Parametric Learning for Blackbox Optimization

Anonymous Author(s)
Affiliation
Address
City, State/Province, Postal Code, Country
email

Abstract

The Probability Collectives (PC) framework for optimization transforms a given optimization problem into into a new optimization problem over a space of probability distributions. Recently, a version of PC was shown to be mathematically identical to the general Parametric Learning (PL) problem. Thus, techniques from PL like cross-validation, bagging, stacking and many others, can now be directly applied to PC, giving rise to a new application domain for PL, namely, optimization. We demonstrate the successful application of cross-validation for regularization in PC, and present experiments showing that the PL technique of stacking for combining models yields better optimizer performance than cross-validation to pick a single model. We also present experiments confirming that a variance-reduction technique for PC significantly improves its optimization performance.

1 Introduction

In many fields, sophisticated simulation techniques are used in conjunction with optimization algorithms to design physical systems. In particular, they are often used to estimate the quality of any design the optimization algorithm may propose. That estimate is then provided to the optimization algorithm, to help propose a new design candidate. These simulations are expensive and time-consuming, thus creating the need for efficient optimization techniques.

In this paper, we concentrate on blackbox optimization algorithms. These algorithms perform optimization by sending queries to an oracle, and getting back only function values (in particular, derivatives are *not* explicitly provided). In light of our concern about the expense of calls to the oracle, the performance of these algorithms is measured in terms of the number of function calls taken to find good solutions. There are several such techniques in existence today, ranging widely in the extent of formalism incorporated. The so-called 'derivative-free' methods include Genetic Algorithms (GAs) [1], Estimation of Distribution Algorithms (EDAs) [2, 3], Simulated Annealing (SA) [4], the Cross-Entropy (CE) method [5], and others. Methods that construct or approximate derivatives include various Response Surface Methods (RSM) [6], trust-region methods [7], and some forms of Sequential Quadratic Programming (SQP) [8, 9].

Probability Collectives (PC) is an optimization framework which transforms an optimization over variables x into a new problem of optimization over probability distributions $q(x)$ [10, 11, 12]. This transform is stochastically inverted, i.e., in order to find a solution x to the original problem, we simply sample the solution q of the transformed problem. This contrasts with conventional transform techniques, where the inversion is deterministic.

Some advantages of the PC approach are as follows. Data types of the underlying space are no longer a cause for concern; any data type is handled identically by the same mathematical machinery. Moreover, all the powerful techniques of optimization on Euclidean spaces, like Newton's method, can be directly applied even to problems with categorical variables or mixed data types. Also, for

suitable data types, probability distributions give sensitivity information about the original problem. Monte Carlo techniques can be leveraged to overcome problems of high dimensionality. Using probabilities, uncertainty, randomness, and also prior knowledge, can be formally incorporated. Finally, many older heuristic techniques can be formally shown to be special cases of PC.

It was recently realized that a version of PC is mathematically identical to the general Parametric Learning (PL) problem[1]. This means that any techniques for PL, including regularization, cross-validation, bagging, active learning, boosting, stacking, kernel machines, and others, can be easily applied to the field of blackbox optimization. This presents a new and useful application domain for PL. This paper briefly describes the PC technique, and subsequently demonstrates the successful use of PL techniques to significantly enhance the performance of PC algorithms.

The rest of the paper is laid out as follows. In section (2), we briefly overview some of the basic theory for PC, and describe the details of the procedures involved, with examples. In section (2.1), we describe a PC technique called immediate sampling[1], which is in sharp contrast to previous work in this area. Next, in section (3), we describe a particular blackbox optimization algorithm using immediate sampling, viz., a Kullback-Leibler distance minimization. In section (3.2) we review the application of the PL technique of cross-validation to PC, using two different held-out performance measures. In section (3.3), we describe an important contribution of this paper, the application of stacking to improve optimizer performance. Finally, in section (4), we present experimental results that show that using stacking to combine models outperforms the use of cross-validation for model selection. We also present results comparing the two different cross-validation approaches presented in section (3.2). Finally we show how a variance-reducing technique vastly improves performance of the algorithm. These evaluations are performed using two widely-known continuous optimization test functions, but in the presence of random noise.

2 Basic Theory for Probability Collectives

We consider oracles that take inputs $x \in X$, and return a potentially stochastic output g. Any such oracle $\mathscr{G}$ is therefore nothing but an x-indexed set of conditional probability distributions on $\mathbb{R}$. We do not know the oracle $\mathscr{G}$, and so $\mathscr{G}$ itself is a random variable. We denote each conditional distribution, the output of the oracle $\mathscr{G}$ for query x, as $P(g \mid x, \mathscr{G})$. Note that a for noise-free oracle corresponding to a single-valued function $G(x)$, this distribution is a delta function: $P(g \mid x, \mathscr{G}) = \delta(g - G(x))$. *Any* general unconstrained optimization problem can be written as

$$\text{P1:} \quad \text{argmin}_{x \in X} E[g|x, \mathscr{G}],$$

where $E_p[\cdot]$ denotes expectation under the distribution p. In PC, we transform (P1) to one over the space $\mathcal{Q}$ of parametric distributions q_θ, as follows.

$$\text{P2:} \quad \text{argmin}_{q_\theta \in \mathcal{Q}} \mathcal{F}_{\mathscr{G}}(q_\theta), \tag{1}$$

for some appropriate function $\mathcal{F}_{\mathscr{G}}$. After solving (P2), we then stochastically invert q_θ – we get an x by sampling q_θ. In many of the variants of PC explored to date, $\mathcal{F}_{\mathscr{G}}(q_\theta)$ is an integral transform[2] over X. Taking the parameterization to be implicit, we abbreviate $\mathcal{F}_{\mathscr{G}}(q_\theta)$ as just $\mathcal{F}_{\mathscr{G}}(\theta)$, and write this integral transform as

$$\mathcal{F}_{\mathscr{G}}(\theta) \quad \triangleq \quad \int dx\, dg\; P(g \mid x, \mathscr{G}) F(g, q_\theta(x)). \tag{2}$$

As an example of such a transform, consider optimization with a noise-free oracle (single-valued function), $P(g \mid x, \mathscr{G}) = \delta(g - G(x))$. Let $\mathcal{F}_{\mathscr{G}}(\theta)$ be the expectation value $E_{q_\theta}[G(x)]$. Then (P2) reduces to

$$\text{P3:} \quad \text{argmin}_{q_\theta \in \mathcal{Q}} \int dx\, q_\theta(x) G(x).$$

Further, let $\mathcal{Q}$ be the space of *all* probability distributions on X. To simplify notation in this general scenario, denote q_θ by q. Under reasonable weak conditions[3], the existence of solutions to (P3) is

[1] Journal paper under review

[2] In general, this need not be the case.

[3] X is countably finite or compact.

guaranteed, but uniqueness is not. If a solution exists, (P3) is a convex optimization problem that can be written as

$$\begin{array}{lll} \text{P4:} & \text{minimize} & \int dx\, q(x)G(x), \\ & \text{subject to} & q(x) \geq 0 \quad \forall \quad x, \\ & & \int dx\, q(x) = 1. \end{array}$$

We can use the conventional technique of barrier functions and Lagrange multipliers [13] to find a solution parametrized by a barrier parameter. If we use an entropic barrier function, the solution is computable closed-form, turns out to be a Boltzmann distribution parametrized by the inverse temperature β, and up to appropriate normalization, is given by

$$p^\beta(x) \propto \exp(-\beta G(x)). \tag{3}$$

The actual solution to the problem (P4) is given by $p^\star(x) = \lim_{\beta\to\infty} p^\beta(x)$. Usually, the analysis leading to equation (3) cannot be directly applied, and $\mathcal{Q}$ is actually some restricted set of parametrized distributions q_θ. In this case, we often take $\mathcal{F}_\mathscr{G}(\theta)$ to be some measure of the distance between q_θ and a target distribution like the Boltzmann distribution (3) derived above. In this paper, we consider an information-theoretic distance, a Kullback-Leibler divergence between q_θ and p^β given by

$$\mathcal{F}_\mathscr{G}(\theta) = \mathrm{KL}(p^\beta \| q_\theta) = \int dx\, p^\beta(x) \ln\left(\frac{p^\beta(x)}{q_\theta(x)}\right). \tag{4}$$

The minimization of this KL distance often turns out to be a convex program.

2.1 Immediate Sampling

Most PC techniques investigated to date have concentrated on pursuing analytic solutions to equation (1) as far as possible, and resorting to Monte Carlo techniques when no further progress is possible using algebra [10, 11, 12]. For instance, when $\mathcal{Q}$ is the space of product distributions, the problem (P2) with $\mathcal{F}_\mathscr{G}(\theta) = \mathrm{KL}(q_\theta \| p^\beta)$ can be solved using gradient descent, but the terms in the analytically computed gradient are conditional expectations. These expectations are then approximated using Monte Carlo sampling methods. This technique, where the Monte Carlo process is delayed as long as possible, is referred to as **delayed sampling**.

In contrast, we can construct sampled-data approximations of the integrals represented by $\mathcal{F}_\mathscr{G}$, and minimize these approximations. In other words, the Monte Carlo process is performed immediately, and the solution to this sampled problem is sought. This general technique for optimizing parametrized integrals is called Monte Carlo Optimization (MCO) [14]. We have shown [4] that MCO is mathematically identical to the general PL problem. When MCO is applied to PC, as here, it is called **immediate sampling**. We now show one way to construct these Monte Carlo approximations and perform the required minimizations.

Immediate sampling starts with the relatively simple technique of importance sampling [15] to approximate integrals. We use a sampling distribution $h(x)$ to generate points in X. We then form sample-pairs (x^i, g^i) by first sampling x according to $h(x)$ and then querying the oracle $\mathscr{G}$, which returns g according to $P(g \mid x, \mathscr{G})$. Then equation (5) below is an unbiased estimate of the integral in equation (2).

$$\begin{aligned} \mathcal{F}_\mathscr{G}(\theta) &\triangleq \int dx\, dg\; P(g \mid x, \mathscr{G}) F(g, q_\theta(x)), \\ &= \int dx\, dg\; h(x) P(g \mid x, \mathscr{G}) \frac{F(g, q_\theta(x))}{h(x)}, \\ &\approxeq \frac{1}{m} \sum_{i=1}^{m} \frac{F(g^i, q_\theta(x^i))}{h(x^i)}. \end{aligned} \tag{5}$$

We search for the q_θ that minimizes this sample mean, and then use some map $\eta\colon \mathcal{Q} \to \mathcal{Q}$ to update our sampling distribution[5] $h(x)$. In other words, the algorithm is iterative, in that we first sample from an initial sampling distribution h_1, perform a search over q_θ, suitably update the sampling distribution to $h_2(x) = \eta(h_1(x))$, and iterate.

[4] Journal paper under review.

[5] The simplest such map, the one that we use here, is $\eta(q_\theta) = q_\theta$.

Suppose that at some stage, we have a total of M sample pairs, with the m_k pairs $(x^{i,k}, g^{i,k})$ drawn from the sampling distribution $h_k(x)$, for $j = 1, \ldots, K$. We can reuse *all* these sample pairs in further computations, by simply considering the weighted average

$$\mathcal{F}_{\mathscr{G}}(\theta) \quad \approxeq \quad \sum_{k=1}^{K} \frac{v_k}{m_k} \sum_{i=1}^{m_k} \frac{F(g^{i,k}, q_\theta(x^{i,k}))}{h_k(x^{i,k})}. \tag{6}$$

The sum in equation (6) is nothing but a convex combination of the K estimates formed using samples from the corresponding sampling distributions h_k. Note that this is also an unbiased estimator of $\mathcal{F}_{\mathscr{G}}$. It is easy to show that the estimator with minimum variance, and hence minimum Mean Squared Error (MSE), is constructed by setting v_k to be inversely proportional to the variance of the k^{th} estimate.

3 Implementation for KL Distance

In this section we elaborate on the details of immediate sampling described in section (2.1), for minimizing KL distance. We use the KL distance given by equation (4), which we will call pq KL distance. We start at the point where we have formed m_k i.i.d. samples from K distributions h_k. Our transformed optimization problem at that point is

$$\text{minimize} \quad \sum_{k=1}^{K} \frac{v_k}{m_k} \sum_{i=1}^{m_k} p^\beta(x^{i,k}) \ln \left(\frac{p^\beta(x^{i,k})}{q_\theta(x^{i,k})} \right).$$

The solution to the above problem is unchanged when the objective function is multiplied throughout by a constant. Also, p^β is independent of q_θ. Consequently, the above problem is equivalent to

$$\text{minimize} \quad -\sum_{k=1}^{K} \sum_{i=1}^{m_k} r^{i,k} \ln[q_\theta(x^{i,k})], \tag{7}$$

where $r^{i,k} = \dfrac{v_k \exp(-\beta G(x^{i,k}))}{m_k h(x^{i,k})}$.

If q_θ is a log-concave density, the above optimization problem is a convex optimization problem that can be solved closed-form. For instance, in the case where q_θ is a multivariate Gaussian density, parametrized by $\theta = (\mu, \Sigma)$, the optimal parameters $\mu^\star$ and $\Sigma^\star$ are given by

$$\begin{aligned} \mu^\star &= \frac{\sum_{k=1}^{K} \sum_{i=1}^{m_k} r^{i,k} x^{i,k}}{\sum_{k=1}^{K} \sum_{i=1}^{m_k} r^{i,k}} \\ \Sigma^\star &= \frac{\sum_{k=1}^{K} \sum_{i=1}^{m_k} r^{i,k} (x^{i,k} - \mu^\star)(x^{i,k} - \mu^\star)^T}{\sum_{k=1}^{K} \sum_{i=1}^{m_k} r^{i,k}}. \end{aligned} \tag{8}$$

3.1 Mixture Models

It is quite straightforward to extend the foregoing from a single Gaussian q_θ to a mixture model of J Gaussians. The resulting optimization can be performed with an algorithm very similar to EM.

E-step: For each i, k, set $Q_{i,k}(z^{i,k}) = p(z^{i,k} | x^{i,k}; \mu, \Sigma, \phi)$,

i.e., $$\begin{aligned} w_j^{i,k} &= p(z^{i,k} = j | x^{i,k}; \mu, \Sigma, \phi), \\ &= \frac{\phi_j q_j(x^{i,k})}{\sum_{j'} \phi_{j'} q_{j'}(x^{i,k})}. \end{aligned}$$

M-step: Set

$$\begin{aligned} \mu_j &= \frac{\sum_{k=1}^{K} \sum_{i=1}^{m} w_j^{i,k} r^{i,k} x^{i,k}}{\sum_{k=1}^{K} \sum_{i=1}^{m} w_j^{i,k} r^{i,k}}, \\ \Sigma_j &= \frac{\sum_{k=1}^{K} \sum_{i=1}^{m} w_j^{i,k} r^{i,k} (x^{i,k} - \mu_j)(x^{i,k} - \mu_j)^T}{\sum_{k=1}^{K} \sum_{i=1}^{m} w_j^{i,k} r^{i,k}}, \\ \phi_j &= \frac{\sum_{k=1}^{K} \sum_{i=1}^{m} w_j^{i,k} r^{i,k}}{\sum_{k=1}^{K} \sum_{i=1}^{m} r^{i,k}}. \end{aligned} \tag{9}$$

The E-M steps are repeated till convergence of the parameters $\{\mu_j\}, \{\Sigma_j\}, \{\phi_j\}$. Since the problem is no longer convex, we run the EM algorithm from several random initial conditions.

3.2 Cross-validation for Regularization

Our ultimate object of interest in optimization is typically an expectation of $G(x)$ rather than a KL distance. One possible global cost function for PC is the expected cost $E_{q_\theta}[G(x)]$ (see [10, 11, 12]). In this context, minimizing KL distance from a Boltzmann distribution p^β can be loosely thought of as minimizing this global cost function in the presence of regularization, with the value of β controlling the amount of regularization (cf. equation 3). If we had infinitely many sample pairs (x^i, g^i), there would be no need for regularization, i.e., β should be infinite. On the other hand, when we have only a few samples, β needs to be small in order to avoid 'over-fitting'.

Traditionally, approaches like Simulated Annealing, and even earlier PC algorithms, had to specify what was called an 'annealing schedule', and the performance of these algorithms was often quite sensitive to these schedules. In light of the identity between immediate sampling and PL, one can use the technique of cross-validation to set the regularization parameter β.

In order to do this, we need an estimate of the objective $E_{q_\theta}[G(x)]$ using held-out samples, so we use the following estimator for $E_{q_\theta}[G(x)]$ (for details, see [15], *pp. 95.*)

$$\widehat{g}(q_\theta) = \frac{\displaystyle\sum_{k=1}^{K} \frac{v_k}{m_k} \sum_{i=1}^{m_k} \frac{q_\theta(x^{i,k}) G(x^{i,k})}{h_k(x^{i,k})}}{\displaystyle\sum_{k=1}^{K} \frac{v_k}{m_k} \sum_{i=1}^{m_k} \frac{q_\theta(x^{i,k})}{h_k(x^{i,k})}}. \tag{10}$$

Different values of β will produce different q_θ based on held-in samples. One can then compare the different values of β based on held-out performance according to equation (10). This is equivalent to having an 'adaptive annealing schedule'. Preliminary experiments indicate that it outperforms any fixed exponential annealing schedule (paper under review).

There are other possible choices for the "ultimate object of interest". One example is the probability mass placed by q_θ on the optimal set $X^\star = \{x \in X \mid G(x) \leq G(x') \text{ for any } x' \in X\}$, i.e., $E_{q_\theta} I\{x \in X^\star\}$, where $I\{\cdot\}$ denotes the indicator function. We can 'work our way' towards optimizing this objective by considering a sequence of indicator functions, (also called Heaviside functions) Θ_K given by

$$\Theta_K(x) = \begin{cases} 1, & \text{G(x)} \leq K, \\ 0, & \text{otherwise.} \end{cases}$$

We would like to maximize $E_{q_\theta}[\Theta_K(x)]$. The estimator for this expectation on held-out samples is simply given by

$$\widehat{g}(\theta) = \sum_{k=1}^{K} \frac{v_k}{m_k} \sum_{i=1}^{m_k} \frac{q_\theta(x^{i,k}) G(x^{i,k})}{h_k(x^{i,k})}. \tag{11}$$

In section (4), we compare these two methods and discuss their performance.

3.3 Combining Models Using Stacking

In addition to using cross-validation for regularization. we can also use it for model-selection. However, previous work in parametric learning [16, 17] has found that stacking, which uses held-out data to combine several predictors, typically outperforms cross-validation, which uses that data to choose one 'best' predictor. In [18], stacking is used in the context of density estimation, where one seeks the linear combination of density models that maximizes held-out log-likelihood. That is, given individual mixture models $q_{\theta_1}, \ldots, q_{\theta_n}$, one seeks a linear combination q (a mixture of mixtures)

$$q = \pi_1 q_{\theta_1} + \ldots + \pi_n q_{\theta_n}.$$

For PC, the ultimate object of interest is not held-out log-likelihood, but held-out $E_{q_\theta} G(x)$, which is linear in the q_{θ_i}. In this case, a linear combination of models reduces to picking the single best

model, exactly as in cross-validation. However, our estimates of $E_{q_\theta} G(x)$, as given by equations (10) and (11), are not exact, and therefore, we must account for our imperfect estimation. One way to do this is by regularization. We use an entropic regularizer $-TS(\pi)$ (where S is Shannon entropy), and seek to minimize $E_\pi \widehat{g}(\theta) - TS(\pi)$ on held-out data-sets, which gives

$$\pi_i^\star \propto \exp[-\frac{\widehat{g}(\theta_i)}{T}].$$

For simplicity, we use $T = 1/\beta$ in this paper, although one could think of better heuristics.

4 Experiments

In this section, we present three experiments on the use of PL techniques to improve immediate sampling blackbox optimization.

1. Comparison of cross-validation using $E_{q_\theta}[G(x)]$ and $E_{q_\theta}[\Theta_K(x)]$,
2. Comparison of variance-minimizing v_k, as described below equation (6), with uniform v_k.
3. Comparison of stacking to combine models and cross-validation to pick a single model.

In all experiments, we use cross-validation to adaptively set β.

Our experiments involve two well-known test functions for continuous unconstrained optimization, the Rosenbrock problem in two dimensions, given by

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

and the Woods problem in four dimensions, given by

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

In the case of the Rosenbrock problem, we add uniform random noise $v \sim \mathcal{U}([-0.25, 0.25])$, and for the Woods problem, we add $v \sim \mathcal{U}([-0.5, 0.5])$ to each function evaluation, demonstrating the ability of PC to handle random oracles. In all these experiments, the broad algorithm will be as shown in algorithm (1).

Algorithm 1 Overview of pq minimization using Gaussian mixtures.

1: Draw uniform random samples on an initial region of interest,
2: Compute $G(x)$ values for those samples,
3: **repeat**
4: Using cross-validation and equations (10) or (11), find regularization parameter β,
5: Using equations (8) and (9), find a mixture distribution q_θ to minimize KL($p^\beta \| q_\theta$),
6: Sample from q_θ,
7: Compute $G(x)$ for those samples.
8: **until** Termination
9: Sample final q_θ to get solution(s).

Note that in all experiments, at any given stage, all samples generated up to that point are used in the optimization. This is in contrast to most population-based algorithms, where only a few best samples are kept and used. All results are based on 50 independent runs of the PC algorithm.

For performance evaluation, the values of $E_{q_\theta}[G(x)]$ shown on the plots are computed using 1000 samples of $G(x)$ drawn from the corresponding q_θ. Also shown (shaded) are the 90% confidence intervals for the average across runs of $E_{q_\theta}[G(x)]$. Since the plots are logarithmic, the upper half of the confidence interval is simply reflected about the mean for purposes of visualization, and to avoid taking logarithms of negative numbers.

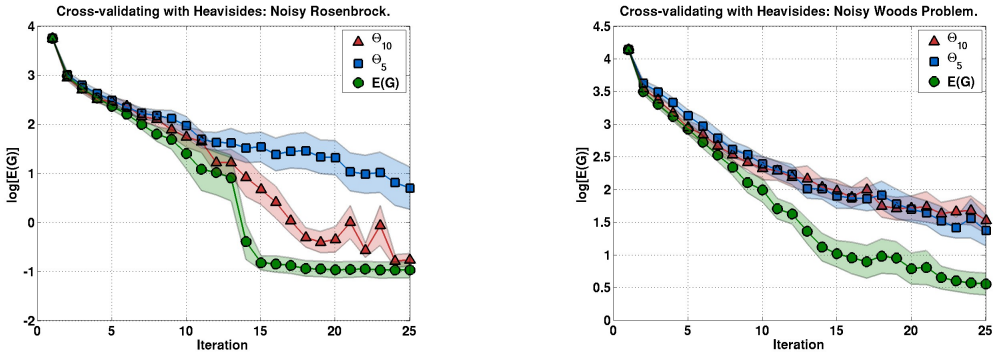
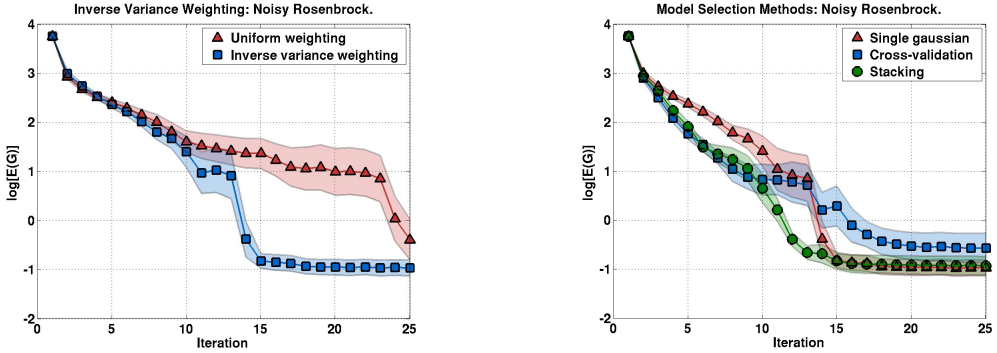


Figure 1: Cross-validation with Heaviside functions.



(a) Minimum variance KL distance estimator.

(b) Stacking outperforms cross-validation.

Figure 2: Inverse variance weighting and stacking.

4.1 Cross-validating with Heaviside Functions

To specify Θ_K, we need to choose the value of K. If $K > \min(G(x))$, the indicator function is everywhere zero, and does not give any information about $X^\star$. Therefore, it is crucial to set K properly. We attempt to set K using the best k-percentile of all samples so far. Shown in figure (1) are the results corresponding to $k = 10$ and 5 respectively, compared to using using $E_{q_\theta}[G(x)]$. It can be seen from these plots that it is difficult to choose a good value of k a priori. This is one of two problems associated with Heaviside functions. The other issue is that the Heaviside function treats alike all points that have $G(x) < K$. However, using $E_{q_\theta}[G(x)]$ ensures that points are treated proportionally to their $G(x)$ values, giving better optimization performance.

4.2 Minimum Variance Estimator

The choice of estimator for sampled KL distance has a significant impact on the performance of the optimization. Equation (6) tells us that we can use any convex weighted combination of the estimates from each sampling distribution. As described previously, the weights that minimize variance (and thus MSE) are inversely proportional to the variance of the estimates. We see from figure 2(a) that using this optimal weighting dramatically improves the performance of the optimization.

4.3 Stacking

As shown in figure 2(b), cross-validation outperforms using a single Gaussian in the early stages, but performs slightly worse in the end. The main reason for this is that the current algorithm does not yet incorporate shape regularization for the Gaussians in the mixture. This often leads to degenerate Gaussians that hurt subsequent performance of the importance sampler, although they give better $E_{q_\theta}[G(x)]$ at some early stages of the algorithm. Moreover, as mentioned earlier, the esti-

mates of $E_{q_\theta}[G(x)]$ are not exact. In fact, the estimator given by equation (10) has a small bias [15], in addition to variance. Stacking overcomes both these problems, by reducing the number of samples drawn from degenerate Gaussians, and also by accounting for the uncertainty in the held-out performance. It combines the early gains afforded by cross-validation, while avoiding some of the problems associated with cross-validation.

5 Conclusions and Future Work

In this paper, we reviewed the application of Parametric Learning (PL) techniques to the blackbox optimization approach called Probability Collectives (PC). A version of PC is a Monte Carlo Optimization (MCO) problem, which we have shown elsewhere to be mathematically identical to the generic PL problem. Consequently, one can apply any PL techniques, e.g., bagging, cross-validation, stacking, etc., to blackbox optimization. We show that combining models using stacking results in better optimization performance than using a single model selected using cross-validation. We also show that a Monte Carlo variance-reducing technique dramatically improves performance. Future work will include investigations of other PL techniques to blackbox optimization, like active learning, kernel methods, Gaussian processes, and many others.

References

[1] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[2] J.A. Lozano, P. Larraaga, I. Inza, and E. Bengoetxa. *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*. Springer Verlag.

[3] J.S. De Bonet, C.L. Isbell Jr., and P. Viola. Mimic: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems - 9*. MIT Press, 1997.

[4] S. Kirkpatrick, C. D. Jr Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[5] R. Rubinstein and D. Kroese. *The Cross-Entropy Method*. Springer, 2004.

[6] Raymond H. Myers and Douglas C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* J. Wiley, New York, 2002.

[7] A. Belegundu and R. Tirupathi. *Optimization Concepts and Applications in Engineering*. Prentice Hall, 1999.

[8] P.E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.

[9] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[10] D. H. Wolpert and S. Bieniawski. Distributed Control by Lagrangian Steepest Descent. pages 1562–1567, 2004.

[11] S. Bieniawski and D. H. Wolpert. Adaptive, distributed control of constrained multi-agent systems. In *Proc. of the Third Intl. Conf. on Autonomous Agents and Multiagent Systems*, 1230-1231.

[12] C. Fan Lee and D. H. Wolpert. Product distribution theory and semi-coordinate transformations. *Proc. of the Third Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 522–529, 2004.

[13] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[14] 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.

[15] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2004.

[16] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259.

[17] L. Breiman. Stacked regression. *Machine Learning*, 24, 1996.

[18] P. Smyth and D. Wolpert. Linearly combining density estimators via stacking. *Machine Learning*, 36(1-2):59–83, 1999.