Journal of Machine Learning Research 7 (2006) 1265–1281 Submitted 7/06; Published 7/06

The Interplay of Optimization and Machine Learning Research

Kristin P. Bennett BENNEK@RPI.EDU
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12018, USA

Emilio Parrado-Hernández EMIPAR@TSC.UC3M.ES
Department of Signal Processing and Communications
University Carlos III de Madrid
Leganés (Madrid), 28911, Spain

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

The fields of machine learning and mathematical programming are increasingly intertwined. Optimization problems lie at the heart of most machine learning approaches. The Special Topic on Machine Learning and Large Scale Optimization examines this interplay. Machine learning researchers have embraced the advances in mathematical programming allowing new types of models to be pursued. The special topic includes models using quadratic, linear, second-order cone, semi-definite, and semi-infinite programs. We observe that the qualities of good optimization algorithms from the machine learning and optimization perspectives can be quite different. Mathematical programming puts a premium on accuracy, speed, and robustness. Since generalization is the bottom line in machine learning and training is normally done off-line, accuracy and small speed improvements are of little concern in machine learning. Machine learning prefers simpler algorithms that work in reasonable computational time for specific classes of problems. Reducing machine learning problems to well-explored mathematical programming classes with robust general purpose optimization codes allows machine learning researchers to rapidly develop new techniques. In turn, machine learning presents new challenges to mathematical programming. The special issue include papers from two primary themes: novel machine learning models and novel optimization approaches for existing models. Many papers blend both themes, making small changes in the underlying core mathematical program that enable the develop of effective new algorithms.

Keywords: machine learning, mathematical programming, convex optimization

1. Introduction

The special topic on "Large Scale Optimization and Machine Learning" focuses on the core optimization problems underlying machine learning algorithms. We seek to examine the interaction of state-of-the-art machine learning and mathematical programming, soliciting papers that either enhanced the scalability and efficiency of existing machine learning models or that promoted new uses of mathematical programming in machine learning. The special topic was an off-shoot of the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) Network of Excellence Workshop on "Machine Learning, SVMs and Large Scale Optimization", held in Thurnau, Germany from March 16 to 18, 2005.

Optimization lies at the heart of machine learning. Most machine learning problems reduce to optimization problems. Consider the machine learning analyst in action solving a problem for some set of data. The modeler formulates the problem by selecting an appropriate family of models and massages the data into a format amenable to modeling. Then the model is typically trained by solving a core optimization problem that optimizes the variables or parameters of the model with respect to the selected loss function and possibly some regularization function. In the process of model selection and validation, the core optimization problem may be solved many times. The research area of mathematical programming intersects with machine learning through these core optimization problems. On one hand, mathematical programming theory supplies a definition of what constitutes an optimal solution – the optimality conditions. On the other hand, mathematical programming algorithms equip machine learning researchers with tools for training large families of models.

In general, a mathematical program is a problem of the form

$$\begin{array}{ll} \min_{\mathbf{s}} & f(\mathbf{s}) \\ \text{subject to} & g(\mathbf{s}) \leq \mathbf{0} \\ & h(\mathbf{s}) = \mathbf{0} \\ & \mathbf{s} \in \Omega \end{array} . \tag{1}$$

The variables $\mathbf{s} \in \Omega$ are determined so as to minimize the objective function f possibly subject to inequality $g(\mathbf{s}) \leq \mathbf{0}$ and equality constraints $h(\mathbf{s}) = \mathbf{0}$. Examples of the set Ω include the n-dimensional real numbers, n-dimensional integers, and the set of positive semi-definite matrices. Convexity plays a key role in mathematical programming. Convex programs minimize convex optimization functions subject to convex constraints ensuring that every local minimum is always a global minimum. In general, convex problems are much more tractable algorithmically and theoretically. The complexity of nonconvex problems can grow enormously. General nonconvex programs are NP-hard. However, local solutions of such problems may be quite useful in machine learning problems, e.g. (Dempster et al., 1977; Bennett and Mangasarian, 1993; Bradley et al., 1997; Bradley and Mangasarian, 1998). Global optimization addresses the issue of nonconvex optimization. Integer or discrete optimization considers nonconvex problems with integer constraints.

A taxonomy of mathematical programs exists based on the types of objectives and constraints. There are now many flavors of mathematical programs: linear, quadratic, semi-definite, semi-infinite, integer, nonlinear, goal, geometric, fractional, etc. For example, linear programs have a linear objective and linear constraints. A more complete description of these problems can be obtained from the mathematical programming glossary (www.cudenver.edu/∼hgreenbe/glossary/) and the NEOS optimization guide (www-fp.mcs.anl.gov/otc/Guide/). Each flavor of mathematical program is a different research area in itself with extensive theory and algorithms. Very brief descriptions of the mathematical programs used in this special issue can be found in the Appendix. Good sources for theory and algorithms concerning nonlinear programming are (Nocedal and Wright, 1999), (Bertsekas, 2004), and (Bazaraa et al., 2006). An introduction to convex optimization including semi-definite programming can be found in (Boyd and Vandenberghe, 2004). Semi-infinite programming theory and algorithms are covered in (Goberna and López, 1998). Information about integer programming can be found in (Nemhauser and Wolsey, 1999).

We observe that the relationship between available mathematical programming models and machine learning models has been increasingly coupled. The adaptation of mathematical programming models and algorithms has helped machine learning research advance. Researchers in neural

networks went from backpropagation in (Rummelhart et al., 1986) to exploring the use of various unconstrained nonlinear programming techniques such as discussed in (Bishop, 1996). The fact that backpropagation worked well in turn stimulated mathematical programmers to work on stochastic gradient descent to better understand its properties, as in (Mangasarian and Solodov, 1994). With the advent of kernel methods (Cortes and Vapnik, 1995), mathematical programming terms such as quadratic program, Lagrange multipliers and duality are now very familiar to well-versed machine learning students. Machine learning researchers are designing novel models and methods to exploit more branches of the mathematical programming tree with a special emphasis on constrained convex optimization. The special topic reflects the diversity of mathematical programming models being employed in machine learning. We see how recent advances in mathematical programming have allowed rich new sets of machine learning models to be explored without initial worries about the underlying algorithm. In turn, machine learning has motivated advances in mathematical programming: the optimization problems arising from large scale machine learning and data mining far exceed the size of the problem typically reported in the mathematical programming literature.

This special topic investigates two majors themes in the interplay of machine learning (ML) and mathematical programming (MP).

The first theme contains the extension of well-known optimization methods to new learning models and paradigms. A wide range of convex programming methods is used to create novel models for problems such as uncertain and missing data, and hypothesis selection. Also, methods are developed for introducing constraints into the learning model in order to incorporate domain knowledge into graphical models and to enforce nonnegativity and sparsity in dimensionality reduction methods.

The second theme collects works aimed at solving existing machine learning models more efficiently. As data set size grows, off-the-shelf optimization algorithms become inadequate. Methods that exploit the properties of learning problems can outperform generic mathematical programming algorithms. Many of the included papers deal with well-known convex optimization problems present in ML tools such as the quadratic and linear programs at the core of the ubiquitous support vector machines (SVM) in either primal or dual forms. Tree re-weighted belief propagation is used to solve LP relaxations of large scale real-world belief nets. We see that the key to top performance is creating algorithms that exploit the structure of the problem and pay careful attention to algorithmic and numeric issues.

Many of the papers cross boundaries of both themes. They make small changes in the underlying models that enable the development of powerful new algorithms. Novel methods are developed for multi-kernel, ranking, graph-based clustering, and structured learning. The resulting algorithms decompose the problem into convex subproblems that can be more readily solved.

To summarize, in this special issue we see novel approaches to machine learning models that require solution of continuous optimization problems including: unconstrained, quadratic, linear, second-order cone, semi-definite, and semi-infinite convex programs. We first examine the interplay of machine learning and mathematical programming to understand the desirable properties of optimization methods used for training a machine learning model. We observe that the desirable properties of an optimization algorithm from a machine learning perspective can differ quite markedly from those typically seen in mathematical programming papers. Then we will examine the papers within and across the two themes and discuss how they contribute to the state of the art.

2. Interplay of Optimization and Machine Learning

The interplay of optimization and machine learning is complicated by the fact that machine learning mixes modeling and methods. In that respect, ML is much like operations research (OR). Mathematical programming/optimization is historically a subfield of OR. OR is concerned with modeling a system. Mathematical programming is concerned with analyzing and solving the model. Both OR and ML analysts address real world problems by formulating a model, deriving the core optimization problem, and using mathematical programming to solve it. According to (Radin, 1998) an OR analyst must trade off tractability – "the degree to which the model admits convenient analysis" and validity – "the degree to which inferences drawn from the model hold for real systems". So at a high level the OR and ML analysts face the same validity and tractability dilemmas and it is not surprising that both can exploit the same optimization toolbox.

In ML, generalization is the most essential property used to validate a novel approach. For a practical ML problem, the ML analyst might pick one or more families of learning models and an appropriate training loss/regularization function, and then search for an appropriate model that performs well according to some estimate of the generalization error based on the given training data. This search typically involves some combination of data preprocessing, optimization, and heuristics. Yet every stage of the process can introduce errors that can degrade the quality of the resulting inductive functions. We highlight three sources of such errors. The first source of error is the fact that the underlying true function and error distribution are unknown, thus any choice of data representation, model family and loss functions may not be suitable for the problem and thus introduce inappropriate bias. The second source of error stems from the fact that only a finite amount of (possibly noisy) data is available. Thus even if we pick appropriate loss functions, models, and out-of-sample estimates, the method may still yield inappropriate results. The third source of error stems from the difficulty of the search problem that underlies the given modeling problem. Reducing the problem to a convex optimization by appropriate choices of loss and constraints or relaxations can greatly help the search problem. Note that in many cases the ML models are made convex by an appropriate definition of the system boundaries that treats parameters as fixed. For example, ridge regression for a fixed ridge parameter is a convex unconstrained quadratic program. The generalized cross-validation method (Golub and von Matt, 1997) treats the ridge parameter as within the system boundary, and thus requires the solution of a nonconvex problem.

Consider tractability of a given model expressed as an optimization problem. Both ML and MP seek algorithms that efficiently compute "appropriate" solutions. The issues that make an algorithm more efficient – complexity, memory usage, etc. – are the same for both communities. But there is a large gap between what are considered an appropriate solutions in the two communities. In MP, "appropriate" solutions are the ones that solve the model with a high degree of accuracy as measured by the optimality conditions. As in ML, MP has large suites of benchmark problems. A benchmark study, typically would address both the speed of the algorithms as measured, for example, by performance profiles (Dolan and Moré, 2002). The quality of the solution would be measured by the objective value, a measure of the violation of the constraints, and a measure of the violations of the Karush-Kuhn-Tucker optimality conditions. Note that all of these metrics of solution quality are rarely reported in the ML literature. In MP, great care may be taken to make sure that solutions of equivalent accuracy are compared (see for example (Dolan and Moré, 2002)).

In ML, appropriateness is a much harder question due to the sources of modeling errors described above. A typical benchmarking study reports generalization errors and possibly compu-

tation times. Little or no attention is paid to how well the underlying optimization problem was solved by any of the metrics typically used in mathematical programming. Convergence tolerances are rarely reported and if they are, they typically are quite large (10^{-2} to 10^{-6}) relative to those seen in optimization papers. In machine learning the optimization problems being solved are only rough approximations of the real problem of finding a model that generalizes well. The ML modeler may change the problem formulation and algorithms, as long as generalization is not compromised. The papers in section 6 of this special topic illustrate how minor model reformulations can lead to significant improvements in algorithms. In general, it does not make much sense to require a ML model to converge to a high accuracy solution. When early stopping is used as a form of regularization, then the algorithm may never need to reach the solution. In this special topic (Keerthi et al., 2006) and (Taskar et al., 2006b) develop algorithms relying on early stopping and find that they offer advantages over alternative parametric approaches. Thus the desirable goal of a machine learning algorithm is to find a somewhat accurate solution efficiently. An optimization algorithm that has a poor asymptotic convergence rate may work quite well for ML. Ill-conditioning of the objective is typically viewed as a negative aspect of a model in MP, but ill-conditioning of the loss function and the resulting slow convergence of gradient methods may prevent overfitting. Thus not only is "good" optimization not necessary, but "bad" optimization algorithms can lead to better machine learning models.

In the ML community, Occam's razor appears to apply to algorithms as well; simpler algorithms are considered to be better. MP seeks robust optimization algorithms that find very accurate solutions to a broad class of functions with a premium for decreases in both theoretical complexity and empirical computation time. The emphasis is on solving the same size problems faster. This leads to complex algorithms. The effort to implement a simplex method for linear programming matching a state-of-art commercial solver such as CPLEX would be immense. The ML analyst's computational needs are different. An algorithm that solves the problem with good generalization in a reasonable amount of time is a good algorithm. Incremental speed increases are not so interesting. Simplicity of the algorithms is considered to be a significant plus. Scalability becomes a bigger issue as data set sizes grow. A general purpose solver is usually not the most scalable choice because it was designed to robustly solve a wide range of problems to high accuracy. However, the ML optimization can be tailored to exploit the structure of the optimization model. Robustness and ill-conditioning are not big issues since the algorithm need only be effective for a narrow class of functions and constraints and high accuracy solutions are frequently unnecessary.

To summarize, desirable properties of an optimization algorithm from the ML perspective are

- good generalization,
- scalability to large problems,
- good performance in practice in terms of execution times and memory requirements,
- simple and easy implementation of algorithm,
- exploitation of problem structure
- fast convergence to an approximate solution of model,
- robustness and numerical stability for class of machine learning models attempted,
- theoretically known convergence and complexity.

3. New Machine Learning Models Using Existing Optimization Methods

The special topic papers include novel machine learning models based on existing primarily convex programs such as linear, second order cone, and semi-definite programming. The reader unfamiliar with the basic convex programs can see their definitions in the Appendix. In these papers, the authors develop novel modeling approaches to uncertainty, hypothesis selection, incorporation of domain constraints, and graph clustering, and they use off-the-shelf optimization packages to solve the models.

3.1 Dealing with Uncertainty Using Second Order Cone Programming

The paper "Second Order Cone Programming Approaches for Handling Missing and Uncertain Data" (Shivaswamy et al., 2006) presents an extension to SVM that deals with situations where the observations are not complete or present uncertainty. The SVM Quadratic Program (QP) problem is cast into a more convenient Second Order Cone Program (SOCP) and uncertainty is represented as probabilistic constraints (SVM slack variables turn out to be random variables). They also come up with an interesting geometrical interpretation of their method as every data point being the center of an ellipsoid and the points within this ellipsoid being assigned to the class of the center. The study is extended to multiclass classification and regression.

3.2 Convex Models for Hypothesis Selection

Two papers address hypothesis selection. (Zhang et al., 2006) looks at pruning an ensemble of classifiers constructed from a pool of already trained classifiers. The goal is to make the performance of the smaller group equivalent to that of the whole pool, thus saving storage and computational resources. Traditionally, this selection process has been carried out using heuristics or by using greedy search. In (Bergkvist et al., 2006), the goal is to identify a small subset of hypotheses that exclude the true targets with a given error probability.

The first paper, "Ensemble Pruning Via Semi-Definite Programming" (Zhang et al., 2006), presents an optimization for pruning classification ensembles. The selection of the classifiers is based on a trade-off between their individual accuracies and the disparity of their predictions. This trade-off determines a quadratic integer program, i.e. a QP where the variables have to be integer numbers. The authors in (Zhang et al., 2006) propose a chain of transformations of the quadratic integer program towards a convex semi-definite program (SDP). Experimental results show that this approach beats the state-of-the-art greedy search methods. In addition, the scheme forms the basis of a powerful framework for sharing classifiers in a distributed learning environment, which enables the attack of large scale problems.

In the second paper, "Linear Programs for Hypotheses Selection in Probabilistic Inference Models", Bergkvist et al. (2006) introduce an LP for hypothesis selection in probabilistic inference problems motivated by a protein structure prediction problem. The model optimizes the expected weight of excluded hypotheses for a given error probability bound. The dual variables of the LP represent worst-case distributions of the hypotheses. The authors employ generic off-the-shelf LP optimizers but hypothesize that more efficient algorithms which exploit the problem structure may exist.

4. Models with Side Constraints

The next two papers look at traditional machine learning models with additional constraints.

Niculescu et al. (2006), in "Bayesian Network Learning with Parameter Constraints", use constraints to incorporate domain knowledge into Bayesian networks. The paper examines the cases for parameter sharing and conjugate constrained Dirichlet priors. They employ existing optimization algorithms to solve the resulting models. Addition of constraints improves generalization. Real-world results are presented for hidden process models applied to fMRI brain imaging. They formally prove that introducing constraints reduces variance.

Non-negative matrix factorisation (NMF) is a very attractive feature selection technique because it favors sparsity and data representations based on parts of the problem. However, it also poses a difficult nonconvex problem that is commonly solved via gradient descent. The paper "Learning Sparse Representations by Non-Negative Matrix Factorization and SCOP" (Heiler and Schnörr, 2006) presents an iterative algorithm to perform a sparse non-negative matrix factorization. They exploit the biconvex nature of Euclidean NMF and the reverse-convex structure of the corresponding sparsity constraints to derive an efficient optimization algorithm. This way, the strongly non-convex NMF is solved through the iterative application of a series of convex SOCP problems.

4.1 SDP Methods for Graph Clustering

The paper "Fast SDP Relaxations of Graph Cut Clustering, Transduction, and Other Combinatorial Problems" (De Bie and Cristianini, 2006) proposes an SDP relaxation to the normalized cut problem. The normalized cut problem arises when one wishes to partition a data set where similarity relationships among instances are defined. The mathematical formulation of this problem leads to an intractable combinatorial optimization problem. Spectral relaxation has been used to avoid this intractability. In spectral relaxation, the combinatorial optimization is cast onto a more simple eigendecomposition problem that gives the subsets of data. The new approach in (De Bie and Cristianini, 2006) consists of an SDP relaxation of the combinatorial problem that turns out to be tighter than the spectral one, although at the expenses of a larger computational burden. Moreover, they also present a scheme to develop a cascade of SDP relaxations that allows control of the trade-off between computational cost and accuracy. This study is extended to applications in semi-supervised learning.

5. Refining the Classics: Improvements in Algorithms for Widely Ssed Models

Widely used methods such as SVM and Bayesian networks have well-accepted core optimization problems and algorithms. The demand for the ability to learn with massive amounts of data is increasing. The immediate answer to this demand from the optimization and machine learning communities is to try to come up with more efficient implementations of these solid and reliable optimization methods.

5.1 Optimization Approaches for Dual SVMs

The SVM formulations for classification, regression, ranking, and novelty detection require the solutions of large dense QPs or LPs. These QP and LP problems were initially solved by general-purpose solvers. Now the demand for more scalable and easier to implement algorithms makes novel algorithms for SVMs an active and dynamic research area.

The primary challenges in solving the LP and QP arises from the linear inequality constraints. If the set of constraints that are active, i.e. satisfied at equality, are known then the problems reduce to

the solution of a set of linear equations. The inactive constraints have no effect on the final solution since they are satisfied as strict inequalities. Thus identification of the active constraints, or active set, represents a key step in LP and QP algorithms. One of the most common ways of solving these large QPs and LPs is to use some active set strategy. An active set strategy estimates the active set, solves the problem with respect to the estimated active set, uses the result to update the active set by adding and dropping constraints, and then repeats until an optimal solution is found. In SVMs, active set methods have a clear machine learning interpretation. For example in SVM classification, the active set in the primal corresponds to data points that are on the margin or in error. In the dual SVM formulation, there is a Lagrangian multiplier associated with each point. In the dual, the active set is determined by whether each Lagrangian multiplier is at bound or not.

The paper "An Efficient Implementation of an Active Set Method for SVMs" (Scheinberg, 2006) adapts "traditional" active set methods to the special structure of SVMs. Traditional active set methods were not thought to be tractable for large scale SVMs, but the paper concludes that they are competitive with popular methods such as SVMlight (Joachims, 1999). SVMlight is an example of a restricted active set method in which only a few variables are allowed to vary at each iteration. The restricted active set method in SVMLight decomposes the QP into subproblems, each identified by a group of variables that form an active set. Only the variables in the active set will be updated through the solution of the subproblem. These subproblems are solved until all the optimality conditions are met. These methods have the disadvantage of slow convergence when close to the optimal solution. The full active set in this paper avoids this problem. When full active sets are used, there is a corresponding speedup in the convergence of the optimization method. The paper provides a careful discussion of the details necessary for efficient implementation, active set selection, and warm starts (very valuable for cross-validation). The computational results find that the full active set method performs faster that SVMlight. This difference is most marked for higher accuracy solutions. The full active set method offers a speed scalability tradeoff, it performs faster that SVM-Light but may reach memory limitations sooner since it requires storage of a matrix of the size of the active set.

Reduced active set methods are taken to the extreme result in the popular sequential minimal optimization (SMO) method (Platt, 1999). In SMO, all variables except for a subset of two samples are fixed at each iteration. With the many subsets, the variable selection method becomes a key aspect in the convergence speed of the algorithm. The paper "Maximum-Gain Working Set Selection for SVMs" (Glasmachers and Igel, 2006) describes a new strategy to select the working set based on a greedy maximization of the progress in each single iteration. The algorithm uses precalculated information, which means no increment of the computational burden. The experiments show significant run time reductions over the broadly used SMO-based LIBSVM (Fan et al., 2005), so that full sets can be used, with a corresponding speedup in the convergence of the optimization method.

The paper "Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems" (Zanni et al., 2006) develops a multiprocessor solver for the standard SVM QP. Recent work in MP is used to develop a parallel gradient-projection-based decomposition technique for handling subproblems of moderate size. The subproblems and gradient calculations are done in parallel. Convergence results prove the algorithm converges to an optimal solution of the original QP. In practice, thanks to a large working set size, the algorithm converges in a few iterations. Details on how to fully exploit multiprocessors using strategies such as parallel kernel caching are provided. Results are reported for SVMs trained on millions of data points.

The paper "Incremental Support Vector Learning: Analysis, Implementation and Applications" (Laskov et al., 2006) aims at the software implementation of an efficient incremental learning algorithm for SVMs. The authors examine incremental SVM learning in two scenarios: active learning and limited resource learning. They propose a new storage strategy, simultaneously column-wise and row-wise, combined with a smarter organization of the computations for the minor iteration in terms of gaxpy-type matrix-vector products. This algorithm drops the training time of an incremental SVM by a factor of 5 to 20.

5.2 Optimization Approaches for Primal SVMs

In SVMs and other kernel methods, the computational cost of predicting a novel instance is directly related to the number of nonzero components or support vectors in the prediction function. Thus methods with a reduced number of support vectors are needed. One approach to doing this is to optimize the SVM in the primal form while directing invoking the representer theorem to allow generalization of kernels. In these direct or primal methods, the prediction function is assumed to consist of a linear combination of basis functions formed by the kernel. Prior work has established that sparse and reduced sparse or reduced complexity SVM functions can be achieved by either introducing one-norm regularization or by introducing early stopping strategies.

With respect to greedy construction methods, the paper "Building Support Vector Machines with Reduced Classifier Complexity" (Keerthi et al., 2006) contains a very efficient algorithm to develop a compact support vector machine. The efficiency of the method relies on both a primal method approach to the optimization and a cheap and accurate selection criterion for kernel basis functions. The experimental work presents a wide and systematic comparison with state-of-the-art column generation methods. This comparison points out the excellent capabilities of the algorithm in terms of compression in the number of basis functions, as well as a classification accuracy comparable to that of the full SVM.

Primal or direct kernel SVM models formulated with absolute value type losses and one-norm regularization produce LP core optimization problems. The one-norm enforces sparsity with the degree of sparsity controllable by a tradeoff parameter. Robust general purpose LP optimization tools that exploit advanced numerical analysis are available that can reliably and accurately solve massive problems. But these codes have several drawbacks from machine learning perspective: they are expensive to buy, they are complicated to implement, they do not exploit problem structure, and finally they are designed to find highly accurate solutions while in machine learning this may not be necessary. Thus alternative efficient and easy to implement LP algorithms for LP SVM type models are sorely needed.

The paper "Exact 1-Norm Support Vector Machines Via Unconstrained Convex Differentiable Minimization" (Mangasarian, 2006) introduces a Newton method for exactly solving the 1-norm SVM problem. It shows that the general LP problem can be recast as an unconstrained undifferentiable piecewise quadratic function using a dual exterior penalty function. Unlike prior penalty formulations like (Fung and Mangasarian, 2004), the penalty parameter is finite. The author introduces a generalized Newton method for solving the revised problem. The resulting algorithm outperforms CPLEX, a widely used commercial LP package.

5.3 LP Relaxations for Belief Propagation

The paper "Linear Programming Relaxations and Belief Propagation – An Empirical Study" (Yanover et al., 2006) introduces a very efficient method to find the most probable configuration of a graphical model by relaxing the corresponding integer program to a LP. The bad news is that the resulting LP has a large number of constraints and variables and it cannot be solved on desktop machines using commercial LP solvers. Fortunately the Tree-Reweighted Belief Propagation (TRBP) algorithm can be used to solve the LP. Results show that the special purpose TRBP LP solver outperforms CPLEX and can be used to solve large scale problems that are not tractable with CPLEX. The CPLEX model represents the graph as a matrix while TRBP directly represents the graph.

6. New Algorithms Starting from Reformulated Models

The special issue also illustrates how small reformulations of the model can yield much better algorithms. In the final four papers, we see how existing formulations are reformulated to admit new types of algorithms. In (Sonnenburg et al., 2006) and (Shalev-Shwartz and Singer, 2006), the revised formulations and novel algorithms can more effectively exploit special structure thus reducing the problem to a series of familiar, more easily solved problems.

6.1 Large Scale Multi-Kernel Learning Via Semi-Infinite Programming

The success of a kernel method is highly dependent on the choice of kernel. The multi-kernel learning (MKL) strategy is to consider a suite of kernels and let the algorithm decide on the choice of kernel. The paper "Large Scale Multiple Kernel Learning" (Sonnenburg et al., 2006) proposes a novel semi-infinite linear program (SILP) for the problem of learning with multiple kernels. Semi-infinite linear programs have a finite number of variables, a linear objective, and an infinite number of linear constraints. For SVM classification, the SILP solution is also optimal for the MKL quadratically constrained quadratic program in (Bach et al., 2004). The SILP is solved using a column generation method which alternates between solving a restricted master problem and an LP. The restricted master problem is solved using the corresponding off-the-shelf single kernel learning algorithms for the given loss. The LP is solved by a generic LP solver. The algorithm is very effective at seeking an approximate solution to the SILP. The authors show how the method can be extended to a variety of loss functions. Discussion of valuable details and variations of the algorithm needed for large scale problems are provided. Large scale results are achieved using parallel algorithms. They provide results for problems with up to 10 million data points and 20 kernels.

6.2 Better Ranking by Exploiting Structure

General purpose optimizers frontiers can be pushed forward in particular cases by exploiting problem structure. Often optimization problems can be cast onto simpler ones provided that the objective function follow certain structure. This is the case in the paper "Efficient Learning of Label Ranking by Soft Projection onto Polyhedra" (Shalev-Shwartz and Singer, 2006). The authors develop a fast and frugal algorithm for learning rankings by comparing the predicted graph with the feedback graph resulting in a QP with linear constraints. The algorithm decomposes the problem into a series of soft projections that can be efficiently solved using an iterative algorithm. The algorithm covers a large class of ranking and classification problems including multiclass and basic SVMs. It reduces to SOR in the classification case (Mangasarian and Musicant, 1999).

6.3 Max Margin Methods for Structured Output

Two of the papers tackled maximum margin methods for outputs defined on graphs by reformulating the problem and developing algorithms that could exploit the special structure. The first looks at hierarchical classification and the second looks at methods for more general structured data.

Maximum margin classification methods have focused on binary output problems. These methods have succesfully adapted to multicategory classification by analyzing the problem as a collection of binary problems (Rifkin and Klautau, 2004). However, emerging scenarios where the output is modeled by a vector demand a more careful analysis since their binarization involves (i) an exponential number of subproblems and (ii) the loss of the information encoded in the structured output. In this sense, (Taskar et al., 2006a) proposes an interesting combination of graphical models and maximum margin classifiers where the former allows use of the structured output information and the latter provides a reliable classification technology. Tractability from the optimization point of view is achieved through the grouping of the variables of the optimization problem into marginals defined by the graphical model.

The paper "Kernel-Based Learning of Hierarchical Multilabel Classification Models" (Rousu et al., 2006) provides a more efficient framework for scenarios where the vector output describes a hierarchical relationship. Their formulation requires the solution of a large scale quadratic program. This method's efficiency relies on a decomposition of the core problem into single variable subproblems and the use of a gradient-based approach. Moreover, the optimization is enhanced by a dynamic program that computes the best update directions in the feasible set.

The paper "Structured Prediction, Dual Extragradient and Bregman Projections" (Taskar et al., 2006b) proposes simple scalable maximimum margin algorithms for structured output models including Markov networks and combinatorial models. The problem is to take training data of instances labeled with desired structured outputs and a parametric scoring function and learn the parameters so that the highest scoring outputs match as closely as possible the desired outputs. Prior maximum margin approaches produced QP models (Taskar et al., 2005). By thinking of the problem one level up as a convex concave saddle point model, the authors can capitalize on the recent advances in optimization on extragradient methods (Nesterov, 2003). The extragradient approach produces a simple algorithm consisting of a gradient and projection step. For the class of models considered, the projection requires solution of dynamic program or network flow models for which very efficient algorithms exist. The method is regularized by early stopping. Interestingly the path of the extragradient algorithm corresponds closely to the parametric solution path of the regularized margin methods in their experiments. This demonstrates the interplay of the optimization algorithm and regularization: the path of the optimization algorithm is part of the regularization and there is no need to accurately solve the model.

7. Conclusion

Research in ML and research in MP have become increasingly coupled. ML researchers are making fuller use of the branches of the MP modeling tree. In this issue we see MP researchers using convex optimization methods including linear, nonlinear, saddle point, semi-infinite, second order cone, and semi-definite programming models. The availability of general MP models, along with robust general purpose solvers, provide tools for ML researchers to explore new ML problems. The resulting ML models challenge the capacity of general purpose solvers resulting in the development of novel special purpose algorithms that exploit problem structure. These special purpose solvers

do not necessarily possess the traits associated with good optimization algorithms. Tractability and scalability are valued in both ML and MP communities. Typically, MP demands that algorithms find high accuracy solutions and that they be robustness across wide classes of problems. In contrast, ML algorithm need to find good solutions to narrow classes of problems with special structure. Models may be reformulated to allow better algorithms provided that generalization is improved or at least not compromised. High accuracy is not required because of the inherent inaccuracies in the machine learning models and the fact that inaccurate solutions are deliberately sought as a form of regularization, for example as in early stopping. Also, ML puts more of a premium on algorithms that are easily implemented and understood at the expense of performance/complexity improvements that are typically studied in mathematical programming. In this special topic large scale problems were successfully tackled by methods that exploited both the novel MP models and their special structure and state-of-the-art MP methods. The special issue illustrates the many forms of convex programs that can be used in ML. But we expect the interplay of MP and ML will increase as more branches of the MP tree are incorporated into ML and the demands of large scale ML models exceed the capacity of existing solvers.

Acknowledgments

We would like to acknowledge support for this project from the IST Programme of the European Community under the PASCAL Network of Excellence IST2002-506788. We thank the authors contributing to this special topic for their helpful comments on this introduction. The efforts of the many anonymous reviewers of this special topic are also much appreciated.

Appendix: Standard Convex Programs

This section reviews the basic convex optimization mathematical programming models used in this special issue.

Quadratic Programming

Quadratic programming is used extensively in machine learning and statistics. The use of the least squares loss function in methods such as ridge regression and the 2-norm regularization in most support vector machine models both lead to quadratic programming models. A quadratic program (QP) has a quadratic objective with linear constraints.

Based on (Nocedal and Wright, 1999), we provide a brief review of quadratic programming and the reader can see (Nocedal and Wright, 1999) for more details. The general quadratic program can be stated as

$$\begin{array}{rll} \min_{\mathbf{s}} & \frac{1}{2}\mathbf{s}'\mathbf{Q}\mathbf{s}+\mathbf{c}'\mathbf{s} & \\ \text{subject to} & \mathbf{a}_i\mathbf{s}\leq \mathbf{b}_i & i\in \mathrm{I} \\ & \mathbf{a}_j\mathbf{s}=\mathbf{b}_j & j\in \varepsilon \end{array} \tag{2}$$

where the Hessian $\mathbf{Q}$ is a $n\times n$ symmetric matrix, I and ε are finite sets of indices and $\mathbf{a}_i$, $i\in \mathrm{I}\cup\varepsilon$ are $n\times 1$ vectors. If $\mathbf{Q}$ is positive semi-definite, i.e. $s'Qs\geq 0$ for any s, then the problem is convex. For convex QP, any local solution is also a global solution. A QP can always be solved or shown to be infeasible in a finite number of iterations. The following necessary and sufficient Karush-Kuhn-Tucker (KKT) optimality conditions of QP are formed with the use of Lagrangian multipliers α_i for

the inequality constraints and β_j for the equality constraints:

$$\begin{array}{lll} \text{Primal Feasibility} & \mathbf{a}_i'\mathbf{s} \leq \mathbf{b}_i & i \in \mathrm{I} \\ & \mathbf{a}_j'\mathbf{s} = \mathbf{b}_j & j \in \varepsilon \\ \text{Dual Feasibility} & \mathbf{Q}\mathbf{s} + \sum_{i \in \mathrm{I}} \mathbf{a}_i \alpha_i + \sum_{j \in \varepsilon} \mathbf{a}_j \beta_j = 0 & \\ & \alpha_i \geq 0 & i \in \mathrm{I} \\ \text{Complementarity} & \alpha_i(\mathbf{a}_i'\mathbf{s} - \mathbf{b}_j) = 0 & i \in \mathrm{I} \end{array} \tag{3}$$

Note that if there are no inequality constraints ($\mathrm{I} = \emptyset$), then a KKT point can be found by simply solving a system of linear equations.

Problems with inequality constraints represent more of a challenge. Two families of QP methods prevail: interior point methods and active-set methods. We focus on the latter since active set algorithms are a key component of this special topic. The optimal active set is the set of constraints satisfied as equalities at the optimal solutions. Active set methods work by making educated guesses as to the active set and solving the resulting equality constrained QP. If the guesses are wrong, the method uses gradient and Lagrangian multiplier information to determine constraints to add to or subtract from the active set.

Classical SVMs and the many subsequent variations require the solution of a QP problem.

Linear Programming

Linear programming optimizes a linear function subject to linear constraints. Since linear functions and constraints are convex, an LP is always a convex program. Linear programming can be thought of as a special case of the QP with the Hessian $\mathbf{Q}$ equal to 0. The general linear program can be stated as

$$\begin{array}{rll} \min_{\mathbf{s}} & \mathbf{c}'\mathbf{s} & \\ \text{subject to} & \mathbf{a}_i\mathbf{s} \leq \mathbf{b}_i & i \in \mathrm{I} \\ & \mathbf{a}_j\mathbf{s} = \mathbf{b}_j & j \in \varepsilon \end{array} \tag{4}$$

Interior point methods and simplex methods (active set methods) are both widely used within general purpose LP solvers.

Second-Order Cone Programming

The second-order cone program (SOCP) problems have a linear objective, second-order cone constraints, and possibly additional linear constraints:

$$\begin{array}{rll} \min_{\mathbf{s}} & \mathbf{c}'\mathbf{s} & \\ \text{subject to} & ||\mathbf{R}_i\mathbf{s} + \mathbf{d}_i||_2 \leq \mathbf{a}_i\mathbf{s} + \mathbf{b}_i & i \in \mathrm{C} \\ & \mathbf{a}_j\mathbf{s} = \mathbf{b}_j & j \in \varepsilon \end{array} \tag{5}$$

where $\mathbf{R}_i \in R^{n_i \times n}$ and $\mathbf{d}_i \in R^{n_i}$. Consult (Boyd and Vandenberghe, 2004) Chapter 4 for an introduction to SOCPs and their application to learning type problems. SOCPs are most often solved using interior point algorithms. See (Mittelmann, 2003) for a benchmark of general purpose SOCP algorithms.

Semidefinite Programming

Semidefinite programs (SDPs) are the generalization of linear programs to matrices. In standard

form an SDP minimizes a linear function of a matrix subject to linear equality constraints and a matrix nonnegativity constraint:

$$\begin{array}{rl} \min_{\mathbf{S}} & \langle \mathbf{C}, \mathbf{S} \rangle \\ \text{subject to} & \langle \mathbf{A}_i, \mathbf{S} \rangle = \mathbf{b}_i \quad i \in \mathrm{I} \\ & \mathbf{S} \succeq 0 \end{array} \tag{6}$$

where $\mathbf{S}$, $\mathbf{C}$, and $\mathbf{A}_i$ are in $R^{n \times n}$ and $\mathbf{b}_i \in R$. Here $\mathbf{S} \succeq 0$ means $\mathbf{S}$ must be positive semidefinite and $\langle \mathbf{C}, \mathbf{S} \rangle = trace(\mathbf{CS})$. SDPs are most commonly solved via interior programming methods. A comparison of SDP codes can be found in (Mittelmann, 2003).

Semi-infinite Programming

Semi-infinite linear programs (SILPs) are linear programs with infinitely many constraints. A SILP minimizes a linear objective subject to an infinite number of linear constraints:

$$\begin{array}{rll} \min_{\mathbf{s}} & \frac{1}{2}\mathbf{c}'\mathbf{s} & \\ \text{subject to} & \mathbf{as} \leq 0 & \mathbf{a} \in \mathcal{A} \\ & \mathbf{bs} = 0 & \mathbf{b} \in \mathcal{B} \end{array} \tag{7}$$

where $\mathcal{A}$ and $\mathcal{B}$ are sets (possibly infinite) of n vectors. Reviews of semi-infinite programming can be found in (Hettich and Kortanek, 1993) and (Reemtsen and Ruckmann, 1998), while the book (Goberna and López, 1998) gives extensive coverage of the topic.

References

F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear Programming Theory and Algorithms*. Wiley, 2006.

K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization & Applications*, 2:207–227, 1993.

A. Bergkvist, P. Damaschke, and M. Lüthi. Linear programs for hypotheses selection in probabilistic inference models. *Journal of Machine Learning Research*, 7:1339–1355, 2006.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Cambridge, 2004.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1996.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 368–374, Cambridge, MA, 1997. MIT Press.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T. De Bie and N. Cristianini. Fast SDP relaxations approaches of graph cut clustering, transduction and other combinatorial problems. *Journal of Machine Learning Research*, 7:1409–1436, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 5(Dec):1889–1918, 2005.

G. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

T. Glasmachers and C. Igel. Maximum-gain working set selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

M. A. Goberna and M. A. López. *Linear Semi-Infinite Optimization*. John Wiley, New York, 1998.

G. H. Golub and U. von Matt. Generalized cross-validation for large scale problems. *Journal of Computational and Graphical Statistics*, 6(1):1–34, 1997.

M. Heiler and C. Schnörr. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *Journal of Machine Learning Research*, 7:1385–1407, 2006.

R. Hettich and K. O. Kortanek. Semi-infinite programming: theory, methods and application. *SIAM Review*, 3:380–429, 1993.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169 –184. MIT Press, Cambridge, MA, 1999.

S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.

O. L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

O. L. Mangasarian and D. Musicant. Success overrelaxation for support vector machines. *IEEE Transaction on Neural Networks*, 10(5):1032–1037, 1999.

O. L. Mangasarian and M. V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4(2):103–116, 1994.

H.D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Mathematical Programming*, 95(2):407–430, 2003.

G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999.

Y. Nesterov. Dual extrapolation and its application to solving variational inequalities and related problems. Core, Catholic University of Louven, 2003.

R. S. Niculescu, T. M. Mitchell, and R. B. Rao. Bayesian network learning with parameter constraints. *Journal of Machine Learning Research*, 7:1357–1383, 2006.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

R. L. Radin. *Optimization in Operations Research*. Prentice-Hall, New Jersey, 1998.

R. Reemtsen and J. J. Ruckmann. *Semi-infinite programming*. Kluwer Academic, 1998.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5(Jan):101–141, 2004.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

D. Rummelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rummelhart and J. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, 1986. MIT Press.

K. Scheinberg. An efficient implementation of an active set method for SVMs. *Journal of Machine Learning Research*, 7:2237–2257, 2006.

S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7:1567–1599, 2006.

P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *International Conference on Machine Learning*, 2005.

B. Taskar, C. Guestrin, V. Chatalbashev, and D. Koller. Max-margin markov networks. *Journal of Machine Learning Research*, pages 1627–1653, 2006a.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006b.

C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation- an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

L. Zanni, T. Serafini, and G. Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7:1467–1492, 2006.

Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.

Academictorrents_collection. journal-of-machine-learning-research. Addeddate. 2020-08-11 06:25:12. External-identifier. urn:academictorrents:f59218767a5dade9760abec4d64dd1ef71bc7581. Identifier. academictorrents_f59218767a5dade9760abec4d64dd1ef71bc7581. Optimization is at the heart of almost all machine learning and statistical techniques used in data science. We discuss the core optimization frameworks behind the most popular machine learning/statistical modeling methods. Tirthajyoti Sarkar. Nov 10, 2018Â·7 min read.Â Introduction. Often, newcomers in data science (DS) and machine learning (ML) are advised to learn all they can on statistics and linear algebra. The utility of a strong foundation in those two subjects is beyond debate for a successful career in DS/ML. However, the topic of optimization, although less celebrated, is equally important to any serious practitioner of data science and analytics. Meta learning is an emerging research direction in machine learning. Roughly speaking, meta learning concerns learning how to learn, and focuses on the understanding and adaptation of the learning itself, instead of just completing a specific learning task. That is, a meta learner needs to be able to evaluate its own learning methods and adjust its own learning methods according to specific learning tasks. His research interests lie in optimisation and machine learning. At Oxford, he is a member of the OVAL group under Prof. Pawan Mudigonda. Research Pankajâ€™s current project is on the optimisation of convolutional neural network (CNN) architecture. The multi-layer topology of CNNs with hierarchical feature representation has been an important factor for their success in computer vision applications. However, there is still a lack of a principled approach to selecting the right architecture for a given problem. In this project, his research group attempts to present a new optimisation framework for simultaneously learning both the CNN structure and the network weights for a given problem. They aim to later extend the method to recurrent neural networks. #TuringShortTalks. Optimization for Machine Learning. Neural Information Processing Series. Michael I. Jordan and Thomas Dietterich, editors. Advances in Large Margin Classiers, Alexander J. Smola, Peter L. Bartlett, Bernhard SchoÂ¨lkopf, and Dale Schuurmans, eds., 2000. Advanced Mean Field Methods: Theory and Practice, Manfred Opper and David Saad, eds., 2001. Probabilistic Models of the Brain: Perception and Neural Function, Rajesh P. N. Rao, Bruno A. Olshausen, and Michael S. Lewicki, eds., 2002.Â Thus, SVMs oer excellent common ground on which to demonstrate the interplay of optimization and machine learning. 1.1 Support Vector Machines. 3. 1.1.1 Background. The problem is one of learning a classication function from a set of labeled. training examples.