Perturbed Model Validation: A New Framework to Validate Model Relevance

Jie Zhang, Earl T Barr, Benjamin Guedj, Mark Harman, John Shawe-Taylor

Perturbed Model Validation: A New Framework to Validate Model Relevance

Jie M. Zhang
University College London

Earl T. Barr
University College London

Benjamin Guedj
Inria and University College London

Mark Harman
Facebook and University College London

John Shawe-Taylor
University College London

{jie.zhang,b.guedj,mark.harman,e.barr,j.shawe-taylor}@ucl.ac.uk

Abstract

This paper introduces *PMV* (Perturbed Model Validation), a new technique to validate model relevance and detect overfitting or underfitting. *PMV* operates by injecting noise to the training data, re-training the model against the perturbed data, then using the training accuracy decrease rate to assess model relevance. A larger decrease rate indicates better concept-hypothesis fit. We realise *PMV* by using label flipping to inject noise, and evaluate *PMV* on four real-world datasets (breast cancer, adult, connect-4, and MNIST) and three synthetic datasets in the binary classification setting. The results reveal that *PMV* selects models more precisely and in a more stable way than cross-validation, and effectively detects both overfitting and underfitting.

1 Introduction

Model selection chooses a statistical model from a set of candidate models based on training data [see 3, for an introduction to the field]. It is important to select a good model whose hypothesis space is close to the true concept space. Overfitting and underfitting, which result from a wrong model selection, both negatively impact the performance of machine learning models, and threaten the reliability and robustness of machine learning applications.

In applied machine learning, cross-validation is widely used to empirically select models and reduce the threat of overfitting or underfitting. Cross-validation uses test error to approximate the generalisation error. It thus relies on the implicit assumption that the test sample is at least somewhat representative of the underlying unknown distribution of data. However, a distribution shift in the test samples has been reported [9]. Furthermore, different models may have very similar cross validation results, making model selection difficult.

In statistical machine learning, VC-dimension [7] and Rademacher complexity [4] both measure the complexity of the hypothesis space. As a theoretical tools, they define upper bounds on the generalisation error, which helps model selection. However, their bounds can be difficult to compute and can be quite loose.

This paper presents *PMV* (Perturbed Model Validation), a new framework for evaluating the fit degree between the concept space [5] and the learner's hypothesis space. It perturbs the training data to

Preprint. Under review.

complicate training samples and measures the training accuracy decrease probability. The intuition is that an overfitted learner tends to fit noise in the training data, and thus may still have good training accuracy despite the noise, while an underfitted learner has poor learnability, and will have low training accuracy regardless the presence of noise. Thus, both overfitting and underfitting tend to be insensitive to noise and exhibit a small probability of accuracy decrease on perturbed data. Larger probability may indicate a better model fit and less overfitting or underfitting.

Unlike cross-validation, *PMV* does not split the data. It relates the changes in data distribution complexity to changes in the training accuracy. Unlike VC-dimension or Rademacher complexity, *PMV* does not assess the complexity of the learner directly, but rather measures whether the complexity matches the data at hand. Additionally, VC-dimension does not depend on the data-distribution; Rademacher complexity does depend on the instance but not the label distribution. In contrast, *PMV* depends on both the instance and the label distributions, so it better capture the data distribution.

In this paper, we realise *PMV* via injecting noise via flipping labels to empirically measure model relevance. *PMV* operates in the following way.

1. It injects noise into a dataset to create new perturbed versions, with different noise degrees.
2. It re-trains the machine learning model against each perturbed training dataset and computes the new training accuracy.
3. It calculates both the accuracy decrease between different perturbed datasets and the performance decrease rate $\hat{k}_S$, its final measurement.

We evaluate *PMV* on four open datasets from the UCI machine learning repository[1] (breast cancer, adult, connect-4, and MNIST), and three synthetic datasets with different data distributions, using popular classification learners such as *Random Forest*, *SVM*, *Decision Tree*, *Naive Bayes*, and *Deep Neural Networks (DNN)*. Under binary classification, we investigate whether *PMV* helps model selection, detects overfitting/underfitting, improves parameter tuning, and outperforms cross validation.

PMV aims to learn inflexible hypothesis classes where we expect bounds on Rademacher complexity to be useful. Generalisation of deep learning approaches does not fall under this umbrella; indeed, learning with random labels has been shown to be possible with some deep learners, indicating very large Rademacher complexity. Nevertheless, we design experiments to check whether *PMV* can help tune hyperparameters for deep learning.

In summary, we make the following contributions:

- **Framework**: We introduce *PMV* to measure the fit between the concept space and hypothesis space of a model, and to help model selection and detect overfitting or underfitting, using accuracy decrease rate against noise degree. In this paper, we empirically injects noise by flipping labels.
- **Empirical Evidence**: We demonstrate the effectiveness of *PMV* on four real datasets and three synthetic datasets in binary classification.

2 Perturbed Model Evaluation

We introduce *PMV* in Section 2.1 and connect it to Structural Risk Minimisation (SRM) in Section 2.2.

2.1 *PMV*

Let S be a training sample. Let r be a noise degree (ratio). S_r is a new training sample constructed by injecting r noise into S. $\mathcal{H}$ is the hypothesis set of the learner, the fixed set of hypothesis functions h that the learner actually learns. Let $\widehat{\mathrm{Acc}}_S(h)$ be the empirical training accuracy of h based on S. $\widehat{\mathrm{Acc}}(S)$ is the maximum empirical training accuracy of $h \in \mathcal{H}$ over training sample S: $\widehat{\mathrm{Acc}}(S) = \mathrm{argmax}_{h \in \mathcal{H}} \hat{Acc}_S(h)$. Let pl be the function of polynomial fit with least squares [2].

Definition 1 (*PMV* Accuracy Decrease Rate) PMV *Accuracy Decrease Rate ($\hat{k}_S$) is the polynomial coefficient of m points $(r_i, \widehat{\mathrm{Acc}}(S_{r_i}))$ where $r_i \leq 0.5$ using the least-squares polynomial fit:*

[1] `https://archive.ics.uci.edu/ml/datasets.php`

$$\hat{k}_S = pl\big((r_1, \widehat{\mathrm{Acc}}(S)), (r_1, \widehat{\mathrm{Acc}}(S_{r_1})), (r_2, \widehat{\mathrm{Acc}}(S_{r_2})), \cdots, (r_m, \widehat{\mathrm{Acc}}(S_{r_m}))\big) \tag{1}$$

The intuition of *PMV* is that an overfitted learner tends to fit noise in the training data, and thus may still have good training accuracy on the perturbed datasets, leading to a small $\hat{k}_S$. An underfitted learner has poor learnability, and will have low training accuracy no matter there is noise or not, leading to a small $\hat{k}_S$ as well. The learner with largest $\hat{k}_S$ may best fit the data.

If we treat empirical accuracy $\widehat{\mathrm{Acc}}$ as a function of noise degree r, we have $\widehat{\mathrm{Acc}} = \hat{k}_S r + \widehat{\mathrm{Acc}}(S)$. Figure 1(a) shows the relationship between $\widehat{\mathrm{Acc}}$, $\hat{k}_S$ and the noise degree r, where $\hat{k}_S$ is the slope of the fitted line.

Retraining may affect the prediction of non-perturbed points, so the accuracy decrease trend may be nonlinear. In our empirical results, however, the line is mostly linear or a curve with very small radian. Still the polynomial fit coefficient indicates the decrease rate. The absolute value of $\hat{k}_S$ is our model relevance measure.

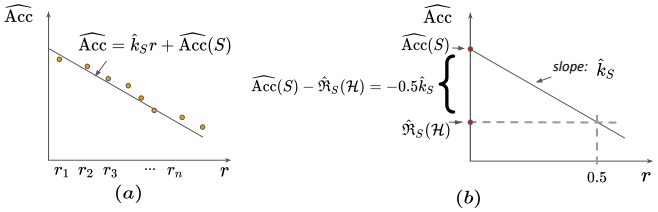

Figure 1: Relationship between accuracy $\widehat{\mathrm{Acc}}$, noise degree r, and $\hat{k}_S$.

In practice, when we randomly choose data points to perturb, we need to make sure that what we inject into S is indeed noise; when we replace $z_j \in S$ with z'_j, we want $z'_j \notin T$, the true data distribution. In this paper, *PMV* flips x_j's label producing $z'_j = (x_j, y'_j)$ with $y'_j \neq y_i$, because, with high probability, $z'_j \notin T$.

2.2 Connection with Structural Risk Minimisation

Structural Risk Minimisation (SRM) [8, 1] is an inductive principle for model selection. SRM balances empirical risk and hypothesis space complexity. Let $\mathcal{H}_1$,$\mathcal{H}_2$,...,$\mathcal{H}_n$ be a hypothesis sequence in the order of smallest to largest in complexity. Below, $\hat{R}_S(h)$ is the empirical error [5, Def 2.2]. $complexity(\mathcal{H}_i, m)$ is the hypothesis complexity. The learned whose hypothesis set $\mathcal{H}_i (0 \leq i \leq n)$ minimises the sum of empirical error and hypothesis complexity will be selected as the approximate learner [5]. Let r^{SRM} be the risk of the learner, then we have $r^{SRM} = \hat{R}_S(h) + complexity(\mathcal{H}_i, m)$. Let $\hat{\mathfrak{R}}_S$ be the empirical Rademacher complexity [5, Def 3.1]. Suppose $\mathcal{H}_i$ take values in $\{-1, +1\}$, then, for any $\delta > 0$, with probability $1 - \delta$ over sample set S of size m, we can use empirical Rademacher Complexity to assess $complexity(\mathcal{H}, m)$ and turn the formula of r^{SRM} into Equation(2):

$$r^{SRM} = \hat{R}_S(h) + \hat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}} \tag{2}$$

$$= \big(1 - \widehat{\mathrm{Acc}}_S(h)\big) + \hat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}} \tag{3}$$

$$= 1 - \big(\widehat{\mathrm{Acc}}_S(h) - \hat{\mathfrak{R}}_S(\mathcal{H})\big) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}} \tag{4}$$

$$= 1 - \big(\widehat{\mathrm{Acc}}_S(h) - E_\sigma[\operatorname*{argmin}_{h\in H_n} \frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)]\big) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}} \tag{5}$$

$$= 1 - 0.5|\hat{k}_S| + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}} \tag{6}$$

Eq.(3) replaces empirical error with empirical accuracy. Eq.(5) replaces Empirical Rademacher Complexity with its definition. In this way, the second part of Eq.(5) calculates the difference between the original empirical accuracy and Empirical Rademacher Complexity. In Eq.(5), because Empirical Rademacher Complexity equals to the expected accuracy when there is 50% noise injected, the accuracy decrease then equals $0.5\hat{k}_S$, as shown in Figure 1(c). Based on (5), we could also use $\hat{k}_S$ to help model selection to reduce risk. Larger absolute $\hat{k}_S$ indicates smaller r_S^{SRM}. Models with larger absolute $\hat{k}_S$ are preferred.

3 Experiments

3.1 Experimental Setup

We design the following research questions to check the effectiveness of *PMV* and compare *PMV* with cross validation.

RQ1: What is the performance of PMV *in model selection and overfitting/underfitting detection?*
RQ2: What is the performance of PMV *in parameter configuration for a particular model?*

We choose four popular datasets with different size, type, and feature numbers from the UCI machine learning repository[2]: *1) Breast Cancer* with 569 instances, and 32 features; *2) Adult* with 32,561 instances and 14 features. *3) Connect-4* with 42 features. To get a binary classification dataset, we remove 6,449 instances with 'draw' and keep the remaining 61,108 instances labelled 'win' and 'loss'. *4) MINST*. Image set of hand-written digits. We only use images labelled with 0 and 1, with 14,780 images. Additionally, to obtain oracles that determine model selection precision, we use datasets generated with three types of data distribution: moon, circle, and linearly-separable. We check whether *PMV* chooses the right model whose decision boundary matches the data distribution.

To compare with 10-fold cross validation, the noise degree r of each label is from 0.05 to 0.50, increasing 0.05 each time, and thus including 10 perturbed datasets. We randomly choose r positive labels and negative labels to flip respectively. We discuss more about the impact of perturbed dataset number in Section 4.

3.2 Results RQ1: Effectiveness in Model Selection

To answer the first research question, we use different models on datasets whose distribution is already known, to learn which type of model is the best fit.

The *scikit-learn* documentation provides a tutorial of comparing several classifiers on synthetic datasets. The tutorial Python script can be found on the *scikit-learn* website [6].

[2] `https://archive.ics.uci.edu/ml/index.php`

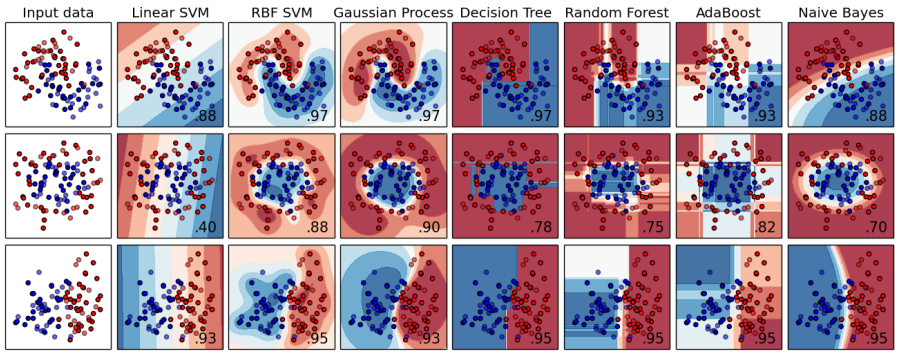

Figure 2: Classifier comparison with cross-validation (taken from [6])

The synthetic datasets follow three distributions: 1) Moon: the data points are distributed with two interleaving half circles. 2) Circle: the data points are distributed in teh form of a large circle containing a smaller circle in two dimensions. 3) Linearly-separable: the data points are linearly separable.

To better illustrate the decision boundaries of different classifiers, Figure 2 shows the generated data points for different data distribution and the decision boundaries identified by different classifiers. The classification accuracy on the test points of each classifier is shown in the lower right corner of each subfigure. We observe that the test accuracy is misleading; it may lead to the selection of overfitted models: for the linearly separable distribution, $DecisionTree$, $RandomForest$ and $AdaBoost$ have very high test accuracy, although their decision boundaries are non-linear.

To investigate the effectiveness of *PMV* in model selection and to compare *PMV* with 10-fold cross-validation, we use exactly the same setting as in the *scikit-learn* documentation. Unlike 10-fold cross-validation, *PMV* does not split the data into 10 folds, but treats all the data as a whole and injects different degrees of noise.

To investigate the effectiveness of *PMV* and cross-validation on noisy training data, we inject different degrees of noise (none, 10%, and 20% noise) when generating the input data.

Table 1 shows the results. The first column shows the data distribution. The '*PMV*' columns show the results of *PMV* on our generated datasets under different degrees of noise. $Acc.t$ is a model's test accuracy $\widehat{\mathrm{Acc}}(test)$ and labels the first 10-fold validation columns; $\Delta = |\widehat{\mathrm{Acc}}(train) - \widehat{\mathrm{Acc}}(test)|$ is the absolute difference between training and test accuracy and labels the second 10-fold validation columns. The cells in dark grey are the largest *PMV* values among all the classifiers for each generated dataset, suggesting that *PMV* deems the corresponding classifier as the best fit for that dataset. The cells in light grey are those suggesting classifiers based on test accuracy or the difference between training accuracy and test accuracy. For example, for the 'no noise' setting and moon data distribution, *PMV* suggests *RBF SVM* finds the best fit, while 10-fold cross-validation suggests *Gaussian Process*, *RBF SVM*, and *AdaBoost* find the best fit.

From Table 1, we observe the following four advantages of *PMV* over cross-validation in our experimental setting:

1) Precise model selection *PMV* selects model that fits the distribution better. For example, for linear distribution, *PMV* selects *Linear SVM*, but cross-validation selects the non-linear *Gaussian Process*.
2) Precise detection of overfitting and underfitting. For classifiers with complex decision boundaries, such as *Decision Tree* and *AdaBoost*, *PMV* gives low measurement values. However, cross-validation still has high training/test accuracy for them, and low Δ values. For underfitted classifiers, such as *Linear SVM* for the moon distribution, *PMV* also gives low measurement values.
3) Stable performance with noise in training data. *PMV* gives identical selections despite the noise injected when creating the datasets. Cross-validation gives different selections for the moon and circle distribution when there is noise.
4) Large kurtosis for easier classifier comparison. The $\hat{k}_S$ values differ much for different classifiers, making it easier to compare and select classifiers. For cross-validation, however, many classifiers have similar, even identical, training/test accuracy.

data dist.	model	no noise			0.1 noise			0.2 noise		
		PMV	10-fold CV		*PMV*	10-fold CV		*PMV*	10-fold CV	
			$Acc.t$	Δ		$Acc.t$	Δ		$Acc.t$	Δ
moon	Gaussian Process	0.61	1.00	0.00	0.41	1.00	0.00	0.18	0.97	0.03
	Decision Tree	0.17	0.96	0.04	0.17	0.91	0.09	0.13	0.94	0.06
	Naive Bayes	0.61	0.86	0.87	0.63	0.87	0.01	0.60	0.87	0.00
	Linear SVM	0.50	0.80	0.01	0.51	0.79	0.00	0.58	0.80	0.00
	RBF SVM	0.87	1.00	0.00	0.88	1.00	0.00	0.76	0.98	0.00
	AdaBoost	0.39	1.00	0.00	0.26	0.95	0.05	0.36	0.93	0.07
	Random Forest	0.10	0.95	0.05	0.14	0.94	0.06	0.07	0.93	0.07
circle	Gaussian Process	0.40	1.00	0.00	0.39	1.00	0.00	0.37	0.89	0.03
	Decision Tree	0.22	1.00	0.00	0.04	0.92	0.08	0.02	0.78	0.22
	Naive Bayes	0.88	1.00	0.00	0.79	0.98	0.02	0.59	0.87	0.05
	Linear SVM	0.27	0.37	0.15	0.22	0.37	0.15	0.22	0.37	0.15
	RBF SVM	0.68	1.00	0.00	0.60	1.00	0.00	0.40	0.87	0.05
	AdaBoost	0.37	0.99	0.01	0.14	0.95	0.05	0.11	0.82	0.18
	Random Forest	0.13	1.00	0.00	0.05	0.96	0.04	0.04	0.82	0.17
linear	Gaussian Process	0.61	0.96	0.00	0.08	0.96	0.00	0.09	0.96	0.00
	Decision Tree	0.27	0.87	0.13	0.07	0.87	0.13	0.04	0.88	0.12
	Naive Bayes	0.71	0.95	0.01	0.78	0.95	0.01	0.74	0.95	0.01
	Linear SVM	0.81	0.94	0.01	0.83	0.94	0.01	0.77	0.94	0.01
	RBF SVM	0.57	0.95	0.02	0.59	0.95	0.02	0.45	0.95	0.02
	AdaBoost	0.29	0.89	0.11	0.20	0.89	0.11	0.16	0.89	0.11
	Random Forest	0.10	0.91	0.09	0.05	0.91	0.06	0.03	0.90	0.10

Table 1: Effectiveness of *PMV* in Model Selection. The first row shows the noise degree in the original generated datasets. Dark grey cells correspond to classifiers *PMV* selects. Light grey cells correspond to classifiers cross-validation selects(based on either test accuracy or Δ).

We also check whether *PMV* could provide an overfitting degree for models that are guaranteed (by construction) to overfit. To construct such a sure-to-overfit model, we use a *Decision Tree* algorithm without setting its parameter max_depth in scikit-learn. With max_depth unset, the algorithm expands the nodes of the constructed trees until all leaves contain fewer than $min\_samples\_split$ samples. As a result, the model will generate a tree branch for every node in the limit.

The experimental results indicate that no matter how much noise is injected in the training data, the training accuracy remains to be 100%, while $\hat{k}_S$ is 0, for all the three datasets.

cross-validation is also used to detect overfitting, via checking the difference between training error and test error. With max_depth unset, for 10-fold cross-validation, we observed that the difference between training accuracy and test accuracy is 0.068, 0.188, 0.242 on the three datasets. However, it is difficult to tell how serious the overfitting is merely from these differences. Additionally, cross-validation yields unstable results with multiple runs. For example, for dataset $Adult$, we observed a difference of 0.013 between results obtained from five runs.

In conclusion, these observations on generated and real-world datasets provide a first proof of conquered sanity check that *PMV can* select models and help detect overfitting/underfitting. Compared to cross-validation, *PMV* provides more precise model selection suggestions, and stably captures the overfitting degree in extreme cases.

3.3 Results for RQ2: Performance in Hyperparameter Optimisation

3.3.1 Hyperparameter Optimisation for *Decision Tree*

In this section, we present the hyperparameter optimisation results of *PMV*, and compare the selection results with 10-fold cross-validation (using grid search).

We choose to turn parameter max_depth in *Decision Tree*. We use $DT(i)$ to denote a decision tree algorithm with the max_depth set to i, and run $DT(i)$ against the three real-world datasets. To get cross-validation results, we use function GridSearchCV provided by scikit-learn. The function returns the best parameter (set) based on grid search.

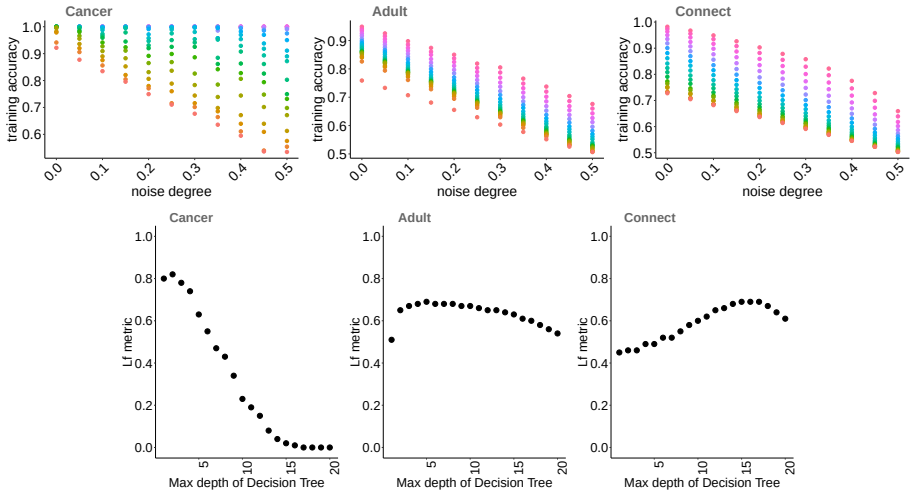

Figure 3: Performance of *PMV* when selecting depths for *Decision Tree*.

The first row of Figure 3 shows the accuracy changes for different *Decision Tree* when the noise degree increases. Different colours represent different depths. The second row shows how $\hat{k}_S$ changes as the depth increases, the y-axis is the value of absolute $\hat{k}_S$. The peak absolute $\hat{k}_S$ values correspond to the depth that *PMV* suggests.

Based on Figure 3, we observe that *PMV* could be adopted to help with hyperparameter optimisation. In particular, *PMV* stably differentiates the *Decision Tree* learners even with only one depth difference. The suggested depth by *PMV* for the three datasets are 2,5, and 16 separately; grid search gives suggested depths of 3,7, and 17 separately.

However, when we compare *PMV* and cross-validation on small datasets: the original cancer data set, 5% of the adult and connect datasets, we observe drastic fluctuation in the results of grid search. For example, on the original cancer dataset, when we apply grid search five times, we get a suggested depth list of $[3, 3, 11, 7, 9]$, while *PMV* gives 3 for all the five runs. This observation indicates that *PMV* is more stable than grid search on small datasets.

In conclusion, *PMV* provides close hyperparameter recommendations with cross-validation using grid search, but is more stable even when the dataset is small.

3.3.2 Hyperparameter Optimisation for Deep Neural Network

To check whether *PMV* could help tune the hyperparameters for DNN, we check $\hat{k}_S$ values with different numbers of hidden-layer neurons $(10, 20, .., 100, 200, ..., 800)$ when classifying the *MINST* dataset. Error curve is a typical way of tuning hyperparameters with the help of validation set. We thus also check the effectiveness of error curve approach by splitting the data into training set and validation set (7:3), and plot the training error and test error.

Figure 4 shows the results. Based on $\hat{k}_S$, the DNN should adopt 40 neurons in the model. Based on error curve, the test error stops dropping when there are 40 to 80 neurons, but then drops again on 90 neurons. It is less indicative about which depth to choose. We also observe that the $\hat{k}_S$ values are relatively low for DNN. Nevertheless, our approach is still able to choose the peak values for hyperparameter optimisation.

Additionally, when there are 200 to 600 neurons, $\hat{k}_S$ keeps dropping but the test accuracy remains stable and has a plateau, indicating that $\hat{k}_S$ may have better ability to indicate overfitting degree. When test error provides identical validation results for some hyperparameters, *PMV* provides further information.

In conclusion, in hyperparameter optimisation task, compared to error curve approach, *PMV* is more stable and easier to make a selection between different hyperparameters.

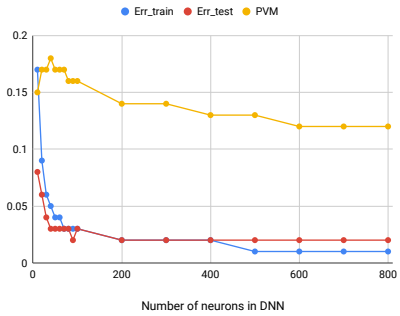

Figure 4: Comparison between training error, test error, and $\hat{k}_S$. Compared to error curve, *PMV* is more stable and easier to make a selection when optimising hyperparameters.

4 Discussion

1) *Noise injection.* Our experiment injects noise via label flipping. There are other ways of injecting noise, such as randomly flipping a label, or to change the feature values. Different ways may have different noise probability of changing the data distribution.

2) *Noise Distribution.* We have different options when choosing which instance to perturb. For example, in this paper, we randomly select a set of instances and perturb it. For random selection, when there is noise in the training set, the probability that the modification indeed injects a noise decreases, because one may modify the noise itself, making the modification unpredictable. Under this circumstance, the approximated value of $\hat{k}_S$ will be lower than without noise, as we observed from Table 1.

Additionally, it is also possible to choose the instance that appears around the decision boundary of a model, which the model may be more sensitive to.

3) *Application Scenario. PMV* is currently evaluated under supervised binary classification, but *PMV* is also capable of validating multiple-label classification tasks. For multiple-label classification, we could swap the labels with each other when generating noise.

4) *Cost of* PMV. Like n-fold cross validation, *PMV* can have different number of perturbed datasets for linear regression. Each perturbed dataset requires one training process to collect the new training accuracy. More perturbed datasets may bring higher accuracy. Developers could choose how many perturbed dataset they would like to use based on their cost concern.

In this paper, we use 10 perturbed datasets to compare *PMV* with 10-fold cross validation. However, our experiments indicate that *PMV* gives very close results even with very few perturbed datasets. For example, for dataset *connect*, *PMV* suggests depth 16 with 10 perturbed datasets and depth 15 with only 2 perturbed datasets. This could also be deduced from the data points shown by Figure 3.

5 Conclusion

We presented *PMV*, a new framework to validate a learner's performance based on the learner's reactivity to injected training data noise. The more accuracy decrease the noise brings, the better the learner is. We implemented *PMV* via flipping labels to inject noise, and evaluated it on four real-world datasets and three synthesis datasets. The results show that

- *PMV* selects models more precisely and stably than cross validation. For example, for linearly-separable synthesis data, *PMV* selects *Linear SVM* with high kurtosis, while cross-validation gives very similar results for almost all the classifiers involved.
- *PMV* effectively detects overfitting and underfitting and reflects overfitting/underfitting degree. For overfitted models such as *Random Forest* for linearly-separable data, the accuracy decrease rate $\hat{k}_S$ is very small. When there is extreme overfitting (*e.g.*, there is no maximum depth for *Decision Tree*), $\hat{k}_S = 0$.
- *PMV* helps tune hyperparameters more stably than error curve approach.

References

[1] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*, pages 2791–2801, 2018.

[2] C-J Kim. Polynomial fit of interferograms. *Applied optics*, 21(24):4521–4525, 1982.

[3] Pascal Massart. *Concentration inequalities and model selection*. Springer, 2007.

[4] Mehryar Mohri and Afshin Rostamizadeh. Rademacher complexity bounds for non-iid processes. In *Advances in Neural Information Processing Systems*, pages 1097–1104, 2009.

[5] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2018. Second edition.

[6] scikit-learn core team. scikit-learn: Classifier comparison. `https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html`, 2019. Accessed: 2019-05-1.

[7] John Shawe-Taylor, Peter L Bartlett, Robert C Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44(5): 1926–1940, 1998.

[8] V. N. Vapnik and A. Y. Chervonenkis. *Theory of Pattern Recognition: Statistical Problems of Learning*. John Wiley and Sons, 1974.

[9] Roman Werpachowski, András György, and Csaba Szepesvári. Detecting overfitting via adversarial examples. *arXiv preprint arXiv:1903.02380*, 2019.