuals

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.)$$ where c_l's are coefficients and w_l's are weak learners

TensorFlow > Learn > TensorFlow Core > Tutorials

Boosted trees using Estimators

Contenido

https://www.tensorflow.org/tutorials/estimator/boosted_trees

https://arxiv.org/pdf/1710.11555.pdf

TF Boosted Trees: A scalable TensorFlow based framework for gradient boosting

Natalia Ponomareva, Soroush Radpour, Gilbert Hendry, Salem Haykal, Thomas Colthurst, Petr Mitrichev, Alexander Grushetsky

Google, Inc.
tfbt-public@google.com

Abstract. TF Boosted Trees (TFBT) is a new open-sourced framework for the distributed training of gradient boosted trees. It is based on TensorFlow, and its distinguishing features include a novel architecture, automatic loss differentiation, layer-by-layer boosting that results in smaller ensembles and faster prediction, principled multi-class handling, and a number of regularization techniques to prevent overfitting.

Gradient Boosting

Gradient boosting casts the problem into a gradient descent one: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

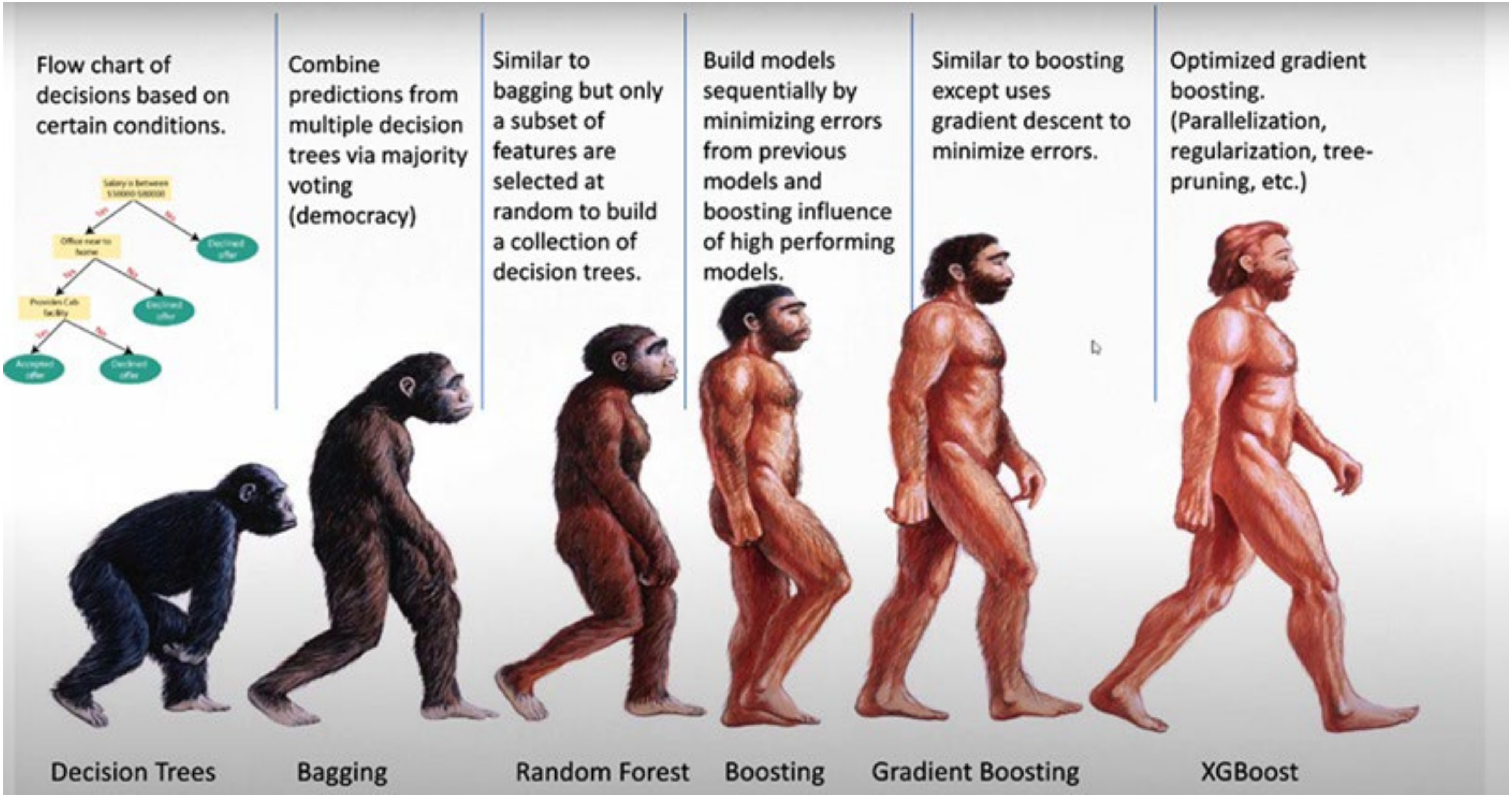
Pseudo-residuals: $-\nabla_{s_{l-1}} E(s_{l-1})(.)$

Step 1: The **pseudo-residuals** are set equal to the observations

Repeat (1:L):
a) Fit the weak learner to pseudo-residuals.
b) Compute the optimal step size of the weak learner.
c) Update the strong learner adding the weak learner.
d) Update the pseudo-residuals

Ensemble: Boosting Methods

Extensions of Gradient Descent Boosting Approach



Source: https://www.kaggle.com/code/rizkia14/extreme-gradient-boosting-xgboost