Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm

Mark Schmidt, Ewout van den Berg,
Michael P. Friedlander, and Kevin Murphy

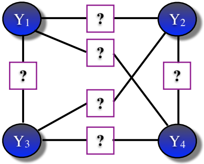
Department of Computer Science
University of British Columbia

April 18, 2009

Outline

Motivating Problem: Structure Learning in Discrete MRFs

- We want to fit a Markov random field to discrete data y, but don't know the graph structure

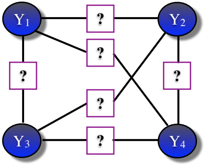


- We can learn a sparse structure by using ℓ_1-regularization of the edge parameters [Wainwright et al. 2006, Lee et al. 2006]
- Since each edge has multiple parameters, we use group ℓ_1-regularization [Bach et al. 2004, Turlach et al. 2005, Yuan & Lin 2006]:

$$\underset{w}{\text{minimize}} \; -\log p(y|w) \quad \text{subject to} \quad \sum_e ||w_e||_2 \leq \tau$$

Motivating Problem: Structure Learning in Discrete MRFs

- We want to fit a Markov random field to discrete data y, but don't know the graph structure

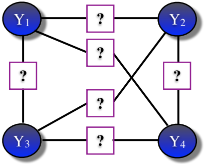


- We can learn a sparse structure by using ℓ_1-regularization of the edge parameters [Wainwright et al. 2006, Lee et al. 2006]
- Since each edge has multiple parameters, we use group ℓ_1-regularization
[Bach et al. 2004, Turlach et al. 2005, Yuan & Lin 2006]:

$$\underset{w}{\text{minimize}} \; -\log p(y|w) \quad \text{subject to} \quad \sum_e ||w_e||_2 \leq \tau$$

Motivating Problem: Structure Learning in Discrete MRFs

- We want to fit a Markov random field to discrete data y, but don't know the graph structure



- We can learn a sparse structure by using ℓ_1-regularization of the edge parameters [Wainwright et al. 2006, Lee et al. 2006]
- Since each edge has multiple parameters, we use group ℓ_1-regularization
  [Bach et al. 2004, Turlach et al. 2005, Yuan & Lin 2006]:

$$\underset{w}{\text{minimize}} \; -\log p(y|w) \quad \text{subject to} \quad \sum_e ||w_e||_2 \leq \tau$$

Optimization Problem Challenges

Solving this optimization problem has 3 complicating factors:

1. the number of parameters is large
2. evaluating the objective is expensive
3. the parameters have constraints

So how should we solve it?

- Interior point methods: the number of parameters is too large
- Projected gradient: evaluating the objective is too expensive
- Quasi-Newton methods (L-BFGS): we have constraints

Optimization Problem Challenges

Solving this optimization problem has 3 complicating factors:

1. the number of parameters is large
2. evaluating the objective is expensive
3. the parameters have constraints

So how should we solve it?

- Interior point methods: the number of parameters is too large
- Projected gradient: evaluating the objective is too expensive
- Quasi-Newton methods (L-BFGS): we have constraints

Extending the L-BFGS Algorithm

Quasi-Newton methods that use L-BFGS updates achieve state of the art performance for unconstrained differentiable optimization [Nocedal 1980, Liu & Nocedal 1989]

L-BFGS updates have also been used for more general problems:

- L-BFGS-B: state of the art performance for bound constrained optimization [Byrd et al. 1995]
- OWL-QN: state of the art performance for ℓ_1-regularized optimization [Andrew & Gao 2007].

The above don't apply since our constraints are not separable

However, the constraints are still simple:

- we can compute the projection in $\mathcal{O}(n)$

Extending the L-BFGS Algorithm

Quasi-Newton methods that use L-BFGS updates achieve state of the art performance for unconstrained differentiable optimization [Nocedal 1980, Liu & Nocedal 1989]

L-BFGS updates have also been used for more general problems:

- L-BFGS-B: state of the art performance for bound constrained optimization [Byrd et al. 1995]
- OWL-QN: state of the art performance for ℓ_1-regularized optimization [Andrew & Gao 2007].

The above don't apply since our constraints are not separable

However, the constraints are still simple:

- we can compute the projection in $\mathcal{O}(n)$

Extending the L-BFGS Algorithm

Quasi-Newton methods that use L-BFGS updates achieve state of the art performance for unconstrained differentiable optimization [Nocedal 1980, Liu & Nocedal 1989]

L-BFGS updates have also been used for more general problems:

- L-BFGS-B: state of the art performance for bound constrained optimization [Byrd et al. 1995]
- OWL-QN: state of the art performance for ℓ_1-regularized optimization [Andrew & Gao 2007].

The above don't apply since our constraints are not separable

However, the constraints are still simple:

- we can compute the projection in $\mathcal{O}(n)$

Extending the L-BFGS Algorithm

Quasi-Newton methods that use L-BFGS updates achieve state of the art performance for unconstrained differentiable optimization [Nocedal 1980, Liu & Nocedal 1989]

L-BFGS updates have also been used for more general problems:

- L-BFGS-B: state of the art performance for bound constrained optimization [Byrd et al. 1995]
- OWL-QN: state of the art performance for ℓ_1-regularized optimization [Andrew & Gao 2007].

The above don't apply since our constraints are not separable

However, the constraints are still simple:

- we can compute the projection in $\mathcal{O}(n)$

Our Contribution

This talk presents an extension of L-BFGS that is suitable when:

1. the number of parameters is large
2. evaluating the objective is expensive
3. the parameters have constraints
4. projecting onto the constraints is substantially cheaper than evaluating the objective function

The method uses a two-level strategy

- At the outer level, L-BFGS updates build a constrained local quadratic approximation to the function
- At the inner level, SPG uses projections to minimize this constrained quadratic approximation

Our Contribution

This talk presents an extension of L-BFGS that is suitable when:

1. the number of parameters is large
2. evaluating the objective is expensive
3. the parameters have constraints
4. projecting onto the constraints is substantially cheaper than evaluating the objective function

The method uses a two-level strategy

- At the outer level, L-BFGS updates build a constrained local quadratic approximation to the function
- At the inner level, SPG uses projections to minimize this constrained quadratic approximation

Our Contribution

This talk presents an extension of L-BFGS that is suitable when:

1. the number of parameters is large
2. evaluating the objective is expensive
3. the parameters have constraints
4. projecting onto the constraints is substantially cheaper than evaluating the objective function

The method uses a two-level strategy

- At the outer level, L-BFGS updates build a constrained local quadratic approximation to the function
- At the inner level, SPG uses projections to minimize this constrained quadratic approximation

Outline

Problem Statement and Assumptions

We address the problem of minimizing a differentiable function $f(x)$ over a convex set $\mathcal{C}$:

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

We assume you can compute the objective $f(x)$, the gradient $\nabla f(x)$, and the projection $\mathcal{P}_{\mathcal{C}}(x)$:

$$\mathcal{P}_{\mathcal{C}}(x) = \arg\min_{c} \quad \|c - x\|_2 \quad \text{subject to} \quad c \in \mathcal{C}.$$

Problem Statement and Assumptions

We address the problem of minimizing a differentiable function $f(x)$ over a convex set $\mathcal{C}$:

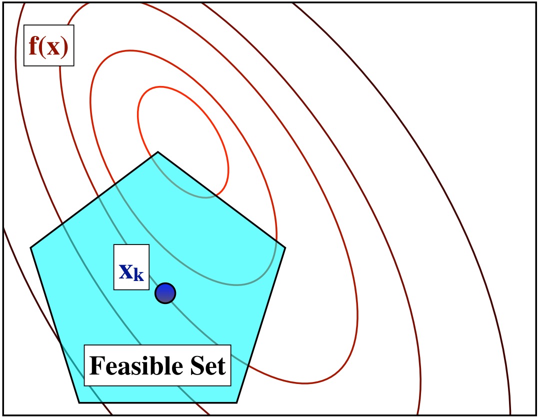
$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

We assume you can compute the objective $f(x)$, the gradient $\nabla f(x)$, and the projection $\mathcal{P}_{\mathcal{C}}(x)$:

$$\mathcal{P}_{\mathcal{C}}(x) = \arg\min_{c} \quad \|c - x\|_2 \quad \text{subject to} \quad c \in \mathcal{C}.$$

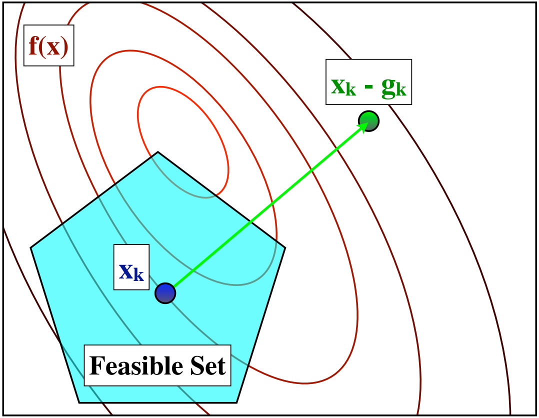
PG: Projected Gradient Algorithm

PG: move towards the projection of the negative gradient



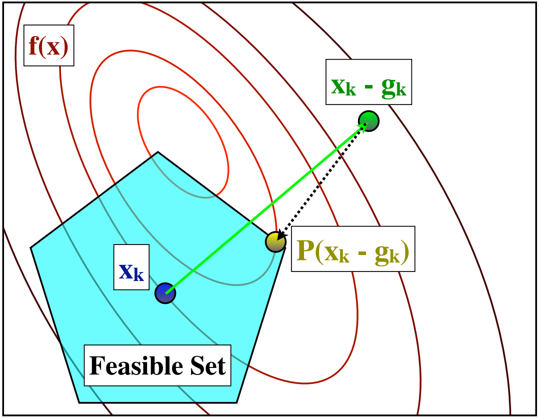
PG: Projected Gradient Algorithm

PG: move towards the projection of the negative gradient



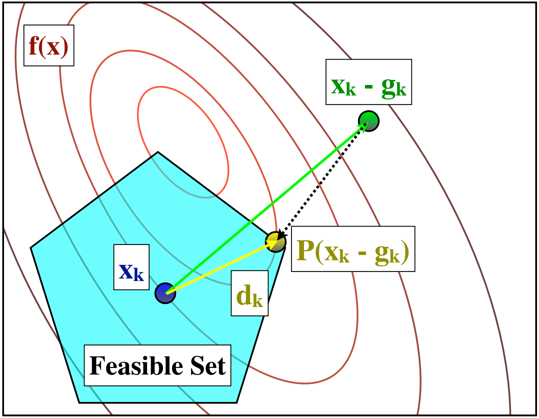
PG: Projected Gradient Algorithm

PG: move towards the projection of the negative gradient



PG: Projected Gradient Algorithm

PG: move towards the projection of the negative gradient



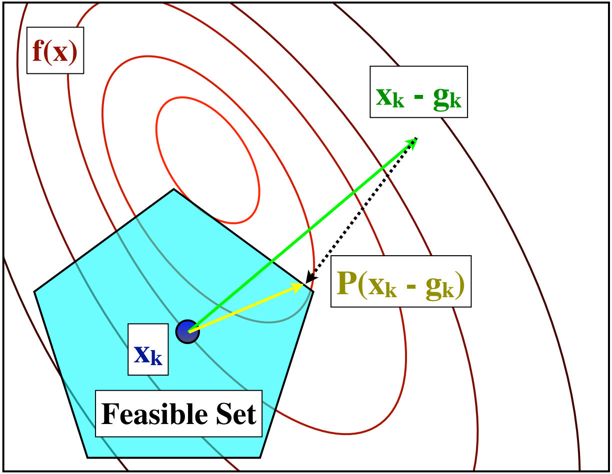
Naive Projected Newton Algorithm

- The problem with projected gradient: slow convergence
- Can we speed this up by projecting the Newton direction?

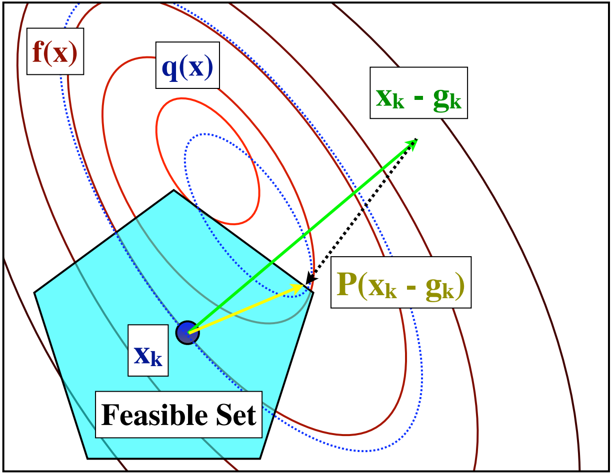
Naive Projected Newton Algorithm

- The problem with projected gradient: slow convergence
- Can we speed this up by projecting the Newton direction?

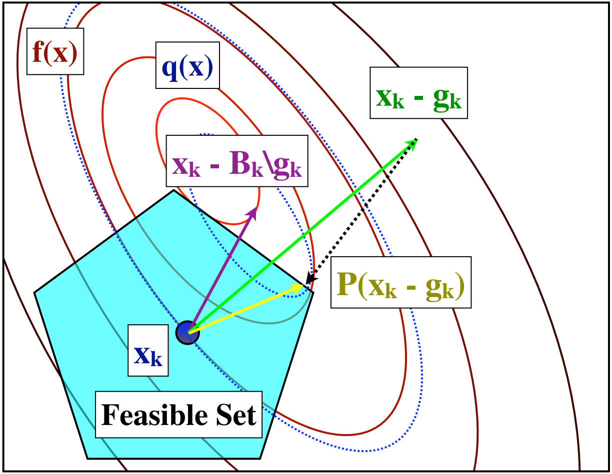
Naive Projected Newton Algorithm



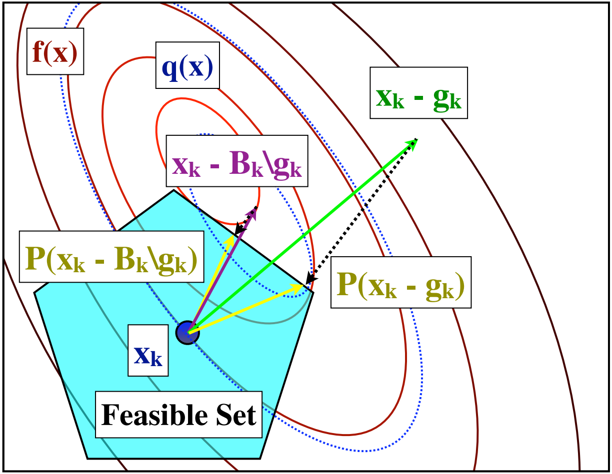
Naive Projected Newton Algorithm



Naive Projected Newton Algorithm



Naive Projected Newton Algorithm



Naive Projected Newton Algorithm

- The problem with projected gradient: slow convergence
- Can we speed this up by projecting the Newton direction?

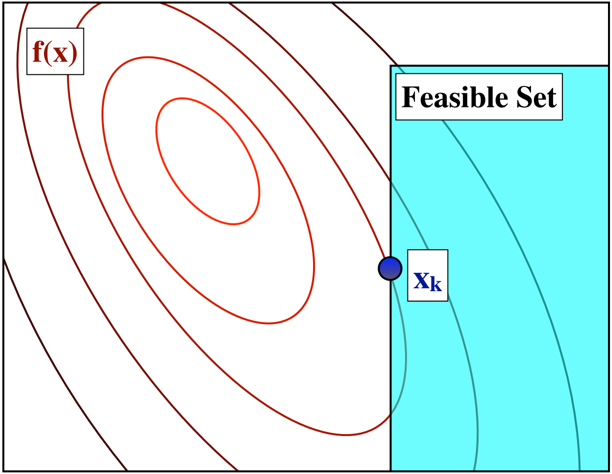
NO! This can point in the wrong direction

Naive Projected Newton Algorithm

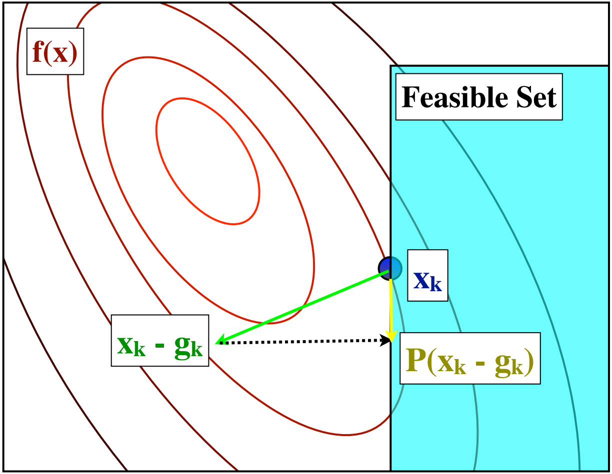
- The problem with projected gradient: slow convergence
- Can we speed this up by projecting the Newton direction?

NO! This can point in the wrong direction

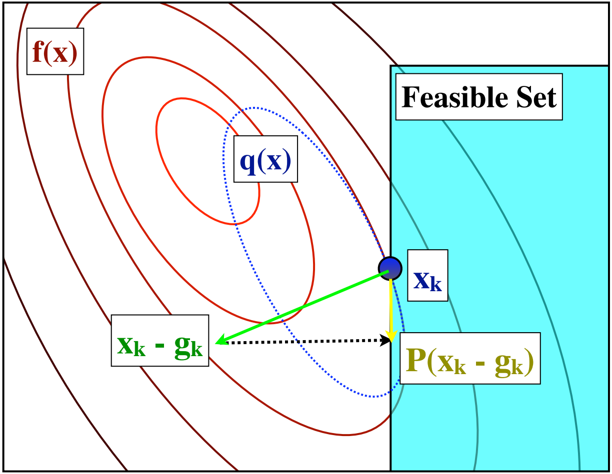
Naive Projected Gradient Algorithm: Problem



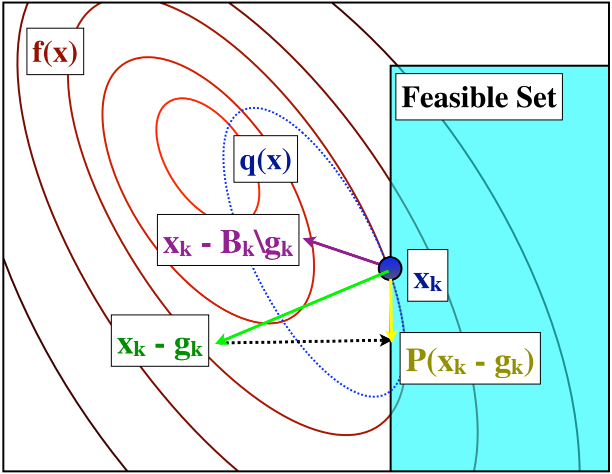
Naive Projected Gradient Algorithm: Problem



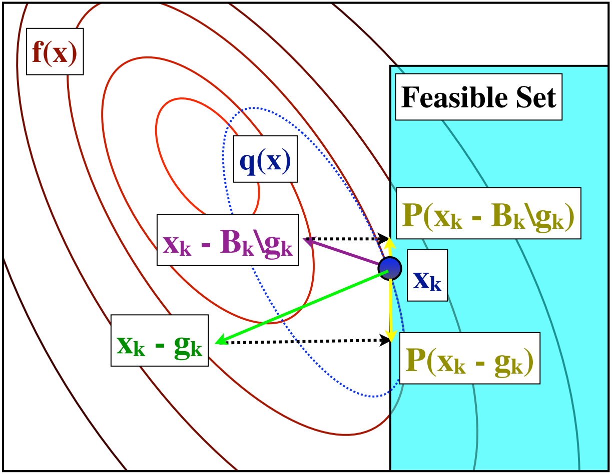
Naive Projected Gradient Algorithm: Problem



Naive Projected Gradient Algorithm: Problem



Naive Projected Gradient Algorithm: Problem



Correct Projected Newton Algorithm

- In projected Newton methods, we form a quadratic approximation to the function around x_k:

$$q_k(x) \triangleq f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$

- At each iteration, we minimize this function over the set:

$$\underset{x}{\text{minimize}} \quad q_k(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

- NOT the same as projecting the unconstrained Newton step
- This generates a feasible descent direction $d_k \triangleq x - x_k$
- The method has a quadratic rate of convergence around a local minimizer [Bertsekas, 1999]

Correct Projected Newton Algorithm

- In projected Newton methods, we form a quadratic approximation to the function around x_k:

$$q_k(x) \triangleq f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$

- At each iteration, we minimize this function over the set:

$$\underset{x}{\text{minimize}} \quad q_k(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

- NOT the same as projecting the unconstrained Newton step
- This generates a feasible descent direction $d_k \triangleq x - x_k$
- The method has a quadratic rate of convergence around a local minimizer [Bertsekas, 1999]

Correct Projected Newton Algorithm

- In projected Newton methods, we form a quadratic approximation to the function around x_k:

$$q_k(x) \triangleq f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$

- At each iteration, we minimize this function over the set:

$$\underset{x}{\text{minimize}} \quad q_k(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

- NOT the same as projecting the unconstrained Newton step
- This generates a feasible descent direction $d_k \triangleq x - x_k$
- The method has a quadratic rate of convergence around a local minimizer [Bertsekas, 1999]

Correct Projected Newton Algorithm

- In projected Newton methods, we form a quadratic approximation to the function around x_k:

$$q_k(x) \triangleq f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$

- At each iteration, we minimize this function over the set:

$$\underset{x}{\text{minimize}} \quad q_k(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

- NOT the same as projecting the unconstrained Newton step
- This generates a feasible descent direction $d_k \triangleq x - x_k$
- The method has a quadratic rate of convergence around a local minimizer [Bertsekas, 1999]

Correct Projected Newton Algorithm

- In projected Newton methods, we form a quadratic approximation to the function around x_k:

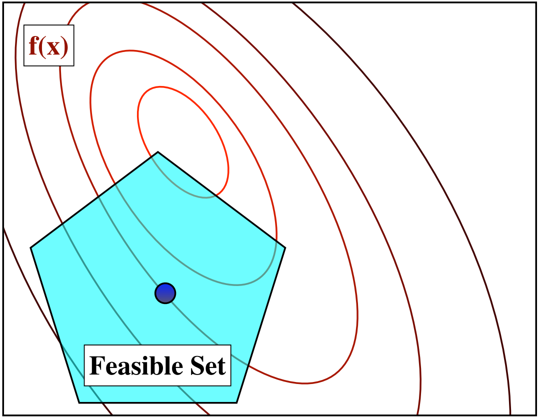
$$q_k(x) \triangleq f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$

- At each iteration, we minimize this function over the set:

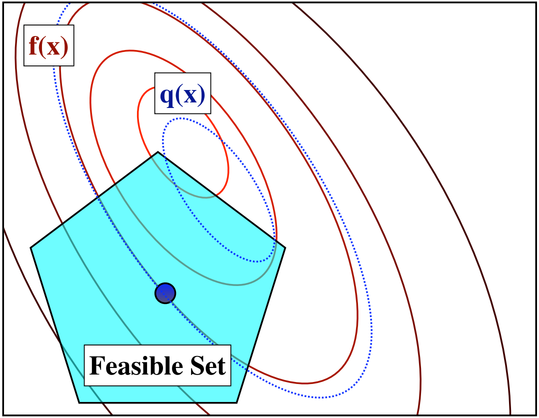
$$\underset{x}{\text{minimize}} \quad q_k(x) \quad \text{subject to} \quad x \in \mathcal{C}$$

- NOT the same as projecting the unconstrained Newton step
- This generates a feasible descent direction $d_k \triangleq x - x_k$
- The method has a quadratic rate of convergence around a local minimizer [Bertsekas, 1999]

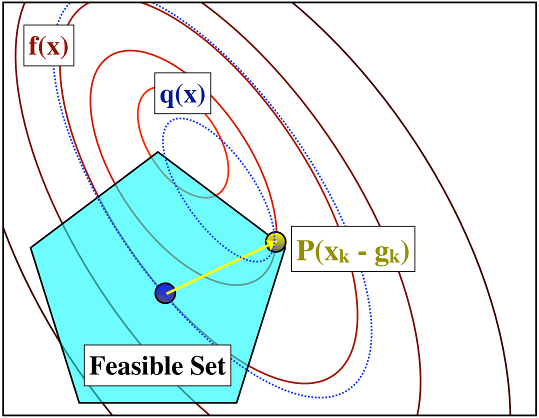
Projected Newton Algorithm



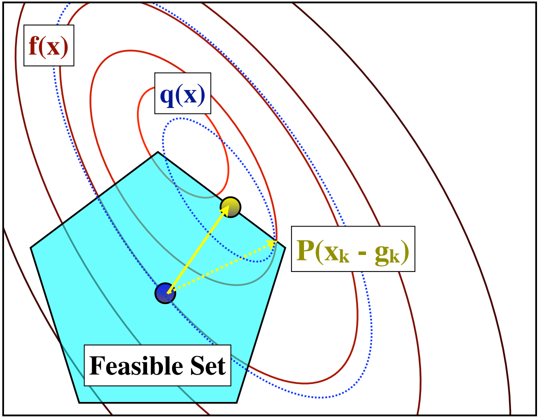
Projected Newton Algorithm



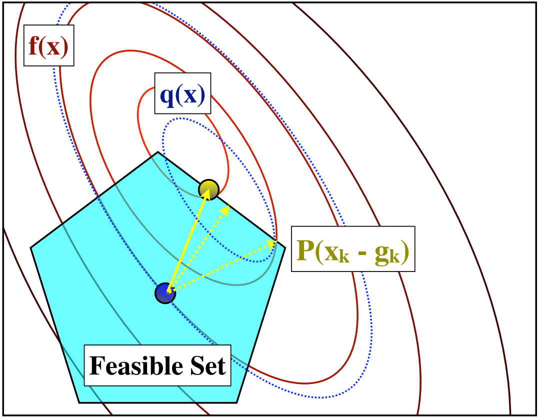
Projected Newton Algorithm



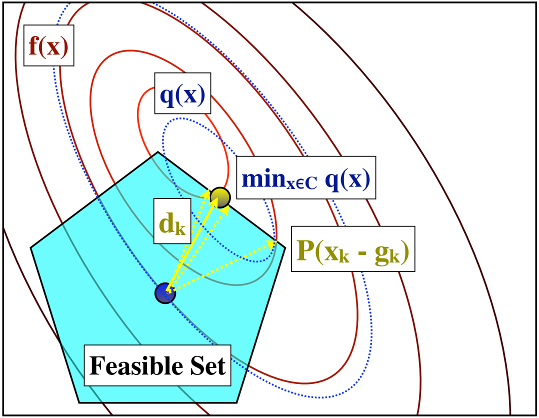
Projected Newton Algorithm



Projected Newton Algorithm



Projected Newton Algorithm



Problems with the Projected Newton Algorithm

Unfortunately, the projected Newton method can be inefficient:

- Computing d_k may be very expensive
- Using a general n-by-n matrix B_k is impratical

Our algorithm is a projected quasi-Newton algorithm where:

- L-BFGS updates construct a diagonal plus low-rank B_k
- SPG efficiently computes d_k with this B_k and projections.

Problems with the Projected Newton Algorithm

Unfortunately, the projected Newton method can be inefficient:

- Computing d_k may be very expensive
- Using a general n-by-n matrix B_k is impratical

Our algorithm is a projected quasi-Newton algorithm where:

- L-BFGS updates construct a diagonal plus low-rank B_k
- SPG efficiently computes d_k with this B_k and projections.

Outline

Broyden-Fletcher-Goldfarb-Shanno (BFGS) Updates

Quasi-Newton methods work with parameter and gradient differences between iterations:

$$s_k \triangleq x_{k+1} - x_k \quad \text{and} \quad y_k \triangleq g_{k+1} - g_k$$

They start with an initial approximation $B_0 \triangleq \sigma I$, and choose B_{k+1} to interpolate the gradient difference:

$$B_{k+1} s_k = y_k$$

Since B_{k+1} is not unique, the BFGS method chooses the matrix whose difference with B_k minimizes a weighted Frobenius norm:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) Updates

Quasi-Newton methods work with parameter and gradient differences between iterations:

$$s_k \triangleq x_{k+1} - x_k \quad \text{and} \quad y_k \triangleq g_{k+1} - g_k$$

They start with an initial approximation $B_0 \triangleq \sigma I$, and choose B_{k+1} to interpolate the gradient difference:

$$B_{k+1} s_k = y_k$$

Since B_{k+1} is not unique, the BFGS method chooses the matrix whose difference with B_k minimizes a weighted Frobenius norm:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) Updates

Quasi-Newton methods work with parameter and gradient differences between iterations:

$$s_k \triangleq x_{k+1} - x_k \quad \text{and} \quad y_k \triangleq g_{k+1} - g_k$$

They start with an initial approximation $B_0 \triangleq \sigma I$, and choose B_{k+1} to interpolate the gradient difference:

$$B_{k+1} s_k = y_k$$

Since B_{k+1} is not unique, the BFGS method chooses the matrix whose difference with B_k minimizes a weighted Frobenius norm:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

L-BFGS: Limited-Memory BFGS

Instead of storing B_k, the limited-memory BFGS (L-BFGS) method just stores the previous m differences s_k and y_k.
[Nocedal 1980, Liu & Nocedal 1989]

These updates applied to $B_0 = \sigma_k I$ can be written compactly in a diagonal plus low-rank form [Byrd et al. 1994]:

$$B_m = \sigma_k I - NM^{-1}N^T$$

This representations makes multiplication with B_k cost $\mathcal{O}(mn)$.

SPG: Spectral Projected Gradient

Recall the projected quasi-Newton sub-problem:

$$\underset{x}{\text{minimize}} \quad f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$
$$\text{subject to } x \in \mathcal{C}$$

With the L-BFGS representation of B_k, we can compute the objective function and gradient in $\mathcal{O}(mn)$.

This still doesn't let us efficiently solve the problem

To solve it, we use the spectral projected gradient (SPG) algorithm.

SPG: Spectral Projected Gradient

Recall the projected quasi-Newton sub-problem:

$$\underset{x}{\text{minimize}} \quad f_k + (x - x_k)^T \nabla f(x_k) + \tfrac{1}{2}(x - x_k)^T B_k (x - x_k)$$
$$\text{subject to } x \in \mathcal{C}$$

With the L-BFGS representation of B_k, we can compute the objective function and gradient in $\mathcal{O}(mn)$.

This still doesn't let us efficiently solve the problem

To solve it, we use the spectral projected gradient (SPG) algorithm.

SPG: Spectral Projected Gradient

Recall the projected quasi-Newton sub-problem:

$$\underset{x}{\text{minimize}} \quad f_k + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k)$$
$$\text{subject to } x \in \mathcal{C}$$

With the L-BFGS representation of B_k, we can compute the objective function and gradient in $\mathcal{O}(mn)$.

This still doesn't let us efficiently solve the problem

To solve it, we use the spectral projected gradient (SPG) algorithm.

SPG: Spectral Projected Gradient

The classic projected gradient takes steps of the form

$$x_{k+1} = \mathcal{P}_{\mathcal{C}}(x_k - \alpha g_k)$$

SPG has two enhancements [Birgin et al. 2000]:

- It uses the Barzilai and Borwein [1988] 'spectral' step length:

$$\alpha_{bb} = \frac{\langle y_{k-1}, y_{k-1} \rangle}{\langle s_{k-1}, y_{k-1} \rangle}$$

- It uses a non-monotone line search [Grippo et al. 1986]

SPG: Spectral Projected Gradient

The classic projected gradient takes steps of the form

$$x_{k+1} = \mathcal{P}_{\mathcal{C}}(x_k - \alpha g_k)$$

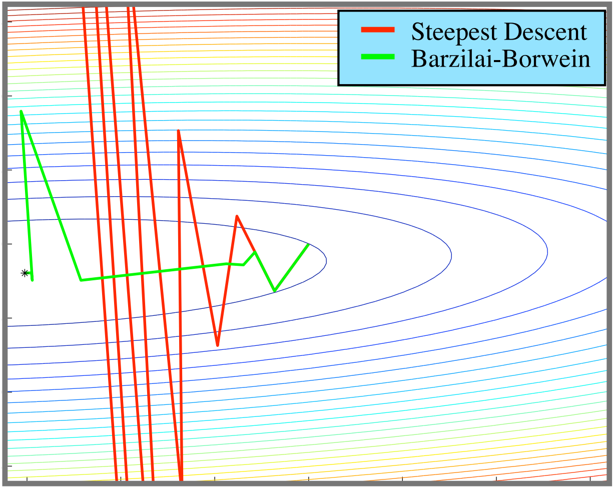
SPG has two enhancements [Birgin et al. 2000]:

- It uses the Barzilai and Borwein [1988] 'spectral' step length:

$$\alpha_{bb} = \frac{\langle y_{k-1}, y_{k-1} \rangle}{\langle s_{k-1}, y_{k-1} \rangle}$$

- It uses a non-monotone line search [Grippo et al. 1986]

Barzilai & Borwein Step Size



SPG: Spectral Projected Gradient

- There is growing interest in SPG for constrained optimization [Dai & Fletcher 2005, van den Berg & Friedlander 2008]

- We apply SPG to minimize the strictly convex constrained quadratic approximations
- Friedlander et al. [1999] show that SPG has a superlinear convergence rate for minimizing strictly convex quadratics

- Instead of 'solving' the sub-problem, we could just perform k iterations of SPG to improve the steepest descent direction.
- In this case, solving the sub-problems is in $\mathcal{O}(mnk)$, plus the cost of computing the projection k times.

SPG: Spectral Projected Gradient

- There is growing interest in SPG for constrained optimization [Dai & Fletcher 2005, van den Berg & Friedlander 2008]

- We apply SPG to minimize the strictly convex constrained quadratic approximations
- Friedlander et al. [1999] show that SPG has a superlinear convergence rate for minimizing strictly convex quadratics

- Instead of 'solving' the sub-problem, we could just perform k iterations of SPG to improve the steepest descent direction.
- In this case, solving the sub-problems is in $\mathcal{O}(mnk)$, plus the cost of computing the projection k times.

SPG: Spectral Projected Gradient

- There is growing interest in SPG for constrained optimization [Dai & Fletcher 2005, van den Berg & Friedlander 2008]

- We apply SPG to minimize the strictly convex constrained quadratic approximations
- Friedlander et al. [1999] show that SPG has a superlinear convergence rate for minimizing strictly convex quadratics

- Instead of 'solving' the sub-problem, we could just perform k iterations of SPG to improve the steepest descent direction.
- In this case, solving the sub-problems is in $\mathcal{O}(mnk)$, plus the cost of computing the projection k times.

SPG: Spectral Projected Gradient

- There is growing interest in SPG for constrained optimization [Dai & Fletcher 2005, van den Berg & Friedlander 2008]

- We apply SPG to minimize the strictly convex constrained quadratic approximations
- Friedlander et al. [1999] show that SPG has a superlinear convergence rate for minimizing strictly convex quadratics

- Instead of 'solving' the sub-problem, we could just perform k iterations of SPG to improve the steepest descent direction.
- In this case, solving the sub-problems is in $\mathcal{O}(mnk)$, plus the cost of computing the projection k times.

Outline of the Method

The projected quasi-Newton (PQN) method:

1. Evaluate the current objective function and gradient
2. Add/remove difference vectors for L-BFGS
3. Run SPG to compute the projected quasi-Newton direction d_k
4. Generate the next iterate with a backtracking line search

The overall algorithm will be most effective when:
computing projections is cheaper than evaluating the objective

Outline of the Method

The projected quasi-Newton (PQN) method:

1. Evaluate the current objective function and gradient
2. Add/remove difference vectors for L-BFGS
3. Run SPG to compute the projected quasi-Newton direction d_k
4. Generate the next iterate with a backtracking line search

The overall algorithm will be most effective when:
computing projections is cheaper than evaluating the objective

Outline

Projection onto Norm-Balls

We are interested in projecting onto balls induced by norms:

$$\mathcal{C} \equiv \{x \mid \|x\| \leq \tau\}$$

This projection can be computed in linear-time for many ℓ_p-norms, such as the ℓ_2-, ℓ_∞-, and ℓ_1-norms [Duchi et al. 2008]

We are also interested in the case of the mixed p, q-norm balls that arise in group variable selection:

$$\|x\|_{p,q} = \left(\sum_i \|x_{\sigma_i}\|_q^p\right)^{1/p}$$

The group-lasso is the special case where $p = 1$, $q = 2$:

$$\|x\|_{1,2} = \sum_i \|x_{\sigma_i}\|_2$$

Projection onto Norm-Balls

We are interested in projecting onto balls induced by norms:

$$\mathcal{C} \equiv \{x \mid \|x\| \leq \tau\}$$

This projection can be computed in linear-time for many ℓ_p-norms, such as the ℓ_2-, ℓ_∞-, and ℓ_1-norms [Duchi et al. 2008]

We are also interested in the case of the mixed p, q-norm balls that arise in group variable selection:

$$\|x\|_{p,q} = \left(\sum_i \|x_{\sigma_i}\|_q^p\right)^{1/p}$$

The group-lasso is the special case where $p = 1$, $q = 2$:

$$\|x\|_{1,2} = \sum_i \|x_{\sigma_i}\|_2$$

Projection onto Mixed Norm-Balls

The following proposition leads to an expected linear-time randomized algorithm for group-lasso projection:

Proposition

Consider $c \in \mathbb{R}^n$ and a set of g disjoint groups $\{\sigma_i\}_{i=1}^g$ such that $\cup_i \sigma_i = \{1, \ldots, n\}$. Then the Euclidean projection $\mathcal{P}_{\mathcal{C}}(c)$ onto the $\ell_{1,2}$-norm ball of radius τ is given by

$$x_{\sigma_i} = \mathrm{sgn}(c_{\sigma_i}) \cdot w_i, \qquad i = 1, \ldots, g,$$

where $w = \mathcal{P}(v)$ is the projection of vector v onto the ℓ_1-norm ball of radius τ, with $v_i = \|c_{\sigma_i}\|_2$.

Outline

Experiments

We performed several experiments to test the new method:

- We first compared to other extensions of L-BFGS [see paper]
- We then compared to state of the art methods for graph structure learning

Experiments

We performed several experiments to test the new method:

- We first compared to other extensions of L-BFGS [see paper]
- We then compared to state of the art methods for graph structure learning

Gaussian Graphical Model Structure Learning

We looked at training a Gaussian graphical model with an ℓ_1 penalty on the precision matrix elements to induce a sparse structure [Banerjee et al. 2006, Friedman et al. 2007]:

$$\underset{K \succ 0}{\text{minimize}} \quad -\log\det(K) + \mathrm{tr}(\hat{\Sigma}K) + \lambda\|K\|_1,$$

We used the Gasch et al. [2000] data with the pre-processing of Duchi et al. [2008], and as with previous work we solve the dual problem:

$$\begin{aligned} \underset{W}{\text{maximize}} \quad & \log\det(\hat{\Sigma} + W) \\ \text{subject to} \quad & \hat{\Sigma} + W \succ 0,\ \|W\|_\infty \leq \lambda \end{aligned}$$

We compared to a projected gradient method [Duchi et al. 2008].

Gaussian Graphical Model Structure Learning

We looked at training a Gaussian graphical model with an ℓ_1 penalty on the precision matrix elements to induce a sparse structure [Banerjee et al. 2006, Friedman et al. 2007]:

$$\underset{K \succ 0}{\text{minimize}} \quad -\log \det(K) + \mathrm{tr}(\hat{\Sigma} K) + \lambda \|K\|_1,$$

We used the Gasch et al. [2000] data with the pre-processing of Duchi et al. [2008], and as with previous work we solve the dual problem:

$$\begin{aligned} \underset{W}{\text{maximize}} \quad & \log \det(\hat{\Sigma} + W) \\ \text{subject to} \quad & \hat{\Sigma} + W \succ 0, \ \|W\|_\infty \le \lambda \end{aligned}$$

We compared to a projected gradient method [Duchi et al. 2008].

Gaussian Graphical Model Structure Learning

We looked at training a Gaussian graphical model with an ℓ_1 penalty on the precision matrix elements to induce a sparse structure [Banerjee et al. 2006, Friedman et al. 2007]:

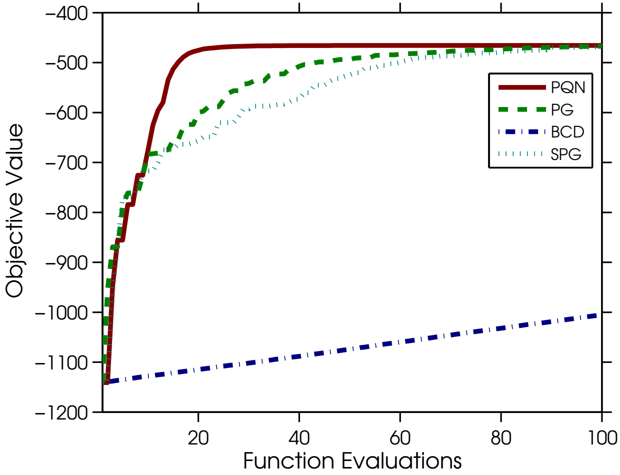
$$\underset{K \succ 0}{\text{minimize}} \quad -\log\det(K) + \mathrm{tr}(\hat{\Sigma}K) + \lambda\|K\|_1,$$

We used the Gasch et al. [2000] data with the pre-processing of Duchi et al. [2008], and as with previous work we solve the dual problem:

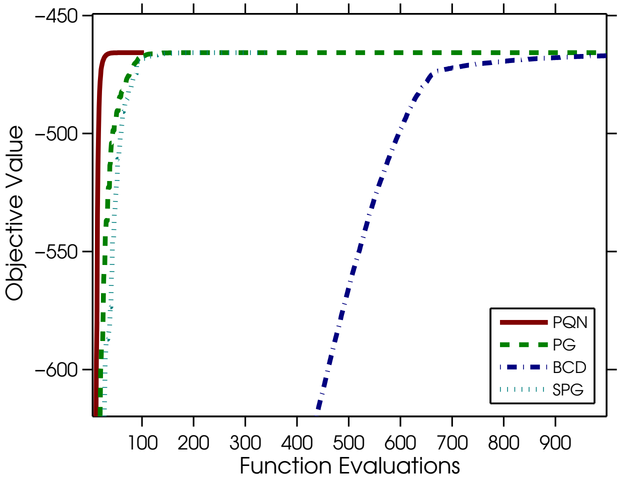
$$\begin{aligned} \underset{W}{\text{maximize}} \quad & \log\det(\hat{\Sigma} + W) \\ \text{subject to} \quad & \hat{\Sigma} + W \succ 0, \; \|W\|_\infty \leq \lambda \end{aligned}$$

We compared to a projected gradient method [Duchi et al. 2008].

Gaussian Graphical Model Structure Learning



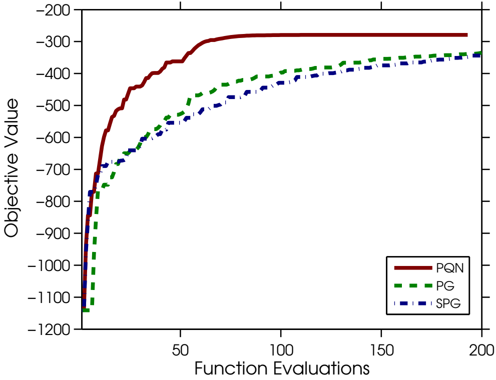
Gaussian Graphical Model Structure Learning



Gaussian Graphical Model Structure Learning with Groups

We also compared the methods when we induce a group-sparse precision matrix using the $\ell_{1,\infty}$-norm [Duchi et al. 2008]:

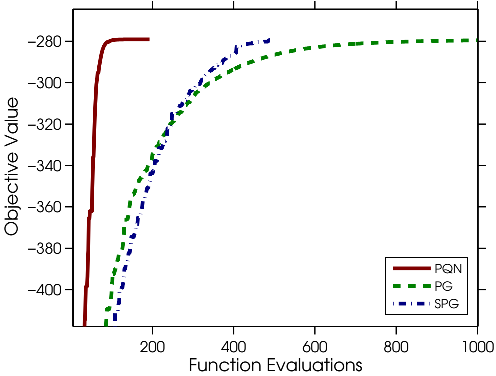
$$\underset{K \succ 0}{\text{minimize}} \quad -\log\det(K) + \text{tr}(\hat{\Sigma}K) + \lambda\|K\|_{1,\infty},$$



Gaussian Graphical Model Structure Learning with Groups

We also compared the methods when we induce a group-sparse precision matrix using the $\ell_{1,\infty}$-norm [Duchi et al. 2008]:

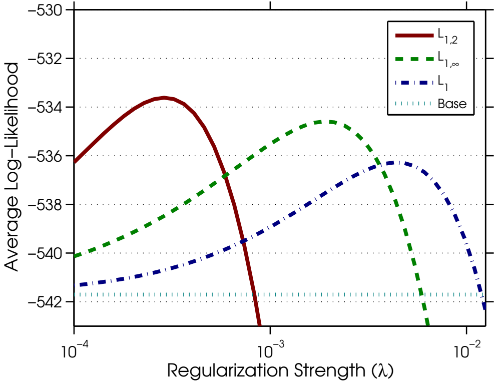
$$\underset{K \succ 0}{\text{minimize}} \quad -\log\det(K) + \mathrm{tr}(\hat{\Sigma}K) + \lambda \|K\|_{1,\infty},$$



Gaussian Graphical Model Structure Learning with Groups

We also used PQN to look at the performance if we replace the $\ell_{1,\infty}$-norm [Duchi et al. 2008] with the $\ell_{1,2}$-norm:

$$\underset{K \succ 0}{\text{minimize}} \quad -\log\det(K) + \mathrm{tr}(\hat{\Sigma}K) + \lambda \|K\|_{1,2},$$



Markov Random Field Structure Learning

Finally, we looked at learning a sparse Markov random field:

$$\underset{w}{\text{minimize}} \; -\log p(y|w) \quad \text{subject to} \quad \sum_{e} ||w_e||_2 \leq \tau$$

We used the trinary data from [Sachs et al. 2005], and compared to Grafting [Lee et al. 2006] and applying SPG to a second-order cone reformulation [Schmidt et al. 2008].

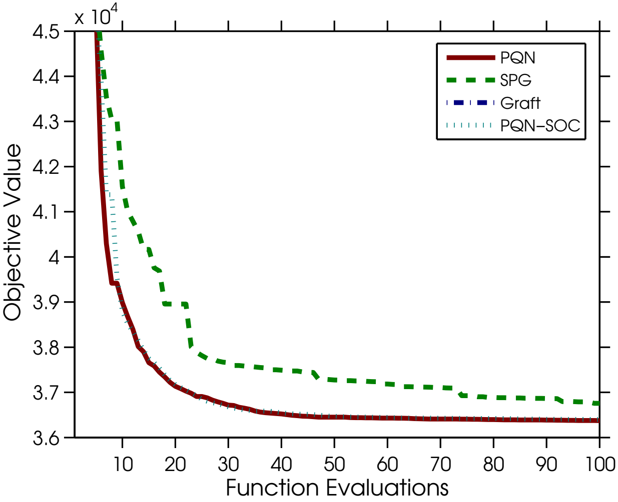
Markov Random Field Structure Learning

Finally, we looked at learning a sparse Markov random field:

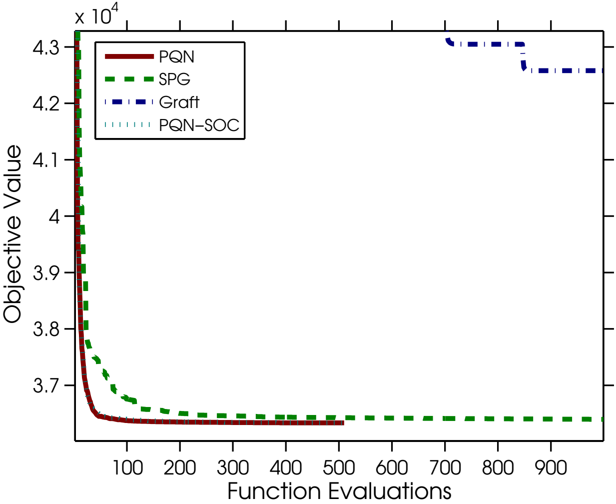
$$\underset{w}{\text{minimize}} \; -\log p(y|w) \quad \text{subject to} \quad \sum_e ||w_e||_2 \leq \tau$$

We used the trinary data from [Sachs et al. 2005], and compared to Grafting [Lee et al. 2006] and applying SPG to a second-order cone reformulation [Schmidt et al. 2008].

Markov Random Field Structure Learning



Markov Random Field Structure Learning



Outline

Extensions to Other Problems

There are many other cases where we can efficiently compute projections:

- Projection onto hyper-planes or half-spaces is trivial
- Projecting onto the probability simplex can be done in $\mathcal{O}(n \log n)$
- Projecting onto the positive semi-definite cone involves truncated the spectral decomposition
- Projecting onto second-order cones of the form $\|x\|_2 \leq y$ can be done in $\mathcal{O}(n)$
- Dykstra's algorithm can be used for combinations of simple constraints [Dykstra, 1983]

Summary

PQN is an extension of L-BFGS that is suitable when:

1. the number of parameters is large
2. evaluating the objective is expensive
3. the parameters have constraints
4. projecting onto the constraints is substantially cheaper than evaluating the objective function

We have found the algorithm useful for a variety of problems, and it is likely useful for others (code online soon)