Parameter tuning using asynchronous parallel pattern search in sparse signal reconstruction

Omar DeGuchy and Roummel F. Marcia

University of California, Merced, 5200 N. Lake Road, Merced, CA 95343 USA

ABSTRACT

Parameter tuning is an important but often overlooked step in signal recovery problems. For instance, the regularization parameter in compressed sensing dictates the sparsity of the approximate signal reconstruction. More recently, there has been evidence that non-convex ℓ_p quasi-norm minimization, where $0 < p < 1$, leads to an improvement in reconstruction over existing models that use convex regularization. However, these methods rely on good estimates of the value of not only p (the choice of norm) but also on the value of the penalty regularization parameter. This paper describes a method for choosing suitable parameters.The method involves creating a score to determine the effectiveness of the choice of parameters by partially reconstructing the signal. We then efficiently search through different combinations of parameters using a pattern search approach that exploits parallelism and asynchronicity to find the pair with the optimal score. We demonstrate the efficiency and accuracy of the proposed method through numerical experiments.

Keywords: Sparsity, signal processing, cross validation, non-convex optimization, Poisson log-likelihood, asynchronous parallel pattern search

1. INTRODUCTION

Parameter tuning in signal recovery problems involves judicious selection of parameters that influence not only the problem model but the solution as well. For example, in compressed sensing and in general sparse signal recovery problems, regularization parameters balance least-squares data fidelity terms with sparsity-promoting penalty terms. More recently, there has been evidence that non-convex ℓ_p quasi-norm minimization, where $0 < p < 1$, leads to an improvement in reconstruction over existing methods that use convex regularization. In this paper, we propose a computationally efficient method for parameter tuning in a parallel but asynchronous environment using derivative-free optimization. To demonstrate its effectiveness, we apply the proposed approach to existing methods for photon-limited imaging.

Related methods. The problem of estimating multiple parameters arises in many other fields. For example, kernel parameters of a support vector machine often have to be estimated.[1–3] Hyper-parameter tuning arises in stochastic gradient descent (SGD) methods for machine learning.[4–6] Automatic parameter selection methods have been previously proposed,[7–9] and pattern search methods are well-known in literature.[10–14] This paper explores *asynchronous* parallel derivative-free approaches for parameter tuning.

2. POISSON PROBLEM FORMULATION

In this section we will motivate the need for parameter tuning within the context of photon-limited imaging. Specifically, we will review the formulation of the Poisson reconstruction problem as the ℓ_p-norm penalized Poisson log-likelihood function. This will provide the bases for the SPIRAL-ℓ_p algorithm.[15]

Photon-limited imaging describes the modality whereby measurements at the detector are corrupted by Poisson noise. Under this regime the arrival of photons at the detector is typically modeled using a Poisson distribution and the vector of observed photon counts is given by the inhomogeneous Poisson process

$$\mathbf{y} \sim \text{Poisson}(\mathbf{A}\mathbf{f}^*),$$

Further author information:
Omar DeGuchy.: E-mail: odeguchy@ucmerced.edu
Roummel F. Marcia.: E-mail: rmarcia@ucmerced.edu, Telephone: 1-209-228-4874

where $\mathbf{y} \in \mathbb{Z}_+^m$, $\mathbf{f}^* \in \mathbb{R}_+^n$ is the true signal and $\mathbf{A}_+^{m\times n}$ is the system matrix projecting the true signal to the detector space.[16] Our interest is in the recovery of $\mathbf{f}^*$ and since the Poisson parameter is unknown, we use the maximum likelihood principle to maximize the probability of observing the vector y. Specifically, if $\mathbf{f}^*$ is known to contain a few non-zero entries, we can use a sparsity promoting penalty term to regularize the Poisson likelihood. The introduction of a penalty term such as the ℓ_1-norm[17] or the ℓ_p-norm[15] requires the tuning of parameters prior to optimization in order to control the severity of the sparsity.

The generalized ℓ_p-norm sparsity-promoting Poisson reconstruction problem can be written as the following constrained minimization problem:

$$\begin{aligned} \hat{\mathbf{f}} \quad = \quad & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad \Phi(\mathbf{f}) \equiv F(\mathbf{f}) + \tau\|\mathbf{f}\|_p^p \\ & \text{subject to} \ \ \mathbf{f} \succeq 0, \end{aligned} \tag{1}$$

where

$$F(\mathbf{f}) = \mathbb{1}^\top \mathbf{A}\mathbf{f} - \sum_{i=1}^{m} y_i \log(e_i^\top \mathbf{A}\mathbf{f} + \beta) \tag{2}$$

is the negative Poisson log-likelihood function, $\mathbb{1}$ is the m-vector of ones, e_i is the i-th column of the $m \times m$ identity matrix and $\beta > 0$ (typically $\beta \ll 1$). We take particular note of the penalty term in (1). Here the parameter $\tau > 0$ serves to control the weight of the penalty and $0 < p < 1$ acts as a bridge from the ℓ_0 counting norm to the convex ℓ_1-norm. The solution to (1) is computed from a sequence of quadratic models using a second-order Taylor approximation to $F(\mathbf{f})$ at each iteration.[15] However, the authors assume that p and τ are known *a priori*.

3. METHODOLOGY

3.1 Cross-validation

In order to determine the pair of parameters p and τ which return the optimal $\widehat{\mathbf{f}}$ that well approximates $\mathbf{f}^*$, it is necessary to define the criteria by which the optimal pair is chosen. This is difficult in the absence of information about the true signal $\mathbf{f}^*$. Typically the observation vector $\mathbf{y} \in \mathbb{R}^m$ and the system matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ are known elements of the problem. Our approach partitions $\mathbf{y}$ and $\mathbf{A}$ in the following manner:

$$\underset{m\times 1}{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_{\text{train}} \\ \mathbf{y}_{\text{test}} \end{bmatrix} \begin{matrix} \}r \\ \}m-r \end{matrix} \qquad \text{and} \qquad \underset{m\times n}{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{\text{train}} \\ \mathbf{A}_{\text{test}} \end{bmatrix} \begin{matrix} \}r \\ \}m-r \end{matrix}$$

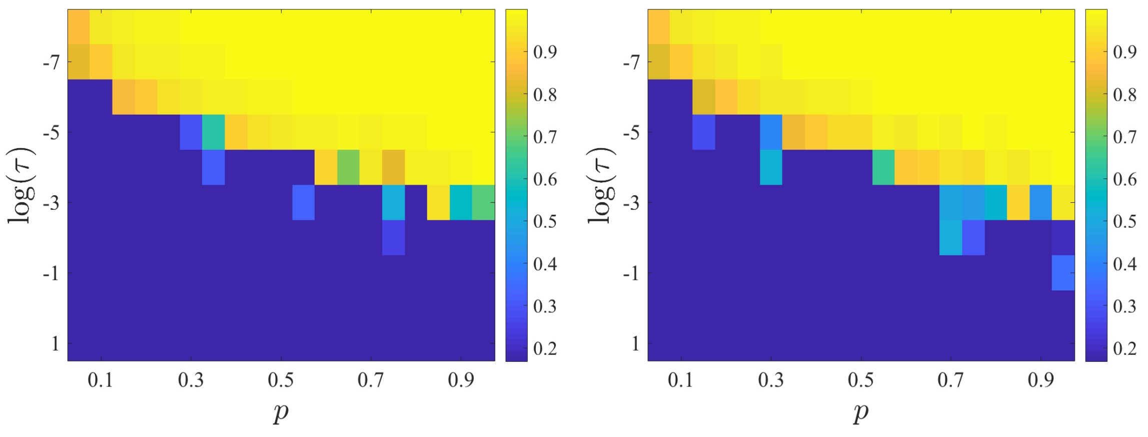
where $\mathbf{y}_{\text{train}} \in \mathbb{R}^r$, $\mathbf{y}_{\text{test}} \in \mathbb{R}^{m-r}$, $\mathbf{A}_{\text{train}} \in \mathbb{R}^{r\times n}$ and $\mathbf{A}_{\text{test}} \in \mathbb{R}^{(m-r)\times n}$, with $r < m$. For a candidate pair of parameters (p, τ), we use $\mathbf{y}_{\text{train}}$ and $\mathbf{A}_{\text{train}}$ to compute $\hat{\mathbf{f}}_{\text{test}} \in \mathbb{R}^n$ using the SPIRAL-ℓ_p algorithm. The signal reconstruction is then used to compute $\hat{\mathbf{y}}_{\text{test}} = \mathbf{A}_{\text{test}}\hat{\mathbf{f}}_{\text{test}}$. Finally, we compute

$$\text{RMSE}_{\text{test}} = \frac{\|\hat{\mathbf{y}}_{\text{test}} - \mathbf{y}_{\text{test}}\|_2}{\|\mathbf{y}_{\text{test}}\|_2}.$$

The choice of $\text{RMSE}_{\text{test}}$ as a metric for choosing the parameters p and τ is motivated by the fact that it behaves similarly to the $\text{RMSE}(\hat{\mathbf{f}})$ describing the difference between the true signal $\mathbf{f}^*$ and the reconstruction $\hat{\mathbf{f}}$ (see Fig. 1). We formulate the parameter search as the following constrained optimization problem:

$$\begin{aligned} \underset{p,\tau\in\mathbb{R}}{\text{minimize}} \quad & \text{RMSE}_{\text{test}}(p,\tau) \\ \text{subject to} \quad & 0 < p < 1, \\ & b_L \leq \tau \leq b_U, \end{aligned} \tag{3}$$

where $\text{RMSE}_{\text{test}}(p,\tau)$ is computed as previously stated for a given parameter pair and $0 < b_L < b_U$ are reasonable lower(b_L) and upper bounds(b_U) for the regularization parameter τ. This approach is based on cross-validation techniques used in statistical inference and machine learning to test how well a predictive model performs.[18]



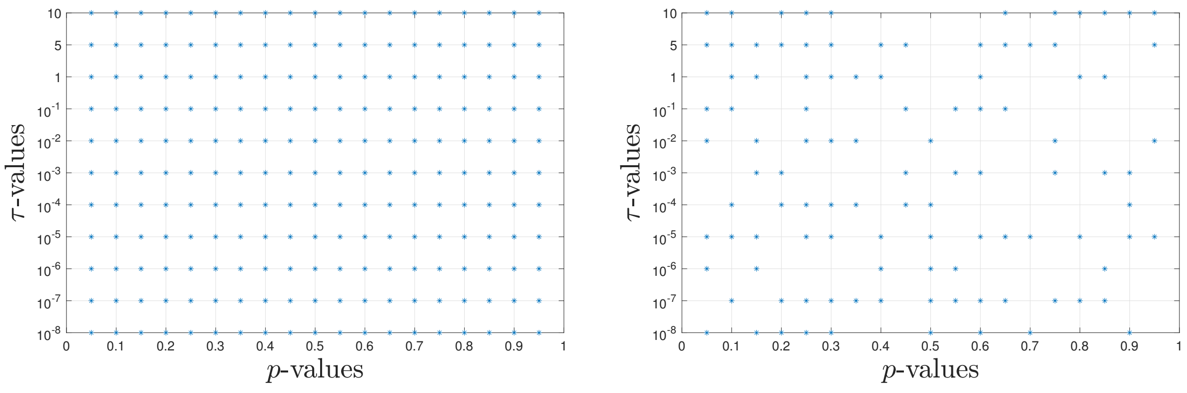
(a) (b)

Figure 1. A comparison of the behavior of (a) the RMSE for $\hat{\mathbf{y}}_{\text{test}}$ ($\text{RMSE}_{\text{test}}$) and (b) the RMSE for $\hat{\mathbf{f}}$ ($\text{RMSE}(\hat{\mathbf{f}})$) for various combinations of the parameters p and τ. Note the near-consistent agreement between $\text{RMSE}_{\text{test}}$ and $\text{RMSE}(\hat{\mathbf{f}})$.

Typically the data set used in these regimes is randomly divided into a test set and a validation set. In the Poisson reconstruction model we must maintain the structure of the data $\mathbf{y}$ limiting the random nature of traditional cross-validation. These techniques also involve input data used to train the model and output data used to test the model. Our problem uses the observation vector $\mathbf{y}$ as both the input data and the output data.

3.2 Asynchronous parallel pattern search

Solving (3) using fast gradient-based optimization techniques is not practical because how to compute (or even estimate) the gradient of $\text{RMSE}_{\text{test}}(p, \tau)$ is not clear. Instead, we propose using a derivative-free approach to minimizing the problem. Instead of an exhaustive grid-search approach or even standard derivative-free methods, we take advantage of the availability of multiple processors that *asynchornously* searches for the optimal parameters. Initially we create a coarse grid of possible values for p and τ.[19] As an alternative to computing $\text{RMSE}_{\text{test}}(p, \tau)$ for every possible combination on the grid (which can be computationally exhaustive) we randomly sample a percentage of combinations from the original grid (see Fig. 2) and compute $\text{RMSE}_{\text{test}}(p, \tau)$ for the resultant candidates.[20] We compare the $\text{RMSE}_{\text{test}}$ values and choose the pairs (p,τ) with the lowest associated values. The number of pairs chosen is based on the order of magnitude of the $\text{RMSE}_{\text{test}}$ with a minimum of 5 pairs and a maximum of 10. These candidates will be further refined as they form the initialization points for the HOPSPACK framework, which we explain next.



(a) (b)

Figure 2. Grid of potential parameter pairs (p,τ). (a) The full grid of 190 potential parameter values where $0 < p < 1$ and $10^{-8} \le \tau \le 10$. (b) The reduction of candidates by randomly choosing 50% of the initial pairs.

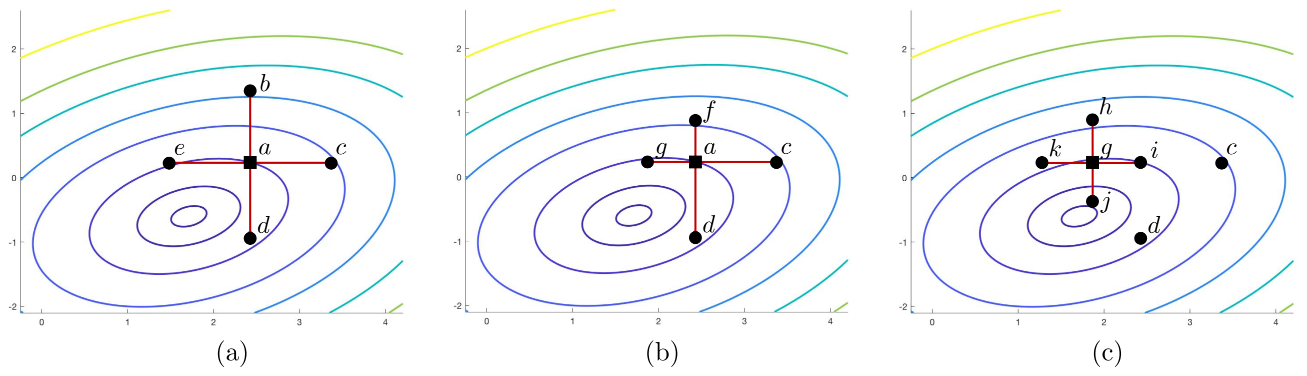


Figure 3. An illustration of an asynchronous parallel search implementation of the Generating Set Search method. (a) An initial (parent) point a and points $(\{b, c, d, e\})$ within a trial step along the standard basis directions are generated. (b). Suppose b and e required less time to evaluate than c and d and both b and e did not improve the objective function value at a, then the trial step is decreased and new trial points f and g are generated while c and d continue to be evaluated. (c) From the contour plots, the objective value at g is lower than at a; therefore g becomes the new parent and new trial points are generated based on the current trial step.

HOPSPACK (Hybrid Optimization Parallel Search PACKage)[21] is an open source frame work that enables users to implement derivative-free algorithms in parallel.[22–25] This paper takes advantage of its asynchronous pattern search implementation of GSS (Generating Set Search).[14] The GSS algorithm begins with an initial point and step size and generates trial points along the available axes in the negative and positive direction. The trial points are evaluated with two possible outcomes. If a trial point improves upon the value returned by the parent (current best point), then it becomes the parent and the process is reimplemented. If a trial point does worse, then the step size is decreased and a new trial point is determined. The algorithm terminates when the step size in all directions has reached a user defined step tolerance. (See Fig. 3 for an illustration.)

Implementing GSS with the HOPSPACK framework provides us with two advantages. The first is that the implementation is asynchronous, meaning that for a given iteration it does not wait for all of the trial point evaluations to be completed. HOPSPACK allows the algorithm to continue testing trial points with a partial set of results from the previous iteration. This characteristic makes the implementation ideal for parallelism and beneficial in the case where function evaluations in certain regions take longer. The latter is of significant interest when computing $\text{RMSE}_{\text{test}}$ as the SPIRAL-ℓ_p algorithm computational time may vary based on the choice of p and τ. The implementation will then continue to search for better trial points in the faster region while the slower region computes its value. The second advantage is that the asynchronous implementation of GSS inherits the properties of convergence of the synchronous implementation.[14]

Using the previously computed candidates from the randomized grid search as initial points in the HOPSPACK implementation of GSS, we further refine our choice of parameters until we have the optimal pair (p, τ). The pair is then used along with $\mathbf{y}$ and $\mathbf{A}$ in the SPIRAL-ℓ_p algorithm to compute the signal reconstruction $\hat{\mathbf{f}}$.

4. NUMERICAL EXPERIMENTS

In this section we implement the previously discussed method for finding the optimal parameter pair (p, τ). We perform 10 similar experiments where we generated measurements from 10 different signals and 10 different noise realizations for each signal. In each experiment, the true signal $\mathbf{f}^*$ has a length of 100,000 with 1,500 nonzero entries and a mean intensity of 15,000. The observation vector $\mathbf{y}$ has a length of 40,000 with a mean intensity much lower than that of $\mathbf{f}^*$ (see Fig. 4 for an example of one Poisson realization of one particular signal).

The goal of each experiment is to construct the best approximation of the true signal $\mathbf{f}^*$ from the observation vector $\mathbf{y}$. Specifically we seek to capture the support (number of non zeros and their location) of $\mathbf{f}^*$ without any prior information on $\mathbf{f}^*$. We initialize the method by creating the vector $\mathbf{y}_{\text{train}}$ using 90% of the length of $\mathbf{y}$. This results in the vector $\mathbf{y}_{\text{train}}$ having a length of 36,000 and $\mathbf{y}_{\text{test}}$ having a length of 4,000. The initial coarse grid was created using the bounds $0 < p < 1$ and $1.0 \times 10^{-8} < \tau < 1.0 \times 10^{3}$. The grid points of p values were uniformly

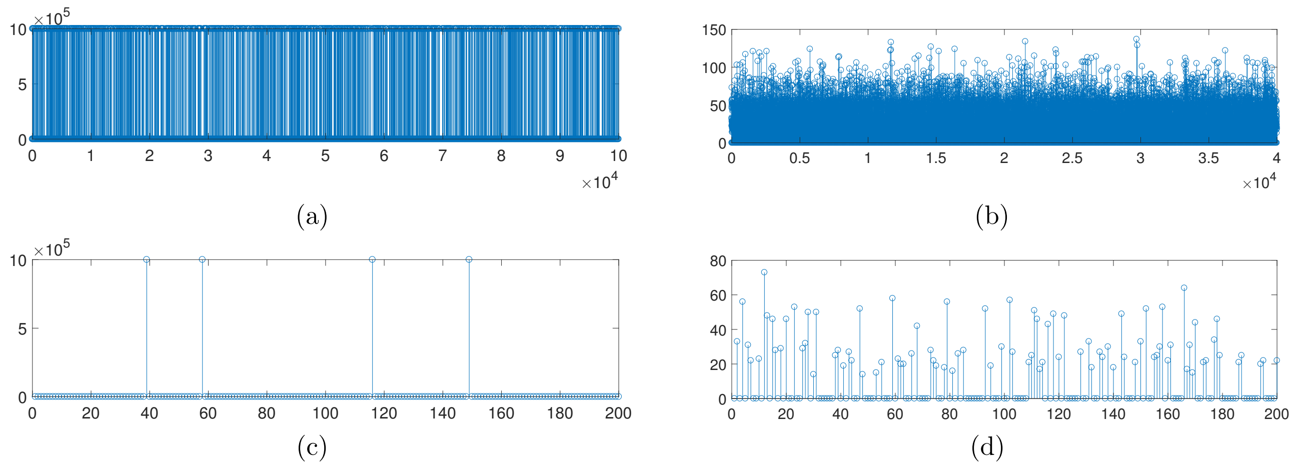


Figure 4. Experimental setup: (a) True signal $\mathbf{f}^*$ of size 100,000 with a mean intensity of 15,000. (b) Observation vector $\mathbf{y}$ of size 40,000 corrupted with Poisson noise. (c) The first 200 entries of $\mathbf{f}^*$. (d) The first 200 entries of $\mathbf{y}$.

spaced in increments of .05 while the τ values were logarithmically spaced. The result was a grid of 285 possible combinations. Fifty percent of the combinations were then randomly chosen. All computations were performed on the MERCED Cluster which consists of 84 compute nodes with a total of 1876 cores at 2301 MHz with a total capacity of approximately 60 TFLOPS. The $\text{RMSE}_{\text{test}}$ values were computed using the MATLAB implementation of the SPIRAL-ℓ_p algorithm. The best candidates from the coarse grid search were chosen by finding the pairs with lowest $\text{RMSE}_{\text{test}}$ values. The pairs with the same initial two significant digits were chosen resulting in a group of 5-10 candidates that were used to initialize the HOPSPACK framework. Searches were performed around each parameter pair by the HOPSPACK multi-threaded executable. Each computation was implemented on a cluster node across 20 cores. The function calls (computation of $\text{RMSE}_{\text{test}}$) were passed to MATLAB and the evaluations were parsed by the HOPSPACK framework. On average each node took approximately 10-20 minutes of CPU time to perform approximately 70-90 function evaluations before converging to a solution. Finally, the optimal pair from all nodes was determined and used to calculate $\hat{\mathbf{f}}$.

In all 100 experiments the true support or location of the nonzero elements (1500) were recovered. In some of the experiments slightly more nonzero elements than the support were also recovered. These extra entires are reflected in the averages shown in Table 1. In the most extreme case 1510 nonzero entries were recovered while most reconstructions captured the true support exactly. This consistency was exhibited across different realizations as well as across different signals. Table 1 also shows the range of average RMSE values for $\hat{\mathbf{f}}$. These values are consistent with those presented in literature[15] where the true signal $\mathbf{f}^*$ was used to tune the parameters. We also present in Table 1 the average computational times and the average number of pair (p, τ) evaluations per experiment.

Exp.	RMSE	Non-zeros	CPU	Evals.
1	0.0600	1501.0	842.48	164.91
2	0.0602	1500.4	595.03	144.42
3	0.0602	1500.4	708.59	167.74
4	0.0602	1501.7	483.54	92.48
5	0.0604	1501.7	484.32	138.31
6	0.0611	1501.7	382.20	116.71
7	0.0597	1501.1	397.94	108.62
8	0.0600	1500.6	433.11	121.29
9	0.0602	1500.5	315.77	96.01
10	0.0600	1500.9	326.40	100.22

Table 1. Average RMSE ($\|\mathbf{f}^* - \hat{\mathbf{f}}\|_2/\|\mathbf{f}^*\|$), number of non-zeros, CPU time (in seconds), and number of pairs (p, τ) evaluated for 10 different realizations of 10 different signals.

5. CONCLUSION

In this paper we created the metric $\text{RMSE}_{\text{test}}$ as a means for validating the parameter choice in the ℓ_p-norm Poisson reconstruction problem. The creation of $\text{RMSE}_{\text{test}}$ is particularly useful in that it assumes that nothing is known about the true signal $\mathbf{f}^*$. This measure of parameter choice coupled with the asynchronous parallelization of GSS implemented by HOPSPACK, accurately, efficiently and consistently produced the support of the true signal and RMSE values comparable to those produced by manual tuning where information about the true signal is known.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation Grant CMMI-1333326 and IIS-1741490. The authors gratefully acknowledge computing time on the Multi-Environment Computer for Exploration and Discovery (MERCED) cluster at UC Merced, which was funded by National Science Foundation Grant No. ACI-1429783.

REFERENCES

[1] Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S., "Choosing multiple parameters for support vector machines," *Machine Learning* **46**(1), 131–159 (2002).

[2] Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J., "The entire regularization path for the support vector machine," *Journal of Machine Learning Research* **5**(Oct), 1391–1415 (2004).

[3] Wu, K.-P. and Wang, S.-D., "Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space," *Pattern Recognition* **42**(5), 710–717 (2009).

[4] Duchi, J. C., Hazan, E., and Singer, Y., "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research* **12**, 2121–2159 (2011).

[5] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *CoRR* **abs/1412.6980** (2014).

[6] Zhang, S., Choromanska, A. E., and LeCun, Y., "Deep learning with elastic averaging SGD," in [*Advances in Neural Information Processing Systems*], 685–693 (2015).

[7] Kohavi, R. and John, G. H., "Automatic parameter selection by minimizing estimated error," in [*Machine Learning Proceedings 1995*], 304–312 (1995).

[8] Yu, L. and Liu, H., "Feature selection for high-dimensional data: A fast correlation-based filter solution," in [*Proceedings of the 20th International Conference on Machine Learning (ICML-03)*], 856–863 (2003).

[9] Loh, W.-Y., "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1**(1), 14–23 (2011).

[10] Hooke, R. and Jeeves, T. A., ""Direct Search" solutiton of numerical and statistical problems," *Journal of the ACM* **8.2**, 212–229 (2010).

[11] Nelder, J. A. and Mead, R., "A simplex method for function minimization," *The Computer Journal* **7.4**, 308–313 (1965).

[12] Dennis Jr, J. E. and Torczon, V., "Direct search methods on parallel machines.," *SIAM Journal on Optimization* **1.4**, 448–474 (1991).

[13] Torczon, V., "On the convergence of pattern search algorithms," *SIAM Journal on Optimization* **7**(1), 1–25 (1997).

[14] Kolda, T. G., Lewis, R. M., and Torczon, V., "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review* **45.3**, 385–482 (2003).

[15] Adhikari, L. and Marcia, R. F., "Nonconvex relaxation for poisson intensity reconstruction," in [*2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 1483–1487 (April 2015).

[16] Snyder, D. L. and Miller, M., [*Random Point Processes in Time and Space*], Springer (1991).

[17] Harmany, Z. T., Marcia, R. F., and Willett, R. M., "This is SPIRAL-TAP: Sparse Poisson intensity reconstruction algorithms; theory and practice," *IEEE Trans. on Image Processing* **21**(3), 1084–1096 (2012).

[18] Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection," in [*Proceedings of the 14th International Joint Conference on Articial Intelligence - Volume 2*], 1137–1143 (August 1995).

[19] Hsu, C., Chang, C., and Lin, C., "A practical guide to support vector classification," tech. rep., Department of Computer Science, National Taiwan University (2003).

[20] Bergstra, J. and Bengio, Y., "Random search for hyper-paramter optimization," *Journal of Machine Learning Research* **9**, 281–305 (2012).

[21] Plantenga, T. D., "HOPSPACK 2.0 User Manual," Tech. Rep. SAND2009-6265, Sandia National Laboratories, Albuquerque, NM and Livermore, CA (October 2009).

[22] Hough, P. D., Kolda, T. G., and Torczon, V. J., "Asynchronous parallel pattern search for nonlinear optimization," *SIAM Journal on Scientific Computing* **23**(1), 134–156 (2001).

[23] Kolda, T. G., "Revisiting asynchronous parallel pattern search for nonlinear optimization," *SIAM Journal on Optimization* **16**(2), 563–586 (2005).

[24] Gray, G. A. and Kolda, T. G., "Algorithm 856: Appspack 4.0: Asynchronous parallel pattern search for derivative-free optimization," *ACM Transactions on Mathematical Software (TOMS)* **32**(3), 485–507 (2006).

[25] Griffin, J. D., Kolda, T. G., and Lewis, R. M., "Asynchronous parallel generating set search for linearly-constrained optimization," *SIAM Journal on Scientific Computing* **30**, 1892–1924 (2008).