What is cross-validation and some hints about it

(Konstantin Burlachenko, 22JULY2018, burlachenkok@gmail.com)

"Boundaries between applied math areas are rather artificial" –

Paraphrase Martin Servin, but he said about it in context of 7 specific areas

About this note

It's just a note about magic step from Statistics which distinguish Machine Learning from some of areas of applied math which have been built on first principles approach in various supervised problems.

Usual cross-validation

In usual cross-validation we randomly split our all available data into two sets:

train(70%-80% of data) and **test(30%-20% of data)**.

We have a finite number of Predictors (or Models). There are three typical sources of this models:

1. We have one predictor schema which use *n* predictor variables/features, but we want to perform feature selection, maybe to decrease number of used predictor variables (or number of features). We in some way received all possible subset 2^n , or some small part of it. Now we evaluate which model is better.

2. We have several meta-parameters (or hyper-parameters or tuning parameters) for our predictor schema in ***objective which we minimize*** to obtain model that we will exploit in future. If they are continuous - we discretized them and each specific value of meta-parameters is associated with each model. If they are categorical – we just enumerate them.

We train each predictor on train set via minimizing score function on available data, which is usually: empirical risk (empirical expectation of the loss) with regularization. Also if we can not handle this optimization problem with minimization loss directly then we find “convex approximation” or just apply “convex optimization method” like steepest descend for non-convex optimization.

The last sentence “apply SGD” for non-convex optimization problem is deeply wrong.
Proper way to handle non-convex optimization is very hard and in general can not be done.

Just couple links to mention w.r.t. to it:

1. Paraphrase Aleksandr Gasnikov from this video:
http://www.mathnet.ru/php/presentation.phtml?option_lang=rus&presentid=18246 , 14:00.
“A.V.Gasnikov: Does it exist some constructive theory with subgradient or gradients for non-convex objective? No!”

2. Boris Polyak about non-convex optimization:
https://sites.google.com/site/burlachenkok/articles/math_optimization_questions
“B.T.Polyak: With non-convex optimization there is a strange confusion when various methods work. And what exactly this area have archieved are beautiful names.”

3. Link to Stephen Boyd answer when apply convex optimization methods for non-convex problems:
https://youtu.be/wqy-og_7SLs?t=2953, 49:49
“Question: If you have non-convex problems and we will use solvers that you mentioned. S.P.Boyd: It depends....The most accurate technical statements is the following: something happens"

After we obtain several models in some way we select model with smallest empirical loss on test set. This “test set” is 30%-20% of data is our proxy to the future, which we don’t use during training(model fitting).

Problems with this approach:

Models which had problem with high variance by itself, will have even more high variance during using this methodology, because train data is smaller. If there is small amount of data, then maybe we even can not fit all parameters of the model.

K fold cross-validation

One way to mitigate problems with simple cross-validation is use K-fold cross-validation.

This technic allow to have a data manipulating schema when number of observation is not very big or even limited and we can not make train set and test set as big as we want.

If you can create train or test set as big as you want then you don’t need this technic.

Algorithm:

1. Split available data into K disjoint folds(or groups, or subsets): D_1, D_2, …, D_k which are in the union gives original available data. For example typical case K = 10

2. For $= \overline{1, K}$:

Train each Predictor (or Model) on Train $\left(\bigcup_i D_i\right)/D_j$ and evaluate empirical loss on Test D_j.

3. For each predictor average empirical loss on all test set D_j. It is an estimation of generalization error.

Here cross-validation estimate the performance of the actual predicting.

4.Select a model with lowest generalization error. One possible way of doing things if there are several models with small generalization error - select “simplest” one

K fold cross-validation. Selecting of K and it’s pros. and cons.

As we start with K=1 this is the same as simple cross-validation.

Advantage (as K increasing):

- Give more train data for each predictor model within we perform model selection

- During providing more data we decrease error due to high variance for each trained model

- Perform train/test in such way that all observation from available data used:

> as train data - i.e. data to perform fit via loss minimization

> as test data (i.e. data to estimate error of our predictor as a proxy for future)

Disadvantage (as K increasing):

- As we increasing K we increase computation time. When K will be equal to number of observations from our original provide data this corner case of performing K-fold cross-validation has a special name: **leave-one-out cross-validation.**

- Briefly I think it can be problems to use K-fold cross-validation for time varying phenomen.

Assume phenomen which we try to model is about to make prediction in nearest future. Real model which we approximate is time-dependent. Then K-fold cross-validation with permutation all available data can be something which is deeply wrong. Looks reasonable in this case prepared and weighted train data (from the past and very past) and weight test data (from the nearest past or from future (if we can simulate it) is looks more reasonable thing todo then K-fold cross-validation.

K fold cross-validation. One standart deviation rule.

In fact in algorithm above we estimate empirical error over folds.

In similar way it's possible to upgrade this schema and incorporate knowledge for example of variance.

Step-1: Vary meta-parameter and check empirical risk on K-fold cross validation and estimate standart deviation.

Step-2: Plot a graphic of *estimatedErrorOverFold(varying parameter)*. Attach to each point in this plot vertical bar with length 2x(standart deviation). In fact beside multiplier 2 we can use 4, 8 and etc.

Step-3: First we find as usual - point with smallest point with estimated cross-validation error in plot without incorporating knowledge of vertical bars from *step-2*.

Step-4: We take top part of vertical bar correspond to metaprameter obtained from step-3.

Step-5: Find smallest metaparameter which contains value from step-4 in it's **bar**.

Personally Jerome H.Friedman not sure about that this rule is good, also this rule can be updated to more standart deviations. Proffesor mentioned that maybe in sometimes this rule can be exploited if you want to incorporate prior knowledge – smallest metapatameter is more good.

Also for example one-standart deviation is used in CART implementation.

K fold cross-validation. Leo Out of bag technic from decision tree literature

"Bootstrap sample" technic was invented by Brad Efron but for problems which has no connection with Machine Learning of Decision Trees. Size of T is N observations. We sample randomly observation from this train data, with possible repeats because after each sample we back sample to train data back. Leo Breiman use this technics in Bagging in 1996.

"The simplicity how Leo figure out how to fix weak learner is awesome by it's simplicity" – J.H.Friedman

Step-1: Take booststrap sample size N

Step-2: Take observation which are not in bootstrap and say that it's *test set*

Step-3: Computer Error in observation from sample from step-2

Step-4: Repeat 1-3 several times

This algorithm is called LOB or Leo-out-Of-Bag cross-validation.

Problem with this LOB technic it leads to very big variability in test error.

And Brad Efron suggest use **632 bootstrap** which is equal to

$$LOB \cdot 0.632 + (1 - 0.632) \cdot TrainError$$

It's not completely heuristic number – in fact if make bootstrap sample of size N from the set of size N then it can be proved that probability that some sample will be in bootstrap is 63.2%

K fold cross-validation. Update schema to incorporate notion of variance

In fact in algorithm above we estimate empirical error over folds. In similar way it's possible to upgrade this schema and incorporate knowledge for example of variance.

Formulate vector quantity (avg.error in folds, deviation across folds,…). Scalarized it via regularization technic from math optimization. And find metaparameters correspond to smallest one. Possibly people in ML community tried it. I didn't carry experiments and currently didn't make deep research into existing papers in the field of Machine Learning.