Computational Intelligence: Methods and Applications

Lecture 36
Meta-learning:
committees, sampling and bootstrap.

Włodzisław Duch
SCE, NTU, Singapore
Google: Duch

One model is not enough

If the problem is difficult, an expert may call for a consultation.

In real-life problems, such as predicting the glucose levels of diabetic patients, a large number of different classification algorithms have been applied, and special issues of journals and books have been written.

What is the optimal way of combining results of many systems?
Create different models (experts), and let them vote.

Committee is **less biased**, results should improve and stabilize, in complex domain modular learning is needed, but ... reasons for decisions taken are difficult to understand (each expert has other reasons).

A collection of models is called **an ensemble**, or **a committee**, hence the ensemble or committee learning problems.

Good models are desired, but those used in committee should specialize, otherwise committee will not give anything new.

Sources of model variability

There are three main reasons why models of the same type differ:

Different data subsets are used for training the same models:

crossvalidation – systematically explore all data
bootstrap – randomly select samples with replacement

Stochastic nature of training:

some learning algorithm contain stochastic elements, ex. all gradient based optimization methods may end up in different local minimum due to different initialization.

Using different parameters to train models:

even if models are completely deterministic, and trained on the same data, different parameters may be used (pruning, features selected for training, discretization parameters etc.

Homo and hetero

A committee may be composed of homo or heterogeneous models.

Homogenous models (most common):
contain the same type of models, for example the same type of a decision tree or the same type of RBF neural mapping, but either trained on different data subsets (as in crossvalidation), or trained with different parameters (different number of nodes in RBF mapping).

Example: use all models created in crossvalidation to form an ensemble.

Heterogeneous models (rarely used):

different type of models are used, for example decision trees, kNN, LDA, kernel methods combined in one committee.

Advantage: explores different biases of classifiers, takes advantage of hyperplanes and local neighborhoods in one system.

Example: GhostMiner committees.

Best book: L. Kuncheva, Combining Pattern Classifiers. Wiley 2004

Voting procedures

Models M_k should return estimation $P(\omega|X;M_k)$

Simple voting: democratic majority decision, most M_k predict ω so predict ω. But ... decisions made by crowds are not always the best.

Linear combination of weighted votes:

$$P(\omega|\mathbf{X};M) = \sum_k W_k P(\omega|\mathbf{X};M_k)$$

Final model M includes partial models M_k and weights W_k; optimize weights to minimize some cost measure on your data.

Select the most confident models - perhaps they know better?
ex. take only $P(\omega|X;M_k) > 0.8$ into account.

Competent voting: in some regions of feature space some models are more competent then others, let only those that work well around X vote.

Use "gating" or a referee meta-model to select who has right to vote.

Bootstrap

Best models (parameters, features), with good generalization, are selected maximizing accuracy on validation partition in crossvalidation.

WEKA has a number of approaches to improve final results.

Analyzing a scene our eyes jump (saccade) all over the scene.

Each second new samples are taken, but unlike in crossvalidation, the same point may be viewed many times.

Bootstrap: form the training set many times, draw data with replacement (i.e. leaving the original data for further drawing).

Bootstrap algorithm:

- from a dataset with n vectors select randomly n samples, without removing the data from the original set;
- use the data that has not been selected for testing;
- repeat it many times and average the results.

0.632 bootstrap

With n vectors, probability that a given vector is not selected is $1-1/n$

Therefore probability that some vector will be used for test is:

$$P(\text{test}) = \left(1-\frac{1}{n}\right)^n \approx \frac{1}{e} = 0.368.. \quad \text{for } n > 40$$

Training data contains the rest, 63.2% of the data; this may lead to overly pessimistic estimation, since < 2/3 of the data is used for training. The 0.632 bootstrap weights the error on the training and on the test

$$E(\text{Data}) = 0.632E(\text{Test}) + 0.368E(\text{Train})$$

Test and train data have thus the same weight in estimation of errors.

This type of bootstrap is recommended by statisticians.

Warning: results for models that overfit the data, giving $E(\text{Train})=0$, are too optimistic, sometimes additional corrections are used.

Bagging (Bootstrap aggregating)

Simplest way of making predictions by voting and averaging.

- Create N_m bootstrap subsets D_i sampling from the training data.
- Train the same model $M(D_i)$ on all data subsets D_i.
- Bagging decision for new vector X:
  - in classification: majority class predicted by N_m models, or probability $P(C_k|X)$ estimated by the proportion of models predicting class C_k.
  - in approximation: average predicted value.

Decreases bias, stabilizes results, especially on noisy data.
Even if individual models are unstable and have high variance, the committee result has lower variance!

Ex: run 10xCV for SSV tree using GM on glass data, accuracy 69±10%, and a committee of 11 trees has 83±2%.

Boosting

Weight performance of the model and encourage development of models that work well on cases that other models fail.

Several variants of Ada Boost (adaptive boosting) exits, creating sequential series of models that specialize in cases that were not handled correctly by previous models. General algorithm is:

- initialize: assign equal weight w_i to each training vector $X^{(i)}$
  - train models $M_k(X)$, k=1 .. m successively on weighted data.
  - for model M_k calculate error E_k using weights w_i
  - increase weights of vectors that were not correctly classified
- Output the decision taking average prediction of all classifiers.

These type of algorithms are quite popular, and may be applied to "weak classifiers" that are slightly better than random guessing, ex. "decision stumps", or one-level trees, sometimes > 100 are used.

AdaBoost M1 version

2 classes ±, n training vectors $X^{(i)}$, m classifiers M_k, true class is $\omega_{\mathbf{X}}$

1. Initialize weights $w_i = 1/n$, for i =1..n,
2. For k=1 .. m train classifier $M_k(X)$ on the weighted data – this is either done by using weights in the error function or replicating samples in the training data.
3. Compute error E_k for each M_k, counting weights w_i as errors
$$E_k = \sum_{i=1}^{n} I\left(M_k\left(\mathbf{X}^{(i)}\right) \neq \omega_{\mathbf{X}}\right) w_i \Big/ \sum_{i=1}^{n} w_i \in [0,1]$$
4. Compute model weights: $\alpha_k = \log\left(\left(1-E_k\right)/E_k\right)$
5. Set new weights for vectors that gave errors: $w_i \leftarrow w_i \exp\left[\alpha_k I\left(M_k\left(\mathbf{X}^{(i)}\right) \neq \omega_{\mathbf{X}}\right)\right]$
6. After all m models are created output the decision $M(\mathbf{X}) = \operatorname{sign}\left[\sum_{k=1}^{m} \alpha_k M_k(\mathbf{X})\right]$

Results of boosting

AdaBoost M1 combined with J.48 WEKA tree algorithm gives in 10CV:

- 10 iterations: test 73.8%
- 20 iterations: test 76.1%
- 30 iterations: test 77.6%
- 50 iterations: test 77.6% - saturated, but significantly better than simple voting committee.

AdaBoost + decision trees have been called "the best off-the-shelf classifiers in the world"! They accept all kinds of attributes and do not require data preprocessing.

Use 1-K scheme to apply it in the K-class case.

Other boosting method exist, for example LogitBoost, with more sophisticated weighting scheme, but they rarely give significant improvements.

Stacking

Stacking: try harder! Do it again!

- Instead of voting use outputs $Y_i=g_i(X)$ of i =1..m classifiers that estimate probabilities, or just give binary values for class labels; train a new meta-classifier on m-dimensional vector Y.
  If not satisfied, repeat the training with more classifiers $Z_i=g_i(Y)$.
- Several new classifiers may be applied to this new data, and meta-meta-classifiers applied; using one LDA model is equivalent to voting with linear weights.

Single–level stacking is equivalent to:

- a committee of classifiers + a threshold function (majority voting);
- a linear classifier (linear weighting);
- or some non-linear weighting functions, like a perceptron $\sigma(W\cdot X)$

Stacking may be a basis for series of useful transformations!

Remarks

Such methods may improve generalization and predictive power of your decision system in many situations, but ...

- algorithms loose their explanatory power: averaging predictions of many decision trees cannot be reduced to any single tree;
- decision process is difficult to comprehend,
- complex decision borders are created;
- overfitting may easily become a problem because more parameters are added.

Constructing better features may lead to more interesting results.

This is rather obvious for signal processing applications, where PCA or ICA features are frequently used.

Meta-learning in WEKA – more examples.

The End

This was only the introduction to some computational intelligence problems, methods and inspirations. The field is vast …

A drop of water in an ocean of knowledge …