Holdouts, Cross-validation and Bootstrapping

The fundamental methodology of data science, machine learning and data analysis

Christos Dimitrakakis

November 24, 2015

1 Introduction

Even with highly complicated tools in our disposal, it is easy for us to make simple mistakes when doing data analysis. The most common one has to do with relying too much on the results on the data we have analysed to obtain conclusions. However, the standard scientific procedure for obtaining conclusions from data involves the following three steps.

1. Obtain data.
2. Analyse data.
3. Use the analysis to formulate a hypothesis.
4. Test the hypothesis on new data.

Can you think of an example in your field where you need to do the following four steps?

Bias in data analysis

There are many reasons why we cannot use the analysis we did on the original data to draw a conclusion directly. This usually because our analysis can be biased in many different ways.

- Data collection (biased sampling). For example, if we want to infer voting intentions for a nation-wide election, and our sample is not representative of the population.
- Feature and attribute selection (cherry picking). This happens when we are drawn to some feature of the data that "stands out", and when we start omitting data that doesn't fit with our preconceptions.

- Model complexity (overfitting). When the algorithm we are using to select the model is run over a very large model space, it is possible that we just 'fit the noise'. Features that may appear important maybe just due to chance, the model may use a very complex relation to predict the given data.
- Objective function (implicit assumptions). Sometimes the objective we use is not really the one we need, when trying to formalise the problem. In fact, the objective that the learning algorithm tries to minimise actually encodes assumptions about the problem is. Sometimes it is computationally difficult to use the right objective and we make do with an easy approximation. Other times there are simply errors in the programming. Finally, it is possible that the data analyst simply misunderstands the objective used by the algorithm.

Remedies

There are a number of simple remedies to make sure our analysis and conclusions are correct.

- Understanding the data collection process. This allows us to uncover any potential biases.
- Unbiased experimental procedures. This enables us to control possible biases.
- Replication: Could somebody else arrive at the same result? In the end, we must check whether it is possible to replicate this result. This can be done by either repeating the procedure on new data to see if we get the same conclusion, or by simply testing our conclusions on new data.

False remedies

There are a number of ways we could try to remedy biases, but they are far from fool-proof.

- p-values and significance tests. These usually entail complex statistical methods, with underlying assumptions that are hard to verify in general. In addition, it is difficult to interpret their result, as well as to separate significance from magnitude of effect.
- Visualisation. This informal method of looking at data, perhaps to remove outliers, or to select important parameters, frequently fails and cannot be part of any unbiased experimental procedure.

The data collection process

A simple example of data collection are election polls.

Example 1.1 (Voting)**.**

- You conduct a fixed-line phone poll of 1000 individuals.
- You ask them the following questions:
  - Their age (<20, 20-40, 41-60, >60)
  - Their education (primary, secondary, tertiary)
  - Their vote intention (A, B, C)

Sample bias

- Most people with a fixed phone are over 40.
- People voting for C are reluctant to say so.

High-dimensional analysis

A number of studies have been showing links between activity in different areas of the brain and thought processes. For example, different parts of the brain seemed to be lighting up when thinking about mathematics than about poetry.

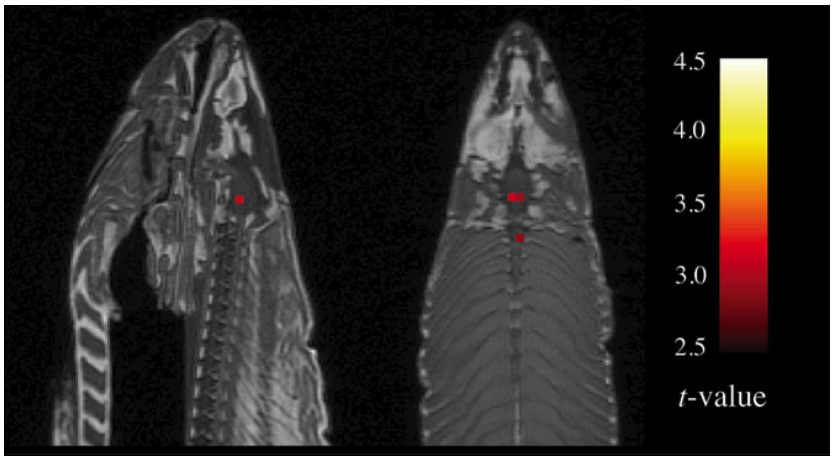
However, most of these studies were badly designed. They fitted a model mapping brain activity to different classes of thought. The features were the activity of different brain areas. However, because there were so many areas (features), very frequently some appeared to be more active than others purely by chance. These teams never again tested their fitted models on new data, and simply published their learned model as though it was a real conclusion.

Later, a team tried a classic experiment, using the exact same methodology of these studies, on a dead salmon. They put the salmon in an MRI machine, and showed it pictures of humans, belonging to two different classes. The fitted model showed "activity" in two different parts of the dead salmon's brain.

Does this mean that the salmon was thinking about images from beyond the grave? Or does it mean that the experimenters should try and verify their conclusion on new data?

Example 1.2 (Dead salmon fMRI)**.**

- Brain fMRI studies are used to discover "active" brain areas for certain tasks.
- Typical statistical tests are used.
- A team used the same techniques on the brain of a dead salmon.



Neural correlates of interspecies perspective taking in the post-mortem Atlantic Salmon: An argument for multiple comparisons correction

Craig M. Bennett[1], Abigail A. Baird[2], Michael B. Miller[1], and George L. Wolford[3]

[1] Psychology Department, University of California Santa Barbara, Santa Barbara, CA; [2] Department of Psychology, Vassar College, Poughkeepsie, NY; [3] Department of Psychological & Brain Sciences, Dartmouth College, Hanover, NH

INTRODUCTION

With the extreme dimensionality of functional neuroimaging data comes extreme risk for false positives. Across the 130,000 voxels in a typical fMRI volume the probability of a false positive is almost certain. Correction for multiple comparisons should be completed with these datasets, but is often ignored by investigators. To illustrate the magnitude of the problem we carried out a real experiment that demonstrates the danger of not correcting for chance properly.

METHODS

Subject. One mature Atlantic Salmon (Salmo salar) participated in the fMRI study. The salmon was approximately 18 inches long, weighed 3.8 lbs, and was not alive at the time of scanning.

Task. The task administered to the salmon involved completing an open-ended mentalizing task. The salmon was shown a series of photographs depicting human individuals in social situations with a specified emotional valence. The salmon was asked to determine what emotion the individual in the photo must have been experiencing.

Design. Stimuli were presented in a block design with each photo presented for 10 seconds followed by 12 seconds of rest. A total of 15 photos were displayed. Total scan time was 5.5 minutes.

Preprocessing. Image processing was completed using SPM2. Preprocessing steps for the functional imaging data included a 6-parameter rigid-body affine realignment of the fMRI timeseries, coregistration of the data to a T_1-weighted anatomical image, and 8 mm full-width at half-maximum (FWHM) Gaussian smoothing.

Analysis. Voxelwise statistics on the salmon data were calculated through an ordinary least-squares estimation of the general linear model (GLM). Predictors of the hemodynamic response were modeled by a boxcar function convolved with a canonical hemodynamic response. A temporal high pass filter of 128 seconds was include to account for low frequency drift. No autocorrelation correction was applied.

Voxel Selection. Two methods were used for the correction of multiple comparisons in the fMRI results. The first method controlled the overall false discovery rate (FDR) and was based on a method defined by Benjamini and Hochberg (1995). The second method controlled the overall familywise error rate (FWER) through the use of Gaussian random field theory. This was done using algorithms originally devised by Friston et al. (1994).

DISCUSSION

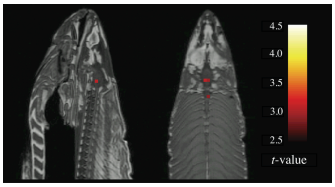
Can we conclude from this data that the salmon is engaging in the perspective-taking task? Certainly not. What we can determine is that random noise in the EPI timeseries may yield spurious results if multiple comparisons are not controlled for. Adaptive methods for controlling the FDR and FWER are excellent options and are widely available in all major fMRI analysis packages. We argue that relying on standard statistical thresholds ($p < 0.001$) and low minimum cluster sizes ($k > 8$) is an ineffective control for multiple comparisons. We further argue that the vast majority of fMRI studies should be utilizing multiple comparisons correction as standard practice in the computation of their statistics.

REFERENCES

Benjamini Y and Hochberg Y (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57:289-300.

Friston KJ, Worsley KJ, Frackowiak RSJ, Mazziotta JC, and Evans AC. (1994). Assessing the significance of focal activations using their spatial extent. *Human Brain Mapping*, 1:214-220.

GLM RESULTS

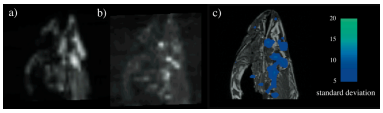


A *t*-contrast was used to test for regions with significant BOLD signal change during the photo condition compared to rest. The parameters for this comparison were $t(131) > 3.15$, p(uncorrected) < 0.001, 3 voxel extent threshold.

Several active voxels were discovered in a cluster located within the salmon's brain cavity (Figure 1, see above). The size of this cluster was 81 mm^3 with a cluster-level significance of $p = 0.001$. Due to the coarse resolution of the echo-planar image acquisition and the relatively small size of the salmon brain further discrimination between brain regions could not be completed. Out of a search volume of 8064 voxels a total of 16 voxels were significant.

Identical *t*-contrasts controlling the false discovery rate (FDR) and familywise error rate (FWER) were completed. These contrasts indicated no active voxels, even at relaxed statistical thresholds ($p = 0.25$).

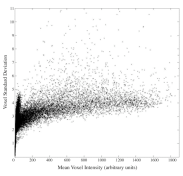
VOXELWISE VARIABILITY



To examine the spatial configuration of false positives we completed a variability analysis of the fMRI timeseries. On a voxel-by-voxel basis we calculated the standard deviation of signal values across all 140 volumes.

We observed clustering of highly variable voxels into groups near areas of high voxel signal intensity. Figure 2a shows the mean EPI image for all 140 image volumes. Figure 2b shows the standard deviation values of each voxel. Figure 2c shows thresholded standard deviation values overlaid onto a high-resolution T_1-weighted image.

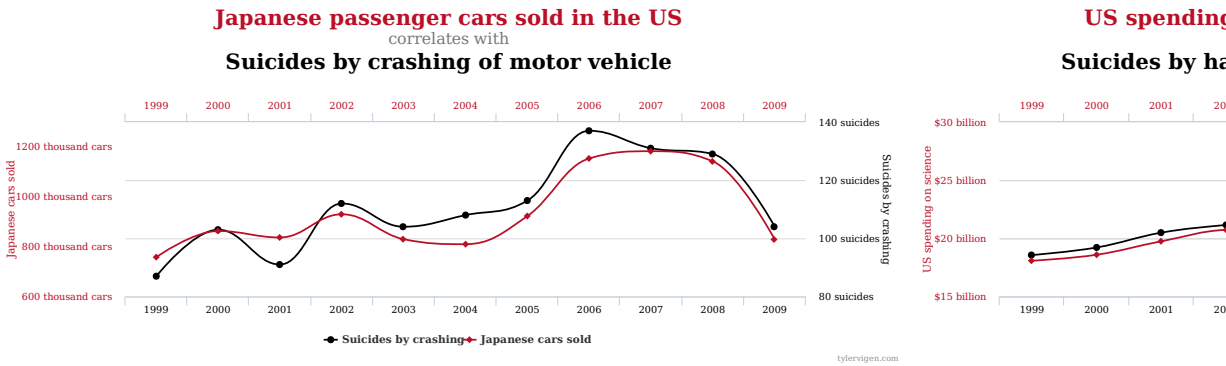
To investigate this effect in greater detail we conducted a Pearson correlation to examine the relationship between the signal in a voxel and its variability. There was a significant positive correlation between the mean voxel value and its variability over time ($r = 0.54$, $p < 0.001$). A scatterplot of mean voxel signal intensity against voxel standard deviation is presented to the right.



2 Correlation and causation

In the dead salmon example, brain activity in certain regions appeared to correlate with the type of images shown to the salmon. However, this was just noise. There were so many brain regions to choose from, that some of them were bound to be highly correlated. The same thing occurs in any sufficiently rich set of data. Two examples are given below.

Spurious correlations



Gene data example

The following data consists of 1000 candidate genes for a specific disease.

- Each attribute is binary, and shows the existence of the gene.
- The class is whether the disease is present.

We will now run an experiment to discover whether there are good predictor genes for the disease.

A small exercise

This simple exercise should give you some idea of how the salmon fMRI analysis was done. Here we'll use trees instead of the method used by the researchers, but the same problem occurs.

Exercise 1. Get the file

`http://goo.gl/FkYL3V`

- Run it as is (with the `source` command).
- Change the parameter `minsplit`.
- What is the parameter that gives the lowest classification error in training?
- What is the parameter that gives the lowest classification error in testing?

- What happens if we use inverse the training and testing set? Do we still get the same tree? If not, why?

You should notice that the error is quite different in either case. It also appears that the trees are very different: different features are used.

Correlation versus causation.

Correlation is not causation.
Two different events may be correlated, but that does not imply causation In general, if some event (e.g. smoking) is related to another (e.g. cancer) in the data, (i.e. cancer is more frequent among smoker), this may imply one of the following.

- Coincidence: the correlation is only there by chance.
- Causation: smoking causes cancer.
- Reverse causation: cancer makes people smoke.
- Common cause: something else causes both cancer and makes people smoke.

Causation can also be indirect. For example, smoking causes local damage in lung cells, which then causes cancer. In temporal events, causation can be cyclic. This is particularly frequent in economics: low demand causes a drop in investment, which leads to lower demand

3 The principle of holdouts / validation

Whenever we use an algorithm or a manual method to select a model, a parameter, a set of features, or any other possible hypothesis, on a set of data, we must always validate our choice on a new set of data. This is the basic principle of data analysis.

There are no exception to this principle.

Methodology of data analysis

So what is the correct way to analyse the brain of a dead salmon? Here is an answer.

Example 3.1 (fMRI analysis of a dead salmon)**.**

- Collect data $\mathcal{D}_T$ to *guess* which brain regions are active.This gives us a set of active brain regions
- See if *the same* brain regions are active in *new data* $\mathcal{D}_H$. If they are not, then the result was just due to noise.

We can now try the same idea using `ID3` for the gene data example. We can see which features are important, by seeing what features are used by the tree to make its decisions.

Example 3.2 (`ID3` for gene data)**.**

- Collect data $\mathcal{D}_T$ to *guess* which features can predict cancer.
- See if *the same* features predict cancer in *new data* $\mathcal{D}_H$.

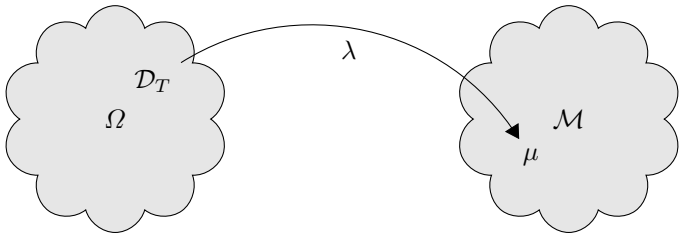
The general problem

Find a model $\mu \in \mathcal{M}$ that

- Explains the given data $\mathcal{D}_T$ (easy).
- Can also explain *new unseen data* $\mathcal{D}_H$ (hard).

The learning algorithm as a function

Learning algorithms can be thought of as functions that take the training data (and sometimes a hyper-parameter) as input, and return a model as the output. This is illustrated in the picture below.



Definition 3.1 (Learning algorithm)**.** An algorithm λ for learning a model or concept $\mu \in \mathcal{M}$ from data $\mathcal{D}_T \in \Omega$ is a mapping $\lambda : \Omega \to \mathcal{M}$.

In this formal definition, a concept or model class $\mathcal{M}$ is the set of all possible concepts we wish to learn. For example, for classification, it could be the set of all possible decision trees that predict a class from some given attributes t. A single $\mu \in \mathcal{M}$ would correspond to a single decision tree.

Example 3.3. In the following example, we specify the learning algorithm to be a decision tree. The concept class is the set of tree models for the given formula, while the input data is `D`. The model returned is the tree.

```
1 tree <- rpart(formula, data = D)
```

Dependency / influence diagram

It helps to visualise the relationships between the learning algorithm, the model and the measured error in terms of an influence diagram.

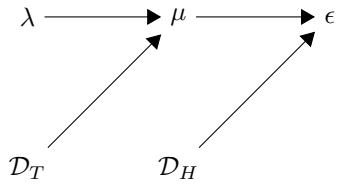


Figure 1: An influence diagram depicting the dependencies between the learning algorithm, data, the resulting model, and the validation error. An arrow from A to B indicates that B depends on A.

It helps to think of the diagram as illustrating the following procedure.

1. We are given an algorithm λ and data $\mathcal{D}_T$.
2. Run the algorithm λ on $\mathcal{D}_T$ to obtain a model μ.
3. Test the model μ on new data $\mathcal{D}_H$ to obtain an error estimate ϵ.

What if we have *multiple* algorithms, or *hyper-parameters*? In that case, we can view the algorithm and parameter selection, as part of the same procedure. We can choose the algorithm minimising the error ϵ. However, that will give us a *biased* estimate of what the error is. To get an unbiased estimate again, we need to test on another dataset.

Using the training set $\mathcal{D}_T$

- An algorithm λ with hyper-parameters θ.
- Run it on $\mathcal{D}_T$ to get output $\mu = \lambda(\mathcal{D}_T, \theta)$.
- Example with decision trees
  - λ is `ID3`.
  - θ is the value of `minsplit`
  - μ is the decision tree.

Using the holdout / validation set $\mathcal{D}_H$

- Test the performance of μ on $\mathcal{D}_H$.
- Use it to select the right algorithm λ and hyper-parameters θ.
- Example with decision trees:
  - For $\theta = 1, 2, \ldots$.
  - Let $\mu_\theta = \lambda(\mathcal{D}_T, \theta)$ be the tree for θ.
  - Measure the error $\epsilon(\mathcal{D}_H, \mu_\theta)$ of that tree on $\mathcal{D}_H$.
  - Choose the model with the lowest error $\mu^* = \arg\min_\theta \epsilon(\mathcal{D}_H, \mu_\theta)$

Features

Decision trees as feature selectors

- Decision trees make decisions using a set of features.
- The "important" features are used first.
- e.g. maybe the genes used to predict cancer actually cause cancer.

What can we infer from a decision tree

- If there is good classification on the training set using 2-3 genes.
- If there is also good classification on the holdout set using 2-3 genes.
- If the error is much higher in the holdout set.

Holdouts and tests

- For a given hyper-parameter θ and algorithm λ, we find the best model $\mu_\theta = \lambda(\mathcal{D}_T; \theta)$, from the data $\mathcal{D}_T$.
- We measure $\epsilon(\mathcal{D}_H, \mu_\theta)$ in a hold-out set $\mathcal{D}_H$ to find the best θ.
- Is this a correct measurement of the error?

The test set

- Given θ, we use $\mathcal{D}_T$ and λ to *choose* μ.
- We use $\mathcal{D}_H$ to *choose* θ.
- We must use a test set $\mathcal{D}^*$ to *measure* $\epsilon(\mathcal{D}^*, \mu_\theta)$.

The principle of testing

If you use data $\mathcal{D}_T$ to select/optimise, you must use *new* data $\mathcal{D}_H$ to measure/test.

Cross-validation

- We must split the labelled data $\mathcal{D}$ into training $\mathcal{D}_T$ and holdout $\mathcal{D}_H$.
- How should we do that?

Definition 3.2 (k-fold cross validation)**.** This procedure creates k different training and hold-out dataset pairs.

1. **Input** Data $\mathcal{D}$, algorithm λ, hyper-parameter θ.
2. Split $\mathcal{D}$ into k *random* equal shares $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
3. **For** $i = 1, \ldots k$
4. Set $\mathcal{D}_H = \mathcal{D}_i$, $\mathcal{D}_T = \bigcup_{j \neq i} \mathcal{D}_j$.
5. Get output $\mu = \lambda(\mathcal{D}_T; \theta)$ and measure error $\epsilon_i = \epsilon(\mu; \mathcal{D}_H)$.
6. **Endfor**
7. Return average error $\epsilon = 1/k \sum_{i=1}^{k} \epsilon_i$.

Leave-one-out

When $k = |Data|$ then the method is called 'leave-one-out' (cross)-validation.

Bootstrapping

Bootstrapping is a separate method from the above. It can be used either with the training set, or the testing set.

- A useful method to estimate the stability of algorithms, or of our estimates.
- Can be used with a separate hold-out set.

The idea is quite simple. Whenever we are using data either as input to the learning algorithm, or as a way to measure the error of a learning algorithm, we *don't* use the data directly. Instead, we use a random sample of the data many times. Each time we use a random sample of the data, we get a different result, but similar to the one we'd get with the original dataset.

Whenever we employ bootstrapping, we take multiple random samples from the dataset, and run our process separately each time. This gives us many different results; the amount by which our results vary tells us how good our estimate is.

When using bootstrapping with the training dataset, each run gives us a new model. If the models are similar, then it's probably safe to draw a conclusion from them. For example, if they all use the same features, then those features are probably important. As another example, if all the models have the same error in the holdout set, then we are probably not overfitting.

Using bootstrapping in the testing or holdout dataset can also be useful, especially when this set is small. Each bootstrap iteration will give us a new, random estimate of our error. If those errors are all close together, then we are pretty sure of what our error is. This idea is particularly useful when we want to compare different methods, especially when it's not possible to construct classic confidence intervals.

The basic bootstrapping algorithm, used on the training data, is shown below.

Definition 3.3 (Bootstrapping)**.**

1. **Input** Training $\mathcal{D}_T$, algorithm λ, hyper-parameter θ, holdout set $\mathcal{D}_H$
2. **For** $i = 1, \ldots k$
3. Let $\mathcal{D}_i$ be a sample of $\mathcal{D}_T$ with replacement.
4. Get output $\mu = \lambda(\mathcal{D}_i; \theta)$ and measure error $\epsilon_i = \epsilon(\mu; \mathcal{D}_H)$.
5. **Endfor**
6. Return average error $\epsilon = 1/k \sum_{i=1}^{k} \epsilon_i$.

Implementing bootstrapping

The basic function of boostrapping, taking a sample with no replacement from a dataset, already exists within the following function in R. This does exactly what we want when we give it a vector of data `x` as an input.

```
sample(x, size, replace = FALSE, prob = NULL)
```

For a vector $\boldsymbol{x}$, return a random vector $\boldsymbol{x}'$ composed of elements of x.

- If replace is false, then no element appears twice, i.e. $\boldsymbol{x}' = (x(1), x(3), x(2))$
- If replace is true, elements can reappear, i.e. $\boldsymbol{x}' = (x(1), x(2), x(1))$

However, our data is usually composed of many features. Each dataset is composed of many rows of data; with each row corresponding to one data point. We want to get a random sample of datapoints.

For a matrix $\boldsymbol{X}$, `sample()` returns a random matrix $\boldsymbol{X}'$ composed of *random columns* of $\boldsymbol{X}$. This clearly is not what we want, as we want to randomly select rows, not columns.

We can sample *random rows* by the following

```
X[sample(nrow(X), replace = TRUE), ]
```

The line `sample(nrow(X), replace = TRUE)` gives us a random sequence of row indices. We can then use that sequence to select the corresponding rows in the data matrix.

Validating a model or a hypothesis

The following exercise illustrates the use of bootstrapping for decision trees.

Exercise 2. For this exercise, use the training gene data from the previous exercise.

- Run a tree on the gene data using a bootstrap sample. Do it at least twice and record the results.
  1. Do the trees use the same features each time?
  2. Do they have the same number of nodes?
  3. Do they have the same error on the holdout set?
- Why should we do holdouts, cross-validation or bootstrapping?
- Is there an advantage between those three methods? When do you think each one of them should be preferred?