Efficient Regularization Parameter Selection for Latent Variable Graphical Models via Bi-Level Optimization

Joachim Giesen[1] , **Frank Nussbaum**[1] and **Christopher Schneider**[2*]
[1]Friedrich-Schiller-Universität Jena
[2]Ernst-Abbe-Hochschule Jena
{joachim.giesen, frank.nussbaum}@uni-jena.de, christopher.schneider@eah-jena.de

Abstract

Latent variable graphical models are an extension of Gaussian graphical models that decompose the precision matrix into a sparse and a low-rank component. These models can be learned with theoretical guarantees from data via a semidefinite program. This program features two regularization terms, one for promoting sparsity and one for promoting a low rank. In practice, however, it is not straightforward to learn a good model since the model highly depends on the regularization parameters that control the relative weight of the loss function and the two regularization terms. Selecting good regularization parameters can be modeled as a bi-level optimization problem, where the upper level optimizes some form of generalization error and the lower level provides a description of the solution gamut. The solution gamut is the set of feasible solutions for all possible values of the regularization parameters. In practice, it is often not feasible to describe the solution gamut efficiently. Hence, algorithmic schemes for approximating solution gamuts have been devised. One such scheme is Benson's generic vector optimization algorithm that comes with approximation guarantees. So far Benson's algorithm has not been used in conjunction with semidefinite programs like the latent variable graphical Lasso. Here, we develop an adaptive variant of Benson's algorithm for the semidefinite case and show that it keeps the known approximation and run time guarantees. Furthermore, Benson's algorithm turns out to be practically more efficient for the latent variable graphical model than the existing solution gamut approximation scheme on a wide range of data sets.

1 Introduction

Multivariate Gaussians $\mathcal{N}(\mu, \Sigma)$ with mean μ and covariance matrix Σ are still among the most popular probabilistic models. The mean vector and the covariance matrix need to be estimated from data. The covariance matrix of an n-variate Gaussian has $\binom{n+1}{2}$ free parameters which is fairly large already for moderately high dimension n. This large number of parameters may lead to overfitting, or even worse, to non-regular estimates of the covariance matrix if there are fewer than n data points.

Factor analysis, developed by [Spearman, 1904], can be used to address the problems of overfitting and non-regular covariance matrices of standard multivariate Gaussians. In a Gaussian factor model the dimensions are subdivided into observed and a small number of latent dimensions. The joint density for all dimensions is assumed to be a multivariate Gaussian. Hence, also the marginals for the observed and latent dimensions, respectively, are multivariate Gaussians. The marginal for the latent dimensions is assumed to be $\mathcal{N}(0, \mathbb{1})$ and the marginal for the observed dimensions is assumed to be $\mathcal{N}(\mu, \Psi + \Gamma\Gamma^\mathsf{T})$, where Ψ is a diagonal matrix with non-zero diagonal entries and Γ is a rectangular matrix that maps from the latent to the observed dimensions. The covariance matrix $\Psi + \Gamma\Gamma^\mathsf{T}$ has only $n + kn$ free parameters, where n is now the number of observed and k the number of latent dimensions. The non-zero diagonal of Ψ ensures the regularity of the covariance matrix.

A different approach for dealing with the overfitting problem is the assumption of a *graphical model structure*. In the case of multivariate Gaussians a graphical model structure is defined by the zeros in the *inverse covariance matrix* (*precision matrix*). The structure, i.e., the precision matrix Λ, of a Gaussian graphical model can be learned consistently through the *graphical Lasso* [Yuan and Lin, 2007], which is given by the following optimization problem

$$\min_{\Lambda \succ 0} \quad \ell(\Lambda) + \lambda \left\|\Lambda\right\|_1 , \tag{GL}$$

where $\ell(\Lambda) = \operatorname{trace}\left(\Lambda\widehat{\Sigma}\right) - \log\det(\Lambda)$ is the negative log-likelihood with empirical covariance matrix $\widehat{\Sigma}$ and *regularization parameter* $\lambda > 0$ for the sparsity promoting ℓ_1-regularization term. The learned graphical model structure of course depends on λ that has to be chosen carefully.

[Chandrasekaran *et al.*, 2012] observed that it is possible to combine the ideas of Gaussian factor models and Gaussian graphical models. As in factor analysis, they assume that the observed dimensions along with a small number of latent dimensions are jointly multivariate Gaussian. The effect of the latent dimensions on the precision matrix of the

*Contact Author

observed dimensions can be computed from the Schur complement $\Lambda_o - \Lambda_{ol}\Lambda_l^{-1}\Lambda_{lo}$, where $\Lambda = \begin{pmatrix} \Lambda_o & \Lambda_{ol} \\ \Lambda_{lo} & \Lambda_l \end{pmatrix}$ is the precision matrix of the joint distribution. Hence, the precision matrix for the observed dimensions is the difference of the two matrices $S = \Lambda_o$ and $L = \Lambda_{ol}\Lambda_l^{-1}\Lambda_{lo}$, where L has *low rank*, if there are much fewer latent than observed dimensions. In the spirit of Gaussian graphical models, the matrix S is assumed to be *sparse*. Thus, the precision matrix of the observed dimensions is the difference of a sparse and a low-rank matrix. [Chandrasekaran *et al.*, 2012] show that these matrices can be learned consistently through the semidefinite program (*latent variable graphical Lasso*)

$$\begin{aligned} \min_{S,\,L} \quad & \ell(S-L) + \lambda \|S\|_1 + \rho \operatorname{trace}(L) \\ \text{s.t.} \quad & S - L \succ 0\,, \quad L \succeq 0\,. \end{aligned} \tag{LVGL}$$

Two examples for such a sparse + low-rank decomposition of the precision matrix can be seen in Figure 1.

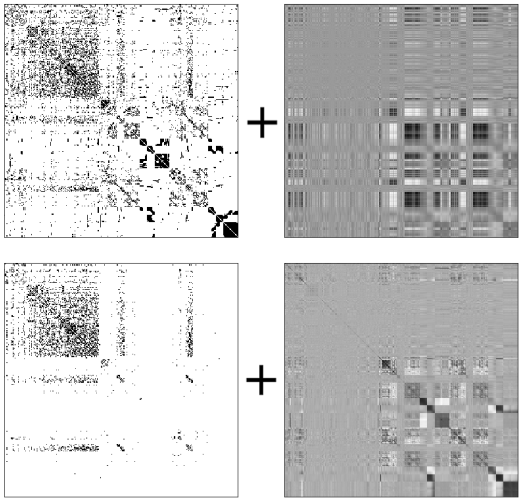

Figure 1: Two alternative sparse (left) + low-rank (right) decompositions of the precision matrix as learned by our bi-level optimization approach for the latent variable graphical Lasso with the `LSVT` data.

In a comment that was published together with the original work on latent variable graphical models it was pointed out by [Giraud and Tsybakov, 2012] that *"Proposing a reasonable data-driven selector for [. . . the regularization parameters λ and ρ . . .] would be very helpful for the practice"*. Choosing a good value for the regularization parameter is already an issue for the graphical Lasso since the statistical performance of the resulting model is highly dependent on this choice. The problem is aggravated for latent variable graphical models, where two instead of only one parameters need to be explored, compare Figure 1.

The problem of choosing a good value for a single regularization parameter has received a lot of attention, see for example [Giraud *et al.*, 2012] who study the special case of the graphical Lasso. The case of two or more regularization parameters has received less attention and is addressed rigorously only in [Blechschmidt *et al.*, 2015] and recently in [Giesen *et al.*, 2019].

Especially in the higher-dimensional case, i.e., for two or more regularization parameters, we found it very convenient to model the parameter-selection task as a *bi-level optimization problem* (Stackelberg game [von Stackelberg, 2010]). The two levels in such an approach are mostly referred to as *upper* and *lower level*, but are also known as *leader* and *follower* in game theoretic settings. Any solution or decision taken by the leaders on the upper level to optimize their goals is affected by the response of the followers on the lower level, who will seek to optimize their own outcomes. Here, on the upper level a measure of *generalization error* is minimized, for instance a cross validation error in a supervised setting or a likelihood value in an unsupervised setting. On the lower level the set of feasible solutions for all combinations of regularization parameters, the so called *solution gamut*, is described.

Typically, it is not feasible to provide a description of the *full solution gamut* and thus schemes for approximating it have been devised. [Blechschmidt *et al.*, 2015] provided the definition of an *approximate solution gamut*, devised an algorithm for computing it, and proved that the algorithmic complexity of their algorithm is asymptotically optimal. Recently, [Giesen *et al.*, 2019] revisited the problem and showed that *Benson's vector optimization algorithm* [Benson, 1998] can be adapted for approximating the solution gamut. So far, Benson's algorithm has been used only for regularized problems on standard vector domains but not for semidefinite problems.

Benson's algorithm avoids many of the practical issues of the original solution gamut approximation algorithm. These issues include choosing a sufficiently fine grid, evaluating the objective function on this grid, and solving dual problems next to the primal ones. The derivation of the dual problem is necessary for providing a stopping criterion in form of a small duality gap, in terms of a prescribed accuracy $\varepsilon > 0$, at every point of the solution gamut. Benson's algorithm uses a slightly different stopping criterion, but nevertheless comes with strong approximation guarantees. Here, we adapt and extend Benson's algorithm to the semidefinite programming setting and apply it to the regularization parameter selection problem for latent variable graphical models. It should be noted, however, that Benson's algorithm, in contrast to the traditional approach, can be used *out-of-the-box* for regularization parameter optimization for any other machine learning problem that is cast as a regularized semidefinite program like for instance [Candès *et al.*, 2011; Candès and Recht, 2009; Chen *et al.*, 2017; Chen *et al.*, 2011; Song *et al.*, 2007].

In Section 2, we provide the details on our bi-level optimization approach and our extension of Benson's algorithm. We also discuss how to implement the traditional solution gamut approximation algorithm for the latent variable graphical model. That includes deriving the dual of Problem (LVGL)—a maximum entropy problem that is interesting in itself. In Section 3, we experimentally compare Benson's algorithm to the original solution gamut approximation algorithm on a range of data sets from various domains. It turns out that Benson's algorithm that, in contrast to the original algorithm, can be used out-of-the-box is also more efficient in practice.

2 Regularization Parameter Selection by Bi-Level Optimization

By rescaling the objective of Problem (LVGL), one can see that the latent variable graphical Lasso is an optimization problem of type

$$\min_{X \in \mathcal{C}} \quad f_w(X) = w_0\, \ell(X) + \sum_{i=1}^{q} w_i\, r_i(X) \qquad (\mathrm{P}_w)$$

with convex feasible set $\mathcal{C} \subseteq \mathbb{R}^{n \times n}$. The functions ℓ and r_i, $i = 1, \ldots, q$, are assumed to be convex. The objective f_w is weighted by the regularization parameters $w = (w_0, w_1, \ldots, w_q) \in \mathcal{S} \subseteq \mathbb{R}^{q+1}$, where

$$\mathcal{S} = \left\{ w \in \mathbb{R}^{q+1} \;\middle|\; w_i \geq 0 \text{ for all } i\,,\ \textstyle\sum_{i=0}^{q} w_i = 1 \right\}$$

defines the q-dimensional standard simplex. By X^w, we denote a global optimal solution to Problem (P_w) for some given weight combination $w \in \mathcal{S}$.

Remark 2.1. Let us emphasize that next to constraints of type $g(X) \leq 0$ with some convex function g, in particular, also positive semidefinite matrix constraints of type $X \succeq 0$ are allowed for defining the feasible set $\mathcal{C}$. ◇

Choosing good weight parameters w is typically done in a two-step procedure:

(1) Solve (P_w) for "every" $w \geq 0$ on *training data*.

(2) Choose "the best" w, such that X^w minimizes some type of *generalization error* $\mathrm{GE}(w)$ on *validation data*.

From the viewpoint of optimization theory, Steps (1) and (2) can be combined and also be understood as a *bi-level optimization problem*

$$\min_{w \geq 0} \quad \mathrm{GE}(w) \quad \text{s.t. } X^w \in \arg\min(\mathrm{P}_w)\,. \qquad (\mathrm{BiP})$$

In this hierarchical optimization problem, the *upper level* aims for optimizing the objective GE with variable w. On the *lower level*, for each choice of w one has to find a solution X^w of Problem (P_w). It is known (cf. [Dempe, 2002]) that, in general, problems of type (BiP) are *non-convex*—even if (P_w) is a convex problem and GE is a convex objective. This can be observed practically in Figure 3, where several *local minima* for GE exist. Indeed, bi-level programming is known to be NP-hard [Hansen *et al.*, 1992].

For solving Problem (BiP), one has to deal with its feasible set, which is given by the set of solutions of the lower level problem (P_w), depending on the upper level variable w. Since, in most cases, it is not possible to give a closed formula for X^w, we use an approximation based on the definition of the *solution gamut* for Problem (P_w), cf. [Blechschmidt *et al.*, 2015].

Definition 2.2. Let $\varepsilon > 0$ be given. We call some function $\widehat{X}\colon \mathcal{S} \to \mathbb{R}^{n \times n}$ an *ε-approximative solution gamut* of Problem (P_w), if for all $w \in \mathcal{S}$

$$\widehat{X}(w) \in \mathcal{C} \quad \text{and} \quad f_w(\widehat{X}(w)) - f_w(X^w) \leq \varepsilon\,. \qquad \diamond$$

In the following, we will briefly present two methods for the computation of such an ε-approximative solution gamut. The first one is the algorithm presented in [Blechschmidt *et al.*, 2015], which we just call *solution gamut method*. The second one is a variant of *Benson's dual algorithm*, an established method from the area of *vector optimization*.

Since both algorithms yield a finite representation of the ε-approximative solution gamut $\widehat{X}$, minimization of the upper level objective GE just boils down to function evaluations.

2.1 Solution Gamut Method

The algorithm described in [Blechschmidt *et al.*, 2015, Sec. 4] computes a piecewise linear ε-approximative solution gamut for Problem (P_w). For that, the regularization parameter domain $\mathcal{S}$ is covered by a sufficiently fine grid. The spacing of this grid depends on the objective and the problem data, and thus is difficult to choose in practice. For the initialization, the method computes the optimal solution X^w to Problem (P_w) for $w = 0_{q+1}$, as well as the optimal solution to the dual problem of (P_w) which has to be derived first. Then, for every grid point, i.e., every weight $w \in \mathcal{S}$, the duality gap of this primal-dual solution pair is computed. The duality gap serves as upper bound to the optimality gap used in Def. 2.2. In the iterative process, the algorithm now chooses the grid point $\hat{w} \in \mathcal{S}$ with the maximal stored duality gap. Then, the primal and dual solutions for Problem (P_w) with $w = \hat{w}$ are computed, and the duality gaps are updated at all grid points, where the stored duality gap is larger than ε. The algorithm stops as soon as the duality gap is smaller than ε for every grid point $w \in \mathcal{S}$.

2.2 Adaptive Benson Algorithm

In a recent work [Giesen *et al.*, 2019], it turned out that the so-called *Benson algorithm* is well-suited for the computation of an ε-approximative solution gamut of Problem (P_w).

Vector Optimization

To apply Benson's algorithm, Problem (P_w) has to be studied in the context of *vector optimization*. Thereby, Problem (BiP) becomes a *semi-vectorial* bi-level optimization problem, where the scalar objective GE has to be minimized over the *Pareto set* of the following convex vector optimization problem

$$\begin{array}{ll} \min\limits_{X \in \mathcal{C}} & F(X) = \Big[\ell(X), r_1(X), \ldots, r_q(X)\Big] \\ \text{w.r.t.} & \leq_{\mathbb{R}^{q+1}_+}\ . \end{array} \qquad (\mathrm{VP})$$

The objective function $F\colon \mathbb{R}^n \to \mathbb{R}^{q+1}$ is vector-valued and minimized w.r.t. the component-wise partial ordering

$$y_1 \leq_{\mathbb{R}^{q+1}_+} y_2 \quad \text{if and only if} \quad y_2 - y_1 \in \mathbb{R}^{q+1}_+\,,$$

where $\mathbb{R}^{q+1}_+ = \{y \in \mathbb{R}^{q+1} \mid y_i \geq 0,\, i = 1, \ldots, q+1\}$. Problem (P_w) is the *weighted sum scalarization* of (VP).

In [Giesen *et al.*, 2019, Sec. 2] it has been observed that the *full solution gamut* (Def. 2.2 with $\varepsilon = 0$) coincides with the set of *weak minimizers* to Problem (VP).

Definition 2.3. A point $X^* \in \mathcal{C}$ is called a *weak minimizer* of Problem (VP) if $(\{F(X^*)\} - \operatorname{int} \mathbb{R}^{q+1}_+) \cap F(\mathcal{C}) = \emptyset$, where $F(\mathcal{C}) = \{F(X) \in \mathbb{R}^{q+1} \mid X \in \mathcal{C}\}$ is the image of the feasible set. ◇

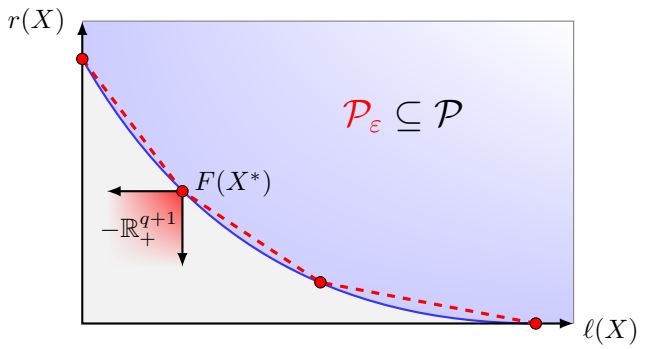

Figure 2: Upper image $\mathcal{P}$ with an inner polyhedral approximation $\mathcal{P}_\varepsilon$ which is obtained by computing a weak ε-solution (Def. 2.4). The red points are weak minimizers of (VP) (Def. 2.3).

Since, in general, the set of weak minimizers is infinite, we have to consider finite ε-solutions. Therefore, it is important to understand the geometry of Problem (VP)—especially its *upper image* defined by

$$\mathcal{P} = \operatorname{closure}(F(\mathcal{C}) + \mathbb{R}^{q+1}_+)\,.$$

The practical computation of an ε-approximation of $\mathcal{P}$ is motivated by the following definition.

Definition 2.4. Let $c \in \operatorname{int} \mathbb{R}^{q+1}_+$ be arbitrary but fixed and assume that Problem (VP) is bounded, i.e., $\mathcal{P} \subseteq \{y\} + \mathbb{R}^{q+1}_+$ holds for some $y \in \mathbb{R}^{q+1}$. Then a non-empty finite set $\mathcal{C}^* \subseteq \mathcal{C}$ is called an *ε-infimizer* if

$$\operatorname{conv} F(\mathcal{C}^*) + \mathbb{R}^{q+1}_+ - \varepsilon\, c \supseteq \mathcal{P}\,.$$

An ε-infimizer $\mathcal{C}^*$ of (VP) is called a *weak ε-solution* to (VP) if it only consists of weak minimizers. ◇

By setting $\mathcal{P}_\varepsilon = \operatorname{conv} F(\mathcal{C}^*) + \mathbb{R}^{q+1}_+$, such an ε-solution $\mathcal{C}^*$ provides an inner polyhedral approximation of the upper image $\mathcal{P}$ by finitely many minimizers, see Figure 2. Following the arguments in [Giesen *et al.*, 2019], a weak ε-solution to Problem (VP) coincides with an ε-approximative solution gamut for Problem (P_w).

Adaptive Benson Algorithm

For the computation of a weak ε-solution for Problem (VP), we develop an adaptive variant of Benson's dual algorithm. The class of dual Benson algorithms aims for approximating the upper image $\mathcal{P}$ by iteratively generating a growing sequence of inner approximating polyhedra until some prescribed accuracy is reached. Since polyhedra play a crucial role in this method, remember that they are either given in *H-representation* (intersection of half spaces) or in *V-representation* (set of vertices and directions). The conversion between both representations is done by *vertex* and *facet enumeration*, resp. [Bremner *et al.*, 1998].

While the standard Benson algorithm requires the choice of an approximation accuracy ε beforehand, we devise a variant which works adaptively, since the choice of ε cannot be done generically in practice. Our Algorithm 1 starts with a coarse accuracy $\varepsilon_0 > 0$ and calculates an ε_0-approximation $\mathcal{I}^{\varepsilon_0}$ of the upper image $\mathcal{P}$. Then the accuracy is successively refined by setting $\varepsilon_{i+1} = \frac{\varepsilon_i}{2}$ until the resulting approximation of $\mathcal{P}$ satisfies some stopping criterion. This leads to a growing sequence of inner approximation polyhedra given by

$$\mathcal{I}^{\varepsilon_0} \subseteq \mathcal{I}^{\varepsilon_1} \subseteq \ldots \subseteq \mathcal{I}^{\varepsilon_i} \subseteq \ldots \subseteq \mathcal{P}\,.$$

To calculate an (intermediate) inner approximation $\mathcal{I}^{\varepsilon_i}$, a hyperplane $w^\mathsf{T} x = b$ from the H-representation of the current inner approximation is moved outwards until it becomes a supporting hyperplane of the upper image $\mathcal{P}$. A contact point of this supporting hyperplane and $\mathcal{P}$ is given by $F(X^w)$, where X^w is the solution to the scalarized Problem (P_w). The distance the hyperplane is moved outwards in a given direction c is tracked and calculated by

$$d_c(X^w) = \frac{c^\mathsf{T} w}{\|c\|_2 \, \|w\|_2} \left(b - w^\mathsf{T} F(X^w)\right).$$

If $d_c(X^w) \geq \varepsilon_i$, the vertex $F(X^w)$ is added to the *working* V-representation followed by an update of the working H-representation, done by facet enumeration. If otherwise $d_c(X^w) < \varepsilon_i$, we continue by checking the next hyperplane. The procedure is repeated until the ε_i-approximation of $\mathcal{P}$ is complete. For the next iteration, we refine the accuracy ε_{i+1} and initialize the working V-representation with all vertices $F(X^w)$ where $d_c(X^w) \geq \varepsilon_{i+1}$, since the inner approximation in the neighborhood of those vertices may need further improvement.

The initialization of Algorithm 1 itself can be done with a V-representation that consists of a single point $F(X^{w_0})$ for some initial weight w_0. For instance, one can choose $w_0 = (\frac{1}{q+1}, \ldots, \frac{1}{q+1}) \in \mathbb{R}^{q+1}$ and set $\mathcal{I}_{\text{poi}} = \{F(X^{w_0})\}$.

Stopping criteria can be (i) attaining a target ε-approximation of $\mathcal{P}$, (ii) completing a maximum number of iterations, or (iii) improving the generalization error by a certain degree.

Algorithm 1 Adaptive Dual Benson Algorithm

Input: Problem data $(F, \mathcal{C})$, initialization $\mathcal{I}_{\text{poi}}$, chosen weights $T = \{w_0\}$, direction c, initial accuracy $\varepsilon = \varepsilon_0$
Output: V-representation $\mathcal{I}_{\text{poi}}$

function AdaptiveBensonAlgorithm
  repeat
    $\mathcal{I}^\varepsilon_{\text{poi}} \leftarrow \{F(X^w) \in \mathcal{I}_{\text{poi}} \mid d_c(X^w) \geq \varepsilon\}$
    compute H-representation $\mathcal{I}^\varepsilon$ of $\mathcal{I}^\varepsilon_{\text{poi}}$
    repeat
      choose $w \in \mathcal{I}^\varepsilon \setminus T$
      $X^w \leftarrow \arg\min(P_w)$
      if $d_c(X^w) > \varepsilon$ **then**
        $\mathcal{I}^\varepsilon_{\text{poi}} \leftarrow \mathcal{I}^\varepsilon_{\text{poi}} \cup \{F(X^w)\}$
        update H-representation $\mathcal{I}^\varepsilon$ of $\mathcal{I}^\varepsilon_{\text{poi}}$
      end if
      $T \leftarrow T \cup \{w\}$
    until $\mathcal{I} \setminus T = \emptyset$
    $\mathcal{I}_{\text{poi}} \leftarrow \mathcal{I}^\varepsilon_{\text{poi}} \cup \mathcal{I}_{\text{poi}}$
    $\varepsilon \leftarrow \varepsilon/2$
  until Stopping Criterion
end function

Complexity. Surprisingly, it seems that Benson's algorithm has never been used for semidefinite programs before. In [Löhne *et al.*, 2014], the authors only allow convex constraints for Problem (VP) defined by polyhedral cones. By adding semidefinite constraints, their proof of convergence still works, since by [Löhne *et al.*, 2014, Rem. 3 (Sec. 4.3)] one only has to assume that (i) $\operatorname{int} \mathcal{C} \neq \emptyset$ (Slater's condition) and (ii) Problem (P_w) has a solution for all $w \in \mathcal{S}$. Thereby, Algorithm 1 returns a weak ε-solution on termination. It is known from [Blechschmidt *et al.*, 2015] that the lower bound for the number of optimization problems that need to be solved for an ε-approximative solution gamut is $\Omega(\varepsilon^{-q/2})$. An upper bound for the complexity of Benson-type algorithms can be found in [Kamenev, 1994, Theorems 3 and 4]. This bound is given by $\mathcal{O}(\varepsilon^{-q})$ and $\mathcal{O}(\varepsilon^{-q/2})$, resp., where for the *sharp* second bound a twice continuously differentiable boundary of the upper image $\mathcal{P}$ is assumed.

3 Experiments

We perform regularization parameter selection for Problem (LVGL) from the introduction using the following data sets [Tsanas *et al.*, 2014; Higuera *et al.*, 2015; Zhou *et al.*, 2014; Dua and Karra Taniskidou, 2017]

- `GENE1` with $n = 100$ features and $m = 255$ samples,
- `TCGA` with $n = 500$ features and $m = 801$ samples,
- `MICE` with $n = 81$ features and $m = 552$ samples,
- `ROSETTA` with $n = 100$ features and $m = 301$ samples,
- `SONAR` with $n = 60$ features and $m = 208$ samples,
- `OR70` with $n = 70$ features and $m = 1059$ samples,
- `LSVT` with $n = 310$ features and $m = 126$ samples,
- `S&P500` with $n = 471$ features and $m = 60$ samples.

These data sets cover a wide range of applications. `GENE1`, `TCGA`, and `MICE` are biological data sets, `ROSETTA` and `SONAR` are geological data sets, `OR70` was recorded for investigating the geographical origins of music, `LSVT` is about voice rehabilitation in psychology, and `S&P500` includes monthly stock return data from major US companies over the course of 5 years. From `S&P500` we removed 29 companies because their data was incomplete. From the original data sets `GENE1`, `ROSETTA`, and `TCGA` we selected the n features with the highest variance, similarly as [Chandrasekaran *et al.*, 2012] who also used only subsets of `GENE1` and `ROSETTA`.

The Dual of Problem (LVGL). The solution gamut method requires the dual of Problem (LVGL) that we derive here. [Dudík *et al.*, 2004] have developed a fairly general duality theory of discrete *maximum likelihood* and *maximum entropy* problems. In this theory, regularization terms on the maximum likelihood side translate themselves into relaxations of moment-matching constraints on the maximum entropy side. The theory can be extended to the continuous case that we need here, if *Shannon entropy* is replaced by *differential entropy*. In our case, it turns out that the maximum entropy solution is a multivariate Gaussian with zero mean and covariance matrix Σ. It follows that the objective function of the dual of Problem (LVGL) is $\log\det(\Sigma)$ since the entropy of a multivariate Gaussian is proportional to the log determinant of its covariance matrix. The regularization terms of Problem (LVGL) become relaxations of the moment-matching constraint $\Sigma = \widehat{\Sigma}$ in the dual maximum entropy problem, which reads as

$$\max_{\Sigma \succ 0} \ \log\det(\Sigma) \quad \text{s.t.} \ \left\|\widehat{\Sigma} - \Sigma\right\|_\infty \leq \lambda\,, \quad \widehat{\Sigma} - \Sigma \preceq \rho\,\mathrm{Id}\ .$$

Data Set	Adaptive Benson		Solution Gamut	
	# Opt.	Time [s]	# Opt.	Time [s]
`GENE1`	15	3.9	36	53.8
`TCGA`	13	91.3	38	842.8
`MICE`	10	1.6	64	55.1
`ROSETTA`	10	2.7	30	36.8
`SONAR`	5	1.0	44	19.2
`OR70`	4	0.7	12	9.3
`LSVT`	5	25.0	20	159.5
`S&P500`	13	127.0	90	1673.5

Table 1: Number of solved optimization problems and CPU time for computing a solution within 1% of the optimum.

Experimental Setup

Any algorithm for approximating the solution gamut needs to guarantee a good solution in terms of the generalization error GE while being computationally efficient. In our experiments we study both aspects, the quality of the solutions and the computational efficiency of both approximation algorithms, the solution gamut method and our adaptive Benson algorithm. For solving the optimization problems, we adapted the ADMM-based algorithm discussed in [Ma *et al.*, 2013] and used CDD [Bremner *et al.*, 1998] for facet enumeration. All experiments were run on a Linux machine with an Intel Core i5-2500K (4×3.30 GHz) CPU and 16 GB RAM.

Solution quality. We prepared the data sets by splitting them into training and validation data in a 2:1 ratio. We also centralized and standardized the data using empirical means and standard deviations of the training data. The generalization error, here the negative log-likelihood function value, is computed on the validation data.

For the experiments, we reformulated Problem (LVGL) such that it conforms to the theory in Section 2, where weights are chosen from a standard simplex.

As a baseline, we used grid search on a fine grid with more than 5000 points. At the grid points we solved Problem (LVGL) and computed the corresponding generalization error, see Figure 3. The interval between the worst and best solution, in terms of generalization error, serves as our estimate of the range of possible generalization errors. We consider a solution as good, if it is close to the best solution. We measure closeness as the normalized distance to the optimum, where the normalization factor is the length of the interval of possible generalization errors. Then, we let both the solution gamut method and the adaptive Benson algorithm run until their solutions were within 1% distance to the optimum from the grid search. On all data sets, we started Benson's algorithm with $\varepsilon_0 = 2^{10}$ and fixed direction $c = (1, 1, 1)^\mathsf{T}$.

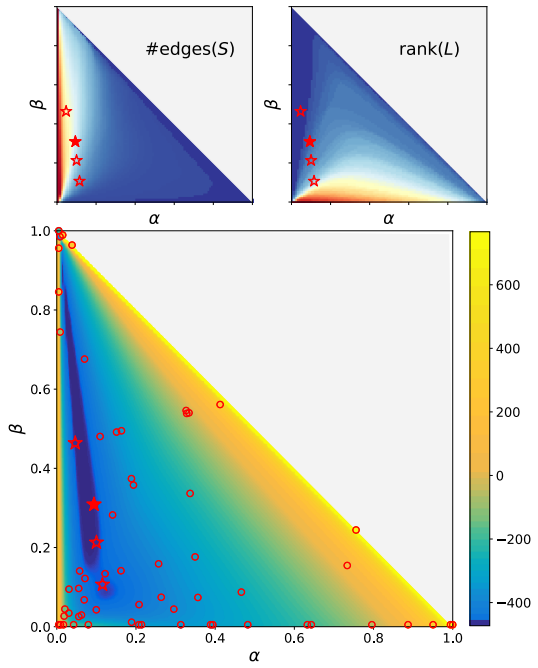

Figure 3: The lower figure shows the generalization errors from the grid search for the LSVT data set. Solutions from running Benson's algorithm are marked by circles. The best solutions are highlighted by stars. The filled star represents the best solution that is also shown in the first row of Figure 1. Above, the sparsity and rank patterns of the baseline are shown, where cold colors indicate high sparsity (left) and low rank (right).

Computational efficiency. It is known that the solution gamut method asymptotically matches the theoretical lower bound of $\Omega(1/\varepsilon)$ optimization problems that need to be solved for an ε-approximation. The pessimistic known theoretical upper bound for Benson's algorithm is only in $O(1/\varepsilon^2)$. In practice, however, one also has to consider the overhead for function evaluations on a fine grid for the solution gamut method and the overhead incurred by the adaptive Benson algorithm for facet enumerations. Hence, in our experiments we counted not only the number of solved optimization problems, but also measured the elapsed CPU time.

Results

Solution quality. On the LSVT data set, we stopped Benson's algorithm at the fairly coarse accuracy of $\varepsilon = 2^6$. This is already sufficient for finding good solutions with different algebraic properties (sparsity and rank), see Figure 3. Although the search region for the upper level problem, i.e., minimizing GE, is non-convex, Benson's algorithm finds solutions close to all local minima. Hence, in practice the best solutions returned by Benson's algorithm should be taken into account, because they may provide alternatives in terms of their algebraic properties, see Figure 1.

Computational efficiency. Experimental results for the comparison of the adaptive Benson algorithm and the solution gamut method are shown in Table 1. The solution gamut method requires solving between two and seven times more optimization problems than Benson's algorithm. The difference becomes even more pronounced when looking at the actual CPU times—it is a factor between 6 and 34 (median 13), i.e., about one order of magnitude.

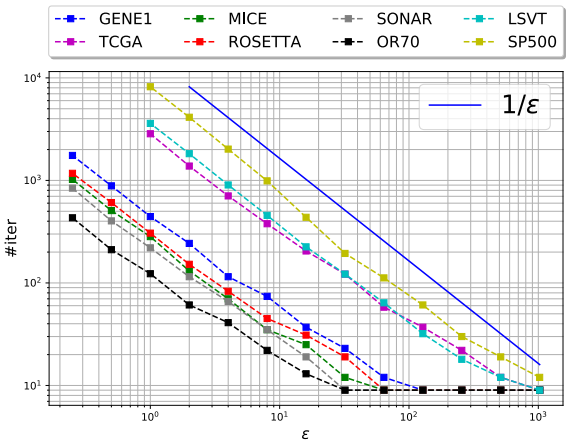

Figure 4: Log-log complexity plot for Benson's algorithm.

Furthermore, it turns out that the adaptive Benson algorithm experimentally matches the theoretical lower bound of $\Omega(1/\varepsilon)$, see Figure 4. This suggests that the optimistic upper bound for Benson's algorithm is realistic in our case.

4 Conclusion

Latent variable graphical models are a sparse + low-rank variant of multivariate Gaussians that address some of the shortcomings of the latter. Like Gaussian factor models or the graphical Lasso, this model is less prone to overfitting and does not suffer from the problem of non-regular covariance matrix estimates. It is more flexible than both factor models and the graphical Lasso, which reflects itself in a higher likelihood value on test data at its optimal solution. A good solution, though, can only be found by choosing the regularization parameters carefully. In practice, this is mostly done by either grid, manual, or random search which do not come with any performance guarantees.

Here, we have modeled regularization parameter selection as a bi-level optimization problem. The lower level of this approach entails describing the solution gamut of all feasible solutions for all combinations of regularization parameters. For approximating the solution gamut, we have adapted and extended Benson's vector optimization algorithm. On data sets from different domains, our bi-level approach with Benson's algorithm for the lower level works better than the standard approach for solution gamut approximation for the latent variable graphical Lasso. Using Benson's algorithm on the lower level is a generic approach that works out-of-the-box for any other problem that is given in the form of a regularized semidefinite program. Also, Benson's algorithm matches the known lower complexity bound for the solution gamut approximation problem.

Acknowledgements

This work has been supported by Carl-Zeiss-Stiftung and the DFG grant GI-711/3-2.

References

[Benson, 1998] Harold P. Benson. An Outer Approximation Algorithm for Generating All Efficient Extreme Points in the Outcome Set of a Multiple Objective Linear Programming Problem. *Journal of Global Optimization*, 13(1):1–24, 1998.

[Blechschmidt *et al.*, 2015] Katharina Blechschmidt, Joachim Giesen, and Sören Laue. Tracking Approximate Solutions of Parameterized Optimization Problems over Multi-Dimensional (Hyper-)Parameter Domains. In *International Conference on Machine Learning (ICML)*, pages 438–447, 2015.

[Bremner *et al.*, 1998] David Bremner, Komei Fukuda, and Ambros Marzetta. Primal-Dual Methods for Vertex and Facet Enumeration. *Discrete & Computational Geometry*, 20(3):333–357, 1998.

[Candès and Recht, 2009] Emmanuel J. Candès and Benjamin Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[Candès *et al.*, 2011] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust Principal Component Analysis? *Journal of the ACM*, 58(3):11:1–11:37, 2011.

[Chandrasekaran *et al.*, 2012] Venkat Chandrasekaran, Pablo A. Parrilo, and Alan S. Willsky. Latent Variable Graphical Model Selection via Convex Optimization. *The Annals of Statistics*, 40(4):1935–1967, 2012.

[Chen *et al.*, 2011] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating Low-Rank and Group-Sparse Structures for Robust Multi-Task Learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 42–50, 2011.

[Chen *et al.*, 2017] Fen Chen, Peng Zhao, Ting Feng Tang, and Yan Zhou. Fast Low-Rank Decomposition Model-Based Hyperspectral Image Classification Method. *IEEE Geoscience Remote Sensing Letters*, 14(2):169–173, 2017.

[Dempe, 2002] Stephan Dempe. *Foundations of Bilevel Programming*. Springer, 2002.

[Dua and Karra Taniskidou, 2017] Dheeru Dua and Efi Karra Taniskidou. UCI Machine Learning Repository, 2017.

[Dudík *et al.*, 2004] Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. Performance Guarantees for Regularized Maximum Entropy Density Estimation. In *Conference on Learning Theory (COLT)*, pages 472–486, 2004.

[Giesen *et al.*, 2019] Joachim Giesen, Andreas Löhne, Sören Laue, and Christopher Schneider. Using Benson's Algorithm for Regularization Parameter Tracking. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[Giraud and Tsybakov, 2012] Christophe Giraud and Alexandre Tsybakov. Discussion: Latent Variable Graphical Model Selection via Convex Optimization. *The Annals of Statistics*, 40(4):1996–2004, 2012.

[Giraud *et al.*, 2012] Christophe Giraud, Sylvie Huet, and Nicolas Verzelen. Graph Selection with GGMselect. *Statistical Applications in Genetics and Molecular Biology*, 11(3), 2012.

[Hansen *et al.*, 1992] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New Branch-and-Bound Rules for Linear Bilevel Programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[Higuera *et al.*, 2015] Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome. *PloS ONE*, 10(6):1–28, 2015.

[Kamenev, 1994] George K. Kamenev. Analysis of an Algorithm for Approximating Convex Bodies. *Computational Mathematics and Mathematical Physics*, 34(4):521–528, 1994.

[Löhne *et al.*, 2014] Andreas Löhne, Birgit Rudloff, and Firdevs Ulus. Primal and Dual Approximation Algorithms for Convex Vector Optimization Problems. *Journal of Global Optimization*, 60(4):713–736, 2014.

[Ma *et al.*, 2013] Shiqian Ma, Lingzhou Xue, and Hui Zou. Alternating Direction Methods for Latent Variable Gaussian Graphical Model Selection. *Neural Computation*, 25(8):2172–2198, 2013.

[Song *et al.*, 2007] Le Song, Alexander J. Smola, Karsten M. Borgwardt, and Arthur Gretton. Colored Maximum Variance Unfolding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1385–1392, 2007.

[Spearman, 1904] Charles Spearman. "General Intelligence," Objectively Determined and Measured. *American Journal of Psychology*, 15:201–293, 1904.

[Tsanas *et al.*, 2014] Athanasios Tsanas, Max A. Little, Cynthia Fox, and Lorraine O. Ramig. Objective Automatic Assessment of Rehabilitative Speech Treatment in Parkinson's Disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):181–190, 2014.

[von Stackelberg, 2010] Heinrich von Stackelberg. *Market Structure and Equilibrium*. Springer Berlin Heidelberg, 2010.

[Yuan and Lin, 2007] Ming Yuan and Yi Lin. Model Selection and Estimation in the Gaussian Graphical Model. *Biometrika*, 94(1):19–35, 2007.

[Zhou *et al.*, 2014] Fang Zhou, Q. Claire, and Ross D. King. Predicting the Geographical Origin of Music. In *IEEE International Conference on Data Mining (ICDM)*, pages 1115–1120, 2014.