Re-sampling and cross-validation

Rafael Pérez-Torres

Cinvestav Tamaulipas

July 17, 2015

Table of contents

Introduction

Why cross validation and resampling?

Cross-validation and resampling methods are validation techniques helpful for:

- Selecting model.
  - Almost all pattern recognition techniques needs one or more parameters.
  - How to select the *optimal* parameters?
- Classifiers performance evaluation.
  - Once the model is selected, how to estimate its performance?
  - The goal is the real error rate, but this is only achievable by performing classification over the whole population.

Introduction

Why cross-validation and resampling?

- Usually the available dataset size is not as large as we would want.
- One approach would be selecting the entire dataset for the classifier training and evaluation, but:
  - This would overfit training data.
  - The error rate estimate might be really optimistic.

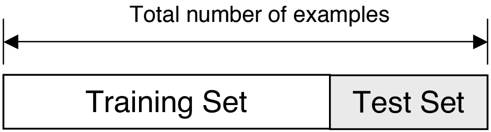
Resampling and cross-validation techniques to the rescue!

Cross-validation techniques

- Basically, they divide available dataset on two global subsets, one for training, the other for testing the classifier.
- Subsets are mutually exclusive, the instance x_i can be only in one of these subsets.

Holdout cross-validation

- It divides the dataset into the training and testing subsets.
- Usually, 2/3 for training, 1/3 for testing.
- A typical usage of holdout is determining a stopping point for the back propagation error in ANN.



Holdout cross-validation

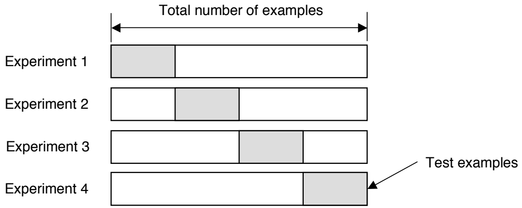
Drawbacks

- For small datasets it is not possible to set aside a portion of the dataset for testing.
- It is a single train-and-test experiment, the estimation of error might be unrealistic.
- There are several alternatives to overcome these issues.

K-fold cross-validation

- Split the dataset in k folds.
- For each k experiments, use $k-1$ folds for training and the remaining one for testing.
- All instances in the dataset are eventually used for both training and testing.
- The estimation of the true error is the average error rate:

$$E = \frac{1}{k}\sum_{i=1}^{k} E_i$$



K-fold cross-validation

How many folds?

For a large number of folds:

- ✓The bias of the true error rate estimator will be small .
- ×The variance of the true error rate estimator will be large.
- ×The computational time will be very large as well (many experiments).

For a small number of folds:

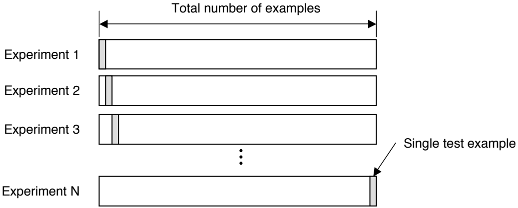
- ✓The number of experiment and computation time are reduced.
- ✓The variance of estmator will be small.
- ×The bias of estimator will be large.

For very sparse datasets, leave-one-out cross validation is preferred in order to train on as many examples as possible. A common choice for K is 10.

Leave-one-out cross-validation

- Special case of the K-cross validation, where $k =$ number of instances.
- For a dataset with N instances, perform N experiments.
- In each experiment use $N - 1$ instances for training and the remaining one for testing.
- The estimation of the true error is the average error rate:

$$E = \frac{1}{n}\sum_{i=1}^{n} E_i$$



Re-sampling techniques

- Split dataset in subsets employing sampling with replacement.
- In this way, an instance can appear in more than one subset and more than once.

Bootstrap family

- The bootstrap techniques family employs a dataset with n, samples it with replacement, creating several *syntethic* datsets of size n, too.
- For obtaining the real error rate, classification is launched with the synthetic datasets created, and further processing the error results.
- In practice, this re-sampling strategy works because if samples are randomly selected, they will keep the same features of the population they came from.

Bootstrap real error rate estimation

- If trainind dataset is $\mathbf{X} = (x_1, \dots, x_N)$ then , it is possible to collect $\mathcal{B}$ bootstrap samples with replacement $\mathbf{Z_1}, \dots, \mathbf{Z_B}$ where each $\mathbf{Z_i}$ contains n instances.
- Then, it is possible to employ the set $\mathbf{Z}$ for estimating the real error rate in classifier prediction as:

$$\mathrm{Err}_{\mathrm{boot}} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f^b(x_i))$$

where $f^b(x_i)$ is the predicted value for x_i according to the model generated for the i bootstrap dataset, and $\mathcal{L}$ is an error measurement function.

L-O-O Bootstrap real error rate estimation

- Previous estimator is not optimal since the samples employed for training the classifier might had contained x_i.
- The leave-one-out bootstrap estimator improves this, trying to imitate the cross-validation approach.
- The LOOB estimator is defined as:

$$\mathrm{Err}_{\mathrm{boot}(1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|C^{-1}|} \sum_{b \in C^{-i}} \mathcal{L}(y_i, f^{b}(x_i))$$

where C^{-i} is the index set of bootstrap sets that do not contain the instance i and $|C^{-i}|$ is the amount of such sets.

Bootstrap .632 real error rate estimation

- $\text{Err}_{\text{boot}(1)}$ fixes overfitting problem but still exposes a bias created by non-distinct observations of the sampling with replacement strategy.
- It has been calculated that the average of distinct instances in wach bootstrap set is approximately $0.632N$.
- Efron and Tibshirani proposed the bootstrap 0.632 error rate estimator, defined as:

$$\text{Err}_{.632} = 0.368\overline{\text{err}} + 0.632\text{Err}_{\text{boot}(1)} \tag{1}$$

where

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, f^b(x_i))$$

- This formulation can be generalized as:

$$\text{Err}_{.632} = w\overline{\text{err}} + (1-w)\text{Err}_{\text{boot}(1)} \tag{2}$$

where $w = 0.632$.

Bootstrap .632+ real error rate estimation

- Under certain scenarios, bootstrap .632 still can describe a large overfitting, when the resubstitution error ($\text{Err}_{\text{boot}(1)}$) is 0.
- The *bootstrap .632+* real error rate estimator puts a larger weight w over $\text{Err}_{\text{boot}(1)}$.
- The w weight is calculated from the *relative overlapping rate* $\hat{R}$ as

$$\hat{R} = \frac{\overline{\text{err}} - \text{Err}_{\text{boot}(1)}}{\hat{\gamma} - \text{Err}_{\text{boot}(1)}}$$

where $\hat{\gamma}$ is the *no-information error rate*.

Bootstrap .632+ real error rate estimation

- $\hat{\gamma}$ (the *no-information error rate*) can be estimated through the permutation of answers y_i and the predictors (patterns) t_j.
- In a dicotomic classification problem, the no-information rate can be calculated as follows:
  - Let $\hat{p}_1$ the proportion of answers $y_i = 1$ and $\hat{q}_1$ the proportion of observed predictions $r_x(t_j) = 1$, then:

$$\hat{\gamma} = \hat{p}_1(1 - \hat{q}_1) + (1 - \hat{p}_1)\hat{q}_1$$

Bootstrap .632+ real error rate estimation

- Finally, the bootstrap .632+ real error rate estimation is defined as:

$$\mathrm{Err}_{.632+} = w\overline{\mathrm{err}} + (1-w)\mathrm{Err}_{\mathrm{boot}(1)}, \;\; w = \frac{.632}{1 - .368\hat{R}} \tag{3}$$

- This estimator is highly realiable when $n > p$ (there are more instances than attributes - dimensions).

Three-way data splits

If model selection and estimation of error need to be computed simultaneously, the data needs to be divided into three disjoint sets:

- **Training set**: A set of examples used for learning and fitting the parameters of the classifier.
- **Validation set**: A set of examples used to tune the parameters of a classifier.
- **Testing set**: A set of instances used **only** to assess the performance of a fully trained classifier.

Three-way data splits

1: Divide the available data into training, validation and test set.
2: Select architecture and training parameters.
3: Train the model using the training set.
4: Evaluate the model using the validation set.
5: Repeat steps 2 through 4 using different architectures and training parameters.
6: Select the best model and train it using data from the training and validation set.
7: Assess this final model using the test set.

If CV or bootstrap are used, steps 3 and 4 have to be repeated for each of the K folds.

Three-way data splits