Applied Machine Learning
Lecture 6-1: Boosting

Selpi (selpi@chalmers.se)

The slides are further development of Richard Johansson's slides

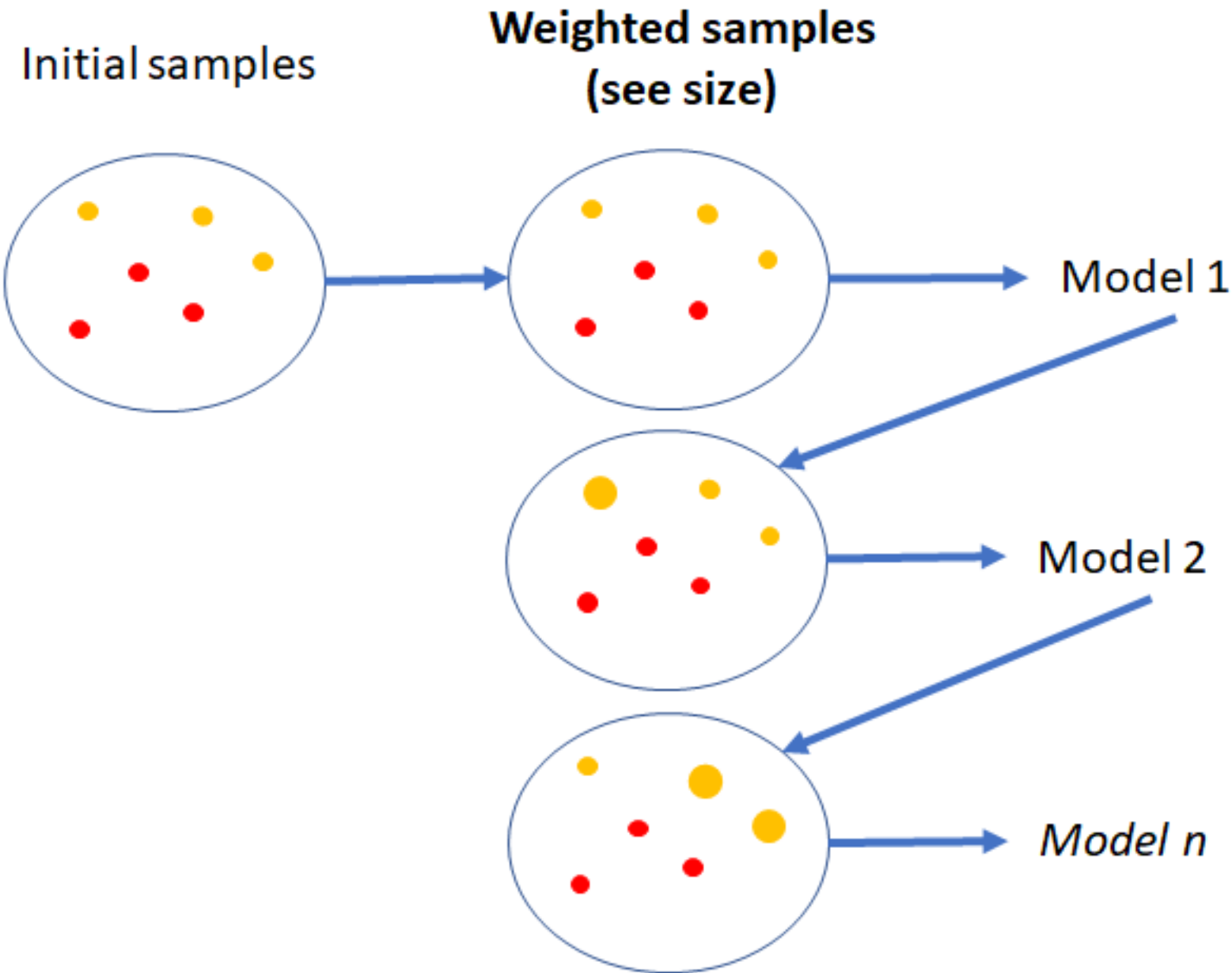
February 11, 2020

Overview

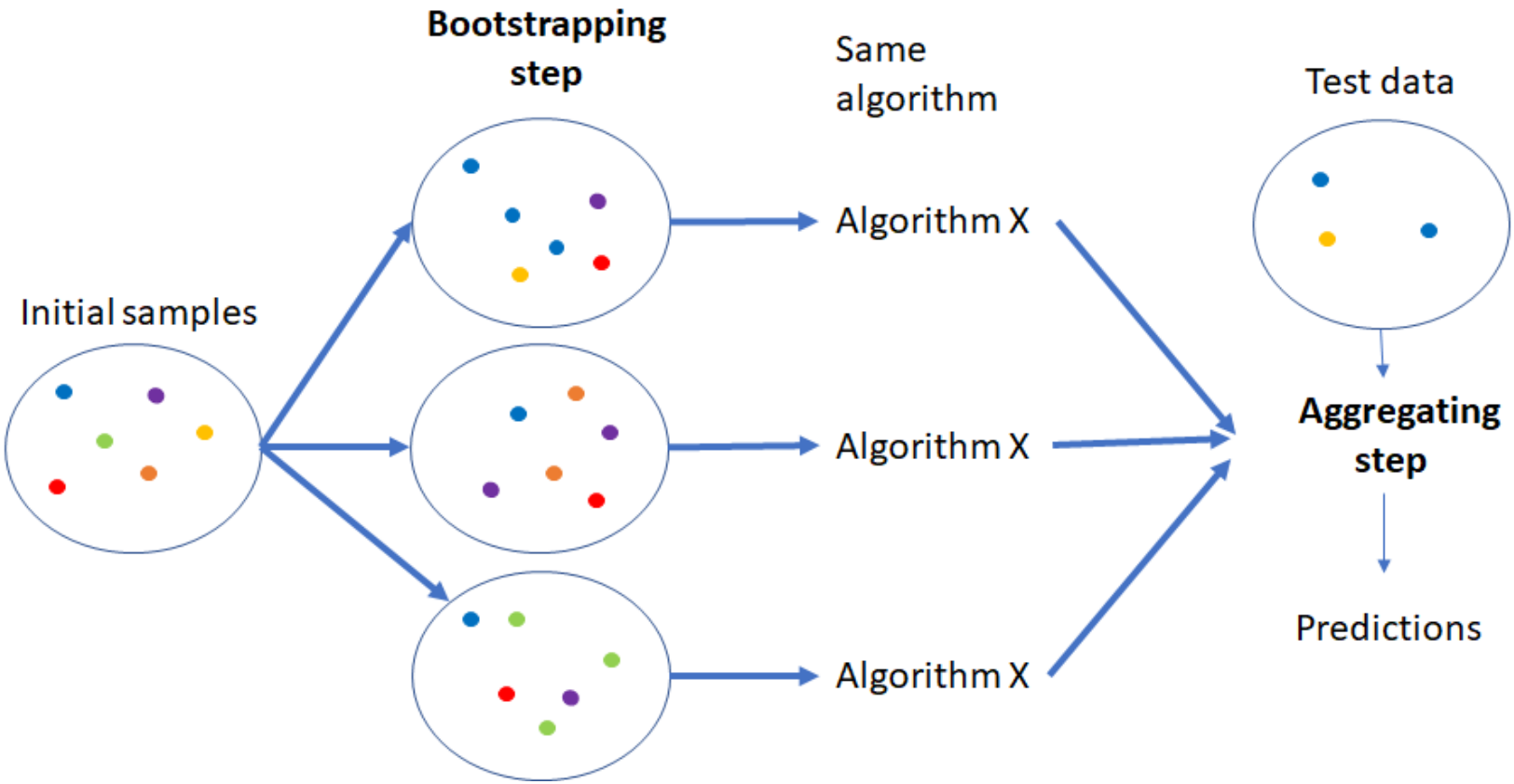
Boosting

What is boosting?

- Boosting is an ensemble technique



Discuss Bagging vs Boosting



toy regression example

adaptive boosting: AdaBoost

- **AdaBoost** is the most famous boosting algorithm
- after each iteration, check which instances are misclassified
  - then give misclassified instances a higher importance before the next iteration
- this can be applied using any base classifier that can handle weighted instances
  - typical choice: small decision trees ("decision stumps")

AdaBoost algorithm

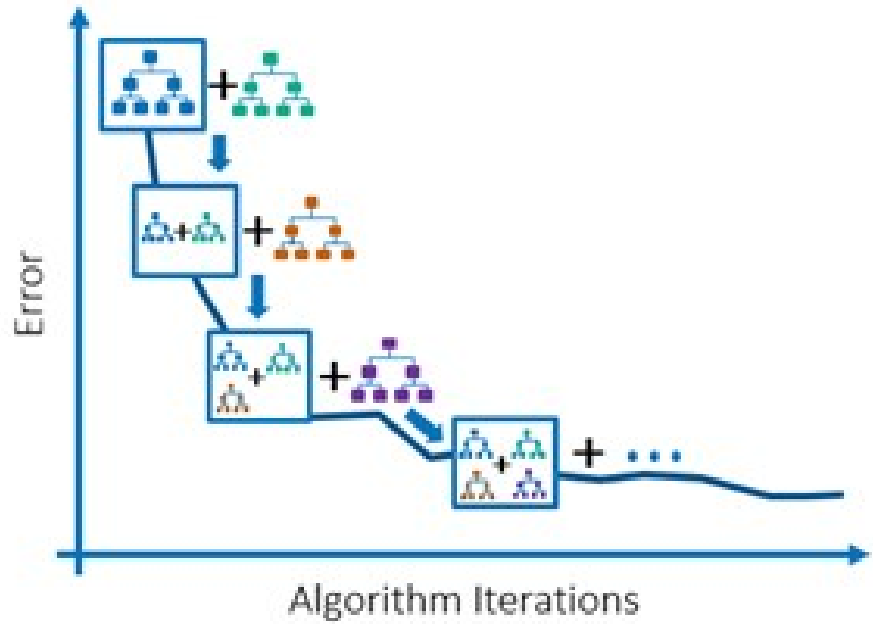
Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

[Hastie et al., *The Elements of Statistical Learning*]

gradient boosting

- **gradient boosting** generalizes AdaBoost
- idea: gradually add sub-classifiers to minimize a loss function
- it is based on a principle of gradient descent



[source]

gradient boosting, formally

let F_0 be a "dummy" constant model
for $m = 1, \ldots, M$
  for each pair $(\boldsymbol{x}_i, y_i)$ in the training set
    let r_i be the **pseudo-residual** $R(y_i, F_{m-1}(\boldsymbol{x}_i))$
  train a sub-model h_m on the pseudo-residuals
  create F_m by adding h_m to F_{m-1}
return F_M

- the **pseudo-residual** $R(y_i, F_{m-1}(\boldsymbol{x}_i))$ is defined as the gradient of the loss function

$$-\nabla_{\hat{y}} \text{Loss}(y, \hat{y})$$

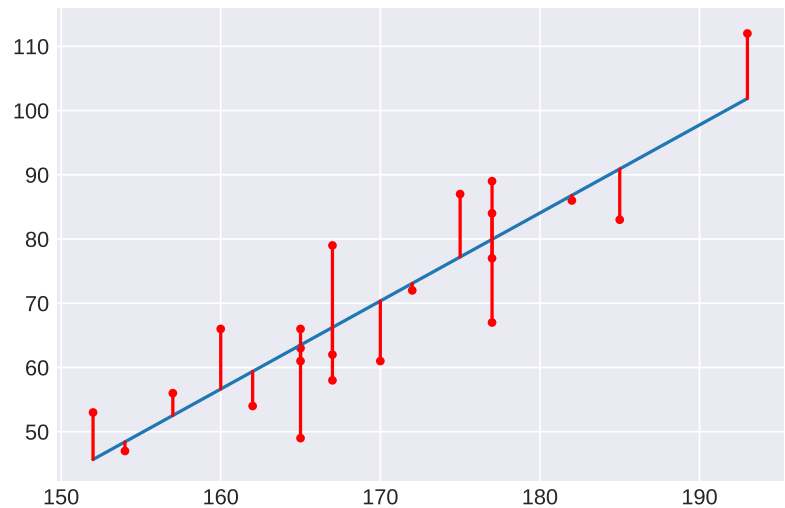
why "pseudo-residual"?

- for the squared error loss

$$\text{Loss}(y, \hat{y}) = (y - \hat{y})^2$$

the negative gradient is equal to (two times) the **residual**

$$-\nabla_{\hat{y}} \text{Loss}(y, \hat{y}) = 2(y - \hat{y})$$



tree ensembles in Kaggle

- GB (and other tree-based ensembles) often dominate in competitions for various prediction tasks, such as **Kaggle**
- why?
  - robust to preprocessing
  - easy to mix different types of features, nice for tabular data
  - easy to tune
- especially the software XGBoost is popular in Kaggle
- see also Fernández-Delgado et al. (2014) *Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?*
  - and this followup

in scikit-learn

- `sklearn.ensemble.GradientBoostingClassifier`
- `sklearn.ensemble.GradientBoostingRegressor`