Using J-K-fold Cross Validation to Reduce Variance When Tuning NLP Models

Henry B. Moss
STOR-i Centre for Doctoral Training, Lancaster University

David S. Leslie
Department of Mathematics and Statistics, Lancaster University

Paul Rayson
School of Computing and Communications, Lancaster University

initial.surname@lancaster.ac.uk

Abstract

K-fold cross validation (CV) is a popular method for estimating the true performance of machine learning models, allowing model selection and parameter tuning. However, the very process of CV requires random partitioning of the data and so our performance estimates are in fact stochastic, with variability that can be substantial for natural language processing tasks. We demonstrate that these unstable estimates cannot be relied upon for effective parameter tuning. The resulting tuned parameters are highly sensitive to how our data is partitioned, meaning that we often select sub-optimal parameter choices and have serious reproducibility issues.

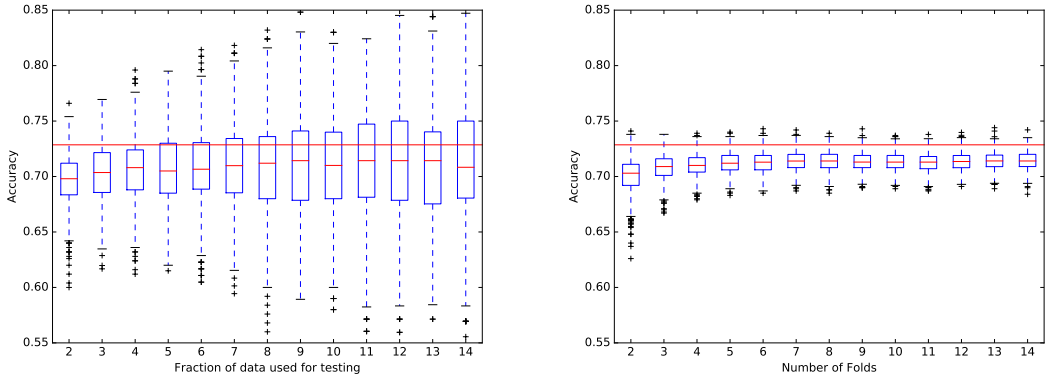
Instead, we propose to use the less variable J-K-fold CV, in which J independent K-fold cross validations are used to assess performance. Our main contributions are extending J-K-fold CV from performance estimation to parameter tuning and investigating how to choose J and K. We argue that variability is more important than bias for effective tuning and so advocate lower choices of K than are typically seen in the NLP literature, instead use the saved computation to increase J. To demonstrate the generality of our recommendations we investigate a wide range of case-studies: sentiment classification (both general and target-specific), part-of-speech tagging and document classification.

1 Motivation

In recent years, the main focus of the machine learning community has been on model performance. We cannot, however, hope to improve our model's predictive strength without being able to confidently identify small (but genuine) improvements in performance. We require an accurate measurement of how well our model will perform once in practical use, known as prediction or generalisation error; the model's ability to generalise from its training set (see Friedman et al (2001) for an in depth discussion). The accurate estimation of model performance is crucial for selecting between models and choosing optimal model parameters (tuning).

Estimating prediction error on the same data used to train a model can lead to severe under-estimation of the prediction error and is unwise. Simple alternatives use a random splitting of data into training and testing sets, or training, validation and testing sets, with the model trained using the training set, tuned on the validation set and performance on the testing set used to report the quality of the fitted model. More sophisticated approaches are based on re-sampling and make more efficient use of the data; including bootstrapping (Efron and Tibshirani, 1994) and K-fold cross validation (CV) (Kohavi, 1995). Since bootstrapping has prohibitive computational cost and is prone to underestimating prediction error, the machine learning community have coalesced around CV as the default method of estimating prediction error. For a snapshot into prediction error techniques currently used in NLP, we look at the proceedings from COLING 2016, focusing on papers that include estimation of model performance. While some submissions use the more sophisticated K-fold CV, the majority use a single data split. Between those that use K-fold CV, 19 use 10-fold, 14 use 5-fold and a further 2 use 3-fold.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2978–2989
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Prediction error estimates based on leaving $\frac{1}{K}$ of the data for testing

(b) Prediction error estimates based on K-fold CV for different choices of K

Figure 1: The substantial variability in performance estimates for IMDB sentiment classification. The horizontal line represents the true performance of the model. Each box-plot summarises 1,000 estimations of prediction error based on a different random shuffle of our $1,000$ reviews.

Each of these estimation methods involves making one or more random partitions of the data. This partitioning means the estimated prediction error is a random quantity, even when conditioned on the data available to the algorithm. This random splitting of the data leads to variation in our prediction estimates, which we define as internal variability. Although this internal variability has been discussed before (Jiang and Wang, 2017; Rodriguez et al., 2010; Bengio and Grandvalet, 2004), it is poorly understood for which datasets and models this is a problem. Note that we are referring to variation between separate estimations by K-fold CV and not variability between the K prediction error estimates that make up a single round of K-fold CV. The estimates are also biased (with their expected value not equal to the truth). Since the model is only trained on a subset it cannot achieve as high performance as if it had access to all the data. Zhang and Yang (2015) argue that evaluating performance, model selection and parameter tuning have distinct requirements in terms of the bias and variance of our estimators. In particular, as long as bias is approximately constant across different models/parameters, it will have little effect on the selected model. However, variability is critical. If our estimates have variance that swamps the real differences in model performance, we cannot tell the difference between genuinely superior parameters and noise. Reducing the internal variance of cross-validation is the main focus of this paper.

To motivate the need for the paper and demonstrate the typical size of this variability, we now introduce a simple NLP task; classifying the sentiment of IMDB movie reviews using a random forest (described in detail in Section 2). Figure 1 shows that for K-fold CV and, to an even greater extent single train-test splits, the prediction error estimates have variability substantially larger than performance differences (less than 1% accuracy improvements between models) identified using the same prediction error estimation methods at COLING 2016. Without analysing the variability of each prediction error estimate it is impossible to say whether we have a genuinely improved model or are just looking at noise. Just because one model outperforms another by a small amount on a particular data-split there is no guarantee that it is genuinely superior and, by changing the partitioning, we may in fact see the opposite. Despite the potentially large variability of the performance estimate, it is common to ignore the stochasticity and just use the results from a single partitioning. The following arguments and experiments are all with respect to the internal variability of K-fold CV. However, as single train-test splits produce less stable prediction error estimates, our arguments are still valid for single data-splits but to an even greater degree.

In the machine learning literature the usual method for reducing the internal variability of prediction error estimates is stratification (Kohavi, 1995), used five times at COLING 2016. This keeps the proportions of classes constant across partitioning and so forces each partition to be more representative of the whole data. Stratification reduces variability due to unbalanced classes. However, as demonstrated in

Figure 1, NLP tasks still suffer from large variability even when the classification classes are perfectly balanced. For natural language, representativity is a much more complicated concept than just matching class proportions. Although having been discussed in the corpus linguistics literature (Biber, 1993), representativity remains a sufficiently abstract task to not have a clear definition that can be used in NLP. We wish to split our data in a way that avoids under-representing the diversity and variability seen in the whole data, however, complicated structural properties like discourse and syntactic rules make it very difficult to measure this variability.

We present, to the best of the authors' knowledge, the first investigation into prediction error variability specifically for NLP and the first in the machine learning literature to investigate the effect of this variability on parameter tuning, questioning both reproducibility and the ability to reliably select the best parameter value. We argue that variability in our prediction error estimates is often significantly larger than the small performance gains searched for by the NLP community, so these estimates are unsuitable for model selection and parameter tuning (Section 2). We instead propose using repeated K-fold CV (Kohavi, 1995) to reduce this variability (Section 3). This is not a new idea, yet has failed to be taken up by the NLP community. Repeated CV was used only twice at COLING 2016 and both times only used for improving the stability of prediction error estimates for a chosen model: 10 repetitions of 10-fold CV by Bhatia (2016) and 2 repetitions of 5-fold CV by Collel (2016). Our main contribution is to extend this method to parameter tuning, alongside investigating guidelines for choosing the number of repetitions and K. Finally, we demonstrate that stable prediction error estimates lead to more effective parameter tuning across a wide range of typical NLP tasks (Section 4).

2 Current Practice: Tuning by K-fold CV

K-fold CV consists of averaging the prediction estimates of K train-test splits, specifically chosen such that each data point is only used in a single test set (for more information see Kohavi (1995)). The data is randomly split into K folds of roughly equal size (known as partitioning the data) before each fold is held out to evaluate a model trained on the remaining $K-1$ folds. Increasing K improves stability by averaging over more models. However, each evaluation is performed on a smaller subset of the available data and so the evaluations themselves become less stable. The combination of these competing variabilities makes up the total internal variance which, as confirmed by Figure 1, is nevertheless smaller than with single train-test splits. If we have enough data that using $1/K^{th}$ of the data still provides stable evaluations then we can see a reduction in internal variability as we increase K. However, this is not a general statement (Bengio and Grandvalet, 2004), as we will demonstrate in Section 4. It is common to choose $K = 5$ or $K = 10$, based on a study by Kohavi (1995) where empirical evidence is presented for these choices producing a reasonable trade-off between bias and variance for some specific statistical examples. We will see that the optimal choice of K is problem-specific so there is no guarantee that Kohavi's studies are comparable to modern NLP tasks. As K-fold CV requires the training of K models, there is a clear incentive to choose the smallest suitable value of K. This idea is developed in Section 3.

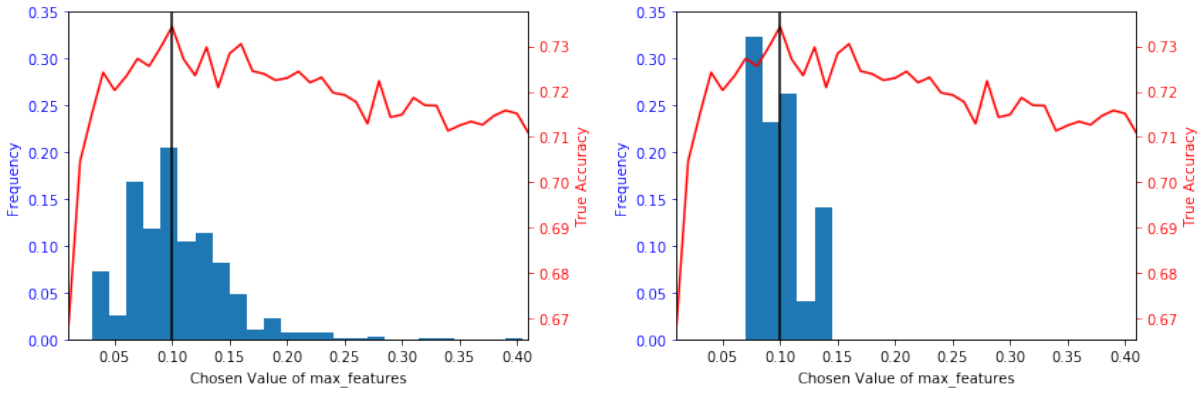
In this paper, we consider an exhaustive method used to tune parameters; grid search. Here we calculate K-fold CV performance estimation across a set of possible parameter values (our grid) and select the value that gives us the best estimated performance. Other popular tuning procedures from the machine learning literature include random search (Bergstra and Bengio, 2012) and Bayesian optimisation (Snoek et al., 2012). Grid search is often not the most efficient approach as it scales poorly with dimensionality. However, its simplicity and interpretability means that it is the standard tuner in NLP and a simple scenario for clearly demonstrating the impact of internal variability on the effectiveness of parameter tuning. Note that all tuning procedures rely on individual prediction error estimates, so our analysis of the problems of grid search is indicative of similar problems with more sophisticated tuning procedures.

To demonstrate the unsuitability of K-fold CV for tuning a typical NLP task, we train a sentiment classifier on a bag-of-word features model using a random forest (Breiman, 2001), a common set-up in the NLP literature (Gokulakrishnan et al., 2012; Da Silva et al., 2014; Fang and Zhan, 2015). We use a large corpus of $25,000$ positive and $25,000$ negative IMDB movie reviews (originally used to train word

embeddings (Maas et al., 2011)) and randomly choose a fixed $1,000$ to act as our training set (on which we need to train and tune our model). The purpose of this contrived task is simply to demonstrate the variability that can result from the random partitioning in K-fold CV. Thus "true performance", consists of the global score on the held-out reviews. We acknowledge that the exact scores depend on which $1,000$ reviews are used for training, so should not be taken as the ground truth. They do, however, allow us to roughly quantify the sub-optimality of our tuned parameters and so measure the effectiveness of different tuning procedures. Even without these scores, the variability of the tuned parameter values raises concerns regarding reproducibility (as decisions cannot be reproduced without exact knowledge of the partitioning).

Using Python's [1] random forest implementation, we consider tuning the maximum proportion of features ($max\_features$) considered at a single time by our model; a parameter whose accurate tuning is crucial to prevent over-fitting. We initially focus on $K = 5$, as Figure 1 shows a lack of significantly improved bias or variance for the increased computational cost of higher K. For each of 1,000 random 5-fold partitions of our training set, we estimate the performance for each parameter value in the set $\{0.01, 0.02, ..., 0.5\}$ and choose the value that gives us the best performance estimate. Aside from maximum depth and number of trees, which we set to 4 and 100 respectively, we keep all other parameters as their SKLEARN defaults. We limit ourselves to the 300 features with the highest tf-idf score (Salton and Buckley, 1988). Note that random forests are stochastic, with a sub-sampling step used to fit the individual tree classifiers. However, as this additional variation cannot be fixed between successive performance estimates (with the sampling being dependent on how our data is partitioned) this should be considered as part of the internal variability.

Figure 2(a) shows that partitioning can have a large effect on the chosen parameter, leading to choices of $max\_features$ across the wide range of 0.03 to 0.4 (almost covering the whole search space). This variability means that our tuning procedure is often failing to choose the parameter that gives the best global performance and so produces sub-optimal models. Depending on the initial partitioning, our tuning procedure can lead to tuned models with global performance varying from 71% to 73.5% accuracy. In Section 4, we will see that the variability is even larger when tuning multiple parameters at once.



(a) Tuning of $max\_features$ by 5-fold CV.

(b) Tuning of $max\_features$ by 10-5-fold CV.

Figure 2: Distribution of our chosen $max\_features$ across $1,000$ random data partitions. We compare the industry standard (a) with our proposed solution (b). The global performance of a model for a given value of $max\_features$ is superimposed in red, with the global optimal choice represented by the vertical line.

3 Proposed Approach: Tuning By Repeated J-K-fold Cross Validation

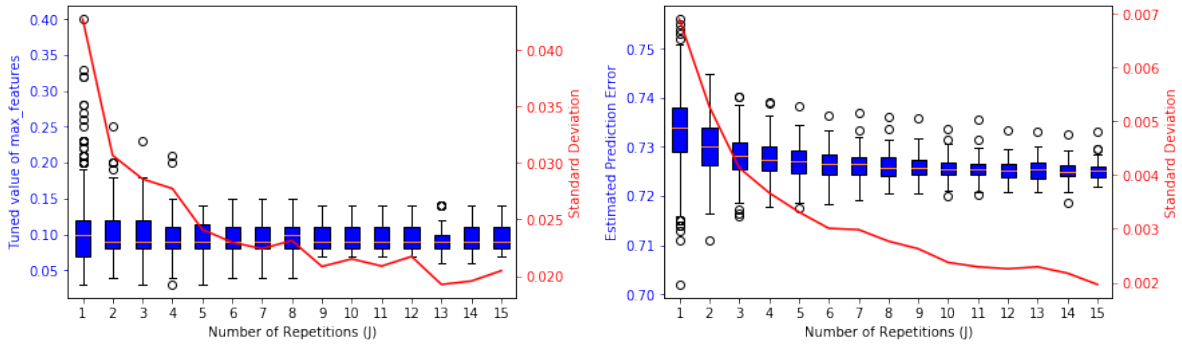
Unfortunately, only considering a single partitioning cannot give us information about the amount of variability present in our performance estimates. To produce Figure 2, we had to look at the tuned model resulting from $1,000$ different partition choices. This observation motivates the use of repeated K-fold

[1] http://scikit-learn.org/stable/index.html

CV, also known as J-K-fold CV; averaging the K-fold CV estimate from J different partition choices. It has been shown empirically that repeated CV reduces internal variability and so stabilises prediction error (Chen et al., 2012; Vanwinckelen and Blockeel, 2015; Jiang and Wang, 2017), especially for smaller datasets (Rodriguez et al., 2010). These authors, however, only investigate the use of J-K-fold CV to reduce the variability of individual performance estimates and do not consider its use for tuning.

We propose the following novel extension to grid search, which extends the improved stability of J-K-fold CV to parameter tuning. For each of J random partitionings, we calculate a K-fold CV estimate for every parameter choice on our grid. The average over these partitions produces performance estimates for each parameter choice and we choose the parameter value that maximises these stable estimates of prediction error. For the rest of the paper we will refer to this procedure as tuning by J-K-fold CV. We believe that using J-K-fold CV will also improve the effectiveness of the more sophisticated parameter tuning procedures, however, this requires further investigation.

Returning to our IMDB example, Figure 2(b) shows that using information from multiple partitioning choices greatly reduces variability in the chosen parameter value, with standard deviation dropping from 0.0427 to 0.0221 when tuning by 10-5-fold CV (Figure 2(b)) instead of vanilla 5-fold CV (Figure 2(a)). This corresponds to the worse performing tuned model over 1,000 random partition choices improving from 71.3% to 72.1% accuracy. We also see a significant reduction in the standard deviation of our performance estimate of the tuned model from 0.700% to 0.233%, greatly improving our ability to discern between closely performing models.



(a) Variability of the tuned $max\_features$ value.

(b) Variability of the performance estimate for the tuned model.

Figure 3: Diminishing reductions in variance as we increase the number of repetitions J for tuning by J-5-fold CV. As before, for each value of J we retune our model over $1,000$ different random partitioning.

	SD of chosen $max\_features$ value	Range of chosen $max\_features$ value	SD of the accuracy estimate of tuned model
1-10-fold CV	0.0396	0.03-0.38	0.550 %
2-5-fold CV	0.0307	0.04-0.25	0.528 %
1-20-fold CV	0.0337	0.03-0.24	0.483 %
2-10-fold CV	0.0301	0.03-0.23	0.400 %
4-5-fold CV	0.0278	0.03-0.21	0.367 %

Table 1: Standard deviation (SD) performance of J-K-fold tuning on the IMDB dataset over $1,000$ different partitions (3 s.f).

We now discuss how to choose the values of J and K in terms of computational cost and effective tuning, a concept we define as selecting near-optimal parameter values irrespective of how the data is partitioned. At a first approximation the cost of J-K-fold CV is proportional to $J * K$ (the number of models trained and evaluated). If, as is the case for sophisticated models, training and evaluation costs are more than linear in the amount of data, then the computational cost grows faster in K, but are still linear in J. So for fixed computational resources we have a trade-off between increasing either J or K.

First, consider increasing K. As previously mentioned there is no clear general relationship between K and the internal variability of K-fold CV. As long as $K \geq 3$ (to guarantee overlapping training sets), we can have comparable variance across K (see Section 4). This means that the only reliable reason to increase K is to reduce bias, however, there is no clear argument for how the bias of each individual performance estimate relates directly to bias in our tuning procedure (the difference between the expected value of our tuned parameter and the true optimal parameter choice). As long as the bias of the estimate for each parameter choice is roughly constant across the parameter grid, then we would expect this bias to have a limited effect on our tuned parameter value (as is assumed for tuning by vanilla K-fold CV). A further research interest is to investigate exactly when this assumption is violated and the effect that this can have on the effectiveness of parameter tuning.

In contrast, increasing J has no effect on bias but does significantly reduce the internal variability. For fixed data, the estimations from each repetition are independent and identically distributed, so prediction error estimates from vanilla K-fold CV have J times as much variability as J-K-fold CV but the same bias. Figure 3 demonstrates that we see similar variance reduction returns in the choice from tuning and also the performance estimate of the tuned model as we increase J. This means we can access the largest drops in variability within the first few repetitions (Kim, 2009).

We can therefore consider our choices of K and J in isolation. K is increased to reduce bias whereas J reduces the internal variability. We commented earlier that effective parameter tuning is more sensitive to variance than bias. This is confirmed by our IMDB task, where we see that although the tuned value from 5-fold CV has a mean close to the best performing parameter choice (calculated for our held-out $49,000$ reviews), the tuned value has significant variability. Sub-optimal tuning is a consequence of this large variance, which overshadows any potential problems from the bias. Therefore we first need to reduce internal variability before our tuning can benefit from reduced bias.

Table 1 summarises the tuning performance of five different J and K choices. We see that between 1-10-fold CV and 2-5-fold CV, and between 2-10-fold, 4-5-fold and 1-20-fold (choices of equivalent computational cost), we have the least variability when reducing K and using the saved computation for increasing J. Note that the naive approach of using all available computation to increase K (1-20-fold CV) produces a wider range of parameter choices than 2-10-fold and 4-5-fold CV. This is despite any bias reductions from choosing a higher K and so we do not recommend 1-$J * K$-fold CV. This analysis questions the rule of thumb selection of $K = 10$ common in NLP, as 2-5 CV seems to produce superior tuning at the same cost, a hypothesis we now test across a range of NLP tasks.

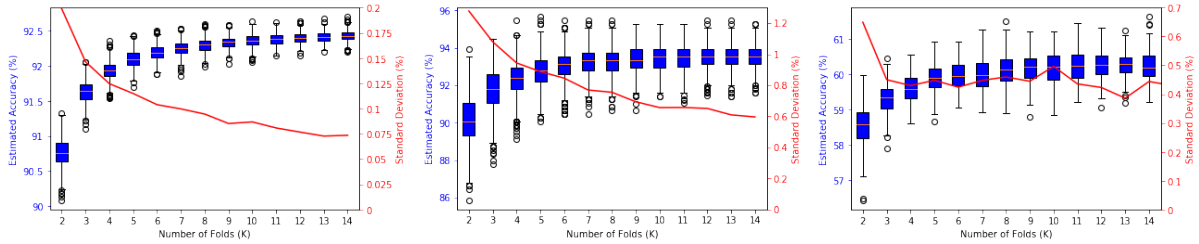
4 Case Studies

We now widen our investigation into the suitability of K-fold CV by analysing three further NLP tasks, chosen to cover a wide range of typical models and datasets. Note that we are not trying to improve the modelling of these standard models, just showing that ineffective tuning due internal variability is a general issue across NLP. We look at a simple part-of-speech tagger using logistic regression, document classification with support vector machines, and target-dependent sentiment classification with an LSTM. Each task is treated the same as our IMDB example; comparing the performances of tuning by 1-10-fold CV (the industry standard) against the computationally equivalent 2-5-fold CV. We also investigate the situation where we have two and four times the computational budget by comparing 2-10-fold with 4-5-fold and 4-10-fold with 8-5-fold. Our first task shows a case where tuning by vanilla K-fold is still reasonably effective. For task two we tune multiple parameters at once, seeing substantial variation in our tuning and so a real need for J-K-fold CV. Our final example shows that our criticisms still hold for more sophisticated models, where it can be necessary to have even more than 8 repetitions for reliable tuning. All code is available [2].

Before investigating tuning the different models, we use these examples to provide empirical evidence for our recommendation of using lower K than the common choice of ten. We argued that increasing K has no clear effect on the internal variance of individual performance estimates from K-fold CV and diminishing bias reduction returns. These are presented in Figure 4. Note that although our POS

[2] https://github.com/henrymoss/COLING2018

tagger and topic classifier do in fact become less variable for larger K, this is not a general rule, with our LSTM providing a counter-example (with comparable variance for choices of K above three). Even when we do see reductions in variability when increasing K beyond five, the following experiments show that reductions in internal variability from increasing J dominate those gained from higher K, failing to provide justification for choosing K as large as ten. Note also that the accuracy estimates for all three tasks show significant variation and so predictions by vanilla K-fold CV are unsuitable for the comparison of even relatively closely performing models.



(a) K-fold CV for logistic regression part-of-speech tagger.

(b) K-fold CV for support vector machine topic classifier

(c) K-fold CV for LSTM target-dependent sentiment classification.

Figure 4: Prediction error estimates based on K-fold CV. The variation is calculated across 1000 random partitionings for (a) and (b)) and 100 for (c).

4.1 Part-of-speech tagger

For our first task we are not directly recreating a method from a particular paper, just demonstrating that J-K-fold CV can improve the tuning of even the most common NLP models. To find such a task we consulted Jurafsky and Martin (2000), choosing logistic regression and part-of-speech tagging. We train and evaluate our model on a fixed $10,000$ word subset of the Brown Corpus (as available in Python's NLTK package (Bird, 2006)). As with our IMDB example, we use the remaining $90,000$ words as an independent global test set to check the validity of our parameter tuning. We classify with respect to the Penn Treebank tagging guidelines (Santorini, 1990) based on simple intuitive features including information about prefixes, suffixes, capital letters, position in sentence and adjacent words. We tune the amount of l2 regularisation (described by C in SKLEARN) on the grid $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\}$.

Table 2 shows that larger choices of J with smaller K provide the most effective tuning at each computational budget. Our most unstable tuning is by 1-10-fold CV. However when checked on the held-out population, the tuned model with the worst true performance only loses 0.05% accuracy from the parameter choice with best global performance. So although increasing J does reduce variability, it is not always necessary for effective tuning, as vanilla 10-fold CV produced consistently near-optimal models. The size of variability, however, is still a concern for reproducibility, as different random partitions can lead to choosing C as 10, 50 or 100.

4.2 Document Classification

We now consider a task where tuning by vanilla K-fold CV is inadvisable; topic classification on the Reuters-21578 dataset (Lewis, 2006) via support vector machines (SVM) (Cortes and Vapnik, 1995). This exact task is well-studied in the NLP literature (Joachims, 1998; Tong and Koller, 2001; Leopold and Kindermann, 2002) however no details are provided regarding the tuning of model parameters. Note that SVM are still commonly used for document classification. This dataset and a specific train-test split (known as the ApteMod version) are available in NLTK. For ease of explanation, we focus on just the corn and wheat categories, producing a binary classification task with 334 instances. We tune two parameters; the flexibility of the decision boundary and the RBF kernel coefficient (denoted in SKLEARN as C and $gamma$ respectively). We search for C in $\{1, 5, 10, 50, 100, 500, 1000, 5000, 10000\}$ and $gamma$ in $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$.

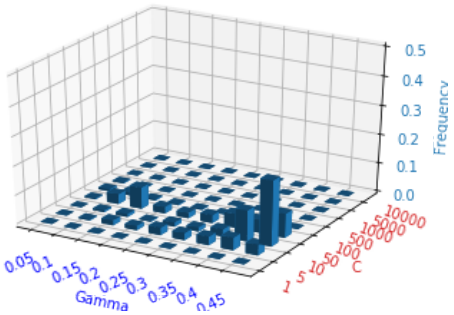
	SD of chosen C value	SD of the accuracy estimate of the tuned model
1-10-fold CV	34.6	0.0915 %
2-5-fold CV	31.0	0.0874 %
2-10-fold CV	29.0	0.0653 %
4-5-fold CV	27.3	0.0653 %
4-10-fold CV	23.2	0.0472 %
8-5-fold CV	24.4	0.0439 %

Table 2: Performance of different J-K-fold tuning procedures for a logistic regression part-of-speech tagger over $1,000$ different partitions (3 s.f).

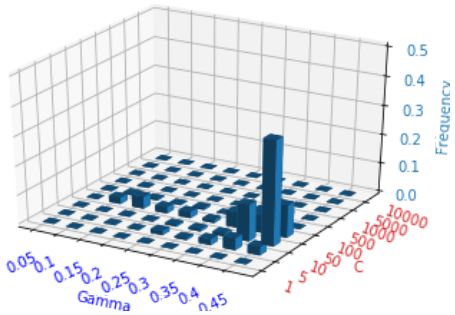
SD of chosen C value	SD of chosen $gamma$ value	SD of the accuracy estimate of the tuned model
28.7	0.121	0.807 %
30.7	0.0987	0.615 %
21.3	0.105	0.580 %
20.4	0.0766	0.445 %
20.0	0.0708	0.407 %
17.5	0.0563	0.330 %

Table 3: Performance of different J-K-fold tuning procedures for a support vector machine topic-classifier (3 s.f).

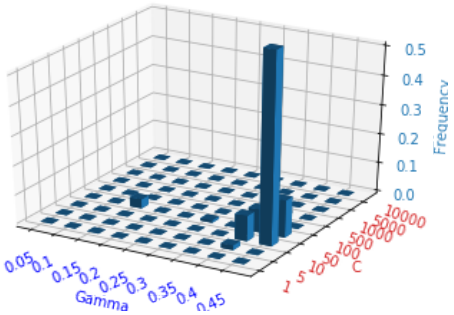
As summarised in Table 3, this task suffers from much larger variability than the part-of-speech tagger. Once again, at each computational cost, the most effective tuning (in terms of $gamma$ and variability of the performance estimate of the tuned model) is from the lower choice of K and higher J. Note that for C variability, there is an slight increase when moving from 1-10-fold to 2-5-fold CV, however, this corresponds to a large drop in the other variabilities and so 2-5-fold CV still provides more effective tuning overall. Also note that we see a larger reduction in $gamma$ variability than C variability for larger J choices. This is explained in Figure 5, where we see substantially more variation in the tuned value of $gamma$ than in C when tuning with no repetitions (Figure 5(a)). Note that effective parameter tuning corresponds to a single predominately large bin, i.e. selecting a single parameter choice irrespective of the partitioning. This is achieved by increasing J (Figures 5(b) and 5(c)). Also note that the accuracy estimate for the model tuned by 1-10-fold CV is not stable enough to allow comparison with other models of close performance. To be able to reliably distinguish between our tuned model and alternatives with performance differing by only a couple of percentage points, we require higher choices of J.



(a) Tuning SVM by vanilla 5-fold CV



(b) Tuning SVM by 2-5-fold CV



(c) Tuning SVM by 8-5-fold CV

Figure 5: Distribution of the tuned parameter values across $1,000$ random partitionings.

	SD of chosen widths	SD of the accuracy estimate of the tuned model
1-10-fold CV	19.9	0.293 %
2-5-fold CV	19.8	0.240 %
2-10-fold CV	19.3	0.213 %
4-5-fold CV	18.4	0.205 %
4-10-fold CV	17.3	0.169 %
8-5-fold CV	16.0	0.166 %

Table 4: Performance of J-K-fold tuning when choosing the width of the LSTM (3 s.f).

SD of chosen bias regularisation	SD of chosen input regularisation	SD of the accuracy estimate of the tuned model
0.0404	0.0160	0.365 %
0.0285	0.00705	0.284 %
0.0358	0.0166	0.254 %
0.0195	0.00995	0.206 %
0.0280	0.0185	0.196 %
0.00815	0.000500	0.148 %

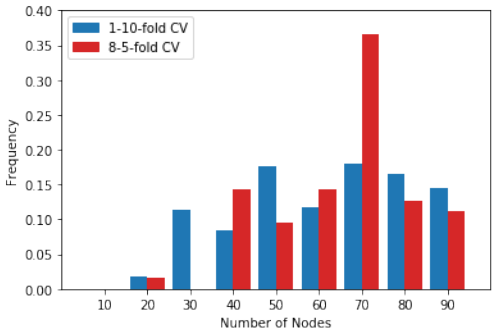
Table 5: Performance of J-K-fold tuning on the IMDB dataset over $1,000$ different partitions (3 s.f).

4.3 Target-Dependent Sentiment Classification

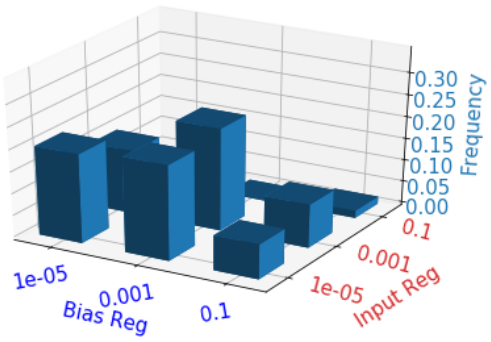
For our final task we consider a much more sophisticated model using LSTMs (with Python's KERAS [3]) to perform target-dependent sentiment classification (Tang et al., 2016) using twitter-specific sentiment word vectors (Tang et al., 2014) on the benchmark twitter dataset collected by Dong et al (2014). Our implementation relies heavily upon the reproduction study of Moore et al (2018). Each of $6,248$ sentences are annotated with a target element and the task is to predict the sentiment regarding that element (positive, negative or neutral). Unfortunately little information is provided about model architecture or parameter tuning. However, we can still investigate the effectiveness of tuning by J-K-fold CV. Throughout these experiments we fix the maximum number of epochs as 100 and stop training when we see no improvements in validation set performance for five successive epochs. For each of our $J * K$ models, the validation set is a random 20% of that model's training data, and so can be thought of just another part of the random partitioning. We use an ADAM optimiser with the default learning parameters and a batch size of 32.

We perform two experiments: tuning the number of nodes in the LSTM layer (known as width) across the grid $\{10, 20, 30, .., 90\}$, and separately tuning the amount of l2 regularisation on the inputs and biases (for a fixed width of 50) each across $\{0.00001, 0.001, 0.1\}$]. Tables 4 and 5 provide further support for our claim that the most effective tuning for a specified computational budget comes from lower K and higher J. Note that, as shown in Figure 6(a), vanilla 1-10-fold CV is not at all suitable for tuning the width of our LSTM, as it produces almost uniform values between 30 and 90. It also fails to consistently select a single choice for the regularisation scheme (Figure 6(b)). In contrast, tuning with 8-5-fold produces much more consistent choices, the majority of the time choosing 70 for the optimal width (Figure 6(a)) and 0.001 for both the input and bias regularisation (Figure 6(c)). Although variability in our chosen LSTM parameters does reduce as we increase J, it remains significantly higher than the gaps in our parameter grid. When coupled with the relative stability of the accuracy estimates of this tuned model, this suggests that the performance differences between the most frequent choices by 8-5-fold CV are small. However, to consistently tune the model to this specificity we require larger choices of J than 8.

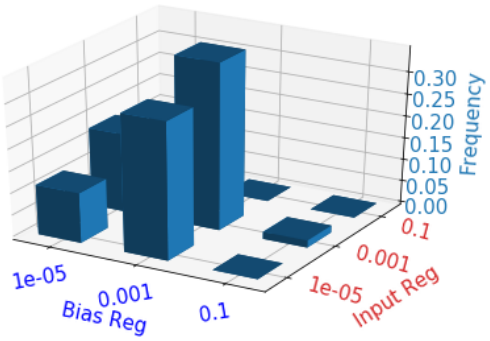
[3] `https://keras.io/`



(a) Tuning width by 1-10-fold CV and 8-5-fold CV



(b) Tuning regularisation by 10-fold CV



(c) Tuning regularisation by 8-5-fold CV

Figure 6: Distribution of tuned LSTM values over random partitionings.

5 Discussion and Conclusions

The aim of this paper has been to demonstrate the significant variability of current performance estimation techniques when applied to NLP tasks. We argue that the size of this variability is poorly understood by practitioners, which has serious implications for both model selection and parameter tuning. We show that the variability in estimates can often be larger than the performance improvements sought by the NLP community; questioning conclusions regarding the comparison of techniques. For parameter tuning, the variability can lead to unstable, irreproducible and significantly sub-optimal parameter choices.

We advocate the use of J-K-fold CV, which we extend to parameter tuning. By using information from multiple estimations we stabilise our tuning procedure. To counteract the computational cost of increasing J, we suggest lower choices of K, as effective tuning is more reliant on variability than bias.

Although we have shown the effectiveness of some specific choices of J and K on our NLP examples, there is still work to be done regarding choosing their optimal configuration, which is problem dependent. Unlike K, J can be chosen adaptively, allowing practitioners to respond to the amount of observed variability and efficiently manage computational resources. We would also like to analyse the current common practice of early stopping (as in our LSTM example), which requires evaluations on another held-out set to prevent over-fitting. This is likely to suffer from the concerns outlined in this report and it is poorly understood how best to incorporate early stopping into a wider parameter tuning framework.

6 Acknowledgements

The authors are grateful to reviewers, whose comments and advice have greatly improved this paper. The research was supported by EPSRC and the STOR-i Centre for Doctoral Training.

References

Yoshua Bengio and Yves Grandvalet. 2004. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Shraey Bhatia, Jey Han Lau, and Timothy Baldwin. 2016. Automatic labelling of topics with neural embeddings. In *The 26th International Conference on Computational Linguistics*, pages 953–963.

Douglas Biber. 1993. Representativeness in corpus design. *Literary and Linguistic Computing*, 8:243–257.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45:5–32.

Cuixian Chen, Yishi Wang, Yaw Chang, and Karl Ricanek. 2012. Sensitivity analysis with cross-validation for feature selection and manifold learning. *Advances in Neural Networks–ISNN 2012*, pages 458–467.

Guillem Collell and Marie-Francine Moens. 2016. Is an image worth more than a thousand words? On the fine-grain semantic differences between visual and linguistic representations. In *The 26th International Conference on Computational Linguistics*, pages 2807–2817.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20:273–297.

Nadia FF Da Silva, Eduardo R Hruschka, and Estevam R Hruschka. 2014. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems*, 66:170–179.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 49–54.

Bradley Efron and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC press.

Xing Fang and Justin Zhan. 2015. Sentiment analysis using product review data. *Journal of Big Data*, 2:5–19.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*, volume 1. Springer Series in Statistics New York.

Balakrishnan Gokulakrishnan, Pavalanathan Priyanthan, Thiruchittampalam Ragavan, Nadarajah Prasath, and AShehan Perera. 2012. Opinion mining and sentiment analysis on a twitter data stream. In *Advances in ICT for emerging regions (ICTer), 2012 International Conference on*, pages 182–188. IEEE.

Gaoxia Jiang and Wenjian Wang. 2017. Error estimation based on variance analysis of k-fold cross-validation. *Pattern Recognition*, 69:94–106.

Thorsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learning*, pages 137–142. Springer.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing*. Prentice Hall.

Ji-Hyun Kim. 2009. Estimating classification error rate: repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, 53(11):3735–3745.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145.

Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, 46:423–444.

David Lewis. 2006. Reuters-21578. *http://www.daviddlewis.com/resources/testcollections/reuters21578*, Last accessed: 14th February 2018.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150.

Andrew Moore and Paul Rayson. 2018. Bringing replication and reproduction together with generalisability in nlp: Three reproduction studies for target dependent sentiment analysis. In *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics: Technical Papers*. The COLING 2018 Organizing Committee.

Juan D Rodriguez, Aritz Perez, and Jose A Lozano. 2010. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:569–575.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24:513–523.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). *Technical Reports (CIS)*, page 570.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective LSTMs for target-dependent sentiment classification. In *The 26th International Conference on Computational Linguistics*, pages 3298–3307.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Gitte Vanwinckelen and Hendrik Blockeel. 2015. Look before you leap: some insights into learner evaluation with cross-validation. In *Statistically Sound Data Mining*, pages 3–20.

Yongli Zhang and Yuhong Yang. 2015. Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, 187:95–112.