

Bias-Variance trade-offs: Novel Applications

Dev Rajnarayan

David Wolpert

May 16, 2007

Synonyms

Bias-variance trade-offs, bias plus variance.

Definition

Consider a given random variable F and a random variable that we can modify, \hat{F} . We wish to use a sample of \hat{F} as an estimate of a sample of F . The mean squared error between such a pair of samples is a sum of four terms. The first term reflects the statistical coupling between F and \hat{F} and is conventionally ignored in bias-variance analysis. The second term reflects the inherent noise in F and is independent of the estimator \hat{F} . Accordingly, we cannot affect this term. In contrast, the third and fourth terms depend on \hat{F} . The third term, called the bias, is independent of the precise samples of both F and \hat{F} , and reflects the difference between the means of F and \hat{F} . The fourth term, called the variance, is independent of the precise sample of F , and reflects the inherent noise in the estimator as one samples it. These last two terms can be modified by changing the choice of the estimator. In particular, on small sample sets, we can often decrease our mean squared error by, for instance, introducing a small bias that causes a large reduction the variance. While most commonly used in statistics / machine learning, this article shows that such bias-variance trade-offs are applicable in a much broader context and in a variety of situations. We also show, using experiments, how techniques for optimizing bias-variance trade-offs introduced in machine learning can be applied in novel circumstances to improve the performance of a class of optimization algorithms.

Motivation and Background

In its simplest form, the bias-variance decomposition is based on the following idea. Say we have a random variable F taking on values F distributed according to a density function $p(F)$. We want to estimate the value of a sample from $p(F)$. To form our estimate, we sample a different random variable \hat{F} taking on values \hat{F} distributed according to $p(\hat{F})$. Assuming a quadratic loss function, the quality of our estimate is measured by its Mean Squared Error (MSE):

$$\text{MSE}(\hat{F}) \equiv \int p(\hat{F}, F) (\hat{F} - F)^2 d\hat{F} dF.$$

In many situations, F and \hat{F} are dependent variables. For example, in supervised machine learning, F is a ‘target’ conditional distribution, stochastically mapping elements of an input space X into a space Y of output variables. The associated distribution $p(F)$ is the ‘prior’ of F . A random sample \mathcal{D} of F , called ‘the training set’, is generated, and \mathcal{D} is used in a ‘learning algorithm’ to produce \hat{F} , which is our estimate of the value of a new sample of F . Clearly, this F and \hat{F} are statistically dependent, via \mathcal{D} .

Indeed, intuitively speaking, the goal in designing a learning algorithm is that the \hat{F} 's it produces are positively correlated with F 's.

In practice this coupling is simply ignored in analyses of bias plus variance, without any justification (one such justification could be that the coupling has little effect on the value of the MSE). We shall follow that practice here, treating \hat{F} and F as independent random variables. Accordingly, our equation for MSE reduces to

$$\text{MSE}(\hat{F}) = \int p(\hat{F})p(F) (\hat{F} - F)^2 d\hat{F}dF. \quad (1)$$

If we were to account for the coupling of \hat{F} and F an additive correction term would need to be added to the right-hand side. For instance, see Wolpert [1997].

Using simple algebra, the right hand side of Eq. 1 can be written as the sum of three terms. The first is the variance of F . Since this is beyond our control in designing the estimator \hat{F} , we ignore it for the rest of this article. The second term involves a mean that describes the deterministic component of the error. This term depends on both the distribution of F and that of \hat{F} , and quantifies how close the means of those distributions are. The third term is a variance that describes stochastic variations from one sample to the next. This term is independent of the random variable being estimated. Formally, up to an overall additive constant, we can write

$$\begin{aligned} \text{MSE}(\hat{F}) &= \int p(\hat{F})(\hat{F}^2 - 2F\hat{F} + F^2)d\hat{F}, \\ &= \int p(\hat{F})\hat{F}^2 d\hat{F} - 2F \int p(\hat{F})\hat{F} d\hat{F} + F^2, \\ &= \overbrace{\mathbb{V}(\hat{F}) + [\mathbb{E}(\hat{F})]^2} - 2F \mathbb{E}(\hat{F}) + F^2, \\ &= \mathbb{V}(\hat{F}) + \underbrace{[F - \mathbb{E}(\hat{F})]^2}, \\ &= \text{variance} + \text{bias}^2. \end{aligned} \quad (2)$$

In light of Eq. 2, one way to try to reduce expected quadratic error is to modify an estimator to trade-off bias and variance. Some of the most famous applications of such bias-variance trade-offs occur in parametric machine learning, where many techniques have been developed to exploit the trade-off. Nonetheless, the trade-off also arises in many other fields, including integral estimation and optimization. In the rest of this paper we present a few novel applications of bias-variance trade-off, and describe some interesting features in each case. A recurring theme is the following: whenever a bias-variance trade-off arises in a particular field, we can use many techniques from parametric machine learning that have been developed for exploiting this trade-off. See Wolpert and Rajnarayan [2007] for further details of many of these applications.

Applications

In this section, we describe some bias-variance tradeoffs. First, we describe Monte Carlo (MC) techniques for the estimation of integrals, and provide a brief analysis of bias-variance trade-offs in this context. Next, we introduce the field of Monte Carlo Optimization (MCO), and illustrate that it involves more subtleties than does simple MC. Then, we describe the field of Parametric Machine Learning, which, as we show, is a special case of MCO. Finally, we describe the application of Parametric Learning (PL) techniques to improve the performance of MCO algorithms. We do this in the context of an MCO problem that addresses black-box optimization.

Monte Carlo Estimation of Integrals Using Importance Sampling

Monte Carlo methods are often the method of choice for estimating difficult high-dimensional integrals. Consider a function $f: X \rightarrow \mathbb{R}$, which we want to integrate over some region $\mathcal{X} \subseteq X$, yielding the value F , as given by

$$F = \int_{\mathcal{X}} dx f(x).$$

We can view this as a random variable \underline{F} , with density function given by a Dirac delta function centered on F . Therefore, the variance of \underline{F} is 0, and Eq. 2 is exact.

A popular MC method to estimate this integral is importance sampling [see Robert and Casella, 2004]. This exploits the law of large numbers as follows: i.i.d. samples $x^{(i)}$, $i = 1, \dots, m$ are generated from a so-called importance distribution $h(x)$ that we control, and the associated values of the integrand, $f(x^{(i)})$ are computed. Denote these ‘data’ by

$$\mathcal{D} = \{(x^{(i)}, f(x^{(i)}), i = 1, \dots, m\}. \quad (3)$$

Now,

$$\begin{aligned} F &= \int_{\mathcal{X}} dx h(x) \frac{f(x)}{h(x)}, \\ &= \lim_{m \rightarrow \infty} \frac{1}{m} \sum_{i=1}^m \frac{f(x^{(i)})}{h(x^{(i)})} \text{ with probability 1.} \end{aligned}$$

Denote by $\hat{\underline{F}}$ the random variable with value given by the sample average for \mathcal{D} :

$$\hat{\underline{F}} = \frac{1}{m} \sum_{i=1}^m \frac{f(x^{(i)})}{h(x^{(i)})}.$$

We use $\hat{\underline{F}}$ as our statistical estimator for \underline{F} , as we broadly described in the introductory section. Assuming a quadratic loss function, $L(\hat{\underline{F}}, F) = (F - \hat{\underline{F}})^2$, the bias-variance decomposition described in Eq. 2 applies *exactly*. It can be shown that the estimator $\hat{\underline{F}}$ is unbiased, that is, $\mathbb{E}(\hat{\underline{F}}) = F$, where the mean is over samples of h . Consequently, the MSE of this estimator is just its variance. The choice of sampling distribution h that minimizes this variance is given by [see Robert and Casella, 2004]

$$h^*(x) = \frac{|f(x)|}{\int_{\mathcal{X}} |f(x')| dx'}.$$

By itself, this result is not very helpful, since the equation for the optimal importance distribution contains a similar integral to the one we are trying to estimate. For non-negative integrands $f(x)$, the VEGAS algorithm [Lepage, 1978] describes an adaptive method to find successively better importance distributions, by iteratively estimating \underline{F} , and then using that estimate to generate the next importance distribution h . In the case of these unbiased estimators, there is no trade-off between bias and variance, and minimizing MSE is achieved by minimizing variance.

Monte Carlo Optimization

Instead of a *fixed* integral to evaluate, consider a parametrized integral

$$F(\theta) = \int_{\mathcal{X}} dx f_{\theta}(x).$$

Further, suppose we are interested in finding the value of the parameter $\theta \in \Theta$ that minimizes $F(\theta)$:

$$\theta^* = \arg \min_{\theta \in \Theta} F(\theta).$$

In the case where the functional form of f_θ is not explicitly known, one approach to solve this problem is a technique called Monte Carlo Optimization (MCO) [see Ermoliev and Norkin, 1998], involving repeated MC estimation of the integral in question with adaptive modification of the parameter θ .

We proceed by analogy to the case with MC. First, we introduce the θ -indexed random variable $\underline{F}(\theta)$, all of whose components have delta-function distributions about the associated values $F(\theta)$. Next, we introduce a θ -indexed vector random variable $\underline{\hat{F}}$ with values

$$\hat{F} \equiv \{\hat{F}(\theta) \forall \theta \in \Theta\}. \quad (4)$$

Each real-valued component $\hat{F}(\theta)$ can be sampled and viewed as an estimate of $\underline{F}(\theta)$.

For example, let \mathcal{D} be a data set as described in Eq. 3. Then for every θ , any sample of \mathcal{D} provides an associated estimate

$$\hat{F}(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{f_\theta(x^{(i)})}{h(x^{(i)})}.$$

That average serves as an estimate of $\underline{F}(\theta)$. Formally, \hat{F} is a function of the random variable \mathcal{D} , and is given by such averaging over the elements of \mathcal{D} . So, a sample of \mathcal{D} provides a sample of \hat{F} . *A priori*, we make no restrictions on \hat{F} , and so, in general, its components may be statistically coupled with one another. Note that this coupling arises even though we are, for simplicity, treating each function $\underline{F}(\theta)$ as a delta function rather than as having a non-zero variance to reflect our lack of knowledge of the $f(\theta)$ functions.

However \hat{F} is defined, given a sample of \hat{F} , one way to estimate θ^* is

$$\hat{\theta}^* = \arg \min_{\theta \in \Theta} \hat{F}(\theta)$$

We call this approach ‘natural’ MCO. As an example, say that \mathcal{D} is a set of m samples of h , and let

$$\hat{F}(\theta) \triangleq \frac{1}{m} \sum_{i=1}^m \frac{f_\theta(x^{(i)})}{h(x^{(i)})},$$

as above. Under this choice for \hat{F} ,

$$\hat{\theta}^* = \arg \min_{\theta \in \Theta} \frac{1}{m} \sum_{i=1}^m \frac{f_\theta(x^{(i)})}{h(x^{(i)})}. \quad (5)$$

We call this approach ‘naive’ MCO.

Consider *any* algorithm that estimates θ^* as a single-valued function of \hat{F} . The estimate of θ^* produced by that algorithm is itself a random variable, since it is a function of the random variable \hat{F} . Call this random variable $\hat{\theta}^*$, taking on values $\hat{\theta}^*$. Any MCO algorithm is defined by $\hat{\theta}^*$; that random variable encapsulates the output estimate made by the algorithm.

To analyze the error of such an algorithm, consider the associated random variable given by the associated integral, $F(\hat{\theta}^*)$. The difference between a sample of $F(\hat{\theta}^*)$ and the true minimal value of the integral, $F(\theta^*) = \min_{\theta} F(\theta)$, is the error introduced by our estimating that optimal θ as a sample of $\hat{\theta}^*$. In light of this, since our aim in MCO is to minimize $F(\theta)$, we adopt the loss function $L(\hat{\theta}^*, \theta^*) \triangleq F(\hat{\theta}^*) - F(\theta^*)$. (This is in contrast to our discussion on MC integration, which involved quadratic loss.) This loss function just equals $F(\hat{\theta}^*)$ up to an additive constant $F(\theta^*)$ that is fixed by the MCO problem at hand and is beyond our control. Up to that additive constant, the associated expected loss is

$$\mathbb{E}(L) = \int d\hat{\theta}^* p(\hat{\theta}^*) F(\hat{\theta}^*). \quad (6)$$

Now change coordinates in this integral from the values of the scalar random variable $\hat{\theta}^*$ to the values of the underlying vector random variable \hat{F} . The expected loss now is written as

$$\mathbb{E}(L) = \int d\hat{F} p(\hat{F}) F(\hat{\theta}^*(\hat{F})).$$

The natural MCO algorithm provides some insight into these results. For that algorithm,

$$\begin{aligned} \mathbb{E}(L) &= \int d\hat{F} p(\hat{F}) F(\arg \min_{\theta} \hat{F}(\theta)) \\ &= \int d\hat{F}(\theta_1) d\hat{F}(\theta_2) \dots p(\hat{F}(\theta_1), \hat{F}(\theta_2), \dots) F(\arg \min_{\theta} \hat{F}(\theta)). \end{aligned} \quad (7)$$

For any fixed θ , there is an error between samples of $\hat{F}(\theta)$ and the true value $F(\theta)$. Bias-variance considerations apply to this error, exactly as in the discussion of MC above.

We are not, however, concerned with \hat{F} for a single component θ , but rather for a set Θ of θ 's. The simplest such case is where the components of \hat{F} are independent. For this case, $\arg \min_{\theta} \hat{F}(\theta)$ is distributed according to the laws for extrema of multiple independent random variables. This distribution depends on moments of all orders of each random variable $\hat{F}(\theta)$. This means that $\mathbb{E}[L]$ also depends on moments of all orders. However only the first two moments contribute to the bias and variance for any single θ . Thus, even in the simplest possible case, the bias-variance considerations for the individual θ do not provide a complete analysis.

In most cases, the components of \hat{F} are *not* independent. Therefore, in order to analyze $\mathbb{E}[L]$, in addition to higher moments of the distribution for each θ , we must now also consider higher-order moments coupling the estimates $\hat{F}(\theta)$ for different θ , e.g., covariances.

Due to these effects, it may be quite acceptable for all the components $\hat{F}(\theta)$ to have both a large bias and a large variance, as long as they still order the θ 's correctly with respect to the true $F(\theta)$. In such a situation, large covariances could ensure that if some $\hat{F}(\theta)$ were incorrectly large, then $\hat{F}(\theta')$, $\theta' \neq \theta$ would also be incorrectly large. This coupling between the components of \hat{F} would preserve the ordering of θ 's under \underline{F} . So, even with large bias and variance for each θ , the estimator as a whole would still work well.

Nevertheless, it *is* sufficient to design estimators $\hat{F}(\theta)$ with sufficiently small bias plus variance for each single θ . More precisely, suppose that those terms are very small on the scale of differences $F(\theta) - F(\theta')$ for any θ and θ' . Then by Chebychev's inequality, we know that the density functions of the random variables $\hat{F}(\theta)$ and $\hat{F}(\theta')$ have almost no overlap. Accordingly, the probability that a sample of $\hat{F}(\theta) - \hat{F}(\theta')$ has the opposite sign of $F(\theta) - F(\theta')$ is almost zero.

Evidently, $\mathbb{E}[L]$ is generally determined by a complicated relationship involving bias, variance, covariance, and higher moments. Natural MCO in general, and naive MCO in particular, ignore all of these effects, and consequently, often perform quite poorly in practice. In the next section we discuss some ways of addressing this problem.

Parametric Machine Learning

There are many versions of the basic MCO problem described in the previous section. Some of the best-explored arise in parametric density estimation and parametric supervised learning, which together comprise the field of Parametric machine Learning (PL).

In particular, parametric supervised learning attempts to solve

$$\arg \min_{\theta \in \Theta} \int dx p(x) \int dy p(y | x) f_{\theta}(x).$$

Here, the values x represent inputs, and the values y represent corresponding outputs, generated according to some stochastic process defined by a set of conditional distributions $\{p(y | x), x \in \mathcal{X}\}$.

Typically, one tries to solve this problem by casting it as an MCO problem. For instance, say we adopt a quadratic loss between a predictor $z_\theta(x)$ and the true value of y . Using MCO notation, we can express the associated supervised learning problem as finding $\arg \min_\theta F(\theta)$, where

$$\begin{aligned} l_\theta(x) &= \int dy p(y | x) (z_\theta(x) - y)^2, \\ f_\theta(x) &= p(x) l_\theta(x), \\ F(\theta) &= \int dx f_\theta(x). \end{aligned} \tag{8}$$

Next, the argmin is estimated by minimizing a sample-based estimate of the $F(\theta)$'s. More precisely, we are given a ‘training set’ of samples of $p(y | x)p(x)$, $\{(x^{(i)}, y^i) | i = 1, \dots, m\}$. This training set provides a set of associated estimates of $F(\theta)$:

$$\hat{F}(\theta) = \frac{1}{m} \sum_{i=1}^m l_\theta(x^{(i)}).$$

These are used to estimate $\arg \min_\theta F(\theta)$, exactly as in MCO. In particular, one could estimate the minimizer of $F(\theta)$ by finding the minimum of $\hat{F}(\theta)$, just as in natural MCO. As mentioned above, this MCO algorithm can perform very poorly in practice. In PL, this poor performance is called ‘overfitting the data’.

PL Techniques That Address Overfitting

There are several formal approaches that have been explored in PL to try to address this ‘overfitting the data’. Interestingly, none are based on direct consideration of the random variable $F(\hat{\theta}^*(\hat{F}))$ and the ramifications of its distribution for expected loss (cf. Eq. 7). In particular, no work has applied the mathematics of extrema of multiple random variables to analyze the bias-variance-covariance trade-offs encapsulated in Eq. 7.

The PL approach that perhaps comes closest to such direct consideration of the distribution of $F(\hat{\theta}^*)$ is uniform convergence theory, which is a central part of Computational Learning Theory [see Angluin, 1992]. Uniform convergence theory starts by crudely encapsulating the quadratic loss formula for expected loss under natural MCO, Eq. 7. It does this by considering the worst-case bound, over possible $p(x)$ and $p(y | x)$, of the probability that $F(\hat{\theta}^*)$ exceeds $\min_\theta F(\theta)$ by more than κ . It then examines how that bound varies with κ . In particular, it relates such variation to characteristics of the set of functions $\{f_\theta : \theta \in \Theta\}$, e.g., the ‘VC dimension’ of that set [see Vapnik, 1982, 1995].

Another, historically earlier approach, is to apply bias-plus-variance considerations to the *entire* PL algorithm $\hat{\theta}^*$, rather than to each $\hat{F}(\theta)$ separately. This approach is applicable for algorithms that do not use natural MCO, and even for non-parametric supervised learning. As formulated for parameteric supervised learning, this approach combines the formulas in Eq. 8 to write

$$F(\theta) = \int dx dy p(x)p(y | x)(z_\theta(x) - y)^2.$$

This is then substituted into Eq. 6, giving

$$\begin{aligned} \mathbb{E}[L] &= \int d\hat{\theta}^* dx dy p(x)p(y | x)p(\hat{\theta}^*)(z_{\hat{\theta}^*}(x) - y)^2 \\ &= \int dx p(x) \left[\int d\hat{\theta}^* dy p(x)p(y | x)p(\hat{\theta}^*)(z_{\hat{\theta}^*}(x) - y)^2 \right]. \end{aligned} \tag{9}$$

The term in square brackets is an x -parameterized expected quadratic loss, which can be decomposed into a bias, variance, etc., in the usual way.

This formulation of $\mathbb{E}[L]$ eliminates any direct concern for issues like the distribution of extrema of multiple random variables, covariances between $\hat{F}(\theta)$ and $\hat{F}(\theta')$ for different values of θ , and so on. It has been invoked to justify, and sometimes derive, many PL techniques that work well in practice. An example is the technique of regularization, in which a data-independent penalty vector is added to the vector \hat{F} . This penalty is known as the regularizer. When used with natural MCO, that regularizer introduces an *a priori* preference for $\hat{\theta}^*$ to favor certain θ 's over others. This preference can be viewed as a bias. The benefit of regularization is that it reduces variance; the hope is that the reduction in variance outweighs the increase in bias.

Often the regularizer has a parameter that determines how strongly it distorts $\hat{\theta}^*$. For example, there is an overall multiplicative parameter that controls the penalty vector's magnitude. Different values of such a regularization parameter change the amount of introduced bias and variance. The technique of cross-validation can be used to search for the optimal value of that parameter, i.e., the optimal trade-off of bias and variance.

There are numerous other approaches for addressing the problems of natural MCO that have been explored in PL. Particularly important among these are Bayesian approaches, e.g., Buntine and Weigend [1991], Berger [1985], Mackay [2003]. Based on these approaches, as well as on intuition, many powerful techniques for addressing data-overfitting have been explored in PL, going beyond regularization and cross-validation, to include stacking, bagging, etc. Essentially all of these techniques can be applied to *any* MCO problem, not just PL problems. Since many of these techniques can be justified using Eq. 9, they provide a way to exploit the bias-variance trade-off in other domains besides PL.

PLMCO

In this section, we illustrate how PL techniques that exploit the bias-variance decomposition of Eq. 9 can be used to improve an MCO algorithm used in a domain outside of PL. This MCO algorithm is a version of adaptive importance sampling, somewhat similar to the CE method [Rubinstein and Kroese, 2004], and is related to function smoothing on continuous spaces. The PL techniques described are applicable to any other MCO problem, and this particular one is chosen just as an example.

MCO Problem Description

The problem is to find the θ -parameterized distribution q_θ that minimizes the associated expected value of a function $G: \mathbb{R}^n \rightarrow \mathbb{R}$, i.e., find

$$\arg \min_{\theta} \mathbb{E}_{q_\theta}[G].$$

We are interested in versions of this problem where we do not know the functional form of G , but can obtain its value $G(x)$ at any $x \in \mathcal{X}$. Similarly we cannot assume that G is smooth, nor can we evaluate its derivatives directly. This scenario arises in many fields, including blackbox optimization [see Wolpert et al., 2006], and risk minimization [see Ermoliev and Norkin, 1998].

We begin by expressing this minimization problem as an MCO problem. We know that

$$\mathbb{E}_{q_\theta}[G] = \int_{\mathcal{X}} dx q_\theta(x) G(x)$$

Using MCO terminology, $f_\theta(x) = q_\theta(x)G(x)$ and $F(\theta) = \mathbb{E}_{q_\theta}[G]$. To apply MCO, we must define a vector-valued random variable \hat{F} with components indexed by θ , and then use a sample of \hat{F} to estimate $\arg \min_{\theta} \mathbb{E}_{q_\theta}[G]$. In particular, to apply naive MCO to estimate $\arg \min_{\theta} \mathbb{E}_{q_\theta}(G)$, we first i.i.d. sample a density function $h(x)$. By evaluating the associated values of $G(x)$ we get a data set

$$\begin{aligned} \mathcal{D} &\equiv (\mathcal{D}_X, \mathcal{D}_G) \\ &= (\{x^{(i)} : i = 1, \dots, m\}, \{G(x^{(i)}) : i = 1, \dots, m\}). \end{aligned}$$

The associated estimates of $F(\theta)$ for each θ are

$$\hat{F}(\theta) \triangleq \frac{1}{m} \sum_{i=1}^m \frac{q_\theta(x^{(i)})G(x^{(i)})}{h(x^{(i)})}. \quad (10)$$

The associated naive MCO estimate of $\arg \min_\theta \mathbb{E}_{q_\theta}[G]$ is

$$\hat{\theta}^* \equiv \arg \min_\theta \hat{F}(\theta).$$

Suppose Θ includes all possible density functions over x 's. Then the q_θ minimizing our estimate is a delta function about the $x^{(i)} \in \mathcal{D}_X$ with the lowest associated value of $G(x^{(i)})/h(x^{(i)})$. This is clearly a poor estimate in general; it suffers from ‘data-overfitting’. Proceeding as in PL, one way to address this data-overfitting is to use regularization. In particular, we can use the entropic regularizer, given by the negative of the Shannon entropy $S(q_\theta)$. This changes the definition of \hat{F} from the function given in Eq. 10 to

$$\hat{F}(\theta) \triangleq \frac{1}{m} \sum_{i=1}^m \frac{\beta q_\theta(x^{(i)})G(x^{(i)})}{h(x^{(i)})} - S(q_\theta).$$

where β is the inverse of the regularization parameter controlling the magnitude of the penalty term $S(q_\theta)$.

Solution Methodology

Unfortunately, it can be difficult to find the θ globally minimizing this new \hat{F} for an arbitrary \mathcal{D} . An alternative is to find a close approximation to that optimal θ . One way to do this is as follows. First, we find the minimizer of

$$\frac{1}{m} \sum_{i=1}^m \frac{\beta p(x^{(i)})G(x^{(i)})}{h(x^{(i)})} - S(p) \quad (11)$$

over the set of *all* possible distributions $p(x)$ with domain \mathcal{X} . We then find the q_θ that has minimal Kullback-Leibler (KL) divergence from this p , evaluated over \mathcal{D}_X . That serves as our approximation to $\arg \min_\theta \hat{F}(\theta)$, and therefore as our estimate of the θ that minimizes $\mathbb{E}_{q_\theta}(G)$.

The minimizer p of Eq. 11 can be found in closed form; over \mathcal{D}_X it is the Boltzmann distribution $p^\beta(x^{(i)}) \propto \exp(-\beta G(x^{(i)}))$. The KL divergence in \mathcal{D}_X from this Boltzmann distribution to q_θ is

$$F(\theta) = \text{KL}(p^\beta \| q_\theta) = \int_{\mathcal{D}_X} dx p^\beta(x) \log \left(\frac{p^\beta(x)}{q_\theta(x)} \right).$$

The minimizer of this KL divergence is given by

$$\theta^\dagger = \arg \min_\theta \left[- \sum_{i=1}^m \frac{\exp(-\beta G(x^{(i)}))}{h(x^{(i)})} \log(q_\theta(x^{(i)})) \right]. \quad (12)$$

This approach is an approximation to a regularized version of the naive MCO estimate of the θ that minimizes $\mathbb{E}_{q_\theta}(G)$. The application of the technique of regularization in this context has the same motivation as it does in PL: to reduce bias plus variance.

Log-concave Densities

If q_θ is log-concave in its parameters θ , then the minimization problem in Eq. 12 is a convex optimization problem, and the optimal parameters can be found closed-form. Denote the likelihood ratios by $s^{(i)} =$

$\exp(-\beta G(x^{(i)}))/h(x^{(i)})$. Differentiating Eq. 12 with respect to the parameters μ and Σ^{-1} and setting them to zero yields

$$\begin{aligned}\mu^* &= \frac{\sum_{\mathcal{D}} s^{(i)} x^{(i)}}{\sum_{\mathcal{D}} s^{(i)}} \\ \Sigma^* &= \frac{\sum_{\mathcal{D}} s^{(i)} (x^{(i)} - \mu^*)(x^{(i)} - \mu^*)^T}{\sum_{\mathcal{D}} s^{(i)}}\end{aligned}$$

Mixture Models

The single Gaussian is a fairly restrictive class of models. Mixture models can significantly improve flexibility, but at the cost of convexity of the KL distance minimization problem. However, a plethora of techniques from supervised learning, in particular the Expectation Maximization (EM) algorithm, can be applied with minor modifications.

Suppose q_θ is a mixture of M Gaussians, that is, $\theta = (\mu, \Sigma, \phi)$ where ϕ is the mixing p.m.f, we can view the problem as one where a hidden variable z decides which mixture component each sample is drawn from. We then have the optimization problem

$$\text{minimize } - \sum_{\mathcal{D}} \frac{p(x^{(i)})}{h(x^{(i)})} \log \left(q_\theta(x^{(i)}, z^{(i)}) \right).$$

Following the standard EM procedure, we get the algorithm described in Eq. 13. Since this is a nonconvex problem, one typically runs the algorithm multiple times with random initializations of the parameters.

$$\begin{aligned}\text{E-step: For each } i, \text{ set } Q_i(z^{(i)}) &= p(z^{(i)}|x^{(i)}), \\ \text{that is, } w_j^{(i)} &= q_{\mu, \Sigma, \phi}(z^{(i)} = j|x^{(i)}), \quad j = 1, \dots, M. \\ \text{M-step: Set } \mu_j &= \frac{\sum_{\mathcal{D}} w_j^{(i)} s^{(i)} x^{(i)}}{\sum_{\mathcal{D}} w_j^{(i)} s^{(i)}}, \\ \Sigma_j &= \frac{\sum_{\mathcal{D}} w_j^{(i)} s^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{\mathcal{D}} w_j^{(i)} s^{(i)}}, \\ \phi_j &= \frac{\sum_{\mathcal{D}} w_j^{(i)} s^{(i)}}{\sum_{\mathcal{D}} s^{(i)}},\end{aligned}\tag{13}$$

Test Problems

To compare the performance of this algorithm with and without the use of PL techniques, we use a couple of very simple academic problems in two and four dimensions - the Rosenbrock function in two dimensions, given by

$$G_R(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

and the Woods function in four dimensions, given by given by

$$\begin{aligned}G_{\text{woods}}(x) &= 100(x_2 - x_1)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &+ 10.1[(1 - x_2)^2 + (1 - x_4)^2] + 19.8(1 - x_2)(1 - x_4).\end{aligned}$$

For the Rosenbrock, the optimum value of 0 is achieved at $x = (1, 1)$, and for the Woods problem, the optimum value of 0 is achieved at $x = (1, 1, 1, 1)$.

Application of PL Techniques

As mentioned above, there are many PL techniques beyond regularization that are designed to optimize the trade-off between bias and variance. So having cast the solution of $\arg \min_{q_\theta} \mathbb{E}(G)$ as an MCO problem, we can apply those other PL techniques instead of (or in addition to) entropic regularization.

This should improve the performance of our MCO algorithm, for the exact same reason that using those techniques to trade off bias and variance improves performance in PL. We briefly mention some of those alternative techniques here.

The overall MCO algorithm is broadly described in Alg. 1. For the Woods problem, 20 samples of x are drawn from the updated q_θ at each iteration, and for the Rosenbrock, 10 samples. For comparing various methods and plotting purposes, 1000 samples of $G(x)$ are drawn to evaluate $\mathbb{E}_{q_\theta}[G(x)]$. Note: in an actual optimization, we will not be drawing these test samples! All the performance results in Fig. 1 are based on 50 runs of the PC algorithm, randomly initialized each time. The sample mean performance across these runs is plotted along with 95% confidence intervals for this sample mean (shaded regions).

Algorithm 1 Overview of pq minimization using Gaussian mixtures

- 1: Draw uniform random samples on X
 - 2: Initialize regularization parameter β
 - 3: Compute $G(x)$ values for those samples
 - 4: **repeat**
 - 5: Find a mixture distribution q_θ to minimize sampled pq KL distance
 - 6: Sample from q_θ
 - 7: Compute $G(x)$ for those samples
 - 8: Update β
 - 9: **until** Termination
 - 10: Sample final q_θ to get solution(s).
-

Cross-validation for Regularization: We note that we are using regularization to reduce variance, but that regularization introduces bias. As is done in PL, we use standard k -fold cross-validation to tradeoff this bias and variance. We do this by partitioning the data into k disjoint sets. The held-out data for the i^{th} fold is just the i^{th} partition, and the held-in data is the union of all other partitions. First, we ‘train’ the regularized algorithm on the held-in data \mathcal{D}_t to get an optimal set of parameters θ^* , then ‘test’ this θ^* by considering unregularized performance on the held-out data \mathcal{D}_v . In our context, ‘training’ refers to finding optimal parameters by KL distance minimization using the held-in data, and ‘testing’ refers to estimating $\mathbb{E}_{q_\theta}[G(x)]$ on the held-out data using the following formula [Robert and Casella, 2004].

$$\hat{g}(\theta) = \frac{\sum_{\mathcal{D}_v} \frac{q_\theta(x^{(i)})G(x^{(i)})}{h(x^{(i)})}}{\sum_{\mathcal{D}_v} \frac{q_\theta(x^{(i)})}{h(x^{(i)})}}.$$

We do this for several values of the regularization parameter β in the interval $k_1\beta < \beta < k_2\beta$, and choose the one that yield the best held-out performance, averaged over all folds. For our experiments, $k_1 = 0.5, k_2 = 3$, and we use 5 equally-spaced values in this interval. Having found the best regularization parameter in this range, we then use *all* the data to minimize KL distance using this optimal value of β . Note that all cross-validation is done *without* any additional evaluations of $G(x)$. Cross-validation for β in PC is similar to optimizing the annealing schedule in simulated annealing. This ‘auto-annealing’ is seen in Fig. 1.a, which shows the variation of β with iterations of the Rosenbrock problem. It can be seen that β value sometimes decreases from one iteration to the next. This can never happen in any kind of ‘geometric annealing schedule’, $\beta \leftarrow k_\beta\beta$, $k_\beta > 1$, of the sort that is often used in most algorithms in the literature. In fact, we ran 50 trials of this algorithm on the Rosenbrock and then computed a best-fit geometric variation for β , that is, a nonlinear least squares fit to variation of β , and a linear least squares fit to the variation of $\log(\beta)$. These are shown in Figs. 1.c. and 1.d. As can be seen, neither is a very good fit. We then ran 50 trials of the algorithm with the fixed update rule obtained by best-fit to $\log(\beta)$, and found that the adaptive setting of β using cross-validation performed an order of magnitude

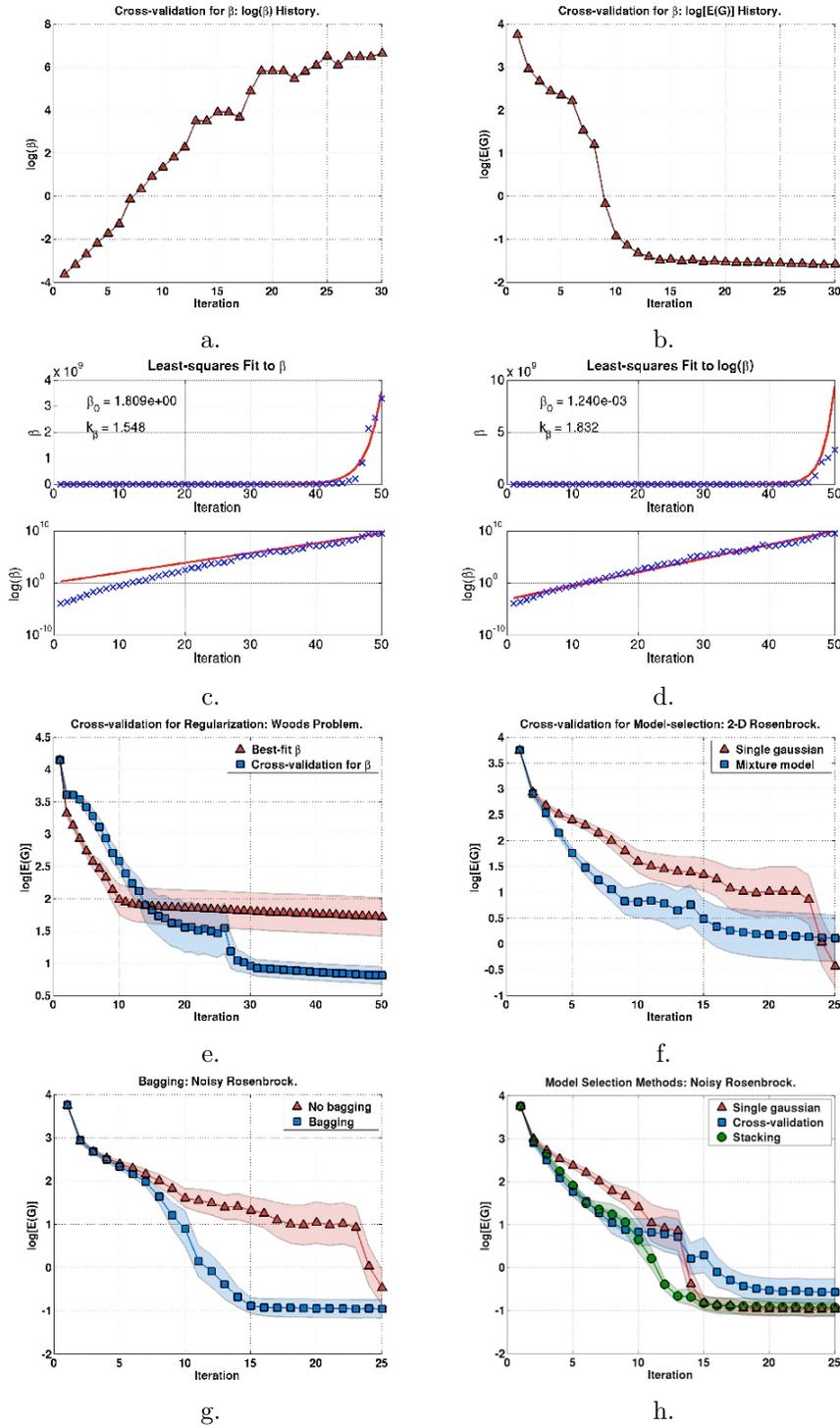


Figure 1: Various PL Techniques Improve MCO performance.

better, as shown in Fig. 1.e.

Cross-validation for Model Selection: Given a set Θ (sometimes called a model class) to choose θ from, we can find an optimal $\theta \in \Theta$. But how do we choose the set Θ ? In PL, this is done using cross-validation. We choose that set Θ such that $\arg \min_{\theta \in \Theta} \hat{F}(\theta)$ has the best held-out performance. As before, we use that model class Θ that yields the lowest estimate of $\mathbb{E}_{q_\theta}[G(x)]$ on the held-out data. We demonstrate the use of this PL technique for minimizing the Rosenbrock problem, which has a long curved valley that is poorly approximated by a single Gaussian. We use cross-validation to choose between a Gaussian mixture with up to 4 components. The improvement in performance is shown in Fig. 1.d.

Bagging: In bagging Breiman [1996a], we generate multiple data sets by resampling the given data set with replacement. These new data sets will, in general, contain replicates. We ‘train’ the learning algorithm on each of these resampled data sets, and average the results. In our case, we average the q_θ got by our KL divergence minimization on each data set. PC works even on stochastic objective functions, and on the noisy Rosenbrock, we implemented PC with bagging by resampling 10 times, and obtained significant performance gains, as seen in Fig. 1.g.

Stacking: In bagging, we combine estimates of the same learning algorithm on different data sets generated by resampling, whereas in stacking Breiman [1996b], Smyth and Wolpert [1999], we combine estimates of different learning algorithms on the same data set. These combined estimates are often better than any of the single estimates. In our case, we combine the q_θ obtained from our KL divergence minimization algorithm using multiple models Θ . Again, Fig. 1.h shows that cross-validation for model selection performs better than a single model, and stacking performs slightly better than cross-validation.

Conclusions

The conventional goal of reducing bias plus variance has interesting applications in a variety of fields. In straightforward applications, the bias-variance trade-offs can decrease the MSE of estimators, reduce the generalization error of learning algorithms, and so on. In this article, we described a novel application of bias-variance trade-offs: we placed bias-variance trade-offs in the context of Monte Carlo Optimization, and discussed the need for higher moments in the trade-off, such as a bias-variance-covariance trade-off. We also showed a way of applying just a bias-variance trade-off, as used in Parametric Learning, to improve the performance of Monte Carlo Optimization algorithms.

References

- D. H. Wolpert. On bias plus variance. *Neural Computation*, 9:1211–1244, 1997.
- David H. Wolpert and D. Rajnarayan. Parametric learning and monte carlo optimization. arXiv:0704.1274v1 [cs.LG], 2007.
- C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2004.
- G. P. Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27:192–203, 1978.
- Y. M. Ermoliev and V. I. Norkin. Monte carlo optimization and path dependent nonstationary laws of large numbers. Technical Report IR-98-009, International Institute for Applied Systems Analysis, March 1998.
- D. Angluin. Computational learning theory: Survey and selected bibliography. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing, May 1992*, 1992.
- V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 1982.

- V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
- W. Buntine and A. Weigend. Bayesian back-propagation. *Complex Systems*, 5:603–643, 1991.
- J. M. Berger. *Statistical Decision theory and Bayesian Analysis*. Springer-Verlag, 1985.
- D. Mackay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.
- R. Rubinstein and D. Kroese. *The Cross-Entropy Method*. Springer, 2004.
- D. H. Wolpert, C. E. M. Strauss, and D. Rajnarayan. Advances in distributed optimization using probability collectives. *Advances in Complex Systems*, 9(4):383–436, 2006.
- L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a.
- L. Breiman. Stacked regression. *Machine Learning*, 24(1):49–64, 1996b.
- P. Smyth and D. Wolpert. Linearly combining density estimators via stacking. *Machine Learning*, 36(1-2):59–83, 1999.