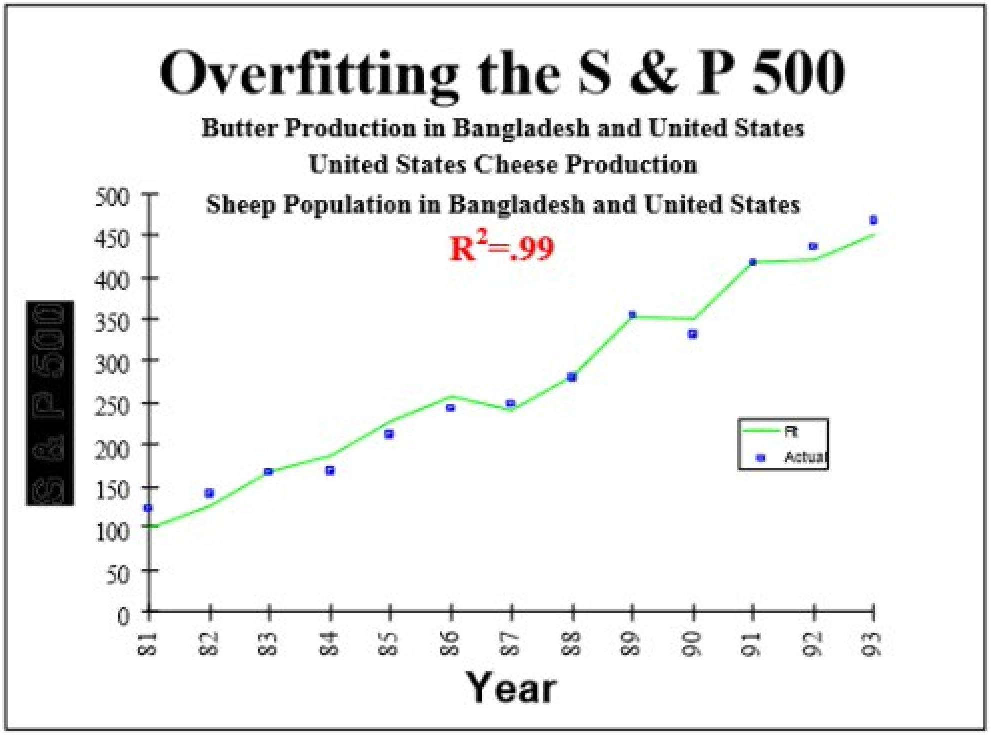
comments

By John Langford (Microsoft, Hunch.net)

(Gregory Piatetsky: I recently came across this classic post from 2005 by John Langford Clever Methods of Overfitting, which addresses one of the most critical issues in Data Science. The problem of overfitting is a major bane of big data, and the issues described below are perhaps even more relevant than before. I have made several of these mistakes myself in the past. John agreed to repost it in KDnuggets, so enjoy and please comment if you find new methods)

“Overfitting” is traditionally defined as training some flexible representation so that it memorizes the data but fails to predict well in the future. For this post, I will define overfitting more generally as over-representing the performance of systems. There are two styles of general overfitting: over-representing performance on particular datasets and (implicitly) over-representing performance of a method on future datasets.



We should all be aware of these methods, avoid them where possible, and take them into account otherwise. I have used “reproblem” and “old datasets”, and may have participated in “overfitting by review”—some of these are very difficult to avoid.

1. Traditional overfitting: Train a complex predictor on too-few examples.

Remedy:

1. Hold out pristine examples for testing.
2. Use a simpler predictor.
3. Get more training examples.
4. Integrate over many predictors.
5. Reject papers which do this.

2. Parameter tweak overfitting: Use a learning algorithm with many parameters. Choose the parameters based on the test set performance.

For example, choosing the features so as to optimize test set performance can achieve this.

Remedy: same as above

3. Brittle measure: Use a measure of performance which is especially brittle to overfitting.

Examples: “entropy”, “mutual information”, and leave-one-out cross-validation are all surprisingly brittle. This is particularly severe when used in conjunction with another approach.

Remedy: Prefer less brittle measures of performance.

4. Bad statistics: Misuse statistics to overstate confidences.

One common example is pretending that cross validation performance is drawn from an i.i.d. gaussian, then using standard confidence intervals. Cross validation errors are not independent. Another standard method is to make known-false assumptions about some system and then derive excessive confidence.

Remedy: Don’t do this. Reject papers which do this.

5. Choice of measure: Choose the best of Accuracy, error rate, (A)ROC, F1, percent improvement on the previous best, percent improvement of error rate, etc.. for your method. For bonus points, use ambiguous graphs.

This is fairly common and tempting.

Remedy: Use canonical performance measures. For example, the performance measure directly motivated by the problem.

6. Incomplete Prediction: Instead of (say) making a multiclass prediction, make a set of binary predictions, then compute the optimal multiclass prediction.

Sometimes it’s tempting to leave a gap filled in by a human when you don’t otherwise succeed.

Remedy: Reject papers which do this.

7. Human-loop overfitting: Use a human as part of a learning algorithm and don’t take into account overfitting by the entire human/computer interaction.

This is subtle and comes in many forms. One example is a human using a clustering algorithm (on training and test examples) to guide learning algorithm choice.

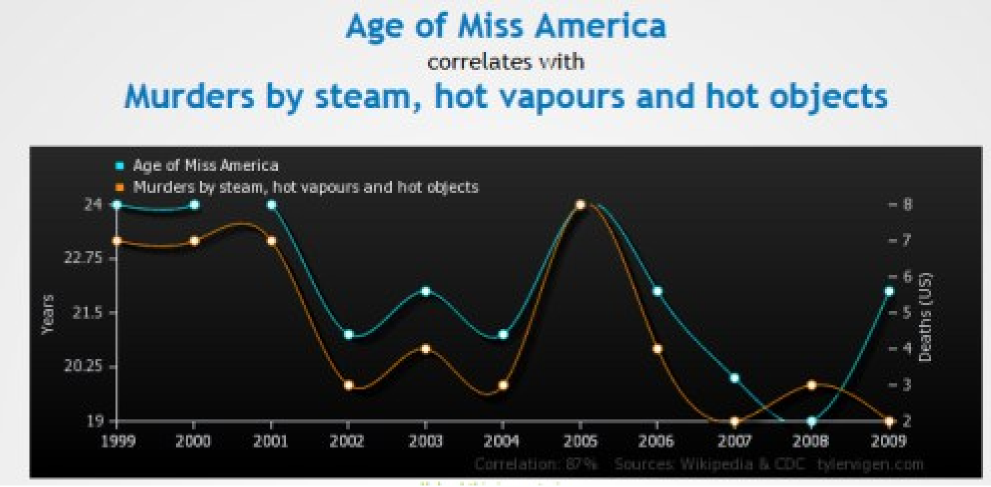
Remedy: Make sure test examples are not available to the human.

8. Data set selection: Chose to report results on some subset of datasets where your algorithm performs well.

The reason why we test on natural datasets is because we believe there is some structure captured by the past problems that helps on future problems. Data set selection subverts this and is very difficult to detect.

Remedy: Use comparisons on standard datasets. Select datasets without using the test set. Good Contest performance can’t be faked this way.

9. Reprobleming: Alter the problem so that your performance improves.



Remedy: For example, take a time series dataset and use cross validation. Or, ignore asymmetric false positive/false negative costs. This can be completely unintentional, for example when someone uses an ill-specified UCI dataset.

Remedy: Discount papers which do this. Make sure problem specifications are clear.

10. Old datasets: Create an algorithm for the purpose of improving performance on old datasets.

After a dataset has been released, algorithms can be made to perform well on the dataset using a process of feedback design, indicating better performance than we might expect in the future. Some conferences have canonical datasets that have been used for a decade…

Remedy: Prefer simplicity in algorithm design. Weight newer datasets higher in consideration. Making test examples not publicly available for datasets slows the feedback design process but does not eliminate it.

11. Overfitting by review: 10 people submit a paper to a conference. The one with the best result is accepted.

This is a systemic problem which is very difficult to detect or eliminate. We want to prefer presentation of good results, but doing so can result in overfitting.

Remedy:

1. Be more pessimistic of confidence statements by papers at high rejection rate conferences.
2. Some people have advocated allowing the publishing of methods with poor performance. (I have doubts this would work.)

I have personally observed all of these methods in action, and there are doubtless others.

Selected comments on John's post:

Negative results:

- Aleks Jakulin: How about an index of negative results in machine learning? There’s a Journal of Negative Results in other domains: Ecology & Evolutionary Biology, Biomedicine, and there is Journal of Articles in Support of the Null Hypothesis. A section on negative results in machine learning conferences? This kind of information is very useful in preventing people from taking pathways that lead nowhere: if one wants to classify an algorithm into good/bad, one certainly benefits from unexpectedly bad examples too, not just unexpectedly good examples.
- *John Langford* I visited the workshop on negative results at NIPS 2002. My impression was that it did not work well. The difficulty with negative results in machine learning is that they are too easy. For example, there are a plethora of ways to say that “learning is impossible (in the worst case)”. On the applied side, it’s still common for learning algorithms to not work on simple-seeming problems. In this situation, positive results (this works) are generally more valuable than negative results (this doesn’t work).

Brittle measures

- What do you mean by “brittle”? Why is mutual information brittle?
- *John Langford* : What I mean by brittle: Suppose you have a box which takes some feature values as input and predicts some probability of label 1 as output. You are not allowed to open this box or determine how it works other than by this process of giving it inputs and observing outputs.

  Let *x* be an input.
  Let *y* be an output.
  Assume *(x,y)* are drawn from a fixed but unknown distribution *D*.
  Let *p(x)* be a prediction.

  For classification error *I(|y – p(x)| < 0.5)* you can prove a theorem of the rough form:
  for all D, with high probability over the draw of m examples independently from D, expected classification error rate of the box with respect to D is bounded by a function of the observations.

  What I mean by “brittle” is that no statement of this sort can be made for any unbounded loss (including log-loss which is integral to mutual information and entropy). You can of course open up the box and analyze its structure or make extra assumptions about *D* to get a similar but inherently more limited analysis.

  The situation with leave-one-out cross validation is not so bad, but it’s still pretty bad. In particular, there exists a very simple learning algorithm/problem pair with the property that the leave-one-out estimate has the variance and deviations of a single coin flip. Yoshua Bengio and Yves Grandvalet in fact proved that there is *no* unbiased estimator of variance. The paper that I pointed to above shows that for K-fold cross validation on *m* examples, all moments of the deviations might only be as good as on a test set of size m/K.

  I’m not sure what a ‘valid summary’ is, but leave-one-out cross validation can not provide results I trust, because I know how to break it.

  I have personally observed people using leave-one-out cross validation with feature selection to quickly achieve a severe overfit.

Related:

- The Cardinal Sin of Data Mining and Data Science: Overfitting
- Big Data Winter ahead – unless we change course, warns Michael Jordan
- The First Law of Data Science: Do Umbrellas Cause Rain?

2 Comments KDnuggets 1 Login

♥ Recommend 2 Share Sort by Best

Join the discussion…

Shishir Pandey • 2 years ago

Could you say more on the meaning of "brittle measures"? In what sense are the mutual information and entropy brittle?

Reply • Share ›

Gregory Piatetsky Mod ➦ Shishir Pandey • 2 years ago

See John technical answer on "brittle" measures above. Intuitively it means that small changes in data can result in large changes in measure value.

Reply • Share ›

✉ Subscribe Ⓓ Add Disqus to your site Add Disqus Add 🔒 Privacy