MVPA Permutation Schemes

Permutation Testing in the Land of Cross-Validation

Joset A. Etzel and Todd S. Braver
Cognitive Control & Psychopathology Lab, Psychology Department
Washington University in St. Louis
St. Louis, MO, USA
jetzel@artsci.wustl.edu, tbraver@artsci.wustl.edu

***Abstract*—Permutation tests are widely used for significance testing in classification-based fMRI analyses, but the precise manner of relabeling varies, and is generally non-trivial for MVPA because of the complex data structure. Here, we describe two common means of carrying out permutation tests. In the first, which we call the "dataset-wise" scheme, the examples are relabeled prior to conducting the cross-validation, while in the second, the "fold-wise" scheme, each fold of the cross-validation is relabeled independently. While the dataset-wise scheme maintains more of the true dataset's structure, additional work is needed to determine which method should be preferred in practice, since the two methods often result in different null distributions (and so p-values).**

Keywords- fMRI; classification; significance; permutation; MVPA; cross-validation;

I. INTRODUCTION

Permutation tests are often preferred for significance testing in fMRI multivariate pattern analysis (MVPA, also called multi-voxel pattern analysis or multivariate pattern classification) application studies, since MVPA generally involves dataset characteristics that violate the assumptions needed for parametric tests, or hypotheses unsuitable for traditional methods [1, 2].

Despite this widespread use, the optimum procedures have not been determined. There is considerable ambiguity and numerous experimenter degrees in freedom in how permutation procedures are implemented. For example, how should the relabelings be done, given the complex cross-validation (CV) and dependency structures (e.g. similarity between volumes collected within the same scanner run) present in fMRI data?

In practice, the statistical testing is usually only briefly described, stating that permutation testing was used, perhaps also giving the number of relabelings. These vague methodological descriptions, combined with complex analyses and multiple reasonable implementations, have led to a proliferation of permutation schemes. Here we describe two common permutation strategies, which we refer to as the "dataset-wise" and "fold-wise" schemes. Given the lack of a fixed terminology for describing MVPA permutation testing, we use a concrete example and numerous illustrations to demonstrate the two schemes in the hope that the figures will bridge terminological shortcomings.

II. RUNNING EXAMPLE

We will describe the dataset-wise and fold-wise permutation schemes using a running example of a minimal, yet representative, task-based classification analysis for a single subject. This person completed three runs of fMRI scanning, each of which contained three blocks each of two different tasks. These task blocks were presented with sufficient rest intervals to allow the task-related BOLD signal to return to baseline, making it reasonable to assume that the task labels can be permuted [2, 3]. We further assume that the image preprocessing (motion correction, etc.) was adequate to remove most linear trends and uninteresting signals. Temporal compression [4] was performed, so that each task block is represented in the final dataset as a single labeled vector of voxel values (Fig. 1). There are *n* entries in each vector, corresponding to the voxels falling within an anatomically-defined region of interest (ROI). We assume that *n* is small enough (e.g. 100) that further feature selection is not necessary.

run	block	task	v_1	v_2	...	v_n
1	1	1	0.484	0.006	...	0.200
1	2	2	0.305	0.230	...	0.142
1	3	1	0.159	0.236	...	0.072
1	4	2	0.157	0.344	...	-0.089
1	5	2	0.288	0.155	...	0.144
1	6	1	0.156	0.073	...	-0.124
2	1	2	0.011	0.146	...	0.175
2	2	1	0.076	0.004	...	-0.104
2	3	1	-0.035	-0.016	...	0.047
2	4	2	-0.180	0.109	...	0.053
2	5	1	-0.058	0.057	...	-0.014
2	6	2	0.046	0.110	...	-0.172
3	1	2	0.017	0.187	...	0.412
3	2	1	0.100	0.215	...	0.365
3	3	2	0.059	0.308	...	0.288
3	4	2	0.107	0.138	...	0.036
3	5	1	-0.052	0.147	...	-0.081
3	6	1	-0.103	0.137	...	-0.088

Figure 1. The dataset for the running example is excerpted at left, arranged in the typical manner for MVPA. The boxes at right introduce the dataset representation used in later figures. In these boxes the task labels sent to the classifier are shown in circles. Since this is the true-labeled data, the circled task labels match those shown in the data table.

978-0-7695-5061-9/13 $26.00 © 2013 IEEE
DOI 10.1109/PRNI.2013.44

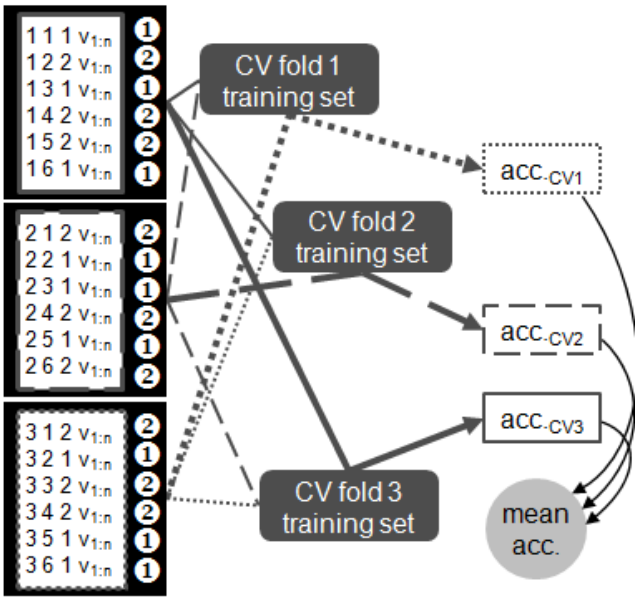

Figure 2. Schematic for determining accuracy. Each rectangle represents the true-labeled data for a run (Fig. 1). Thin lines indicate which runs make up the training set for each fold (i.e. partitioning on the runs), while thick arrows indicate the test set. The final accuracy ("mean acc.") is calculated by averaging the accuracies from each of the three cross-validation folds.

We wish to use a classification algorithm (e.g. linear support vector machines) to distinguish the two tasks, using all *n* voxels listed in the dataset. For simplicity, we will partition the data on the runs (three-fold CV): leave out one run, train on the two remaining runs, and repeat, leaving out each run in turn. The three test set accuracies are then averaged to obtain the overall classification accuracy (Fig. 2), which, if greater than chance, we interpret as indicating that the voxels' BOLD varied with task.

III. Permutation Testing

All permutation tests involve assigning new labels to the observations. For the current running example, as in most task-based MVPA, this entails randomly shuffling the task labels, then retraining the classifier, in order to compare the mean accuracy obtained with the true data labels with the accuracies obtained using the permuted labels [5]. But how *exactly* should the task labels be shuffled?

Figure 3. The twenty possible ways to order the task labels for a single run. These colors are used for the rearrangments in subsequent figures.

It is generally agreed that stratified relabeling should be used: the label permutations need to reflect the variance structure of the data. For the running example, we should thus permute the labels within each individual run (six rows, two task types, ${}_6C_3 = 20$ possible labelings, one of which matches the true task labels for each run), not the dataset as a whole (eighteen rows). This ensures that each run has three examples of each type in all permutations. The permutation test can be thought of as selecting strips at random from a "bag", where each strip lists one of the possible ways to order the task labels within a single run (Fig. 3).

A. Dataset-Wise Permutation Schemes

We first describe the "dataset-wise" scheme, permuting both the training and testing set labels, as shown in Fig. 4. Each iteration of this permutation test starts by selecting three relabelings (as listed in Fig. 3), with replacement, not allowing any run to be given its *true* labeling.

The cross-validation is then performed on each relabeled dataset, in the same manner as on the true-labeled dataset (Fig. 4). Notice the similarity between Figs. 2 and 4: the classifier always sees the same set of labels for a particular run, regardless of whether that run is in the training or testing set on the current CV fold. The dataset-wise scheme can also be used when only the training set is relabeled (instead of the entire dataset, as in Fig. 4.): we keep the true task labeling for each test set, changing only the training set labels, as shown in Fig. 5. Each run now has two labelings within each iteration of the permutation test: one when it is in a training set, and another, the true task labels, when it is in a test set.

B. Fold-Wise Permutation Schemes

The fold-wise permutation scheme also draws new task labels at random from the twenty listed in Fig. 3, but the labels are drawn *independently* on each cross-validation fold. Fig. 6 shows a single iteration of the fold-wise scheme, when

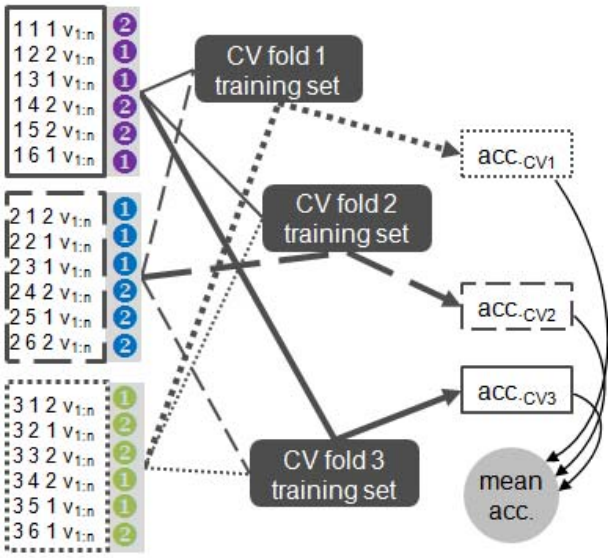

Figure 4. Single iteration of a dataset-wise permutation scheme, when both the training and testing sets are relabeled. The classifier sees the permuted task labels (colored strips), rather than the true task labels.

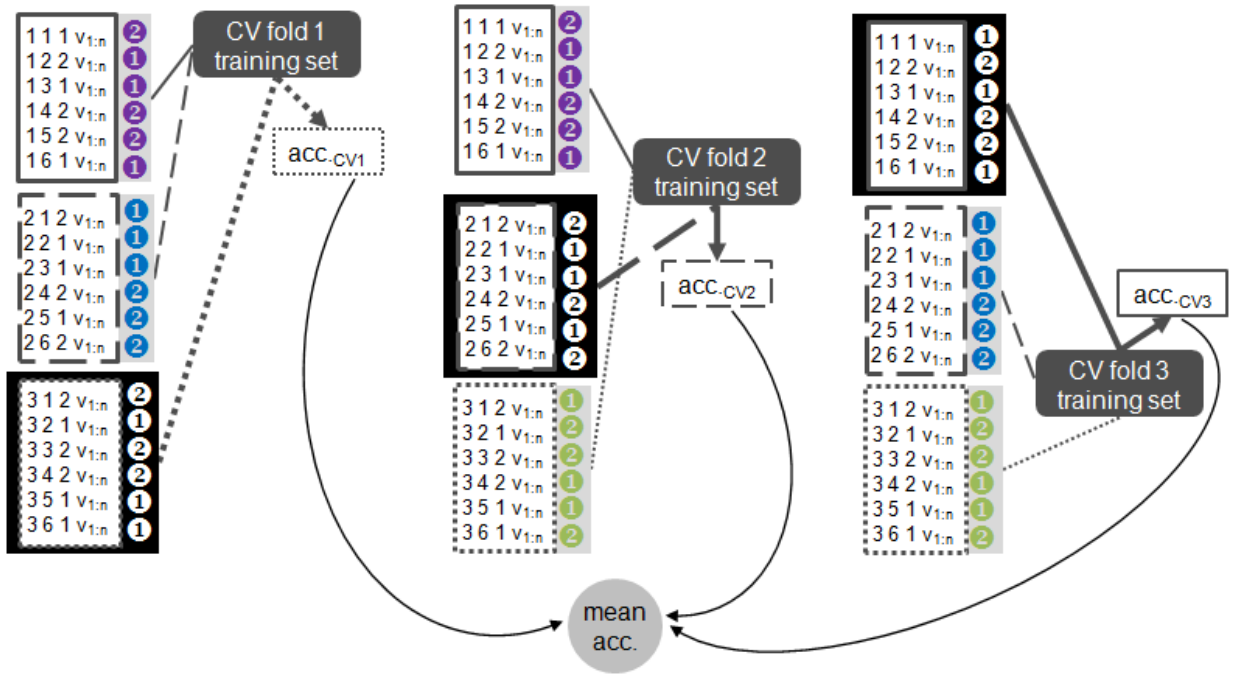

Figure 5. Single iteration of a dataset-wise permutation scheme, when only the training sets are relabeled. The same relabeling is assigned to each run whenever it appears in a training set (as in Fig. 4), but the true labeling is used when it is in the testing set.

only the training set labels are permuted (paralleling dataset-wise Fig. 5). While not shown, a diagram illustrating the fold-wise scheme for permuting both the training and testing sets would be the same as Fig. 6, except that new labels would be assigned to the testing sets (independently on each fold), instead of the true labels shown in Fig. 6.

While Fig. 6 is similar in structure to Fig. 5, under the fold-wise scheme there is no consistency (statistical dependency) in the labeling a single run is given; different labelings can be used on each of the CV folds making up each iteration of the permutation test. Concretely, under the dataset-wise scheme a particular case (say the third row of the Fig. 1 dataset) will always have the same label (1, in Fig. 5) in a single permutation, while under the fold-wise scheme each case can be assigned multiple labels within a single iteration of the permutation test (the third row is assigned 1 on the first fold and 2 on the second in the iteration shown in Fig. 6).

IV. Extension to Arbitrary Designs

The running example is purposely simple, containing the minimum structure necessary to distinguish the dataset-wise and fold-wise permutation schemes. We now briefly describe extending the schemes to arbitrary single-subject designs.

Under the dataset-wise scheme we attempt to keep the treatment of the dataset created for each iteration of the permutation test as similar as possible to the true-labeled dataset: the examples are relabeled first, then the classification is performed *exactly* as on the true data. This naturally extends to more complex designs, but sometimes the number of permutation strata that must be considered is non-trivial. For example, suppose that rather than using the runs for the CV folds in the running example we leave out one example of each class on each fold (i.e. two rows of Fig. 1 make up each test set, the remaining sixteen comprise the training set), for nine-fold CV. There are many ways the two examples making up each test set can be chosen; the test set examples for each fold should therefore be recorded so that the same partitioning can be used for all permutation iterations. Thus, the accuracy obtained on the true-labeled dataset using a particular CV (i.e. which examples are in the training set each CV fold) is compared to the distribution of accuracies obtained by changing the task labels, but in which the CV structure is held constant. Similarly, other dataset changes (e.g. omitting examples to ensure balance in the training data) should be mirrored in the permutation test (e.g. by omitting the same examples).

Under the fold-wise scheme we maximize the randomness of the labels seen by each classifier; each CV fold is relabeled independently. Consider again the case of leave-one-example-out CV. It is still advisable to match which examples are left out each fold in the permutation distribution to those left out in the true-labeled dataset, but the similarity of the nine training sets making up a single iteration is lost. This discards some of the structure of the true data, structure kept with the dataset-wise scheme.

V. Discussion

Both the dataset-wise and fold-wise permutation schemes are currently in use; pyMVPA [6] implements the fold-wise scheme by default, while other authors prefer the dataset-wise scheme. Many manuscripts do not describe the mechanics of the permutation test in sufficient detail for

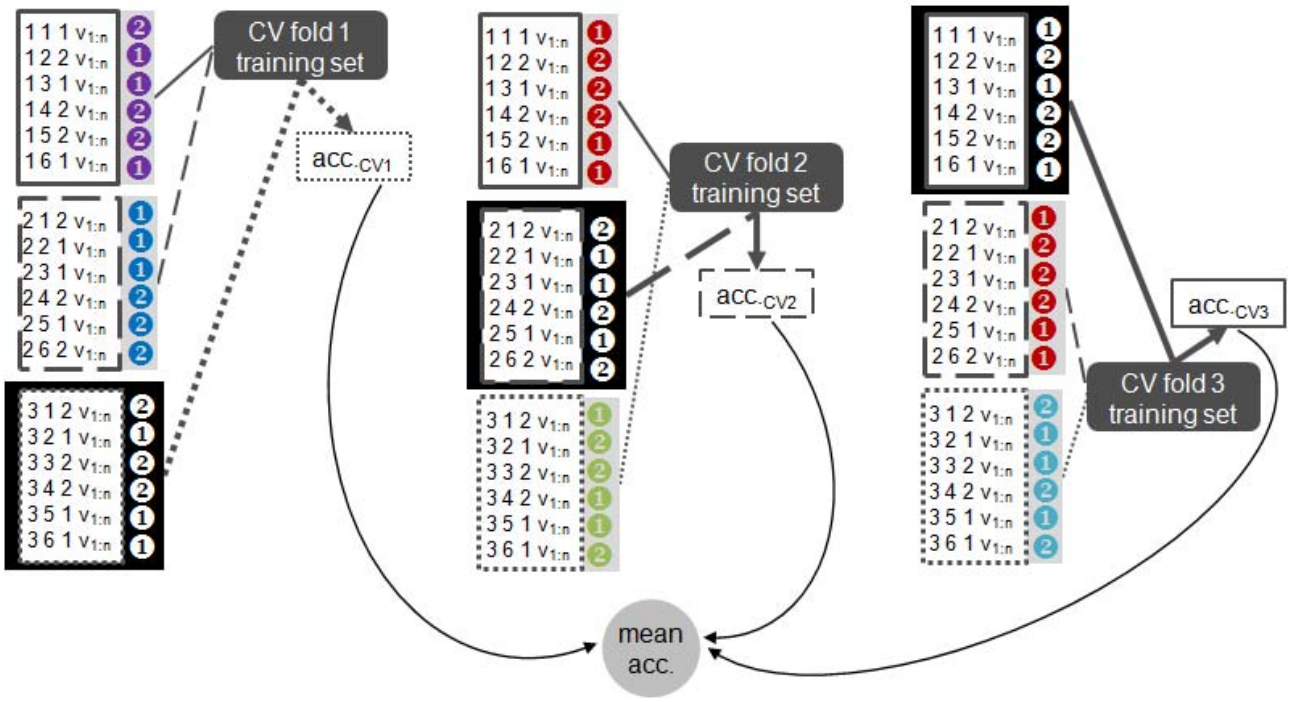

Figure 6. Single iteration of a fold-wise permutation scheme, when only the training sets are relabeled. While similar in structure to Fig. 5 (dataset-wise relabeling the training sets only), under the fold-wise scheme a unique relabeling is chosen for *each* training set run on *each* CV fold: there is no relationship between the labeling on different folds. The same diagram applies if both the testing and training sets are permuted fold-wise, except that new labels are selected for the testing sets as well (i.e. the true-labeled data would not be used).

confident scheme identification. But does it need to be described in more detail? Does the scheme matter?

We have conducted simulations and comparisons exploring the influence of permutation scheme on null distributions (and so p-values). The fold-wise scheme often results in a lower-variance null distribution (and so more significant p-value for any given accuracy), particularly when the number of CV folds is large. For example, in one simulation dataset-wise permutation resulted in a p-value of 0.061 while fold-wise gave a p-value of 0.032, a potentially crucial difference in practice (a detailed description and code to reproduce the simulations is available in https://dl.dropboxusercontent.com/u/13098670/MVPA_perm_schemes.zip).

Theoretical treatment is needed to properly address the question of which permutation scheme provides the best balance of power and leniency, treatment beyond the scope of the current work. Intuitively, the dataset-wise scheme seems more consistent with the principle underlying permutation testing: the relabeled datasets are treated in the same manner as the true data. But this intuition is not universal, and making methodological decisions on the basis of intuition is often unwise.

ACKNOWLEDGMENT

This research was supported by NIH 3RO1MH66078-06A1W1 to Todd S. Braver and 1R01AG031150 to Jeffrey M. Zacks. The authors thank Mohammed S. Al-Rawi, Malin Björnsdotter, and Jeffrey M. Zacks for helpful feedback on earlier versions of this manuscript.

REFERENCES

[1] P. Golland, F. Liang, S. Mukherjee, and D. Panchenko, "Permutation Tests for Classification," in *Lecture Notes in Computer Science*. vol. 3559: Springer Berlin Heidelberg, 2005, pp. 501-515.

[2] T. E. Nichols and A. P. Holmes, "Nonparametric permutation tests for functional neuroimaging: A primer with examples," *Human Brain Mapping,* vol. 15, pp. 1-25, 2001.

[3] F. Pereira and M. Botvinick, "Information mapping with pattern classifiers: A comparative study," *NeuroImage,* vol. 56, pp. 476-496, 2011.

[4] J. Mourao-Miranda, E. Reynaud, F. McGlone, G. Calvert, and M. Brammer, "The impact of temporal compression and space selection on SVM analysis of single-subject and multi-subject fMRI data," *NeuroImage,* vol. 33, pp. 1055-1065, 2006/12 2006.

[5] S. Mukherjee, P. Golland, and D. Panchenko, "Permutation Tests for Classification," in *AI Memo 2003-019*: Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2003.

[6] M. Hanke, Y. Halchenko, P. Sederberg, S. Hanson, J. Haxby, and S. Pollmann, "PyMVPA: a Python Toolbox for Multivariate Pattern Analysis of fMRI Data," *Neuroinformatics,* vol. 7, pp. 37-53, 2009.