Graphical Model Structure Learning with ℓ_1-Regularization

Mark Schmidt

July 27, 2010

Outline

Motivation for Graphical Model Structure Learning

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

Motivation for Graphical Model Structure Learning

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- What words are related?

Motivation for Graphical Model Structure Learning

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- What words are related?
- Is a post with (car,drive,hockey,pc,win) spam?

Motivation for Graphical Model Structure Learning

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

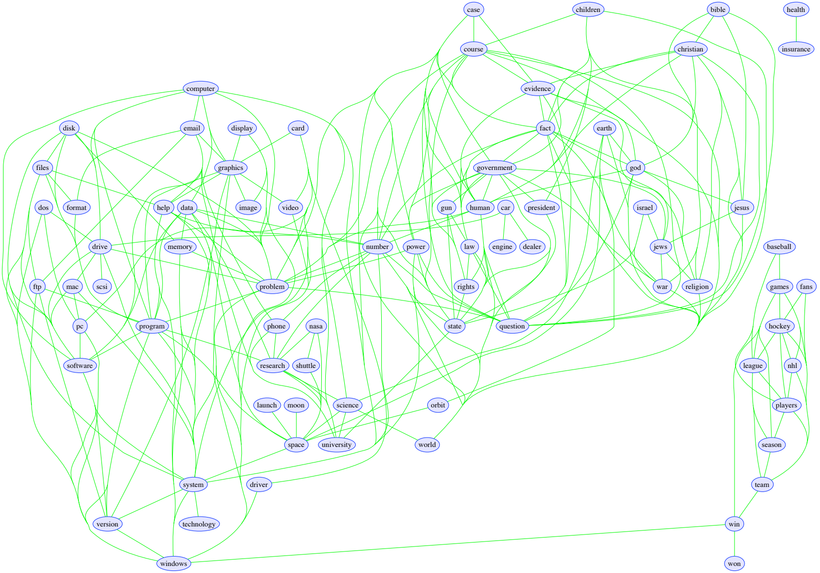
- What words are related?
- Is a post with (car,drive,hockey,pc,win) spam?
- What is $p(\text{car}|\text{drive})$? What about $p(\text{car}|\text{drive,files})$?

Motivation for Graphical Model Structure Learning

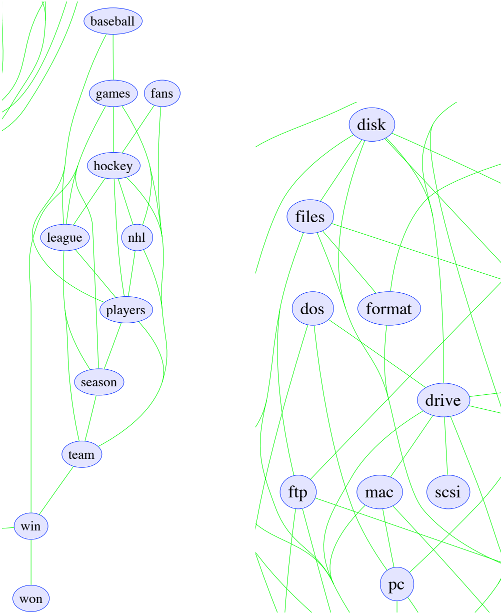
car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- What words are related?
- Is a post with (car,drive,hockey,pc,win) spam?
- What is $p(\text{car}|\text{drive})$? What about $p(\text{car}|\text{drive},\text{files})$?
- Given the values of some variables, what is the most likely way to fill-in the other variables?

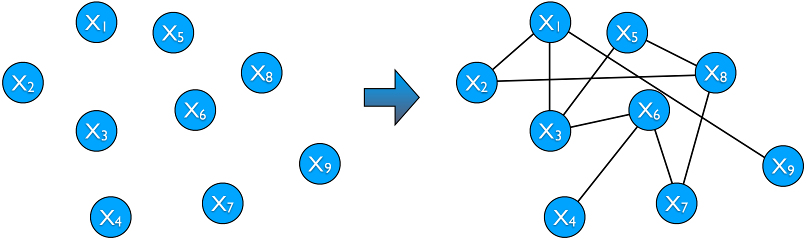
Example of Learned Graph Structure



Example of Learned Graph Structure

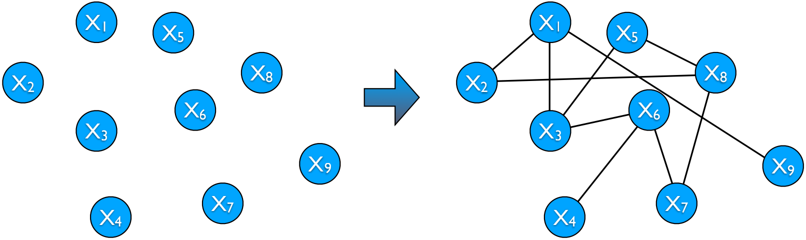


Graphical Model Structure Learning with ℓ_1-Regularization



- We consider parameter estimation in graphical models *without a known structure*.

Graphical Model Structure Learning with ℓ_1-Regularization



- We consider parameter estimation in graphical models *without a known structure*.
- There has been growing interest in ℓ_1-regularization:
  - Gives regularized estimate (like ℓ_2-regularization).
  - Gives sparse estimate (like subset selection).
  - Formulated as a convex optimization.

Example: Ising Graphical Models of Binary Data

- In Ising graphical models the probability of binary variables x_i is:

$$p(\mathbf{x} \,|\mathbf{w}, \mathbf{b}) \propto \exp(\sum_{i=1}^{p} x_i b_i + \sum_{(i,j)\in E} x_i x_j w_{ij})$$

- Our goal is to estimate the weights $\{\mathbf{w}, \mathbf{b}\}$ *and* edge set E.

Example: Ising Graphical Models of Binary Data

- In Ising graphical models the probability of binary variables x_i is:

$$p(\mathbf{x} \,|\mathbf{w}, \mathbf{b}) \propto \exp(\sum_{i=1}^{p} x_i b_i + \sum_{(i,j)\in E} x_i x_j w_{ij})$$

- Our goal is to estimate the weights $\{\mathbf{w}, \mathbf{b}\}$ *and* edge set E.
- Note that $w_{ij} = 0$ is equivalent to removing (i, j) from E.

Example: Ising Graphical Models of Binary Data

- In Ising graphical models the probability of binary variables x_i is:

$$p(\mathbf{x} \,|\mathbf{w}, \mathbf{b}) \propto \exp(\sum_{i=1}^{p} x_i b_i + \sum_{(i,j)\in E} x_i x_j w_{ij})$$

- Our goal is to estimate the weights $\{\mathbf{w}, \mathbf{b}\}$ *and* edge set E.
- Note that $w_{ij} = 0$ is equivalent to removing (i, j) from E.
- So we can fit a fully connected model with ℓ_1-regularization for simultaneous parameter and structure learning:

$$\min_{\mathbf{w},\mathbf{b}} \sum_{m=1}^{M} -\log p(\mathbf{x} \,|\mathbf{w}, \mathbf{b}) + \lambda \sum_{i=1}^{p} \sum_{j=i+1}^{p} |w_{ij}|$$

Example: Ising Graphical Models of Binary Data

- In Ising graphical models the probability of binary variables x_i is:

$$p(\mathbf{x} \mid \mathbf{w}, \mathbf{b}) \propto \exp(\sum_{i=1}^{p} x_i b_i + \sum_{(i,j) \in E} x_i x_j w_{ij})$$

- Our goal is to estimate the weights $\{\mathbf{w}, \mathbf{b}\}$ *and* edge set E.
- Note that $w_{ij} = 0$ is equivalent to removing (i, j) from E.
- So we can fit a fully connected model with ℓ_1-regularization for simultaneous parameter and structure learning:

$$\min_{\mathbf{w}, \mathbf{b}} \sum_{m=1}^{M} -\log p(\mathbf{x} \mid \mathbf{w}, \mathbf{b}) + \lambda \sum_{i=1}^{p} \sum_{j=i+1}^{p} |w_{ij}|$$

- When each edge has multiple parameters, we can use *group* ℓ_1-regularization.

Limitations of Prior Work and Contributions

- Existing optimization methods are innefficient for these non-smooth, high-dimensional problems with costly objectives.

Limitations of Prior Work and Contributions

- Existing optimization methods are innefficient for these non-smooth, high-dimensional problems with costly objectives.
- Further, existing work on ℓ_1-regularization for structure learning has focused on:
  - Undirected models.
  - One-to-one correspondence between parameters and edges.
  - Pairwise potentials.

Limitations of Prior Work and Contributions

- Existing optimization methods are innefficient for these non-smooth, high-dimensional problems with costly objectives.
- Further, existing work on ℓ_1-regularization for structure learning has focused on:
  - Undirected models.
  - One-to-one correspondence between parameters and edges.
  - Pairwise potentials.

In this thesis we:

- Describe limited-memory quasi-Newton methods for optimizing high-dimensional costly objective functions with:
  - Chapter 2: ℓ_1-regularization
  - Chapter 3: Group ℓ_1-regularization

Limitations of Prior Work and Contributions

- Existing optimization methods are innefficient for these non-smooth, high-dimensional problems with costly objectives.
- Further, existing work on ℓ_1-regularization for structure learning has focused on:
  - Undirected models.
  - One-to-one correspondence between parameters and edges.
  - Pairwise potentials.

In this thesis we:

- Describe limited-memory quasi-Newton methods for optimizing high-dimensional costly objective functions with:
  - Chapter 2: ℓ_1-regularization
  - Chapter 3: Group ℓ_1-regularization
- Consider using ℓ_1-regularization for structure learning with:
  - Chapter 4: Directed acyclic graphical models.
  - Chapter 5: Multi-parameter edges and edge groups.
  - Chapter 6: Higher-order dependencies.

Outline

Optimization with ℓ_1-Regularization Problem

- We want to optimize a differentiable function $L(\mathbf{w})$ with (non-differentiable) ℓ_1-regularization:

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq L(\mathbf{w}) + \sum_i \lambda_i |w_i|$$

Optimization with ℓ_1-Regularization Problem

- We want to optimize a differentiable function $L(\mathbf{w})$ with (non-differentiable) ℓ_1-regularization:

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq L(\mathbf{w}) + \sum_i \lambda_i |w_i|$$

- We focus on the case of logistic regression.

Optimization with ℓ_1-Regularization Problem

- We want to optimize a differentiable function $L(\mathbf{w})$ with (non-differentiable) ℓ_1-regularization:

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq L(\mathbf{w}) + \sum_i \lambda_i |w_i|$$

- We focus on the case of logistic regression.
- In the maximum likelihood case, L-BFGS methods are among the most efficient.
- Methods proposed for addressing the non-differentiability are typically slower than maximum likelihood L-BFGS methods.

Adapting L-BFGS to ℓ_1-Regularization

- Can we adapt L-BFGS to solve ℓ_1-regularization problems?

Adapting L-BFGS to ℓ_1-Regularization

- Can we adapt L-BFGS to solve ℓ_1-regularization problems?
- Yes, but previous methods all lose something:
  - Algorithm may get stuck.
  - Double the number of variables.
  - Only make 1 variable non-zero at a time.
  - Iterations require more than $\mathcal{O}(p)$.
  - Iterations are not sparse.
  - Only take L-BFGS step on subset of the non-zero variables.

Adapting L-BFGS to ℓ_1-Regularization

- Can we adapt L-BFGS to solve ℓ_1-regularization problems?
- Yes, but previous methods all lose something:
  - Algorithm may get stuck.
  - Double the number of variables.
  - Only make 1 variable non-zero at a time.
  - Iterations require more than $\mathcal{O}(p)$.
  - Iterations are not sparse.
  - Only take L-BFGS step on subset of the non-zero variables.
- This work: L-BFGS method for solving ℓ_1-regularization problems without any of these disadvantages.

Projected Scaled Sub-Gradient (Gafni-Bertsekas variant)

- Basic L-BFGS step on non-zero variables $\mathcal{N}$

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})$$

Projected Scaled Sub-Gradient (Gafni-Bertsekas variant)

- Basic L-BFGS step on non-zero variables $\mathcal{N}$

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})$$

- Diagonally-scaled steepest descent step on zero variables $\mathcal{Z}$:

$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})$$

Projected Scaled Sub-Gradient (Gafni-Bertsekas variant)

- Basic L-BFGS step on non-zero variables $\mathcal{N}$

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})$$

- Diagonally-scaled steepest descent step on zero variables $\mathcal{Z}$:

$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})$$

- Project both steps onto orthant containing previous iteration:

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})]$$
$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})]$$

Projected Scaled Sub-Gradient (Gafni-Bertsekas variant)

- Basic L-BFGS step on non-zero variables $\mathcal{N}$

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})$$

- Diagonally-scaled steepest descent step on zero variables $\mathcal{Z}$:

$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})$$

- Project both steps onto orthant containing previous iteration:

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})]$$
$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})]$$

- α selected by Armijo condition along projection arc.

Projected Scaled Sub-Gradient (Gafni-Bertsekas variant)

- Basic L-BFGS step on non-zero variables $\mathcal{N}$

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})$$

- Diagonally-scaled steepest descent step on zero variables $\mathcal{Z}$:

$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})$$

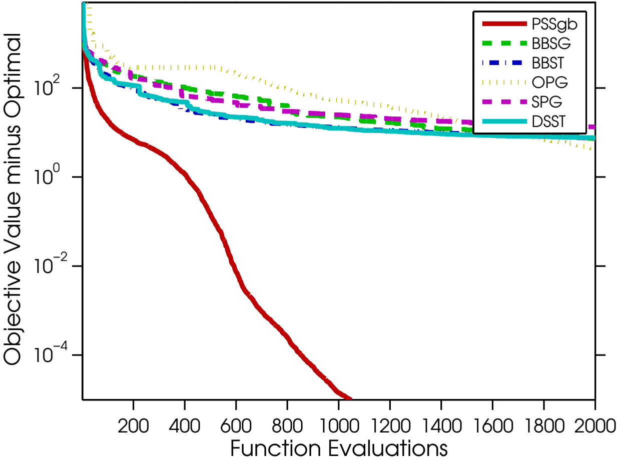
- Project both steps onto orthant containing previous iteration:

$$\mathbf{w}_{\mathcal{N}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{w}_{\mathcal{N}} - \alpha H_{\mathcal{N}}^{-1} \nabla_{\mathcal{N}} f(\mathbf{w})]$$
$$\mathbf{w}_{\mathcal{Z}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{w}_{\mathcal{Z}} - \alpha D \tilde{\nabla}_{\mathcal{Z}} f(\mathbf{w})]$$

- α selected by Armijo condition along projection arc.
- Simple method that doesn't have any of these drawbacks.
- Chapter 2 describes two other PSS methods.

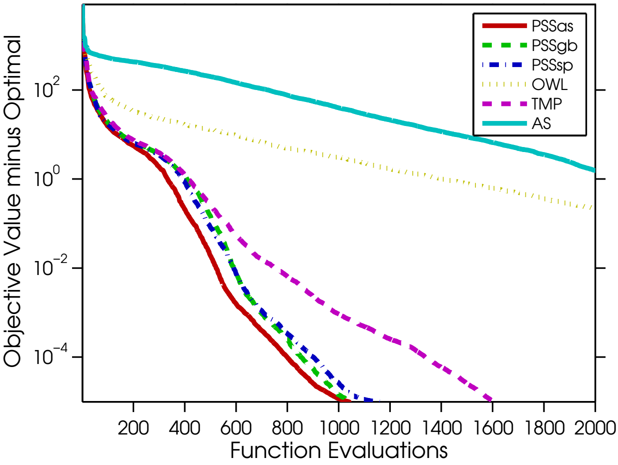
Comparing PSS methods to non-L-BFGS methods

PSS against methods not based on L-BFGS (*sido* data):



Comparing PSS methods to other L-BFGS methods

PSS against other methods based on L-BFGS (*sido* data):



Selected Extensions, Completed Work, and Future Work

- (*Completed*) PSS methods can be applied to optimize any differentiable function subject to ℓ_1-regularization:
  - Generalized linear models.
  - Huber and student t robust regression models.
  - Gaussian graphical models.
  - Ising graphical models.
  - Conditional random fields.
  - Neural networks.
  - etc.

Selected Extensions, Completed Work, and Future Work

- (*Completed*) PSS methods can be applied to optimize any differentiable function subject to ℓ_1-regularization:
  - Generalized linear models.
  - Huber and student t robust regression models.
  - Gaussian graphical models.
  - Ising graphical models.
  - Conditional random fields.
  - Neural networks.
  - etc.
- (*Future work*) Can generalize to problems of the form:

$$\min_{\mathbf{l} \preceq \mathbf{w} \preceq \mathbf{r}} L(\mathbf{w}) + R(\mathbf{w}),$$

where $R(\mathbf{w})$ is separable and each component is differentiable almost everywhere.

Outline

1. Introduction

2. Optimization with ℓ_1-Regularization

3. Optimization with Group ℓ_1-Regularization
   Schmidt, van den Berg, Friedlander, Murphy, AI-Stats 2009.

4. Directed Graphical Model Structure Learning

5. Undirected Graphical Model Structure Learning

6. Hierarchical Log-Linear Model Structure Learning

7. Discussion

Optimization with Group ℓ_1-Regularization Problem

- We now consider the more general group ℓ_1-regularization:

$$\min_{\mathbf{x}} L(\mathbf{w}) + \sum_{A} \lambda_A ||\mathbf{w}_A||_2.$$

- Non-differentiable when a whole group $\mathbf{w}_A$ is zero.

Optimization with Group ℓ_1-Regularization Problem

- We now consider the more general group ℓ_1-regularization:

$$\min_{\mathbf{x}} L(\mathbf{w}) + \sum_A \lambda_A ||\mathbf{w}_A||_2.$$

- Non-differentiable when a whole group $\mathbf{w}_A$ is zero.
- We focus on the case of discrete undirected graphical models, where function evaluations are very expensive.

Optimization with Group ℓ_1-Regularization Problem

- We now consider the more general group ℓ_1-regularization:

$$\min_{\mathbf{x}} L(\mathbf{w}) + \sum_A \lambda_A ||\mathbf{w}_A||_2.$$

- Non-differentiable when a whole group $\mathbf{w}_A$ is zero.
- We focus on the case of discrete undirected graphical models, where function evaluations are very expensive.
- We can generalize the methods of Chapter 2 that are not based on L-BFGS (SPG).
- We can't generalize the methods of Chapter 2 that are based on L-BFGS (PSS).

Optimization with Group ℓ_1-Regularization Problem

- We now consider the more general group ℓ_1-regularization:

$$\min_{\mathbf{x}} L(\mathbf{w}) + \sum_{A} \lambda_A ||\mathbf{w}_A||_2.$$

- Non-differentiable when a whole group $\mathbf{w}_A$ is zero.
- We focus on the case of discrete undirected graphical models, where function evaluations are very expensive.
- We can generalize the methods of Chapter 2 that are not based on L-BFGS (SPG).
- We can't generalize the methods of Chapter 2 that are based on L-BFGS (PSS).
- Since the methods based on L-BFGS require fewer evaluations, we want a different generalization of L-BFGS methods.

Formulating as a Constrained Optimization

- We re-write the non-smooth

$$\min_{\mathbf{w}} L(\mathbf{w}) + \sum_{A} \lambda_A ||\mathbf{w}_A||_2$$

as a differentiable optimization over a convex set:

$$\min_{\mathbf{w},\mathbf{g}} \quad L(\mathbf{w}) + \sum_{A} \lambda_A \, g_A$$
$$\text{s.t.} \quad ||\mathbf{w}_A||_2 \leq g_A, \forall_A$$

Formulating as a Constrained Optimization

- We re-write the non-smooth

$$\min_{\mathbf{w}} L(\mathbf{w}) + \sum_{A} \lambda_A ||\mathbf{w}_A||_2$$

as a differentiable optimization over a convex set:

$$\min_{\mathbf{w},\mathbf{g}} \quad L(\mathbf{w}) + \sum_{A} \lambda_A \, g_A$$

$$\text{s.t.} \quad ||\mathbf{w}_A||_2 \leq g_A, \forall_A$$

- We can efficiently project onto the feasible set:

$$\mathcal{P}(\mathbf{w}_A, g_A) = \begin{cases} (\mathbf{w}_A, g_A) & \text{if } ||\mathbf{w}_A||_2 \leq g_A \\ \frac{1+g_A/||\mathbf{w}_A||_2}{2}(\mathbf{w}_A, ||\mathbf{w}_A||_2) & \text{if } ||\mathbf{w}_A||_2 > |g_A| \\ (\mathbf{0}, 0) & \text{if } ||\mathbf{w}_A||_2 \leq -g_A \end{cases}$$

Optimizing Costly Functions with Simple Constraints

- This formulation has:
  - a large number of parameters.
  - an expensive objective function.
  - constraints on the parameters.

Optimizing Costly Functions with Simple Constraints

- This formulation has:
  - a large number of parameters.
  - an expensive objective function.
  - constraints on the parameters.
- But, projecting onto the constraints is cheap compared to evaluating the objective function.

Optimizing Costly Functions with Simple Constraints

- This formulation has:
  - a large number of parameters.
  - an expensive objective function.
  - constraints on the parameters.
- But, projecting onto the constraints is cheap compared to evaluating the objective function.
- We give a new method for problems with this structure:
  - At the outer level, L-BFGS updates build a quadratic approximation to the function.
  - At the inner level, SPG iterations approximately minimize this quadratic over the convex set.

Optimizing Costly Functions with Simple Constraints

- This formulation has:
  - a large number of parameters.
  - an expensive objective function.
  - constraints on the parameters.
- But, projecting onto the constraints is cheap compared to evaluating the objective function.
- We give a new method for problems with this structure:
  - At the outer level, L-BFGS updates build a quadratic approximation to the function.
  - At the inner level, SPG iterations approximately minimize this quadratic over the convex set.
- The inner level uses projections but not function evaluations.
- The iteration cost is still $\mathcal{O}(p)$.

Limited-Memory Projected Quasi-Newton Method

1. Use a fixed number of SPG iterations to approximately minimize the L-BFGS approximation over the convex set:

$$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}\in\mathcal{C}} f(\mathbf{w}_k)+(\mathbf{w}-\mathbf{w}_k)^T\nabla f(\mathbf{w}_k)+\frac{1}{2}(\mathbf{w}-\mathbf{w}_k)^T B_k(\mathbf{w}-\mathbf{w}_k)$$

Limited-Memory Projected Quasi-Newton Method

1. Use a fixed number of SPG iterations to approximately minimize the L-BFGS approximation over the convex set:

$$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}\in\mathcal{C}} f(\mathbf{w}_k)+(\mathbf{w}-\mathbf{w}_k)^T\nabla f(\mathbf{w}_k)+\frac{1}{2}(\mathbf{w}-\mathbf{w}_k)^T B_k(\mathbf{w}-\mathbf{w}_k)$$

2. If we initialize with $\mathbf{w}_k$, this gives a feasible descent direction

$$\mathbf{d}^k \leftarrow \mathbf{w}^* - \mathbf{w}_k$$

Limited-Memory Projected Quasi-Newton Method

1. Use a fixed number of SPG iterations to approximately minimize the L-BFGS approximation over the convex set:

$$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}\in\mathcal{C}} f(\mathbf{w}_k)+(\mathbf{w}-\mathbf{w}_k)^T\nabla f(\mathbf{w}_k)+\frac{1}{2}(\mathbf{w}-\mathbf{w}_k)^T B_k(\mathbf{w}-\mathbf{w}_k)$$

2. If we initialize with $\mathbf{w}_k$, this gives a feasible descent direction

$$\mathbf{d}^k \leftarrow \mathbf{w}^* - \mathbf{w}_k$$

3. Select $\alpha \in (0,1]$ by a backtracking line search to satisfy the Armijo condition and set:

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}_k + \alpha\mathbf{d}^k.$$

Limited-Memory Projected Quasi-Newton Method

1. Use a fixed number of SPG iterations to approximately minimize the L-BFGS approximation over the convex set:

$$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}\in\mathcal{C}} f(\mathbf{w}_k)+(\mathbf{w}-\mathbf{w}_k)^T\nabla f(\mathbf{w}_k)+\frac{1}{2}(\mathbf{w}-\mathbf{w}_k)^T B_k(\mathbf{w}-\mathbf{w}_k)$$

2. If we initialize with $\mathbf{w}_k$, this gives a feasible descent direction

$$\mathbf{d}^k \leftarrow \mathbf{w}^* - \mathbf{w}_k$$

3. Select $\alpha \in (0,1]$ by a backtracking line search to satisfy the Armijo condition and set:

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}_k + \alpha\mathbf{d}^k.$$

4. Update the L-BFGS approximation and repeat.

Limited-Memory Projected Quasi-Newton Method

1. Use a fixed number of SPG iterations to approximately minimize the L-BFGS approximation over the convex set:

$$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}\in\mathcal{C}} f(\mathbf{w}_k)+(\mathbf{w}-\mathbf{w}_k)^T\nabla f(\mathbf{w}_k)+\frac{1}{2}(\mathbf{w}-\mathbf{w}_k)^T B_k(\mathbf{w}-\mathbf{w}_k)$$

2. If we initialize with $\mathbf{w}_k$, this gives a feasible descent direction

$$\mathbf{d}^k \leftarrow \mathbf{w}^* - \mathbf{w}_k$$

3. Select $\alpha \in (0,1]$ by a backtracking line search to satisfy the Armijo condition and set:

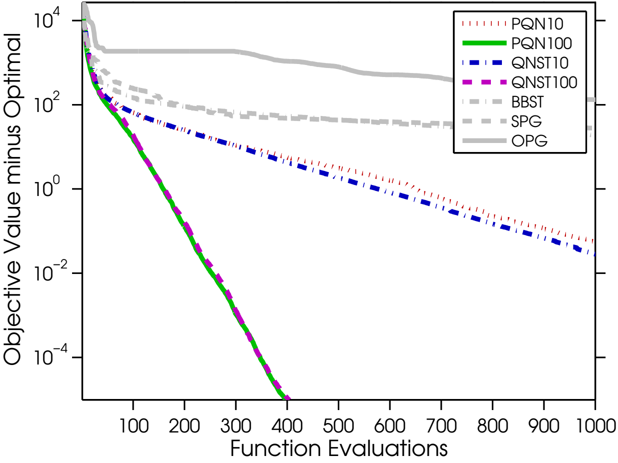
$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}_k + \alpha\mathbf{d}^k.$$

4. Update the L-BFGS approximation and repeat.

Chapter 3 describes a variant for non-smooth optimization that can directly solve group ℓ_1-regularization problems.

Comparing L-BFGS to non-L-BFGS Methods

PQN/QNST vs. methods not based on L-BFGS (*cyto* data):



Selected Extensions, Completed Work, and Future Work

- (*Completed*) PQN/QNST can be applied to optimize any differentiable function with simple constraints/regularizers:
  - Blockwise-sparse Gaussian graphical models.
  - Feature selection in conditional random fields.
  - Variational mean field.
  - Other group-norms (Chapter 5).
  - Overlapping groups (Chapter 6).
  - Etc.

Outline

Motivation for Directed Acyclic Graphical Models

- Prior work on structure learning with ℓ_1-regularization has largely focused on undirected models.
- However, it is NP-hard (or worse) to perform standard operations in general undirected graphical models.

Motivation for Directed Acyclic Graphical Models

- Prior work on structure learning with ℓ_1-regularization has largely focused on undirected models.
- However, it is NP-hard (or worse) to perform standard operations in general undirected graphical models.
- In directed acyclic graph models we can perform some operations in polynomial-time:
  - Calculate probability of a vector.
  - Generate unbiased samples.
  - Approximate arbitrary marginals.
  - Approximate some conditionals.

Motivation for Directed Acyclic Graphical Models

- Prior work on structure learning with ℓ_1-regularization has largely focused on undirected models.
- However, it is NP-hard (or worse) to perform standard operations in general undirected graphical models.
- In directed acyclic graph models we can perform some operations in polynomial-time:
  - Calculate probability of a vector.
  - Generate unbiased samples.
  - Approximate arbitrary marginals.
  - Approximate some conditionals.
- Futher, parameter independence lets us:
  - Locally estimate parameters.
  - Locally tune hyper-parameters.
  - Mix variable types.

Motivation for Directed Acyclic Graphical Models

- Prior work on structure learning with ℓ_1-regularization has largely focused on undirected models.
- However, it is NP-hard (or worse) to perform standard operations in general undirected graphical models.
- In directed acyclic graph models we can perform some operations in polynomial-time:
  - Calculate probability of a vector.
  - Generate unbiased samples.
  - Approximate arbitrary marginals.
  - Approximate some conditionals.
- Futher, parameter independence lets us:
  - Locally estimate parameters.
  - Locally tune hyper-parameters.
  - Mix variable types.
- However, enforcing acyclicity makes structure learning hard.

DAG Structure Learning given an Ordering

- We focus on DAGs with logistic regression conditional probability distributions (CPDs):

$$p(x_i | \mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i) = \frac{1}{1 + \exp(-x_i(\mathbf{w}^T \mathbf{x}_{\pi(i)} + b_i))}$$

DAG Structure Learning given an Ordering

- We focus on DAGs with logistic regression conditional probability distributions (CPDs):

$$p(x_i|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i) = \frac{1}{1 + \exp(-x_i(\mathbf{w}^T \mathbf{x}_{\pi(i)} + b_i))}$$

- Prior work focuses on using ℓ_1-regularization to fit each CPD given an ordering.
- In general we don't have an ordering, and without this the graph is unlikely to be acyclic.

DAG Structure Learning without an Ordering

State of the art methods for DAG learning without an ordering have two components:

1. **Pruning**: Use a series of (conditional) (in-)dependence tests to prune the set of possible edges.
2. **Search**: Search for a structure that optimizes a scoring criteria (BIC, validation set likelihood)

DAG Structure Learning without an Ordering

State of the art methods for DAG learning without an ordering have two components:

1. **Pruning**: Use a series of (conditional) (in-)dependence tests to prune the set of possible edges.
2. **Search**: Search for a structure that optimizes a scoring criteria (BIC, validation set likelihood)

In current methods:

- The pruning phase ignores structure in the CPDs.
- The pruning phase ignores the score.

A Hybrid Method based on ℓ_1-Regularization

- We propose the following simple method:
  1. **L1MB**: Fit each CPD with all parents and ℓ_1-regularized logistic regression, using the scoring citerion to select λ.
  2. **DAG-Search**: Search through the space of possible DAG structures, restricted to candidate edges.

A Hybrid Method based on ℓ_1-Regularization

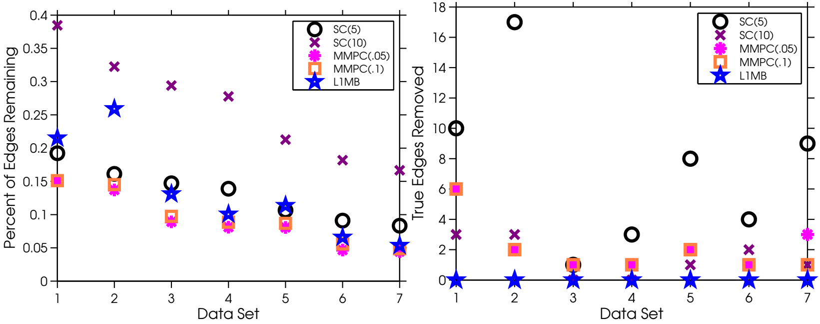
- We propose the following simple method:
  1. **L1MB**: Fit each CPD with all parents and ℓ_1-regularized logistic regression, using the scoring citerion to select λ.
  2. **DAG-Search**: Search through the space of possible DAG structures, restricted to candidate edges.
- The pruning phase uses the scoring criterion and the structure of the CPDs.

A Hybrid Method based on ℓ_1-Regularization

- We propose the following simple method:
  1. **L1MB**: Fit each CPD with all parents and ℓ_1-regularized logistic regression, using the scoring citerion to select λ.
  2. **DAG-Search**: Search through the space of possible DAG structures, restricted to candidate edges.
- The pruning phase uses the scoring criterion and the structure of the CPDs.
- Chapter 4 extends this algorithm to causal DAGs, and the Appendix gives structures for testing whether edge additions/reversals cause a cycle in $\mathcal{O}(1)$.

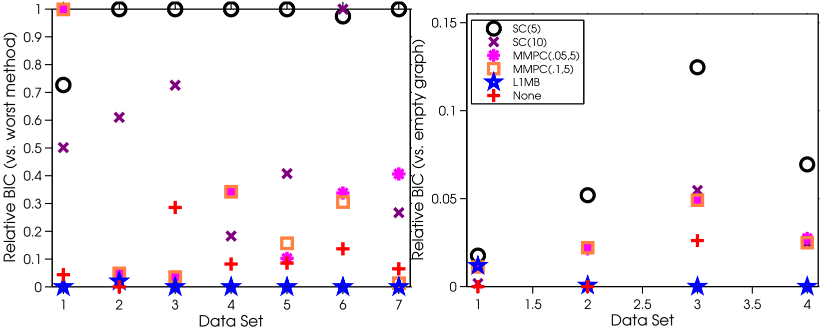
Comparing Edge Pruning Strategies

L1MB vs. other pruning strategies (5000 synthetic data samples):



Comparing DAG-Search Strategies

L1MB+DAG-search vs. other search strategies (synthetic/real data):



Selected Extensions, Completed Work, and Future Work

- (*Completed*) We can use the same procedure with other linearly-parameterized CPDs
  - Gaussian
  - Student's t
  - Probit
  - Extreme-value
  - Multinomial
  - Ordinal
  - Etc.

Selected Extensions, Completed Work, and Future Work

- (*Completed*) We can use the same procedure with other linearly-parameterized CPDs
  - Gaussian
  - Student's t
  - Probit
  - Extreme-value
  - Multinomial
  - Ordinal
  - Etc.
- (*Completed by another group*) We can replace the DAG-search with other search strategies:
  - Greedy equivalence search
  - Constrained optimal search.

Outline

1. Introduction
2. Optimization with ℓ_1-Regularization
3. Optimization with Group ℓ_1-Regularization
4. Directed Graphical Model Structure Learning
5. Undirected Graphical Model Structure Learning
   Schmidt, Murphy, Fung, Rosales, CVPR 2008.
6. Hierarchical Log-Linear Model Structure Learning
7. Discussion

Undirected Graphical Model Structure Learning

- Prior work has largely focused on sparsity in the individual parameters.
- In many scenarios we want sparsity in parameter groups:
  - In multi-state models each edge has multiple parameters.
  - In blockwise-sparse models we want sparsity in groups of edges.
  - In conditional random fields (CRFs) each edge has multiple features.
- In these cases, ℓ_1-regularization does not encourage the appropriate sparsity patterns.

Example: Multi-Parameter Edges

- In binary Ising models, each edge has only one parameter:

$$\log \phi_{ij}(x_i, x_j) = x_i x_j w_{ij}$$

- In multi-state models, each edge can have multiple parameters:

$$\begin{aligned}\log \phi_{ij}(x_i, x_j) = \mathbb{I}(x_i = 1, x_j = 1)w_{ij11} + \mathbb{I}(x_i = 1, x_j = 2)w_{ij12} + \mathbb{I}(x_i = 1, x_j = 3)w_{ij13} \\ + \mathbb{I}(x_i = 2, x_j = 1)w_{ij21} + \mathbb{I}(x_i = 2, x_j = 2)w_{ij22} + \mathbb{I}(x_i = 2, x_j = 3)w_{ij23} \\ + \mathbb{I}(x_i = 3, x_j = 1)w_{ij31} + \mathbb{I}(x_i = 3, x_j = 2)w_{ij32} + \mathbb{I}(x_i = 3, x_j = 3)w_{ij23},\end{aligned}$$

- Removing the edge is equivalent to setting all edge parameters to zero.

Different Choices of Norm

- With multi-parameter edges, we can encourage graphical sparsity with group ℓ_1-regularization:

$$\min_{\mathbf{w},\mathbf{b}} -\sum_{m=1}^{n} \log p(\mathbf{x}^m; \mathbf{w}, \mathbf{b}) + \lambda \sum_{i=1}^{p} \sum_{j=i+1}^{p} ||\mathbf{w}_{ij}||_2$$

Different Choices of Norm

- With multi-parameter edges, we can encourage graphical sparsity with group ℓ_1-regularization:

$$\min_{\mathbf{w},\mathbf{b}} - \sum_{m=1}^{n} \log p(\mathbf{x}^m; \mathbf{w}, \mathbf{b}) + \lambda \sum_{i=1}^{p} \sum_{j=i+1}^{p} ||\mathbf{w}_{ij}||_2$$

- We can also consider different choices of the group norm

Different Choices of Norm

- With multi-parameter edges, we can encourage graphical sparsity with group ℓ_1-regularization:

$$\min_{\mathbf{w},\mathbf{b}} -\sum_{m=1}^{n} \log p(\mathbf{x}^m; \mathbf{w}, \mathbf{b}) + \lambda \sum_{i=1}^{p} \sum_{j=i+1}^{p} ||\mathbf{w}_{ij}||_2$$

- We can also consider different choices of the group norm
- Different choices encourage structure in the edge potentials:
  - The ℓ_∞ norm encourages parameter tieing.
  - The *nuclear* norm encourages low rank.

Optimization with General Norms

- The optimization methods of Chapter 3 can easily be extended to use a general norm:

$$\min_{\mathbf{x}} L(\mathbf{x}) + \sum_{A} \lambda_A ||\mathbf{x}_A||_p.$$

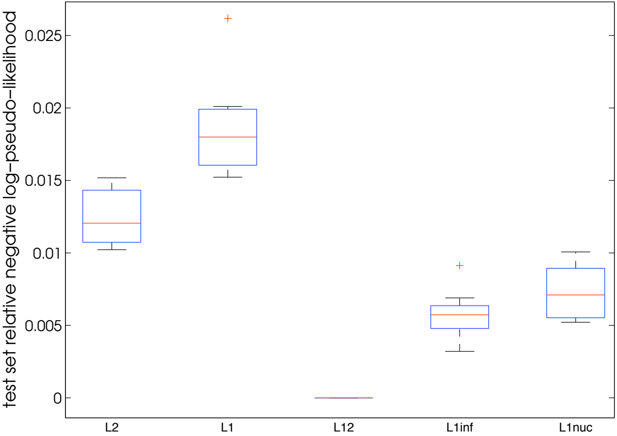
- The corresponding constrained formulation:

$$\min_{\mathbf{x},\mathbf{g}} \; L(\mathbf{x}) + \sum_{A} \lambda_A \, g_A, \;\; \text{subject to} \;\; g_A \geq ||\mathbf{x}_A||_p, \forall_A.$$

- For the ℓ_∞ norm, the projection can be computed in $\mathcal{O}(|A| \log |A|)$ using sorting.
- For the nuclear norm, the projection can be computed in $\mathcal{O}(|A|^{3/2})$ using SVD.

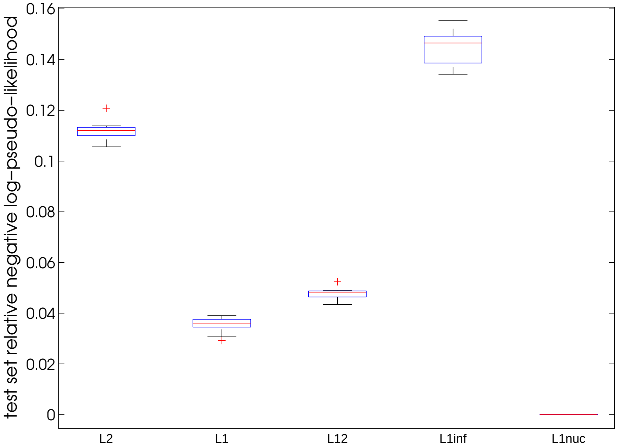
Comparison of Different Norms

Comparing regularization types on *traffic* data



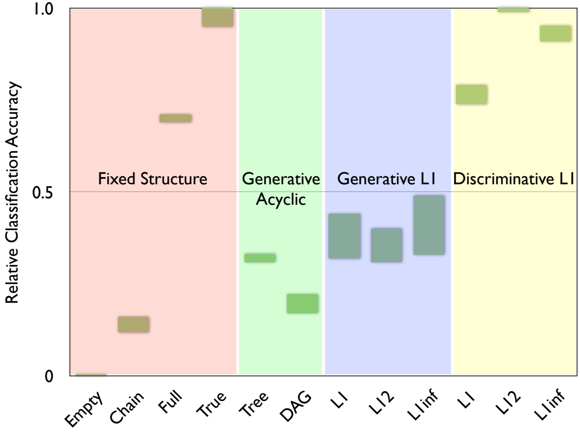
Comparison of Different Norms

Comparing regularization types on *usps8* data:



Comparing Methods for CRF Structure Learning

Comparing CRF structure learning methods (*synthetic* data):



Selected Extensions, Completed Work, and Future Work

- (*Completed*) We can use these ideas in more advanced scenarios:
  - Learn conditional graphical sparsity with binary features.
  - Learn the variables types in blockwise-sparse models.
  - Causal learning with interventional potentials/nodes.

Selected Extensions, Completed Work, and Future Work

- (*Completed*) We can use these ideas in more advanced scenarios:
  - Learn conditional graphical sparsity with binary features.
  - Learn the variables types in blockwise-sparse models.
  - Causal learning with interventional potentials/nodes.
- (*Future Work*) Could use more advanced approximate objectives:
  - Block pseudo-likelihood
  - More advanced variational methods

Outline

1. Introduction
2. Optimization with ℓ_1-Regularization
3. Optimization with Group ℓ_1-Regularization
4. Directed Graphical Model Structure Learning
5. Undirected Graphical Model Structure Learning
6. Hierarchical Log-Linear Model Structure Learning
   Schmidt and Murphy, AI-Stats 2010.
7. Discussion

General Log-Linear Model Structure Learning

- Nearly all of the prior work on using ℓ_1-regularization for structure learning has focused on pairwise models.
- For some data sets, higher-order interactions may be important.

General Log-Linear Model Structure Learning

- Nearly all of the prior work on using ℓ_1-regularization for structure learning has focused on pairwise models.
- For some data sets, higher-order interactions may be important.
- We could consider learning general log-linear models using

$$\min_{\mathbf{w}} -\sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A ||\mathbf{w}_A||_2$$

General Log-Linear Model Structure Learning

- Nearly all of the prior work on using ℓ_1-regularization for structure learning has focused on pairwise models.
- For some data sets, higher-order interactions may be important.
- We could consider learning general log-linear models using

$$\min_{\mathbf{w}} -\sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A ||\mathbf{w}_A||_2$$

- However, the exponential number of variables makes this difficult without a severe cardinality restriction.
- Further, group sparsity does *not* correspond to conditional independence.

Hierarchical Log-Linear Model Structure Learning

- We consider an alternative to a cardinality restriction:
  - **Hierarchical Inclusion Restriction**: If $\mathbf{w}_A = \mathbf{0}$ and $A \subset B$, then $\mathbf{w}_B = \mathbf{0}$.
- The class of hierarchical log-linear models.

Hierarchical Log-Linear Model Structure Learning

- We consider an alternative to a cardinality restriction:
  - **Hierarchical Inclusion Restriction**: If $\mathbf{w}_A = \mathbf{0}$ and $A \subset B$, then $\mathbf{w}_B = \mathbf{0}$.
- The class of hierarchical log-linear models.
- Allows interactions of any order.
- Group sparsity corresponds to conditional independence.
- But, imposes sparsity constraints that can't be obtained using disjoint group ℓ_1-regularization.

Encouraging Hierarchical Sparsity

- However, we can encourage hierarchical sparsity using overlapping group ℓ_1-regularization.

Encouraging Hierarchical Sparsity

- However, we can encourage hierarchical sparsity using overlapping group ℓ_1-regularization.
- We can encourage the solution to be a hierarchical using:

$$\min_{\mathbf{w}} - \sum_{i=1}^{n} \log p(\mathbf{x}^i | \mathbf{w}) + \sum_{A \subseteq S} \lambda_A \Big(\sum_{\{B | A \subseteq B\}} ||\mathbf{w}_B||_2^2 \Big)^{1/2}.$$

Encouraging Hierarchical Sparsity

- However, we can encourage hierarchical sparsity using overlapping group ℓ_1-regularization.
- We can encourage the solution to be a hierarchical using:

$$\min_{\mathbf{w}} - \sum_{i=1}^{n} \log p(\mathbf{x}^i | \mathbf{w}) + \sum_{A \subseteq S} \lambda_A \Big(\sum_{\{B | A \subseteq B\}} ||\mathbf{w}_B||_2^2 \Big)^{1/2}.$$

- We can extend the methods of Chapter 3 to solve overlapping group ℓ_1-regularization problems using Dykstra's cyclic projection algorithm.

Hierarchical Search for Hierarchical Models

- We still have an exponential number of variables to consider.
- But we know the solution is hierarchical.

Hierarchical Search for Hierarchical Models

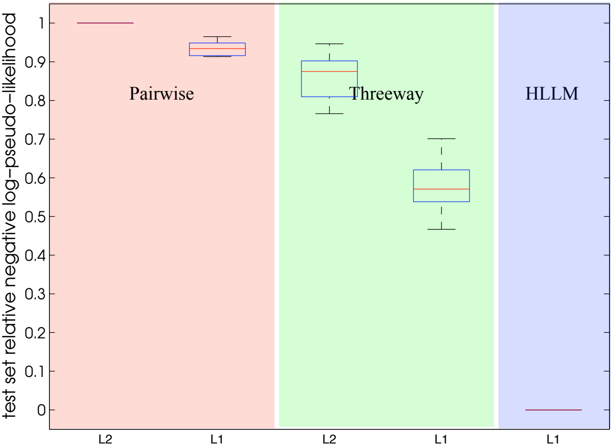
- We still have an exponential number of variables to consider.
- But we know the solution is hierarchical.
- We propose a heuristic search through the space of hierarchical models:
  1. Find non-zero groups, and other groups that satisfy hierarchical inclusion and violate optimality conditions.
  2. Solve the problem with respect to these groups.
  3. Repeat.

Hierarchical Search for Hierarchical Models

- We still have an exponential number of variables to consider.
- But we know the solution is hierarchical.
- We propose a heuristic search through the space of hierarchical models:
  1. Find non-zero groups, and other groups that satisfy hierarchical inclusion and violate optimality conditions.
  2. Solve the problem with respect to these groups.
  3. Repeat.
- This procedure converges to a solution satisfying necessary optimality conditions, and a weak form of sufficient optimality conditions.

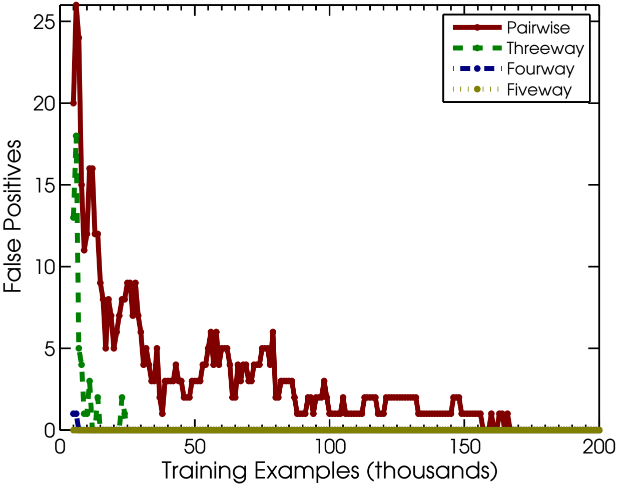
Experiments with Different Orders

Experiments on *traffic* data with models of different orders:



Experiments on Structure Learning

False positives of different orders for data generated from (1)(2,3)(4,5,6)(7,8,9,10):



Selected Extensions, Completed Work, and Future Work

- (*Future Work*) We can apply the methods in more general scenarios:
  - Conditional hierarchical log-linear models.
  - Interventional hierarchical log-linear models.

Selected Extensions, Completed Work, and Future Work

- (*Future Work*) We can apply the methods in more general scenarios:
  - Conditional hierarchical log-linear models.
  - Interventional hierarchical log-linear models.
- (*Future Work*) We can modify the search to satisfy stronger sufficient optimality conditions:
  - Test optimality conditions for an extended boundary.

Outline

Other Selected Extensions

Some topics not discussed in main body:

- The methods can be extended to handle missing data or hidden variables.
- We can consider mixtures of sparse graphical models.
- We can use projection and stochastic approximation to allow stochastic inference methods.
- Methods can be applied to other types of structure learning, such as chain graphs and relational models.
- Methods can be useful as sub-routines for variational Bayesian methods.
- Code is on-line (or will be soon).

Summary of Contributions

- **Chapter 2**: Limited-memory quasi-Newton methods for ℓ_1-regularization with several appealing properties.
- **Chapter 3**: Limited-memory quasi-Newton methods for optimizing costly functions with simple constraints or regularizers.
- **Chapter 4**: Edge pruning strategy for linearly-parameterized DAG structure learning based on ℓ_1-regularization that takes advantage of the structure of the CPDs and the score.
- **Chapter 5**: Different choices of the group norm (including nuclear norm) for multi-parameter, blockwise-sparse, and conditional undirected graphical models, the latter is the first structured classification method that simultaneously and discriminatively learns structure and parameters.
- **Chapter 6**: Overlapping group ℓ_1-regularization formulation for learning hierarchical log-linear models (with no restriction on the cardinality of the potentials), and an active set method for searching the exponential space of higher-order potentials.

Other Work

- (Vishwanathan et al., ICML 2006): Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods.
- (Carbonetto et al., NIPS 2008): An interior-point stochastic approximation method and an ℓ_1-regularized delta rule.
- (van den Berg et al., TR 2008): Group Sparsity via Linear-Time Projection.
- (Cobzas and Schmidt, CVPR 2009): Increased Discrimination in Level Set Methods with Embedded Conditional Random Fields.
- (Marlin et al., UAI 2009): Group Sparse Priors for Covariance Estimation.
- (Schmidt and Murphy, UAI 2009): Modeling Discrete Interventional Data using Directed Cyclic Graphical Models.
- (Duvenaud et al., JMLR W&CP 2010): Causal Learning without DAGs.
- (Yan et al., AI-Stats 2010): Modeling annotator expertise: Learning when everybody knows a bit of something.