First-Order Optimization Algorithms for Machine Learning

Structured Regularization

Mark Schmidt

University of British Columbia

Summer 2020

Last Time: Proximal-Gradient

- We discussed proximal-gradient methods for problems of the form

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \underbrace{f(w)}_{\text{smooth}} + \underbrace{r(w)}_{\text{simple}},$$

  where specifically $f \in C^1$ and r is convex.
- These methods use the iteration

$$w^{k+\frac{1}{2}} = w^k - \alpha_k \nabla f(w^k) \qquad \text{(gradient step)}$$
$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 + \alpha_k r(v) \right\} \qquad \text{(proximal step)}$$

- Examples of simple functions include:
  - L1-regularization.
  - Group L1-regularization (today).
- Proximal operators for these cases are soft-thresholds: sets variables/groups to 0.

Motivation for Group Sparsity

- Recall that multi-class logistic regression uses
$$\hat{y}^i = \operatorname*{argmax}_c \{w_c^\top x^i\},$$
where we have a parameter vector w_c for each class c.
- We typically use softmax loss and write our parameters as a matrix,
$$W = \begin{bmatrix} | & | & | & & | \\ w_1 & w_2 & w_3 & \cdots & w_k \\ | & | & | & & | \end{bmatrix}$$
- Suppose we want to use L1-regularization for feature selection,
$$\operatorname*{argmin}_{W \in \mathbb{R}^{d \times k}} \underbrace{f(W)}_{\text{softmax loss}} + \underbrace{\lambda \sum_{c=1}^{k} \|w_c\|_1}_{\text{L1-regularization}}.$$
- Unfortunately, setting elements of W to zero may not select features.

Motivation for Group Sparsity

- Suppose L1-regularization gives a sparse W with a non-zero in each row:

$$W = \begin{bmatrix} -0.83 & 0 & 0 & 0 \\ 0 & 0 & 0.62 & 0 \\ 0 & 0 & 0 & -0.06 \\ 0 & 0.72 & 0 & 0 \end{bmatrix}.$$

- Even though it's very sparse, it uses all features.
    - Remember that classifier multiplies feature j by each value in row j.
    - Feature 1 is used in w_1.
    - Feature 2 is used in w_3.
    - Feature 3 is used in w_4.
    - Feature 4 is used in w_2.
- In order to remove a feature, we need its entire row to be zero.

Motivation for Group Sparsity

- What we want is group sparsity:

$$W = \begin{bmatrix} -0.77 & 0.04 & -0.03 & -0.09 \\ 0 & 0 & 0 & 0 \\ 0.04 & -0.08 & 0.01 & -0.06 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

- Each row is a group, and we want groups (rows) of variables that have all zeroes.
  - If row j is zero, then x_j is not used by the model.
- Pattern arises in other settings where each row gives parameters for one feature:
  - Multiple regression, multi-label classification, and multi-task classification.

Motivation for Group Sparsiy

- Categorical features are another setting where group sparsity is needed.
- Consider categorical features encoded as binary indicator features ("1 of k"):

City	Age
Vancouver	22
Burnaby	35
Vancouver	28

Vancouver	Burnaby	Surrey	Age ≤ 20	20 < Age ≤ 30	Age > 30
1	0	0	0	1	0
0	1	0	0	0	1
1	0	0	0	1	0

- A linear model would use

$$\hat{y}^i = w_1 x_{\text{van}} + w_2 x_{\text{bur}} + w_3 x_{\text{sur}} + w_4 x_{\leq 20} + w_5 x_{21-30} + w_6 x_{>30}.$$

- If we want feature selection of original categorical variables, we have 2 groups:
  - $\{w_1, w_2, w_3\}$ correspond to "City" and $\{w_4, w_5, w_6\}$ correspond to "Age".

Group L1-Regularization

- Consider a problem with a set of disjoint groups $\mathcal{G}$.
  - For example, $\mathcal{G} = \{\{1,2\},\{3,4\}\}$.

- Minimizing a function f with group L1-regularization:

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|_p,$$

where g refers to individual group indices and $\|\cdot\|_p$ is some norm.

- For certain norms, it encourages sparsity in terms of groups g.
  - Variables x_1 and x_2 will either be both zero or both non-zero.
  - Variables x_3 and x_4 will either be both zero or both non-zero.

Group L1-Regularization

- Why is it called group L1-regularization?
- Consider $G = \{\{1,2\},\{3,4\}\}$ and using L2-norm,

$$\sum_{g\in G} \|w_g\|_2 = \sqrt{w_1^2 + w_2^2} + \sqrt{w_3^2 + w_4^2}.$$

- If vector v contains the group norms, it's the L1-norm of v:

$$\text{If } v \triangleq \begin{bmatrix} \|w_{12}\|_2 \\ \|w_{34}\|_2 \end{bmatrix} \text{ then } \sum_{g\in G} \|w_g\|_2 = \|w_{12}\|_2 + \|w_{34}\|_2 = v_1 + v_2 = |v_1| + |v_2| = \|v\|_1.$$

- So group L1-regularization encourages sparsity in the group norms.
  - When the norm of the group is 0, all group elements are 0.

Group L1-Regularization: Choice of Norm

- The group L1-regularizer is sometimes written as a "mixed" norm,

$$\|w\|_{1,p} \triangleq \sum_{g\in\mathcal{G}} \|w_g\|_p.$$

- The most common choice for the norm is the L2-norm:
  - If $\mathcal{G} = \{\{1,2\},\{3,4\}\}$ we obtain

$$\|w\|_{1,2} = \sqrt{w_1^2 + w_2^2} + \sqrt{w_3^2 + w_4^2}.$$

- Another common choice is the L∞-norm,

$$\|w\|_{1,\infty} = \max\{|w_1|,|w_2|\} + \max\{|w_3|,|w_4|\}.$$

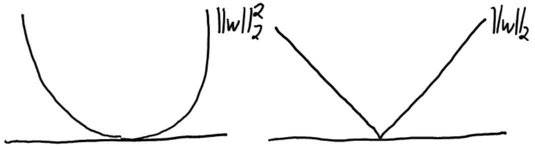
- But note that the L1-norm does not give group sparsity,

$$\|w\|_{1,1} = |w_1| + |w_2| + |w_3| + |w_4| = \|w\|_1,$$

as it's equivalent to non-group L1-regularization.

Sparsity from the L2-Norm?

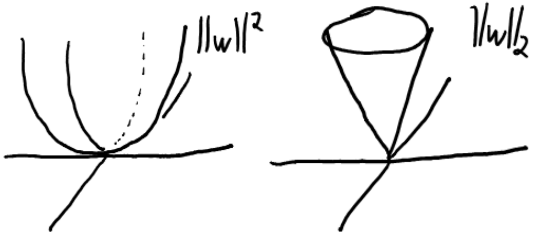
- Didn't we say sparsity comes from the L1-norm and not the L2-norm?
  - Yes, but we were using the squared L2-norm.
- Squared vs. non-squared L2-norm in 1D:



- Non-squared L2-norm is absolute value.
  - Non-squared L2-regularizer will set all $w_j = 0$ for some finite λ.
- Squaring the L2-norm gives a smooth function but destroys sparsity.

Sparsity from the L2-Norm?

- Squared vs. non-squared L2-norm in 2D:



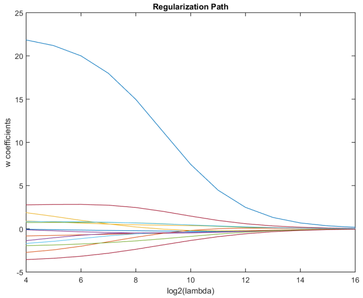
- The squared L2-norm is smooth and has no sparsity.
- Non-squared L2-norm is non-smooth at the zero vector.
  - It doesn't encourage us to set any $w_j = 0$ as long as one $w_{j'} \neq 0$.
  - But if λ is large enough it encourages all w_j to be set to 0.

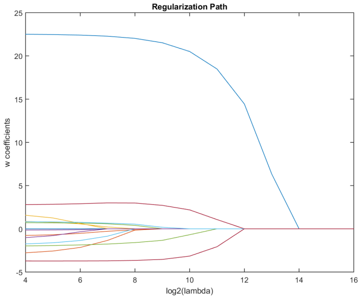
L2 and L1 Regularization Paths

- The regularization path is the set of w values as λ varies,

$$w^\lambda = \underset{w \in \mathbb{R}^d}{\text{argmin}}\, f(w) + \lambda r(w),$$

- Squared L2-regularization path vs. L1-regularization path:





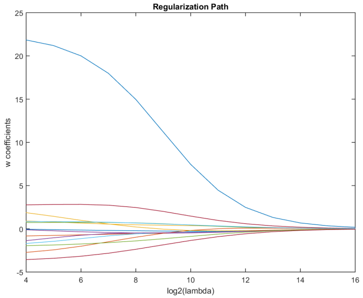
- With $r(w) = \|w\|^2$, each w_j gets close to 0 but is never exactly 0.
- With $r(w) = \|w\|_1$, each w_j gets set to exactly zero for a finite λ.

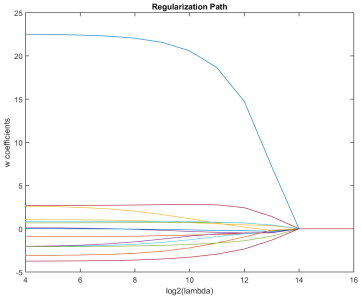
L2^2 and L2 Regularization Paths

- The regularization path is the set of w values as λ varies,

$$w^\lambda = \underset{w \in \mathbb{R}^d}{\text{argmin}}\, f(w) + \lambda r(w),$$

- Squared L2-regularization path vs. non-squared path:

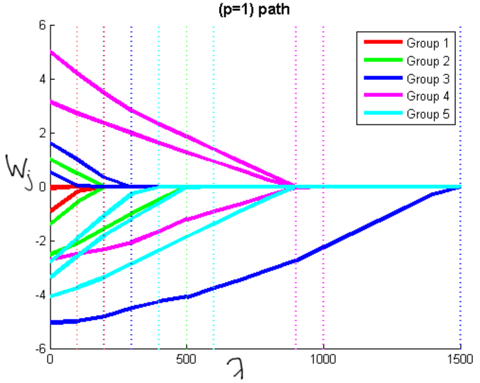


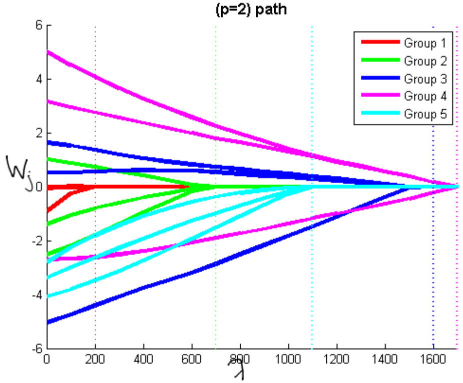


- With $r(w) = \|w\|^2$, each w_j gets close to 0 but is never exactly 0.
- With $r(w) = \|w\|_2$, all w_j get set to exactly zero for same finite λ.

Group L1-Regularization Paths

- The regularization path for group L1-regularizaiton for different p values:

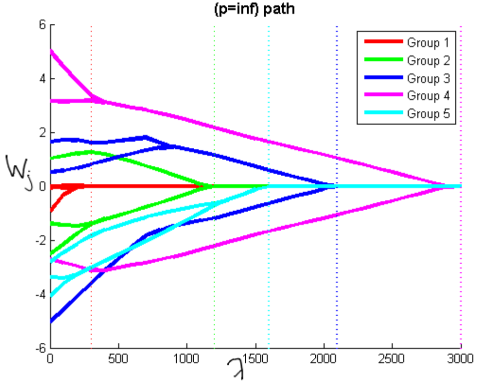


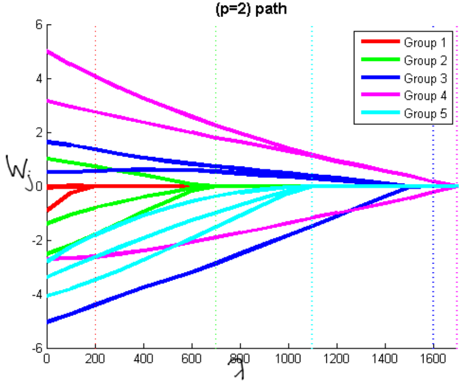


- With $p = 1$ there is no grouping effect.
- With $p = 2$ the groups become zero at the same time.

Group L1-Regularization Paths

- The regularization path for group L1-regularizaiton for different p values:





- With $p = 1$ there is no grouping effect.
- With $p = 2$ the groups become zero at the same time.
- With $p = \infty$ the groups converge to same magnitude which then goes to 0.

Sub-differential of Group L1-Regularization

- For our group L1-regularization objective with the 2-norm,

$$F(w) = f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|_2,$$

  the indices g in the sub-differential are given by

$$\partial_g F(w) \equiv \nabla_g f(w) + \lambda \partial \|w_g\|_2.$$

- In order to have $0 \in \partial F(w)$, we thus need for each group that

$$0 \in \nabla_g f(w) + \lambda \partial \|w_g\|_2,$$

  and subtracting $\nabla_g f(w)$ from both sides gives

$$-\nabla_g f(w) \in \lambda \partial \|w_g\|_2.$$

Sub-differential of Group L1-Regularization

- So at minimizer w^* we must have for all groups that

$$-\nabla_g f(w^*) \in \lambda \partial \|w_g^*\|_2.$$

- The sub-differential of the scaled L2-norm is given by the "signum" function,

$$\partial\|w\|_2 = \begin{cases} \left\{\frac{w}{\|w\|_2}\right\} & w \neq 0 \\ \{v \mid \|v\|_2 \leq 1\} & w = 0. \end{cases}$$

- So at a solution w^* we have for each group that

$$\begin{cases} -\nabla_g f(w^*) = \lambda \frac{w_g^*}{\|w_g^*\|_2} & w_g \neq 0, \\ \|\nabla_g f(w^*)\| \leq \lambda & w_g^* = 0. \end{cases}$$

- For sufficiently-large λ we'll set the group to zero.
  - With squared group norms we would need $\nabla_g f(w^*) = 0$ with $w_g^* = 0$ (unlikely).

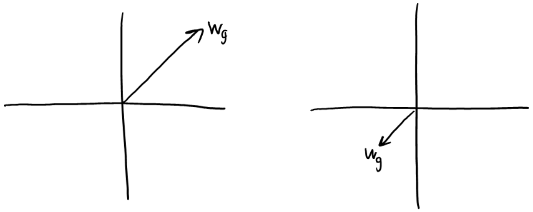
Proximal-Gradient for Group L1-Regularization

- The proximal operator for group L1-regularization,

$$\operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - w\|^2 + \alpha_k \lambda \sum_{g \in G} \|v\|_2 \right\},$$

applies a soft-threshold group-wise,

$$w_g \leftarrow \frac{w_g}{\|w_g\|_2} \max\{0, \|w_g\|_2 - \alpha_k \lambda\}.$$



- So we can solve group L1-regularization problems as fast as smooth problems.

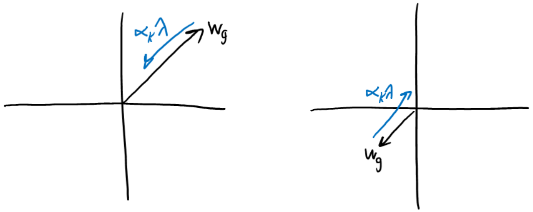
Proximal-Gradient for Group L1-Regularization

- The proximal operator for group L1-regularization,

$$\underset{v\in\mathbb{R}^d}{\text{argmin}}\left\{\frac{1}{2}\|v-w\|^2+\alpha_k\lambda\sum_{g\in G}\|v\|_2\right\},$$

  applies a soft-threshold group-wise,

$$w_g \leftarrow \frac{w_g}{\|w_g\|_2}\max\{0,\|w_g\|_2-\alpha_k\lambda\}.$$



- So we can solve group L1-regularization problems as fast as smooth problems.

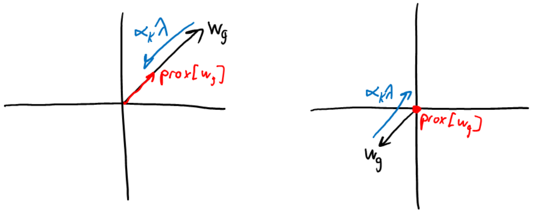
Proximal-Gradient for Group L1-Regularization

- The proximal operator for group L1-regularization,

$$\operatorname*{argmin}_{v\in\mathbb{R}^d}\left\{\frac{1}{2}\|v-w\|^2+\alpha_k\lambda\sum_{g\in G}\|v\|_2\right\},$$

applies a soft-threshold group-wise,

$$w_g \leftarrow \frac{w_g}{\|w_g\|_2}\max\{0,\|w_g\|_2-\alpha_k\lambda\}.$$



- So we can solve group L1-regularization problems as fast as smooth problems.

Outline

Structured Regularization

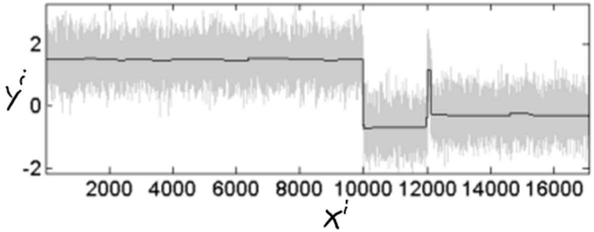
- There are many other patterns that regularization can encourage.
  - We'll call this structured regularization.
- The three most common cases:
  - Total-variation regularization encourages slow/sparse changes in w.
  - Nuclear-norm regularization encourages sparsity in rank of matrices.
  - Structured sparsity encourages sparsity in variable patterns.

Total-Variation Regularization

- 1D total-variation regularization ("fused LASSO") takes the form

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} f(w) + \lambda \sum_{j=1}^{d-1} |w_j - w_{j+1}|.$$

- Encourages consecutive parameters to have same value.
- Often used for time-series or sequence data.



http://statweb.stanford.edu/~bjk/regreg/examples/fusedlassoapprox.html

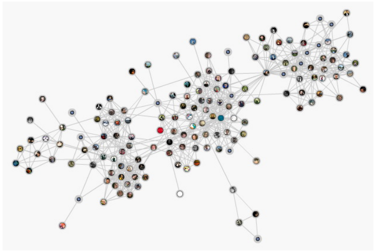
Here we're trying to estimate de-noised w_i of y^i at each time x^i.

Total-Variation Regularization

- More generally, we could penalizes differences on general graph between variables.
- An example is social regularization in recommeder systems:
  - Penalizing the difference between your parameters and your friends' parameters.

$$\operatorname*{argmin}_{W \in \mathbb{R}^{d \times k}} f(W) + \lambda \sum_{(i,j) \in \text{Friends}} \|w_i - w_j\|^2.$$

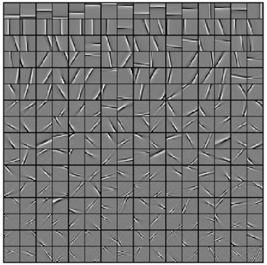
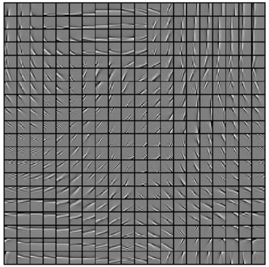
  - Typically use L2-regularization (we aren't aiming for identical parameters).



http://asawicki.info/news_1453_social_graph_-_application_on_facebook.html

Total-Variation Regularization

- Consider applying latent factor models (from 340) on image patches.
  - Similar to learning first layer of convolutional neural networks.
- Latent-factors discovered on patches with/without TV regularization.
  - Encouraging neighbours in a spatial grid to have similar filters.

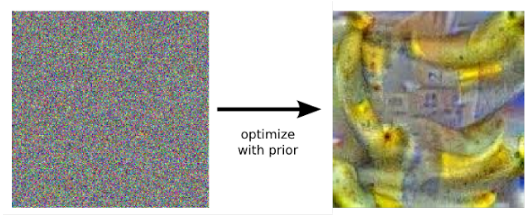



http://lear.inrialpes.fr/people/mairal/resources/pdf/review_sparse_arxiv.pdf

- Similar to “cortical columns” theory of visual cortex.

Total-Variation Regularization

- Another application is inceptionism.



https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

- Find image x that causes strongest activation of class c in neural network.

$$\operatorname*{argmin}_x f(v_c^\top h(W^{(m)}h(W^{(m-1)}\cdots h(W^{(1)}x) + \lambda \sum_{(x_i,x_j)\in\text{neigh.}} (x_i - x_j)^2,$$

- Total variation based on neighbours in image (needed to get interpretable images).

Nuclear Norm Regularization

- With matrix parameters an alternative is nuclear norm regularization,

$$\operatorname*{argmin}_{W \in \mathbb{R}^{d \times k}} f(W) + \lambda \|W\|_*,$$

  where $\|W\|_*$ is the sum of singular values.

- "L1-regularization of the singular values".
  - Encourages parameter matrix to have low-rank.

- Consider a multi-class logistic regression with a huge number of features/labels,

$$W = \begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix} = UV^\top, \quad \text{with} \quad U = \begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix}, V = \begin{bmatrix} | & | \\ v_1 & v_2 \\ | & | \end{bmatrix},$$

  U and V can be much smaller, and $XW = (XU)V^\top$ can be computed faster:
  - $O(ndk)$ cost reduced to $O(ndr + nkr)$ for rank r, much faster if $r < \min\{d, k\}$.

Structured Sparsity

- Structured sparsity is variation on group L1-regularization,

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} f(w) + \sum_{g \in \mathcal{G}} \lambda_g \|w_g\|_p,$$

  where now the groups g can overlap.

- Why is this interesting?
  - Consider the case of two groups, $\{1\}$ and $\{1,2\}$,

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} f(w) + \lambda_1 |w_1| + \lambda_2 \sqrt{w_1^2 + w_2^2}.$$

  - This encourages 3 non-zero "patterns": $\{\}$, $\{w_2\}$, $\{w_1, w_2\}$.
    - "You can only take w_1 if you've already taken w_2."

  - If $w_1 \neq 0$, the third term is smooth and doesn't encourage w_2 to be zero.
  - If $w_2 \neq 0$, we still pay a λ_1 penalty for making w_1 non-zero.
  - We can use this type of "ordering" to impose patterns on our sparsity.

Structured Sparsity

- Consider a problem with matrix parameters W.
- We want W to be "band-limited":
  - Non-zeroes only on the main diagonals.

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & w_{23} & w_{24} & 0 & 0 & 0 \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & 0 & 0 \\ 0 & w_{42} & w_{43} & w_{44} & w_{45} & w_{46} & 0 \\ 0 & 0 & w_{53} & w_{54} & w_{55} & w_{56} & w_{57} \\ 0 & 0 & 0 & w_{64} & w_{65} & w_{66} & w_{67} \\ 0 & 0 & 0 & 0 & w_{75} & w_{76} & w_{77} \end{bmatrix}.$$

  - This makes many computations much faster.
- We can enforce this with structured sparsity:
  - Only allow non-zeroes on ± 1 diagonal if you are non-zero on main diagonal.
  - Only allow non-zeroes on ± 2 diagonal if you are non-zero on ± 1 diagonal.
  - Only allow non-zeroes on ± 3 diagonal if you are non-zero on ± 2 diagonal.

Structured Sparsity

- Consider a linear model with higher-order terms,

$$\hat{y}^i = w_0 + w_1 x_1^i + w_2 x_2^i + w_3 x_3^i + w_{12} x_1^i x_2^i + w_{13} x_1^i x_3^i + w_{23} x_2^i x_3^i + w_{123} x_1^i x_2^i x_3^i.$$

  - If d is non-trivial, then the number of higher-order terms is too large.

- We can use structured sparsity to enforce a hierarchy.
  - We only allow $w_{12} \neq 0$ if $w_1 \neq 0$ and $w_2 \neq 0$.
    - You can enforce this using the groups $\{\{w_{12}\}, \{w_1, w_{12}\}, \{w_2, w_{12}\}\}$:

$$\underset{w}{\operatorname{argmin}} f(w) + \lambda_{12}|w_{12}| + \lambda_1 \sqrt{w_1^2 + w_{12}^2} + \lambda_2 \sqrt{w_2^2 + w_{12}^2}.$$

Structured Sparsity

- We can use structured sparsity to enforce a hierarchy.
  - We only allow $w_{12} \neq 0$ if $w_1 \neq 0$ and $w_2 \neq 0$.
  - We only allow $w_{123} \neq 0$ if $w_{12} \neq 0$, $w_{13} \neq 0$, and $w_{23} \neq 0$.
  - We only allow $w_{1234} \neq 0$ if all threeway interactions are present.

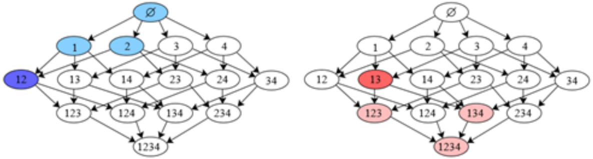


Fig 9: Power set of the set $\{1, \dots, 4\}$: in blue, an authorized set of selected subsets. In red, an example of a group used within the norm (a subset and all of its descendants in the DAG).

http://arxiv.org/pdf/1109.2397v2.pdf

- For certain bases, you can work with the full hierarchy in polynomial time.
  - Otherwise, a heuristic is to gradually "grow" the set of allowed bases.

Structured Sparsity

- Structured sparsity encourages zeroes to be any intersections of groups.
  - Possible non-zeroes are given by $\cap_{g \in \mathcal{G}'} g^c$ for all $\mathcal{G}' \subseteq \mathcal{G}$.
    - Equivalently, the set of zeroes is any $\cup_{g \in \mathcal{G}'} g$.
  - Our first example used $\{1\}$ and $\{1, 2\}$ so possible non-zeroes $\{\}$, $\{2\}$, or $\{1, 2\}$.
    - E.g., $\{2\}$ is $\{1, 2\} \cap \{1\}^c = \{1, 2\} \cap \{2\}$.
- Example is enforcing convex non-zero patterns:

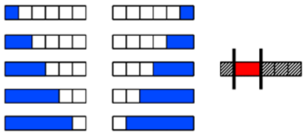
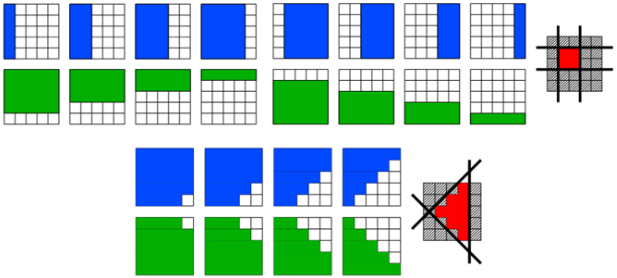


Fig 3: (Left) The set of blue groups to penalize in order to select contiguous patterns in a sequence. (Right) In red, an example of such a nonzero pattern with its corresponding zero pattern (hatched area).

https://arxiv.org/pdf/1109.2397v2.pdf

Structured Sparsity

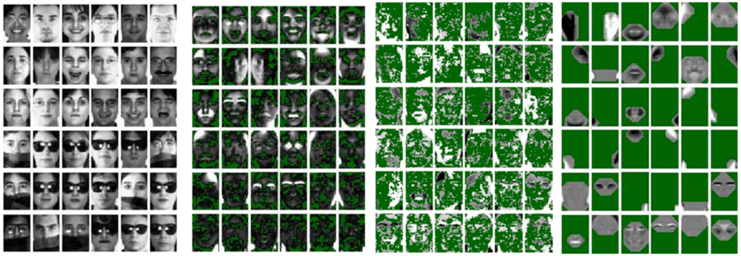
- Structured sparsity encourages zeroes to be any intersections of groups.
  - Possible non-zeroes are given by $\cap_{g \in \mathcal{G}'} g^c$ for all $\mathcal{G}' \subseteq \mathcal{G}$.
    - Equivalently, the set of zeroes is any $\cup_{g \in \mathcal{G}'} g$.
  - Our first example used $\{1\}$ and $\{1,2\}$ so possible non-zeroes $\{\}$, $\{2\}$, or $\{1,2\}$.
    - E.g., $\{2\}$ is $\{1,2\} \cap \{1\}^c = \{1,2\} \cap \{2\}$.
- Example is enforcing convex non-zero patterns:



https://arxiv.org/pdf/1109.2397v2.pdf

Structured Sparsity

- Structured sparsity encourages zeroes to be any intersections of groups.
  - Possible non-zeroes are given by $\cap_{g \in \mathcal{G}'} g^c$ for all $\mathcal{G}' \subseteq \mathcal{G}$.
    - Equivalently, the set of zeroes is any $\cup_{g \in \mathcal{G}'} g$.
  - Our first example used $\{1\}$ and $\{1, 2\}$ so possible non-zeroes $\{\}$, $\{2\}$, or $\{1, 2\}$.
    - E.g., $\{2\}$ is $\{1, 2\} \cap \{1\}^c = \{1, 2\} \cap \{2\}$.
- Example is enforcing convex non-zero patterns:



https://arxiv.org/pdf/1109.2397v2.pdf

- Left-to-right: data, NMF, sparse PCA, and PCA with structured sparsity.

Structured Sparsity

- Structured sparsity encourages zeroes to be any intersections of groups.
  - Possible non-zeroes are given by $\cap_{g \in \mathcal{G}'} g^c$ for all $\mathcal{G}' \subseteq \mathcal{G}$.
    - Equivalently, the set of zeroes is any $\cup_{g \in \mathcal{G}'} g$.
  - Our first example used $\{1\}$ and $\{1, 2\}$ so possible non-zeroes $\{\}$, $\{2\}$, or $\{1, 2\}$.
    - E.g., $\{2\}$ is $\{1, 2\} \cap \{1\}^c = \{1, 2\} \cap \{2\}$.
- Example is enforcing convex non-zero patterns:

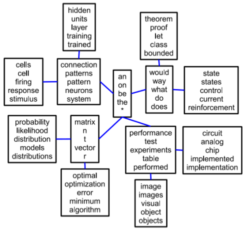

hidden units layer training trained
theorem proof let class bounded
cells cell firing response stimulus
connection patterns pattern neurons system
an on be the *
would way what do does
state states control current reinforcement
probability likelihood distribution models distributions
matrix n t vector r
performance test experiments table performed
circuit analog chip implemented implementation
optimal optimization error minimum algorithm
image images visual object objects

Figure 4. Example of a topic hierarchy estimated from 1714 NIPS proceedings papers (from 1988 through 1999). Each node corresponds to a topic whose 5 most important words are displayed. Single characters such as n, t, r are part of the vocabulary and often appear in NIPS papers, and their place in the hierarchy is semantically relevant to children topics.

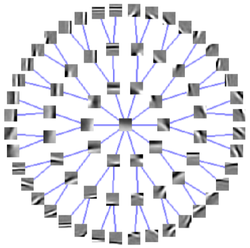


Figure 3. Learned dictionary with tree structure of depth 4. The root of the tree is in the middle of the figure. The branching factors are $p_1 = 10$, $p_2 = 2$, $p_3 = 2$. The dictionary is learned on $50,000$ patches of size 16×16 pixels.

www.di.ens.fr/~fbach/icml2010a.pdf

- There is also a variant ("over-LASSO") that considers unions of groups.

Summary

- Group L1-regularization encourages sparsity in variable groups.
- Structured regularization encourages more-general patterns in variables.
- Total-variation penalizes differences between variables.
- Structured sparsity can enforce sparsity hierarchies.

- Next time: finding all the cat videos on YouTube.

Debugging a Proximal-Gradient Code

- In general, debugging optimization codes can be difficult.
  - The code can appear to work even if it's wrong.
- A reasonable strategy is to test things you expect to be true.
  - And keep a set of tests that should remain true if you update the code.
- For example, for proximal-gradient methods you could check:
  - Does it decrease the objective function for a small enough step-size?
  - Are the step-sizes sensible (are they much smaller than $1/L$)?
  - Is the optimality condition going to zero as you run the algorithm?
- For group L1-regularization, some other checks that you can do:
  - Set $\lambda = 0$ and see if you get the unconstrained solution.
  - Assign each variable to its own group and see if you get the L1-regularized solution.
  - Assign all variables to the same group and see if you get an L2-regularization solution (and 0 for large-enough λ).