Model Ensemble

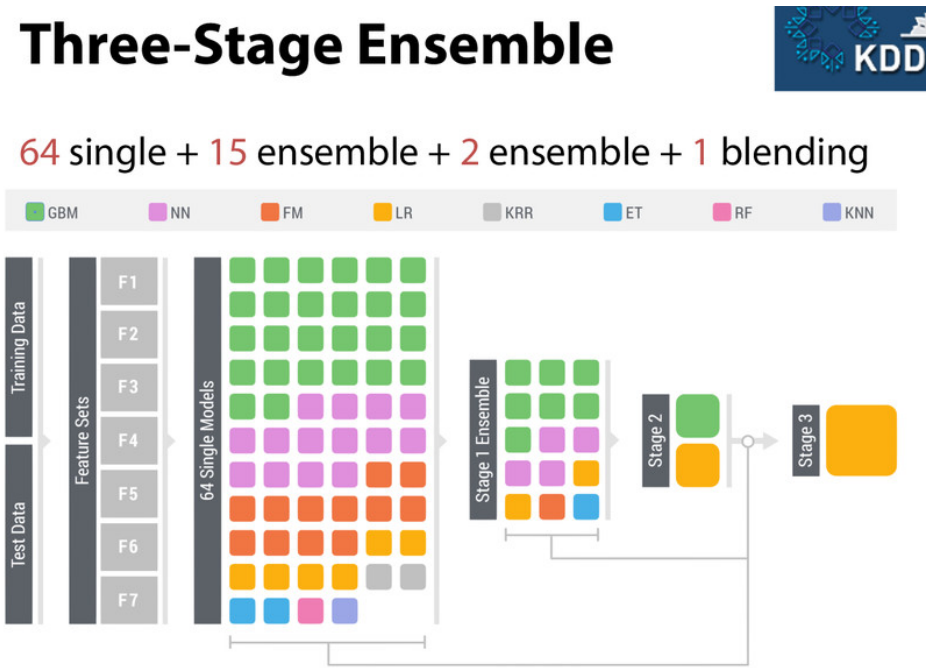
Makoto Yamada, Hisashi Kashima
myamada@i.kyoto-u.ac.jp

Kyoto University

July/1/2019

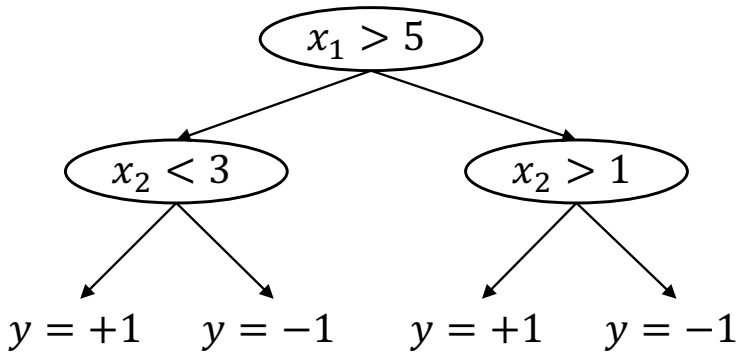
Model ensemble: Combining different models to improve performance

- One model cannot fit all
- Combine different predictors to improve performance
- Commonly used technique in predictive modeling competitions (e.g., Kaggle)



Decision trees: An off-the-shelf non-linear predictor

Decision tree: Build a classifier by hierarchically dividing input space.



Pros

- Relatively fast (can scale with respect to the number of samples)
- High interpretability

Cons

- Instable (like other non-linear models)
- Sensitive to noise and hyper-parameters

Variants: regression trees, piecewise linear prediction, etc.

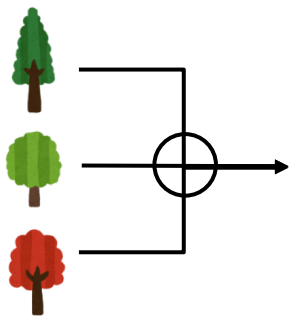
Two ways of ensemble: Horizontal ensemble and vertical ensemble

Horizontal ensemble

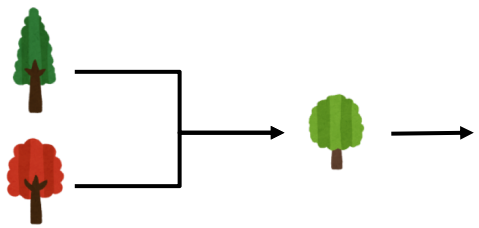
- Construct a set of models in parallel
- Integrate their outputs to make final predictions
- E.g., bagging, boosting

Vertical ensemble

- Make a cascade of models
- Outputs of ℓ-th level models are used as inputs of $\ell + 1$-th level models
- E.g., Stacking, "deep" neural networks.



Horizontal Ensemble



Vertical Ensemble

Parallel ensemble methods: Bagging and boosting

Bagging

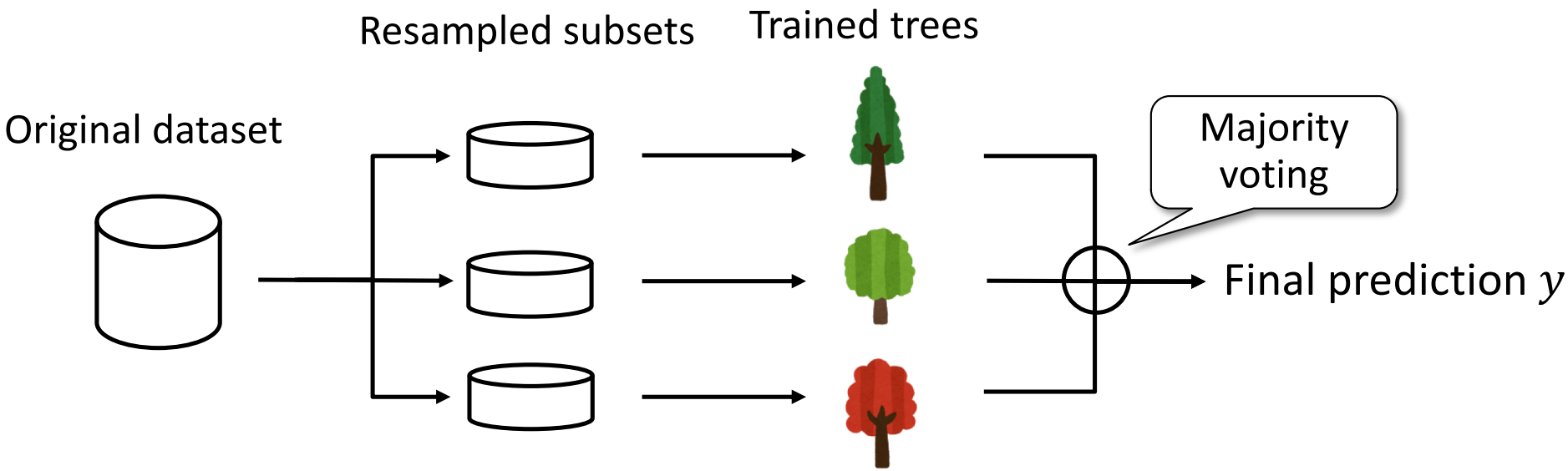
- Simple ensemble method based on data resampling
- Random forest is often "the first choice"
- Easily parallelizable

Boosting

- Adaptive version of bagging
- Gradient boosted decision trees (GBDT)
- XGBoost is the regular winner in competitions (e.g., Kaggle)

Bagging: Majority voting with different predictors

- Decision trees are sensitive to data change
- Bootstrapping: Train multiple classifiers using randomly resampled subsets of the original dataset
- Majority voting to aggregate predictions
- Stable, but lost interpretability

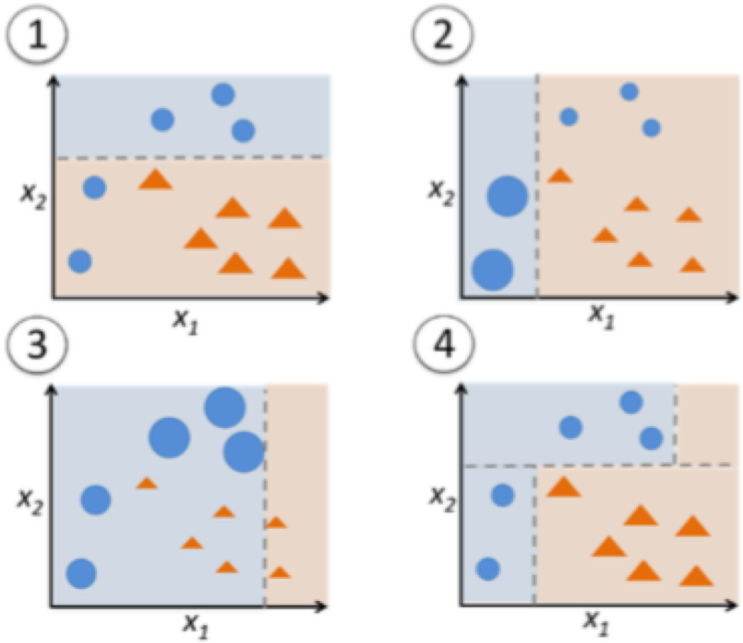


Random forest: Also uses feature resampling for diversity

- Randomly resample not only the data but also features
- Example: 5 samples $\{1, 2, 3, 4, 5\}$ and 4 features $\{A, B, C, D\}$
  - 1st tree trained with 2 features $\{A, B\}$ on subsamples $\{1, 3, 5\}$
  - 2nd tree trained with $\{A, B, C, D\}$ on subsamples $\{1, 2, 3\}$
  - 3rd tree trained with $\{A, C, D\}$ on subsamples $\{2, 3, 4, 5\}$
  - ...
- Off-the-shelf non-linear model: stable and high performance

Boosting: Adaptive resampling to difficult examples

- In bagging, all models are independently trained using uniformly-random resamples
- In boosting, the next model is trained focusing on the "difficult" data that the current model cannot correctly classify
- Very nice introduction of bagging, booting, and random forest `https://sebastianraschka.com/faq/docs/bagging-boosting-rf.html`



Gradient boosting: Incremental addition of new model to current ensemble

- Loss function of the current model $L(F_{t-1}) = \sum_{i=1}^{n} \ell^{(i)}(F_{t-1})$
  - $F_{t-1} = \sum_{\tau=1}^{t-1} \alpha_\tau f_\tau$. f_τ: τ-th component model, α_τ: weight.
- Loss function of the updated ensemble $L(F_{t-1} + \alpha_\tau f_\tau)$.
- Find α_τ and f_τ that minimizes L
- Taylor expansion (first order approximation):

$$L(F_{t-1} + \alpha_t f_t) = \sum_{i=1}^{n} \ell^{(i)}(F_{t-1}) + \sum_{i=1}^{n} \frac{\partial \ell^{(i)}(F_{t-1})}{\partial F_{t-1}} \alpha_t f_t + \cdots$$

XGBoost: A popular boosting implementation

- Implementation using gradient boosting + decision tree (Often appears in top ranked methods in competition)
- Training with constraints on the number of decision tree leaves and the total weights
- Parallel implementation `https://www.quora.com/What-machine-learning-approaches-have-won-most-Kaggle-com`

Model stacking: Vertical ensemble method

Stacking: vertical ensemble method

- is similar to the multi-layer neural network (Neural Network is a stacked linear classification models)
- but can have heterogeneous components
- Outputs of ℓ-th level models are used as inputs to $\ell + 1$-th level models
  - Outputs of 0-th level models $\boldsymbol{y}_0$ is original feature vector $\boldsymbol{x}$
  - Outputs of ℓ-th level models $\boldsymbol{y}_\ell$
  - inputs to $\ell + 1$-th level models

$$\boldsymbol{x}_{\ell+1} = \begin{bmatrix} \boldsymbol{x}_\ell \\ \boldsymbol{y}_\ell \end{bmatrix}$$

Difficulty in model stacking: An easy solution is biased

How can we train stacked models?
An easy solution:

- Train a classifier f using the training dataset L
- add the prediction values of f as a new feature
- ...

this seems to be working... but actually does NOT!

Remember the first principle: you cannot make a prediction for the data you used in the training! Easily overfitting to the data

The prediction value to the training data are biased because your model has been trained to reproduce the labels

How to stack: Use cross-validation to extend features

Divide a given dataset into K non-overlapping sets

- Use $K-1$ of them for training a model
- Use the model to add a new feature to the remaining set
- Doing steps 1 and 2 for K hold out sets gives the new feature for the whole dataset

Train the level-2 predictor using the extended dataset

Finally, the level-1 predictor is (re-)trained using the original whole dataset

Because the extended feature for training the level-2 predictor is produced by different level-1 predictors

Summary

Summary of this class

- Horizontal ensemble (Bagging, boosting)
- Vertical ensemble (Stacking, "deep" neural networks)