Boosting, XGBoost and stacking

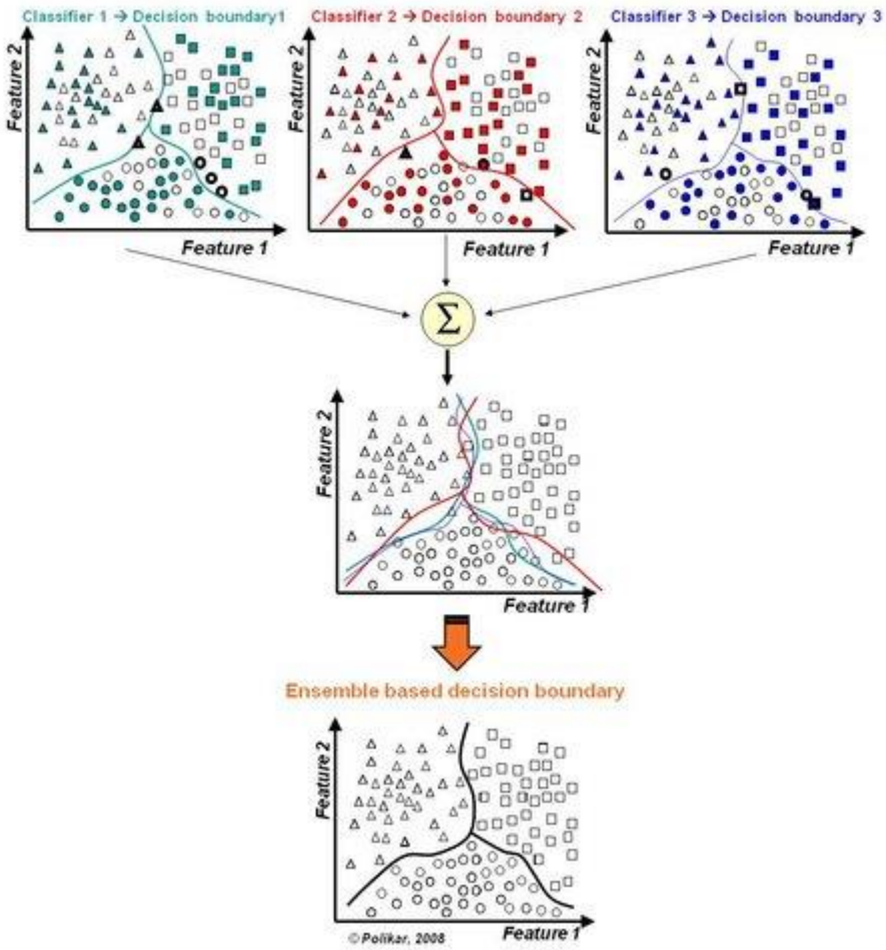
Taehoon Ko (taehoonko@snu.ac.kr)

목표

- 앙상블 방법 중 하나인 **부스팅 (Boosting)** 을 이해하고, 앞서 배운 **배깅과 부스팅의 차이**를 이해한다.
- 부스팅 트리 모델과 **XGBoost** 에 대해 이해한다.
- **스택킹 (Stacking)** 을 이해한다.

앙상블 (Ensemble)

머신러닝에서 앙상블이란 단일 모델이 아닌 여러 모델을 혼합하여 의사결정을 내리는 방법



https://music.ubc.ca/student-ensembles/
http://magizbox.com/training/machinelearning/site/ensemble/

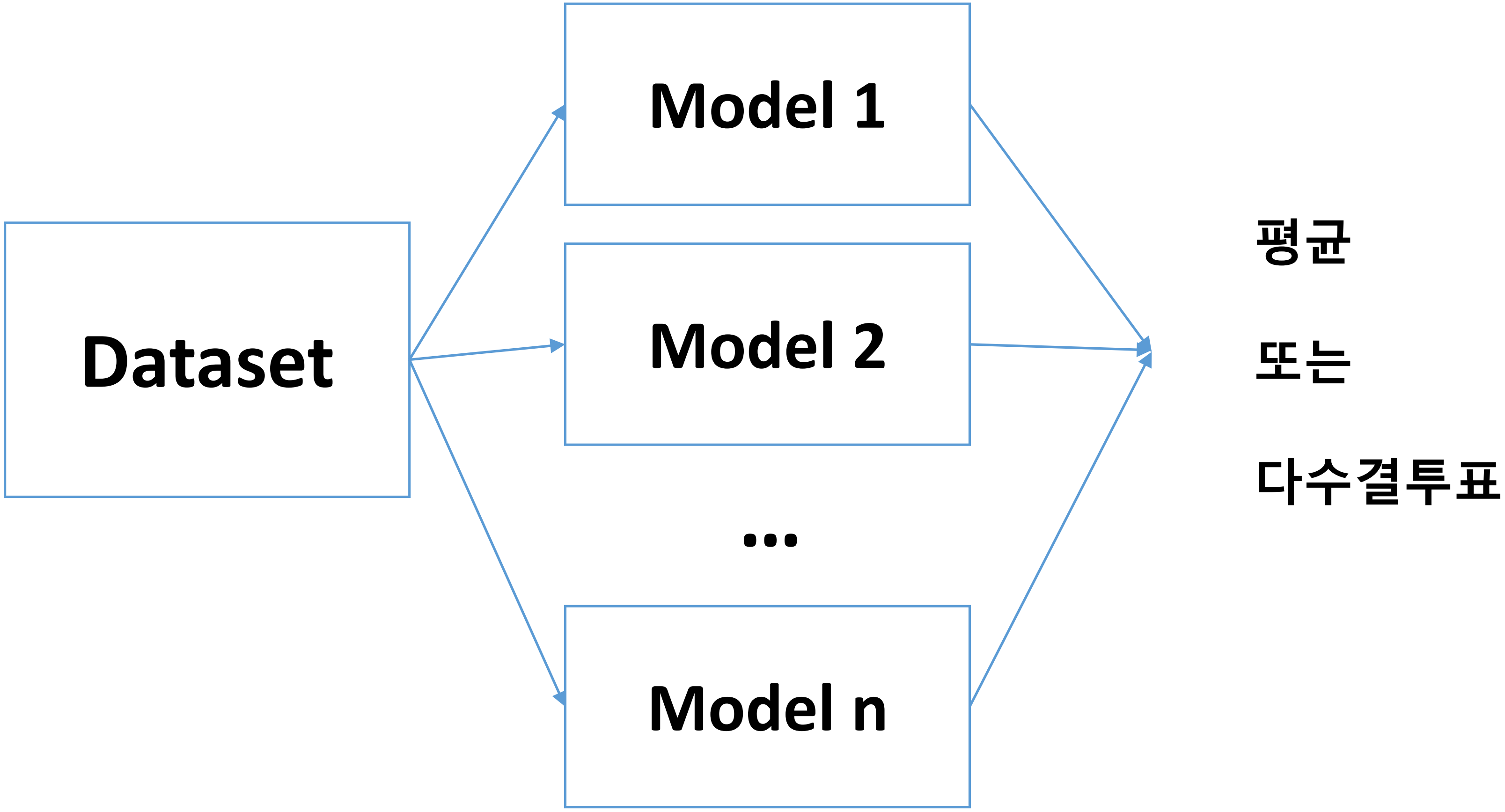
앙상블 (Ensemble)

여러 가지 유형의 앙상블 방법이 존재함

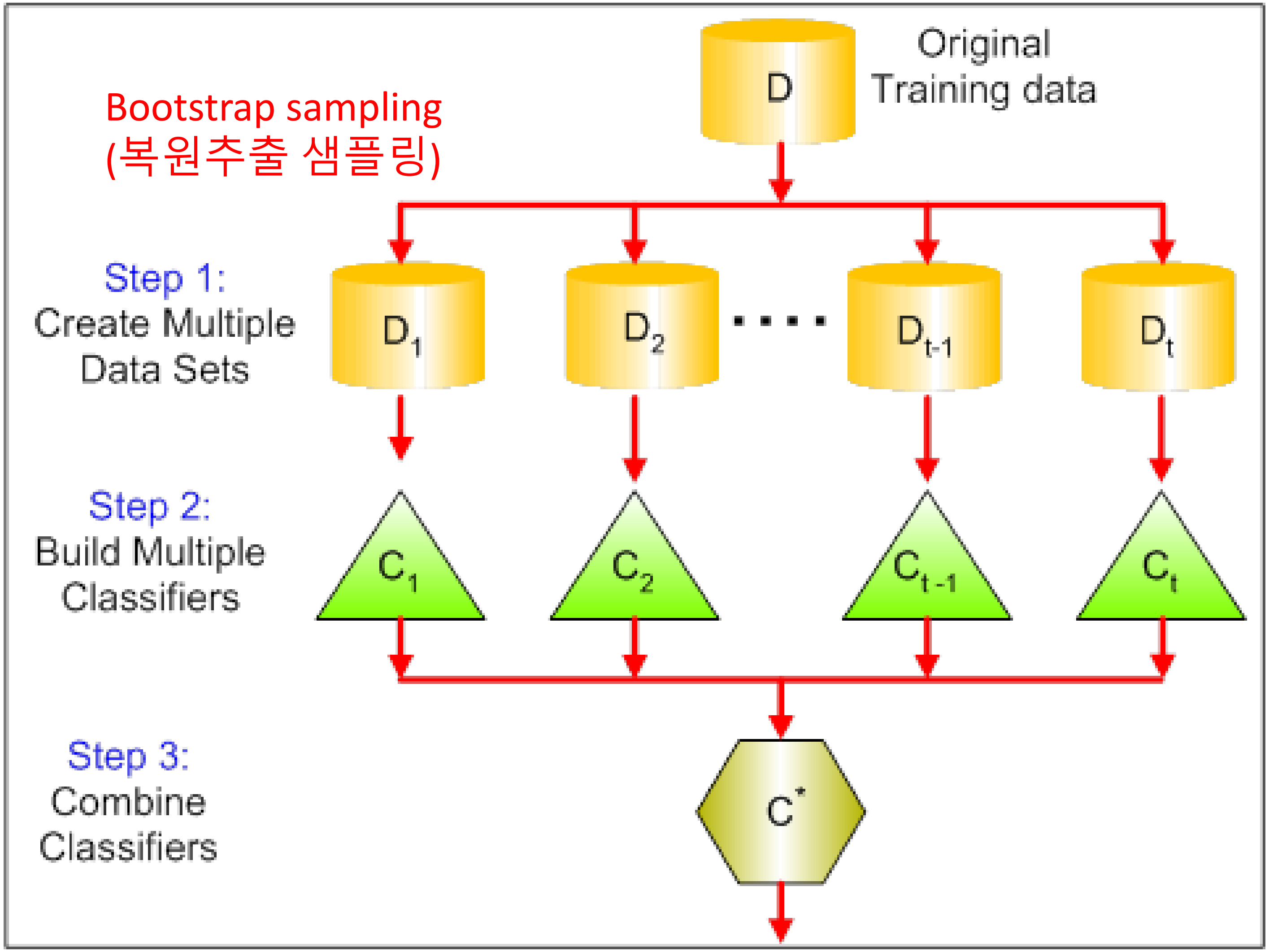
- **단순/가중 평균 (Simple/weighted average)**
- **배깅 (Bagging: Bootstrap aggregating)**
- **부스팅 (Boosting)**
- **스택킹 (Stacking)**
- **메타 학습 (Meta-learning)**
- **…**

단순 / 가중 평균

하나의 데이터셋에 여러 모델을 학습한 후, 각 모델들 결과의 평균 (혹은 다수결 투표) 으로 최종 산출



Bagging



https://www.analyticsvidhya.com/blog/2015/09/questions-ensemble-modeling/

Boosting a.k.a. additive training

- **Boosting combines weak "learners" into a single strong learner, in an iterative fashion.**

$$\hat{y}_i^{(0)} = 0$$

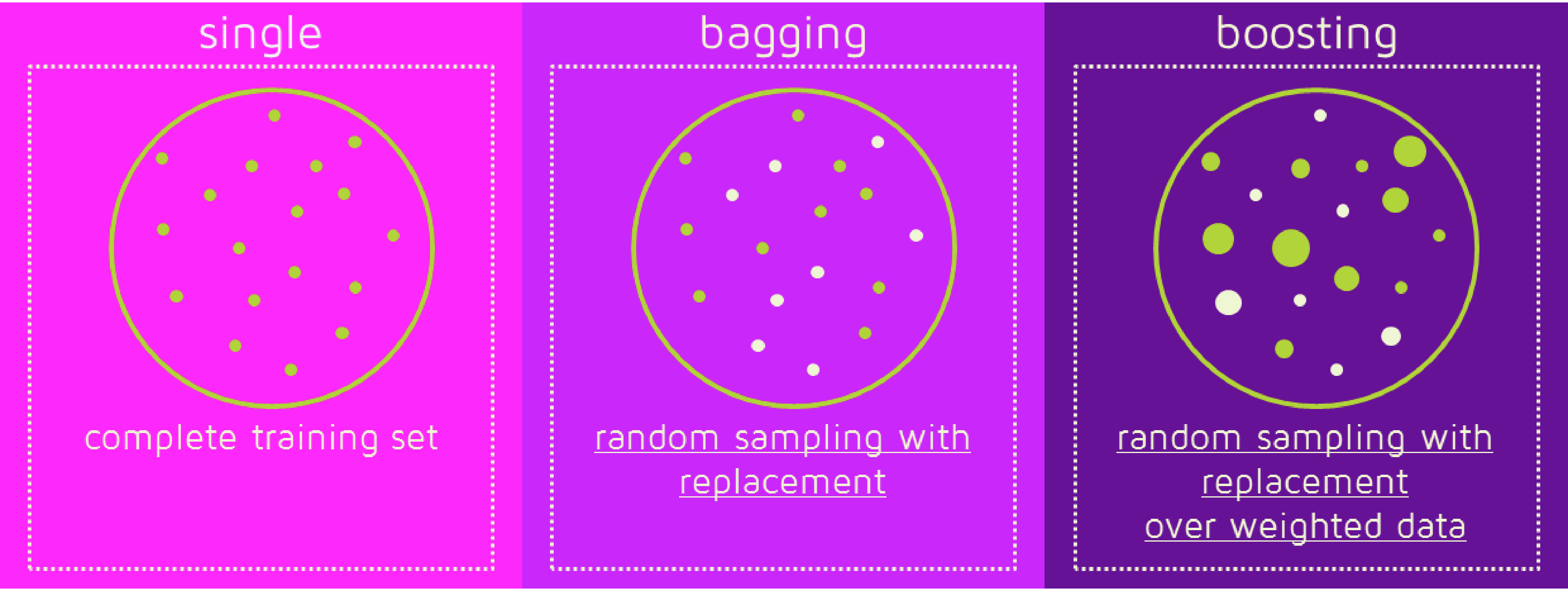
$$\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + f_1(x_i) = F_1(x_i)$$

$$\hat{y}_i^{(2)} = \hat{y}_i^{(1)} + f_2(x_i) = f_1(x_i) + f_2(x_i) = F_1(x_i) + f_2(x_i) = F_2(x_i)$$

...

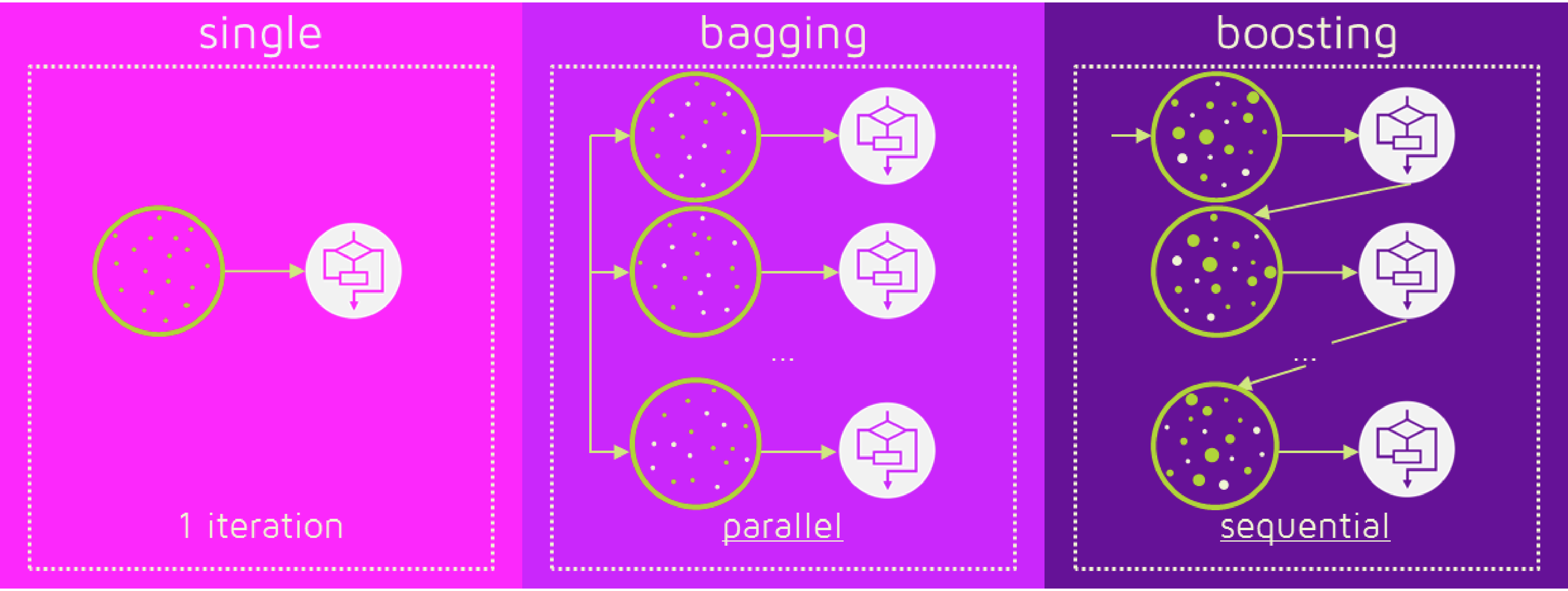
$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) = \sum_{k=1}^{t-1} f_k(x_i) + f_t(x_i) = F_{t-1}(x_i) + f_t(x_i) = F_t(x_i)$$

Bagging vs. Boosting



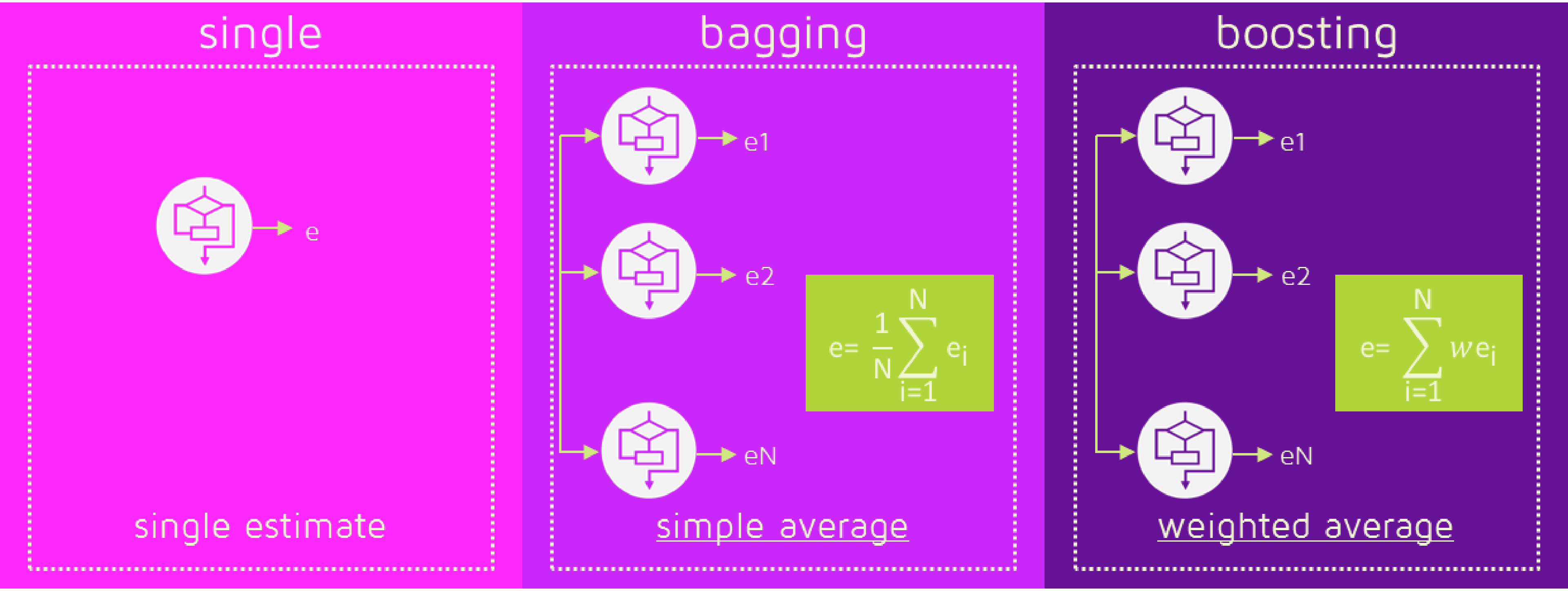
http://quantdare.com/2016/04/what-is-the-difference-between-bagging-and-boosting/

Bagging vs. Boosting



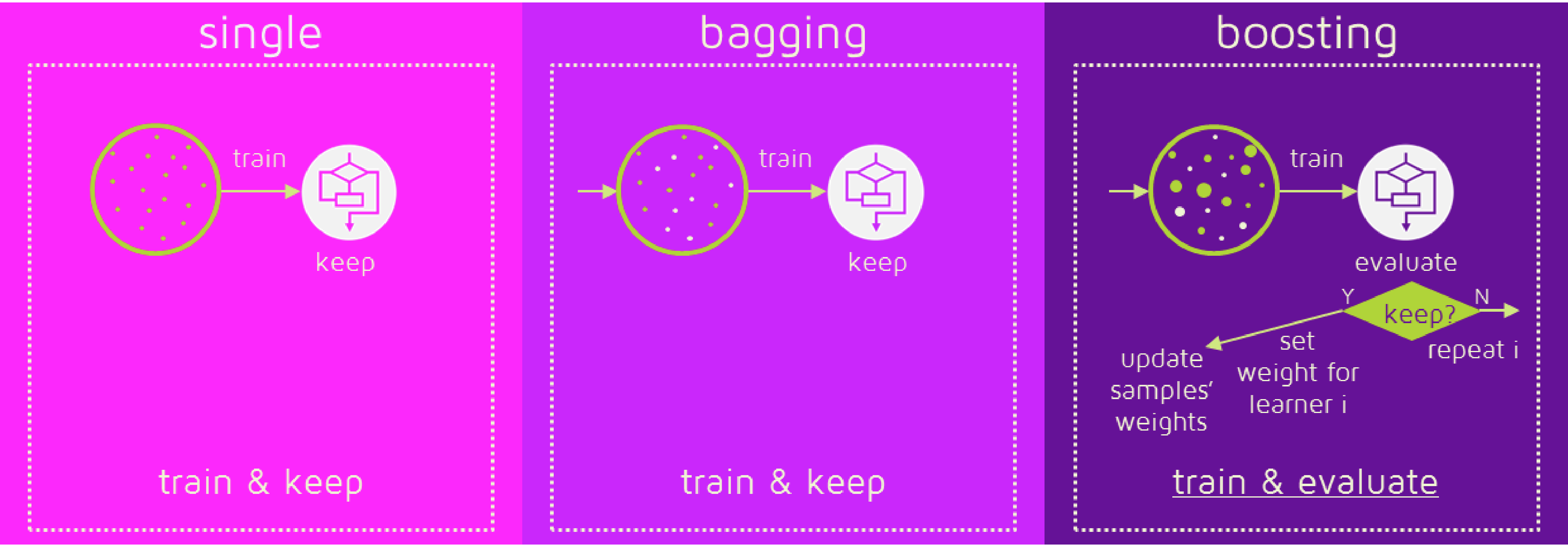
http://quantdare.com/2016/04/what-is-the-difference-between-bagging-and-boosting/

Bagging vs. Boosting



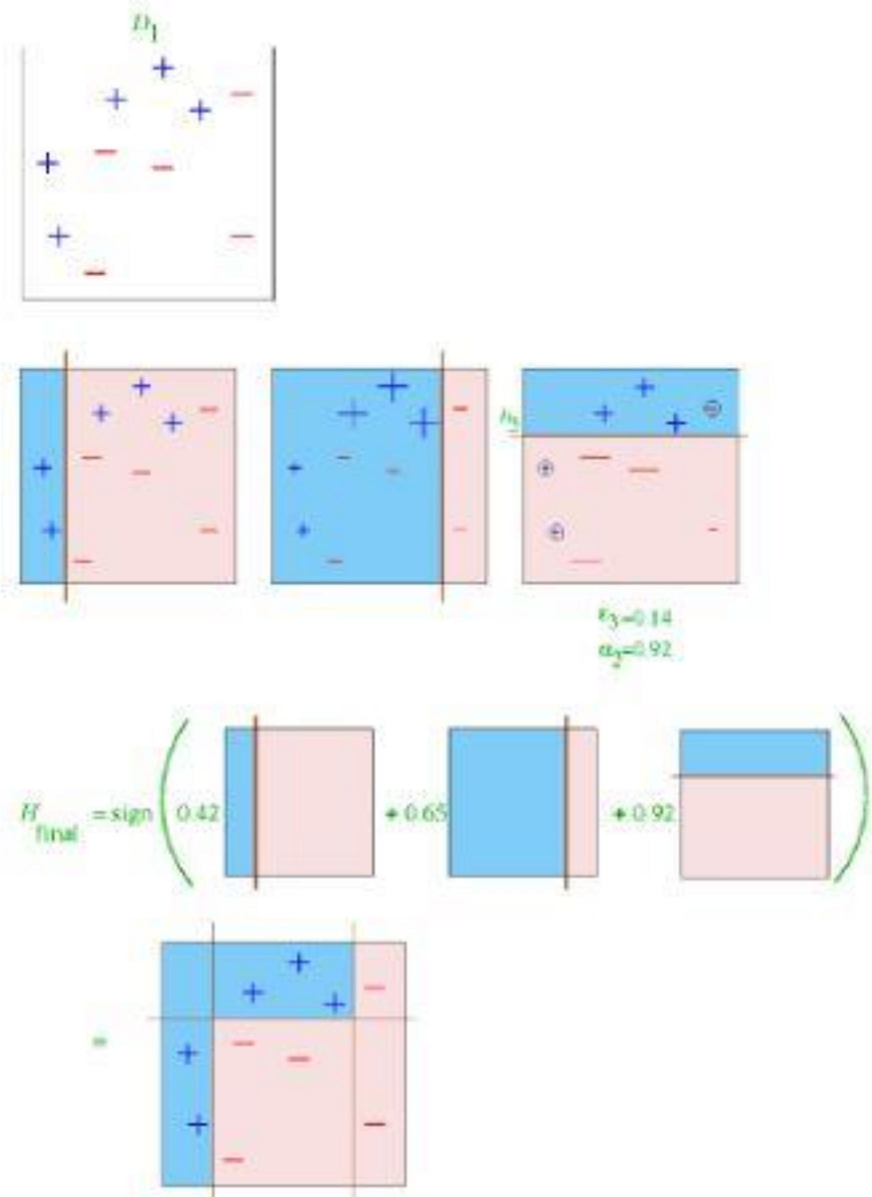
http://quantdare.com/2016/04/what-is-the-difference-between-bagging-and-boosting/

Bagging vs. Boosting

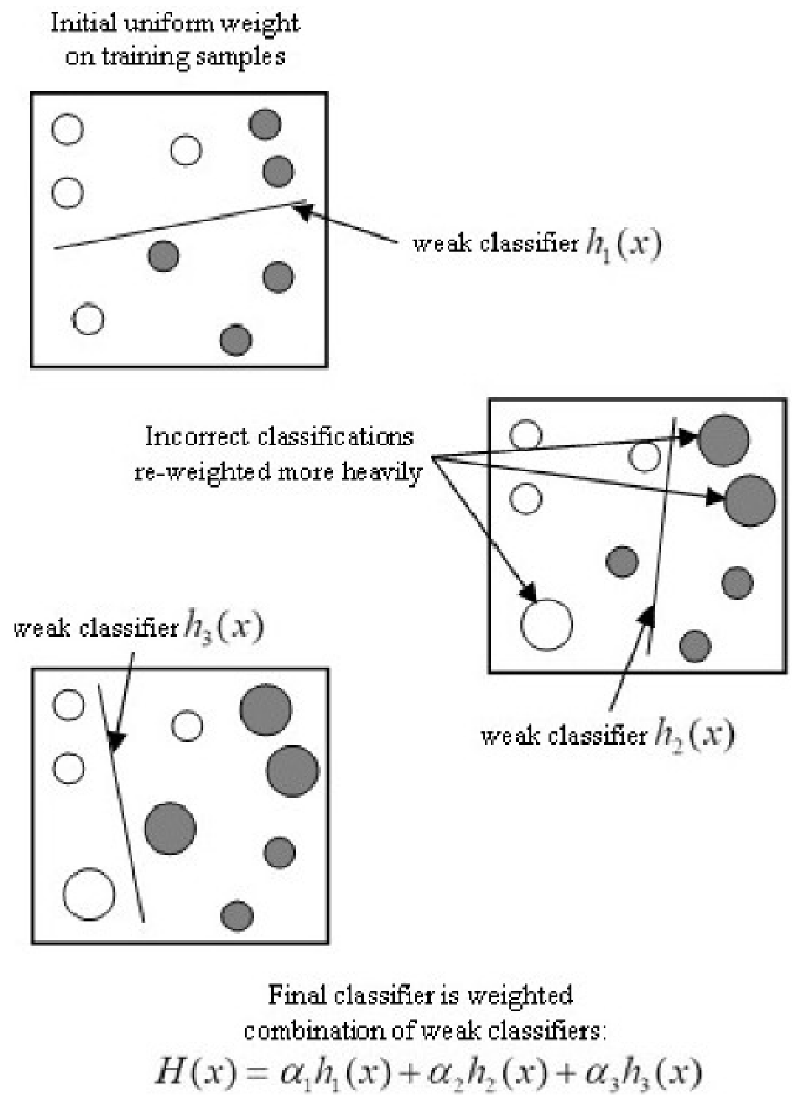


http://quantdare.com/2016/04/what-is-the-difference-between-bagging-and-boosting/

Adaboost



ε3=0.14
α3=0.92
H final = sign 0.42 + 0.65 + 0.92



Initial uniform weight
on training samples
weak classifier $h_1(x)$
Incorrect classifications
re-weighted more heavily
weak classifier $h_3(x)$
weak classifier $h_2(x)$
Final classifier is weighted
combination of weak classifiers:
$H(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)$

Gradient boosting

- Gradient boosting method assumes a real-valued y and seeks an approximation $\hat{F}(x)$ in the form of a weighted sum of functions $h_i(x)$.
- In the training phase, we should define loss function $L(y, F(x))$.

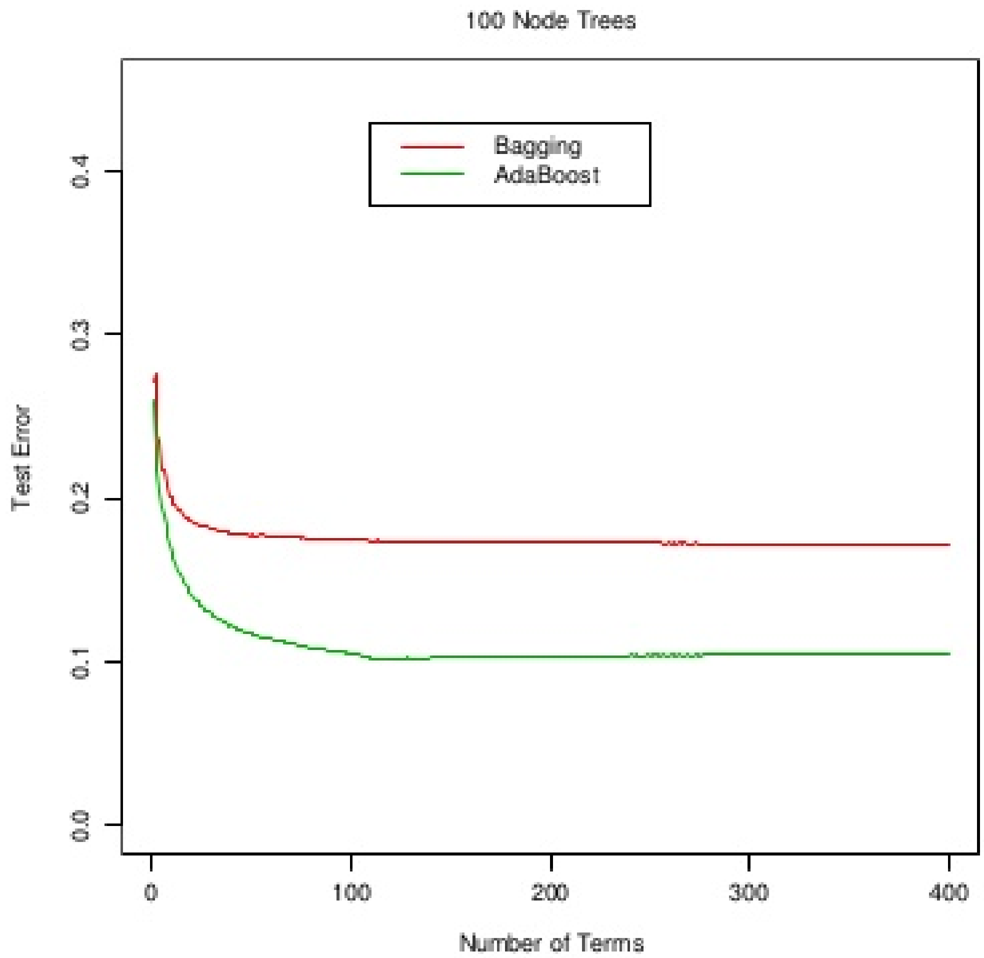
$$\hat{F}(x) = argmin_F \mathbb{E}_{x,y}[L(y, F(x)]$$

$$F_t(x) = F_{t-1}(x) + argmin_f \sum_{i=1}^{n} L(y_i, F_{t-1}(x_i) + f(x_i))$$

$$= F_{t-1}(x) - \gamma_t \sum_{i=1}^{n} \nabla_{F_{t-1}} L(y_i, F_{t-1}(x_i))$$

$$\gamma_t = argmin_\gamma \sum_{i=1}^{n} L\left(y_i, F_{t-1}(x_i) - \gamma \frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}\right)$$

Bagging vs. Boosting



Boosting vs Bagging

- 2000 points from Nested Spheres in R^{10}
- Bayes error rate is 0%.
- Trees are grown *best first* without pruning.
- Leftmost term is a single tree.

Bagging vs. Boosting

- **Many researchers think that boosting is better than bagging.**
  - Bagging: Strong learner를 평균화하여 모델링
    - Overfitting된 (bias가 낮은) 모델을 섞어서 variance를 낮추자.
  - Boosting: Weak learner를 순차적으로 학습하여 모델링
    - Underfitting된 (variance가 낮은) 모델을 섞어서 bias를 낮추자.
  - Bagging으로 만들어진 모델을 데이터에 overfitting될 가능성이 매우 높다.

- **왜 boosting이 bagging에 밀렸는가?**
  - 너무 많은 계산량
  - 튜닝해야 할 파라미터가 너무 많다.
  - bagging ➔ 분산 컴퓨팅 가능, boosting ➔ 분산 컴퓨팅이 어려움

XGBoost (eXtreme Gradient Boosting tree)

- **Gradient Boosting tree를 만들어내는 과정에서 근사적 알고리즘이 들어감**
- **더 많은 나무를 scalable하게 순차적으로 학습하는 것이 가능해짐**

- **Random Forest vs. XGBoost**
  - 둘 다 feature importance를 생성하는 것이 가능
  - 갑론을박 중
  - Random Forest는 큰 파라미터 튜닝 없이 좋은 성능이 나오는 것이 장점
  - XGBoost는 최적의 파라미터 튜닝이 이루어지면 Random Forest보다 좋은 성능이 나옴

Approximation in XGBoost: Weighted quantile sketch

- **기존 의사결정나무: 모든 후보군으로 split한 후, 이를 평가하여 split point를 결정**
- **XGBoost: 후보군을 근사적으로 (혹은 대충) 정해놓은 다음에 이를 평가하여 split point를 결정**
  - XGBoost가 제안되기 이전에도 사용되던 방법이나, 이 근사법이 어느 정도의 정확도 수준을 유지해준다는 수학적 증명은 XGBoost 논문이 처음
  - Weak learner를 학습한다는 측면에서 문제 없음

Algorithm 1: Exact Greedy Algorithm for Split Finding

```
Input: I, instance set of current node
Input: d, feature dimension
gain ← 0
G ← Σ_{i∈I} g_i, H ← Σ_{i∈I} h_i
for k = 1 to m do
    G_L ← 0, H_L ← 0
    for j in sorted(I, by x_jk) do
        G_L ← G_L + g_j, H_L ← H_L + h_j
        G_R ← G − G_L, H_R ← H − H_L
        score ← max(score, G_L^2/(H_L+λ) + G_R^2/(H_R+λ) − G^2/(H+λ))
    end
end
Output: Split with max score
```

Algorithm 2: Approximate Algorithm for Split Finding

```
for k = 1 to m do
    Propose S_k = {s_k1, s_k2, ··· s_kl} by percentiles on feature k.
    Proposal can be done per tree (global), or per split(local).
end
for k = 1 to m do
    G_kv ←= Σ_{j∈{j | s_k,v ≥ x_jk > s_k,v−1}} g_j
    H_kv ←= Σ_{j∈{j | s_k,v ≥ x_jk > s_k,v−1}} h_j
end
Follow same step as in previous section to find max
score only among proposed splits.
```

Approximation in XGBoost: Sparsity-aware algorithm

- **In training phase, XGBoost considers the sparsity pattern in the data.**
- **What makes out data sparse?**
  - presence of missing values in the data
  - frequent zero entries in the statistics
  - artifacts of feature engineering such as one-hot encoding
- **Idea**
  - If an instance x has a missing value, just let it go to the next node through 'default' direction.

Data

Example	Age	Gender
X1	?	male
X2	15	?
X3	25	female

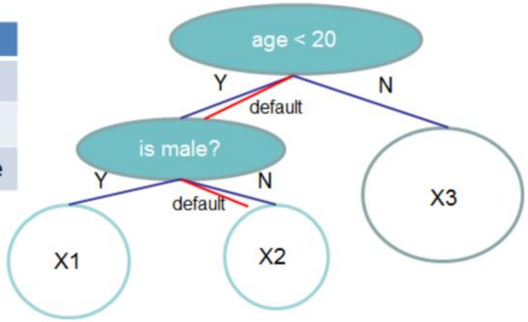


Figure 4: Tree structure with default directions. An example will be classified into the default direction when the feature needed for the split is missing.

Approximation in XGBoost: Sparsity-aware algorithm

- **By this algorithm, we can add a default direction in each tree node.**

Algorithm 3: Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ *to* m **do**
  // *enumerate missing value goto right*
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I_k,$ *ascent order by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // *enumerate missing value goto left*
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*$(I_k,$ *descent order by* $\mathbf{x}_{jk})$ **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

System design for scalability in XGBoost

- **Column black for parallel learning**
- **Cache-aware access**
- **Blocks for out-of-core computation**

- **XGBoost가 Scikit-learn에 있는 GradientBoostingClassifier보다 속도가 빠르다고 한다. ➔ 저자가 논문 제목에 "Scalable"을 넣은 이유**

- **참고**
  - XGBoost: A scalable tree boosting system - http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf
  - Xgboost: Supplementary meterial - http://homes.cs.washington.edu/~tqchen/pdf/xgboost-supp.pdf
  - Introduction to boosted trees - https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

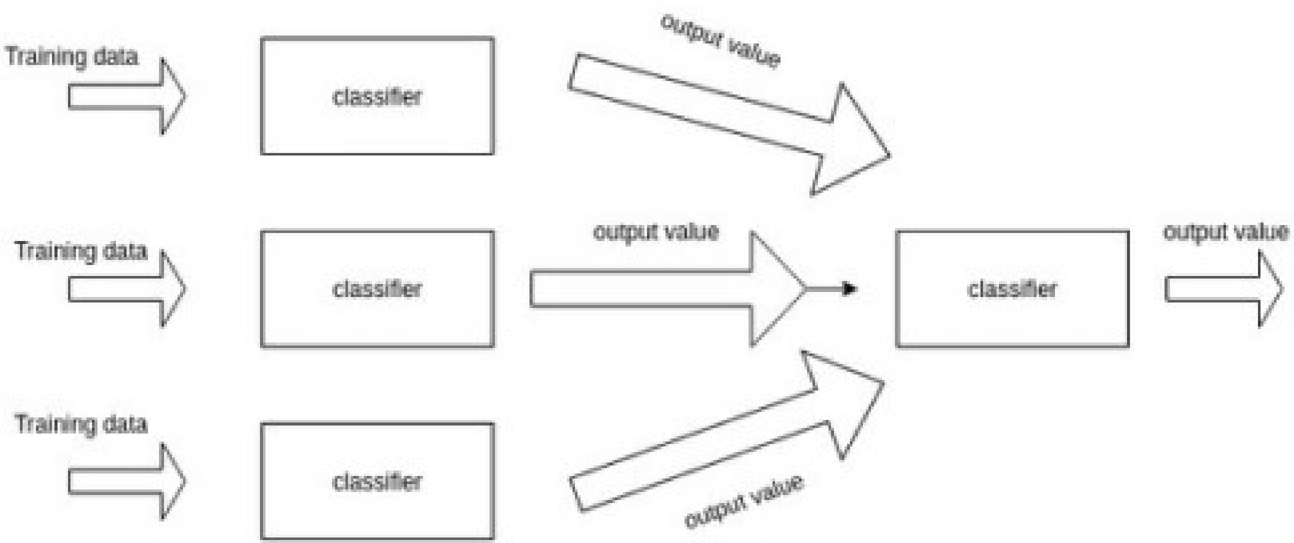
How to install XGBoost package

- **Linux, OS X and Windows 공통**
  - Github에서 XGBoost 소스를 가져온 후, compile 수행
  - Compile 수행 후, 설치 안내대로 python package 설치
  - 자세한 사항은 https://xgboost.readthedocs.io/en/latest/build.html 에서 [Build the Shared Library]와 [Python Package Installation] 참고

- **Only Windows**
  - "Gohlke's unofficial Windows binaries for Python extension packages"로부터 자신의 환경에 맞는 whl 파일을 다운로드 후, pip install xgboost-OOOO.whl
  - http://www.lfd.uci.edu/~gohlke/pythonlibs/#xgboost

Stacking (or stacked generalization)

- Stacking (sometimes called *stacked generalization*) involves training a learning algorithm to combine the predictions of several other learning algorithms.
- Stacking typically yields performance better than any single one of the trained models.
- Netflix challenge 이후부터 많은 machine learning competition에서 사용되기 시작한 방법



http://www.kdnuggets.com/2016/11/data-science-basics-intro-ensemble-learners.html

Stacking (or stacked generalization)

- http://blog.kaggle.com/2016/04/08/homesite-quote-conversion-winners-write-up-1st-place-kazanova-faron-clobber/