A Review of Automatic Selection Methods for Machine Learning Algorithms and Hyper-parameter Values

Gang Luo (corresponding author)
Department of Biomedical Informatics, University of Utah, Suite 140, 421 Wakara Way, Salt Lake City, UT 84108, USA
gang.luo@utah.edu
Phone: 1-801-213-3565
Fax: 1-801-581-4297

Abstract

Machine learning studies automatic algorithms that improve themselves through experience. It is widely used for analyzing and extracting value from large biomedical data sets, or "big biomedical data;" advancing biomedical research; and improving healthcare. Before a machine learning model is trained, the user of a machine learning software tool typically must manually select a machine learning algorithm and set one or more model parameters termed hyper-parameters. The algorithm and hyper-parameter values used can greatly impact the resulting model's performance, but their selection requires special expertise as well as many labor-intensive manual iterations. To make machine learning accessible to layman users with limited computing expertise, computer science researchers have proposed various automatic selection methods for algorithms and/or hyper-parameter values for a given supervised machine learning problem. This paper reviews these methods, identifies several of their limitations in the big biomedical data environment, and provides preliminary thoughts on how to address these limitations. These findings establish a foundation for future research on automatically selecting algorithms and hyper-parameter values for analyzing big biomedical data.

Keywords: Machine learning, big biomedical data, automatic algorithm selection, automatic hyper-parameter value selection

1. Introduction

1.1 The use of machine learning in analyzing big biomedical data

Due to widespread use of applications such as electronic medical records, genomic sequencing, and mobile sensors, biomedical data are being accumulated at an exponentially growing rate annually. According to the current trend, the volume of healthcare data is expected to increase from 500 petabytes in 2012 to 25,000 petabytes by 2020 (Roski et al. 2014). Machine learning studies automatic algorithms that improve themselves through experience. It is a key technology to transform large biomedical data sets, or "big biomedical data," into actionable knowledge.

Machine learning is widely used in many biomedical applications, including predictive modeling for healthcare (Steyerberg 2009), computer-aided diagnosis, and biomedical natural language processing. For instance, machine learning models can be used to determine an individual's health risk, future behavior, or outcomes to provide appropriate and timely care, such as:

(1) Predict whether an asthma patient will be hospitalized in the following year. Enroll patients at high risk of hospitalization in an asthma case management program (Luo et al. 2015b).
(2) Predict a diabetic patient's total healthcare cost in the following year. Enroll patients at high risk of incurring high costs in a diabetes case management program (Luo et al. 2015b).
(3) Predict whether a child with clinically significant bronchiolitis will develop asthma. Schedule more frequent physician visits for children at high risk of asthma development to help physicians make timely asthma diagnoses and begin asthma treatment earlier (Luo et al. 2015a; Luo et al. 2014). An episode of bronchiolitis is clinically significant if it causes an emergency department visit, outpatient clinic visit, and/or hospitalization.
(4) In the emergency department, predict appropriate hospital admissions for bronchiolitis patients to guide disposition decisions (Luo et al. 2014; Luo et al. 2016).

The input variables used for building machine learning models can come from various sources, such as structured data in healthcare administrative systems, electronic medical records, and government databases (demographics, insurance, claims, diagnoses, allergies, immunizations, lab results, medications, smoking status, vital signs, family history, mortality record, aggregated census record, etc.), attributes extracted from genomic sequences, medical images, and clinical notes, and aggregated results from environmental monitors and mobile sensors. The data set can expand a wide spectrum in size, including from several dozen to millions of rows and from a few to several thousand attributes. Many more examples of the use of machine learning in biomedicine are described in the books (Steyerberg 2009; Cleophas and Zwinderman 2013a; Cleophas and Zwinderman 2013b; Cleophas and Zwinderman 2013c).

1.2 The state of the art of building machine learning models

To lower the entry bar to using machine learning, computer science and statistics researchers have developed open source software such as Weka (Witten et al. 2011), RapidMiner, R, and KNIME (Jovic et al. 2014) that integrate a wide variety of machine learning algorithms as well as provide an intuitive graphical user interface. Despite these efforts, it is still a challenging task to use machine learning effectively, partly because of the difficulty in manually selecting an effective combination of an algorithm and hyper-parameter values for a given supervised machine learning problem, a.k.a. model selection.

There are many machine learning algorithms, most of which are complex. Each machine learning algorithm has two types of model parameters: ordinary parameters that are automatically optimized or learned in a model training phase, and hyper-parameters that are typically set by the user of a machine learning software tool manually before a machine learning model is trained. A list of example machine learning algorithms, ordinary parameters, and hyper-parameters is shown in Table 1. Some of these examples will also be mentioned in the text below.

Table 1 A list of example machine learning algorithms, ordinary parameters, and hyper-parameters

Machine learning algorithm	Example ordinary parameters	Example hyper-parameters
Decision tree	the input variable used at each internal node, the threshold value chosen at each internal node	the minimal number of data instances at a leaf node, the pruning strategy for the tree after training
Random forest	the input variable used at each internal node of a decision tree, the threshold value chosen at each internal node of a decision tree	the number of decision trees, the number of input variables to consider at each internal node of a decision tree
Support vector machine	the support vectors, the Lagrange multiplier for each support vector	the kernel to use, the degree of a polynomial kernel, the regularization constant C, the ε for round-off error, the tolerance parameter
Neural network	the weight on each edge	the number of hidden layers, the number of nodes on each hidden layer, the number of

		epochs to train through, the learning rate for the backpropagation algorithm
k-nearest neighbor		the number of nearest neighbors used (*k*), the mechanism of weighting the nearest neighbors, the distance function to use
Naïve Bayes	the probability of an input variable taking on a particular value given a specific class $p(x_i\|c)$, the prior probability of each class $p(c)$	whether kernel density estimator or normal distribution is used for numeric attributes, the window width of a kernel density estimator
Multiboost with decision stumps	the input variable used by each decision stump, the threshold value chosen by each decision stump, the weight of each decision stump	the number of iterations, the number of sub-committees, whether resampling is used for boosting

Given a supervised machine learning problem such as predicting whether an asthma patient will be hospitalized in the following year, a researcher usually builds machine learning models in a manual and iterative way. First, the researcher manually selects a machine learning algorithm from a long list of applicable algorithms such as the 39 classification algorithms available in Weka (Thornton et al. 2013): decision tree, random forest, support vector machine, neural network, *k*-nearest neighbor, naïve Bayes, multiboost with decision stumps, etc. Second, the researcher manually determines the values of the chosen algorithm's hyper-parameters. For example, if *k*-nearest neighbor is used, the researcher needs to determine the value of *k*, the mechanism of weighting the nearest neighbors, the distance function to use, etc. As another example, if support vector machine is used, the researcher needs to determine the kernel to use, the value of the regularization constant *C*, the value of the ε for round-off error, the value of the tolerance parameter, etc. In the case of the polynomial kernel being selected, the researcher also needs to determine the polynomial's degree. Third, the researcher trains the machine learning model to automatically optimize the ordinary parameters of the chosen algorithm. If the model achieves satisfactory prediction accuracy, the model building process is complete. Otherwise, the researcher manually changes the values of the hyper-parameters and/or the algorithm and re-trains the model. This process is repeated until the researcher obtains a model with satisfactory accuracy, runs out of time, or thinks that the model's accuracy cannot be improved much further any more. Due to the enormous number of possible combinations of algorithms and hyper-parameter values, the model building process can easily take hundreds or thousands of manual iterations and is labor intensive.

Moreover, the machine learning algorithm and hyper-parameter values used affect the resulting model's accuracy, in some cases changing it from 1% to 95% (Thornton et al. 2013; Petrak 2000). As shown in Thornton et al. (2013), for the 39 algorithms available in Weka, the average change in model accuracy on 21 data sets caused by the algorithm and hyper-parameter values used is 46%. Even if only a few popular algorithms (support vector machine, random forest, decision tree, neural network, Adaboost, *k*-nearest neighbor, logistic regression) are considered, the change in model accuracy caused by the algorithm and hyper-parameter values used is still over 20% on 14 of 21 data sets. Also, the effective algorithm and hyper-parameter values vary by the specific machine learning problem (Thornton et al. 2013; Komer et al. 2014). Among the 39 algorithms available in Weka, each produces a model accuracy that is at least 22% worse than that produced by the best algorithm on at least one of 21 data sets. Selecting the effective algorithm and hyper-parameter values is currently an art requiring both deep machine learning knowledge and repeated trials. This is not only beyond the capability of layman users with limited computing expertise, but also often a non-trivial task even for machine learning experts (Sparks et al. 2015).

1.3 Automatic selection of machine learning algorithms and hyper-parameter values

To make machine learning accessible to layman users, computer science researchers have proposed various automatic selection methods for machine learning algorithms and/or hyper-parameter values for a given supervised machine learning problem. These methods' goal is to quickly find, within a pre-specified resource limit, an effective algorithm and/or combination of hyper-parameter values that maximize an accuracy measure on the given machine learning problem and data set. An example accuracy measure is area under the receiver operating characteristic curve. The resource limit is typically specified by the amount of time, the number of algorithms and/or combinations of hyper-parameter values tested on the data set, or the number of scans over the training data (Komer et al. 2014). Using an automatic selection method, the user of a machine learning software tool can skip the manual and iterative process of selecting an effective algorithm and/or combination of hyper-parameter values, which is labor intensive and requires a high skill set in machine learning. This skip applies to every supervised machine learning problem because the automatic selection methods rely on no special property of any specific problem.

Although further improvement is needed, some automatic selection methods for machine learning algorithms and/or hyper-parameter values can already find equally good or better results than careful manual tuning by machine learning experts (Komer et al. 2014; Snoek et al. 2012; Bergstra et al. 2011). This shows that automatic selection can produce meaningful results. As mentioned in Section 2.2, many automatic model selection methods in the statistics literature

focus on probabilistic or nested models and are not ideally suited for machine learning, in which many models are neither probabilistic nor nested. In this paper, we review the existing automatic selection methods from the computer science literature, present their limitations and knowledge gaps in the big biomedical data environment, and discuss specific responses to selected gaps and limitations. By establishing this foundation, we hope to stimulate future research on automatically selecting algorithms and hyper-parameter values for analyzing big biomedical data.

2. Problem Statement and Related Topics

2.1 Problem statement

Given a set of machine learning algorithms $\mathcal{A}$ and a data set D, the goal of algorithm selection is to find the algorithm $A^* \in \mathcal{A}$ having the highest generalization accuracy among all algorithms in $\mathcal{A}$. The generalization accuracy of an algorithm $A \in \mathcal{A}$ is A's accuracy for new data instances not in D. It is estimated by $M(A, D)$, the accuracy achieved by A when trained and tested on D, e.g., through stratified multi-fold cross validation to reduce the likelihood of overfitting (Witten et al. 2011). Using this estimate, the goal of algorithm selection is to find $A^* \in argmax_{A \in \mathcal{A}} M(A, D)$. In practice due to resource constraints, it is often not possible to test every algorithm in $\mathcal{A}$ thoroughly and find the algorithm with the highest generalization accuracy with full certainty. Instead, we strive to find, within a pre-specified resource limit, an algorithm that can achieve high generalization accuracy. Similarly, given an algorithm A, a hyper-parameter space Λ, and a data set D, the goal of hyper-parameter value selection is to find the combination of hyper-parameter values $\lambda^* \in \Lambda$ having the highest generalization accuracy among all combinations of hyper-parameter values in Λ. That is, $\lambda^* \in argmax_{\lambda \in \Lambda} M(A_\lambda, D)$. Here, A_λ denotes the algorithm A using the combination of hyper-parameter values λ. Put together, the goal of combined algorithm and hyper-parameter value selection is to find the algorithm $A^* \in \mathcal{A}$ and combination of hyper-parameter values $\lambda^* \in \Lambda$ having the highest generalization accuracy among all algorithms in $\mathcal{A}$ and all combinations of hyper-parameter values in Λ. That is, $A^*_{\lambda^*} \in argmax_{A \in \mathcal{A},\, \lambda \in \Lambda} M(A_\lambda, D)$.

Although machine learning is a computer science field, the problem of automatic selection of machine learning algorithms and/or hyper-parameter values has a close relationship with several fields outside of computer science. All of these fields share the commonality of using a specific quality criterion to compare different objects, with the goal of selecting a good one. In particular, many ideas from statistical model selection and traditional optimization have been borrowed in machine learning for selecting algorithms and hyper-parameter values. Before moving to this paper's focus on reviewing the automatic selection work in computer science, we briefly discuss a few aspects of the several related fields outside of computer science. The discussion is not intended to be exhaustive. Instead, it is used for describing several unique properties of automatic selection of algorithms and/or hyper-parameter values in machine learning within the context of the relationship among all of these fields.

2.2 Relationship with statistical model selection

Model selection has a large body of literature and a long history in statistics and related areas, such as econometrics and quantitative social science (Claeskens and Hjort 2008; Burnham and Anderson 2003; Hendry and Doornik 2014). In the statistics literature, model selection can have three different meanings:

(1) Selecting effective hyper-parameter values for a given algorithm/approach and modeling problem. For example, in time series analysis, for an autoregressive moving average ARMA(p, q) model we need to determine both the number of autoregressive terms p and the number of moving average terms q. As another example, for polynomial regression we need to determine the polynomial's degree.
(2) Selecting an effective algorithm/approach for a given modeling problem.
(3) Feature/variable selection (Kadane and Lazar 2004).

In this paper when discussing statistics, we use model selection to refer to the first two meanings. For machine learning, we mention selection of algorithms and selection of hyper-parameter values explicitly for clarity.

In the statistics literature, many automatic model selection methods focus on probabilistic models (particularly for Bayesian statistics (Gelman et al. 2013)) or nested models (for frequentist statistics) (Claeskens and Hjort 2008; Kadane and Lazar 2004). The methods focusing on probabilistic or nested models are not ideally suited for machine learning, as many machine learning models are neither probabilistic nor nested. Examples of such machine learning models include k-nearest neighbor, decision tree, and support vector machine. An ideal automatic selection method for machine learning algorithms and/or hyper-parameter values should cover both probabilistic and non-probabilistic models, as well as both nested and non-nested models. Some automatic model selection methods in the statistics literature are based on traditional optimization techniques. For the reasons listed in Section 2.3, many of those methods are not ideally suited for machine learning.

2.3 Relationship with traditional optimization

Optimization studies the minimization or maximization of an objective function. In optimization, frequently the goal is to find a point at which the function value is as close to the global optimum as possible by avoiding being trapped in a local optimum. Automatic selection of machine learning algorithms and/or hyper-parameter values is an optimization problem. Optimization has a large body of literature and a long history (Nocedal and Wright 2006; Bertsekas 1999). However, many traditional optimization methods are not necessarily well suited for this problem, which has its own unique properties:

(1) The optimization target $M(A, D)$ is usually a non-concave function. Hence, convex optimization methods (Boyd and Vandenberghe 2004) are not always suitable for this problem.
(2) The optimization target $M(A_\lambda, D)$ is usually non-differentiable, making gradient-based optimization methods not always suitable for hyper-parameter value selection. Also, some traditional derivative-free optimization methods (Nocedal and Wright 2006) perform poorly at selecting hyper-parameter values, partly because the optimization target lacks smoothness (Sparks et al. 2015).
(3) The choice of algorithm is a categorical variable. Also, hyper-parameters include categorical (e.g., the type of kernel in a support vector machine), discrete (e.g., the number of nodes in the first hidden layer in a neural network), and continuous variables. Thus, numerical optimization methods (Nocedal and Wright 2006; Bertsekas 1999) dealing with numerical variables are not always suitable for this problem.
(4) For a given combination of hyper-parameter values, it is usually computationally expensive to train a machine learning algorithm on a data set of moderate or large size. In other words, each function evaluation is extremely expensive (Snoek et al. 2012), particularly for big biomedical data. As a result, handling this problem often requires using techniques such as sampling of the data set to obtain approximate values of the optimization target function $M(A, D)$. An ideal automatic selection method for machine learning algorithms and/or hyper-parameter values should be able to use these approximate values. In contrast, function evaluation is inexpensive in many other black-box optimization problems. Thus, many black-box optimization methods require using exact values of the function.

2.4 Relationship with statistical model averaging and ensemble methods

Multiple base models combined together frequently outperform any single one of them. In statistics, the technique of combining multiple base models together is known as model averaging (Claeskens and Hjort 2008). In machine learning, this is known as ensemble methods (Zhou 2012). Neither model averaging nor ensemble methods eliminate the need for automatically selecting machine learning algorithms and/or hyper-parameter values. In fact, some automatic selection methods were designed specifically to address the issue of automatically forming and selecting ensembles (Thornton et al. 2013; Lacoste et al. 2014a, b).

In the statistics literature, many model averaging methods focus on probabilistic models (Claeskens and Hjort 2008). This is particularly the case for Bayesian statistics (Gelman et al. 2013). The methods focusing on probabilistic models are not ideally suited for machine learning, as many machine learning models are non-probabilistic.

In the machine learning literature, existing ensemble methods (Zhou 2012) usually do not address the issue of automatically selecting an effective combination of hyper-parameter values for each individual base model. An ensemble of multiple base models is unlikely to achieve its maximum performance without each individual base model doing so first. Moreover, existing ensemble methods frequently do not address the issue of automatically selecting the number and types of individual base models for the ensemble. Without choosing the appropriate number and types of individual base models, an ensemble is unlikely to achieve its maximum performance.

Due to the reasons mentioned above, many new methods have been developed for automatic selection of machine learning algorithms and/or hyper-parameter values. These methods are often related to, but are different from, traditional statistical model selection and optimization methods. Two other topics closely related to automatic selection of machine learning algorithms and/or hyper-parameter values are automatic feature selection (Liu and Motoda 2013) and model evaluation/validation (Steyerberg 2009; Alpaydin 2014). Each topic has its own literature that has been well covered in machine learning textbooks, and is not discussed in this paper.

3. Existing Automatic Selection Methods for Machine Learning Algorithms and/or Hyper-parameter Values

In this section, we review the existing automatic selection methods for machine learning algorithms and/or hyper-parameter values for a given supervised machine learning problem. A summary of these methods is given in Table 2. Some automatic selection methods require knowledge from experiments with previous machine learning problems, while others do not. In practice, a comprehensive database of experiments with previous machine learning problems is often unavailable. In this case, only methods that do not require such a database can be used. So far, most work on automatic selection focuses on either selecting algorithms or selecting hyper-parameter values for a specific algorithm. The first published and implemented work on automatically selecting both algorithms and hyper-parameter values appeared in 2013 (Thornton et al. 2013).

Table 2 Categorization of existing automatic selection methods for machine learning algorithms and/or hyper-parameter values

Category		**Article**	**Method**	**Can efficiently handle big data**	**Can handle a wide range of algorithms**	**Can handle various types of hyper-parameters**	**Can handle any number of hyper-parameter value combinations**
Select machine learning algorithms	Independent of previous problems	Pfahringer et al. 2000	Landmarking	×	×	×	×
		Petrak 2000	Sampling-based landmark	✓	✓	×	×
		Soares et al. 2001	Sampling-based landmark	✓	✓	×	×
		Maron and Moore 1993	Gradually expand the test set	×	✓	×	×
	Require knowledge from experiments with previous problems	Brazdil et al. 2003	Meta-learning	✓	✓	×	×
		Leite and Brazdil 2005	Sampling + meta-learning	✓	✓	×	×
		Leite and Brazdil 2010	Sampling + meta-learning	✓	✓	×	×
		van Rijn JN et al. 2015	Meta-learning	✓	✓	×	×
		Leite et al. 2012	Tournament testing + meta-learning	×	✓	×	×
Select hyper-parameter values	Independent of previous problems	Bergstra and Bengio 2012	Random search	×	×	✓	✓
		Snoek et al. 2012	Sequential model-based optimization	×	×	×	✓
		Bergstra et al. 2011	Sequential model-based optimization	×	×	×	✓
		Hutter et al. 2011	Sequential model-based optimization	×	×	✓	✓
		Wang et al. 2015	Sampling + sequential model-based optimization	✓	×	✓	✓
		Swersky et al. 2014	Freeze-thaw	×	×	×	✓
		Bengio 2000	Gradient-based	×	×	×	✓
		Guo et al. 2008	Particle swarm optimization	×	×	×	✓
		Adankon and Cheriet 2009	Gradient descent	×	×	×	✓
		Domhan et al. 2015	Sequential model-based optimization + terminate unpromising runs early	×	×	✓	✓
	Require knowledge from experiments with previous problems	Bardenet et al. 2013	Surrogate-based ranking + sequential model-based optimization	×	×	×	✓
		Swersky et al. 2013	Sequential model-based optimization	×	×	×	✓
		Yogatama and Mann 2014	Sequential model-based optimization	×	×	×	✓
		Wistuba et al. 2015a	Sequential model-based optimization	×	×	×	✓

		Wistuba et al. 2015b	Sequential model-based optimization	×	×	×	✓
Select both machine learning algorithms and hyper-parameter values	Independent of previous problems	Thornton et al. 2013	Sequential model-based optimization	×	✓	✓	✓
		Lacoste et al. 2014a	Sequential model-based optimization	×	✓	×	✓
		Lacoste et al. 2014b	Sequential model-based optimization	×	✓	✓	✓
		Komer et al. 2014	Sequential model-based optimization	×	✓	✓	✓
		Sparks et al. 2015	Multi-armed bandit + batching	×	✓	✓	✓
		Hoffman et al. 2014	Multi-armed bandit	×	✓	✓	×
		Sabharwal et al. 2016	Cost-sensitive training data allocation	✓	✓	✓	×
		Ali et al. 2014	Active learning + model selection	✓	✓	✓	×
	Require knowledge from experiments with previous problems	Feurer et al. 2015a	Sequential model-based optimization + meta-learning	×	✓	✓	✓
		Feurer et al. 2015b	Sequential model-based optimization + meta-learning	×	✓	✓	✓

3.1 Automatic selection methods for machine learning algorithms

Several methods have been proposed for automatically selecting algorithms for a given supervised machine learning problem. In this case, an algorithm's hyper-parameters are typically set to their default values (Leite et al. 2012).

3.1.1 Methods independent of previous machine learning problems

Landmarking (Pfahringer et al. 2000) is a method for automatically selecting machine learning algorithms. To quickly obtain a rough estimate of an algorithm's accuracy on a data set, the method runs a simplified version of the algorithm called a landmarker on the data set. For instance, a simplified version of a decision tree classifier is its top node. The accuracy estimates are used to select the algorithm to be used. In the selection process, we can use either the landmarkers' absolute accuracy measures or the relationship of the landmarkers' accuracy relative to each other (Fürnkranz and Petrak 2001). Often, a landmarker's accuracy cannot represent the original algorithm's accuracy well, causing a less effective algorithm to be selected.

Sampling-based landmark (Petrak 2000; Fürnkranz and Petrak 2001; Soares et al. 2001) is another method for automatically selecting machine learning algorithms. To quickly obtain a rough estimate of each algorithm's accuracy on a data set, the method applies the algorithm on a sample of the data set. The accuracy estimates are used to select the algorithm to be used on the whole data set.

Consider a fixed set of models, one per machine learning algorithm. Maron and Moore (1993) proposed a method to expedite the process of discarding bad models. First, the same training set is used to train each model. Then the test set is gradually expanded to obtain a progressively more precise estimate of each model's accuracy together with a confidence bound. When a model is clearly outperformed by another in accuracy, as determined by their confidence bounds, the former is discarded.

3.1.2 Methods requiring knowledge from experiments with previous machine learning problems

Meta-learning (Brazdil et al. 2003) is a method for automatically selecting machine learning algorithms. The method stores previous experimental results of different algorithms' accuracy on various machine learning problems and data sets. Each data set is characterized by several measures such as the number of data points present, the number of categorical attributes present, the number of numerical attributes present, and the entropy of classes. A predictive model is built on previous experimental results to predict each algorithm's accuracy on a new machine learning problem and data set. The algorithm with the highest predicted accuracy is selected.

By combining the ideas of sampling and meta-learning, Leite and Brazdil (2005) proposed a method for automatically selecting machine learning algorithms. For each algorithm and previous data set, a learning curve (Provost et al. 1999) is computed and stored. As shown in Fig. 1, the learning curve shows how the machine learning model's accuracy improves when more data are used to train the model. Consider a new machine learning problem and data set. For each algorithm, several samples of the new data set are used to train and

evaluate the machine learning model to quickly obtain an initial segment of the learning curve. A regression model is built from prior experimental results on previous data sets to predict the machine learning model's accuracy when (a large sample of) the whole new data set is used to train and evaluate the machine learning model. The algorithm with the highest predicted accuracy is selected.

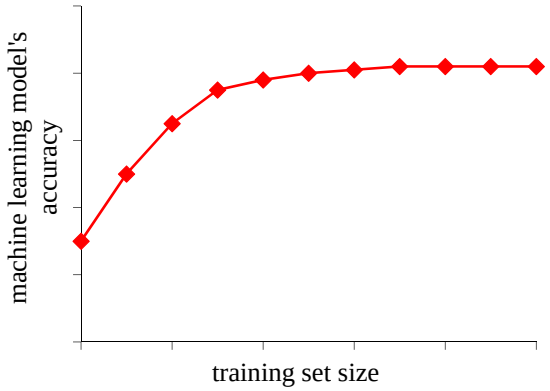

Fig. 1 An example learning curve

The above method was refined in Leite and Brazdil (2010). There, instead of following a fixed sequence of sample sizes one by one, the plan of conducting experiments on samples of the new data set is built up gradually, by considering prior experimental results on both previous data sets and the new data set. Consequently, differing sequences of sample sizes are used for different machine learning algorithms.

By considering the amount of time needed for testing a machine learning algorithm on the data set in the meta-learning process, van Rijn JN et al. (2015) proposed a method for automatically selecting algorithms. The method optimizes the ratio of forecasted accuracy to the *r*-th square root of the amount of time needed for testing an algorithm on the data set, where *r* is a pre-determined positive constant controlling the importance of time. This gives preference to algorithms that are not only likely to produce accurate machine learning models, but also likely to be evaluated quickly.

By combining the ideas of tournament testing and meta-learning, Leite et al. (2012) proposed a method for automatically selecting machine learning algorithms. The method proceeds in rounds. In each round, the similarity between the new data set and every previous data set is re-computed based on prior experimental results of different algorithms' accuracy on various machine learning problems and data sets. Among all algorithms that have been evaluated on the new data set so far, the one achieving the highest accuracy is the current best candidate algorithm. Based on data set similarities and different algorithms' accuracy on previous data sets, a challenger algorithm most likely to outperform the current best candidate algorithm on the new data set is selected. The challenger algorithm is evaluated and its accuracy is obtained on the new data set. Then the whole process is repeated until a stopping criterion is satisfied. Usually, a good algorithm can be found after a small portion of all algorithms is evaluated on the new data set.

3.2 Automatic selection methods for hyper-parameter values for a given machine learning algorithm

For a given supervised machine learning problem, a key question in comparing different machine learning algorithms is to determine whether one algorithm is fundamentally superior to another, or the former outperforms the latter just because hyper-parameters have been better tuned for the former. In addition to making machine learning accessible to layman users, automatic selection methods for hyper-parameter values also facilitate comparing different algorithms based on generalization accuracy through automatically tuning hyper-parameters (Hutter et al. 2009).

In selecting hyper-parameter values for a machine learning algorithm, both conditional and unconditional hyper-parameters need to be considered. In contrast to the case of an unconditional hyper-parameter, a conditional hyper-parameter's relevance depends on another hyper-parameter's value. For instance, in the case of support vector machine, certain kernel parameters are relevant only if the corresponding kernel type is selected. In general, as shown in Fig. 2, all hyper-parameters of an algorithm form a tree or, sometimes, a directed acyclic graph.

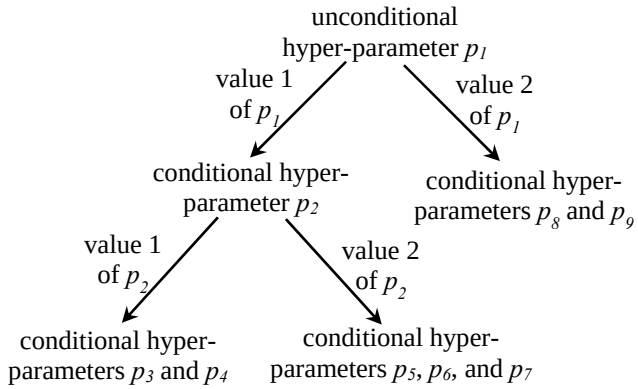

Fig. 2 An example dependency tree formed by all hyper-parameters of a machine learning algorithm

3.2.1 Methods independent of previous machine learning problems

Bergstra and Bengio (2012) demonstrated that for a given machine learning algorithm, random search is an effective method for selecting hyper-parameter values. It is more effective than an exhaustive search over a grid of hyper-parameter values. For a specific machine learning problem, usually only some hyper-parameters really matter. The others have little impact on the machine learning model's accuracy. The set of important hyper-parameters varies by the machine learning problem. A method for quantifying the importance of different hyper-parameters is described in Hutter et al. (2014).

Sequential model-based optimization (Snoek et al. 2012; Bergstra et al. 2011; Hutter et al. 2011; Shahriari et al. 2015) is a commonly used method for automatically selecting hyper-parameter values for a given machine learning algorithm. The method first tests one or more combinations of hyper-parameter values, e.g., each is a random combination or the algorithm's default, and obtains the corresponding machine learning models' accuracy on the data set. A regression model is built to predict the accuracy of a machine learning model based on hyper-parameter values. Random forest and Gaussian process are two regression models commonly used for this purpose. For a specific combination of hyper-parameter values, evaluating the regression model's output is less expensive than training the machine learning model and assessing its accuracy on the data set. When training the regression model and using it for predictions, unused conditional hyper-parameters are assigned to their default values (Thornton et al. 2013). Then the following three steps are repeated until reaching a pre-determined stopping condition: use the regression model to select a promising combination of hyper-parameter values c_o to test next; train the machine learning model and assess its accuracy a on the data set at c_o; and use the new data point (c_o, a) to modify the regression model. In reality, the regression model can be misled. To achieve good performance even in this scenario, every second combination of hyper-parameter values to assess next is randomly selected. Thus, new areas of the hyper-parameter space can be probed (Hutter et al. 2011).

In selecting a promising combination of hyper-parameter values to test next, a typical approach is to optimize a specific measure of expected improvement in accuracy (Snoek et al. 2012). Frequently, the time $t(\lambda)$ needed for testing a combination of hyper-parameter values λ on the data set varies significantly from one combination to another. As a result, this approach can be sub-optimal under a fixed total amount of time allowed for automatically selecting hyper-parameter values. A better approach is to consider both expected improvement in accuracy and $t(\lambda)$ simultaneously. Along these lines, Snoek et al. (2012) used a separate Gaussian process to model $\ln(t(\lambda))$ and optimized the ratio of expected improvement in accuracy to $t(\lambda)$. This gives preference to combinations of hyper-parameter values that are not only likely to produce accurate machine learning models, but also likely to be evaluated quickly.

Wang et al. (2015) proposed combining sampling and sequential model-based optimization to automatically select hyper-parameter values for a given machine learning algorithm. The proposed method first performs sequential model-based optimization on a relatively small random sample of the data set, which is used to quickly provide a rough estimate of the accuracy that a combination of hyper-parameter values can achieve on the whole data set. Then the top few candidate combinations of hyper-parameter values producing the highest accuracies are used to initialize sequential model-based optimization on the whole data set. If needed, the proposed method can proceed in multiple stages by gradually expanding the random sample of the data set.

Assuming that the error rate of a model roughly follows an exponential decay during the model training process, Swersky et al. (2014) developed a freeze-thaw method to automatically select hyper-parameter values for a given machine learning algorithm. At any time during the sequential model-based optimization process, the method keeps a set of partially completed models and uses their forecasted final accuracies to determine whether to freeze training an old model, continue training a partially completed model, or start training a new model with a different combination of hyper-parameter values. This saves unnecessary overhead due to continuing training unpromising, partially completed models to completion.

Bengio (2000) proposed a gradient-based method to automatically select hyper-parameter values for a given machine learning algorithm. At any step, the search direction is defined by the gradient of a model selection criterion at the current point in the hyper-parameter space.

Guo et al. (2008) used particle swarm optimization to automatically select hyper-parameter values for the machine learning algorithm of least-squares support vector machine. As a population-based optimization method, particle swarm optimization simulates the behavior of a group of birds looking for food randomly in an area.

Through minimizing an empirical error criterion, Adankon and Cheriet (2009) used a gradient descent method to automatically select hyper-parameter values for the least-squares support vector machine.

To speed up automatically selecting hyper-parameter values for deep neural networks, Domhan et al. (2015) conducted early termination of unpromising runs training the model for a combination of hyper-parameter values.

3.2.2 Methods requiring knowledge from experiments with previous machine learning problems

By combining surrogate-based ranking and sequential model-based optimization, Bardenet et al. (2013) proposed a method for automatically selecting hyper-parameter values for a given machine learning algorithm. The method stores previous experimental results of the algorithm's accuracy for different combinations of hyper-parameter values on various machine learning problems and data sets. Each data set is characterized by several measures. The method first builds a Gaussian process regression model on previous experimental results to predict the ranking of various combinations of hyper-parameter values on a new machine learning problem and data set. The ranking is based on the algorithm's accuracy. Then the following three steps are repeated until arriving at a pre-determined stopping condition: use the regression model to find the highest ranked combination of hyper-parameter values c_o to assess next; train the machine learning model and assess its accuracy a on the data set at c_o; and use the new data point (c_o, a) to modify the regression model.

Using sequential model-based optimization and experimental results on previous data sets, Swersky et al. (2013) proposed a method for automatically selecting hyper-parameter values for a given machine learning algorithm. The method builds a Gaussian process regression model, with normalized deviations from the mean per data set being the response values. The parameter estimates' posterior probabilities are computed using a maximum likelihood approach. Yogatama and Mann (2014) proposed a similar method, where the parameter estimates' posterior probabilities are computed using a Monte Carlo approach.

Using sequential model-based optimization and experimental results on previous data sets, Wistuba et al. (2015a, b) proposed two methods for automatically selecting hyper-parameter values for a given machine learning algorithm. In each iteration of the sequential model-based optimization process, the first method (Wistuba et al. 2015a) uses knowledge extracted from experimental results on previous data sets to prune regions of the hyper-parameter space unlikely to contain good combinations of hyper-parameter values. When initializing the selection process of hyper-parameter values, the second method (Wistuba et al. 2015b) determines the initial combinations of hyper-parameter values through optimizing for a hyper-parameter loss function. In this way, the initial combinations are not limited to the ones that have been tried in previous experiments. Also, meta-features of data sets are not required because data set similarities do not need to be computed.

3.3 Automatic selection methods for both machine learning algorithms and hyper-parameter values

Several papers have been published on automatically and simultaneously selecting algorithms and hyper-parameter values for a given supervised machine learning problem.

3.3.1 Methods independent of previous machine learning problems

Auto-WEKA (Thornton et al. 2013) is the first published and implemented work on automatically selecting algorithms and hyper-parameter values for a given machine learning problem. Auto-WEKA runs on one computer and is based on Weka (Witten et al. 2011), a widely used open-source machine learning and data mining toolkit written in Java. Auto-WEKA considers all 39 machine learning classification algorithms implemented in Weka. By treating the choice of algorithm as a new hyper-parameter at the root level, Auto-WEKA maps the problem of selecting algorithms and hyper-parameter values to the problem of selecting hyper-parameter values. Auto-WEKA uses sequential model-based optimization and a random forest regression model to approximate the dependence of a model's accuracy on the algorithm and hyper-parameter values. By treating the choice of feature selection technique as a hyper-parameter, Auto-WEKA can automatically choose feature selection techniques during the model building process. Auto-WEKA limits each ensemble classifier to use no more than five base classifiers. Lacoste et al. (2014a, b) extended sequential model-based optimization to remove this limitation.

Using an approach similar to that in Auto-WEKA, Komer et al. developed the software hyperopt-sklearn (Komer et al. 2014; Bergstra et al. 2013), which automatically selects machine learning algorithms and hyper-parameter values for scikit-learn (Pedregosa et al. 2011). Scikit-learn is a library of machine learning algorithms written in Python. If desired, the user can specify a narrower search space of algorithms and hyper-parameter values in hyperopt-sklearn to improve search speed. Since scikit-learn can handle only small to medium-sized data sets (Feurer et al. 2015a), any software developed on top of it will have the same limitation.

MLbase (Sparks et al. 2015; Kraska et al. 2013) is the first published work on automatically selecting algorithms and hyper-parameter values for a given machine learning problem that supports distributed computing on a cluster of commodity computers. MLbase is based on MLlib (Kraska et al. 2013; Sparks et al. 2013), Spark's machine learning library. Spark (Zaharia et al. 2010) is a widely used open source big data software system supporting Google's MapReduce framework (Dean and Ghemawat 2004) for distributed computing. Spark was developed on top of the Hadoop distributed file system, the open source implementation of Google's BigTable file system (Chang et al. 2006). To improve performance, Spark executes most operations in memory and avoids disk inputs/outputs whenever possible.

MLbase is a framework yet to be fully implemented. It is designed to find a reasonably good combination of a machine learning algorithm and hyper-parameter values early, so that the user can experiment with it. The combination is continuously refined via additional exploration in the background for further improvement. In this way, the process becomes more interactive.

To improve efficiency, MLbase batches together training of multiple models, one per combination of a machine learning algorithm and hyper-parameter values. This achieves better central processing unit (CPU) utilization and amortizes the overhead of task scheduling and network latency in a distributed computing environment. Typically, many passes through the training data are needed to train a model. Based on this observation, MLbase uses a multi-armed bandit method (White 2013) to allocate resources among different combinations of algorithms and hyper-parameter values. Initially, a fixed number of passes through the training data are allocated to each combination. Based on the quality of the model trained through these passes, MLbase decides whether to allocate additional passes to the combination to train the model further, e.g., to completion. In this way, unpromising combinations are pruned early on.

By modeling correlations among the arms, Hoffman et al. (2014) developed a multi-armed bandit method to automatically select machine learning algorithms and hyper-parameter values. This method evaluates a combination of an algorithm and hyper-parameter values on a small fixed percentage of the whole data set. Sabharwal et al. (2016)

developed a cost-sensitive training data allocation method to automatically select machine learning algorithms and hyper-parameter values. That method evaluates a combination of an algorithm and hyper-parameter values initially on a small random sample of the data set and gradually expands the random sample over time for re-evaluating the combination if it looks promising. By combining active learning with model selection, Ali et al. (2014) developed a method for actively sampling data requiring labeling to train a pre-determined set of candidate models, each of which corresponds to a combination of an algorithm and hyper-parameter values, and to select a good model from that set simultaneously. For every algorithm, each of the three methods can handle only a pre-determined set of combinations of hyper-parameter values. In practice, the set is likely to miss many better combinations of hyper-parameter values for the algorithm, causing all three methods to be sub-optimal for selecting algorithms and hyper-parameter values.

3.3.2 Methods requiring knowledge from experiments with previous machine learning problems

When searching algorithms and hyper-parameter values for a new machine learning problem, Feurer et al. (2015b) used meta-learning to choose a good starting point for sequential model-based optimization. The method stores the best combination of an algorithm and hyper-parameter values that has been found for each previous machine learning problem and data set. Each data set is characterized by several measures. Based on the similarity between the new data set and every previous data set, several previously stored highest ranked combinations are used to initialize sequential model-based optimization. Feurer et al. (2015b) demonstrated that within the same resource limit, this starting point can lead to better search results than one using one or more random or user-defined default combinations.

By improving the approach used in Auto-WEKA, Feurer et al. (2015a) developed the software auto-sklearn to automatically select machine learning algorithms and hyper-parameter values for scikit-learn. The method uses meta-learning to choose a good starting point for sequential model-based optimization. Noticing that meta and ensemble models taking base models as inputs (Thornton et al. 2013) are slow to train, the method ignores meta and ensemble models in the sequential model-based optimization process. Instead, after completing the process, the method uses the approach described in Caruana et al. (2004) to automatically construct an ensemble model from the base models found during the process with a low overhead. By treating the choices of feature selection and data pre-processing techniques as hyper-parameters, auto-sklearn can automatically choose feature selection and data pre-processing techniques during the model building process.

The Google Prediction API (Google 2016) is Google's work for machine learning problems with some degree of automation. Its internal workings are unpublished. Also, it limits the maximum training data size to 2.5GB. A list of similar, commercially available machine learning services was provided in Feurer et al. (2015a).

4. Limitations of Existing Methods in the Big Biomedical Data Environment and Opportunities for Improvement

In practice, it is insufficient to select only machine learning algorithms or hyper-parameter values for a specific algorithm. What is most needed is to automatically and simultaneously select both algorithms and hyper-parameter values for a given supervised machine learning problem. The discussion in this section focuses on such automatic selection methods. Automatic selection can save effort for machine learning tool users, reduce the machine learning skill required of users, and improve model accuracy (Komer et al. 2014; Snoek et al. 2012; Bergstra et al. 2011).

Other things being equal, we prefer automatic selection methods independent of previous machine learning problems. In reality and particularly in biomedicine, a comprehensive database of experiments with previous machine learning problems is often unavailable, creating difficulty in using methods that require knowledge from experiments with previous machine learning problems. Biomedical data sets have different characteristics from non-biomedical ones. Even if experimental results on non-biomedical data sets are available, their use for automatically selecting machine learning algorithms and hyper-parameter values on biomedical data sets can be limited. An ideal method should support distributed computing for scalable parallel processing on a cluster of commodity computers. This is critical for finishing the analysis of big biomedical data in a reasonable amount of time. So far, MLbase (Sparks et al. 2015; Kraska et al. 2013) is the only method designed specifically for distributed computing.

Similar to MLbase, an ideal automatic selection method should find a reasonably good combination of a machine learning algorithm and hyper-parameter values as quickly as possible. Realizing that a good combination can reach only low accuracy can prompt examination of feature engineering and/or other solutions. Then time does not need to be spent on continually searching for a much better combination unlikely to exist. If time allows, additional fine-tuning of the best combination identified so far can be performed in the background.

4.1 Limitations of existing methods

In the big biomedical data environment, existing automatic selection methods for machine learning algorithms and hyper-parameter values have limitations in efficiency. When a wide range of algorithms is considered, none of the existing methods can effectively select algorithms and hyper-parameter values for a large biomedical data set in a short amount of time. This limits these methods' practical usefulness.

A fundamental obstacle to automatic selection is the long time required to test a combination of a machine learning algorithm and hyper-parameter values on the whole data set.

To find a good combination, existing methods test many combinations on the whole data set. This can take several days on a data set with a moderate number of data points and attributes (Thornton et al. 2013).

In practice, search time can be much longer for three reasons. First, machine learning is an iterative process. If a designated set of biomedical attributes produces low prediction accuracy, the analyst is likely to consider other available but unused biomedical attributes that may have predictive power. Each iteration requires a new search. Second, a data set can contain a large number of data points, e.g., from multiple healthcare systems. Third, a data set can have a large number of attributes, such as those extracted from genomic and/or textual data. A machine learning algorithm's execution time often grows superlinearly with the number of data points and at least linearly with the number of attributes. To realize personalized medicine, thousands of predictive modeling problems must be solved for various diseases and outcomes. Search time will be a bottleneck in this case, regardless of whether it is an issue for a single predictive modeling problem. As machine learning software keeps incorporating more algorithms and algorithms with more hyper-parameters (e.g., deep neural network) keep getting created and used, the issue of search time will become more severe in the future.

4.2 Opportunities for improvement

New approaches are needed for improving the efficiency of selecting algorithms and hyper-parameter values for a specific supervised machine learning problem on a large biomedical data set. Within a fixed resource limit, the more efficiently a search is performed, the better the search result quality reflected by the machine learning model's accuracy. In the following, we provide some preliminary thoughts on how to use progressive sampling (Provost et al. 1999) to improve search efficiency (Luo 2015).

Our idea is to conduct inexpensive trials on small samples of the data set to eliminate unpromising combinations of machine learning algorithms and hyper-parameter values as much and as early as possible, and to devote more computational resources to fine-tuning promising combinations. More specifically, we use a relatively small random sample of the data set to test various combinations and find multiple promising combinations as the basis of a reduced search space. We then expand the random sample, test and adjust these combinations on the expanded sample, and find fewer promising combinations as the basis of a further reduced search space. We repeat this process for several rounds. As the random sample expands, the search space shrinks. In the last round, we use the whole data set to find a good combination of an algorithm and hyper-parameter values.

There are various approaches for determining the initial sample's size and for expanding the sample over rounds. One possible approach is to set the initial sample's size to the number of input variables of the model multiplied by a pre-determined constant, such as 10, and then to expand the sample size exponentially over rounds (Provost et al. 1999). Sampling has been used before to select either machine learning algorithms (Petrak 2000; Leite et al. 2012; Fürnkranz and Petrak 2001; Soares et al. 2001; Leite and Brazdil 2005; Provost et al. 1999; Leite and Brazdil 2010; John and Langley 1996; Gu et al. 2001) (sometimes each with a pre-determined set of combinations of hyper-parameter values (Hoffman et al. 2014; Sabharwal et al. 2016)) or hyper-parameter values (Wang et al. 2015), but not for selecting algorithms and hyper-parameter values simultaneously without limiting the candidate combinations of hyper-parameter values for an algorithm to a pre-determined set.

To realize personalized medicine, multiple predictive modeling problems, each with its own prediction target, often need to be solved using the same data set or overlapping portions of it. An example case is that each problem uses a set of clinical attributes to predict a different outcome on patients from the same healthcare system. Overlap exists among the sets of clinical attributes used in these problems. In the presence of overlap, we would expect some degree of similarity among the effective machine learning algorithms and hyper-parameter values for these problems. This property can be used to expedite the process of selecting algorithms and hyper-parameter values for these problems (Swersky et al. 2013). For instance, we can solve these problems one by one. The effective algorithms and hyper-parameter values found for the previous problems are used to generate a good starting point of the search process for the current problem (Feurer et al. 2015b). Within the same resource limit, a good starting point can lead to better search results than a mediocre one (Feurer et al. 2015b).

Besides the difficulty in selecting a good combination of an algorithm and hyper-parameter values for a given supervised machine learning problem, biomedical researchers face another challenge using machine learning effectively. Raw clinical data are typically stored in the Entity-Attribute-Value format (Nadkarni 2011), e.g., using a table schema of (hospital admission id, lab test id, test result value) for lab tests. These data must be transformed by pivot operations into the standard relational table format (Luo and Frey 2016), e.g., one lab test per column, before machine learning can be performed. Since machine learning is an iterative process, such data extraction is often performed repeatedly. Each round of data extraction requires work from a computing professional (Einbinder et al. 2001), creating dependency. To address this issue, it would be desirable to build new software supporting the process of iterative machine learning on big biomedical data, including clinical parameter extraction (Luo and Frey 2016), feature construction, algorithm and hyper-parameter value selection, model building, model evaluation (Luo 2015), and explanation of machine learning classification/prediction results (Luo et al. 2015b; Luo 2016). Such software would enable biomedical researchers with limited computing expertise to perform machine learning effectively. This will open the use of big biomedical data to

thousands of biomedical researchers and increase the ability to foster biomedical discovery and improve healthcare.

In healthcare, efficiently building accurate machine learning models is not the final goal. Instead, model building is part of the process to achieve the final goal of improving health outcomes and reducing costs. Often, a patient has bad outcomes or incurs high cost for multiple seasons, some at the patient level and others at the system level. To effectively improve outcomes and reduce cost, healthcare professionals need to know these reasons and provide tailored interventions. To facilitate this, we recently proposed a method to automatically explain machine learning classification/prediction results without losing accuracy and suggest tailored interventions at both the patient and system level (Luo et al. 2015b; Luo 2016). For example, if we discover that a patient is likely to have bad outcomes because he/she lives in a low-income neighborhood and cannot afford expensive medications, we can give him/her special discounts for some of the medications. As another example, if we discover that a patient is likely to have bad outcomes because he/she lives far from his/her physician, we can provide transportation or telemedicine for him/her. As a third example, if we discover that many patients have bad outcomes because they live in an area with no primary care clinic nearby, we can recommend opening a new primary care clinic in this area. So far, our method has been demonstrated on a single test case (Luo 2016). It would be an interesting area for future work to fine tune our method and test it on more cases.

5. Conclusions

Automating machine learning model selection is a hot topic in computer science with an active open competition ongoing (Guyon et al. 2015). We reviewed the literature on automatic selection methods for machine learning algorithms and/or hyper-parameter values for a given supervised machine learning problem. Our results show that these methods have limitations in the big biomedical data environment. Future studies will need to address these limitations to achieve better results.

Conflict of interest

The author reports no conflicts of interest.

Acknowledgments

We thank Qing T. Zeng, Michael Conway, Philip J. Brewster, David E. Jones, Angela P. Presson, Yue Zhang, Tom Greene, Alun Thomas, and Selena B. Thomas for helpful discussions.

References

Adankon MM, Cheriet M (2009) Model selection for the LS-SVM. Application to handwriting recognition. *Pattern Recognition* 42(12):3264-70.

Ali A, Caruana R, Kapoor A (2014) Active learning with model selection. *Proc. AAAI* 2014:1673-9.

Alpaydin E (2014) *Introduction to Machine Learning*, 3rd ed. The MIT Press, Cambridge, MA, USA.

Bardenet R, Brendel M, Kégl B, Sebag M (2013) Collaborative hyperparameter tuning. *Proc. ICML* 2013:199-207.

Bengio Y (2000) Gradient-based optimization of hyperparameters. *Neural Computation* 12(8):1889-1900.

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. *Proc. NIPS* 2011:2546-54.

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13:281-305.

Bergstra J, Yamins D, Cox DD (2013) Hyperopt: a Python library for optimizing the hyperparameters of machine learning algorithms. *Proc. SciPy* 2013:13-20.

Bertsekas DP (1999) *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont, MA, USA.

Boyd S, Vandenberghe L (2004) *Convex Optimization*. Cambridge University Press, Cambridge, UK.

Brazdil P, Soares C, da Costa JP (2003) Ranking learning algorithms: using IBL and meta-learning on accuracy and time results. *Machine Learning* 50(3):251-77.

Burnham KP, Anderson DR (2003) *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, 2nd ed. Springer, New York, NY, USA.

Caruana R, Niculescu-Mizil A, Crew G, Ksikes A (2004) Ensemble selection from libraries of models. *Proc. ICML* 2004.

Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M et al. (2006) Bigtable: a distributed storage system for structured data. *Proc. OSDI* 2006:205-18.

Claeskens G, Hjort N (2008) *Model Selection and Model Averaging*. Cambridge University Press, Cambridge, UK.

Cleophas TJ, Zwinderman AH (2013a) *Machine Learning in Medicine*. Springer, New York, NY, USA.

Cleophas TJ, Zwinderman AH (2013b) *Machine Learning in Medicine: Part Two*. Springer, New York, NY, USA.

Cleophas TJ, Zwinderman AH (2013c) *Machine Learning in Medicine: Part Three*. Springer, New York, NY, USA.

Dean J, Ghemawat S (2004) MapReduce: Simplified data processing on large clusters. *Proc. OSDI* 2004:137-50.

Domhan T, Springenberg JT, Hutter F (2015) Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. *Proc. IJCAI* 2015:3460-8.

Einbinder JS, Scully KW, Pates RD, Schubart JR, Reynolds RE (2001) Case study: a data warehouse for an academic medical center. *J Healthc Inf Manag*. 15(2):165-75.

Feurer M, Klein A, Eggensperger K, Springenberg J, Blum M, Hutter F (2015a) Efficient and robust automated machine learning. *Proc. NIPS* 2015:2944-52.

Feurer M, Springenberg T, Hutter F (2015b) Initializing Bayesian hyperparameter optimization via meta-learning. *Proc. AAAI* 2015:1128-35.

Fürnkranz J, Petrak J (2001) An evaluation of landmarking variants. *Proc. ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning* 2001:57-68.

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013) *Bayesian Data Analysis*, 3rd ed. Chapman and Hall/CRC, Boca Raton, FL, USA.

Google Prediction API homepage (2016) https://cloud.google.com/prediction/docs. Accessed January 20, 2016.

Gu B, Liu B, Hu F, Liu H (2001) Efficiently determining the starting sample size for progressive sampling. *Proc. ECML* 2001:192-202.

Guo XC, Yang JH, Wu CG, Wang CY, Liang YC (2008) A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. *Neurocomputing* 71(16-18):3211-5.

Guyon I, Bennett K, Cawley GC, Escalante HJ, Escalera S, Ho TK, Macià N, Ray B, Saeed M, Statnikov AR, Viegas E (2015) Design of the 2015 ChaLearn AutoML challenge. *Proc. IJCNN* 2015:1-8.

Hendry DF, Doornik JA (2014) *Empirical Model Discovery and Theory Evaluation: Automatic Selection Methods in Econometrics*. The MIT Press, Cambridge, MA, USA.

Hoffman MD, Shahriari B, de Freitas N (2014) On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. *Proc. AISTATS* 2014:365-74.

Hutter F, Hoos H, Leyton-Brown K (2014) An efficient approach for assessing hyperparameter importance. *Proc. ICML* 2014:754-62.

Hutter F, Hoos HH, Leyton-Brown K, Stützle T (2009) ParamILS: An automatic algorithm configuration framework. *J. Artif. Intell. Res.* 36:267-306.

Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential model-based optimization for general algorithm configuration. *Proc. LION* 2011:507-23.

John GH, Langley P (1996) Static versus dynamic sampling for data mining. *Proc. KDD* 1996:367-70.

Jovic A, Brkic K, Bogunovic N (2014) An overview of free software tools for general data mining. *Proc. MIPRO* 2014:1112-7.

Kadane JB, Lazar NA (2004) Methods and criteria for model selection. *J. Am. Stat. Assoc*. 99(465):279-90.

Komer B, Bergstra J, Eliasmith C (2014) Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. *Proc. SciPy* 2014:33-9.

Kraska T, Talwalkar A, Duchi JC, Griffith R, Franklin MJ, Jordan MI (2013) MLbase: a distributed machine-learning system. *Proc. CIDR* 2013.

Lacoste A, Larochelle H, Marchand M, Laviolette F (2014a) Sequential model-based ensemble optimization. *Proc. UAI* 2014:440-8.

Lacoste A, Marchand M, Laviolette F, Larochelle H (2014b) Agnostic Bayesian learning of ensembles. *Proc. ICML* 2014:611-9.

Leite R, Brazdil P (2005) Predicting relative performance of classifiers from samples. *Proc. ICML* 2005:497-503.

Leite R, Brazdil P (2010) Active testing strategy to predict the best classification algorithm via sampling and metalearning. *Proc. ECAI* 2010:309-14.

Leite R, Brazdil P, Vanschoren J (2012) Selecting classification algorithms with active testing. *Proc. MLDM* 2012:117-31.

Liu H, Motoda H (2013) *Feature Selection for Knowledge Discovery and Data Mining*. Springer, New York, NY, USA.

Luo G (2015) MLBCD: a machine learning tool for big clinical data. *Health Inf Sci Syst*. 3:3.

Luo G, Frey LJ (2016) Efficient execution methods of pivoting for bulk extraction of Entity-Attribute-Value-modeled data. *IEEE J Biomed Health Inform.* 20(2):644-54.

Luo G, Nkoy FL, Gesteland PH, Glasgow TS, Stone BL (2014) A systematic review of predictive modeling for bronchiolitis. *Int J Med Inform*. 83(10):691-714.

Luo G, Nkoy FL, Stone BL, Schmick D, Johnson MD (2015a) A systematic review of predictive models for asthma development in children. *BMC Med Inform Decis Mak*. 15(1):99.

Luo G, Stone BL, Sakaguchi F, Sheng X, Murtaugh MA (2015b) Using computational approaches to improve risk-stratified patient management: rationale and methods. *JMIR Res Protoc*. 4(4):e128.

Luo G (2016) Automatically explaining machine learning prediction results: a demonstration on type 2 diabetes risk prediction. *Health Inf Sci Syst*. 4:2.

Luo G, Stone BL, Johnson MD, Nkoy FL (2016) Predicting appropriate admission of bronchiolitis patients in the emergency room: rationale and methods. *JMIR Res Protoc*. 5(1):e41.

Maron O, Moore AW (1993) Hoeffding races: Accelerating model selection search for classification and function approximation. *Proc. NIPS* 1993:59-66.

Nadkarni PM (2011) *Metadata-driven Software Systems in Biomedicine: Designing Systems that can Adapt to Changing Knowledge*. Springer, New York, NY, USA.

Nocedal J, Wright S (2006) *Numerical Optimization*, 2nd ed. Springer, New York, NY, USA.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O et al. (2011) Scikit-learn: machine learning in Python. *Journal of Machine Learning Research* 12:2825-30.

Petrak J (2000) Fast subsampling performance estimates for classification algorithm selection. *Proc. ECML Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination* 2000:3-14.

Pfahringer B, Bensusan H, Giraud-Carrier CG (2000) Meta-learning by landmarking various learning algorithms. *Proc. ICML* 2000:743-50.

Provost FJ, Jensen D, Oates T (1999) Efficient progressive sampling. *Proc. KDD* 1999:23-32.

Roski J, Bo-Linn GW, Andrews TA (2014) Creating value in health care through big data: opportunities and policy implications. *Health Aff.* (Millwood) 33(7):1115-22.

Sabharwal A, Samulowitz H, Tesauro G (2016) Selecting near-optimal learners via incremental data allocation. *Proc. AAAI* 2016.

Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N (2015) Taking the human out of the loop: a review of Bayesian optimization. *Proceedings of the IEEE* 104(1):148-75.

Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. *Proc. NIPS* 2012:2960-8.

Soares C, Petrak J, Brazdil P (2001) Sampling-based relative landmarks: systematically test-driving algorithms before choosing. *Proc. EPIA* 2001:88-95.

Sparks ER, Talwalkar A, Haas D, Franklin MJ, Jordan MI, Kraska T (2015) Automating model search for large scale machine learning. *Proc. SoCC* 2015: 368-80.

Sparks ER, Talwalkar A, Smith V, Kottalam J, Pan X, Gonzalez JE et al. (2013) MLI: an API for distributed machine learning. *Proc. ICDM* 2013:1187-92.

Steyerberg EW (2009) *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating*. Springer, New York, NY, USA.

Swersky K, Snoek J, Adams RP (2013) Multi-task Bayesian optimization. *Proc. NIPS* 2013:2004-12.

Swersky K, Snoek J, Adams RP (2014) Freeze-thaw Bayesian optimization. http://arxiv.org/abs/1406.3896. Accessed January 20, 2016.

Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. *Proc. KDD* 2013:847-55.

van Rijn JN, Abdulrahman SM, Brazdil P, Vanschoren J (2015) Fast algorithm selection using learning curves. *Proc. IDA* 2015:298-309.

Wang L, Feng M, Zhou B, Xiang B, Mahadevan S (2015) Efficient hyper-parameter optimization for NLP applications. *Proc. EMNLP* 2015:2112-7.

White JM (2013) *Bandit Algorithms for Website Optimization*. O'Reilly Media, Sebastopol, CA, USA.

Wistuba M, Schilling N, Schmidt-Thieme L (2015a) Hyperparameter search space pruning - a new component for sequential model-based hyperparameter optimization. *Proc. ECML/PKDD* (2) 2015:104-19.

Wistuba M, Schilling N, Schmidt-Thieme L (2015b) Learning hyperparameter optimization initializations. *Proc. DSAA* 2015:1-10.

Witten IH, Frank E, Hall MA (2011) *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, Burlington, MA, USA.

Yogatama D, Mann G (2014) Efficient transfer learning method for automatic hyperparameter tuning. *Proc. AISTATS* 2014:1077-85.

Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I (2010) Spark: cluster computing with working sets. *Proc. HotCloud* 2010.

Zhou Z (2012) *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC, Boca Raton, FL, USA.