I2ML :: CHEAT SHEET

The **I2ML**: Introduction to Machine Learning course offers an introductory and applied overview of "supervised" Machine Learning. It is organized as a digital lecture.

Bagging Ensembles

Basic Idea:

- Bagging is short for **B**ootstrap **Agg**regation.
- An **ensemble method** that combines many models into one big "meta-model".
- The constituent models of an ensemble are called **base learners** (BLs).
- Homogeneous ensembles, i.e., all base learners are of the same type.

Bagging improve performance by reducing the variance of predictions, but (slightly) increases the bias: training data is reused, so small mistakes can get amplified. Works best if BLs' predictions are only weakly correlated.

Bagging Training:

Base learners , $m = 1, \dots, M$ are trained on M **bootstrap** samples of training data $\mathcal{D}$.

Algorithm 1 Bagging algorithm: Training

1: Input: Dataset $\mathcal{D}$, type of BLs, number of bootstraps M
2: **for** $m = 1 \to M$ **do**
3: Draw a bootstrap sample $\mathcal{D}^{[m]}$ from $\mathcal{D}$
4: Train BL on $\mathcal{D}^{[m]}$ to obtain model $\hat{b}^{[m]}$
5: **end for**

Bagging Aggregating:

Aggregate the predictions of M fitted BLs to get the ensemble model:

$$\hat{f}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{f}^{[m]}(\mathbf{x}) \right) \text{ (regression / decision score, use } \hat{f}_k \text{ in multi-class)}$$

$$\hat{h}(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \sum_{m=1}^{M} \mathbb{I}\left(\hat{h}^{[m]}(\mathbf{x}) = k \right) \text{ (majority voting)}$$

$$\hat{\pi}_k(\mathbf{x}) = \begin{cases} \frac{1}{M} \sum_{m=1}^{M} \hat{\pi}_k^{[m]}(\mathbf{x}) & \text{(probabilities through averaging)} \\ \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}\left(\hat{h}^{[m]}(\mathbf{x}) = k \right) & \text{(probabilities through class frequencies)} \end{cases}$$

Random Forests

Introduction:

- Random Forests (RFs) use **bagging with CARTs** as BLs.
- **Random feature sampling** decorrelates the BLs.
- **Fully expanded trees** further increase variability of trees.

Advantages: Less preprocessing; Can handle missing value; Easy to parallelize; Work well on high-dimensional / noisy data.
Disadvantages: Extrapolation problem; Hard to interpret; Memory-hungry implementation; Computationally demanding for prediction of large ensembles.

Random feature sampling:

- For each node, randomly draw $\texttt{mtry} \leq p$ features to find the best split.
- Previous analysis was simplified. The more we decorrelate by this, the more random the trees become.
- Classification: $\texttt{mtry} = \lfloor \sqrt{p} \rfloor$; Regression: $\texttt{mtry} = \lfloor p/3 \rfloor$

Note:

- Other hyperparameters: `Min. nr. of obs.` in each decision tree node, `depth of each tree`.
- RF usually use fully expanded trees, without aggressive early stopping or pruning, to further **increase variability of each tree**.
- RFs are usually better if ensemble is large.
- RFs can overfit but generally less prone to than individual CARTs.
- Since each tree is trained *individually and without knowledge of previously trained trees*, increasing `ntrees` generally reduces variance **without increasing the chance of overfitting**!

Out-of-Bag (OOB)

Algorithm 2 Out-Of-Bag error estimation to evaluate different RF hyperparameter configurations

1: Input: $\text{OOB}^{[m]} = \{i \in \{1, \dots, n\} | \left(\mathbf{x}^{(i)}, y^{(i)} \right) \notin \mathcal{D}^{[m]}\}$, $\hat{b}^{[m]} \; \forall m \in \{1, \dots, M\}$
2: **for** $i = 1 \to n$ **do**
3: Compute the ensemble OOB prediction for observation i, e.g., for regression:

$$\hat{f}_{\text{OOB}}^{(i)} = \frac{1}{S_{\text{OOB}^{(i)}}} \sum_{m=1}^{M} \mathbb{I}(i \in \text{OOB}^{[m]}) \cdot \hat{f}^{[m]}(\mathbf{x}^{(i)})$$

$S_{\text{OOB}}^{(i)} = \sum_{m=1}^{M} \mathbb{I}(i \in \text{OOB}^{[m]})$ is the nr. of trees where i-th observation is OOB.

4: **end for**
5: Average losses over all observations: $_{\text{OOB}} = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, \hat{f}_{\text{OOB}}^{(i)})$

OOB Size: The probability that an observation is out-of-bag (OOB) is:

$$\mathbb{P}\left(i \in \text{OOB}^{[m]} \right) = \left(1 - \frac{1}{n} \right)^n \xrightarrow{n \to \infty} \frac{1}{e} \approx 0.37$$

$\Rightarrow$ similar to holdout or 3-fold CV (1/3 validation, 2/3 training)

OOB Error Usage: Get first impression of RF performance; Select ensemble size; Evaluate different RF hyperparameter configurations.

Feature Importance

RFs improve accuracy by aggregating multiple decision trees but **lose interpretability** compared to a single tree. **Feature importance** mitigates this problem by asking how much does performance decrease, if feature is removed / rendered useless.

Algorithm 3 Feature importance based on permutations

1: Calculate $_{\text{OOB}}$ using set-based metric ρ
2: **for** features x_j, $j = 1 \to p$ **do**
3: **for** Some statistical repetitions **do**
4: Distort feature-target relation: permute x_j with ψ_j
5: Compute all n OOB-predictions for permuted feature data, obtain all $\hat{f}_{\text{OOB},\psi_j}^{(i)}$
6: Arrange predictions in $\hat{\boldsymbol{F}}_{\text{OOB},\psi_j}$; Compute $_{\text{OOB},j} = \rho(\mathbf{y}, \hat{\boldsymbol{F}}_{\text{OOB},\psi_j})$
7: Estimate importance of j-th feature: $\widehat{\text{FI}}_j = _{\text{OOB},j} - _{\text{OOB}}$
8: **end for**
9: Average obtained $\widehat{\text{FI}}_j$ values over reps
10: **end for**

Algorithm 4 Feature importance based on impurity

1: **for** features x_j, $j = 1 \to p$ **do**
2: **for** all models $\hat{b}^{[m]}$, $m = 1 \to M$ **do**
3: Find all splits in $\hat{b}^{[m]}$ on x_j
4: Extract improvement / risk reduction for these splits
5: Sum them up
6: **end for**
7: Add up improvements over all trees for FI of x_j
8: **end for**

Proximities

Proximities measure of similarity ("closeness" or "nearness") of observations derived from random forests. The proximity $\text{prox}\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \right)$ between two observations $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ is calculated by measuring the number of times that these two observations are placed in the same terminal node of the same tree of random forest, divided by the number of trees in the forest. It forms an intrinsic similarity measure between pairs of observations.

Proximities usage: Imputing missing data, locating outliers, identifying mislabeled data, visualizing the forest.