5: Overtraining and Cross-validation

Machine Learning and Real-world Data

Simone Teufel

Computer Laboratory
University of Cambridge

Last session: Significance Testing

- You have implemented various system improvements, e.g., concerning the (Laplace) smoothing parameter.
- You have investigated whether a manipulation leads to a statistically significant difference.
- Let us now think about what our NB classifier has learned:
  - has it has learned that “excellent” is an indicator for positive sentiment?
  - or has is learned that certain people are bad actors?

Ability to Generalise

- We want a classifier that performs well on new, never-before seen data.
- That is equivalent to saying we want our classifier to generalise well.
- In detail, we want it to:
  - recognise only those characteristics of the data that are general enough to also apply to some unseen data
  - ignore the characteristics of the training data that are specific to the training data
- Because of this, we never test on our training data, but use separate test data.

Overtraining

- Overtraining is when you think you are making improvements (because the performance you measure goes up) . . .
- . . . but in reality you are making your classifier worse because it generalises less well to real unseen data.
- Until deployed to real unseen data, there is a danger that overtraining will go unnoticed.
- Other names for this phenomenon:
  - Overfitting
  - Type III errors
    - Testing hypotheses suggested by the data
    - Choosing the test falsely to suit the significance of the sample

The “Wayne Rooney” effect

- One way to notice overtraining is by time effects.
  - Time changes public opinion on particular people or effects.
  - Vampire movies go out of fashion, superhero movies come into fashion.
  - People who were hailed as superstars in 2003 might later get bad press in 2010
  - Called the “Wayne Rooney” effect
- You will test how well your system (trained on data from up to 2004) performs on reviews from 2015/6

Overtraining with repeated use of test data

Scenario with separate test data:

- Make some improvement to your classifier.
- Measure performance of system variant on test data.
- Repeat
- Choose the one system variant that performs best on test data, and declare as final best system.

What is wrong with this scenario?

- Repeatedly use of test data means you still overtrained.
- The classifier has now indirectly also picked up accidental properties of the (small) test data.
- You have lost the effect of the data being surprising.

Overtraining, the hidden danger

ML researchers often overlook their own overtraining. There are reasons for this:

- You have to actively work harder (be vigilant) in order to notice that it's happening
- But you may be tempted not to notice it
  - performance "increases" are always tempting
  - (even if you know they might be unjustified).

It's a question of scientific ethics and "truth-finding".

The first principle is that you must not fool yourself, and you are the easiest person to fool. (Richard Feynman)

Am I overtraining?

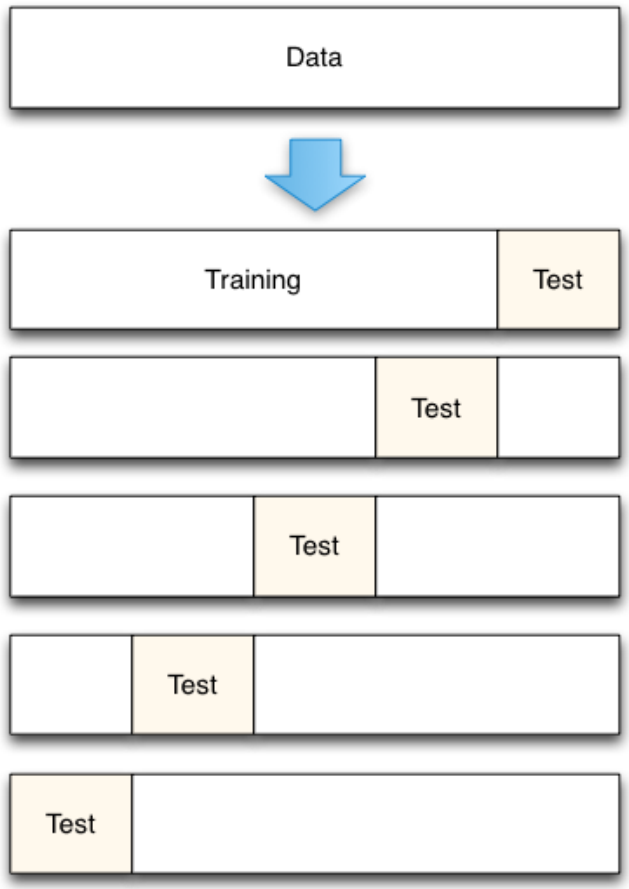
- You can be confident you are not overtraining if you have large amounts of test data, and use new (and large enough) test data each time you make an improvement.
- You can't be sure if you are overtraining if you make incremental improvements to your classifier and repeatedly optimise the system based on its performance on the same small test data.
- One way to detect overtraining is to inspect the most characteristic features for each class (cf. starred tick). You may find features that are unlikely to generalise; sign of overtraining.

Cross-validation: motivation

- We can't afford getting new test data each time.
- We want to use as much training data as possible (because ML systems trained on more data are almost always better).
- We want to use as much test data as possible (because then even smaller effect sizes will show up as significant)
- But whatever we do, we must never test on the training set.
- We can achieve this by using every little bit of training data for testing
- by cleverly iterating the test and training split around

N-Fold Cross-validation

- Split data randomly into N equal-sized folds
- For each fold X, use all other folds for training, test on fold X only
- The final performance is the average of the performances for each fold



N-Fold Cross-Validation and Variance between splits

- If all splits perform equally well, this is a good sign
- We can calculate variance:

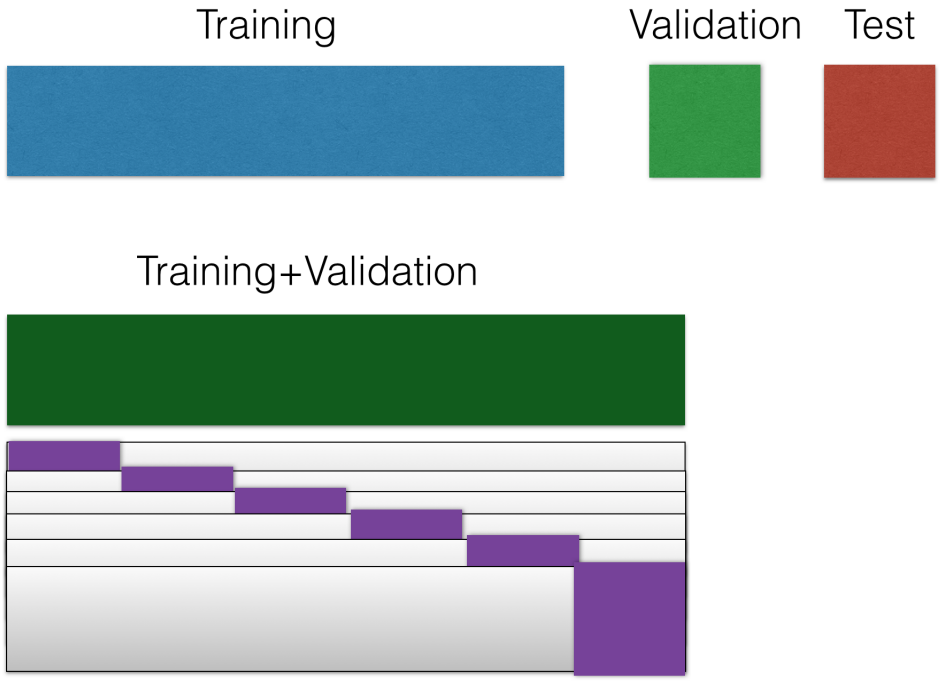
$$var = \frac{1}{n}\sum_{i}^{n}(x_i - \mu)^2$$

  - x_i: the score of the i^{th} fold
  - $\mu : avg_i(x_i)$: the average of the scores

Significance testing under N-Fold Cross-validation

- Compare two systems under N-Fold Cross-validation with each other.
- Consider all of the X test folds together as one overall experiment.
- Not as X different experiments
- Perform one test, counting positives, negatives and null out of the total number of mini-events
- We don't care which fold a mini-event came from, as there won't be any repetition
- You might see significance where there wasn't one before, because you now have gained a lot more test data.

Data splits in our experiment



- Training set (1,600)
- Validation (development) set (200) – old “test” set
- Real test set (200) – new today!
- Use training + validation corpus for cross-validation

Variations on cross-validation

- Stratified cross-validation is a special case of cross-validation where each split is done in such a way that it mirrors the distribution of classes observed in the overall data.
- Jack-knifing (leave one out cross-validation): extreme case of folding where you fold on individual data points
- Dependency-sensitive cross-validation: You fold in such a way that known characteristics of data are isolated in a fold (e.g. one fold per genre or journal).

Cross-validation doesn't solve all our problems

- Cross-validation gives us some safety from overtraining.
- Nevertheless, even with cross-validation we still use data that is in some sense "seen".
- So it is no good for incremental, small improvements reached via feature engineering.
- We also cannot use the cross-validation trick to set global parameters
- because we only want to accept parameters that are independent of *any* training.
- As always, the danger is learning accidental properties that don't generalise.
- Enter the validation corpus

Validation Corpus

- The validation corpus is never used in training or testing.
- We can therefore use this corpus for two things which are useful:
  - We can use it to set any parameters in any algorithm, before we start with training/testing.
  - We can also use this corpus as a stopping criterion for feature engineering
    - We can detect “improvements” that help in crossvalidation over the train corpus, but lead to performance losses on the validation corpus
    - We stop “fiddling” with the features when the result on validation corpus start decreasing (in comparison to the cross-validation results on training set alone).
  - Then, and only then, do we measure on the test corpus (once).
- Validation corpus is sometimes called “development corpus”.

For trouble shooting: Confusion Matrix

System says:

TRUTH:

	POS	NEG	Total
POS	V	W	900
NEG	X	Y	900
Total	V+X	W+Y	1800

Note that cells along the diagonal (red) are correct decisions.

First task today

- Write code that prints out your best system's confusion matrix
- We recommend that you write your confusion matrix printer in a parameterised way so that you can reuse it for different distributions of classes

Second task today

- Implement two different cross-validation schemes:
  - Random
  - Random Stratified
- Observe results. Calculate variance between splits.

Third task today

- Use the precious test data for the first time (on the best system NB you currently have)
- Compare results with the those from testing on the validation set (as you did before today).
- Download the 2015/16 review data; run your system on it
- Test your sentiment lexicon system on the new test data and on 2015/16 data.
- Differences?

Literature

- James, Witten, Hastie and Tibshirani (2013). *An introduction to statistical learning*, Springer Texts in Statistics. Section 5.1.3 p. 181–183 (k-fold Cross-Validation)