Structured Sparsity in Machine Learning:
Models, Algorithms, and Applications

André F. T. Martins

Joint work with:

Mário A. T. Figueiredo, Instituto de Telecomunicações, Lisboa, Portugal
Noah A. Smith, Language Technologies Institute, Carnegie Mellon University, USA

BCS/AG DANK 2013, November 2013

Outline

Our Setup

- Input set $\mathcal{X}$, output set $\mathcal{Y}$
- Linear model:

$$\widehat{y} := \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(x, y)$$

  where $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ is a feature map
- Learning the model parameters from data $\{(x_n, y_n)\}_{n=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$:

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \underbrace{\frac{1}{N} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)}_{\text{empirical risk}} + \underbrace{\Omega(\mathbf{w})}_{\text{regularizer}}$$

Our Setup

- Input set $\mathcal{X}$, output set $\mathcal{Y}$
- Linear model:

$$\widehat{y} := \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(x, y)$$

  where $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ is a feature map
- Learning the model parameters from data $\{(x_n, y_n)\}_{n=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$:

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \underbrace{\frac{1}{N}\sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)}_{\text{empirical risk}} + \underbrace{\Omega(\mathbf{w})}_{\text{regularizer}}$$

This talk: we focus on the **regularizer** Ω

The Bet On Sparsity (Friedman et al., 2004)

Sparsity hypothesis: not all dimensions of $\mathbf{f}$ are needed (many features are *irrelevant*)

Setting the corresponding weights to zero leads to a **sparse** $\mathbf{w}$

Models with just a few features:

- are easier to explain/interpret
- have a smaller memory footprint
- are faster to run (less features need to be evaluated)
- generalize better

(Automatic) Feature Selection

Domain experts are often good at engineering features.

Can we automate the process of selecting which ones to keep?

Three main classes of methods (Guyon and Elisseeff, 2003):

1. filters
2. wrappers
3. embedded methods

(Automatic) Feature Selection

Domain experts are often good at engineering features.

Can we automate the process of selecting which ones to keep?

Three main classes of methods (Guyon and Elisseeff, 2003):

1. filters (inexpensive and simple, but very suboptimal)
2. wrappers
3. embedded methods

(Automatic) Feature Selection

Domain experts are often good at engineering features.

Can we automate the process of selecting which ones to keep?

Three main classes of methods (Guyon and Elisseeff, 2003):

1. filters (inexpensive and simple, but very suboptimal)
2. wrappers (better, but very expensive)
3. embedded methods

(Automatic) Feature Selection

Domain experts are often good at engineering features.

Can we automate the process of selecting which ones to keep?

Three main classes of methods (Guyon and Elisseeff, 2003):

1. filters (inexpensive and simple, but very suboptimal)
2. wrappers (better, but very expensive)
3. embedded methods **(this talk)**

Embedded Methods for Feature Selection

Formulate the learning problem as a trade-off between

- minimizing loss (fitting the training data, achieving good accuracy on the training data, etc.)
- choosing a desirable model (e.g., with no more features than needed)

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$$

Embedded Methods for Feature Selection

Formulate the learning problem as a trade-off between

- minimizing loss (fitting the training data, achieving good accuracy on the training data, etc.)
- choosing a desirable model (e.g., with no more features than needed)

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$$

Design Ω to select relevant features (*sparsity-inducing regularization*)

Embedded Methods for Feature Selection

Formulate the learning problem as a trade-off between

- minimizing loss (fitting the training data, achieving good accuracy on the training data, etc.)
- choosing a desirable model (e.g., with no more features than needed)

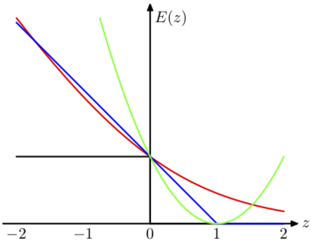
$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$$

Design Ω to select relevant features (*sparsity-inducing regularization*)

Key advantage: declarative statements of model “desirability” often lead to well-understood, convex optimization problems.

Convex Loss Functions

Squared (linear regression)	$\frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2$
Log-linear (MaxEnt, CRF, logistic)	$-\mathbf{w}^\top \mathbf{f}(x,y) + \log \sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \mathbf{f}(x,y'))$
Hinge (SVMs)	$-\mathbf{w}^\top \mathbf{f}(x,y) + \max_{y' \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{f}(x,y') + c(y,y')\right)$
Perceptron	$-\mathbf{w}^\top \mathbf{f}(x,y) + \max_{y' \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(x,y')$



Regularization Formulations

- Tikhonov regularization: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \lambda\Omega(\mathbf{w}) + \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)$
- Ivanov regularization

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ &\quad \text{subject to } \Omega(\mathbf{w}) \leq \tau \end{aligned}$$

- Morozov regularization

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) \\ &\quad \text{subject to } \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \leq \delta \end{aligned}$$

Regularization Formulations

- Tikhonov regularization: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \lambda\Omega(\mathbf{w}) + \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)$
- Ivanov regularization

$$\begin{aligned}\widehat{\mathbf{w}} \quad = \quad & \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ & \text{subject to } \Omega(\mathbf{w}) \leq \tau\end{aligned}$$

- Morozov regularization

$$\begin{aligned}\widehat{\mathbf{w}} \quad = \quad & \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) \\ & \text{subject to } \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \leq \delta\end{aligned}$$

Equivalent, under mild conditions (namely convexity).

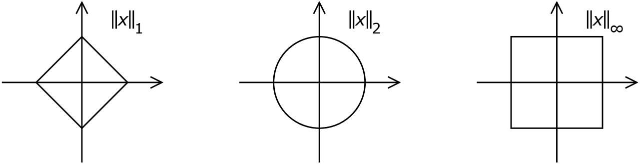
Norms: a Quick Review

- Any norm is a convex function (follows from triangle inequality)

Norms: a Quick Review

- Any norm is a convex function (follows from triangle inequality)

- **ℓ_p-norms ($p \geq 1$):** $\boxed{\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}}$

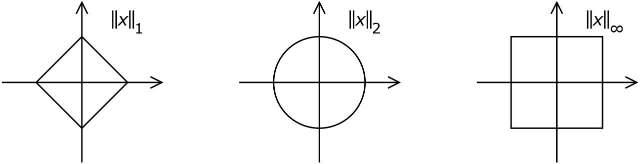


$$\|\mathbf{w}\|_1 = \sum_i |w_i|, \quad \|\mathbf{w}\|_2 = \sum_i w_i^2, \quad \|\mathbf{w}\|_\infty = \max_i |w_i|$$

Norms: a Quick Review

- Any norm is a convex function (follows from triangle inequality)

- **ℓ_p-norms ($p \geq 1$):** $\boxed{\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}}$



$$\|\mathbf{w}\|_1 = \sum_i |w_i|, \quad \|\mathbf{w}\|_2 = \sum_i w_i^2, \quad \|\mathbf{w}\|_\infty = \max_i |w_i|$$

- **Side note:** the infamous ℓ_0 "norm" (**non-convex, not a norm**):

$$\|\mathbf{w}\|_0 = \lim_{p \to 0} \|\mathbf{w}\|_p^p = |\{i : w_i \neq 0\}|$$

Ridge and Lasso Regularizers

Ridge or ℓ_2 regularization: $\boxed{\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2}$

- goes back to Tikhonov (1943) and Wiener (1949)
- corresponds to a zero-mean Gaussian prior
- **Pros**: smooth and convex, thus benign for optimization.
- **Cons**: doesn't promote sparsity (no explicit feature selection)

Ridge and Lasso Regularizers

Ridge or ℓ_2 regularization: $\boxed{\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2}$

- goes back to Tikhonov (1943) and Wiener (1949)
- corresponds to a zero-mean Gaussian prior
- **Pros**: smooth and convex, thus benign for optimization.
- **Cons**: doesn't promote sparsity (no explicit feature selection)

Lasso or ℓ_1 regularization: $\boxed{\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_1}$

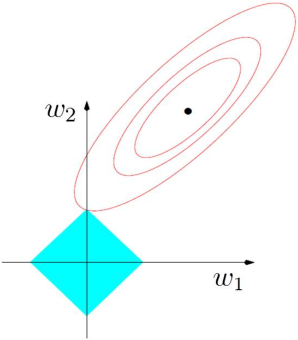
- goes back to Claerbout and Muir (1973); Taylor et al. (1979); Tibshirani (1996)
- corresponds to zero-mean Laplacian prior
- **Pros**: encourages sparsity: embedded feature selection.
- **Cons**: convex, but non-smooth: more challenging optimization.

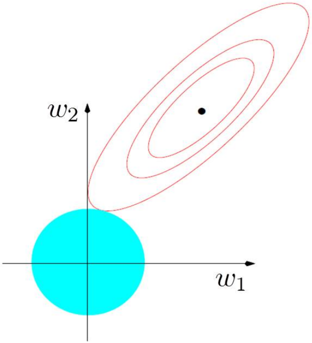
The Lasso and Sparsity

Why does the Lasso yield sparsity?

The Lasso and Sparsity

Why does the Lasso yield sparsity?





$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2 \quad \text{subject to } \|\mathbf{w}\|_1 \leq \tau$$

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2 \quad \text{subject to } \|\mathbf{w}\|_2^2 \leq \tau$$

Take-Home Messages

- Sparsity is desirable for interpretability, computational savings, and generalization
- ℓ_1-regularization gives an embedded method for feature selection
- Another view of ℓ_1: a convex surrogate for direct penalization of cardinality (ℓ_0)
- Under some conditions, ℓ_1 guarantees exact support recovery (Candès et al., 2006; Donoho, 2006)
- However: the currently known sufficient conditions are too strong and not met in typical ML problems

Outline

Models

ℓ_1 regularization promotes **sparse models**

A very simple sparsity pattern: **small cardinality**

Models

ℓ_1 regularization promotes **sparse models**

A very simple sparsity pattern: **small cardinality**

Main question: how to promote less trivial sparsity patterns?

Structured Sparsity and Groups

Main goal: promote **structural patterns**, not just penalize cardinality

Structured Sparsity and Groups

Main goal: promote **structural patterns**, not just penalize cardinality

Group sparsity: discard entire *groups* of features

- **density** inside each group
- **sparsity** with respect to the groups which are selected
- choice of groups: prior knowledge about the intended *sparsity patterns*

Structured Sparsity and Groups

Main goal: promote **structural patterns**, not just penalize cardinality

Group sparsity: discard entire *groups* of features

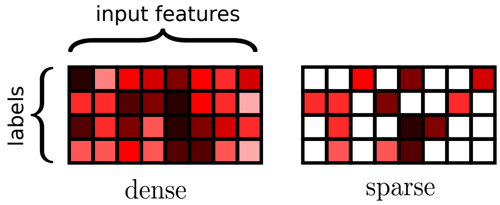
- **density** inside each group
- **sparsity** with respect to the groups which are selected
- choice of groups: prior knowledge about the intended *sparsity patterns*

Yields statistical gains if prior assumptions are correct (Stojnic et al., 2009)

Example: Sparsity in a Grid

Assume the feature map decomposes as $\mathbf{f}(x, y) = \mathbf{f}(x) \otimes \mathbf{e}_y$

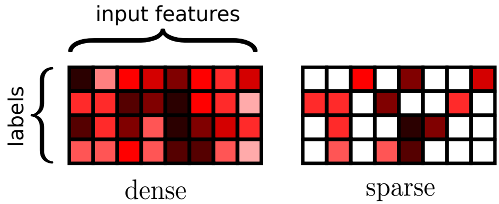
In words: we're conjoining each input feature with each output class



Example: Sparsity in a Grid

Assume the feature map decomposes as $\mathbf{f}(x, y) = \mathbf{f}(x) \otimes \mathbf{e}_y$

In words: we're conjoining each input feature with each output class



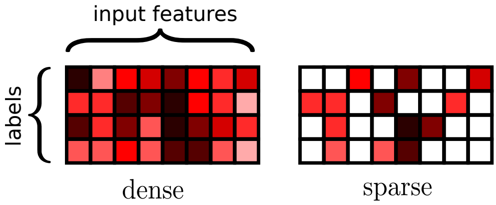
"Standard" sparsity is wasteful—we may still need all the input features

What we want: discard some input features

Example: Sparsity in a Grid

Assume the feature map decomposes as $\mathbf{f}(x, y) = \mathbf{f}(x) \otimes \mathbf{e}_y$

In words: we're conjoining each input feature with each output class



"Standard" sparsity is wasteful—we may still need all the input features

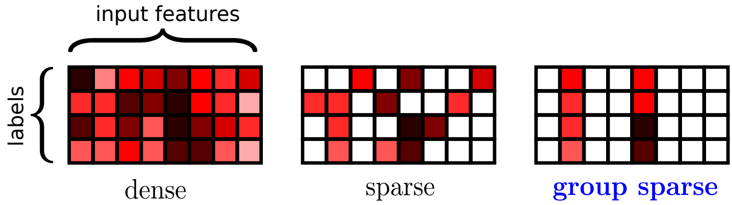
What we want: discard some input features

Solution: one group per *input* feature (conjoined with each of the labels)

Example: Sparsity in a Grid

Assume the feature map decomposes as $\mathbf{f}(x, y) = \mathbf{f}(x) \otimes \mathbf{e}_y$

In words: we're conjoining each input feature with each output class



"Standard" sparsity is wasteful—we may still need all the input features

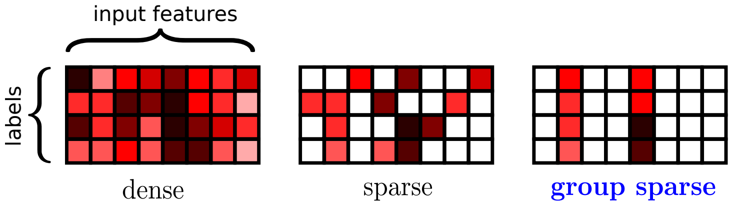
What we want: discard some input features

Solution: one group per *input* feature (conjoined with each of the labels)

Example: Sparsity in a Grid

Assume the feature map decomposes as $\mathbf{f}(x, y) = \mathbf{f}(x) \otimes \mathbf{e}_y$

In words: we're conjoining each input feature with each output class



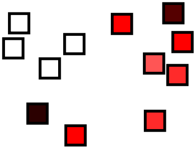
"Standard" sparsity is wasteful—we may still need all the input features

What we want: discard some input features

Solution: one group per *input* feature (conjoined with each of the labels)

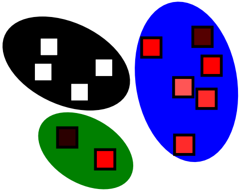
Similar structure: multi-task learning (Caruana, 1997; Obozinski et al., 2010), multiple kernel learning (Lanckriet et al., 2004)

Group Sparsity



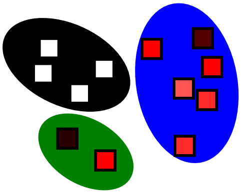
- D features

Group Sparsity



- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

Group Sparsity

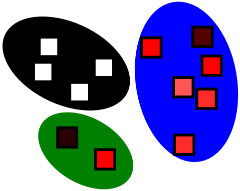


- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

Group-Lasso (Bakin, 1999; Yuan and Lin, 2006):

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \|\mathbf{w}_m\|_2}$$

Group Sparsity



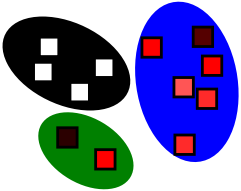
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

Group-Lasso (Bakin, 1999; Yuan and Lin, 2006):

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \|\mathbf{w}_m\|_2}$$

- Intuitively: the ℓ_1 **norm** of the ℓ_2 **norms**
- Technically, still a norm (called a *mixed* norm, denoted $\ell_{2,1}$)

Group Sparsity



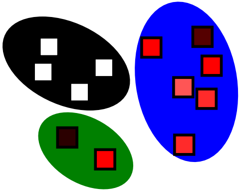
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

Group-Lasso (Bakin, 1999; Yuan and Lin, 2006):

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

- Intuitively: the ℓ_1 **norm** of the ℓ_2 **norms**
- Technically, still a norm (called a *mixed* norm, denoted $\ell_{2,1}$)
- λ_m: prior weight for group G_m (different groups have different sizes)

Group Sparsity



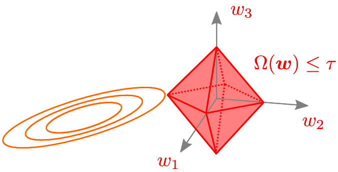
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

Group-Lasso (Bakin, 1999; Yuan and Lin, 2006):

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

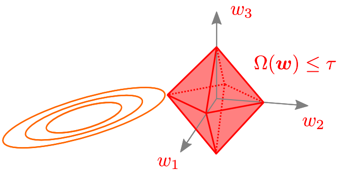
- Intuitively: the ℓ_1 **norm** of the ℓ_2 **norms**
- Technically, still a norm (called a *mixed* norm, denoted $\ell_{2,1}$)
- λ_m: prior weight for group G_m (different groups have different sizes)
- Statisticians call these **composite absolute penalties** (Zhao et al., 2009)

Lasso versus group-Lasso

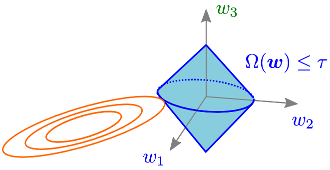


$$\boxed{\Omega(\boldsymbol{w}) = |w_1| + |w_2| + |w_3|}$$

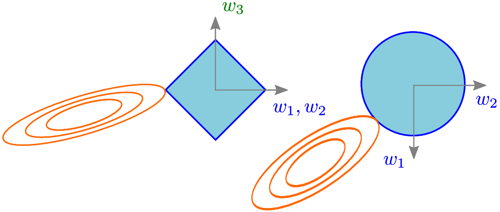
Lasso versus group-Lasso



$$\boxed{\Omega(\boldsymbol{w}) = |w_1| + |w_2| + |w_3|}$$



$$\boxed{\Omega(\boldsymbol{w}) = \sqrt{w_1^2 + w_2^2} + |w_3|}$$



Three Scenarios

- Non-overlapping groups
- Tree-structured groups
- Arbitrary groups

Three Scenarios

- Non-overlapping groups
- Tree-structured groups
- Arbitrary groups

Non-overlapping Groups

Assume $G_1, \ldots, G_M$ are **disjoint**

$\Rightarrow$ Each feature belongs to exactly one group

Non-overlapping Groups

Assume $G_1, \ldots, G_M$ are **disjoint**

$\Rightarrow$ Each feature belongs to exactly one group

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

Trivial choices of groups recover *unstructured* regularizers:

Non-overlapping Groups

Assume $G_1, \ldots, G_M$ are **disjoint**

$\Rightarrow$ Each feature belongs to exactly one group

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

Trivial choices of groups recover *unstructured* regularizers:

- ℓ_2-regularization: one large group $G_1 = \{1, \ldots, D\}$
- ℓ_1-regularization: D singleton groups $G_d = \{d\}$

Non-overlapping Groups

Assume $G_1, \ldots, G_M$ are **disjoint**

$\Rightarrow$ Each feature belongs to exactly one group

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

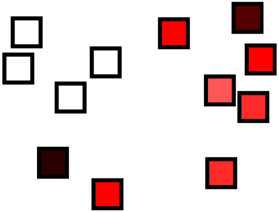
Trivial choices of groups recover *unstructured* regularizers:

- ℓ_2-regularization: one large group $G_1 = \{1, \ldots, D\}$
- ℓ_1-regularization: D singleton groups $G_d = \{d\}$

Examples of non-trivial groups:

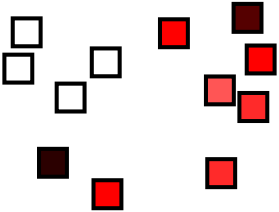
- label-based groups (groups are columns of a matrix)
- template-based groups (next)

Example: Feature Template Selection



Example: Feature Template Selection

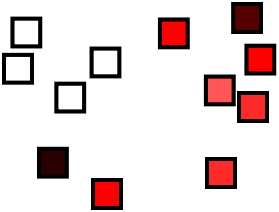
Input:	We	want	to	explore	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)



Example: Feature Template Selection

Input:	We	want	to	explore	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

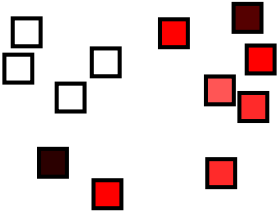
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

				▽			
Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

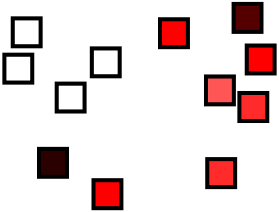
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

				▽			
Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

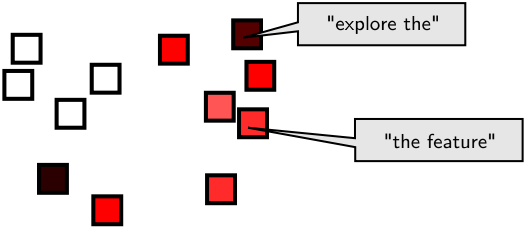
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

					▽		
Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

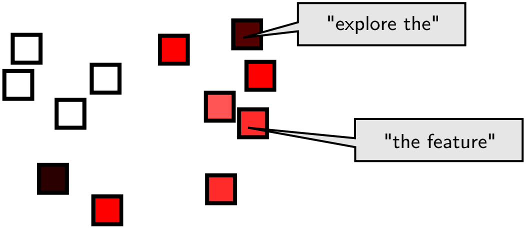
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

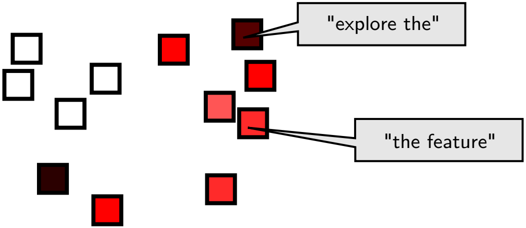
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

				▽			
Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

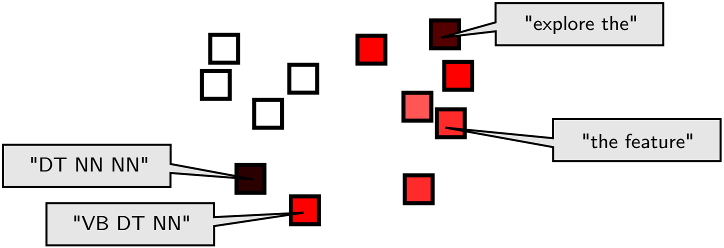
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

					▽		
Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

- Goal: Select relevant **feature templates**

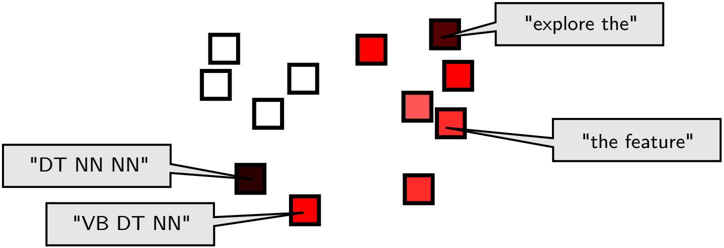


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

- Goal: Select relevant **feature templates**

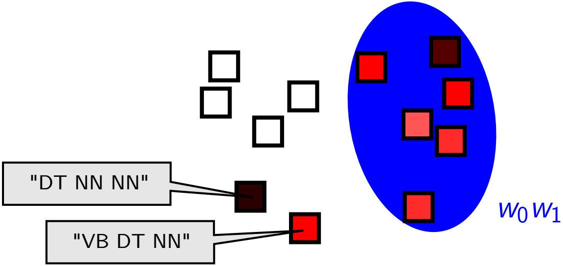
  ⇒ Make each group correspond to a feature template



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

- Goal: Select relevant **feature templates**
  $\Rightarrow$ Make each group correspond to a feature template

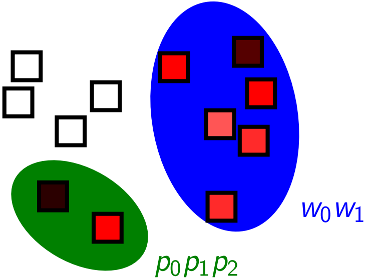


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

- Goal: Select relevant **feature templates**

  $\Rightarrow$ Make each group correspond to a feature template

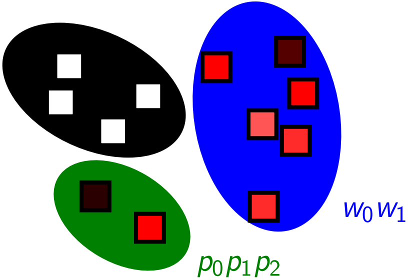


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

- Goal: Select relevant **feature templates**

  ⇒ Make each group correspond to a feature template

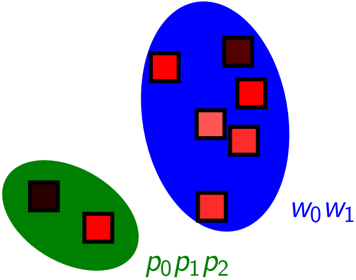


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	(NP)	(VP	VP	VP)	(NP	NP	NP)

- Goal: Select relevant **feature templates**

  $\Rightarrow$ Make each group correspond to a feature template



Three Scenarios

- Non-overlapping groups
- Tree-structured groups
- Arbitrary groups

Three Scenarios

- Non-overlapping groups
- Tree-structured groups
- Arbitrary groups

Tree-Structured Groups

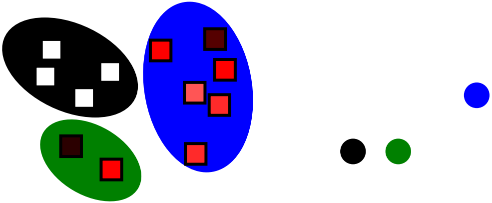
Assumption: if two groups overlap, one contains the other

$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)

Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

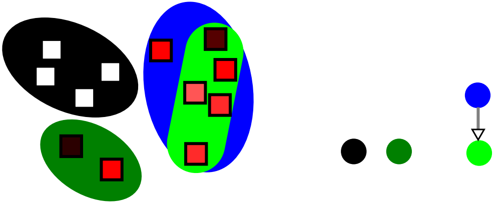
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

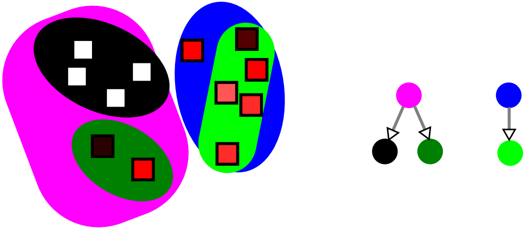
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

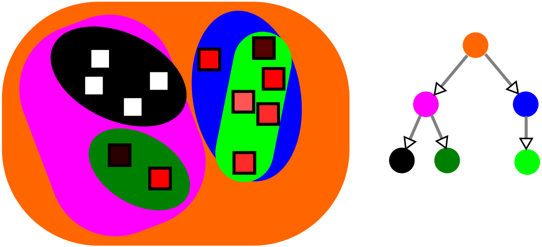
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

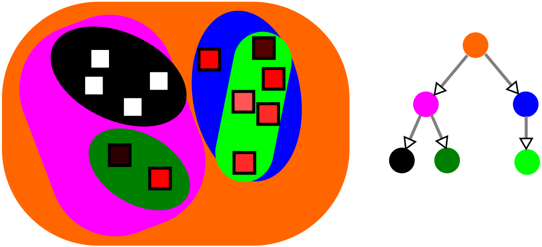
⇒ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

⇒ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)

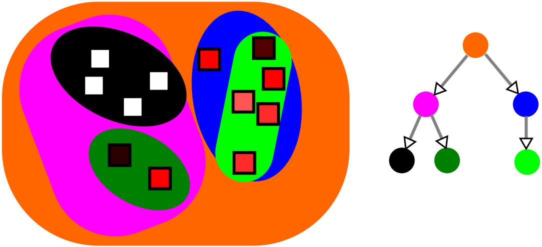


- What is the **sparsity pattern**?

Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



- What is the **sparsity pattern**?
- **If a group is discarded, all its descendants are also discarded**

Three Scenarios

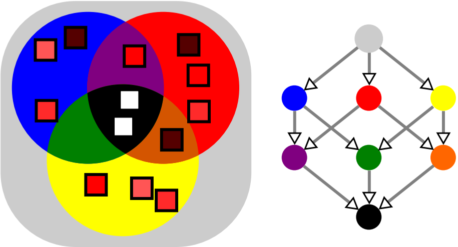
- Non-overlapping groups
- Tree-structured groups
- Arbitrary Groups

Three Scenarios

- Non-overlapping groups
- Tree-structured groups
- Arbitrary Groups

Arbitrary Groups

In general: groups can be represented as a **directed acyclic graph**

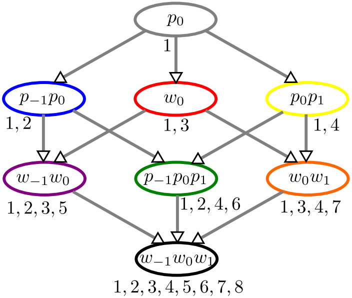


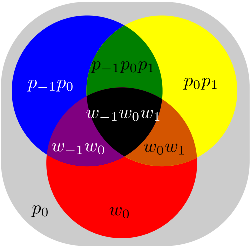
- set inclusion induces a **partial order** on groups (Jenatton et al., 2009)
- feature space becomes a **poset**
- **sparsity patterns**: given by this poset

Example: Coarse-to-Fine Regularization

1. Define a partial order between basic feature templates (e.g., $p_0 \preceq w_0$)
2. Extend this partial order to all templates by lexicographic closure: $p_0 \preceq p_0 p_1 \preceq w_0 w_1$

Goal: only include *finer* features if *coarser* ones are also in the model





Things to Keep in Mind

- **Structured sparsity** cares about the *structure* of the feature space
- **Group-Lasso regularization** generalizes ℓ_1 and it's still convex

Things to Keep in Mind

- **Structured sparsity** cares about the *structure* of the feature space
- **Group-Lasso regularization** generalizes ℓ_1 and it's still convex
- **Choice of groups:** problem dependent, opportunity to use prior knowledge to favour certain structural patterns

Things to Keep in Mind

- **Structured sparsity** cares about the *structure* of the feature space
- **Group-Lasso regularization** generalizes ℓ_1 and it's still convex
- **Choice of groups:** problem dependent, opportunity to use prior knowledge to favour certain structural patterns
- **Next:** algorithms
- We'll see that optimization is easier with non-overlapping or tree-structured groups than with arbitrary overlaps

Outline

Learning the Model

Recall that learning involves solving

$$\min_{\mathbf{w}} \quad \underbrace{\Omega(\mathbf{w})}_{\text{regularizer}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)}_{\text{total loss}},$$

Learning the Model

Recall that learning involves solving

$$\min_{\mathbf{w}} \quad \underbrace{\Omega(\mathbf{w})}_{\text{regularizer}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)}_{\text{total loss}},$$

Two kinds of optimization algorithms:

- batch algorithms (attacks the complete problem);
- online algorithms (use the training examples one by one)

Learning the Model

Recall that learning involves solving

$$\min_{\mathbf{w}} \quad \underbrace{\Omega(\mathbf{w})}_{\text{regularizer}} \; + \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)}_{\text{total loss}},$$

Two kinds of optimization algorithms:

- batch algorithms (attacks the complete problem);
- online algorithms (use the training examples one by one)

We'll focus on proximal gradient algorithms (both batch and online)

A Key Ingredient: Proximity Operator

The **Ω-proximity operator** is the following $\mathbb{R}^D \to \mathbb{R}^D$ map:

$$\boxed{\mathbf{w} \mapsto \mathrm{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|^2 + \Omega(\mathbf{u})}$$

(A generalization of Euclidean projection)

A Key Ingredient: Proximity Operator

The **Ω-proximity operator** is the following $\mathbb{R}^D \to \mathbb{R}^D$ map:

$$\boxed{\mathbf{w} \mapsto \mathrm{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|^2 + \Omega(\mathbf{u})}$$

(A generalization of Euclidean projection)

- ℓ_2 regularization $\boxed{\Omega(\mathbf{w}) \;=\; \frac{\lambda}{2}\|\mathbf{w}\|_2^2}$ $\Rightarrow$ **scaling operation**

A Key Ingredient: Proximity Operator

The **Ω-proximity operator** is the following $\mathbb{R}^D \to \mathbb{R}^D$ map:

$$\boxed{\mathbf{w} \mapsto \mathrm{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|^2 + \Omega(\mathbf{u})}$$

(A generalization of Euclidean projection)

- ℓ_2 regularization $\boxed{\Omega(\mathbf{w}) \;=\; \frac{\lambda}{2}\|\mathbf{w}\|_2^2}$ $\Rightarrow$ **scaling operation**

A Key Ingredient: Proximity Operator

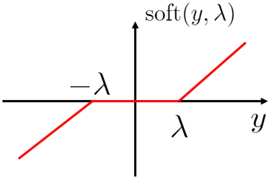
The **Ω-proximity operator** is the following $\mathbb{R}^D \to \mathbb{R}^D$ map:

$$\boxed{\mathbf{w} \mapsto \mathrm{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|^2 + \Omega(\mathbf{u})}$$

(A generalization of Euclidean projection)

- ℓ_2 regularization $\boxed{\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2}$ $\Rightarrow$ **scaling operation**
- ℓ_1 regularization $\boxed{\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_1}$ $\Rightarrow$ **soft-thresholding**:

$$[\mathrm{prox}_{\Omega}(\mathbf{w})]_d = \begin{cases} w_d - \lambda & \text{if } w_d > \lambda \\ 0 & \text{if } |w_d| \leq \lambda \\ w_d + \lambda & \text{if } w_d < -\lambda. \end{cases}$$



Proximity Operators for Structured Sparsity

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

- **Non-overlapping** $\Rightarrow$ **vector soft-thresholding**:

$$[\mathrm{prox}_{\Omega}(\mathbf{w})]_m = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{w}_m\|_2 \leq \lambda_m \\ \frac{\|\mathbf{w}_m\|_2 - \lambda_m}{\|\mathbf{w}_m\|_2}\mathbf{w}_m & \text{otherwise.} \end{cases}$$

Proximity Operators for Structured Sparsity

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

- **Non-overlapping** $\Rightarrow$ **vector soft-thresholding**:

$$[\text{prox}_\Omega(\mathbf{w})]_m = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{w}_m\|_2 \leq \lambda_m \\ \frac{\|\mathbf{w}_m\|_2 - \lambda_m}{\|\mathbf{w}_m\|_2}\mathbf{w}_m & \text{otherwise.} \end{cases}$$

- **Tree-structured**: can be computed recursively (Jenatton et al., 2010)

Proximity Operators for Structured Sparsity

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

- **Non-overlapping** $\Rightarrow$ **vector soft-thresholding**:

$$[\mathrm{prox}_\Omega(\mathbf{w})]_m = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{w}_m\|_2 \leq \lambda_m \\ \frac{\|\mathbf{w}_m\|_2 - \lambda_m}{\|\mathbf{w}_m\|_2} \mathbf{w}_m & \text{otherwise.} \end{cases}$$

- **Tree-structured**: can be computed recursively (Jenatton et al., 2010)
- **Arbitrary groups: no efficient procedure is known**

Proximity Operators for Structured Sparsity

$$\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_2}$$

- **Non-overlapping** $\Rightarrow$ **vector soft-thresholding**:

$$[\mathrm{prox}_\Omega(\mathbf{w})]_m = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{w}_m\|_2 \leq \lambda_m \\ \frac{\|\mathbf{w}_m\|_2 - \lambda_m}{\|\mathbf{w}_m\|_2} \mathbf{w}_m & \text{otherwise.} \end{cases}$$

- **Tree-structured**: can be computed recursively (Jenatton et al., 2010)
- **Arbitrary groups: no efficient procedure is known**

  The problem can be sidestepped with sequential proximity steps (Martins et al., 2011a) (more later).

Outline

Iterative Shrinkage-Thresholding (IST)

$$\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}, \quad \text{where } \Lambda(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)$$

Iterative Shrinkage-Thresholding (IST)

$$\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}, \quad \text{where } \Lambda(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)$$

Building blocks:

- loss gradient/subgradient $\nabla\Lambda$, proximity operator prox_{Ω}

Iterative Shrinkage-Thresholding (IST)

$$\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}, \quad \text{where } \Lambda(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)$$

Building blocks:

- loss gradient/subgradient $\nabla\Lambda$, proximity operator prox_{Ω}

$$\boxed{\mathbf{w}_{t+1} \leftarrow \mathrm{prox}_{\eta_t\Omega}\left(\mathbf{w}_t - \eta_t \nabla\Lambda(\mathbf{w}_t)\right)}$$

Iterative Shrinkage-Thresholding (IST)

$$\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}, \quad \text{where } \Lambda(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)$$

Building blocks:

- loss gradient/subgradient $\nabla\Lambda$, proximity operator prox_Ω

$$\boxed{\mathbf{w}_{t+1} \leftarrow \text{prox}_{\eta_t\Omega}\left(\mathbf{w}_t - \eta_t \nabla\Lambda(\mathbf{w}_t)\right)}$$

Can be derived with different tools:

- expectation-maximization (EM) (Figueiredo and Nowak, 2003);
- majorization-minimization (Daubechies et al., 2004);
- forward-backward splitting (Combettes and Wajs, 2006);
- separable approximation (Wright et al., 2009).

Iterative Shrinkage-Thresholding (IST)

$$\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}, \quad \text{where } \Lambda(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i)$$

Building blocks:

- loss gradient/subgradient $\nabla\Lambda$, proximity operator prox_{Ω}

$$\boxed{\mathbf{w}_{t+1} \leftarrow \mathrm{prox}_{\eta_t \Omega}\left(\mathbf{w}_t - \eta_t \nabla\Lambda(\mathbf{w}_t)\right)}$$

Can be derived with different tools:

- expectation-maximization (EM) (Figueiredo and Nowak, 2003);
- majorization-minimization (Daubechies et al., 2004);
- forward-backward splitting (Combettes and Wajs, 2006);
- separable approximation (Wright et al., 2009).

Convergence: requires $O(1/\epsilon)$ iterations for ϵ-accurate objective.

Other Proximal-Gradient Variants

SpaRSA (Wright et al., 2009): the same IST update scheme, but setting η_t to mimic a Newton step (Barzilai and Borwein, 1988):

$$\eta_t^{-1}\mathbf{I} \approx \mathbf{H}(\mathbf{w}_t) \quad \text{(Hessian)}$$

- Works very well in pratice!

Other Proximal-Gradient Variants

SpaRSA (Wright et al., 2009): the same IST update scheme, but setting η_t to mimic a Newton step (Barzilai and Borwein, 1988):

$$\eta_t^{-1}\mathbf{I} \approx \mathbf{H}(\mathbf{w}_t) \quad \text{(Hessian)}$$

- Works very well in pratice!

FISTA (Beck and Teboulle, 2009): compute $\mathbf{w}_{t+1}$ based, not only on $\mathbf{w}_t$, but also on $\mathbf{w}_{t-1}$ (Nesterov, 1983):

$$\boxed{\begin{array}{lcl} b_{t+1} & = & \frac{1+\sqrt{1+4\,b_t^2}}{2} \\ \mathbf{z} & = & \mathbf{w}_t + \frac{b_t-1}{b_{t+1}}(\mathbf{w}_t - \mathbf{w}_{t-1}) \\ \mathbf{w}_{t+1} & = & \mathrm{prox}_{\eta\Omega}(\mathbf{z} - \eta\nabla\Lambda(\mathbf{z})) \end{array}}$$

Other Proximal-Gradient Variants

SpaRSA (Wright et al., 2009): the same IST update scheme, but setting η_t to mimic a Newton step (Barzilai and Borwein, 1988):

$$\eta_t^{-1}\mathbf{I} \approx \mathbf{H}(\mathbf{w}_t) \quad \text{(Hessian)}$$

- Works very well in pratice!

FISTA (Beck and Teboulle, 2009): compute $\mathbf{w}_{t+1}$ based, not only on $\mathbf{w}_t$, but also on $\mathbf{w}_{t-1}$ (Nesterov, 1983):

$$\boxed{\begin{array}{lcl} b_{t+1} & = & \frac{1+\sqrt{1+4\,b_t^2}}{2} \\ \mathbf{z} & = & \mathbf{w}_t + \frac{b_t-1}{b_{t+1}}(\mathbf{w}_t - \mathbf{w}_{t-1}) \\ \mathbf{w}_{t+1} & = & \mathrm{prox}_{\eta\Omega}(\mathbf{z} - \eta\nabla\Lambda(\mathbf{z})) \end{array}}$$

- **Iteration bound:** $O(1/\sqrt{\epsilon})$ as opposed to $O(1/\epsilon)$.

Many Other Batch Algorithms

- coordinate descent (Shevade and Keerthi, 2003; Genkin et al., 2007; Krishnapuram et al., 2005; Tseng and Yun, 2009)
- Least Angle Regression (LARS) and homotopy/continuation methods (Efron et al., 2004; Osborne et al., 2000; Figueiredo et al., 2007)
- shooting method (Fu, 1998)
- grafting (Perkins et al., 2003) and grafting-light (Zhu et al., 2010)
- orthant-wise limited-memory quasi-Newton (OWL-QN) (Andrew and Gao, 2007; Gao et al., 2007)
- alternating direction method of multipliers (ADMM) (Afonso et al., 2010; Figueiredo and Bioucas-Dias, 2011).

...several more; this is an active research area!

Outline

Why Online?

1. Suitable for large datasets

Why Online?

1. Suitable for large datasets
2. Suitable for structured prediction

Why Online?

1. Suitable for large datasets
2. Suitable for structured prediction
3. Faster to approach a near-optimal region

Why Online?

1. Suitable for large datasets
2. Suitable for structured prediction
3. Faster to approach a near-optimal region
4. Slower convergence, but this is fine in machine learning ("the tradeoffs of large scale learning" by Bottou and Bousquet (2007))

Plain Stochastic (Sub-)Gradient Descent

$$\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i),$$

initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  (sub-)gradient step: $\mathbf{w} \ \leftarrow \ \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla}L(\mathbf{w}; x_t, y_t)\right)$
end for

Plain Stochastic (Sub-)Gradient Descent

$$\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i),$$

initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  (sub-)gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla} L(\mathbf{w}; x_t, y_t)\right)$
end for

- ℓ_1-regularization $\boxed{\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \mathrm{sign}(\mathbf{w})$

$$\mathbf{w} \leftarrow \underbrace{\mathbf{w} - \eta_t \lambda \mathrm{sign}(\mathbf{w})}_{\text{constant penalty}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

Plain Stochastic (Sub-)Gradient Descent

$$\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i),$$

> initialize $\mathbf{w} = \mathbf{0}$
> **for** $t = 1, 2, \ldots$ **do**
>   take training pair (x_t, y_t)
>   (sub-)gradient step: $\mathbf{w} \ \leftarrow \ \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla} L(\mathbf{w}; x_t, y_t)\right)$
> **end for**

- ℓ_1-regularization $\boxed{\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \mathrm{sign}(\mathbf{w})$

$$\mathbf{w} \ \leftarrow \ \underbrace{\mathbf{w} - \eta_t \lambda \mathrm{sign}(\mathbf{w})}_{\text{constant penalty}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

Plain Stochastic (Sub-)Gradient Descent

$$\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, y_i),$$

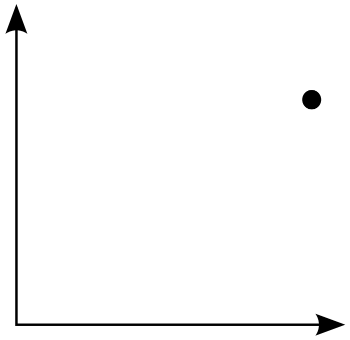
```
initialize w = 0
for t = 1, 2, ... do
    take training pair (x_t, y_t)
    (sub-)gradient step: w ← w − η_t (∇̃Ω(w) + ∇̃L(w; x_t, y_t))
end for
```

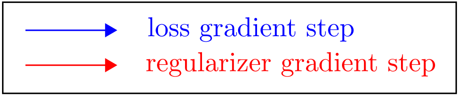
- ℓ_1-regularization $\boxed{\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_1} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \mathrm{sign}(\mathbf{w})$

$$\mathbf{w} \leftarrow \underbrace{\mathbf{w} - \eta_t \lambda \mathrm{sign}(\mathbf{w})}_{\text{constant penalty}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

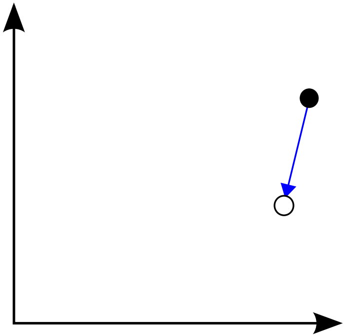
- **Problem: iterates are never sparse!**

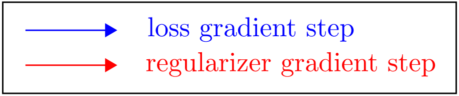
Plain SGD with ℓ_1-regularization



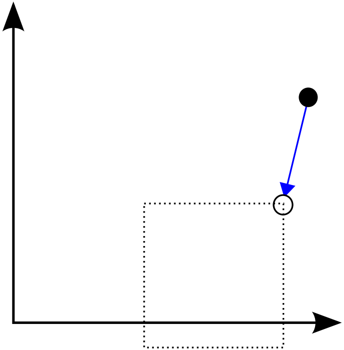


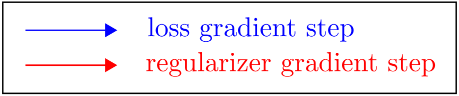
Plain SGD with ℓ_1-regularization



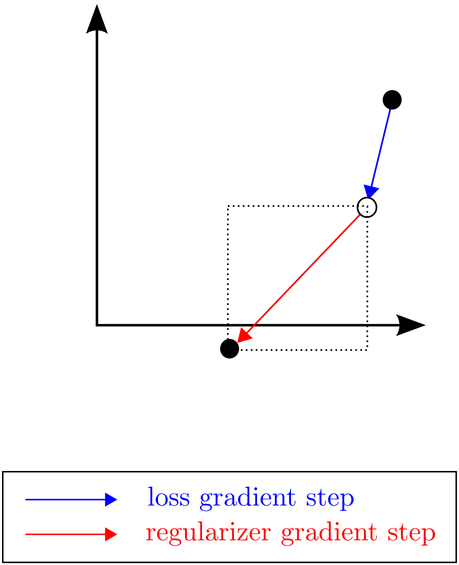


Plain SGD with ℓ_1-regularization

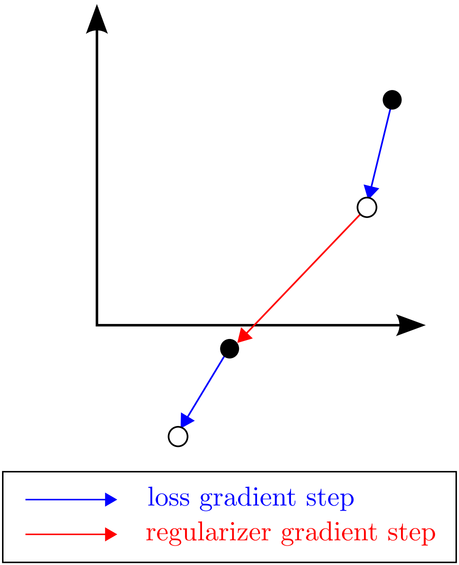




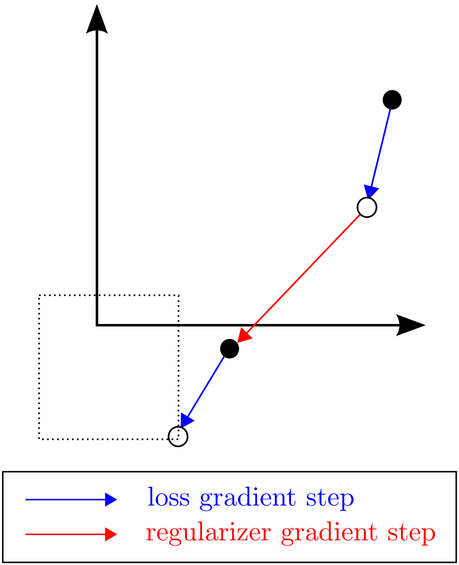
Plain SGD with ℓ_1-regularization



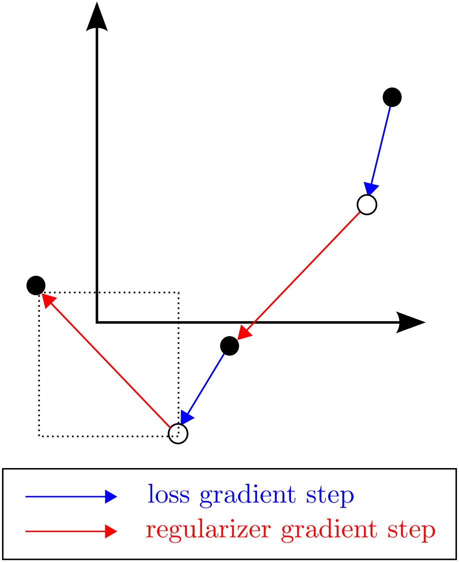
Plain SGD with ℓ_1-regularization



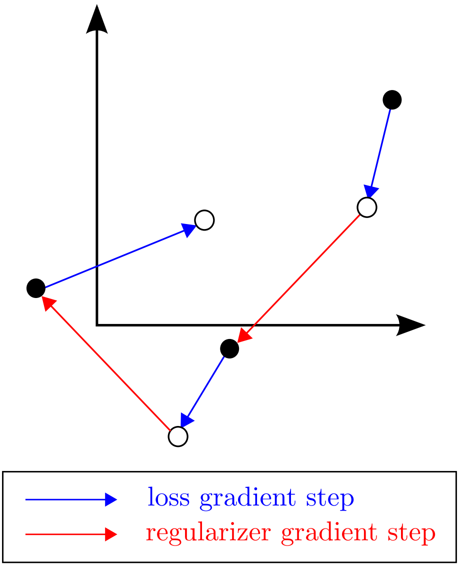
Plain SGD with ℓ_1-regularization



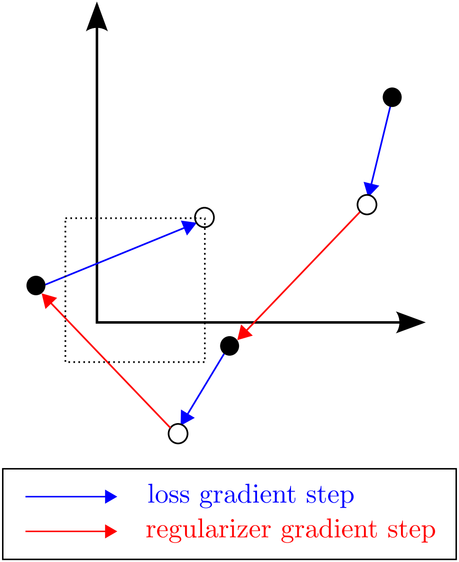
Plain SGD with ℓ_1-regularization



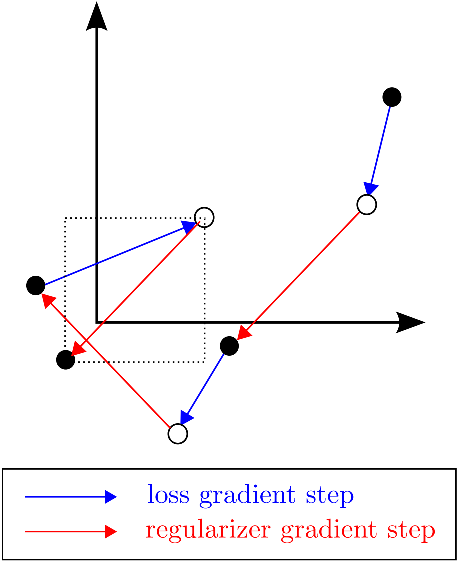
Plain SGD with ℓ_1-regularization



Plain SGD with ℓ_1-regularization



Plain SGD with ℓ_1-regularization

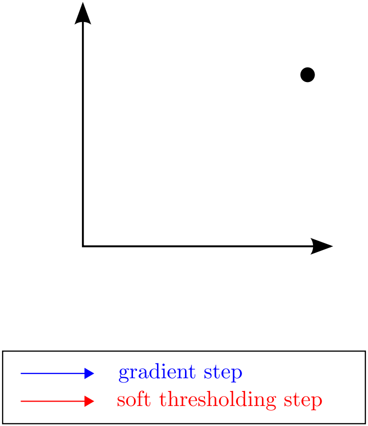


“Sparse” Online Algorithms

- Truncated Gradient (Langford et al., 2009)
- Online Forward-Backward Splitting (Duchi and Singer, 2009)
- Regularized Dual Averaging (Xiao, 2010)
- Online Proximal Gradient (Martins et al., 2011a)

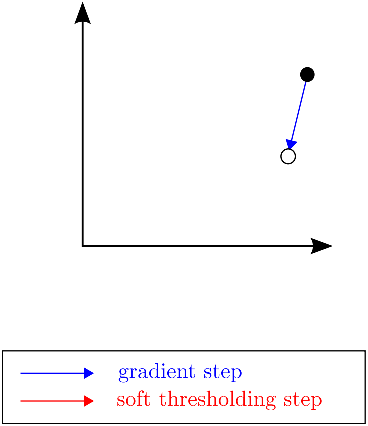
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



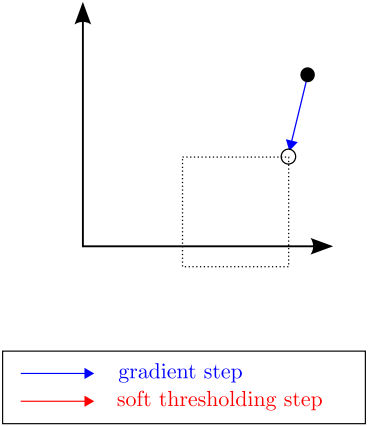
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



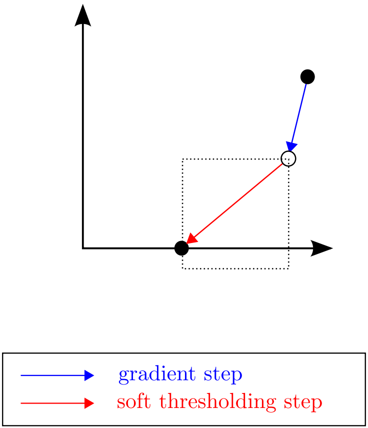
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



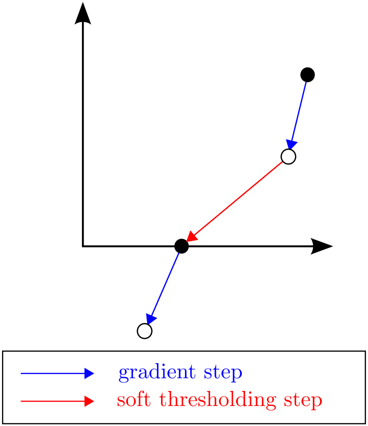
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



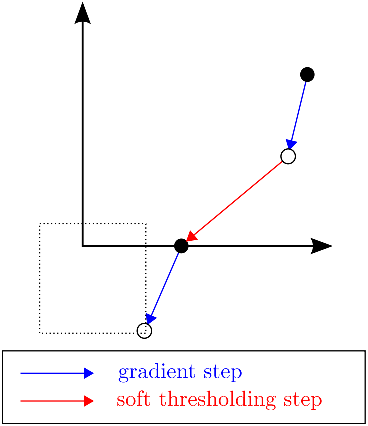
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



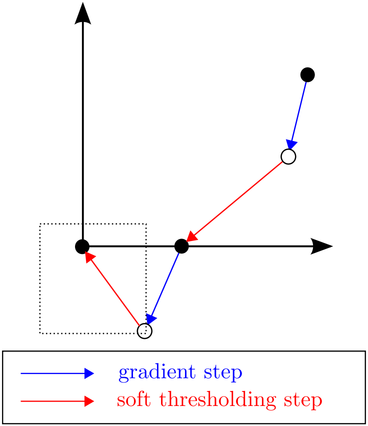
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



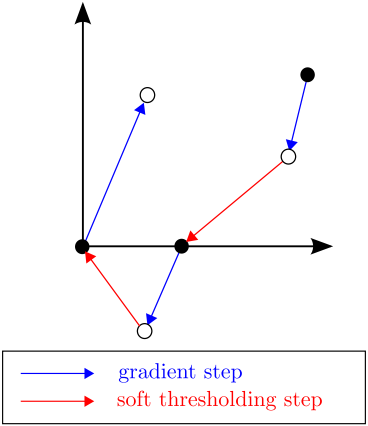
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



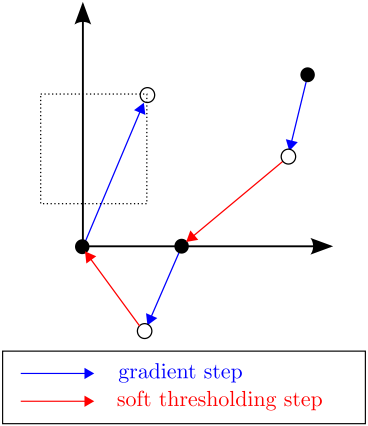
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



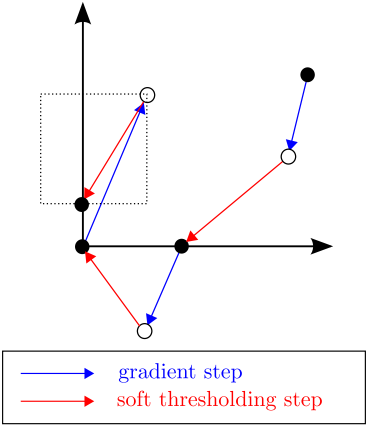
Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



Truncated Gradient (Langford et al., 2009)

- take gradients-step **only with respect to the loss**
- apply soft-thresholding
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**



"Sparse" Online Algorithms

- Truncated Gradient (Langford et al., 2009)
- Online Forward-Backward Splitting (Duchi and Singer, 2009)
- Regularized Dual Averaging (Xiao, 2010)
- Online Proximal Gradient (Martins et al., 2011a)

Online Forward-Backward Splitting (Duchi and Singer, 2009)

initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla L(\mathbf{w}; x_t, y_t)$
  proximal step: $\mathbf{w} \leftarrow \mathrm{prox}_{\eta_t \Omega}(\mathbf{w})$
end for

- generalizes truncated gradient to arbitrary regularizers Ω
  - can tackle non-overlapping or hierarchical group-Lasso, but arbitrary overlaps are difficult to handle (more later)
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**

“Sparse” Online Algorithms

- Truncated Gradient (Langford et al., 2009)
- Online Forward-Backward Splitting (Duchi and Singer, 2009)
- Regularized Dual Averaging (Xiao, 2010)
- Online Proximal Gradient (Martins et al., 2011a)

Prox-Grad with Overlaps (Martins et al., 2011a)

Key idea: decompose $\Omega(\mathbf{w}) = \sum_{j=1}^{J} \Omega_j(\mathbf{w})$, where each Ω_j is non-overlapping, and apply **sequential proximal steps**:

$$\textbf{gradient step}: \ \mathbf{w} \ \leftarrow \ \mathbf{w} - \eta_t \nabla L(\theta; x_t, y_t)$$

$$\textbf{proximal steps}: \ \mathbf{w} \ \leftarrow \ \mathrm{prox}_{\eta_t \Omega_J}\left(\mathrm{prox}_{\eta_t \Omega_{J-1}}\left(\ldots \mathrm{prox}_{\eta_t \Omega_1}(\mathbf{w})\right)\right)$$

Prox-Grad with Overlaps (Martins et al., 2011a)

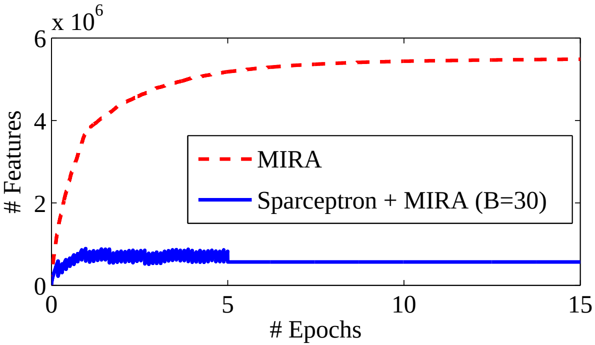
Key idea: decompose $\Omega(\mathbf{w}) = \sum_{j=1}^{J} \Omega_j(\mathbf{w})$, where each Ω_j is non-overlapping, and apply **sequential proximal steps**:

> **gradient step**: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla L(\boldsymbol{\theta}; x_t, y_t)$
>
> **proximal steps**: $\mathbf{w} \leftarrow \mathrm{prox}_{\eta_t \Omega_J}\left(\mathrm{prox}_{\eta_t \Omega_{J-1}}\left(\ldots \mathrm{prox}_{\eta_t \Omega_1}(\mathbf{w})\right)\right)$

- still convergent, same $O(1/\epsilon^2)$ iteration bound
- **gradient step:** linear in # of features that fire, *independent* of D.
- **proximal steps:** linear in # of groups M.
- other implementation tricks (debiasing, budget-driven shrinkage, etc.)

Memory Footprint

- 5 epochs for identifying relevant groups, 10 epochs for debiasing



Summary of Algorithms

	Converges	Rate	Sparse	Groups	Overlaps
Coord. desc.	✓	?	✓	Maybe	No
Prox-grad	✓	$O(1/\epsilon)$	Yes/No	✓	Not easy
OWL-QN	✓	?	Yes/No	No	No
SpaRSA	✓	$O(1/\epsilon)$ or better	Yes/No	✓	Not easy
FISTA	✓	$O(1/\sqrt{\epsilon})$	Yes/No	✓	Not easy
ADMM	✓	$O(1/\epsilon)$	No	✓	✓
Online subgrad.	✓	$O(1/\epsilon^2)$	No	✓	No
Truncated grad.	✓	$O(1/\epsilon^2)$	✓	No	No
FOBOS	✓	$O(1/\epsilon^2)$	Sort of	✓	Not easy
Online prox-grad	✓	$O(1/\epsilon^2)$	✓	✓	✓

Outline

Applications of Structured Sparsity in ML

We will focus on two recent NLP applications (Martins et al., 2011b):

- Named entity recognition
- Dependency parsing

We use **feature templates** as groups.

Named Entity Recognition

Only	France	and	Britain	backed	Fischler	's	proposal	.
RB	NNP	CC	NNP	VBD	NNP	POS	NN	.

Named Entity Recognition

Only	France	and	Britain	backed	Fischler	's	proposal	.
RB	NNP	CC	NNP	VBD	NNP	POS	NN	.
	LOCATION		LOCATION		PERSON			

Named Entity Recognition

Only	France	and	Britain	backed	Fischler	's	proposal	.
RB	NNP	CC	NNP	VBD	NNP	POS	NN	.
	LOCATION		LOCATION		PERSON			

- Spanish, Dutch, and English CoNLL datasets
- 452 feature templates using POS tags, words, shapes, affixes, with various context sizes

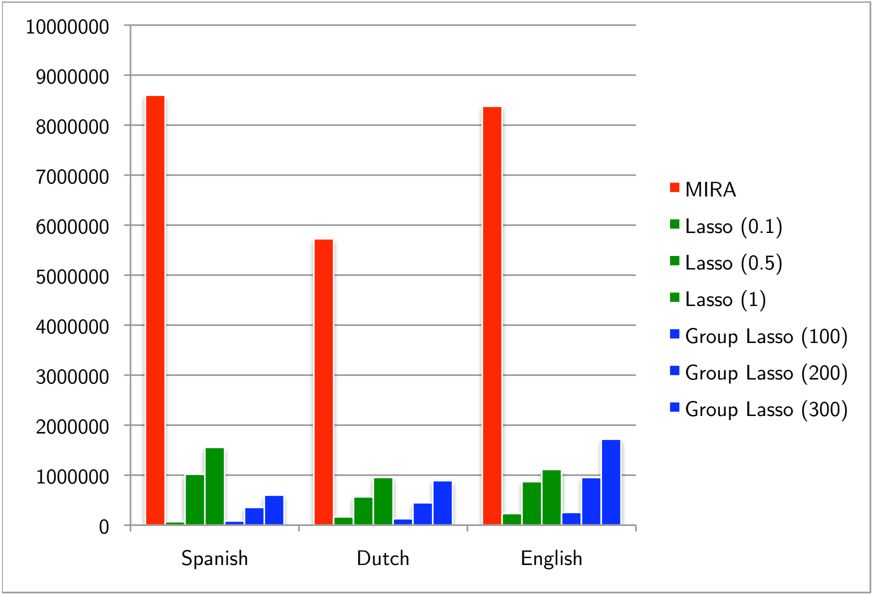
Named Entity Recognition

Only	France	and	Britain	backed	Fischler	's	proposal	.
RB	NNP	CC	NNP	VBD	NNP	POS	NN	.
	LOCATION		LOCATION		PERSON			

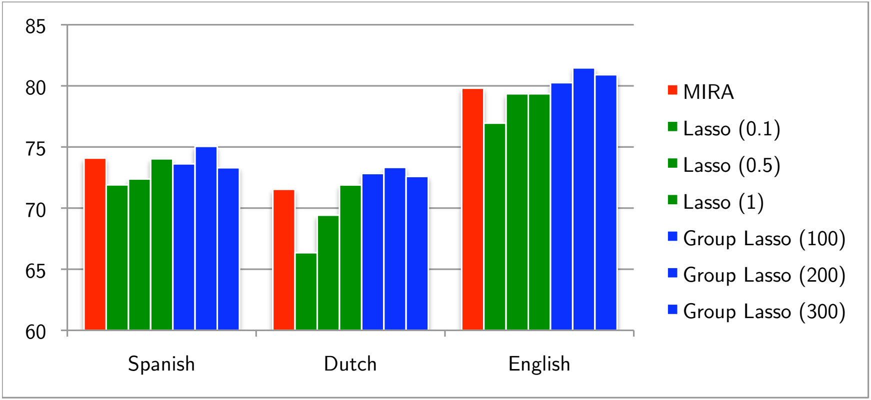
- Spanish, Dutch, and English CoNLL datasets
- 452 feature templates using POS tags, words, shapes, affixes, with various context sizes

Comparison between:

- ℓ_2-regularization (MIRA), best λ on dev-set, all features
- ℓ_1-regularization (Lasso), varying λ
- $\ell_{2,1}$-regularization (**Group Lasso**), varying the template budget

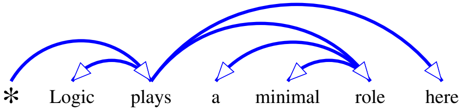


Named entity models: number of features. (Lasso $C = 1/\lambda N$.)

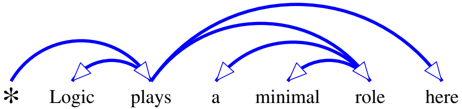


Named entity models: F_1 accuracy on the test set. (Lasso $C = 1/\lambda N$.)

Dependency Parsing

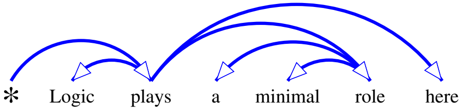


Dependency Parsing



- Arabic, Danish, Dutch, Japanese, Slovene, Spanish CoNLL datasets
- 684 feature templates (using words, lemmas, POS, contextual POS, arc length and direction)

Dependency Parsing

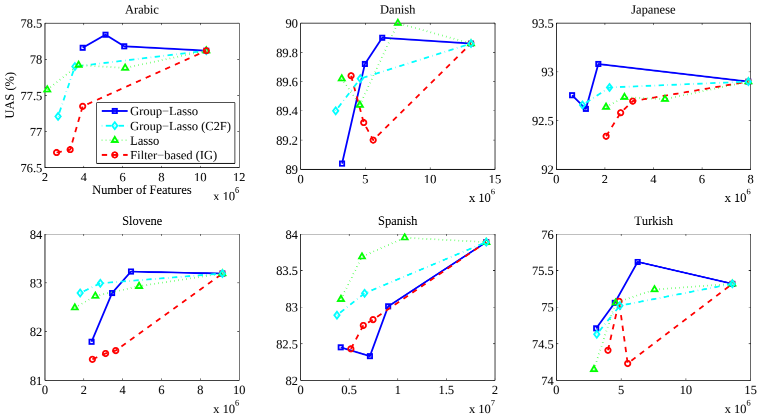


- Arabic, Danish, Dutch, Japanese, Slovene, Spanish CoNLL datasets
- 684 feature templates (using words, lemmas, POS, contextual POS, arc length and direction)

Comparison between:

- ℓ_2-regularization (MIRA), all features
- filter-based template selection (information gain)
- ℓ_1-regularization (Lasso)
- $\ell_{2,1}$-regularization (**Group Lasso**, coarse-to-fine regularization)

Dependency Parsing (c'ed)



Template-based group lasso is better at selecting feature templates than the IG criterion, and slightly better than **coarse-to-fine**.

Outline

Summary

- Sparsity is desirable in machine learning: *feature selection*, *runtime*, *memory footprint*, *interpretability*
- Beyond plain sparsity: **structured sparsity** can be promoted through group-Lasso regularization
- Choice of groups reflects prior knowledge about the desired sparsity patterns.
- Small/medium scale: many batch algorithms available, with fast convergence (IST, FISTA, SpaRSA, ...)
- Large scale: online proximal-gradient algorithms suitable to explore large feature spaces

Thank you!

- Questions?

Acknowledgments

- National Science Foundation (USA), CAREER grant IIS-1054319
- Fundação para a Ciência e Tecnologia (Portugal), grant PEst-OE/EEI/LA0008/2011.
- Fundação para a Ciência e Tecnologia and Information and Communication Technologies Institute (Portugal/USA), through the CMU-Portugal Program.
- Priberam: QREN/POR Lisboa (Portugal), EU/FEDER programme, Discooperio project, contract 2011/18501.

References I

Afonso, M., Bioucas-Dias, J., and Figueiredo, M. (2010). Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19:2345–2356.

Andrew, G. and Gao, J. (2007). Scalable training of L1-regularized log-linear models. In *Proc. of ICML*. ACM.

Bakin, S. (1999). *Adaptive regression and model selection in data mining problems*. PhD thesis, Australian National University.

Barzilai, J. and Borwein, J. (1988). Two point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Bolstad, A., Veen, B. V., and Nowak, R. (2009). Space-time event sparse penalization for magnetoelectroencephalography. *NeuroImage*, 46:1066–1081.

Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. *NIPS*, 20.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

Candès, E., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52:489–509.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Claerbout, J. and Muir, F. (1973). Robust modelling of erratic data. *Geophysics*, 38:826–844.

Combettes, P. and Wajs, V. (2006). Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4:1168–1200.

Daubechies, I., Defrise, M., and De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 11:1413–1457.

Donoho, D. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the L1-ball for learning in high dimensions. In *ICML*.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2873–2908.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32:407–499.

References II

Eisenstein, J., Smith, N. A., and Xing, E. P. (2011). Discovering sociolinguistic associations with structured sparsity. In *Proc. of ACL*.

Figueiredo, M. and Bioucas-Dias, J. (2011). An alternating direction algorithm for (overlapping) group regularization. In *Signal processing with adaptive sparse structured representations–SPARS11*. Edinburgh, UK.

Figueiredo, M. and Nowak, R. (2003). An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12:986–916.

Figueiredo, M., Nowak, R., and Wright, S. (2007). Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing: Special Issue on Convex Optimization Methods for Signal Processing*, 1:586–598.

Friedman, J., Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004). Discussion of three boosting papers. *Annals of Statistics*, 32(1):102–107.

Fu, W. (1998). Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, pages 397–416.

Gao, J., Andrew, G., Johnson, M., and Toutanova, K. (2007). A comparative study of parameter estimation methods for statistical natural language processing. In *Proc. of ACL*.

Genkin, A., Lewis, D., and Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49:291–304.

Graça, J., Ganchev, K., Taskar, B., and Pereira, F. (2009). Posterior vs. parameter sparsity in latent variable models. *Advances in Neural Information Processing Systems*.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Hastie, T., Taylor, J., Tibshirani, R., and Walther, G. (2007). Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29.

Jenatton, R., Audibert, J.-Y., and Bach, F. (2009). Structured variable selection with sparsity-inducing norms. Technical report, arXiv:0904.3523.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2010). Proximal methods for sparse hierarchical dictionary learning. In *Proc. of ICML*.

References III

Kim, S. and Xing, E. (2010). Tree-guided group lasso for multi-task regression with structured sparsity. In *Proc. of ICML*.

Krishnapuram, B., Carin, L., Figueiredo, M., and Hartemink, A. (2005). Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27:957–968.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72.

Langford, J., Li, L., and Zhang, T. (2009). Sparse online learning via truncated gradient. *JMLR*, 10:777–801.

Mairal, J., Jenatton, R., Obozinski, G., and Bach, F. (2010). Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems*.

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2011a). Online learning of structured predictors with multiple kernels. In *Proc. of AISTATS*.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011b). Structured Sparsity in Structured Prediction. In *Proc. of Empirical Methods for Natural Language Processing*.

McDonald, R. T., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Doklady*, 27:372–376.

Obozinski, G., Taskar, B., and Jordan, M. (2010). Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252.

Osborne, M., Presnell, B., and Turlach, B. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–403.

Perkins, S., Lacker, K., and Theiler, J. (2003). Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356.

Quattoni, A., Carreras, X., Collins, M., and Darrell, T. (2009). An efficient projection for $l_{1,\infty}$ regularization. In *Proc. of ICML*.

Sang, E. (2002). Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

Sang, E. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

References IV

Shevade, S. and Keerthi, S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19:2246–2253.

Stojnic, M., Parvaresh, F., and Hassibi, B. (2009). On the reconstruction of block-sparse signals with an optimal number of measurements. *Signal Processing, IEEE Transactions on*, 57(8):3075–3085.

Taylor, H., Bank, S., and McCoy, J. (1979). Deconvolution with the ℓ_1 norm. *Geophysics*, 44:39–52.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B.*, pages 267–288.

Tikhonov, A. (1943). On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198.

Tseng, P. and Yun, S. (2009). A coordinate gradient descent method nonsmooth seperable approximation. *Mathematical Programmin (series B)*, 117:387–423.

Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. Wiley, New York.

Wright, S., Nowak, R., and Figueiredo, M. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57:2479–2493.

Xiao, L. (2010). Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society (B)*, 68(1):49.

Zhao, P., Rocha, G., and Yu, B. (2009). Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497.

Zhu, J., Lao, N., and Xing, E. (2010). Grafting-light: fast, incremental feature selection and structure learning of markov random fields. In *Proc. of International Conference on Knowledge Discovery and Data Mining*, pages 303–312.