Differential Description Length for Hyperparameter Selection in Supervised Learning

Mojtaba Abolfazli, Anders Høst-Madsen, June Zhang
Department of Electrical Engineering
University of Hawaii at Manoa, Honolulu, USA
{ahm,mojtaba,zjz}@hawaii.edu

***Abstract*—Minimum description length (MDL) is an established method for model selection. For supervised learning problems, cross-validation is often used for model selection in practice. Reasons are 1) MDL is difficult to apply directly to data; 2) MDL may make restrictive statistical assumptions that decrease performance; and 3) MDL does not directly aim to minimize generalization error. In this paper, we introduce a modification to MDL, which we call *differential description length* (DDL). DDL partitions the data so that the codelength(s) it computes, reflects the conditional probability of seeing 'new' data given 'old' data. This differential codelength is what allows DDL to estimate generalization error like cross-validation. DDL is also better than cross-validation because it allows the learning algorithm to use the entire data without having to withhold subsets for validation and testing. Compared with MDL, DDL has both better performance (in finding models with smaller generalization error) and is easier to compute. Experiments with linear regression and deep neural networks show that DDL also outperforms cross-validation.**

I. Introduction

Consider the supervised learning problem with features x and labels y. Given a set of training data $((x_1, y_1), \dots, (x_n, y_n))$, the goal is to build a predictor $f(x, \theta_h, h)$ of the y. The parameters of the model, θ_h, are estimated from the training data. The set of *hyperparameters*, h, are chosen before the learning process; hyperparameters are values that control the characteristics of θ_h. Examples of hyperparameters are the model order in regression, regularization parameters, and early stopping times [1]. Consequently, the performance and computational efficiency of learning models depend on using an optimal h.

The goal of learning is to minimize the theoretical risk (also called generalization error)

$$E_{x,y}[L(y, f(x; \theta_h, h))], \tag{1}$$

where $L(\cdot)$ is some loss function. In practice, we usually have only a subset of data available and therefore can only compute the empirical risk

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i; \theta_h, h))]. \tag{2}$$

Finding the optimal hyperparameters h by minimizing empirical risk can lead to overfitting [1], [2]. Cross-validation is a common heuristic in which the performance of the model is tested with data independent from the training data. However, there are also downsides to cross-validation: 1) having to hold out data for validation, thereby reducing the amount of data used for training the model; 2) computation complexity since we have to search through a combination of hyperparameter values; 3) lack of theoretical analysis of performance.

An ideal hyperparameter selection method should combine the ability of cross-validation to work directly with the data via generalization error and also the statistical formalism of general model selection methods such as Bayes factor/ Bayesian information criteria (BIC), Akaike information criteria (AIC), and minimum description length (MDL) [3]–[6]. In this paper, we propose just such a method called differential description length (DDL), which is a modification of MDL.

A few notes on notation. All logarithms referred to in this paper are base 2. Let (x_i^j, y_i^j) denote the set of features and labels $((x_i, y_i), (x_{i+1}, y_{i+1}), \dots, (x_j, y_j))$. Let (x^n, y^n) denote the entire set of data. We call a subset of the entire data set, $(x^m, y^m), m < n$, the *initialization* set.

II. Minimum description length (MDL) and Differential Description Length (DDL)

A. Minimum description length (MDL)

Consider a set of data with features x^n and labels y^n. Bayesian model selection [1, Section 3.4] picks the ith model that maximizes

$$P_i(y^n|x^n) = \int P(y^n|x^n, \theta_i; i) P_i(\theta_i) d\theta_i. \tag{3}$$

where $P_i(\theta_i)$ is a prior distribution. MDL can be considered as a variation of the Bayesian approach to model selection. MDL was developed in the pioneering papers by Rissanen [3]–[5]. Rather than considering the posterior probability, MDL uses the codelength $C_i(y^n|x^n)$ to choose the best ith model. MDL is closely related to the Bayesian approach due to Kraft inequality [7], which relates codelength to (sub)probability distribution

$$2^{-C_i(y^n|x^n)} = P_i(y^n|x^n),$$

which can then be used for model selection. The advantage of MDL over the Bayesian approach is that we do not need to learn/assume a prior distribution $P_i(\theta_i)$, nor do we need to calculate the integral (3).

Supported in part by NSF grant CCF-1908957.

In traditional MDL, codelength is often approximated, as the maximum likelihood (ML) solution plus a term that takes into account complexity. For example, the original formula in [4] is

$$C(y^n|x^n) = \min_{\hat{\theta}_h} -\log P(y^n|x^n; \hat{\theta}_h(x^n, y^n), h) + \frac{K}{2}\log n,$$

where $\hat{\theta}_h(x^n, y^n)$ is the ML estimator of the model parameters using the dataset (x^n, y^n), and K is the number of unknown parameters. The codelength can be used for model selection by picking the model with the shortest $C(y^n|x^n)$.

The issue with Bayes and MDL for model selection is that it does not directly consider the theoretical risk underlying a statistical learning problem. Our aim is not to find the "correct" model, or the model that makes the observed data most likely, but the model that minimizes the theoretical risk (1).

B. Cross-Validation

Cross-validation is one of the most often used methods for model selection. Cross-validation (also called K-fold cross-validation) divides the data (x^n, y^n) into K chunks. The empirical risk is computed for each of the kth chunk. The cross-validation score is the average of the K empirical risks as an estimate of the theoretical risk. We choose the model with the smallest cross-validation score.

When $K = n$, the cross-validation is called leave one out cross validation (LOOCV). This becomes prohibitively expensive for all but the simplest model.

C. Differential Description Length (DDL)

While differential description length (DDL) uses codelength like MDL, it also leverages the availability of data for estimating theoretical risk like in cross-validation. Like simple cross-validation, DDL divides the data into $(x^m, y^m), m < n$ and (x^n_{m+1}, y^n_{m+1}). To select the best model, we calculate the codelength $C_i(y^n_{m+1}|x^n, y^m)$, and choose the ith model with the shortest codelength. We call such conditional description length the *differential description length* (DDL).

Through Kraft's inequality, we can interpret the DDL codelength, $C_i(y^n_{m+1}|x^n, y^m)$, as a representation of the conditional probability, $P_i(y^n_{m+1}|x^n, y^m)$, which is the likelihood of seeing the dataset (x^n_{m+1}, y^n_{m+1}) given the "past" dataset (x^m, y^m), thereby taking into account the learning aspect of the problem compared to MDL. This can be interpreted in the Bayesian model selection context as

$$P_i(y^n_{m+1}|x^n, y^m) = \int P(y^n_{m+1}|x^n_{m+1}; \theta_i, i) P_i(\theta_i|x^m, y^m) d\theta_i. \tag{4}$$

DDL is different from cross-validation because $C_i(y^n_{m+1}|x^n, y^m)$ is not computed based on a fixed parameter $\hat{\theta}_h(x^m, y^m)$. Instead $\hat{\theta}$ is updated with (x^n_{m+1}, y^n_{m+1}), subject to a decodability condition.

In cross-validation, separate validation trials has to be done for different model parameter θ. In DDL, our estimate of θ updates iteratively as we see more and more of the data. This is because the codelength $C_i(y^n_{m+1}|x^n, y^m)$ is related to the distribution $P_i(y^n_{m+1}|x^n_{m+1}; \hat{\theta}(x^m, y^m), h)$, for which θ is iteratively updated.

III. Analysis

DDL requires that we set aside some portion of the data, $(x^m, y^m), m < n$ for initialization. In this section, we will analyze how the performance of DDL depends on $\alpha = \frac{m}{n}$. For illustration purposes, we will work with a simple model selection problem. Unfortunately, even for this simple problem, exact analysis is infeasible, so some approximations will be made.

The features x is from a finite alphabet with K symbols, while the labels y are binary. We consider two possible models: (model 1) y is independent of x, or (model 2) y is dependent on x. In the latter case, the model parameter, θ, is the set of K conditional distributions $P(y|x)$. Let

$$P(1|x) \doteq P(y = 1|x), x \in \{1, \ldots, K\}.$$

We consider the loss function to be log-loss or cross-entropy

$$G = E\left[-\log P_{\hat{\theta}}(y|x)\right]. \tag{5}$$

The optimum discriminative model is the one minimizing (5), which may not be the model that actually generated the data.

A. Model selection using exact generalization loss

For both of these models, we can calculate the generalization loss exactly, up to an $o(\cdot)$ term.

For model 1: independent model, the generalization loss is

$$\begin{aligned} G_1 &= H(Y) + D(P\|\hat{P}) \\ &= H(Y) + \frac{(\hat{P}(1|\cdot) - P(1|\cdot))^2}{P(1|\cdot)(1 - P(1|\cdot))\ln 4} + o\left(\frac{1}{n}\right) \end{aligned} \tag{6}$$

where $P(1|\cdot)$ denotes $P(y = 1)$, independent of x.

For model 2: dependent model, the generalization loss is

$$\begin{aligned} G_2 &= H(Y|X) + D(P\|\hat{P}) \\ &= \sum_{x=1}^{K} H(Y|x)P(x) + D(P(Y|x)\|\hat{P}(Y|x))P(x) \\ &= H(Y|X) + \sum_{x=1}^{K} \frac{(\hat{P}(1|x) - P(1|x))^2}{P(1|x)(1 - P(1|x))\ln 4} P(x) \\ &\quad + o\left(\frac{1}{n}\right). \end{aligned} \tag{7}$$

In both cases, H is entropy and $D(\cdot\|\cdot)$ is relative entropy [7], $\hat{P}(1|x)$ is an estimate of $P(1|x)$, which can be taken as the ML estimate, and the last equality is obtained by series expansion of $D(\cdot\|\cdot)$.

We choose the dependent model (model 2) if $G_1 - G_2 > 0$. This means we would choose model 2 if the threshold, t_{exact} is less than or equal to the mutual information, $I(Y; X)$:

$$\begin{aligned} t_{exact} &= \sum_{x=1}^{K} \frac{(\hat{P}(1|x) - P(1|x))^2}{P(1|x)(1 - P(1|x))\ln 4} P(x) \\ &\quad - \frac{(\hat{P}(1|\cdot) - P(1|\cdot))^2}{P(1|\cdot)(1 - P(1|\cdot))\ln 4} \leq I(Y; X). \end{aligned} \tag{8}$$

ignoring the $o(\cdot)$ terms.

B. Model selection using approximate generalization loss

We will now show how coding can be used for model selection. It is clear that for coding (and estimation) the data can be divided into K independent substreams corresponding to $x_i \in \{1, \ldots, K\}$. Each substream can then be coded as in [7, Section 13.2]. In practice, there are many ways codelength can be computed. The different methods all give approximately the same codelength. Therefore in this paper, we do not go into detail of the differences in computing codelength. We can use a block coder to encode the different substreams. The encoder counts the number k_x of ones in y^n for the samples where $x = x_i$. It transmits k_x using $\log n_x$ bits (n_x is the number of samples where $x = x_i$). It then tells the decoder which sequences with k_x bits were seen; this requires $\log \binom{n_x}{k_x}$ bits. The codelengths can be calculated quite accurately from [8]

$$C(y^n|x^n) = \sum_{x=1}^{K} n_x H\left(\frac{k_x}{n_x}\right) + \frac{1}{2}\log\frac{n_x}{2} + \epsilon\left(\frac{1}{n}\right) \quad (9)$$

where $H(\cdot)$ is entropy and the term $\epsilon\left(\frac{1}{n}\right) \to 0$ as $n \to \infty$.

We can estimate the generalization error from the codelength by

$$\hat{G} = \frac{C(y^n|x^n) - C(y^m|x^m)}{n-m},$$

Under model 1: independent model, the codelength (9) is

$$\begin{aligned} C(y^n|x^n) &= nH\left(\frac{k}{n}\right) + \frac{1}{2}\log\frac{n}{2} + \epsilon\left(\frac{1}{n}\right) \\ &= nH(Y) + \frac{1}{2}\log\frac{n}{2} \\ &\quad + n\log\left(P(1|\cdot)^{-1} - 1\right)\left(\hat{P}_n(1) - P(1|\cdot)\right). \end{aligned} \quad (10)$$

The approximate generalization error is

$$\begin{aligned} \hat{G}_1 &\approx H(Y) + \frac{1}{2(n-m)}\log\frac{n}{m} + \log\left(P(1|\cdot)^{-1} - 1\right) \\ &\quad \times \left(\frac{n\hat{P}_n(1|\cdot) - m\hat{P}_m(1|\cdot)}{n-m} - P(1|\cdot)\right) \end{aligned} \quad (11)$$

For model 2: dependent model, we can approximate the codelength (9) using series expansion

$$\begin{aligned} \frac{C(y^n|x^n)}{n} &= \sum_{x=1}^{K} \hat{P}(x) H\left(\hat{P}(1|x)\right) + \frac{1}{2n}\log\frac{\hat{P}(x)}{2} + \\ &\quad \frac{K}{2n}\log n + o\left(\frac{1}{n}\right) \\ &= H(Y|X) + \sum_{x=1}^{K}\Bigg[(\hat{P}(x) - P(x))H(P(1|x)) \\ &\quad + P(x)\log\left(P(1|x)^{-1} - 1\right)\left(\hat{P}(1|x) - P(1|x)\right) \\ &\quad + \frac{1}{2n}\log\frac{P(x)}{2}\Bigg] + \frac{K}{2n}\log n + o\left(\frac{1}{n}\right). \end{aligned} \quad (12)$$

The approximate generalization error derived from codelength is

$$\begin{aligned} \hat{G}_2 &\approx H(Y|X) + \frac{K}{2(n-m)}\log\frac{n}{m} \\ &\quad + \sum_{x=1}^{K}\left(\frac{n\hat{P}_n(x) - m\hat{P}_m(x)}{n-m} - P(x)\right) H(P(1|x)) \\ &\quad + \sum_{x=1}^{K} P(x)\log\left(P(1|x)^{-1} - 1\right) \\ &\quad \times \left(\frac{n\hat{P}_n(1|x) - m\hat{P}_m(1|x)}{n-m} - P(1|x)\right) \end{aligned} \quad (13)$$

We choose the dependent model (model 2) if $\hat{G}_1 - \hat{G}_2 > 0$:

$$\begin{aligned} t_{approx} &= \frac{K}{2(n-m)}\log\frac{n}{m} \\ &\quad + \sum_{x=1}^{K}\left(\frac{n\hat{P}_n(x) - m\hat{P}_m(x)}{n-m} - P(x)\right) H(P(1|x)) \\ &\quad + \sum_{x=1}^{K} P(x)\log\left(P(1|x)^{-1} - 1\right) \\ &\quad \times \left(\frac{n\hat{P}_n(1|x) - m\hat{P}_m(1|x)}{n-m} - P(1|x)\right) \\ &\quad - \frac{1}{2(n-m)}\log\frac{n}{m} - \log\left(P(1|\cdot)^{-1} - 1\right) \\ &\quad \times \left(\frac{n\hat{P}_n(1) - m\hat{P}_m(1)}{n-m} - P(1|\cdot)\right) \leq I(Y;X) \end{aligned} \quad (14)$$

C. Choosing $\alpha = \frac{m}{n}$

In this example, a reasonable measure of performance of using codelength for model selection is to compare t_{approx} with t_{exact}. If the thresholds are identical, then model selection using codelength will always result in the correct choice. We can find the α that minimizes the mean square difference of t_{approx} with t_{exact}

$$\arg\min_{\alpha} E[(t_{approx} - t_{exact})^2]. \quad (15)$$

For simplification, we ignore the high order terms

$$E[(t_{approx} - t_{exact})^2] \approx \quad = \frac{(K-1)^2}{4n^2(1-\alpha)^2}\log^2\alpha + \frac{f(K)}{n(1-\alpha)}. \quad (16)$$

where $f(K) < 3$ is some function. We can find the α that minimizes this by taking the derivative with respect to α

$$\alpha \approx \frac{(K-1)^2 W\left(\frac{nf(K)\ln 2\ln 4}{(K-1)^2}\right)}{nf(K)\ln 2\ln 4} \approx \frac{(K-1)^2}{f(K)\ln 2\ln 4}\frac{\ln n}{n} \quad (17)$$

where W is the Lambert function.

The optimum value of α converges to zero as $n \to \infty$, i.e., as the dataset gets larger, the *fraction* of prior knowledge we need to initialize the coder decreases. This might be somewhat surprising. In cross-validation, usually a fixed percentage of data is used for validation. If one thinks of the data $(x_{m+1}, y_{m+1}), \ldots, (x_n, y_n)$ as 'validation' data in DDL,

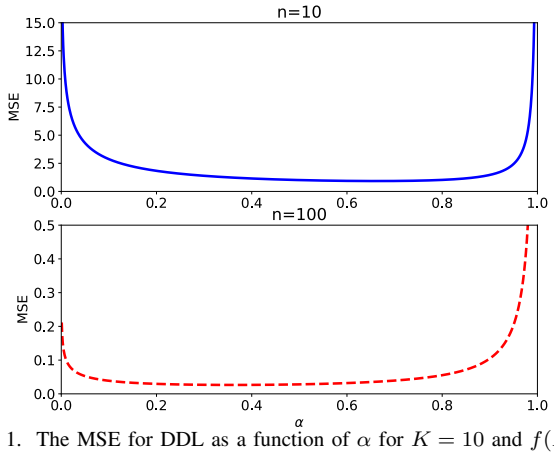

Fig. 1. The MSE for DDL as a function of α for $K = 10$ and $f(K) = 1$.

almost the entire training set is used for validation. Of course, ordinary MDL in principle corresponds to $\alpha = 0$, so in some sense this validates using MDL for machine learning. Yet, $\alpha = 0$ gives an infinite error in (16), so the take away is that the gain from DDL is to avoid this singularity. Another insight is that the optimum value of α increases as $(K-1)^2$ with the number of parameters. The factor $f(K)$ is less predictable, but $f(K)^{-1}$ is also an expression of model complexity: for example, if all probabilities are $\frac{1}{2}$ in the dependent model, $f(K) = 3$. But if y is a deterministic function of x, $f(K) = 0$. Thus, complex models require a large value of α.

The function in (16) is also quite insensitive to α. This means that the performance of DDL does not depend on any particular α value. This is most easily seen numerically, see Fig. 1. There is a large plateau from approximately $\alpha = 0.1$ to 0.8 where the error is close to the minimum.

IV. Finding Optimal Regularization Parameters

In this section, we show how DDL can be used for choosing the optimum regularization parameter, λ, in more realistic model selection scenarios. In these more complex scenarios, we do not have a way to compute the exact (or even approximately realistic) codelengths mathematically. Therefore, codelength must be derived from data by using methods that actually encode the data. These methods, such as Rissanen's predictive MDL [5], sequential normalized maximum likelihood [9], and the sufficient statistic method [10], generally proceed in a sequential fashion through the dataset.

Using the method in [5]

$$C_s(y_{m+1}^n|x^n, y^m) = -\sum_{i=m}^{n-1} \log P(y_{i+1}|x_{i+1}; \hat{\theta}_h(x^i, y^i), h), \tag{18}$$

where $\hat{\theta}_h(x^i, y^i)$ is the ML estimate derived using the dataset $(x^i, y^i) = ((x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i))$.

We will measure the performance of using codelength for estimating generalization error using *regret*

$$\text{Regret} = E_{x,y}[L(y, f(x; \theta_h, \hat{h}))] - \inf_h E_{x,y}[L(y, f(x; \theta_h, h))]. \tag{19}$$

A. Linear Regression

Consider a linear regression model where the predictor, $f(x; \theta_h, h)$, is a linear function of the features x. The unknown parameters θ_h are the weights, and the noise variance. We want to choose a predictor that minimizes the loss function

$$\min \sum_{i=1}^{n} L(y_i, f(x_i; \theta_h, h)) + \lambda ||\theta_h||_1,$$

where $L(\cdot)$ is the mean-square error loss, which is also the log loss in this case. We generate the data using the following rules. Let $X_i \sim \mathcal{N}(0, 1)$ for $i \leq 5$ and $X_i \sim \mathcal{N}(0, 10)$ for $5 < i \leq 20$, and let

$$Y = \sum_{i=1}^{K} X_i + W$$

where $W \sim \mathcal{N}(0, 1)$. Because the variance of X_i is 10 for $i > 5$, regularization is crucial to avoid including these in the sum. We compute the codelength of encoding the features and labels using (18); we assume that the conditional probability is Gaussian.

In Fig. 2 we show regret versus α. This confirms the theoretical predictions that α should be fairly small, but that the specific value is of less importance. We can also conclude that DDL is indeed better than MDL. Again recall that MDL in principle corresponds to $\alpha = 0$, although we cannot calculate it as (18) is not defined for $m = 0$. Still, as α becomes very small, there is a sharp increase in generalization error. Thus, DDL both increases performance and makes computation feasible compared to MDL.

We can also see how regret (19) depends on the proportion of initialization dataset α. Fig. 2 shows how α affects the performance of using DDL choosing λ in order to minimize generalization error. We see that the best α should be small but not too small. In this example, we pick an optimum $\alpha = 0.25$.

We compare the performance of choosing λ with DDL versus LOOCV. LOOCV is generally considered the best way to do cross-validation [2]. We run our experiment 5000 times to obtain the empirical distribution of regret (19) for both DDL and LOOCV. We can see from Fig. 3 that DDL is more likely to choose λ such that the result predictor has smaller generalization error than LOOCV.

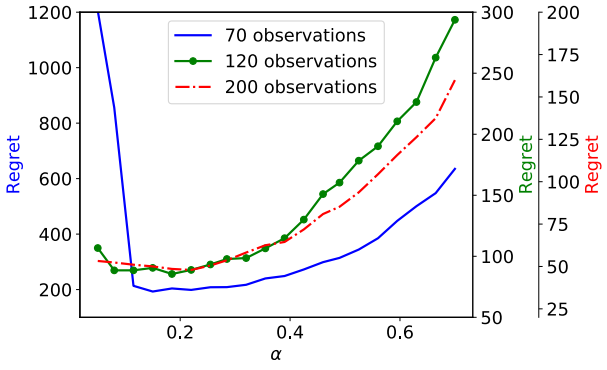

Fig. 2. Average generalization error of DDL for linear regression versus α.

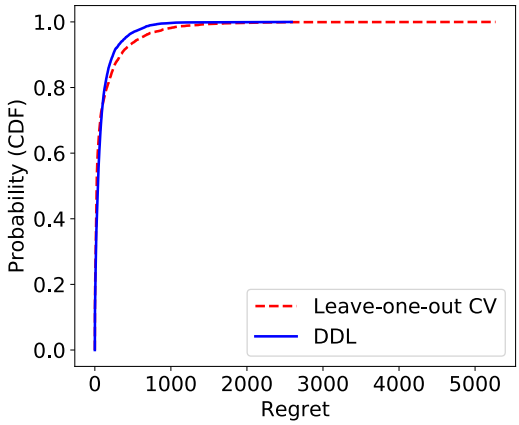

Fig. 3. Distribution of regret for linear regression with 120 observations.

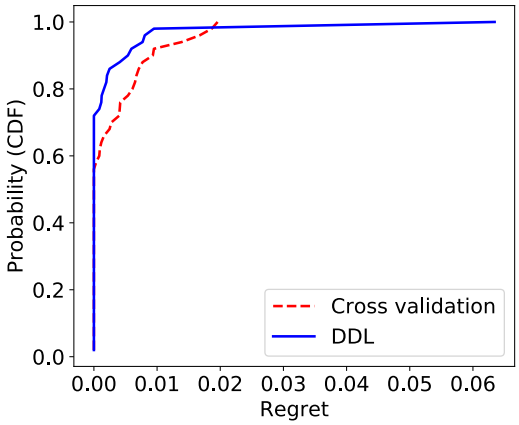

Fig. 4. Distribution of regret for binary classification on IMDB dataset.

B. Neural Networks

In this section, we show that it is feasible to use DDL to optimally select regularization parameter (L_2 in this case) for training neural networks using large dataset. We test our methodology on the IMDB dataset [11]. A multi-layer neural network composed of an encoding module followed by two fully connected layers is used. The encoding module is a multi-hot encoder that converts the input sequences of words, restricted to top 5,000 most frequently occurring words, into a vector of 0s and 1s. Each of fully connected layers are made up of 16 neurons with ReLU activation function followed by the dropout layer with the rate of 0.5. Finally, a sigmoid function is applied to map output into $[0, 1]$ range.

Since we do not know the conditional distribution of label given features, we can not compute the codelength using (18). Instead, we know the following methods to obtain a codelength. First, we notice that if the output layer is a sigmoid or softmax function, we can interpret it as a probability that can be used in (18). Second, to use (18), we need to initialize the coder with our best estimate of our model parameter using the prior knowledge data set $\hat{\theta}(x^m, y^m)$. This is more complicated for neural networks than it is for linear regression for the following reasons: 1) the unknown parameters, θ_h, for neural networks are the weights of the network. And this has very high dimensionality; 2) the loss function for linear regression is convex. Therefore, we are guaranteed that $\hat{\theta_h}(x^m, y^m)$ will converge to $\hat{\theta_h}(x^n y^n)$ as m increases. This is not a guarantee in general for neural networks.

For neural networks, our method for ensuring that $\hat{\theta_h}(x^m, y^m)$ will converge to $\hat{\theta_h}(x^n, y^n)$ as m increases is the following heuristic:

1) Train the neural network with the entire data set to find $\hat{\theta_h}(x^n, y^n)$
2) Train the neural network with a subset of the data (x^m, y^m), *but* with $\hat{\theta_h}(x^n, y^n)$ as the initialization point to obtain "related" solution
3) Using (18) to compute codelength

We compare the performance of choosing λ with DDL versus holdout cross-validation (but not LOOCV as that would be prohibitively expensive to compute) where 20% of the data is held out for validation. We run our experiment 50 times to obtain the empirical distribution of regret (19).

We can see from Fig. 4 that DDL is more likely to choose λ such that the result predictor has smaller generalization error than cross-validation.

V. Conclusion

This paper has developed the framework for DDL. DDL can be seen as a modification of MDL so it is better suited for machine learning problems. DDL has two advantages. First, the paper has shown both theoretically experimentally that DDL performs better than MDL in selecting a model with the small generalization error; this is because DDL specifically addresses generalization. Second, it has computational advantages, in that it overcomes the initialization problems of certain MDL methods.

We also have experimental results showing DDL performing better than cross-validation, even LOOCV.

References

[1] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning, 2nd Edition*. Spring, 2009.
[3] J. Rissanen, "Modeling by shortest data description," *Automatica*, pp. 465–471, 1978.
[4] ——, "A universal prior for integers and estimation by minimum description length," *The Annals of Statistics*, no. 2, pp. 416–431, 1983.
[5] ——, "Stochastic complexity and modeling," *The Annals of Statistics*, no. 3, pp. 1080–1100, Sep. 1986.
[6] P. D. Grünwald, *The minimum description length principle*. MIT press, 2007.
[7] T. M. Cover and J. A. Thomas, "Elements of information theory (2. ed.)." John Wiley & Sons, 2006.
[8] G. Shamir, "On the mdl principle for i.i.d. sources with large alphabets," *Information Theory, IEEE Transactions on*, vol. 52, no. 5, pp. 1939–1955, May 2006.
[9] T. Roos and J. Rissanen, "On sequentially normalized maximum likelihood models," in *Workshop on Information Theoretic Methods in Science and Engineering (WITMSE-08)*, 2008.
[10] E. Sabeti and A. Host-Madsen, "Enhanced mdl with application to atypicality," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017.
[11] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of ACL*, 2011.