Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Model Selection: Beyond the Bayesian/Frequentist Divide

Isabelle Guyon GUYON@CLOPINET.COM
ClopiNet, 955 Creston Road, Berkeley, CA 94708, USA

Amir Saffari SAFFARI@ICG.TUGRAZ.AT
Graz University of Technology, Austria

Gideon Dror GIDEON@MTA.AC.IL
Academic College of Tel-Aviv-Yaffo, Israel

Gavin Cawley GCC@CMP.UEA.AC.UK
University of East Anglia, UK

Editor: Lawrence Saul

Abstract

The principle of parsimony also known as "Ockham's razor" has inspired many theories of model selection. Yet such theories, all making arguments in favor of parsimony, are based on very different premises and have developed distinct methodologies to derive algorithms. We have organized challenges and edited a special issue of JMLR and several conference proceedings around the theme of model selection. In this editorial, we revisit the problem of avoiding overfitting in light of the latest results. We note the remarkable convergence of theories as different as Bayesian theory, Minimum Description Length, bias/variance tradeoff, Structural Risk Minimization, and regularization, in some approaches. We also present new and interesting examples of the complementarity of theories leading to hybrid algorithms, neither frequentist, nor Bayesian, or perhaps both frequentist and Bayesian!

Keywords: Model selection, Ensemble methods, Multilevel inference, Multilevel optimization, Performance prediction, Bias-variance tradeoff, Bayesian priors, Structural risk minimization, Guaranteed Risk minimization, Over-fitting, Regularization, Minimum Description Length.

1. Introduction

The problem of learning is often decomposed into the tasks of fitting parameters to some training data, and then selecting the best model using heuristic or principled methods, collectively referred to as *model selection* methods. Model selection methods range from simple yet powerful cross-validation based methods to the optimization of cost functions penalized for model complexity, derived from performance bounds or Bayesian priors.

This paper is not intended as a general review of the state-of-the-art in model selection nor a tutorial; instead it is a synthesis of the collection of papers that we have assembled. It also provides a unifying perspective on Bayesian and frequentist methodologies used in

various model selection methods. We highlight a new trend in research on model selection that blends these approaches.

The reader is expected to have some basic knowledge of familiar learning machines (linear models, neural networks, tree classifiers and kernel methods) and elementary notions of learning theory (bias/variance tradeoff, model capacity or complexity, performance bounds). Novice readers are directed to the companion paper (Guyon, 2009), which reviews basic learning machines, common model selection techniques, and provides elements of learning theory.

When we started organizing workshops and competitions around the problem of model selection (of which this collection of papers is the product), both theoreticians and practitioners welcomed us with some scepticism; model selection being often viewed as somewhat "old hat". Some think that the problem is solved, others that it is not a problem at all! For Bayesian theoreticians, the problem of model selection is circumvented by averaging all models over the posterior distribution. For risk minimization theoreticians (called "frequentists" by the Bayesians) the problem is solved by minimizing performance bounds. For practitioners, the problem is solved using cross-validation. However, looking more closely, most theoretically grounded methods of solving or circumventing model selection have at least one hyper-parameter left somewhere, which ends up being optimized by cross-validation. Cross-validation seems to be the universally accepted ultimate remedy. But it has its dark sides: (a) there is no consensus on how to choose the fraction of examples reserved training and for validation; (b) the overall learning problem may be prone to over-fitting the cross-validation error (Cawley and Talbot, 2009). Therefore, from our point of view, the problem of optimally dividing the learning problem into multiple levels of inference and optimally allocating training data to these various levels remains unsolved, motivating our efforts. From the novel contributions we have gathered, we are pleased to see that researchers are going beyond the usual Bayesian/frequentist divide to provide new creative solutions to those problems: we see the emergence of multi-level optimization methods, which are both Bayesian and frequentist. How can that be? Read on!

After explaining in Section 2 our notational conventions, we briefly review a range of different Bayesian and frequentist approaches to model selection in Section 3, which we then unify in Section 4 under the framework of multi-level optimization. Section 5 then presents the advances made by the authors of papers that we have edited. In Section 6, we open a discussion on advanced topics and open problems. To facilitate reading, a glossary is appended; throughout the paper, words found in the glossary are indicated in boldface.

2. Notations and conventions

In its broadest sense, *model selection* designates an ensemble of techniques used to select a model, that best explains some data or phenomena, or best predicts future data, observations or the consequences of actions. This broad definition encompasses both scientific and statistical modeling. In this paper, we address only the problem of statistical modeling and are mostly concerned with **supervised learning** from independently and identically distributed (*i.i.d.*) data. Extensions to **unsupervised learning** and non *i.i.d.* cases will be discussed in Section 6.

The goal of supervised learning is to predict a target variable $y \in \mathcal{Y}$, which may be continuous (regression) or categorical or binary (classification). The predictions are made using observations $\mathbf{x}$ from a domain $\mathcal{X}$, often a vectorial space of dimension n, the number of features. The data pairs $\{\mathbf{x}, y\}$ are independently and identically distributed according to an unknown (but fixed) distribution $P(\mathbf{x}, y)$. A number m of pairs drawn from that distribution are given, forming the training data $D = \{(\mathbf{x}_k, y_k), k = 1, ...m\}$. We will denote by $X = [x_{ki}]$, $k = 1, ...m, i = 1, ...n$, the matrix of dimensions (m, n) whose rows are the training patterns and whose columns are the features. Finally, we denote by $\mathbf{y}$ the column vector of dimensions $(m, 1)$ containing the target values y_k.

There are several formulations of the supervised learning problem:

- **Function approximation** (induction) methods seek a function f (called *model* or *learning machine*) belonging to a model class $\mathcal{F}$, which minimizes a specified risk functional (or maximizes a certain utility). The goal is to minimize an *expected risk* $R[f] = \int \mathcal{L}(f(\mathbf{x}), y)\ dP(\mathbf{x}, y)$, also called *generalization error*, where $\mathcal{L}(f(\mathbf{x}), y)$ is a loss function (often a negative log likelihood) measuring the discrepancy between $f(\mathbf{x})$ and y. Since $P(\mathbf{x}, y)$ is unknown, only estimates of $R[f]$ can be computed, which we call *evaluation functions* or *estimators*. Function approximation methods differ in the choice of evaluation function and optimization algorithm and include **risk minimization**, **PAC leaning**, **maximum likelihood optimization**, and **MAP learning**.

- **Bayesian and ensemble methods** make predictions according to model averages that are convex combinations of models $f \in \mathcal{F}$, *i.e.,* which belong to the convex closure of the model class $\mathcal{F}^*$. Such methods differ in the type of model averaging performed. **Bayesian learning** methods approximate $E_f(y|\mathbf{x}) = \int_{f \in \mathcal{F}} f(\mathbf{x})\ dP(f)$, an expectation taken over a class of models $\mathcal{F}$, using an unknown probability distribution $P(f)$ over the models. Starting from a "prior", our knowledge of this distribution is refined into a "posterior" when we see some data. **Bagging** ensemble methods approximate $E_D(f(\mathbf{x}, D))$, where $f(\mathbf{x}, D)$ is a function from the model class $\mathcal{F}$, trained with m examples and $E_D(\cdot)$ is the mathematical expectation over all training sets of size m. The key point in these methods is to generate a diverse set of functions, each providing a different perspective over the problem at hand, the ensemble thus forming a concensus view.

- **Transduction** methods make direct predictions of y given $\mathbf{x}$ and X, bypassing the modeling step. We do not address such methods in this paper.

The desired properties of the chosen predictor include: good generalization performance, fast training/prediction, and ease of interpretation of the predictions. Even though all of these aspects are important in practice, we will essentially focus on the first aspect: obtaining the best possible generalization performance. Some of the other aspects of model selection will be discussed in Section 6.

The parametrization of f differentiates the problem of *model selection* from the general machine learning problem. Instead of parameterizing f with one set of parameters, the model selection framework distinguishes between *parameters* and *hyper-parameters*. We adopt the simplified notation $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$ for a model of parameters $\boldsymbol{\alpha}$ and hyper-parameters

$\boldsymbol{\theta}$. It should be understood that different models may be parameterized differently. Hence by $f(\mathbf{x}; \boldsymbol{\alpha}\boldsymbol{\theta})$ we really mean $f(\mathbf{x}; \boldsymbol{\alpha}(\boldsymbol{\theta}), \boldsymbol{\theta})$ or $f_{\boldsymbol{\theta}}(\mathbf{x}; \boldsymbol{\alpha})$. For instance, for a linear model $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$, $\boldsymbol{\alpha} = \mathbf{w}$; for a kernel method $f(\mathbf{x}, \boldsymbol{\alpha}) = \sum_k \alpha_k K(\mathbf{x}, \mathbf{x}_k)$, $\boldsymbol{\alpha} = [\alpha_k]$. The hyper-parameters may include indicators of presence or absence of features, choice of preprocessing methods,, choice of algorithm or model sub-class (*e.g.,* linear models, neural networks, kernel methods, etc.), algorithm or model sub-class parameters (*e.g.,* number of layers and units per layer in a neural network, maximum degree of a polynomial, bandwidth of a kernel), choice of post-processing, etc. We also refer to the parameters of the prior $P(f)$ in **Bayesian/MAP learning** and the parameters of the **regularizer** $\Omega[f]$ in **risk minimization** as hyper-parameters even if the resulting predictor is not an explicit function of those parameters, because they are used in the process of learning. In what follows, we relate the problem of model selection to that of *hyper-parameter selection*, taken in is broadest sense and encompassing all the cases mentioned above.

We refer to the adjustment of the model parameters $\boldsymbol{\alpha}$ as the *first level of inference*. When data are split in several subsets for the purpose of training and evaluating models, we call m_{tr} the number of *training examples* used to adjust $\boldsymbol{\alpha}$. If the hyper-parameters $\boldsymbol{\theta}$ are adjusted from a subset of data of size m_{va}, we call the examples used to adjust them at this *second level of inference* the *"validation sample"*. Finally we call m_{te} the number of test examples used to evaluate the final model. The corresponding empirical estimates of the *expected risk* $R[f]$, denoted $R_{tr}[f]$, $R_{va}[f]$, and $R_{te}[f]$, will be called respectively *training error*, *validation error*, and *test error*.

3. The many faces of model selection

In this section, we track *model selection* from various angles to finally reduce it to the unified view of *multilevel inference*.

3.1 Is model selection "really" a problem?

It is legitimate to first question whether the distinction between parameters and hyper-parameters is relevant. Splitting the learning problem into two levels of inference may be convenient for conducting experiments. For example, combinations of preprocessing, feature selection, and post-processing are easily performed by fixing $\boldsymbol{\theta}$ and training $\boldsymbol{\alpha}$ with off-the-shelf programs. But, the distinction between parameters and hyper-parameters is more fundamental. For instance, in the model class of kernel methods $f(\mathbf{x}) = \sum_k \boldsymbol{\alpha}_k K(\mathbf{x}, \mathbf{x}_k; \boldsymbol{\theta})$, why couldn't we treat both $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ as regular parameters?

One common argument is that, for fixed values of $\boldsymbol{\theta}$, the problem of learning $\boldsymbol{\alpha}$ can be formulated as a convex optimization problem, with a single unique solution, for which powerful mathematical programming packages are available, while the overall optimization of $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ in non-convex. Another compelling argument is that, splitting the learning problem into several levels might also benefit to the performance of the learning machine by "alleviating" (but not eliminating) the problem of **over-fitting**. Consider for example the Gaussian redial basis function kernel $K(\mathbf{x}, \mathbf{x}_k; \theta) = \exp(-\|\mathbf{x} - \mathbf{x}_k\|^2/\theta^2)$. The function $f(\mathbf{x}) = \sum_{k=1}^m \alpha_k K(\mathbf{x}, \mathbf{x}_k; \theta)$ is a universal approximator if θ is let to vary and if the sum runs over the training examples. If both $\boldsymbol{\alpha}$ and θ are optimized simultaneously, solutions with a small value of θ^2 might be picked, having zero training error but possibly very poor general-

ization performance. The model class $\mathcal{F}$ to which f belongs has infinite capacity $C(\mathcal{F})$. In contrast, for a fixed value of the hyper-parameter θ^o, the model $f(\mathbf{x}) = \sum_{k=1}^{m} \boldsymbol{\alpha}_k K(\mathbf{x}, \mathbf{x}_k; \theta^o)$ is linear in its parameters $\boldsymbol{\alpha}_k$ and has a finite capacity, bounded by m. In addition, the capacity of $f(\mathbf{x}) = \sum_{k=1}^{m} \boldsymbol{\alpha}_k^o K(\mathbf{x}, \mathbf{x}_k^o; \theta)$ of parameter θ for fixed values $\boldsymbol{\alpha}_k^o$ and x_k^o is very low (to see that, note that very few examples can be learned without error by just varying the kernel width, given fixed vectors x_k^o and fixed parameters $\boldsymbol{\alpha}_k^o$). Hence, *using multiple levels of inference may reduce over-fitting, while still searching for solutions in a model class of universal approximators.*

This last idea has been taken one step further in the method of *structural risk minimization* (Vapnik, 1979), by introducing new hyper-parameters in learning problems, which initially did not have any. Consider for instance the class of linear models $f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i$. It is possible to introduce hyper-parameters by imposing a structure in parameter space. A classical example is the structure $\|\mathbf{w}\|^2 \leq A$, where $\|\mathbf{w}\|$ denotes the Euclidean norm and A is a positive hyper-parameter. For increasing values of A the space of parameters is organized in nested subsets. Vapnik (1998) proves for Support Vector Machines (SVM) and Bartlett (1997) for neural networks that tighter performance bounds are obtained by increasing A. The newly introduced parameter allows us to monitor the **bias/variance tradeoff**. Using a Lagrange multiplier, the problem may be replaced by that of minimizing a regularized risk functional $R_{reg} = R_{tr} + \gamma \|\mathbf{w}\|^2$, $\gamma > 0$, where the training loss function is the so-called "hinge loss" (see *e.g.,* Hastie et al., 2000). The same regularizer $\|\mathbf{w}\|^2$ is used in ridge regression (Hoerl, 1962), "weight decay" neural networks (Werbos, 1988), regularized radial-basis function networks (Poggio and Girosi, 1990), Gaussian processes (MacKay, 1992), together with the square loss function. Instead of the Euclidean norm or 2-norm, the 1-norm regularizer $\|\mathbf{w}\|_1 = \sum_i |w_i|$ is used in LASSO (Tibshirani, 1994) and 1-norm versions of SVMs (see *e.g.,* Zhu et al., 2003), logistic regression (Friedman et al., 2009), and Boosting (Rosset et al., 2004). Weston et al. (2003) have proposed a 0-norm regularizer $\|\mathbf{w}\|_0 = \sum_i \mathbf{1}(w_i)$, where $\mathbf{1}(x) = 1$, if $x \neq 0$ and 0 otherwise.

Interestingly, each method stems from a different theoretical justification (some are Bayesian, some are frequentist and some a a little bit of both like PAC-Bayesian bounds, see *e.g.,* Seeger, 2003, for a review), showing a beautiful example of theory convergence (Guyon, 2009). Either way, for a fixed value of the hyper-parameter A or γ the complexity of the learning problem is lower than that of the original problem. We can optimize A or γ at a second level of inference, for instance by cross-validation.

3.2 Bayesian model selection

In the Bayesian framework, there is no model selection *per se*, since learning does not involve searching for an optimum function, but averaging over a posterior distribution. For example, if the model class $\mathcal{F}$ consists of models $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$, the Bayesian assumption is that the parameters $\boldsymbol{\alpha}$ and hyper-parameters $\boldsymbol{\theta}$ of the model used to generate the data are drawn from a prior $P(\boldsymbol{\alpha}, \boldsymbol{\theta})$. After observing some data D the predictions should be made according to:

$$E_{\boldsymbol{\alpha},\boldsymbol{\theta}}(y|\mathbf{x}, D) = \int\int f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})\ P(\boldsymbol{\alpha}, \boldsymbol{\theta}|D)\ d\boldsymbol{\alpha}\ d\boldsymbol{\theta}\ . \tag{1}$$

Hence there is no selection of a single model, but a summation over models in the model class $\mathcal{F}$, weighed by $P(\boldsymbol{\alpha}, \boldsymbol{\theta}|D)$. The problem is to integrate over $P(\boldsymbol{\alpha}, \boldsymbol{\theta}|D)$.[1] A two-level decomposition can be made by factorizing $P(\boldsymbol{\alpha}, \boldsymbol{\theta}|D)$ as $P(\boldsymbol{\alpha}, \boldsymbol{\theta}|D) = P(\boldsymbol{\alpha}|\boldsymbol{\theta}, D)P(\boldsymbol{\theta}|D)$:

$$E_{\boldsymbol{\alpha},\boldsymbol{\theta}}(y|\mathbf{x}, D) = \int \left(\int f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta}) P(\boldsymbol{\alpha}|\boldsymbol{\theta}, D) \; d\boldsymbol{\alpha} \right) P(\boldsymbol{\theta}|D) \; d\boldsymbol{\theta} \; . \tag{2}$$

Bayesian model selection decomposes the prior $P(\boldsymbol{\alpha}, \boldsymbol{\theta})$ into parameter prior $P(\boldsymbol{\alpha}|\boldsymbol{\theta})$ and a "hyper-prior" $P(\boldsymbol{\theta})$. In **MAP learning**, the type-II likelihood (also called the "evidence") $P(D|\boldsymbol{\theta}) = \sum_{\boldsymbol{\alpha}} P(D|\boldsymbol{\alpha}, \boldsymbol{\theta}) P(\boldsymbol{\alpha}|\boldsymbol{\theta})$ is maximized with respect to the hyper-parameters $\boldsymbol{\theta}$ (therefore assuming a flat prior for $\boldsymbol{\theta}$), while the "regular" parameters $\boldsymbol{\alpha}$ are obtained by maximizing the posterior $\boldsymbol{\alpha}^* = \text{argmax}_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}|\boldsymbol{\theta}, D) = \text{argmax}_{\boldsymbol{\alpha}} P(D|\boldsymbol{\alpha}, \boldsymbol{\theta}) P(\boldsymbol{\alpha}|\boldsymbol{\theta})$.[2]

3.3 Frequentist model selection

While Bayesians view probabilities as being (realized in the idea of "prior" and "posterior" knowledge of distributions), frequentists define probability in terms of *frequencies of occurrence of events*. In this section, the "frequentist" approach is equated with risk minimization.

There are obvious ties between the problem of *model selection* and that of *performance prediction*. *Performance prediction* is the problem of estimating the *expected risk* or *generalization error* $R[f]$. *Model selection* is the problem of adjusting the capacity or complexity of the models to the available amount of training data to avoid either **under-fitting** or **over-fitting**. Solving the *performance prediction* problem would also solve the *model selection* problem, but model selection is an easier problem. If we find an ordering index $r[f]$ such that for all pairs of functions $r[f_1] < r[f_2] \Rightarrow R[f_1] < R[f_2]$, then the index allows us to correctly carry out *model selection*. Theoretical performance bounds providing a *guaranteed risk* have been proposed as ranking indices (Vapnik, 1998). Arguably, the tightness of the bound is of secondary importance in obtaining a good ranking index. Bounds of the form $r[f] = R_{tr}[f] + \epsilon(C/m_{tr})$, where C characterizes the capacity or complexity of the model class, penalizes complex models, but the penalty vanishes as $m_{tr} \to \infty$. Some learning algorithms, *e.g.,* SVMs (Boser et al., 1992) or boosting (Freund and Schapire, 1996), optimize a *guaranteed risk* rather than the *empirical risk* $R_{tr}[f]$, and therefore provide some guarantee of good *generalization*. Algorithms derived in this way have an embedded model selection mechanism. Other closely related penalty-based methods include Bayesian **MAP learning** and **regularization**.

Many models (and particularly *compound models* including feature selection, preprocessing, learning machine, and post-processing) are not associated with known performance bounds. Common practice among frequentists is to split available training data into m_{tr} training examples to adjust parameters and m_{va} validation examples to adjust

1. The calculation of the integral in closed form may be impossible to carry out; in this case, variational approximations are made or numerical simulations are performed, sampling from $P(\boldsymbol{\alpha}, \boldsymbol{\theta}|D)$, and replacing the integral by the summation over a finite number of models.
2. In some Bayesian formulations of multi-layer Perceptrons, the evidence framework maximizes over $\boldsymbol{\theta}$ but marginalises over the weights, rather than maximizing, so in this case the MAP can apply to the parameters or the hyper-parameters or both.

hyper-parameters. In an effort to reduce variance, the validation error $R_{va}[f]$ may be averaged over many data splits, leading to a cross-validation (CV) estimator $R_{CV}[f]$. The most widely used CV method is *K-fold cross-validation*. It consists in partitioning training data into $K \simeq (m_{tr} + m_{va})/m_{va}$ disjoint subsets of roughly equal sizes (up to rounding errors), each corresponding to one validation set (the complement being used as training set). In stratified cross-validation, the class proportions of the full datasets are respected in all subsets. The variance of the results may be reduced by performing Q times K-fold cross-validation and averaging the results of the Q runs. Another popular method consists in holding out a single example at a time for validation purposes. The resulting cross-validation error is referred to as "leave-one-out" error $R_{LOO}[f]$. Some preliminary study design is necessary to determine the sufficient amount of test data to obtain a good estimate of the generalization error (Langford, 2005), the sufficient amount of training data to attain desired generalization performances, and an adequate split of the training data between training and validation set. See (Guyon, 2009) for a discussion of these issues.

4. Multi-level inference: a unifying view of model selection

What is common among the various views of model selection is the idea of multiple levels of inference, each level corresponding to one set of parameters or hyper-parameters. Consider a two-level case for a model class $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$ parameterized by one set of parameters $\boldsymbol{\alpha}$ and one set of hyper-parameters $\boldsymbol{\theta}$. From the frequentist (risk minimization) point of view, instead of jointly optimizing a risk functional with respect to all parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, one creates a hierarchy of optimization problems:[3]

$$\boxed{f^{**} = \text{argmin}_{\boldsymbol{\theta}} R_2[f^*, D] \ , \quad \text{such that} \quad f^* = \text{argmin}_{\boldsymbol{\alpha}} R_1[f, D]} \tag{3}$$

where R_1 and R_2 are first and second level risk functionals.

From the Bayesian point of view, the goal is to estimate the integral of Equation 2. There are striking similarities between the two approaches. To make the similarity more obvious, we can rewrite Equation 2 to make it look more like Equation 3, using the notation f^{**} for $E_{\boldsymbol{\alpha},\boldsymbol{\theta}}(y|\mathbf{x}, D)$:

$$\boxed{f^{**} = \int f^* \ e^{-R_2} \ d\boldsymbol{\theta} \ , \quad \text{such that} \quad f^* = \int f \ e^{-R_1} \ d\boldsymbol{\alpha}} \tag{4}$$

where $R_1 = -\ln P(\boldsymbol{\alpha}|\boldsymbol{\theta}, D)$ and $R_2 = -\ln P(\boldsymbol{\theta}|D)$. Note that in Bayesian multi-level inference f^* and f^{**} do not belong to $\mathcal{F}$ but to $\mathcal{F}^*$, the closure of $\mathcal{F}$ under convex combinations.

More generally, we define a *multi-level inference problem* as a learning problem organized into a hierarchy of learning problems. Formally, consider a machine learning toolkit which includes a choice of learning machines $\mathcal{A}[\mathcal{B}, R]$, where $\mathcal{B}$ is a model space of functions $f(\mathbf{x}; \boldsymbol{\theta})$, of parameters $\boldsymbol{\theta}$ and R is an evaluation function (*e.g.,* a risk functional or a negative log posterior). We think of $\mathcal{A}[\mathcal{B}, R]$ not as a procedure, but as an "object", in the sense of object

3. It would be more correct if the argmin was assigned to parameters not functions, since the search domain is over parameters, and write $\boldsymbol{\theta}^{**} = \arg\min_{\boldsymbol{\theta}} R_2[f^*, D]$, such that $\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} R_1[f, D], f^* = f(\mathbf{x}, \boldsymbol{\alpha}^*)$, but we adopt a shorthand to emphasize the similarities between the frequentist and Bayesian approaches.

oriented programming, equipped with a method "train", which processes data according to a training algorithm[4]:

$$f^{**} = \text{train}(\mathcal{A}[\mathcal{B}, R_2], D); \tag{5}$$

This framework embodies the second level of inference of both Equations 3 and 4. The solution f^{**} belongs to $\mathcal{B}^*$, the convex closure of $\mathcal{B}$. To implement the first level of inference, we will consider that $\mathcal{B}$ is itself a learning machine and not just a model space. Its model space $\mathcal{F}$ includes functions $f(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\alpha})$ of variable parameters $\boldsymbol{\alpha}$ ($\boldsymbol{\theta}$ is fixed), which are adjusted by the "train" method of $\mathcal{B}$:

$$f^{*} = \text{train}(\mathcal{B}[\mathcal{F}, R_1], D); \tag{6}$$

The solution f^* belongs to $\mathcal{F}^*$, the convex closure of $\mathcal{F}$. The method "train" of $\mathcal{A}$ should call the method "train" of $\mathcal{B}$ as a subroutine, because of the nested nature of the learning problems of Equations 3 and 4. Notice that it is possible that different subsets of the data D are used at the different levels of inference.

We easily see two obvious extensions:

(i) *Multi-level inference:* Equation 5 and 6 are formally equivalent, so this formalism can be extended to more than two levels of inference.

(ii) *Ensemble methods:* The method "train" returns either a single model or a linear combination of models, so the formalism can include all ensemble methods.

We propose in the next section a new classification of *multi-level inference* methods, orthogonal to the classical Bayesian *vs.* frequentist divide, referring to the way in which data are processed rather than the means by which they are processed.

5. Advances in multi-level inference

We dedicate this section to reviewing the methods proposed in the collection of papers that we have edited. We categorize multi-level inference modules, each implementing one level of inference, into *filter*, *wrapper*, and *embedded methods*, borrowing from the conventional classification of feature selection methods (Kohavi and John, 1997; Blum and Langley, 1997; Guyon et al., 2006a). *Filters* are methods for narrowing down the model space, without training the learning machine. Such methods include preprocessing, feature construction, kernel design, architecture design, choice of prior or regularizers, choice of a noise model, and filter methods for feature selection. They constitute the highest level of inference[5]. *Wrapper methods* consider the learning machine as a black-box capable of learning from examples and making predictions once trained. They operate with a search algorithm in hyper-parameter space (for example grid search or stochastic search) and an evaluation function assessing the trained learning machine performances (for example the cross-validation error or the

4. We adopt a Matlab-style notation: the first argument is the object of which the function is a method; the function "train" is overloaded, there is one for each algorithm. The notations are inspired and adapted from the conventions of the Spider package and the CLOP packages (Saffari and Guyon, 2006).
5. Preprocessing is often thought of as a "low-level" operation. However, with respect to model selection, the selection of preprocessing happens generally in the "outer loop" of selection, hence it is at the highest level.

Bayesian evidence). They are the *middle-ware* of multi-level inference. *Embedded* methods are similar to wrappers, but they exploit the knowledge of the learning machine algorithm to make the search more efficient and eventually jointly optimize parameters and hyper-parameters, using multi-level optimization algorithms. They are usually used at the lowest level of inference.

5.1 Filters

Filter methods include a broad class of techniques aiming to reduce the model space $\mathcal{F}$ prior to training the learning machine. Such techniques may use "prior knowledge" or "domain knowledge", data from prior studies or from R&R (repeatability and reproducibility) studies, and even the training data themselves. But they do not produce the final model used to make predictions. Several examples of filter methods are found in the collection of papers we have edited:

Preprocessing and feature construction. An important part of machine learning is to find a good data representation, but choosing an appropriate data representation is very domain dependent. In benchmark experiments, it has often been found that generating a large number of low-level features yields better result than hand-crafting a few features incorporating a lot of expert knowledge (Guyon et al., 2007). The feature set can then be pruned by feature selection. In the challenges we have organized (Clopinet, 2004-2009) the data were generally already preprocessed to facilitate the work of the participants. However, additional normalizations, space dimensionality reduction and discretization were often performed by the participants. Of all space dimensionality reduction methods *Principal Component Analysis* (PCA) remains the most widely used. Several top-ranking participants to challenges we organized used PCA, including Neal and Zhang (2006), winners of the NIPS 2003 feature selection challenge, and Lutz (2006), winner of the WCCI 2006 performance prediction challenge. *Clustering* is also a popular preprocessing method of dimensionality reduction, championed by Mehreen Saeed (Saeed, 2009) who used a Bernoulli mixture model as an input to an artificial neural network. In his paper on data grid models Marc Boullé (2009) proposes a new method of *data discretization*. It can be used directly as part of a learning machine based on data grids (stepwise constant predictors) or as a preprocessing to other learning machines, such as the Naïve Bayes classifier. Of particular interest in this paper is the use of *data dependent priors*.

Designing kernels and model architectures. Special purpose neural network architectures implementing the idea of "weight sharing" such as Time Delay Neural Networks (Waibel, 1988) or two-dimensional convolutional networks (LeCun et al., 1989) have proved to be very effective in speech and image processing. More recently a wide variety of special purpose kernels have been proposed to incorporate domain knowledge in kernel learning algorithms. Examples include kernels invariant under various transforms (Simard et al., 1993; Pozdnoukhov and Bengio, 2006), string matching kernels (Watkins, 2000), and other sequence and tree kernels (Vishwanathan and Smola, 2003). Along these lines, in our collection of papers, Chloé Agathe Azencott and Pierre Baldi have proposed two-dimensional kernels for high-thoughput screening (Azencott and Baldi, 2009). Design effort has also be put into general purpose

kernels. For instance, in the paper of Adankon and Cheriet (2009) , the SVM regularization hyper-parameter C (box-constraint) is incorporated in the kernel function. This facilitates the task of multi-level inference algorithms.

Defining regularizers or priors. Designing priors $P(f)$ or **regularizers** $\Omega[f]$ or structuring parameter space into parameters and several levels of hyper-parameters can also be thought of as a filter method. Most priors commonly used do not embed domain knowledge, they just enforce Ockham's razor by favoring simple (smooth) functions or eliminating irrelevant features. Priors are also often chosen out of convenience to facilitate the closed-form calculation of Bayesian integrals (for instance the use of so-called "conjugate priors", see *e.g.,* Neal and Zhang, 2006). The 2-norm regularizer $\Omega[f] = \|f\|^2_{\mathcal{H}}$ for kernel ridge regression, Support Vector Machines (SVM) and Least-Square Support Vector Machines (LSSVM) have been applied with success by many top-ranking participants of the challenges we organized. Gavin Cawley was co-winner of the WCCI 2006 performance prediction challenge using LSSVMs (Cawley, 2006). Another very successful regularizer is the **Automatic Relevance Determination** (ARD) prior. This regularizer was used in the winning entry of Radford Neal in the NIPS 2003 feature selection challenge (Neal and Zhang, 2006). Gavin Cawley also made top ranking reference entries in the IJCNN 2007 ALvsPK challenge (Cawley and Talbot, 2007b) using a similar ARD prior. For linear models, the 1-norm regularizer $\|\mathbf{w}\|$ is also popular (see *e.g.,* Pranckeviciene and Somorjai, 2009), but this has not been quite as successful in challenges as the 2-norm regularizer or the ARD prior.

Noise modeling. While the prior (or the regularizer) embeds our prior or domain knowledge of the model class, the likelihood (or the loss function) embeds our prior knowledge of the noise model on the predicted variable y. In regression, the square loss corresponds to Gaussian noise model, but other choices are possible. For instance, recently, Gavin Cawley and Nicola Talbot implemented Poisson regression for kernel machines (Cawley et al., 2007). For classification, the many loss functions proposed do not necessarily correspond to a noise model, they are often just bounding the 0/1 loss and are used for computational convenience. In the Bayesian framework, an sigmoidal function is often used (like the logistic or probit functions) to map the output of a discriminant function $f(\mathbf{x}_k)$ to probabilities p_k. Assuming target values $y_k \in \{0, 1\}$, the likelihood $\Pi_k p_k^{y_k}(1-p_k)^{1-y_k}$ corresponds to the cross-entropy cost function $\sum_k y_k \ln p_k + (1-y_k)\ln(1-p_k)$. A clever piece-wise S-shaped function, flat on the asymptotes, was used in (Chu et al., 2006) to implement sparsity for a Bayesian SVM algorithm. Noise modeling is not limited to noise models for the target y, it also concerns modeling noise on the input variables $\mathbf{x}$. Many authors have incorporated noise models on $\mathbf{x}$ as part of the kernel design, *e.g.,* by enforcing invariance (Simard et al., 1993; Pozdnoukhov and Bengio, 2006). A simple but effective means of using a noise model is to generate additional training data by distorting given training examples. Additional "unsupervised" data is often useful to fit a noise model on the input variables $\mathbf{x}$. Repeatability and reproducibility ($R\&R$) studies may also provide data to fit a noise model.

Feature selection filters. Feature selection, as a filter method, allows us to reduce the dimensionality of the feature space, to ease the computations performed by learning machines. This is often a necessary step for computationally expensive algorithms such as neural networks. Radford Neal for instance, used filters based on univariate statistical tests to prune the feature space before applying his Bayesian neural network algorithm (Neal and Zhang, 2006). Univariate filters were also widely used in the KDD cup 2009, which involved classification tasks on a very large database, to cut down computations (Guyon et al., 2009b). Feature selection filters are not limited to univariate filters. Markov blanket methods, for instance, provide powerful feature selection filters (Aliferis et al., 2003). A review of filters for feature selection can be found in (Guyon et al., 2006a, Chapter 3).

5.2 Wrappers

Wrapper methods consider learning machines as *black boxes* capable of internally adjusting their parameters $\boldsymbol{\alpha}$ given some data D and some hyper-parameter values $\boldsymbol{\theta}$. No knowledge either of the architecture, of the learning machines, or of their learning algorithm should be required to use a wrapper. Wrappers are applicable to selecting a classifier from amongst a finite set of learning machines ($\boldsymbol{\theta}$ is then a discrete index), or an infinite set (for continuous values of $\boldsymbol{\theta}$). Wrappers can also be used to build ensembles of learning machines, including Bayesian ensembles. Wrappers use a *search algorithm* or a *sampling algorithm* to explore hyper-parameter space and an *evaluation function* (a risk functional $R_D[f(\boldsymbol{\theta})]$, a posterior probability $P(f(\boldsymbol{\theta})|D)$, or any model selection index $r[f(\boldsymbol{\theta})]$) to assess the performance of the sample of trained learning machines , and, either select one single best machine or create an ensemble of machine voting to make predictions.

Search and sampling algorithms. Because the learning machines in the wrapper setting are "black boxes", we cannot sample directly from the posterior distribution $P(f(\boldsymbol{\theta})|D)$ (or according to $\exp -R_D[f(\boldsymbol{\theta})]$ or $\exp -r[f(\boldsymbol{\theta})]$). We can only compute the evaluation function for given values of $\boldsymbol{\theta}$ for which we run the learning algorithm of $f(\boldsymbol{\theta})$, which internally adjusts its parameters $\boldsymbol{\alpha}$. A **search strategy** defines which hyper-parameter values will be considered and in which order (in case a halting criterion ends the search prematurely). Gavin Cawley, in his challenge winning entries, used the Nelder-Mead simplex algorithm (Cawley and Talbot, 2007a). **Monte-Carlo Markov Chain MCMC methods** are used in Bayesian modeling to sample the posterior probability and have given good results in challenges (Neal and Zhang, 2006). The resulting ensemble is a simple average of the sampled functions $F(\mathbf{x}) = (1/s) \sum_{i=1}^{s} f(\mathbf{x}|\boldsymbol{\theta}_k)$. Wrappers for *feature selection* use all sort of techniques, but sequential *forward selection* or *backward elimination* methods are most popular (Guyon et al., 2006a, Chapter 4). Other stochastic search methods include biologically inspired methods such as *genetic algorithms* and *particle swarm optimization*. Good results have been obtained with this last method in challenges (H. J. Escalante, 2009), showing that extensive search does not necessarily yield over-fit solutions, if some regularization mechanism is used. The authors of that paper rely for that purpose on weight decay and early stopping. Frequentist ensemble methods, including

Random Forests (Breiman, 2001) and Logitboost (Friedman et al., 2000) also gave good results in challenges (Lutz, 2006; Tuv et al., 2009; Dahinden, 2009).

Evaluation functions. For Bayesian approaches, the standard evaluation function is the "evidence", that is the marginal likelihood (also called type-II likelihood)(Neal and Zhang, 2006), or, in other words, the likelihood at the second level of inference. For frequentist approaches, the most frequently used evaluation function is the cross-validation estimator. Specifically, K-fold cross-validation is most often used (H. J. Escalante, 2009; Dahinden, 2009; Lutz, 2006; Reunanen, 2007). The values $K = 10$ or $K = 5$ are typically used by practitioners regardless of the difficulty of the problem (error rate, number of examples, number of variables). Computational considerations motivate this choice, but the authors report a relative insensitivity of the result in that range of values of K. The leave-one-out (LOO) estimator is also used, but due to its high variance, it should rather be avoided, except for computational reasons (see in Section 5.3 cases in which the LOO error is inexpensive to compute). These estimators may be poor predictors of the actual learning machine performances, but they are decent model selection indices, provided that the same data splits are used to compute the evaluation function for all models. For **bagging** methods (like Random Forests, Breiman, 2001), the bootstrap estimator is a natural choice: the "out-of-bag" samples, which are those samples not used for training, are used to predict performance. Using empirical estimators at the second level on inference poses the problem of possibly over-fitting them. Some authors advocate using evaluation functions based on prediction risk bounds: (Koo and Kil, 2008) and (Claeskens et al., 2008) derive in this way information criteria for regression models (respectively called "modulus of continuity information criterion" or MCIC and "kernel regression information criterion" or KRIC) and (Claeskens et al., 2008) and (Pranckeviciene and Somorjai, 2009) propose information criteria for classification problems (respectively called "support vector machine information criterion" SVMIC and "transvariation intensity"). The effectiveness of these new criteria is compared empirically in the papers to the classical "Akaike information criterion" or AIC (Akaike, 1973) and the "Bayesian information criterion" or BIC (Schwarz, 1978).

5.3 Embedded methods

Embedded methods are similar to wrappers. They need an evaluation function and a search strategy to explore hyper-parameter space. But, unlike wrapper methods, they exploit specific features of the learning machine architecture and/or learning algorithm to perform multi-level inference. It is easy to appreciate that knowledge of the nature and structure of a learning machine can allow us to search hyper-parameter space in a more efficient way. For instance, the function $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$ may be differentiable with respect to hyper-parameters $\boldsymbol{\theta}$ and it may be possible to use *gradient descent* to optimize an evaluation function $r[f]$. Embedded methods have been attracting substantial attention within the machine learning community in the past few years because of the mathematical elegance of some of the new proposed methods.

Bayesian embedded methods. In the Bayesian framework, the embedded search, sampling or summation over parameters and hyper-parameters is handled in an elegant and consistent way by defining priors both for parameters and hyper-parameters, and computing the posterior, perhaps in two steps, as indicated in Equation 4. Of course, it is more easily said than done and the art is to find methods to carry out this integration, particularly when it is analytically intractable. Variational methods are often used to tackle that problem. Variational methods convert a complex problem into a simpler problem, but the simplification introduces additional "variational" parameters, which must then be optimized, hence introducing another level of inference. Typically, the posterior is bounded from above by a family of functions parameterized by given variational parameters. Optimizing the variational parameters yields the best approximation of the posterior (see *e.g.,* Seeger, 2008). Bayesian pragmatists optimize the evidence (also called type-II likelihood or marginal likelihood) at the second level of inference, but non-purists sometimes have a last recourse to cross-validation. The contributions of (Boullé, 2007, 2009) stand out in that respect because they propose model selection methods for classification and regression, which have no last recourse to cross-validation, yet performed well in recent benchmarks (Guyon et al., 2008a, 2009b). Such methods have been recently extended to the less studied problem of rank regression (Hue and Boullé, 2007). The methods used are Bayesian in spirit, but make use of original data-dependent priors.

Regularized functionals. In the frequentist framework, the choice of a prior is replaced by the choice of a regularized functional. Those are two-part evaluation functions including the empirical risk (or the negative log-likelihood) and a regularizer (or a prior). For kernel methods, a 2-norm regularizer is often used, yielding the classical penalized functional $R_{reg}[f] = R_{emp}[f] + \gamma \|f\|^2_{\mathcal{F}}$. (Pranckeviciene and Somorjai, 2009) explore the possibilities offered by a 1-norm regularizer. Such approaches provide an embedded method of feature selection, since the constraints thus imposed on the weight vector drive some weights to exactly zero. We emphasized in the introduction that, in some cases, decomposing the inference problem into multiple levels allows us to conveniently regain the convexity of the optimization problem involved in learning. Ye et al. (2008) propose a multiple kernel learning (MKL) method, in which the optimal kernel matrix is obtained as a linear combination of pre-specified kernel matrices, which can be brought back to a convex program. Few approaches are fully embedded and a wrapper is often used at the last level of inference. For instance, in kernel methods, the kernel parameters may be optimized by gradient descent on the regularized functional, but then the regularization parameter is selected by cross-validation. One approach is to use a bound on the generalization error at the second level of inference. For instance, Guermeur (2007) proposes such a bound for the multi-class SVM, which can be used to choose the values of the "soft margin parameter" C and the kernel parameters. Cross-validation may be preferred by practitioners because it has performed consistently well in benchmarks (Guyon et al., 2006b). This motivated (Kunapuli et al., 2009) to integrate the search for optimal parameters and hyper-parameters into a multi-level optimization program, using a regularized functional at the lower level, and cross-validation at the upper level. An-

other way of integrating a second level of inference performed by cross-validation and the optimization of a regularized functional at the first level of inference is to use a closed-form expression of the leave-one-out error (or a bound) and optimize it by gradient descent or another classical optimization algorithm. Such *virtual leave-one-out* estimators, requiring training a single classifier on all the data (see *e.g.,* Cawley and Talbot, 2007a; Debruyne et al., 2–8, in the collection of papers we have assembled)).

6. Advanced topics and open problems

We have left aside many important aspects of model selection, which, space permitting, would deserve a longer treatment. We briefly discuss them in this section.

6.1 Ensemble methods

In Section 4, we have made an argument in favor of unifying *model selection* and *ensemble methods*, stemming either from a Bayesian or frequentist perspective, in the common framework of *multi-level optimization*. In Sections 5.1, 5.2, and 5.3, we have given examples of model selection and ensemble methods following *filter*, *wrapper* or *embedded* strategies. While this categorization has the advantage of erasing the dogmatic origins of algorithms, it blurs some of the important differences between model selection and ensemble methods. Ensemble methods can be thought of as a way of circumventing model selection by voting among models rather than choosing a single model. Recent challenges results have proved their effectiveness (Guyon et al., 2009b). Arguably, model selection algorithms will remain important in applications where model simplicity and data understanding prevail, but ever increasing computer power has brought ensemble methods to the forefront of multi-level inference techniques. For that reason, we would like to single out those papers of our collection that have proposed or applied ensemble methods:

Lutz (2006) used boosted shallow decision trees for his winning entries in two consecutive challenges. Boosted decision trees have often ended up among the top ranking methods in other challenges (Guyon et al., 2006a, 2009b). The particular implementation of Lutz of the Logitboost algorithm (Friedman et al., 2000) use a "shrinkage" regularization hyper-parameter, which seems to be key to attain good performance, and is adjusted by cross-validation as well as the total number of *base learners*. Dahinden (2009) successfully applied the Random Forest (RF) algorithm (Breiman, 2001) in the performance prediction challenge (Guyon et al., 2006b). She demonstrated that with minor adaptations (adjustment of the bias value for improved handling of unbalanced classes), the RF algorithm can be applied without requiring user intervention. RF continues to be a popular and successful method in challenges (Guyon et al., 2009b). The top ranking models use very large ensembles of hundreds of trees. One of the unique features of RF algorithms is that they subsample both the training examples and the features to build base learners. Using random subsets of features seems to be a winning strategy, which was applied by others to ensembles of trees using both boosting and bagging (Tuv et al., 2009) and to other base learners (Nikulin, 2009). Boullé (2007) also adopts the idea of creating ensembles using base learners constructed with different subsets of features. Their base learner is the naïve Bayes classifier and, instead of using random subsets, they select subsets with a forward-backward method, using a maximum A Posteriori (MAP) evaluation function (hence not requiring

cross-validation). The base learners are then combined with an weighting scheme based on an information theoretic criterion, instead on weighting the models with the posterior probability as in Bayesian model averaging. This basically boils down to using the logarithm of the posterior probabilities instead of the posterior probabilities themselves for weighting the models. The weights have an interpretation in terms of model compressibility. The authors show that this strategy outperforms Bayesian model averaging on several benchmark datasets. This can be understood by the observation that when the posterior distribution is sharply peaked around the posterior mode, averaging is almost the same as selecting the MAP model. Robustness is introduced by performing a more balanced weighting of the base learners. In contrast with the methods we just mentioned, which choose identical base learners (trees of naïve Bayes), other successful challenge participants have built heterogeneous ensembles of learning machines (including *e.g.,* linear models, kernel methods, trees, naïve Bayes, and neural networks), using cross-validation to evaluate their candidates for inclusion in the ensemble (Wichard, 2007; IBM team, 2009). While Wichard (2007) evaluates classifiers independently, IBM team (2009) uses a forward selection method, adding a new candidate in the ensemble based on the new performance of the ensemble.

6.2 PAC Bayes approaches

Unifying Bayesian and frequentist model selection procedures under the umbrella of *multi-level inference* may shed new light on correspondences between methods and have a practical impact on the design of toolboxes incorporating model selection algorithms. But there are yet more synergies to be exploited between the Bayesian and the frequentist framework. In this section, we would like to capture the spirit of the PAC Bayes approach and outline possible fruitful directions of research.

The PAC learning framework (Probably Approximately Correct), introduced by Valiant (1984) and later recognized to closely resemble the approach of the Russian school popularized in the US by Vapnik (1979), has become the beacon of frequentist learning theoretic approaches. It quantifies the generalization performance (the *Correct* aspect) of a learning machine via performance bounds (the *Approximate* aspect) holding in probability (the *Probable* aspect):

$$Prob\Big[(R[f] - R_{emp}[f]) \leq \epsilon(\delta)\Big] \geq (1-\delta) \ , \tag{7}$$

In this equation, the confidence interval $\epsilon(\delta)$ (*Approximate* aspect) bounds, with probability $(1-\delta)$ (*Probable* aspect),the difference between the *expected risk* or generalization error $R[f]$ and the *empirical risk*[6] $R_{emp}[f]$ (*Correct* aspect). Recently, many bounds have been proposed to quantify the generalization performance of algorithms (see *e.g.,* Langford, 2005, for a review). The idea of deriving new algorithms, which optimize a bound $\epsilon(\delta)$ (*guaranteed risk* optimization) has been popularized by the success of SVMs (Boser et al., 1992) and boosting (Freund and Schapire, 1996).

The PAC framework is rooted in the frequentist philosophy of defining probability in terms of *frequencies of occurrence of events* and bounding differences between mathematical expectations and frequencies of events, which vanish with increasingly large sample sizes

6. at the first level of inference, this would be the training error $R_{tr}[f]$; at the second level of inference this may be the validation error $R_{va}[f]$

(law of large numbers). Yet, since the pioneering work of Haussler et al. (1994), many authors have proposed so-called PAC-Bayes bounds. Such bounds assess the performance of existing Bayesian algorithms (see *e.g.,* Seeger, 2003), or are used to derive new Bayesian algorithms optimizing a guaranteed risk functional (see Germain et al. (2009) and references therin).

This is an important paradigm shift, which bridges the gap between the frequentist *structural risk minimization* approach to model selection (Vapnik, 1998) and the *Bayesian prior* approach. It erases the need for assuming that *the model used to fit the data comes from a concept space of functions that generated the data*. Instead, priors may be used to provide a "structure" on a chosen model space (called *hypothesis space* to distinguish it from the *concept space*), which does not necessarily coincide with the *concept space*, of which we often know nothing. Reciprocally, we can interpret structures imposed on a hypothesis space as our prior belief that certain models are going to perform better than others (see for instance the examples at the end of Section 3.1).

This opens the door to also regularizing the second level of inference by using performance bounds on the cross-validation error, as was done for instance in (Cawley and Talbot, 2007a; Guyon, 2009).

6.3 Open problems

- **Domain knowledge:** From the earliest embodiments of Okcham's razor using the number of free parameters to modern techniques of regularization and bi-level optimization, model selection has come a long way. The problem of finding the right structure remains, the rights prior or the right regularizer. Hence know-how and domain knowledge are still required. But in a recent challenge we organized called "agnostic learning *vs.* prior knowledge" (Guyon et al., 2008b) it appeared that the relatively small incremental improvements gained with prior knowledge came at the expense of important human effort. In many domains, collecting more data is less costly than hiring a domain expert. Hence there is pressure towards improving machine learning toolboxes and, in particular equipping them with model selection tools. For the competitions we organized (Clopinet, 2004-2009), we made a toolbox available with state-of-the-art models (Saffari and Guyon, 2006), which we progressively augmented with the best performing methods. The Particle Swarm Optimization (PSO) model selection method can find the best models in the toolbox and reproduce the results of the challenges (H. J. Escalante, 2009). Much remains to be done to incorporate filter and wrapper model selection algorithms in machine learning toolboxes.

- **Unsupervised learning:** Multi-level optimization and model selection are also central problems for **unsupervised learning**. When no target variable is available as "teaching signal" one can still define regularized risk functionals and multi-level optimization problems (Smola et al., 2001). Hyper-parameters (*e.g.,* "number of clusters") can be adjusted by optimizing a second level objective such as model stability (Ben-Hur et al., 2002), which is an erzatz of cross-validation. The primary difficulty with model selection for unsupervised learning is to validate the selected model. To this day, there is no consensus on how to benchmark methods, hence it is very difficult to

quantify progress in this field. This is why we have so far shied away from evaluating unsupervised learning algorithms, but this remains on our agenda.

- **Semi-supervised learning:** Very little has been done for model selection in **semi-supervised learning** problems, in which only some training instances come with target values. Semi-supervised tasks can be challenging for traditional model selection methods, such as cross-validation, because the number of labeled data is often very small. Schuurmans and Southey (2001) used the unlabeled data to test the consistency of a model, by defining a metric over the hypothesis space. Similarly, Madani et al. (2005) introduced the co-validation method, which uses the disagreement of various models on the predictions over the unlabeled data as a model selection tool. In some cases there is no performance gain by using the unlabeled data for training (Singh et al., 2008). Deciding whether all or part of the unlabeled data should be used for training (*data selection*) may also be considered a model selection problem.

- **Non *i.i.d.* data:** The problem of non *i.i.d.* data raises a number of other questions because if there are significant differences between the distribution of the training and the test data, the cross-validation estimator may be worthless. For instance, in causal discovery problems, training data come from a "natural" distribution while test data come from a different "manipulated" distribution (resulting from some manipulations of the system by an external agent, like clamping a given variable to given values). Several causal graphs may be consistent with the "natural distribution" (not just with the training data, with the true unknown distribution), but yield very different predictions of manipulated data. Rather selecting a single model, it make more sense to select a model class. We have started a program of data exchange and benchmarks to evaluate solutions to such problems (Guyon et al., 2008a, 2009a).

- **Computational considerations:** The selection of the model best suited to a given application is a multi-dimensional problem in which prediction performance is only one of the dimensions. Speed of model building and processing efficiency of deployed models are also important considerations. Model selection algorithms (or ensemble methods) which often require many models to be trained (*e.g.,* wrapper methods with extensive search strategies and using cross-validation to validate models) may be unable to build solutions in a timely manner. At the expense of some acceptable loss in prediction performance, methods using *greedy search strategies* (like forward selection methods) and *single-pass evaluation functions* (requiring the training of only a single model to evaluate a given hyper-parameter choice), may considerably cut the training time. Greedy search methods include forward selection and backward elimination methods. Single-pass evaluation functions include penalized training error functionals (regularized functionals, MAP estimates) and virtual-leave-one-out estimators. The latter allows users to compute the leave-one-out-error at almost no additional computational expense than training a single predictor on all the training data (see *e.g.,* Guyon et al., 2006a, Chapter 2, for a review). Other tricks-of-the-trade include following *regularization paths* to sample the hyper-parameter space more effectively (Rosset and Zhu, 2006; Hastie et al., 2004). For some models, the evaluation function is piecewise linear between a few discontinuous changes occurring for a few finite hyper-

parameter values. The whole path can be reconstructed from only the values of the evaluation function at those given points. Finally, Reunanen (2007) proposed clever ways of organizing nested cross-validation evaluations in multiple level of inference model selection using cross-indexing. The author also explored the idea of spending more time to refine the evaluation of the most promising models. Further work needs to be put into model selection methods, which simultaneously address multiple objectives, including optimizing prediction performance and computational cost.

7. Conclusion

In the past twenty years, much effort has been expended towards finding the best regularized functionals. The many embodiments of Ockham's razor in machine learning have converged towards similar regularizers. Yet, the problem of model selection remains because we need to optimize the regularization parameter(s) and often we need to select among various preprocessings, learning machines, and post-processings. In the proceedings of three of the challenges we organized around the problem of model selection, we have collected a large number of papers, which testify of the vivid activity of the field. Several researchers do not hesitate to propose heretic approaches transcending the usual "frequentist" or Bayesian dogma. We have seen the idea of using the Bayesian machinery to design regularizers with "data-dependent priors" emerge (Boullé, 2007, 2009), much like a few years ago data-dependent performance bounds (Bartlett, 1997; Vapnik, 1998) and PAC-Bayes bounds (Haussler et al., 1994; Seeger, 2003) revolutionized the "frequentist" camp, up to then very fond of uniform convergence bounds and the VC-dimension (Vapnik and Chervonenkis, 1971). We have also seen the introduction of regularization of cross-validation estimators using Bayesian priors (Cawley and Talbot, 2007a). Ensemble methods may be thought of as a way of circumventing model selection. Rather, we think of model selection and ensemble methods as two options to perform multi-level inference, which can be formalized in a unified way.

Within this general framework, we have categorized approaches into *filter*, *wrapper* and *embedded* methods. These methods complement each other and we hope that in a not too distant future, they will be integrated into a consistent methodology: Filters first can prune model space; Wrappers can perform an outer level model selection to select pre/post processings and feature subsets; Embedded methods can perform an inner level hyperparameter selection integrated within a bi-level optimization program. We conclude that we are moving towards a unified framework for model selection and there is a beneficial synergy between methods, both from a theoretical and from a practical perspective.

Acknowledgments

This project was supported by the National Science Foundation under Grant N0. ECS-0424142. Amir Saffari was supported by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

M. Adankon and M. Cheriet. Unified framework for SVM model selection. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and F. Csaki, editors, *2nd International Symposium on Information Theory*, pages 267–281. Akademia Kiado, Budapest, 1973.

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON, a novel Markov blanket algorithm for optimal variable selection. In *2003 American Medical Informatics Association (AMIA) Annual Symposium*, pages 21–25, 2003.

C.-A. Azencott and P. Baldi. Virtual high-throughput screening with two-dimensional kernels. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

P. L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 134, Cambridge, MA, 1997. MIT Press.

A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*, pages 6–17, 2002.

A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, December 1997.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.

M. Boullé. Compression-based averaging of selective naive bayes classifiers. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 8, pages 1659–1685, Jul 2007. URL `http://www.jmlr.org/papers/volume8/boulle07a/boulle07a.pdf`.

M. Boullé. Data grid models for preparation and modeling in supervised learning. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

G. Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *IJCNN*, pages 1661–1668, 2006.

G. Cawley and N. Talbot. Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 8, pages 841–861, Apr 2007a. URL `http://www.jmlr.org/papers/volume8/cawley07a/cawley07a.pdf`.

G. Cawley and N. Talbot. Over-fitting in model selection and subsequent selection bias in performance evaluation. *JMLR, submitted*, 2009.

G. C. Cawley and N. L. C. Talbot. Agnostic learning versus prior knowledge in the design of kernel machines. In *Proc. IJCNN07*, Orlando, Florida, Aug 2007b. INNS/IEEE.

G.C. Cawley, G.J. Janacek, and N.L.C. Talbot. Generalised kernel machines. In *International Joint Conference on Neural Networks*, pages 1720 – 1725. IEEE, August 2007.

W. Chu, S. Keerthi, C. J. Ong, and Z. Ghahramani. Bayesian Support Vector Machines for feature ranking and selection. In I. Guyon, et al., editor, *Feature Extraction, Foundations and Applications*, 2006.

G. Claeskens, C. Croux, and J. Van Kerckhoven. An information criterion for variable selection in Support Vector Machines. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 9, pages 541–558, Mar 2008. URL `http://www.jmlr.org/papers/volume9/claeskens08a/claeskens08a.pdf`.

Clopinet. Challenges in machine learning, 2004-2009. URL `http://clopinet.com/challenges`.

Corinne Dahinden. An improved Random Forests approach with application to the performance prediction challenge datasets. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

M. Debruyne, M. Hubert, and J. Suykens. Model selection in kernel based regression using the influence function. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 9, pages 2377–2400, Oct 2–8. URL `http://www.jmlr.org/papers/volume9/debruyne08a/debruyne08a.pdf`.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression, a statistical view of boosting. *Annals of Statistics*, 28:337374, 2000.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software (to appear)*, 2009.

P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. PAC-Bayesian learning of linear classifiers. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 353–360, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1.

Y. Guermeur. VC theory of large margin multi-category classifiers. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 8, pages 2551–2594, Nov 2007. URL `http://www.jmlr.org/papers/volume8/guermeur07a/guermeur07a.pdf`.

I. Guyon. A practical guide to model selection. In J. Marie, editor, *Machine Learning Summer School*. Springer, to appear, 2009.

I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov. Design and analysis of the causation and prediction challenge. In *JMLR W&CP*, volume 3, pages 1–33, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008a. URL `http://jmlr.csail.mit.edu/papers/topic/causality.html`.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Editors. *Feature Extraction, Foundations and Applications*. Studies in Fuzziness and Soft Computing. With data, results and sample code for the NIPS 2003 feature selection challenge. Physica-Verlag, Springer, 2006a. URL `http://clopinet.com/fextract-book/`.

I. Guyon, D. Janzing, and B. Schölkopf. Causality: objectives and assessment. In *NIPS 2008 workshop on causality*, volume 7. JMLR W&CP, in press, 2009a.

I. Guyon, V. Lemaire, M. Boullé, Gideon Dror, and David Vogel. Analysis of the KDD cup 2009: Fast scoring on a large orange customer database. In *KDD cup 2009, in press*, volume 8. JMLR W&CP, 2009b.

I. Guyon, A. Saffari, G. Dror, and J. Buhmann. Performance prediction challenge. In *IEEE/INNS conference IJCNN 2006*, Vancouver, Canada, July 16-21 2006b.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Agnostic learning vs. prior knowledge challenge. In *IEEE/INNS conference IJCNN 2007*, Orlando, Florida, August 12-17 2007.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Analysis of the IJCNN 2007 agnostic learning vs. prior knowledge challenge. In *Neural Networks*, volume 21, pages 544–550, Orlando, Florida, March 2008b.

L. E. Sucar H. J. Escalante, M. Montes. Particle swarm model selection. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 10, pages 405–440, Feb 2009. URL `http://www.jmlr.org/papers/volume10/escalante09a/escalante09a.pdf`.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *JMLR*, 5:1391–1415, 2004. URL `http://jmlr.csail.mit.edu/papers/volume5/hastie04a/hastie04a.pdf`.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction*. Springer Verlag, 2000.

D. Haussler, M. Kearns, and R. Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the vc dimension. *Machine Learning*, 14(1):83–113, 1994. ISSN 0885-6125.

A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.

C. Hue and M. Boullé. A new probabilistic approach in rank regression with optimal Bayesian partitioning. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 8, pages 2727–2754, Dec 2007. URL `http://www.jmlr.org/papers/volume8/hue07a/hue07a.pdf`.

IBM team. Winning the KDD cup orange challenge with ensemble selection. In *KDD cup 2009, in press*, volume 8. JMLR W&CP, 2009.

R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97(1-2): 273–324, December 1997.

I. Koo and R. M. Kil. Model selection for regression with continuous kernel functions using the modulus of continuity. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 9, pages 2607–2633, Nov 2008. URL `http://www.jmlr.org/papers/volume9/koo08b/koo08b.pdf`.

G. Kunapuli, J.-S. Pang, and K. Bennett. Bilevel cross-validation-based model selection. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

John Langford. Tutorial on practical prediction theory for classification. *JMLR*, 6:273–306, Mar 2005. URL `http://jmlr.csail.mit.edu/papers/volume6/langford05a/langford05a.pdf`.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.

R. W. Lutz. Logitboost with trees applied to the WCCI 2006 performance prediction challenge datasets. In *Proc. IJCNN06*, pages 2966–2969, Vancouver, Canada, July 2006. INNS/IEEE.

D. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

O. Madani, D. M. Pennock, and G. W. Flake. Co-validation: Using model disagreement to validate classification algorithms. In *NIPS*, 2005.

M. Momma and K. Bennett. A pattern search method for model selection of Support Vector Regression. In *In Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2002.

R. Neal and J. Zhang. High dimensional classification with Bayesian neural networks and dirichlet diffusion trees. In I. Guyon, et al., editor, *Feature Extraction, Foundations and Applications*, 2006.

Vladimir Nikulin. Classification with random sets, boosting and distance-based clustering. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945):978 – 982, February 1990.

Alexei Pozdnoukhov and Samy Bengio. Invariances in kernel methods: From samples to objects. *Pattern Recogn. Lett.*, 27(10):1087–1097, 2006. ISSN 0167-8655.

E. Pranckeviciene and R. Somorjai. Liknon feature selection: Behind the scenes. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

J. Reunanen. Model selection and assessment using cross-indexing. In *Proc. IJCNN07*, Orlando, Florida, Aug 2007. INNS/IEEE.

S. Rosset and J. Zhu. Sparse, flexible and efficient modeling using L1 regularization. In I. Guyon, et al., editor, *Feature Extraction, Foundations and Applications*, 2006.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

M. Saeed. Hybrid learning using mixture models and artificial neural networks. In I. Guyon, et al., editor, *Hands on Pattern Recognition*. Microtome, 2009.

A. Saffari and I. Guyon. Quick start guide for CLOP. Technical report, Graz University of Technology and Clopinet, May 2006. URL `http://clopinet.com/CLOP/`.

D. Schuurmans and F. Southey. Metric-based methods for adaptive model selection and regularization. *Machine Learning, Special Issue on New Methods for Model Selection and Model Combination*, 48:51–84, 2001.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

M. Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *JMLR*, 3:233–269, 2003. URL `http://jmlr.csail.mit.edu/papers/volume3/seeger02a/seeger02a.pdf`.

M. Seeger. Bayesian inference and optimal design for the sparse linear model. *JMLR*, 9: 759–813, 2008. ISSN 1533-7928.

P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufmann.

Aarti Singh, Robert Nowak, and Xiaojin Zhu. Unlabeled data: Now it helps, now it doesn't. In *NIPS*, 2008.

A. Smola, S. Mika, B. Schölkopf, and R. Williamson. Regularized principal manifolds. *JMLR*, 1:179–209, 2001. URL `http://jmlr.csail.mit.edu/papers/volume1/smola01a/smola01a.pdf`.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

E. Tuv, A. Borisov, G. Runger, and K. Torkkola. Feature selection with ensembles, artificial variables, and redundancy elimination. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 10, pages 1341–1366, Jul 2009. URL `http://www.jmlr.org/papers/volume10/tuv09a/tuv09a.pdf`.

L. Valiant. A theory of the learnable. *Communications of the ACM,*, 27(11):1134–1142, 1984.

V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, N.Y., 1998.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–180, 1971.

S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In *Advances in Neural Information Processing Systems 15*, pages 569–576. MIT Press, 2003. URL `http://books.nips.cc/papers/files/nips15/AA11.pdf`.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

A. Waibel. Consonant recognition by modular construction of large phonemic time-delay neural networks. In *NIPS*, pages 215–223, 1988.

C. Watkins. Dynamic alignment kernels. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press. URL `http://www.cs.rhul.ac.uk/home/chrisw/dynk.ps.gz`.

P. Werbos. Backpropagation: Past and future. In *International Conference on Neural Networks*, pages 343–353. IEEE, IEEE press, 1988.

J. Weston, A. Elisseff, B. Schoelkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *JMLR*, 3:1439–1461, 2003.

J. Wichard. Agnostic learning with ensembles of classifiers. In *Proc. IJCNN07*, Orlando, Florida, Aug 2007. INNS/IEEE.

J. Ye, S. Ji, and J. Chen. Multi-class discriminant kernel learning via convex programming. In I. Guyon and A. Saffari, editors, *JMLR, Special topic on model selection*, volume 9, pages 719–758, Apr 2008. URL `http://www.jmlr.org/papers/volume9/ye08b/ye08b.pdf`.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In *NIPS*, 2003.

Appendix A. Glossary

Automatic Relevance Determination (ARD) prior. The ARD prior was invented for neural networks (MacKay, 1992): all network input variables and all neuron outputs (internal features) are weighed by a scaling factor κ_i, before being independently weighted by the network connections. A hyper-prior must be chosen to favor small values of the κ_i, which makes the influence of irrelevant variables or features naturally fade away. For kernel methods, ARD falls under the same framework as the $\|f\|^2_{\mathcal{H}}$ regularizer, for a special class of kernels using variable (feature) scaling factors. For instance, the ARD prior is implemented by defining the Gaussian kernel (for positive hyper-parameters κ_i):

$$K(\mathbf{x}_h, \mathbf{x}_k) = \exp\left\{-\sum_{j=1}^{n} \kappa_i (x_{h,j} - x_{k,j})^2\right\} \tag{8}$$

instead of the regular Gaussian kernel $K(\mathbf{x}_h, \mathbf{x}_k) = \exp\left\{-\kappa \|\mathbf{x}_h - \mathbf{x}_k\|^2\right\}$.

Base learner. In an ensemble method, the individual learning machines that are part of the ensemble.

Bagging. Bagging stands for bootstrap aggregating. Bagging is a parallel ensemble method (all **base learners** are built independently from training data subsets). Several data subsets of size m are drawn independently with replacement from the training set of m examples. On average each subset thus built contains approximately 2/3 of the training examples. The ensemble predictions are made by averaging the predictions of the baser learners. The ensemble approximates $E_D(f(\mathbf{x}, D))$, where $f(\mathbf{x}, D)$ is a function from the model class F, trained with m examples and $E_D(.)$ is the mathematical expectation over all training sets of size m. The rationale comes from the **bias/variance decomposition** of the **generalization error**. The "out-of-bag" samples (samples not used for learning for each data subset drawn for training) may be used to create a bootstrap prediction of performance.

Bayesian learning. Under the Bayesian framework, it is assumed that the data were generated from a double random process: (1) a model is first drawn according to a **prior** distribution in a **concept space**; (2) data are produced using the model. In the particular case of supervised learning, as for **maximum likelihood learning**, a three-part data generative model is assumed: $P(\mathbf{x})$, $f \in \mathcal{F}$, and a zero-mean noise model. But, it is also assumed that the function f was drawn according to a prior distribution $P(f)$. This allows us to compute the probability of an output y given an input $\mathbf{x}$, $P(y|\mathbf{x}) = \int_{f\in\mathcal{F}} P(y|\mathbf{x}, f) dP(f)$, or its mathematical expectation $E(y|\mathbf{x}) = \int_{f\in\mathcal{F}} f(\mathbf{x}) dP(f)$, averaging out the noise. After training data D are observed, the prior $P(f)$ is replaced by the **posterior** $P(f|D)$. The mathematical expectation of y given $\mathbf{x}$ is estimated as: $E(y|\mathbf{x}, D) = \int_{f\in\mathcal{F}} f(\mathbf{x}) dP(f|D)$. Hence, learning consists of calculating the posterior distribution $P(f|D)$ and integrating over it. The predictions are made according to $E(y|\mathbf{x}, D)$, a function not necessarily belonging to $\mathcal{F}$. In the case of classification, $E(y|\mathbf{x}, D)$ does not take values in $\mathcal{Y}$ (although

thresholding the output just takes care of the problem). If we want a model in $\mathcal{F}$, we can use the Gibbs algorithm, which picks one sample in $\mathcal{F}$ according to the posterior distribution $P(f|D)$, or use the **MAP learning** approach. In Bayesian learning, analytically integrating over the posterior distribution is often impossible and the integral may be approximated by finite sum of models, weighted by positive coefficients (see **variational methods**) or by sampling models from the posterior distribution (see **Weighted majority algorithm** and **Monte-Carlo Markov Chain** or MCMC). The resulting estimators of $\hat{E}(y|\mathbf{x}, D)$ are convex combinations of functions in $\mathcal{F}$ and, in that sense, Bayesian learning is similar to **ensemble methods**.

Bias/variance decomposition. In the case of a least-square loss, the bias/variance decomposition is given by $E_D[(f(\mathbf{x}; D) - E[y|\mathbf{x}])^2] = (E_D[f(\mathbf{x}; D)] - E(y|\mathbf{x}))^2 + E_D[(f(\mathbf{x}; D) - E_D[f(\mathbf{x}; D)])^2]$. The second term (the "variance" of the estimator $f(\mathbf{x}, D)$) vanishes if $f(\mathbf{x}; D)$ equals $E_D[f(\mathbf{x}; D)$. The motivates the idea of using an approximation of $E_D[f(\mathbf{x}; D)$ as a predictor. In **bagging** the approximation is obtained by averaging over functions trained from m examples drawn at random with replacement from the training set D (bootstrap method). The method works best if $\mathcal{F}$ is not biased (*i.e.,* contains $E(y|\mathbf{x})$). Most models with low bias have a high variance and vice versa, hence the well-known bias/variance tradeoff.

Concept space. A space of data generative models from which the data are drawn. Not to be confused with **model space** or **hypothesis space**.

Empirical risk. An estimator of the **expected risk** that is the average of the loss over a finite number of examples drawn according to $P(\mathbf{x}, y)$: $R_{emp} = (1/m) \sum_{k=1}^{m} \mathcal{L}(f(\mathbf{x}_k), y_k)$.

Ensemble methods. Methods of building predictors using multiple **base learners**, which vote to make predictions. Predictions of y are made using a convex combination of functions $f_j \in \mathcal{F}$: $F(\mathbf{x}) = \sum_j p_j f_j(\mathbf{x})$, where p_j are positive coefficients. The two most prominent ensemble methods are bagging (Breiman, 1996) and boosting (Freund and Schapire, 1996). Bagging is a parallel ensemble method (all trees are built independently from training data subsets), while boosting is a serial ensemble method (trees complementing each other are progressively added to decrease the residual error). Random Forests (RF) (Breiman, 2001) are a variant of bagging methods in which both features and examples are subsampled. Boosting methods come in various flavors including Adaboost, Gentleboost, and Logitboost. The original algorithm builds successive models (called "weak learners") by resampling data in a way that emphasizes examples that have proved hardest to learn. Newer versions use a weighting scheme instead of resampling (Friedman, 2000).

Expected risk. The mathematical expectation of a risk functional over the unknown probability distribution $P(\mathbf{x}, y)$: $R[f] = \int \mathcal{L}(f(\mathbf{x}), y)\, dP(\mathbf{x}, y)$. Also called **generalization error**.

Generalization error. See **expected risk**.

Greedy search strategy. A search strategy, which does not revisit partial decisions already made, is called "greedy". Examples include forward selection and backward elimination in feature selection.

Guaranteed risk. A bound on the **expected risk**. See **PAC learning** and **Structural Risk Minimization** (SRM).

Hypothesis space. A space of models, which are fit to data, not necessarily identical to the **concept space** (which is often unknown).

Loss function. A function $\mathcal{L}(f(\mathbf{x}), y)$, which measures the discrepancy between target values y and model predictions $f(\mathbf{x})$. Examples include the square loss $(y - f(\mathbf{x}))^2$ for regression of the 0/1 loss $\mathbf{1}[f(\mathbf{x}) \neq y]$ for classification).

MAP learning. Maximum a posteriori (MAP) learning shares the same framework as Bayesian learning, but it is further assumed that the posterior $P(f|D)$ is concentrated and that $E(y|\mathbf{x}, D)$ can be approximated by $f^*(\mathbf{x})$, with $f^* = \operatorname{argmax}_f P(f|D) = \operatorname{argmax}_f P(D|f)P(f) = \operatorname{argmin}_f -\ln P(D|f) - \ln P(f)$. If we assume a uniform prior, we are brought back to maximum likelihood learning. If both $P(D|f)$ and $P(f)$ are exponentially distributed ($P(y|\mathbf{x}, f) = exp - \mathcal{L}(f(\mathbf{x}), y)$ and $P(f) = exp - \Omega[f]$), then MAP learning is equivalent to the minimization of a regularized risk functional.

Maximum likelihood learning. It is assumed that the data were generated by an input distribution $P(\mathbf{x})$, a function f from a **model space** $\mathcal{F}$ coinciding with the **concept space**, and a zero-mean **noise model**. For regression, for instance, if Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is assumed, y is distributed according to $P(y|\mathbf{x}, f) = \mathcal{N}(f(\mathbf{x}), \sigma^2)$. In the simplest case, $P(\mathbf{x})$ and the noise model are not subject to training (the values of $\mathbf{x}$ are fixed and the noise model is known). Learning then consists in searching for the function f^*, which maximizes the likelihood $P(D|f)$, or equivalently (since $P(\mathbf{x})$ is not subject to training) $f^* = \operatorname{argmax}_f P(\mathbf{y}|X, f) = \operatorname{argmin}_f -\ln P(\mathbf{y}|X, f)$. With the *i.i.d.* assumption, $f^* = \operatorname{argmax}_f \Pi_{k=1}^m P(y_k|\mathbf{x}_k, f) = \operatorname{argmin}_f \sum_{k=1}^m -\ln P(y_k|\mathbf{x}_k, f)$. For distributions belonging to the exponential family $P(y|\mathbf{x}, f) = \exp\{-\mathcal{L}(f(\mathbf{x}), y)\}$, the maximum likelihood method is equivalent to the method of minimizing the **empirical risk**. In the case of Gaussian noise, this corresponds to the method of least squares.

Model space. A space of predictive models, which are fit to data. Synonym of **hypothesis space**. For Bayesian models, also generally coincides with the **concept space**, but not for frequentists.

Monte-Carlo Markov Chain (MCMC) method. To approximate Bayesian integrals one can sample from the posterior distribution $P(f|D)$ following a Monte-Carlo Markov chain (MCMC), then make predictions according to $\hat{E}(y|\mathbf{x}, D) = \sum_j f_j(\mathbf{x})$. In a MCMC, at each step new candidate models $f_j \in \mathcal{F}$ are considered, in a local neighborhood of the model selected at the previous step. The new model is accepted if it provides a better fit to the data according to the posterior distribution or, if not, a random decision is made to accept it, following the Gibbs distribution (better models having a greater chance of acceptance).

Over-fitting avoidance. Model selection is traditionally associated with the so-called problem of **over-fitting** avoidance. Over-fitting means fitting the training examples well (*i.e.,* obtaining large model likelihood or low empirical risk values, computed from training data), but generalizing poorly on new test examples. Over-fitting is usually blamed on too large a large number of free parameters to be estimated, relative to the available number of training examples. The most basic model selection strategy is therefore to restrict the number of free parameters according to "strict necessity". This heuristic strategy is usually traced back in history to the principle known as Ockham's razor "Plurilitas non est ponenda sin necessitate" (William of Ockham, 14^{th} century). In other words, of two theories providing similarly good predictions, the simplest one should be preferred, *i.e.,* shave off unnecessary parameters. Most modern model selection strategies claim some affiliation with Ockham's razor, but the number of free parameters is replaced by a measure of **capacity** or **complexity** of the model class, $C[\mathcal{F}]$. Intuitively, model classes with large $C[\mathcal{F}]$ may include the correct model, but it is hard to find. In this case, even models with a low training error may have a large generalization error (high "variance"; over-fitting problem). Conversely, model classes with small $C[\mathcal{F}]$ may yield "biased" models, *i.e.,* with both high training and generalization error (under-fitting). See **bias/variance decomposition.**.

PAC learning. The "probably approximately correct" (PAC) learning procedures, seek a function minimizing a **guaranteed risk** $R_{gua}[f] = R_{emp}[f] + \epsilon(C, \delta)$ such that with (high) probability $(1 - \delta)$, $R[f] \leq R_{gua}[f]$. C is a measure of capacity or complexity.

Regularizers and regularization. The regularization method consists of replacing the minimization of the **empirical risk** $R_{emp}[f]$ by that of $R_{reg}[f] = R_{emp} + \Omega[f]$. A regularizer $\Omega[f]$ is a functional penalizing "complex" functions. If both $R_{tr}[f]$ and $\Omega[f]$ are convex, there is a unique minimum of $R_{reg}[f]$ with respect to f. In **MAP learning**, $-\ln P(f)$ can be thought of as a **regularizer**. One particularly successful regularizer is the 2-norm regularizer $\|f\|^2_{\mathcal{H}}$ for model functions $f(\mathbf{x}) = \sum_{k=1}^{m} \alpha_k K(\mathbf{x}, \mathbf{x}_k)$ belonging to a Reproducing Kernel Hilbert Space $\mathcal{H}$ (kernel methods). In the particular case of the linear model $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, we have $\|f\|^2_{\mathcal{H}} = \|\mathbf{w}\|^2$, a commonly used regularized found in many algorithms including ridge regression (Hoerl, 1962) and SVMs (Boser et al., 1992). In the general case, $\|f\|^2_{\mathcal{H}} = \mathbf{f} K^{-1} \mathbf{f} = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$, where $\mathbf{f} = [f(\mathbf{x}_k)]_{k=1}^m$ is the vector of predictions of the training examples, $\boldsymbol{\alpha} = [\alpha_k]_{k=1}^m$ and $K = [K(\mathbf{x}_h, \mathbf{x}_k]$, $h = 1, ...m$ $k = 1, ...m$. Due to the duality between RKHS and stochastic processes (Wahba, 1990), the functions in the RKHS can also be explained as a family of random variables in a Gaussian process, assuming a prior $P(f)$ proportional to $\exp(-\gamma \|f\|_{\mathcal{H}}) = \exp(-\gamma \mathbf{f} K^{-1} \mathbf{f})$ and the kernel matrix K is interpreted as a covariance matrix $K(\mathbf{x}_h, \mathbf{x}_k) = \text{cov}[f(x_k), f(x_k)]$.

Risk minimization. Given a **model space** or **hypothesis space** $\mathcal{F}$ of functions $y = f(\mathbf{x})$, and a **loss function** $\mathcal{L}(f(\mathbf{x}), y)$, we want to find the function $f^* \in \mathcal{F}$ that minimizes the **expected risk** $R[f] = \int \mathcal{L}(f(\mathbf{x}), y)\, dP(\mathbf{x}, y)$. Since $P(\mathbf{x}, y)$ is unknown, only estimations of $R[f]$ can be computed. The simplest estimator is the average of the loss over a finite number of examples drawn according to $P(\mathbf{x}, y)$ called the **empirical risk**: $R_{emp} = (1/m) \sum_{k=1}^{m} \mathcal{L}(f(\mathbf{x}_k), y_k)$. The minimization of the empirical risk is the

basis of many machine learning approaches to selecting f^*, but minimizing regularized risk functionals is often preferred. See **regularization**. Also, related are the **PAC learning** procedures and the method of **Structural Risk Minimization** (SRM).

Search strategy. There are *optimal search strategies*, which guarantee that the optimum of the evaluation function will be found, including the *exhaustive search* method, for discrete hyper-parameter spaces. The popular *grid search* method for continuous hyper-parameter spaces performs an exhaustive search, up to a certain precision. A related *stochastic search* method is *uniform sampling*. Uniformly sampling parameter space may be computationally expensive and inefficient. If we use a non-uniform distribution $G(\boldsymbol{\theta})$ to sample hyper-parameter space, which resembles $P(f(\boldsymbol{\theta})|D)$, the search can be made more efficient. This idea is exploited in *rejection sampling* and *importance sampling*: according to these methods a Bayesian ensemble $F(\mathbf{x}) = \sum_k w_k f(\mathbf{x}; \boldsymbol{\theta}_k)$ would use weight w_k proportional to $P(f(\boldsymbol{\theta})|D)/G(\boldsymbol{\theta})$. Because of the computational burden of (near) optimum strategies, other strategies are often employed, usually yielding only a *local optimum*. These include *sequential search strategies* such as *coordinate ascent* or descent (making small steps along coordinate axes) or *pattern search* (Momma and Bennett, 2002) (making local steps according to a certain pattern), which, by accepting only moves that improve the evaluation function, find the local optimum nearest to the starting point. Some stochastic search methods accept moves not necessarily improving the value of the evaluation function, like simulated annealing or **Markov chain Monte Carlo (MCMC) methods**. Both methods accept all moves improving the evaluation function and some moves that do not, *e.g.,* with probability $\exp -\Delta r/T$, where T is a positive parameter (T=1 for MCMC and progressively diminishes for simulated annealing). Such stochastic methods search hyper-parameter space more intensively and do not become stuck in the nearest local optimum of the evaluation function.

Semi-supervised learning. In semi-supervised learning, in addition to the labeled data, the learning machine is given a (possibly large) set of unlabeled data. Such unlabeled data may be used for training or model selection.

Structural Risk Minimization. The method of Structural Risk Minimization (SRM) provides aeans of building regularized risk functionals (see **Regularization**), using the idea of **guaranteed risk** minimization, but not requiring the calculation of the model class capacity or complexity, which is often unknown or hard to compute. In the risk minimization framework, it is not assumed that the model space includes a function or "concept", which generated the data (see **concept space** and **hypothesis space**).

Supervised learning. Learning with teaching signal or target y.

Under-fitting. While **over-fitting** is the problem of learning the training data too well the expense of a large **generalization error**, under-fitting is the problem of having a too weak model not even capable of learning the training data and also generalizing poorly.

Unsupervised learning. Learning in the absence of teaching signal or target y.

Weighted majority algorithm. To approximate Bayesian integrals one can draw samples f_j uniformly from the model space of functions $\mathcal{F}$ and make predictions according to $\hat{E}(y|\mathbf{x}, D) = \sum_j P(f_j|D) f_j(\mathbf{x})$.