Stacking

Ensemble models: **recap**

- **Bagging** : train models in parallel on bootstrapped data and aggregate
- **Boosting** : train models sequentially taking the residuals of the last model into account

CAMBRIDGE SPARK

Ensemble models: **recap**

- **Bagging** : train models in parallel on bootstrapped data and aggregate
- **Boosting** : train models sequentially taking the residuals of the last model into account

Both usually work on a fixed family of model (e.g.: fixed depth DT)

→ random forest, extra trees, gradient boosting etc.

The idea behind stacking

Take the output of a model as a new feature.

- Train a few different models $h_1, \ldots, h_K$

The idea behind stacking

Take the output of a model as a new feature.

- Train a few different models $h_1, \ldots, h_K$
- Take the outputs $\hat{y}_1, \ldots \hat{y}_K$ and form a new feature matrix Z of size $n \times K$

CAMBRIDGE SPARK

The idea behind stacking

Take the output of a model as a new feature.

- Train a few different models $h_1, \ldots, h_K$
- Take the outputs $\hat{y}_1, \ldots \hat{y}_K$ and form a new feature matrix Z of size $n \times K$
- Train a model with Z as feature matrix and y as response

Learn the aggregation rule for an arbitrary set of base models

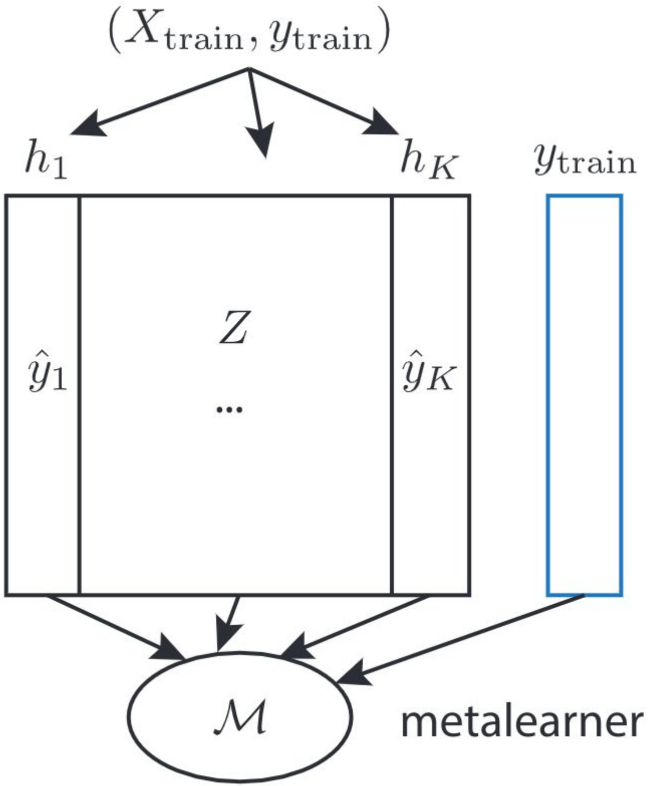
CAMBRIDGE SPARK

Definitions for stacking

- **level-0 data**: the original feature matrix X_{train} for training
- **level-0 learners**: the set of K models fitted on (X_{train}, y_{train})
- **level-1 data**: the derived feature matrix Z
- **level-1 model** or **metalearner**: the model fitted on (Z, y_{train})

CAMBRIDGE SPARK

Stacking, visually



Applying the stacked model

For a point x in the test set:

- Form the vector z with $z_k = h_k(x)$ for $k = 1, \dots, K$
- Predicted response to return: $\mathcal{M}(z)$

Stacking in practice

- **Complete flexibility** for level-0 classifiers
  - Models trained on subsets of the feature (e.g.: only numerical features)
  - Ensemble models, . . .

CAMBRIDGE SPARK

Stacking in practice

- **Complete flexibility** for level-0 classifiers
    - Models trained on subsets of the feature (e.g.: only numerical features)
    - Ensemble models, . . .
- **Complete flexibility** for the metalearner
    - Linear or logistic regression
    - Tree classifier or regressor
    - Ensemble models or even a small neural network. . .

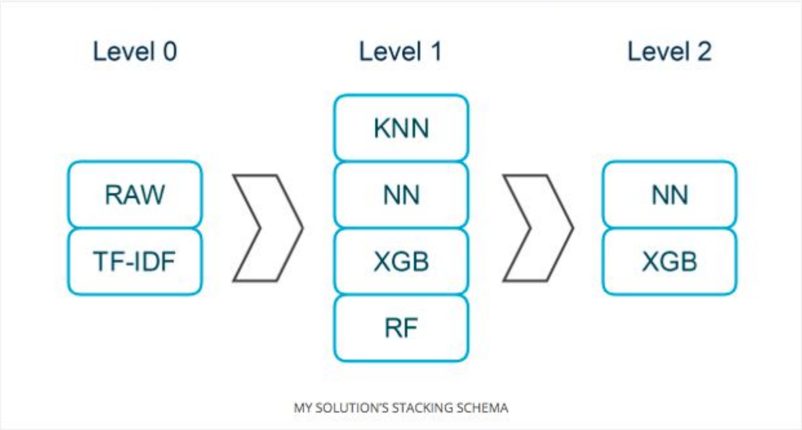
Can do level 2 but it gets quite complex/expensive

CAMBRIDGE SPARK

Meanwhile on Kaggle. . .

Description of a 2nd place model for a 2015 contest:

What preprocessing and supervised learning methods did you use?



- L0 = our feature engineering (tf-idf is a technique for text)
- L1 = our level 0 and L2 = our level 1
- Level 2 not specified (aggregation of NN and XGB)

CAMBRIDGE SPARK

Stacking with cross validation

Usually, an extra step is involved to avoid overfitting:

- partition X_{train} in *T folds*, let $X_{train}^{-\ell}$ the data without fold ℓ

Stacking with cross validation

Usually, an extra step is involved to avoid overfitting:

- partition X_{train} in *T folds*, let $X_{train}^{-\ell}$ the data without fold ℓ
- for each model h_k with $k = 1, \dots, K$:
  - train the model on $X_{train}^{-\ell}$ and predict on fold ℓ: $\hat{y}_k^{\ell}$
  - aggregate $\hat{y}_k^{1}, \dots y_k^{T}$ to form $\hat{y}_k$

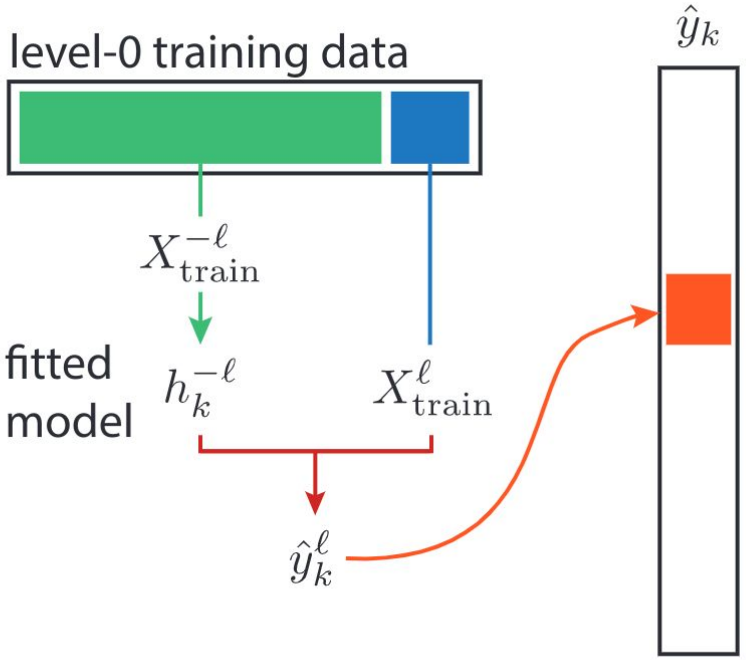
CAMBRIDGE SPARK

Stacking with cross validation

Usually, an extra step is involved to avoid overfitting:

- partition X_{train} in *T folds*, let $X_{train}^{-\ell}$ the data without fold ℓ
- for each model h_k with $k = 1, \dots, K$:
  - train the model on $X_{train}^{-\ell}$ and predict on fold ℓ : $\hat{y}_k^{\ell}$
  - aggregate $\hat{y}_k^{1}, \dots y_k^{T}$ to form $\hat{y}_k$
- build Z with the $\hat{y}_1, \dots, \hat{y}_K$ and proceed as before

Stacking with cross validation



CAMBRIDGE SPARK

Stacking in SkLearn

- There is no tool to directly do stacking in SkLearn
- Another library called **mlxtend** can be used for this

```
from mlxtend.classifier import StackingCVClassifier
```

Hands-on session

stacking.ipynb