Machine Learning

Model quality and Ensemble Techniques

S. Nõmm

[1]Department of Software Science, Tallinn University of Technology

27.04.2021

Structure of the sample

- Does the sample representative?
- Does it well balanced?
- Is there any other information to take into account?
- Keep in mind the difference between the data mining (data exploration) and targeted machine learning.

Measures of goodness I

- Let us remind: TP- true positive, TN - true negative, FP - false positive, FN - false negative.
- Keep in mind the difference between the cases of information retrieval and true classification.
- Accuracy, recall, precision, $f1$ - score, ROC-AUC score.
- Sensitivity & specificity
  - Sensitivity is the synonym of recall, also may be referred as True Positive Rate (TPR) or simply hit rate.
  - Specificity is the True Negative Rate (TNR) also referred as selectivity is given by
  $$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$
- Negative predictive value is given by:

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}$$

Measures of goodness II

- False negative rate
$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}$$
- False omission rate
$$\mathrm{FOR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}}$$
- Fall-out or false positive rate
$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}$$
- You are welcome to continue this list ... .

Let us remind the main idea of Cross Validation

- The method to estimate the expected extra-sample error $\mathcal{E} = E[L(Y, \hat{f}(X))]$ (average generalized error) when the method $\hat{f}(X)$ is applied to and independent test sample from the joint distribution of X an Y (L denots loss function here.)
- Cross-validation estimate of prediction error is given by:

$$\mathcal{E}_{CV} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-k(i)}(x_i)).$$

- Usually 5 or 10 fold cross validation is recommended.

Cross Validation within Machine Learning Work-flow

- Up to a present time we have used synthetic sets of a very small power, treating them as the samples.
- For the real life applications when one have the sample only and not entire population this may lead to serious errors.
- One possible way to fix the problem is to perform feature selection within the cross validation loop. (Point to discuss!!!)

Hastie & Tibshirian view on cross validation

- Consider to study in detail section $7.10.2$
- Classification problem with a large number of predictors.
- What would be the strategy to implement ML work flow?

Example p. 245

- $N = 50$ samples, binary case, two equal sized classes.
- Let the power of feature set be $p = 5000$, each feature normally distributed and independent of class labels.
- True error rate for any classifier is 0.5
- Let us suppose that 100 predictors is chosen.
- 1-nearest neighbour classifier was chosen.
- 50 simulations will result in cross validation error of 0.03, whereas true error rate is 0.5
- Leaving samples out after the feature selection does not mimic correctly the application of the classifier to a previously unseen data.

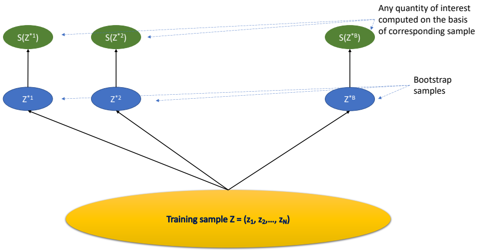
H & T suggest that this is the (correct) way :˜)

- Divide the data set into K cross-validation folds.
- For each fold k perform:
- Use all the folds except the fold k to perform the feature selection and model training.
- Use fold k for model validation.
- Use the results for each k to compute error estimates.

What is the drawback of cross validation?

Bootstrap I

- Let $Z = (z_1, \dots, z_n)$ is the training set.
- Draw randomly data sets with replacement (the samples are independent) from Z. This will result in B *bootstrap* data sets.
- Fit the model for each of B data sets. Examine behaviour over B replacements.
- This approach allows to estimate any aspect of distribution $S(Z)$.


Any quantity of interest computed on the basis of corresponding sample
S(Z*1)
S(Z*2)
S(Z*B)
Bootstrap samples
Z*1
Z*2
Z*B
Training sample Z = (z1, z2,…, zN)

Bootstrap II

- Let $f^{*b}(x_i)$ be the predicted value at x_i from the model fitted to the b^{th} bootstrap dataset.
- Error estimate is given by:

$$\mathcal{E}_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L\Big(y_i, \hat{f}^{*b}(x_i)\Big).$$

- Better bootstrap estimate may be derived by mimicking cross-validation. For each observation we will keep track of predictions from bootstrap samples not containing this observation. This is referred as leave-one-out bootstrap estimate of prediction error and is defined by the following equation.

$$\mathcal{E}_{boot}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{C^{-i}}\sum_{b\in C^{-i}} L\Big(y_i, f^{*b}(x_i)\Big).$$

- Notation here may cause a problem. You are welcome to fix it :) .

Bagging

- Induced from the bootstrap technique (which is used to assess accuracy of estimate).
- Draw B samples with replacements and train the model on each sample.
- The bagging estimate then is defined by:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

Random Forests

The idea is to build large collection of de-correlated trees, and then average them.

- For $b = 1$ to B:
  - Draw a bootstrap sample Z^* of size N from the available training data.
  - Grow tree T_b. Repeat recursively for each terminal node until minimum node size is reached.
    - Select m variables from p.
    - Pick the best variable among m.
    - Split the node.
- Output the ensemble of trees $\{T_b\}_1^B$.
- Prediction:
  - Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.
  - Classification: $\hat{C}_{\text{rf}}^B(x) = \text{mode}\{\hat{C}_b(x)\}_1^B$.

Committee learning

- Some times referred as ensemble learning.
- The idea is to combine a number of weak (accuracy is slightly larger than of random guessing) classifiers into a powerful committee.
- Motivation is to improve estimate by reducing variance and sometimes bias.

Boosting

- The final prediction is given by:

$$G(x) = \operatorname{sign}\Big(\sum_{m=1}^{M} \alpha_m G_m(x)\Big).$$

which is weighted majority vote of classifiers $G_m(x)$. Here α_m are weights describing contribution of each classifier.

- While on the first view result is very similar to the bagging, there are some major differences.
- Two class problem where output variable coded as $Y \in \{-1, 1\}$.
- For the classifier $G(X)$ error rate is given by:

$$\overline{\mathrm{err}} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i)),$$

where N is the power of training data set.

Ada Boost

AdaBoost.M1. by Freund and Shcapire (1997).

- Initialize observation weights $w_i = 1/N$, $i = 1, \dots, N$.
- For $m = 1$ to M:
    - Fit weak classifier G_m that minimizes the weighted sum error for misclassified points.

$$\epsilon_m = \frac{\sum_{i=1}^{N} w_i I(G_m(x_i) \neq y_i)}{\sum_{i=1}^{N} w_i}$$

    - Compute $\alpha_m = \log((1 - \epsilon_m)/\epsilon_m)$.
    - Update weights w_i as

$$w_i = w_i * \exp(\alpha_m * I(y_i \neq G_m(x_i))), \quad i = 1, \dots, N.$$

- Output classifier:

$$G(x) = \operatorname{sign}\Big(\sum_{m=1}^{M} \alpha_m G_m(x)\Big).$$