5: Overtraining and Cross-validation

Machine Learning and Real-world Data

Simone Teufel and Ann Copestake

Computer Laboratory
University of Cambridge

Lent 2017

Last session: smoothing and significance testing

- You improved your NB system by smoothing.
- You were then able to estimate whether such a manipulation makes a statisticall significant change.
- Let us now think about what our NB classifier has learned.
  - We hope is has learned that “excellent” is an indicator for Positive
  - We hope it hasn’t learned that certain people are bad actors.
- Why?

Let's do something crazy

- Subtask 1 today – Test on training data
- You were told earlier never to do this
- Because the result would not be realistic
- Because optimising on such a result would create a classifier that would do exactly the wrong thing

Beware overtraining!

- The danger here is one of the biggest dangers in ML,
- an undesired effect is called overtraining.
- Overtraining is learning accidental, non-generalising properties from our dataset . . .
- . . . which are however not representative of the overall population.
- Other names for this phenomenon:
  - Overfitting
  - Type III errors
  - “Testing hypotheses suggested by the data” errors

What we really want

- We want a classifier that performs well on new, never-before seen data.
- We don't really care how it does on out test data, but the test data is all we have to assess how well we are doing.
- We can't afford getting new test data each time.
- You will test how well your system (trained on data from up to 2001) performs on reviews from 2015

Danger of overtraining

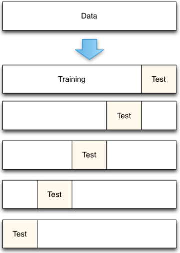
- Maybe vampires rather than superheros were in fashion in 2001 than in 2016.
- New actors, new directors, good directors gone bad...
- Overtraining is when you think you are making improvements (because your performance on the test data goes up) ...
- ...but in reality you are making your classifier worse because it generalises less well to data other than your test data.
- It has picked up accidental properties of the (small) test data.

Crossvalidation: idea

- We can alleviate this problem a bit by at least using all our data as test data, so that we have more of a diagnostic.
- But hang on, can we do this?
- Isn't there an imperative of never testing on training?
- This rule can never be broken.
- But we can still manage to use every little bit of training data for testing sometimes.
- By cleverly iterating the test and training split around

N-Fold Cross-Validation: Splitting

- Split data into N foldss
- For each fold X – use all others for training, test on fold X only
- The final performance is the average of the performances for each fold



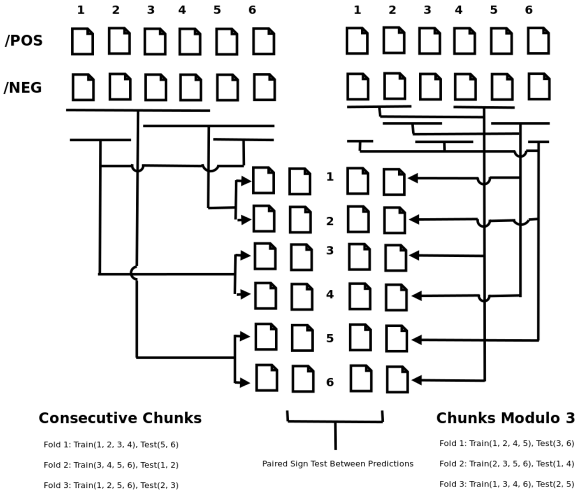
N-Fold Cross-Validation and statistical testing

- Apply sign test across splitting methods – why does it make sense to do so? What does it tell us if we pass this test?
- Stratified cross-validation: each split is arranged in such a way that it mirrors the distribution of classes observed in the overall data.
- There are dvantages and disadvantages of doing it this way

First task today

- Combine training and testset into one pool.
- Implement different cross validation schemes:
  - Random
  - Random Stratified
  - Sequential
- Measure performance and compare to performance on test corpus alone

Statistical testing across N-fold Cross Validation



Cross-validation doesn’t solve all our problems

- OK, we have Cross-validation and 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 crossvalidation trick to set global parameters
- because we only want to accept parameters that are independent.
- As always, the danger is learning accidental properties that don’t generalise.
- Remember the evaluation corpus we set aside in the first session?

Evaluation Corpus

- The evaluation 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 test and train corpus, but lead to performance losses on the evaluation corpus
    - We stop "fiddling" with the features when the result on evaluation corpus start decreasing (in comparison to the crossvalidation results).

Second task today

- Use your evaluation corpus as an alternative to cross-validation.
  - Use it for parameter setting to find a good weight for your symbolic system from Task 1 – maybe you should weight strongly positive words more? But how much more?
  - Figure out a way of triggering a "stop" condition on incremental changes (incremental changes could be punctuation treatment etc).