Structured Sparsity in Natural Language Processing:

Models, Algorithms, and Applications

André F. T. Martins[1,3] Dani Yogatama[2] Noah A. Smith[2]
Mário A. T. Figueiredo[1]

[1]Instituto de Telecomunicações
Instituto Superior Técnico, Lisboa, Portugal

[2]Language Technologies Institute, School of Computer Science
Carnegie Mellon University, Pittsburgh, PA, USA

[3]Priberam, Lisboa, Portugal

EACL 2014 Tutorial, Gothenburg, Sweden, April 27, 2014
Slides online at http://tiny.cc/ssnlp14

Welcome

This tutorial is about **sparsity**, a topic of great relevance to NLP.

- Sparsity relates to *feature selection*, *model compactness*, *runtime*, *memory footprint*, *interpretability* of our models.

New idea in the last 7 years: **structured sparsity**. This tutorial tries to answer:

- What is structured sparsity?
- How do we apply it?
- How has it been used so far?

Outline

Notation

Many NLP problems involve mapping from one structured space to another. Notation:

- Input set $\mathcal{X}$
- For each $x \in \mathcal{X}$, candidate outputs are $\mathcal{Y}(x) \subseteq \mathcal{Y}$
- Mapping is $h_{\mathbf{w}} : \mathcal{X} \rightarrow \mathcal{Y}$

Linear Models

Our predictor will take the form

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

where:

- $\mathbf{f}$ is a vector function that encodes all the relevant things about (x, y); the result of a theory, our knowledge, feature engineering, etc.
- $\mathbf{w} \in \mathbb{R}^D$ are the weights that parameterize the mapping.

NLP today: D is often in the tens or hundreds of millions.

Learning Linear Models

Max ent, perceptron, CRF, SVM, even supervised generative models all fit the linear modeling framework.

General training setup:

- We observe a collection of examples $\{\langle x_n, y_n \rangle\}_{n=1}^N$.
- Perform statistical analysis to discover $\mathbf{w}$ from the data.
  Ranges from "count and normalize" to complex optimization routines.

Optimization view:

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

This tutorial will focus on the regularizer, Ω.

What is Sparsity?

The word "sparsity" has (at least) four related meanings in NLP!

1. Data sparsity: N is too small to obtain a good estimate for $\mathbf{w}$.
   Also known as "curse of dimensionality."
   (Usually bad.)
2. "Probability" sparsity: I have a probability distribution over events (e.g., $\mathcal{X} \times \mathcal{Y}$), most of which receive *zero* probability.
   (Might be good or bad.)
3. Sparsity in the dual: associated with SVMs and other kernel-based methods; implies that the predictor can be represented via kernel calculations involving just a few training instances.
4. Model sparsity: Most dimensions of $\mathbf{f}$ are not needed for a good $h_{\mathbf{w}}$; those dimensions of $\mathbf{w}$ can be zero, leading to a sparse $\mathbf{w}$ (model).

This tutorial is about sense **#4**: today, (model) sparsity is a good thing!

Why Sparsity is Desirable in NLP

Occam's razor and interpretability.

The **bet on sparsity** (Friedman et al., 2004): it's often correct. When it isn't, there's no good solution anyway!

Models with just a few features are

- easy to explain and implement
- attractive as linguistic hypotheses
- reminiscent of classical symbolic systems

Final decision list for *plant* (abbreviated)		
LogL	Collocation	Sense
10.12	*plant* growth	$\Rightarrow$ A
9.68	car (within $\pm k$ words)	$\Rightarrow$ B
9.64	*plant* height	$\Rightarrow$ A
9.61	union (within $\pm k$ words)	$\Rightarrow$ B
9.54	equipment (within $\pm k$ words)	$\Rightarrow$ B
9.51	assembly *plant*	$\Rightarrow$ B
9.50	nuclear *plant*	$\Rightarrow$ B
9.31	flower (within $\pm k$ words)	$\Rightarrow$ A
9.24	job (within $\pm k$ words)	$\Rightarrow$ B
9.03	fruit (within $\pm k$ words)	$\Rightarrow$ A
9.02	*plant* species	$\Rightarrow$ A
...	...	

A decision list from Yarowsky (1995).

Why Sparsity is Desirable in NLP

Computational savings.

- $w_d = 0$ is equivalent to erasing the feature from the model; smaller effective D implies smaller memory footprint.
- This, in turn, implies faster decoding runtime.
- Further, sometimes entire *kinds* of features can be eliminated, giving asymptotic savings.

Why Sparsity is Desirable in NLP

Generalization.

- The challenge of learning is to extract from the data only what will generalize to new examples.
- Forcing a learner to use few features is one way to discourage overfitting.
- Text categorization experiments in Kazama and Tsujii (2003): $+3$ accuracy points with 1% as many features

(Automatic) Feature Selection

Human NLPers are good at thinking of features.

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

Three kinds of methods:

1. filters
2. wrappers
3. embedded methods (this tutorial)

(Automatic) Feature Selection

Human NLPers are good at thinking of features.

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

Three kinds of methods:

1. filters
2. wrappers
3. embedded methods (this tutorial)

Filter-based Feature Selection

For each candidate feature f_d, apply a heuristic to determine whether to include it. (Excluding f_d equates to fixing $w_d = 0$.)

Examples:

- Count threshold: is $|\{n \mid f_d(x_n, y_n) > 0\}| > \tau$?
  (Ignore rare features.)
- Mutual information or correlation between features and labels

Advantage: speed!

Disadvantages:

- Ignores the learning algorithm
- Thresholds require tuning

Ratnaparkhi (1996), on his POS tagger:

> *The behavior of a feature that occurs very sparsely in the training set is often difficult to predict, since its statistics may not be reliable. Therefore, the model uses the heuristic that any feature which occurs less than 10 times in the data is unreliable, and ignores features whose counts are less than 10.*[1] *While there are many smoothing algorithms which use techniques more rigorous than a simple count cutoff, they have not yet been investigated in conjunction with this tagger.*

[1] Except for features that look only at the current word, i.e., features of the form $w_i =$ `<word>` and $t_i =$ `<TAG>`. The count of 10 was chosen by inspection of Training and Development data.

(Automatic) Feature Selection

Human NLPers are good at thinking of features.

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

Three kinds of methods:

1. filters
2. wrappers
3. embedded methods (this tutorial)

(Automatic) Feature Selection

Human NLPers are good at thinking of features.

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

Three kinds of methods:

1. filters
2. wrappers
3. embedded methods (this tutorial)

Wrapper-based Feature Selection

For each subset $\mathcal{F} \subseteq \{1, 2, \ldots D\}$, learn $h_{\mathbf{w}_{\mathcal{F}}}$ for features $\{f_d \mid d \in \mathcal{F}\}$.

$2^D - 1$ choices; so perform a *search* over subsets.

Cons:

- NP-hard problem (Amaldi and Kann, 1998; Davis et al., 1997)
- Must resort to greedy methods
- Even those require iterative calls to a black-box learner
- Danger of overfitting in choosing $\mathcal{F}$.
  (Typically use development data or cross-validate.)

Della Pietra et al. (1997) add features one at a time. Step (3) involves re-estimating parameters:

Field Induction Algorithm

Initial Data:

A reference distribution $\tilde{p}$ and an initial model q_0.

Output:

A field q_* with active features $f_0, \ldots, f_N$ such that

$$q_* = \arg\min_{q \in \overline{\mathcal{Q}}(f, q_0)} D(\tilde{p} \,\|\, q).$$

Algorithm:

(0) Set $q^{(0)} = q_0$.

(1) For each candidate $g \in \mathcal{C}(q^{(n)})$ compute the gain $G_{q^{(n)}}(g)$.

(2) Let $f_n = \arg\max_{g \in \mathcal{C}(q^{(n)})} G_{q^{(n)}}(g)$ be the feature with the largest gain.

(3) Compute $q_* = \arg\min_{q \in \overline{\mathcal{Q}}(f, q_0)} D(\tilde{p} \,\|\, q)$, where $f = (f_0, f_1, \ldots, f_n)$.

(4) Set $q^{(n+1)} = q_*$ and $n \leftarrow n + 1$, and go to step (1).

(Automatic) Feature Selection

Human NLPers are good at thinking of features.

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

Three kinds of methods:

1. filters
2. wrappers
3. embedded methods (this tutorial)

(Automatic) Feature Selection

Human NLPers are good at thinking of features.

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

Three kinds of methods:

1. filters
2. wrappers
3. embedded methods (this tutorial)

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., one with no more features than needed)

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

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

Useful Papers on Feature Selection and Sparsity

- Overview of many feature selection methods:
  Guyon and Elisseeff (2003)
- Greedy wrapper-based method used for max ent models in NLP:
  Della Pietra et al. (1997)
- Early uses of sparsity in NLP:
  Kazama and Tsujii (2003); Goodman (2004)

Outline

Learning Problem

Recall that we formulate the learning problem as:

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

Loss functions (I)

- Regression ($y \in \mathbb{R}$) typically uses the **squared error** loss:

$$L_{\mathrm{SE}}(\mathbf{w}; x, y) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2$$

Loss functions (I)

- Regression ($y \in \mathbb{R}$) typically uses the **squared error** loss:

$$L_{\text{SE}}(\mathbf{w}; x, y) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2$$

- Total loss:

$$\frac{1}{2}\sum_{n=1}^{N}\left(y_n - \mathbf{w}^\top \mathbf{f}(x_n)\right)^2 = \frac{1}{2}\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2$$

Loss functions (I)

- Regression ($y \in \mathbb{R}$) typically uses the **squared error** loss:

$$L_{\mathrm{SE}}(\mathbf{w}; x, y) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2$$

- Total loss:

$$\frac{1}{2}\sum_{n=1}^{N}\left(y_n - \mathbf{w}^\top \mathbf{f}(x_n)\right)^2 = \frac{1}{2}\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2$$

- Design matrix: $\mathbf{A} = [A_{ij}]_{i=1,\ldots,N;\, j=1,\ldots,D}$, where $A_{ij} = f_j(x_i)$.

Loss functions (I)

- Regression ($y \in \mathbb{R}$) typically uses the **squared error** loss:

$$L_{\mathrm{SE}}(\mathbf{w}; x, y) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2$$

- Total loss:

$$\frac{1}{2}\sum_{n=1}^{N}\left(y_n - \mathbf{w}^\top \mathbf{f}(x_n)\right)^2 = \frac{1}{2}\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2$$

- Design matrix: $\mathbf{A} = [A_{ij}]_{i=1,\dots,N;\, j=1,\dots,D}$, where $A_{ij} = f_j(x_i)$.
- Response vector: $\mathbf{y} = [y_1, \dots, y_N]^\top$.

Loss functions (I)

- Regression ($y \in \mathbb{R}$) typically uses the **squared error** loss:

$$L_{\text{SE}}(\mathbf{w}; x, y) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2$$

- Total loss:

$$\frac{1}{2}\sum_{n=1}^{N}\left(y_n - \mathbf{w}^\top \mathbf{f}(x_n)\right)^2 = \frac{1}{2}\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2$$

- Design matrix: $\mathbf{A} = [A_{ij}]_{i=1,\ldots,N;\, j=1,\ldots,D}$, where $A_{ij} = f_j(x_i)$.
- Response vector: $\mathbf{y} = [y_1, \ldots, y_N]^\top$.
- Arguably, the most/best studied loss function (statistics, machine learning, signal processing).

Loss functions (II)

- Classification and structured prediction using **log-linear models** (logistic regression, max ent, conditional random fields):

$$
\begin{aligned}
L_{\mathrm{LR}}(\mathbf{w}; x, y) &= -\log P(y|x;\mathbf{w}) \\
&= -\log \frac{\exp(\mathbf{w}^\top \mathbf{f}(x,y))}{\sum_{y' \in \mathcal{Y}(x)} \exp(\mathbf{w}^\top \mathbf{f}(x,y'))} \\
&= -\mathbf{w}^\top \mathbf{f}(x,y) + \log Z(\mathbf{w}, x)
\end{aligned}
$$

Loss functions (II)

- Classification and structured prediction using **log-linear models** (logistic regression, max ent, conditional random fields):

$$
\begin{aligned}
L_{\mathrm{LR}}(\mathbf{w}; x, y) &= -\log P(y|x; \mathbf{w}) \\
&= -\log \frac{\exp(\mathbf{w}^\top \mathbf{f}(x, y))}{\sum_{y' \in \mathcal{Y}(x)} \exp(\mathbf{w}^\top \mathbf{f}(x, y'))} \\
&= -\mathbf{w}^\top \mathbf{f}(x, y) + \log Z(\mathbf{w}, x)
\end{aligned}
$$

- Partition function:

$$
Z(\mathbf{w}, x) = \sum_{y' \in \mathcal{Y}(x)} \exp(\mathbf{w}^\top \mathbf{f}(x, y')).
$$

Loss functions (II)

- Classification and structured prediction using **log-linear models** (logistic regression, max ent, conditional random fields):

$$\begin{aligned} L_{\mathrm{LR}}(\mathbf{w}; x, y) &= -\log P(y|x; \mathbf{w}) \\ &= -\log \frac{\exp(\mathbf{w}^\top \mathbf{f}(x, y))}{\sum_{y' \in \mathcal{Y}(x)} \exp(\mathbf{w}^\top \mathbf{f}(x, y'))} \\ &= -\mathbf{w}^\top \mathbf{f}(x, y) + \log Z(\mathbf{w}, x) \end{aligned}$$

- Partition function:

$$Z(\mathbf{w}, x) = \sum_{y' \in \mathcal{Y}(x)} \exp(\mathbf{w}^\top \mathbf{f}(x, y')).$$

- Related loss functions: **hinge loss** (in SVM) and the **perceptron loss**.

Main Loss Functions: Summary

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	$-\mathbf{w}^\top \mathbf{f}(x,y) + \max_{y' \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(x,y')$

(in the SVM loss, $c(y, y')$ is a cost function.)

Main Loss Functions: Summary

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	$-\mathbf{w}^\top \mathbf{f}(x, y) + \max_{y' \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(x, y')$

(in the SVM loss, $c(y, y')$ is a cost function.)

The log-linear, hinge, and perceptron losses are particular cases of general family (Martins et al., 2010).

Regularization Formulations

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

Regularization Formulations

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

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

Regularization Formulations

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

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

Equivalent, under mild conditions (namely convexity).

Regularization

Why regularize?

Regularization

Why regularize?

- Improve generalization by avoiding over-fitting.

Regularization

Why regularize?

- Improve generalization by avoiding over-fitting.
- Express prior knowledge about $\mathbf{w}$.

Regularization

Why regularize?

- Improve generalization by avoiding over-fitting.
- Express prior knowledge about $\mathbf{w}$.
- Select relevant features (via sparsity-inducing regularization).

Regularization

Why regularize?

- Improve generalization by avoiding over-fitting.
- Express prior knowledge about $\mathbf{w}$.
- Select relevant features (via sparsity-inducing regularization).

Regularization vs. Bayesian estimation

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) + \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)$

...interpretable as Bayesian **maximum a posteriori** (MAP) estimate:

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \underbrace{\exp\left(-\Omega(\mathbf{w})\right)}_{\text{prior } p(\mathbf{w})} \underbrace{\prod_{n=1}^{N} \exp\left(-L(\mathbf{w}; x_n, y_n)\right)}_{\text{likelihood (i.i.d. data)}}.$$

Regularization vs. Bayesian estimation

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) + \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)$

...interpretable as Bayesian **maximum a posteriori** (MAP) estimate:

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \underbrace{\exp\left(-\Omega(\mathbf{w})\right)}_{\text{prior } p(\mathbf{w})} \underbrace{\prod_{n=1}^{N} exp\left(-L(\mathbf{w}; x_n, y_n)\right)}_{\text{likelihood (i.i.d. data)}}.$$

- This interpretation underlies the logistic regression (LR) loss: $L_{\mathrm{LR}}(\mathbf{w}; x_n, y_n) = -\log\ P\left(y_n | x_n; \mathbf{w}\right)$.

Regularization vs. Bayesian estimation

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) + \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)$

...interpretable as Bayesian **maximum a posteriori** (MAP) estimate:

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \underbrace{\exp\left(-\Omega(\mathbf{w})\right)}_{\text{prior } p(\mathbf{w})} \underbrace{\prod_{n=1}^{N} exp\left(-L(\mathbf{w}; x_n, y_n)\right)}_{\text{likelihood (i.i.d. data)}}.$$

- This interpretation underlies the logistic regression (LR) loss: $L_{\text{LR}}(\mathbf{w}; x_n, y_n) = -\log\ P\left(y_n | x_n; \mathbf{w}\right)$.
- Same is true for the squared error (SE) loss: $L_{\text{SE}}(\mathbf{w}; x_n, y_n) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2 = -\log \mathcal{N}(y|\mathbf{w}^\top \mathbf{f}(x), 1)$

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- **Ridge regression** (SE loss): Hoerl and Kennard (1962 and 1970).

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- **Ridge regression** (SE loss): Hoerl and Kennard (1962 and 1970).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in NLP: Chen and Rosenfeld (1999).

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- **Ridge regression** (SE loss): Hoerl and Kennard (1962 and 1970).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in NLP: Chen and Rosenfeld (1999).
- Closely related to Tikhonov (1943) and Wiener (1949).

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- **Ridge regression** (SE loss): Hoerl and Kennard (1962 and 1970).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in NLP: Chen and Rosenfeld (1999).
- Closely related to Tikhonov (1943) and Wiener (1949).
- **Pros**: smooth and convex, thus benign for optimization.

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- **Ridge regression** (SE loss): Hoerl and Kennard (1962 and 1970).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in NLP: Chen and Rosenfeld (1999).
- Closely related to Tikhonov (1943) and Wiener (1949).
- **Pros**: smooth and convex, thus benign for optimization.
- **Cons**: doesn't promote sparsity (no explicit feature selection).

Classical Regularizers: Ridge

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

Arguably, the most classical choice: squared ℓ_2 norm: $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$

- Corresponds to zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- **Ridge regression** (SE loss): Hoerl and Kennard (1962 and 1970).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in NLP: Chen and Rosenfeld (1999).
- Closely related to Tikhonov (1943) and Wiener (1949).
- **Pros**: smooth and convex, thus benign for optimization.
- **Cons**: doesn't promote sparsity (no explicit feature selection).
- **Cons**: only encodes trivial prior knowledge.

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp(-\lambda |w_i|)$

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp(-\lambda |w_i|)$
- Best known as: **least absolute shrinkage and selection operator (Lasso)** (Tibshirani, 1996).

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp(-\lambda |w_i|)$
- Best known as: **least absolute shrinkage and selection operator (Lasso)** (Tibshirani, 1996).
- Used earlier in signal processing (Claerbout and Muir, 1973; Taylor et al., 1979) neural networks (Williams, 1995),...

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp\left(-\lambda |w_i|\right)$
- Best known as: **least absolute shrinkage and selection operator (Lasso)** (Tibshirani, 1996).
- Used earlier in signal processing (Claerbout and Muir, 1973; Taylor et al., 1979) neural networks (Williams, 1995),...
- In NLP: Kazama and Tsujii (2003); Goodman (2004).

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp(-\lambda |w_i|)$
- Best known as: **least absolute shrinkage and selection operator (Lasso)** (Tibshirani, 1996).
- Used earlier in signal processing (Claerbout and Muir, 1973; Taylor et al., 1979) neural networks (Williams, 1995),...
- In NLP: Kazama and Tsujii (2003); Goodman (2004).
- **Pros**: encourages sparsity: embedded feature selection.

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp(-\lambda |w_i|)$
- Best known as: **least absolute shrinkage and selection operator (Lasso)** (Tibshirani, 1996).
- Used earlier in signal processing (Claerbout and Muir, 1973; Taylor et al., 1979) neural networks (Williams, 1995),...
- In NLP: Kazama and Tsujii (2003); Goodman (2004).
- **Pros**: encourages sparsity: embedded feature selection.
- **Cons**: convex, but non-smooth: challenging optimization.

Classical Regularizers: Lasso

Regularized parameter estimate: $\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) + \Omega(\mathbf{w})$

The new classic is the ℓ_1 norm: $\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^{D} |w_i|$.

- Corresponds to zero-mean Laplacian prior $p(w_i) \propto \exp\left(-\lambda |w_i|\right)$
- Best known as: **least absolute shrinkage and selection operator (Lasso)** (Tibshirani, 1996).
- Used earlier in signal processing (Claerbout and Muir, 1973; Taylor et al., 1979) neural networks (Williams, 1995),...
- In NLP: Kazama and Tsujii (2003); Goodman (2004).
- **Pros**: encourages sparsity: embedded feature selection.
- **Cons**: convex, but non-smooth: challenging optimization.
- **Cons**: only encodes trivial prior knowledge.

The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest case:

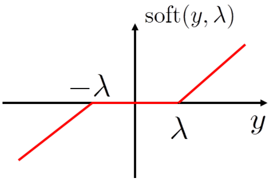
$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \lambda|w| = \text{soft}(y,\lambda) = \begin{cases} y-\lambda & \Leftarrow \quad y > \lambda \\ 0 & \Leftarrow \quad |y| \leq \lambda \\ y+\lambda & \Leftarrow \quad y < -\lambda \end{cases}$$

The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest case:

$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \lambda|w| = \text{soft}(y,\lambda) = \begin{cases} y-\lambda & \Leftarrow \quad y > \lambda \\ 0 & \Leftarrow \quad |y| \leq \lambda \\ y+\lambda & \Leftarrow \quad y < -\lambda \end{cases}$$

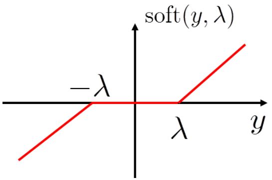


The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest case:

$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \lambda|w| = \mathrm{soft}(y,\lambda) = \begin{cases} y-\lambda & \Leftarrow \quad y > \lambda \\ 0 & \Leftarrow \quad |y| \le \lambda \\ y+\lambda & \Leftarrow \quad y < -\lambda \end{cases}$$



Contrast with the squared ℓ_2 (ridge) regularizer (linear scaling):

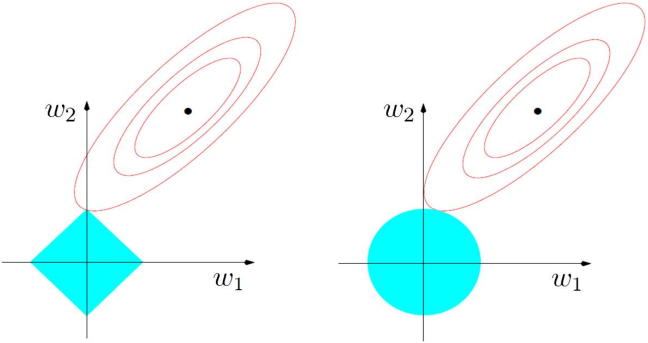
$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \frac{\lambda}{2}w^2 = \frac{1}{1+\lambda}\,y$$

The Lasso and Sparsity (II)

Why does the Lasso yield sparsity?

The Lasso and Sparsity (II)

Why does the Lasso yield sparsity?



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

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

Norms: A Quick Review

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha|\|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Examples of norms:

- $\|\mathbf{w}\|_1 = \left(\sum_i |w_i|\right)^1 = \sum_i |w_i|$.

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Examples of norms:

- $\|\mathbf{w}\|_1 = \left(\sum_i |w_i|\right)^1 = \sum_i |w_i|$.
- $\|\mathbf{w}\|_2 = \left(\sum_i |w_i|^2\right)^{1/2} = \sqrt{\sum_i |w_i|^2}$.

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Examples of norms:

- $\|\mathbf{w}\|_1 = \left(\sum_i |w_i|\right)^1 = \sum_i |w_i|$.
- $\|\mathbf{w}\|_2 = \left(\sum_i |w_i|^2\right)^{1/2} = \sqrt{\sum_i |w_i|^2}$.
- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Examples of norms:

- $\|\mathbf{w}\|_1 = \left(\sum_i |w_i|\right)^1 = \sum_i |w_i|$.
- $\|\mathbf{w}\|_2 = \left(\sum_i |w_i|^2\right)^{1/2} = \sqrt{\sum_i |w_i|^2}$.
- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- $\|\mathbf{w}\|_\infty = \lim_{p \to \infty} \|\mathbf{w}\|_p = \max\{|w_i|, i = 1, ..., D\}$

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha|\|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Examples of norms:

- $\|\mathbf{w}\|_1 = \left(\sum_i |w_i|\right)^1 = \sum_i |w_i|$.
- $\|\mathbf{w}\|_2 = \left(\sum_i |w_i|^2\right)^{1/2} = \sqrt{\sum_i |w_i|^2}$.
- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- $\|\mathbf{w}\|_\infty = \lim_{p \to \infty} \|\mathbf{w}\|_p = \max\{|w_i|, i = 1, ..., D\}$

Fact: all norms are convex.

Norms: A Quick Review

A norm is a function satisfying:

- $\|\alpha \mathbf{w}\| = |\alpha| \|\mathbf{w}\|$, for any $\mathbf{w}$ (homogeneity);
- $\|\mathbf{w} + \mathbf{w}'\| \leq \|\mathbf{w}\| + \|\mathbf{w}'\|$, for any $\mathbf{w}, \mathbf{w}'$ (triangle inequality);
- $\|\mathbf{w}\| = 0$ if and only if $\mathbf{w} = 0$.

Examples of norms:

- $\|\mathbf{w}\|_1 = (\sum_i |w_i|)^1 = \sum_i |w_i|$.
- $\|\mathbf{w}\|_2 = \left(\sum_i |w_i|^2\right)^{1/2} = \sqrt{\sum_i |w_i|^2}$.
- $\|\mathbf{w}\|_p = (\sum_i |w_i|^p)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- $\|\mathbf{w}\|_\infty = \lim_{p \to \infty} \|\mathbf{w}\|_p = \max\{|w_i|, i = 1, ..., D\}$

Fact: all norms are convex.

Also important (but not a norm): $\|\mathbf{w}\|_0 = \lim_{p \to 0} \|\mathbf{w}\|_p^p = |\{i : w_i \neq 0\}|$

Relationship Between ℓ_1 and ℓ_0

The ℓ_0 "norm" (number of non-zeros): $\|\mathbf{w}\|_0 = |\{i : w_i \neq 0\}|$.

Not convex, but...

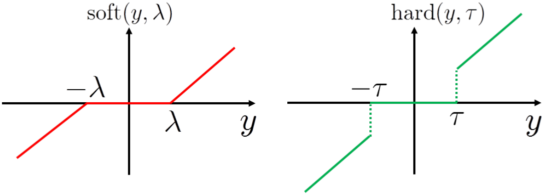
$$\widehat{w} = \arg\min_w \frac{1}{2}(w - y)^2 + \lambda |w|_0 = \mathrm{hard}(y, \sqrt{2\lambda}) = \begin{cases} y & \Leftarrow \quad |y| > \sqrt{2\lambda} \\ 0 & \Leftarrow \quad |y| \leq \sqrt{2\lambda} \end{cases}$$

Relationship Between ℓ_1 and ℓ_0

The ℓ_0 "norm" (number of non-zeros): $\|\mathbf{w}\|_0 = |\{i : w_i \neq 0\}|$.

Not convex, but...

$$\widehat{w} = \arg\min_w \frac{1}{2}(w-y)^2 + \lambda|w|_0 = \mathrm{hard}(y, \sqrt{2\lambda}) = \begin{cases} y & \Leftarrow \quad |y| > \sqrt{2\lambda} \\ 0 & \Leftarrow \quad |y| \leq \sqrt{2\lambda} \end{cases}$$

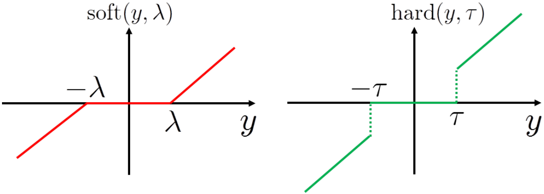


Relationship Between ℓ_1 and ℓ_0

The ℓ_0 "norm" (number of non-zeros): $\|\mathbf{w}\|_0 = |\{i : w_i \neq 0\}|$.

Not convex, but...

$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \lambda |w|_0 = \mathrm{hard}(y, \sqrt{2\lambda}) = \begin{cases} y & \Leftarrow \quad |y| > \sqrt{2\lambda} \\ 0 & \Leftarrow \quad |y| \leq \sqrt{2\lambda} \end{cases}$$



The "ideal" feature selection criterion (best subset):

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ &\quad \text{subject to } \|\mathbf{w}\|_0 \leq \tau \qquad \text{(limit the number of features)} \end{aligned}$$

Relationship Between ℓ_1 and ℓ_0 (II)

The best subset selection problem

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ &\quad \text{subject to } \|\mathbf{w}\|_0 \le \tau \end{aligned}$$

Relationship Between ℓ_1 and ℓ_0 (II)

The best subset selection problem is NP-hard Amaldi and Kann (1998)(Davis et al., 1997).

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ &\quad \text{subject to } \|\mathbf{w}\|_0 \leq \tau \end{aligned}$$

.

Relationship Between ℓ_1 and ℓ_0 (II)

The best subset selection problem is NP-hard Amaldi and Kann (1998)(Davis et al., 1997).

$$\begin{aligned}
\widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\
&\quad \text{subject to } \|\mathbf{w}\|_0 \leq \tau
\end{aligned}$$

A closely related problem,

$$\begin{aligned}
\widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \|\mathbf{w}\|_0 \\
&\quad \text{subject to } \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \leq \delta
\end{aligned}$$

.

Relationship Between ℓ_1 and ℓ_0 (II)

The best subset selection problem is NP-hard Amaldi and Kann (1998)(Davis et al., 1997).

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ & \quad \text{subject to } \|\mathbf{w}\|_0 \le \tau \end{aligned}$$

A closely related problem, also NP-hard (Muthukrishnan, 2005).

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \|\mathbf{w}\|_0 \\ & \quad \text{subject to } \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \le \delta \end{aligned}$$

Relationship Between ℓ_1 and ℓ_0 (II)

The best subset selection problem is NP-hard Amaldi and Kann (1998)(Davis et al., 1997).

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ &\quad \text{subject to } \|\mathbf{w}\|_0 \leq \tau \end{aligned}$$

A closely related problem, also NP-hard (Muthukrishnan, 2005).

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \|\mathbf{w}\|_0 \\ &\quad \text{subject to } \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \leq \delta \end{aligned}$$

In some cases, one may replace ℓ_0 with ℓ_1 and obtain "similar" results: central issue in compressive sensing (CS) (Candès et al., 2006; Donoho, 2006).

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)
- There are compelling algorithmic reasons for using convex surrogates like ℓ_1

Outline

Models

ℓ_1 regularization promotes **sparse models**

A very simple sparsity pattern: prefer models with **small cardinality**

Models

ℓ_1 regularization promotes **sparse models**

A very simple sparsity pattern: prefer models with **small cardinality**

Our main question: how can we promote less trivial sparsity patterns?

Models

ℓ_1 regularization promotes **sparse models**

A very simple sparsity pattern: prefer models with **small cardinality**

Our main question: how can we promote less trivial sparsity patterns?

We'll talk about structured sparsity and group-Lasso regularization.

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*

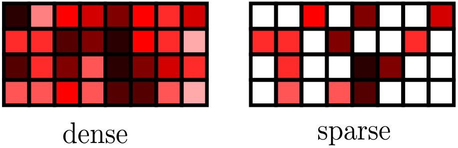
Leads to statistical gains if the prior assumptions are correct (Stojnic et al., 2009)

Tons of Uses

- feature template selection (Martins et al., 2011b)
- multi-task learning (Caruana, 1997; Obozinski et al., 2010)
- multiple kernel learning (Lanckriet et al., 2004)
- learning the structure of graphical models (Schmidt and Murphy, 2010)

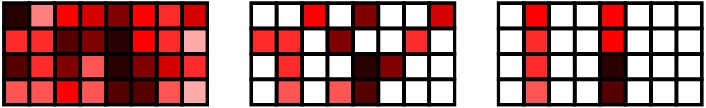
"Grid" Sparsity

For feature spaces that can be arranged as a grid (examples next)



"Grid" Sparsity

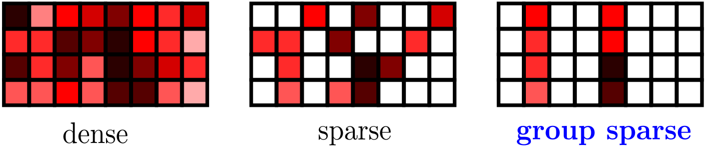
For feature spaces that can be arranged as a grid (examples next)



dense sparse **group sparse**

"Grid" Sparsity

For feature spaces that can be arranged as a grid (examples next)



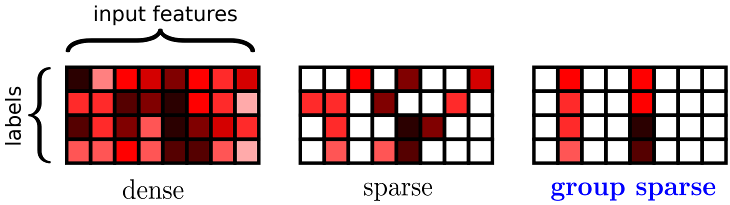
Goal: push *entire columns* to have zero weights

The groups are the columns of the grid

Example 1: Sparsity with Multiple Classes

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

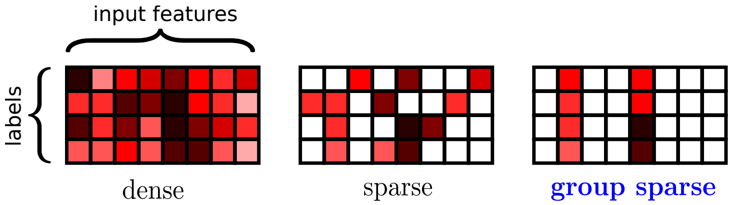
In words: we're conjoining each input feature with each output class



Example 1: Sparsity with Multiple Classes

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



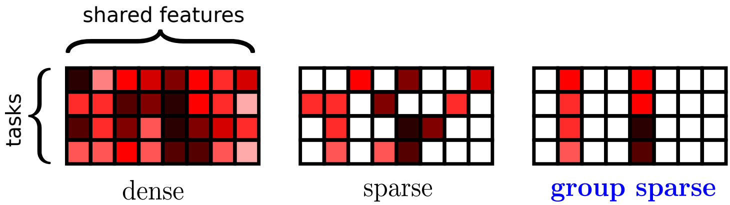
"Standard" sparsity is wasteful—we still need to hash all the input features

What we want: discard some input features, along with *each* class they conjoin with

Solution: one group per *input* feature

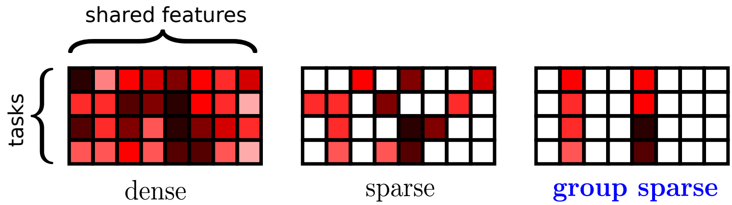
Example 2: Multi-Task Learning (Caruana, 1997; Obozinski et al., 2010)

Same thing, except now rows are **tasks** and columns are **features**



Example 2: Multi-Task Learning (Caruana, 1997; Obozinski et al., 2010)

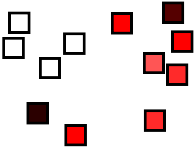
Same thing, except now rows are **tasks** and columns are **features**



What we want: discard features that are irrelevant for *all* tasks

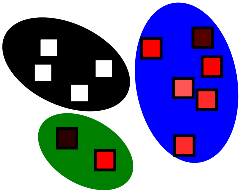
Solution: one group per feature

Group Sparsity



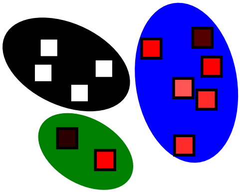
- D features

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

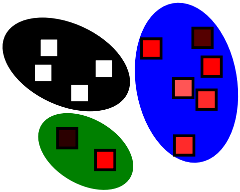


- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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

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

Group Sparsity



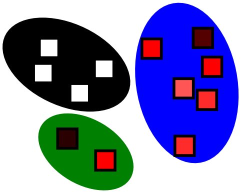
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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)

Regularization Formulations (reminder)

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

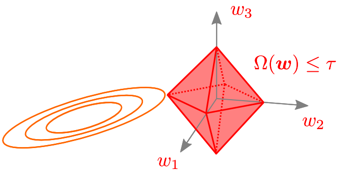
$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \\ & \text{subject to } \Omega(\mathbf{w}) \leq \tau \end{aligned}$$

- Morozov regularization

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

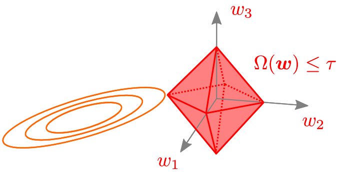
Equivalent, under mild conditions (namely convexity).

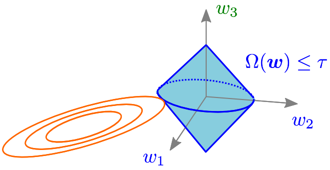
Lasso versus group-Lasso



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

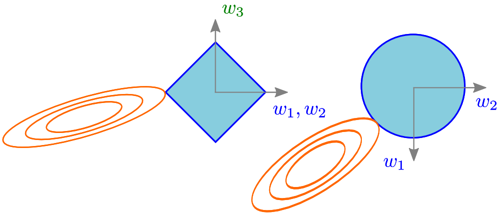
Lasso versus group-Lasso





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

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



Other names, other norms

Statisticians call these **composite absolute penalties** (Zhao et al., 2009)

In general: the (weighted) ℓ_r-**norm** of the ℓ_q-**norms** ($r \geq 1, q \geq 1$), called the mixed $\ell_{q,r}$ norm

$$\boxed{\Omega(\mathbf{w}) = \left(\sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_q^r\right)^{1/r}}$$

Other names, other norms

Statisticians call these **composite absolute penalties** (Zhao et al., 2009)

In general: the (weighted) ℓ_r-**norm** of the ℓ_q-**norms** ($r \geq 1, q \geq 1$), called the mixed $\ell_{q,r}$ norm

$$\boxed{\Omega(\mathbf{w}) = \left(\sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_q^r\right)^{1/r}}$$

Group sparsity corresponds to $r = 1$

Other names, other norms

Statisticians call these **composite absolute penalties** (Zhao et al., 2009)

In general: the (weighted) ℓ_r**-norm** of the ℓ_q**-norms** ($r \geq 1, q \geq 1$), called the mixed $\ell_{q,r}$ norm

$$\boxed{\Omega(\mathbf{w}) = \left(\sum_{m=1}^{M} \lambda_m \|\mathbf{w}_m\|_q^r\right)^{1/r}}$$

Group sparsity corresponds to $r = 1$

This talk: $q = 2$

However $q = \infty$ is also popular (Quattoni et al., 2009; Graça et al., 2009; Wright et al., 2009; Eisenstein et al., 2011)

Three Scenarios

- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured Groups

Three Scenarios

- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured 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

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

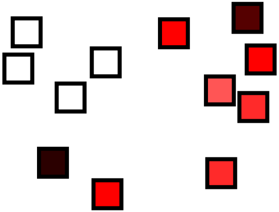
Trivial choices of groups recover *unstructured* regularizers:

- 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:

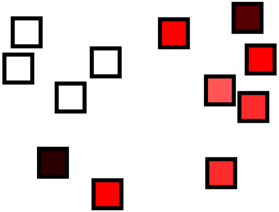
- label-based groups (groups are columns of a matrix)
- template-based groups (next)

Example: Feature Template Selection



Example: Feature Template Selection

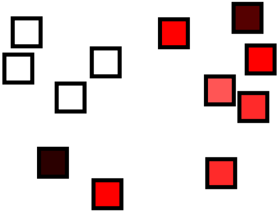
Input:	We	want	to	explore	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP



Example: Feature Template Selection

Input:	We	want	to	explore	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

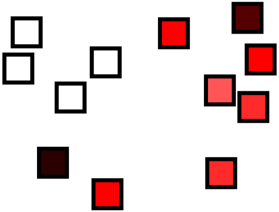
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

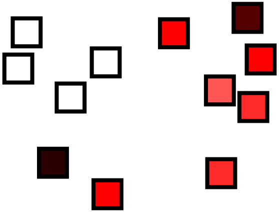
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

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

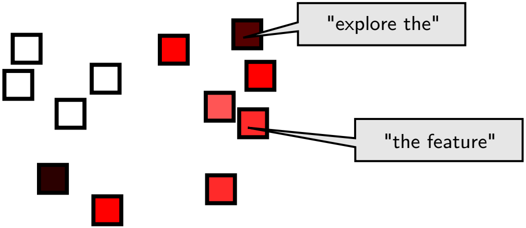
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

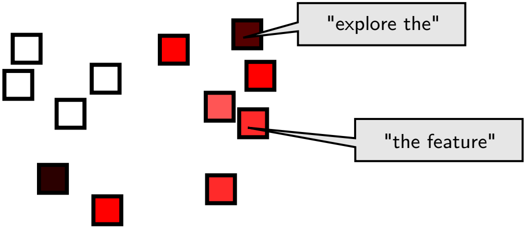
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

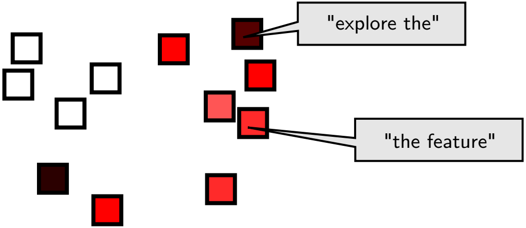
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

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

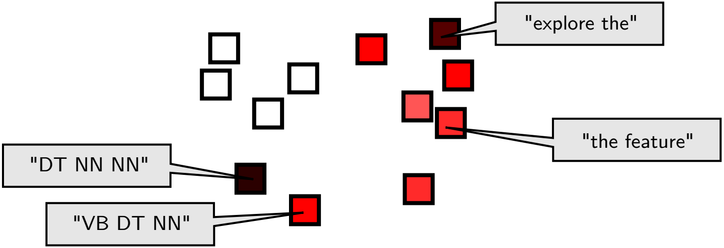
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

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

- Goal: Select relevant **feature templates**

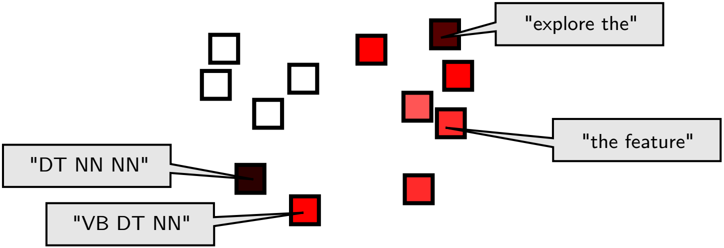


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

- Goal: Select relevant **feature templates**

  ⇒ Make each group correspond to a feature template

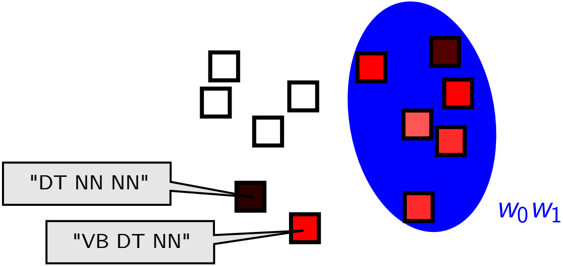


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

- Goal: Select relevant **feature templates**

  ⇒ Make each group correspond to a feature template

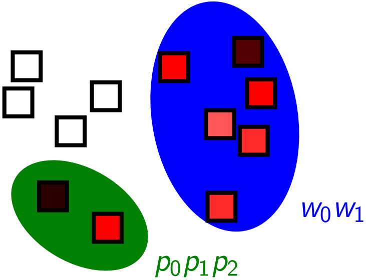


Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

- Goal: Select relevant **feature templates**

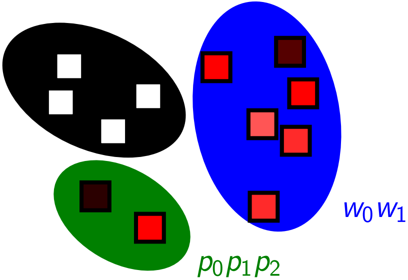
  $\Rightarrow$ Make each group correspond to a feature template



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

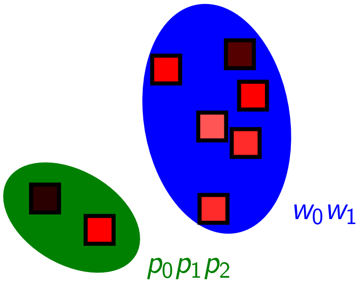
- Goal: Select relevant **feature templates**
  $\Rightarrow$ Make each group correspond to a feature template



Example: Feature Template Selection

Input:	We	want	to	*explore*	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP

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



Three Scenarios

- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured Groups

Three Scenarios

- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured Groups

Tree-Structured Groups

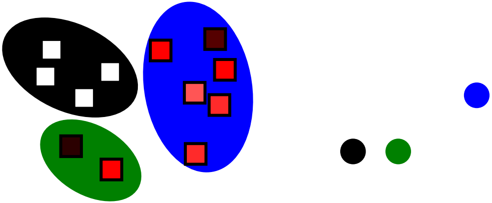
Assumption: if two groups overlap, one is contained in the other

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

Tree-Structured Groups

Assumption: if two groups overlap, one is contained in the other

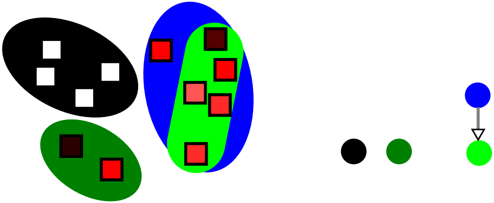
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one is contained in the other

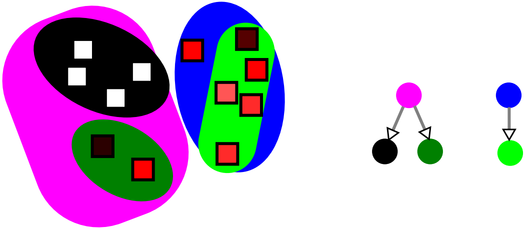
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one is contained in the other

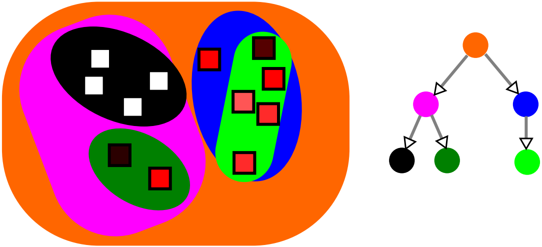
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one is contained in the other

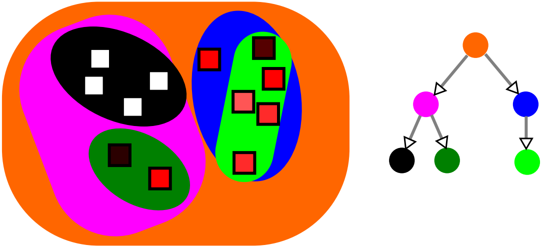
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one is contained in the other

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

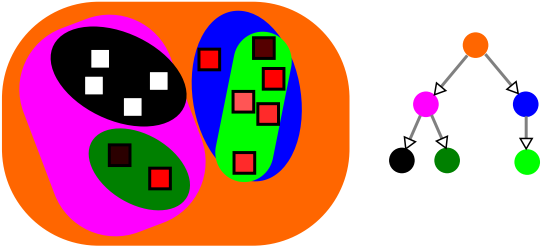


- What is the **sparsity pattern**?

Tree-Structured Groups

Assumption: if two groups overlap, one is contained in the other

⇒ **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

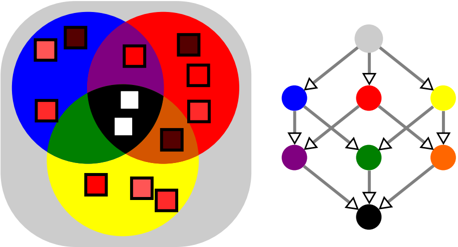
- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured Groups

Three Scenarios

- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured Groups

Graph-Structured Groups

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

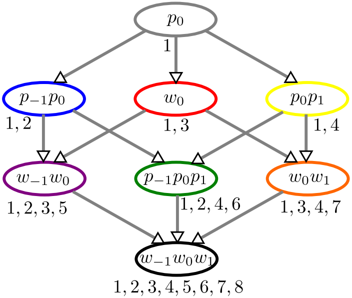


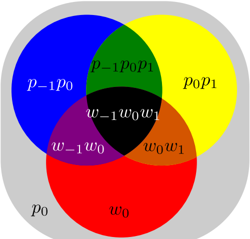
- set inclusion induces a **partial order** on groups (Jenatton et al., 2009)
- feature space becomes a **poset**
- **sparsity patterns**: given by this poset

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

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

Learning the Model

Recall that learning involves solving

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

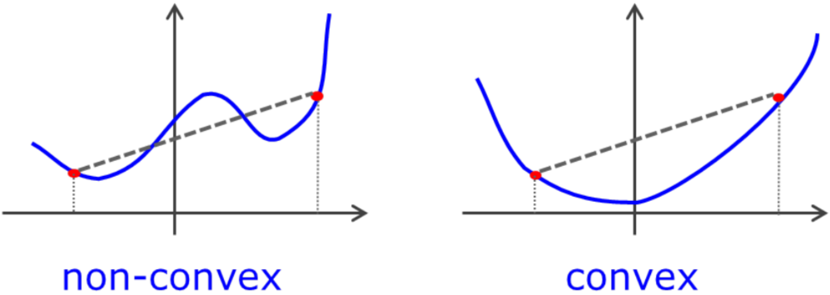
We'll address two kinds of optimization algorithms:

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

Key Concepts: Convex Functions

f is a convex function if:

$$\forall \lambda \in [0,1], x \text{ and } x' \in \text{domain}(f)$$
$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$



Outline

1. Introduction
2. Loss Functions and Sparsity
3. Structured Sparsity
4. **Algorithms**
   - Batch Algorithms
   - Online Algorithms
5. Applications
6. Conclusions

Batch Algorithms

- Subgradient methods
- Proximal methods
- Alternating direction method of multipliers

Key Concepts: Subgradients

Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

Subgradients generalize gradients for (maybe non-diff.) convex functions:

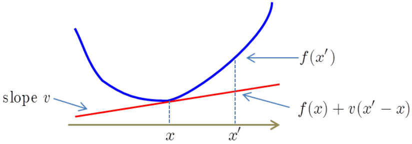
Key Concepts: Subgradients

Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

Subgradients generalize gradients for (maybe non-diff.) convex functions:

$$\boxed{\mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x} \text{ if } f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})}$$

Subdifferential: $\partial f(\mathbf{x}) = \{\mathbf{v} : \mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x}\}$



linear lower bound

Key Concepts: Subgradients

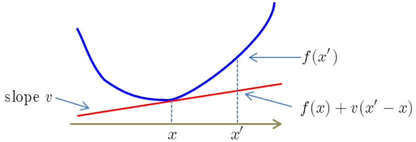
Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

Subgradients generalize gradients for (maybe non-diff.) convex functions:

$$\boxed{\mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x} \text{ if } f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})}$$

Subdifferential: $\partial f(\mathbf{x}) = \{\mathbf{v} : \mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x}\}$

If f is differentiable, $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$



linear lower bound

Key Concepts: Subgradients

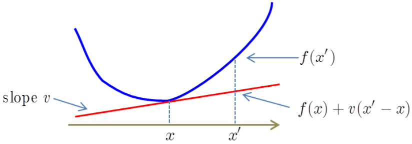
Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

Subgradients generalize gradients for (maybe non-diff.) convex functions:

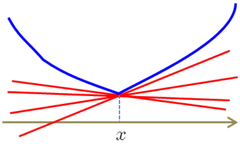
$$\boxed{\mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x} \text{ if } f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})}$$

Subdifferential: $\partial f(\mathbf{x}) = \{\mathbf{v} : \mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x}\}$

If f is differentiable, $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$



linear lower bound



non-differentiable case

Key Concepts: Subgradients

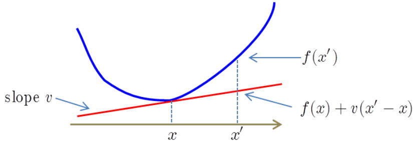
Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

Subgradients generalize gradients for (maybe non-diff.) convex functions:

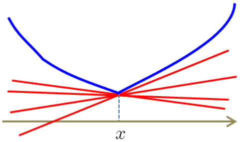
$$\boxed{\mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x} \text{ if } f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})}$$

Subdifferential: $\partial f(\mathbf{x}) = \{\mathbf{v} : \mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x}\}$

If f is differentiable, $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$



linear lower bound



non-differentiable case

Notation: $\tilde{\nabla} f(\mathbf{x})$ is a subgradient of f at $\mathbf{x}$

Subgradient Methods

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

Subgradient Methods

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

Subgradient methods were invented by Shor in the 1970's (Shor, 1985):

input: stepsize sequence $(\eta_t)_{t=1}^T$
initialize $\mathbf{w}$
for $t = 1, 2, \ldots$ **do**
  (sub-)gradient step: $\mathbf{w} \;\leftarrow\; \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla}\Lambda(\mathbf{w}) \right)$
end for

Subgradient Methods

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

Subgradient methods were invented by Shor in the 1970's (Shor, 1985):

input: stepsize sequence $(\eta_t)_{t=1}^{T}$
initialize $\mathbf{w}$
for $t = 1, 2, \ldots$ **do**
  (sub-)gradient step: $\mathbf{w} \;\leftarrow\; \mathbf{w} - \eta_t\left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla}\Lambda(\mathbf{w})\right)$
end for

Key disadvantages:

- The step size η_t needs to be annealed for convergence: very slow!
- Doesn't explicitly capture the sparsity promoted by sparse regularizers.

Key Concepts: Proximity Operators

Let $\Omega : \mathbb{R}^D \to \bar{\mathbb{R}}$ be a convex function.

Key Concepts: Proximity Operators

Let $\Omega : \mathbb{R}^D \to \bar{\mathbb{R}}$ be a convex function.

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

$$\boxed{\mathbf{w} \mapsto \mathrm{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

...always well defined, because $\|\mathbf{u} - \mathbf{w}\|_2^2$ is strictly convex.

Key Concepts: Proximity Operators

Let $\Omega : \mathbb{R}^D \to \bar{\mathbb{R}}$ be a convex function.

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

$$\boxed{\mathbf{w} \mapsto \text{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

...always well defined, because $\|\mathbf{u} - \mathbf{w}\|_2^2$ is strictly convex.

Classical examples:

- Squared ℓ_2 regularization, $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$: **scaling operation**

$$\text{prox}_{\Omega}(\mathbf{w}) = \frac{1}{1+\lambda}\mathbf{w}$$

Key Concepts: Proximity Operators

Let $\Omega : \mathbb{R}^D \to \bar{\mathbb{R}}$ be a convex function.

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

$$\boxed{\mathbf{w} \mapsto \text{prox}_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

...always well defined, because $\|\mathbf{u} - \mathbf{w}\|_2^2$ is strictly convex.

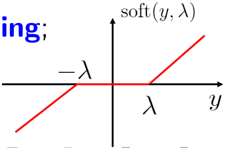
Classical examples:

- Squared ℓ_2 regularization, $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$: **scaling operation**

$$\text{prox}_{\Omega}(\mathbf{w}) = \frac{1}{1+\lambda}\mathbf{w}$$

- ℓ_1 regularization, $\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_1$: **soft-thresholding**;

$$\text{prox}_{\Omega}(\mathbf{w}) = \text{soft}(\mathbf{w}, \lambda)$$



Key Concepts: Proximity Operators (II)

$$\boxed{prox_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

Key Concepts: Proximity Operators (II)

$$\boxed{prox_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

- ℓ_2 regularization, $\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_2$: **vector soft thresholding**

$$\text{prox}_{\Omega}(\mathbf{w}) = \begin{cases} 0 & \Leftarrow \quad \|\mathbf{w}\| \leq \lambda \\ \frac{\mathbf{w}}{\|\mathbf{w}\|}\left(\|\mathbf{w}\| - \lambda\right) & \Leftarrow \quad \|\mathbf{w}\| > \lambda \end{cases}$$

Key Concepts: Proximity Operators (II)

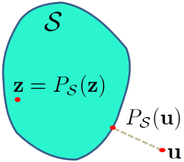
$$\boxed{prox_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

- ℓ_2 regularization, $\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_2$: **vector soft thresholding**

$$\text{prox}_{\Omega}(\mathbf{w}) = \begin{cases} 0 & \Leftarrow \quad \|\mathbf{w}\| \leq \lambda \\ \frac{\mathbf{w}}{\|\mathbf{w}\|}(\|\mathbf{w}\| - \lambda) & \Leftarrow \quad \|\mathbf{w}\| > \lambda \end{cases}$$

- indicator function, $\Omega(\mathbf{w}) = \iota_{\mathcal{S}}(\mathbf{w}) = \begin{cases} 0 & \Leftarrow \quad \mathbf{w} \in \mathcal{S} \\ +\infty & \Leftarrow \quad \mathbf{w} \notin \mathcal{S} \end{cases}$

$$\text{prox}_{\Omega}(\mathbf{w}) = P_{\mathcal{S}}(\mathbf{w})$$



Euclidean projection

Key Concepts: Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_m$ is a sub-vector of $\mathbf{w}$ with the indices in G_m.

Key Concepts: Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_m$ is a sub-vector of $\mathbf{w}$ with the indices in G_m.

- **Non-overlapping groups** ($G_m \cap G_n = \emptyset$, for $m \neq n$): separable prox operator

$$[\text{prox}_{\Omega}(\mathbf{w})]_m = \text{prox}_{\Omega_m}(\mathbf{w}_m)$$

Key Concepts: Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_m$ is a sub-vector of $\mathbf{w}$ with the indices in G_m.

- **Non-overlapping groups** ($G_m \cap G_n = \emptyset$, for $m \neq n$): separable prox operator
$$[\text{prox}_\Omega(\mathbf{w})]_m = \text{prox}_{\Omega_m}(\mathbf{w}_m)$$
- **Tree-structured groups:** (two groups are either non-overlapping or one contais the other) prox_Ω can be computed recursively (Jenatton et al., 2011).

Key Concepts: Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_m$ is a sub-vector of $\mathbf{w}$ with the indices in G_m.

- **Non-overlapping groups** ($G_m \cap G_n = \emptyset$, for $m \neq n$): separable prox operator
$$[\mathrm{prox}_\Omega(\mathbf{w})]_m = \mathrm{prox}_{\Omega_m}(\mathbf{w}_m)$$
- **Tree-structured groups:** (two groups are either non-overlapping or one contais the other) prox_Ω can be computed recursively (Jenatton et al., 2011).
- **Arbitrary groups:**
  - For $\Omega_j(\mathbf{w}_m) = \|\mathbf{w}_m\|_2$: solved via convex smooth optimization (Yuan et al., 2011).
  - Sequential proximity steps (Martins et al., 2011a) (more later).

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key assumptions: $\nabla\Lambda(\mathbf{w})$ and prox_{Ω} "easy".

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key assumptions: $\nabla\Lambda(\mathbf{w})$ and prox_{Ω} "easy".

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

Key feature: each steps decouples the loss and the regularizer.

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key assumptions: $\nabla\Lambda(\mathbf{w})$ and prox_{Ω} "easy".

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

Key feature: each steps decouples the loss and the regularizer.

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key assumptions: $\nabla\Lambda(\mathbf{w})$ and prox_{Ω} "easy".

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

Key feature: each steps decouples the loss and the regularizer.

Projected gradient is a particular case, for $\mathrm{prox}_{\Omega} = P_{\mathcal{S}}$.

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key assumptions: $\nabla\Lambda(\mathbf{w})$ and prox_{Ω} "easy".

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

Key feature: each steps decouples the loss and the regularizer.

Projected gradient is a particular case, for $\mathrm{prox}_{\Omega} = P_{\mathcal{S}}$.

Often called iterative shrinkage thresholding (IST).

Proximal Gradient

Recall the problem: $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key assumptions: $\nabla\Lambda(\mathbf{w})$ and prox_{Ω} "easy".

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

Key feature: each steps decouples the loss and the regularizer.

Projected gradient is a particular case, for $\text{prox}_{\Omega} = P_{\mathcal{S}}$.

Often called iterative shrinkage thresholding (IST).

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).

Monotonicity and Convergence

Proximal gradient, a.k.a., iterative shrinkage thresholding (IST):

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

Assume $\Lambda(\mathbf{w})$ has L-Lipschitz gradient: $\|\nabla\Lambda(\mathbf{w}) - \nabla\Lambda(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$.

Monotonicity and Convergence

Proximal gradient, a.k.a., iterative shrinkage thresholding (IST):

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

Assume $\Lambda(\mathbf{w})$ has L-Lipschitz gradient: $\|\nabla \Lambda(\mathbf{w}) - \nabla \Lambda(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$.

Monotonicity: if $\eta_t \leq 1/L$, then $\Lambda(\mathbf{w}_{t+1}) + \Omega(\mathbf{w}_{t+1}) \leq \Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)$.

Monotonicity and Convergence

Proximal gradient, a.k.a., iterative shrinkage thresholding (IST):

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

Assume $\Lambda(\mathbf{w})$ has L-Lipschitz gradient: $\|\nabla\Lambda(\mathbf{w}) - \nabla\Lambda(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$.

Monotonicity: if $\eta_t \leq 1/L$, then $\Lambda(\mathbf{w}_{t+1}) + \Omega(\mathbf{w}_{t+1}) \leq \Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)$.

Convergence of objective value (Beck and Teboulle, 2009)

$$\left(\Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)\right) - \left(\Lambda(\mathbf{w}^*) + \Omega(\mathbf{w}^*)\right) = O\left(\frac{1}{\epsilon}\right)$$

Accelerating IST: FISTA

Idea: compute $\mathbf{w}_{t+1}$ based, not only on $\mathbf{w}_t$, but also on $\mathbf{w}_{t-1}$.

Accelerating IST: FISTA

Idea: compute $\mathbf{w}_{t+1}$ based, not only on $\mathbf{w}_t$, but also on $\mathbf{w}_{t-1}$.

Fast IST algorithm (FISTA) (Beck and Teboulle, 2009):

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

Accelerating IST: FISTA

Idea: compute $\mathbf{w}_{t+1}$ based, not only on $\mathbf{w}_t$, but also on $\mathbf{w}_{t-1}$.

Fast IST algorithm (FISTA) (Beck and Teboulle, 2009):

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

Convergence of objective value (Beck and Teboulle, 2009)

$$\left(\Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)\right) - \left(\Lambda(\mathbf{w}^*) + \Omega(\mathbf{w}^*)\right) = O\left(\frac{1}{\sqrt{\epsilon}}\right) \quad \text{(vs } O(1/\epsilon) \text{ for IST)}$$

Accelerating IST: FISTA

Idea: compute $\mathbf{w}_{t+1}$ based, not only on $\mathbf{w}_t$, but also on $\mathbf{w}_{t-1}$.

Fast IST algorithm (FISTA) (Beck and Teboulle, 2009):

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

Convergence of objective value (Beck and Teboulle, 2009)

$$\big(\Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)\big) - \big(\Lambda(\mathbf{w}^*) + \Omega(\mathbf{w}^*)\big) = O\left(\frac{1}{\sqrt{\epsilon}}\right) \quad \text{(vs } O(1/\epsilon) \text{ for IST)}$$

Other IST variants: Nesterov's method (Nesterov, 2007), SpaRSA (Wright et al., 2009), TwIST (two-step IST; Bioucas-Dias and Figueiredo, 2007).

Alternating Direction Method of Multipliers

Combine benefits of dual decomposition and augmented Lagrangian methods for constrained optimization (Hestenes, 1969; Powell, 1969).

Alternating Direction Method of Multipliers

Combine benefits of dual decomposition and augmented Lagrangian methods for constrained optimization (Hestenes, 1969; Powell, 1969).

Key ideas

- break down the optimization problem into subproblems, each depending on a subset of $\mathbf{w}$.
- each subproblem p receives a "copy" of the subvector $\mathbf{w}$, denoted by $\mathbf{v}_p$.
- encode constraints forcing each $\mathbf{v}_p$ to "agree" with the global solution $\mathbf{w}$.

Alternating Direction Method of Multipliers

Combine benefits of dual decomposition and augmented Lagrangian methods for constrained optimization (Hestenes, 1969; Powell, 1969).

Key ideas

- break down the optimization problem into subproblems, each depending on a subset of $\mathbf{w}$.
- each subproblem p receives a "copy" of the subvector $\mathbf{w}$, denoted by $\mathbf{v}_p$.
- encode constraints forcing each $\mathbf{v}_p$ to "agree" with the global solution $\mathbf{w}$.

Particularly suitable for distributed optimization.

Alternating Direction Method of Multipliers

Original problem $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$ where $\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)}$.

Alternating Direction Method of Multipliers

Original problem $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$ where $\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)}$.

ADMM objective $\boxed{\min_{\mathbf{w},\mathbf{v}} \; \Omega(\mathbf{v}) + \Lambda(\mathbf{w})}$ subject to $\boxed{\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} = \mathbf{c}}$

Alternating Direction Method of Multipliers

Original problem $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$ where $\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)}$.

ADMM objective $\boxed{\min_{\mathbf{w},\mathbf{v}} \ \Omega(\mathbf{v}) + \Lambda(\mathbf{w})}$ subject to $\boxed{\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} = \mathbf{c}}$

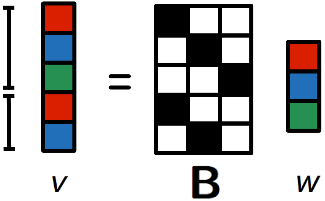
For example, in the overlapping group lasso case, we have $\mathbf{A} = \mathbf{I}$ and $\mathbf{c} = \mathbf{0}$. The constraint becomes $\mathbf{v} = -\mathbf{B}\mathbf{w}$.

Alternating Direction Method of Multipliers

Original problem $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$ where $\boxed{\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_m(\mathbf{w}_m)}$.

ADMM objective $\boxed{\min_{\mathbf{w},\mathbf{v}} \; \Omega(\mathbf{v}) + \Lambda(\mathbf{w})}$ subject to $\boxed{\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} = \mathbf{c}}$

For example, in the overlapping group lasso case, we have $\mathbf{A} = \mathbf{I}$ and $\mathbf{c} = \mathbf{0}$. The constraint becomes $\mathbf{v} = -\mathbf{B}\mathbf{w}$.



Alternating Direction Method of Multipliers

The augmented Lagrangian is:

$$\boxed{\Omega(\mathbf{v}) \quad + \Lambda(\mathbf{w}) + \mathbf{u}^\top(\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2}$$

Alternating Direction Method of Multipliers

The augmented Lagrangian is:

$$\boxed{\Omega(\mathbf{v}) \quad + \Lambda(\mathbf{w}) + \mathbf{u}^\top(\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2}$$

ADMM iteratively solves:

$$\boxed{\begin{aligned}
\hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \Lambda(\mathbf{w}) + \mathbf{u}^\top \mathbf{B}\mathbf{w} + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2 \\
\hat{\mathbf{v}} &= \arg\min_{\mathbf{v}} \Omega(\mathbf{v}) + \mathbf{u}^\top \mathbf{A}\mathbf{v} + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2 \\
\mathbf{u} &= \mathbf{u} + \rho(\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c})
\end{aligned}}$$

Alternating Direction Method of Multipliers

The augmented Lagrangian is:

$$\boxed{\Omega(\mathbf{v}) \quad +\Lambda(\mathbf{w}) + \mathbf{u}^\top(\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2}$$

ADMM iteratively solves:

$$\boxed{\begin{aligned} \hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \Lambda(\mathbf{w}) + \mathbf{u}^\top\mathbf{B}\mathbf{w} + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2 \\ \hat{\mathbf{v}} &= \arg\min_{\mathbf{v}} \Omega(\mathbf{v}) + \mathbf{u}^\top\mathbf{A}\mathbf{v} + \frac{\rho}{2}\|\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}\|_2^2 \\ \mathbf{u} &= \mathbf{u} + \rho(\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} - \mathbf{c}) \end{aligned}}$$

Key advantage: the minimization of **v** can be done in parallel.

Alternating Direction Method of Multipliers

Convergence of ADMM in theory (Boyd et al., 2010)

Alternating Direction Method of Multipliers

Convergence of ADMM in theory (Boyd et al., 2010)

Assumptions:

- Λ and Ω are closed, proper, and convex.
- The unaugmented Lagrangian has a saddle point

Alternating Direction Method of Multipliers

Convergence of ADMM in theory (Boyd et al., 2010)

Assumptions:

- Λ and Ω are closed, proper, and convex.
- The unaugmented Lagrangian has a saddle point

As $t \to \infty$, we have:

- Residual convergence: $\mathbf{Av} + \mathbf{Bw} - \mathbf{c} \to 0$.
- Primal convergence: $\Lambda(\mathbf{w}_t) + \Omega(\mathbf{v}_t) \to p^*$ where p^* is the optimal value.
- Dual convergence: $\mathbf{u}_t \to \mathbf{u}^*$.

Alternating Direction Method of Multipliers

ADMM can handle various kinds of regularizers by adapting **A** and **B**.

Alternating Direction Method of Multipliers

ADMM can handle various kinds of regularizers by adapting $\mathbf{A}$ and $\mathbf{B}$.

ADMM is well suited for structured sparse models with group overlaps because we can design $\mathbf{A}$ and $\mathbf{B}$ such that $\Omega(\mathbf{v})$ no longer has overlapping groups. Hence, we can solve each subproblem separately in parallel.

Alternating Direction Method of Multipliers

ADMM can handle various kinds of regularizers by adapting $\mathbf{A}$ and $\mathbf{B}$.

ADMM is well suited for structured sparse models with group overlaps because we can design $\mathbf{A}$ and $\mathbf{B}$ such that $\Omega(\mathbf{v})$ no longer has overlapping groups. Hence, we can solve each subproblem separately in parallel.

Practical considerations:

- ADMM can be slow to converge in practice, but tens of iterations are often enough to produce good results.
- ADMM only produces weakly sparse solution (we only get sparsity in the limit).

Alternating Direction Method of Multipliers

Recall that the ADMM objective is:

$$\boxed{\min_{\mathbf{w},\mathbf{v}} \ \Omega_{\text{struct}}(\mathbf{v}) + \Lambda(\mathbf{w})} \text{ subject to } \boxed{\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} = \mathbf{c}}$$

Alternating Direction Method of Multipliers

Recall that the ADMM objective is:

$$\boxed{\min_{\mathbf{w},\mathbf{v}} \; \Omega_{\text{struct}}(\mathbf{v}) + \Lambda(\mathbf{w})} \text{ subject to } \boxed{\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} = \mathbf{c}}$$

We can introduce an additional lasso penalty (sparse group lasso; Friedman et al., 2010):

$$\boxed{\min_{\mathbf{w},\mathbf{v}} \; \Omega_{\text{struct}}(\mathbf{v}) + \Omega_{\text{lasso}}(\mathbf{w}) + \Lambda(\mathbf{w})} \text{ subject to } \boxed{\mathbf{A}\mathbf{v} + \mathbf{B}\mathbf{w} = \mathbf{c}}$$

Alternating Direction Method of Multipliers

Recall that the ADMM objective is:

$$\boxed{\min_{\mathbf{w},\mathbf{v}} \; \Omega_{\text{struct}}(\mathbf{v}) + \Lambda(\mathbf{w})} \text{ subject to } \boxed{\mathbf{Av} + \mathbf{Bw} = \mathbf{c}}$$

We can introduce an additional lasso penalty (sparse group lasso; Friedman et al., 2010):

$$\boxed{\min_{\mathbf{w},\mathbf{v}} \; \Omega_{\text{struct}}(\mathbf{v}) + \Omega_{\text{lasso}}(\mathbf{w}) + \Lambda(\mathbf{w})} \text{ subject to } \boxed{\mathbf{Av} + \mathbf{Bw} = \mathbf{c}}$$

We get sparse solutions and can still guarantee convergence (Yogatama and Smith, 2014a).

Summary of Algorithms

	Converges?	Rate?	Sparse?	Groups?	Overlaps?
Prox-grad (IST)	✓	$O(1/\epsilon)$	✓	✓	Not easy
FISTA	✓	$O(1/\sqrt{\epsilon})$	✓	✓	Not easy
ADMM	✓	$O(1/\epsilon)$	No	✓	✓

Summary of Algorithms

	Converges?	Rate?	Sparse?	Groups?	Overlaps?
Prox-grad (IST)	✓	$O(1/\epsilon)$	✓	✓	Not easy
FISTA	✓	$O(1/\sqrt{\epsilon})$	✓	✓	Not easy
ADMM	✓	$O(1/\epsilon)$	No	✓	✓

Note that we can still get sparsity for ADMM with sparse group lasso (Yogatama and Smith, 2014a).

Some Stuff We Didn't Talk About

- shooting method (Fu, 1998);
- grafting (Perkins et al., 2003) and grafting-light (Zhu et al., 2010); (Afonso et al., 2010; Figueiredo and Bioucas-Dias, 2011).
- forward stagewise regression (Hastie et al., 2007).
- homotopy/continuation method (Osborne et al., 2000; Efron et al., 2004; Figueiredo et al., 2007; Hale et al., 2008).

Some Stuff We Didn't Talk About

- shooting method (Fu, 1998);
- grafting (Perkins et al., 2003) and grafting-light (Zhu et al., 2010); (Afonso et al., 2010; Figueiredo and Bioucas-Dias, 2011).
- forward stagewise regression (Hastie et al., 2007).
- homotopy/continuation method (Osborne et al., 2000; Efron et al., 2004; Figueiredo et al., 2007; Hale et al., 2008).

Next: We'll talk about online algorithms.

Outline

Why Online?

Batch

Online

Why Online?

Batch

Online

1. Suitable for large datasets

Why Online?

Batch

Online

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

Why Online?

Batch

Online

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

Why Online?

Batch

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
   - cf. "the tradeoffs of large scale learning" (Bottou and Bousquet, 2007)

Why Online?

Batch

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
   - cf. "the tradeoffs of large scale learning" (Bottou and Bousquet, 2007)

What we will say can be straighforwardly extended to the mini-batch case.

Plain Stochastic (Sub-)Gradient Descent

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

Plain Stochastic (Sub-)Gradient Descent

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

input: stepsize sequence $(\eta_t)_{t=1}^{T}$
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

What's the Problem with SGD?

(Sub-)gradient step: $\boxed{\mathbf{w} \;\leftarrow\; \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla} L(\mathbf{w}; x_t, y_t)\right)}$

What's the Problem with SGD?

(Sub-)gradient step: $\boxed{\mathbf{w} \leftarrow \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla} L(\mathbf{w}; x_t, y_t)\right)}$

- ℓ_2-regularization $\boxed{\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \mathbf{w}$

$$\mathbf{w} \leftarrow \underbrace{(1 - \eta_t \lambda)\mathbf{w}}_{\text{scaling}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

What's the Problem with SGD?

(Sub-)gradient step: $\boxed{\mathbf{w} \;\leftarrow\; \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla} L(\mathbf{w}; x_t, y_t)\right)}$

- ℓ_2-regularization $\boxed{\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \mathbf{w}$

$$\mathbf{w} \;\leftarrow\; \underbrace{(1 - \eta_t \lambda)\mathbf{w}}_{\text{scaling}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

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

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

What's the Problem with SGD?

(Sub-)gradient step: $\boxed{\mathbf{w} \leftarrow \mathbf{w} - \eta_t \left(\tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla} L(\mathbf{w}; x_t, y_t)\right)}$

- ℓ_2-regularization $\boxed{\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \mathbf{w}$

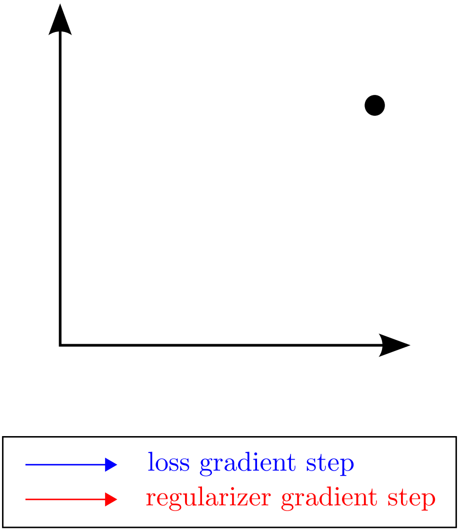
$$\mathbf{w} \leftarrow \underbrace{(1 - \eta_t\lambda)\mathbf{w}}_{\text{scaling}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

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

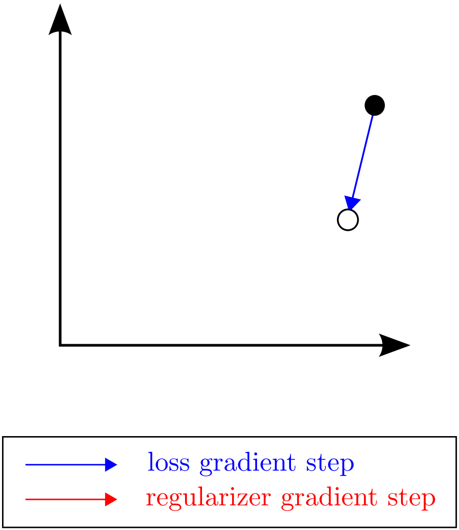
$$\mathbf{w} \leftarrow \underbrace{\mathbf{w} - \eta_t\lambda \mathrm{sign}(\mathbf{w})}_{\text{constant penalty}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

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

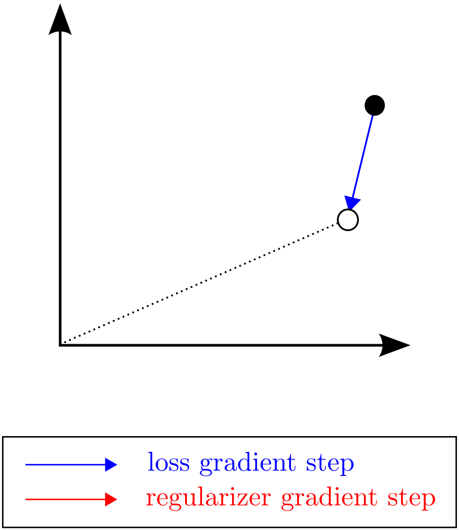
Plain SGD with ℓ_2-regularization



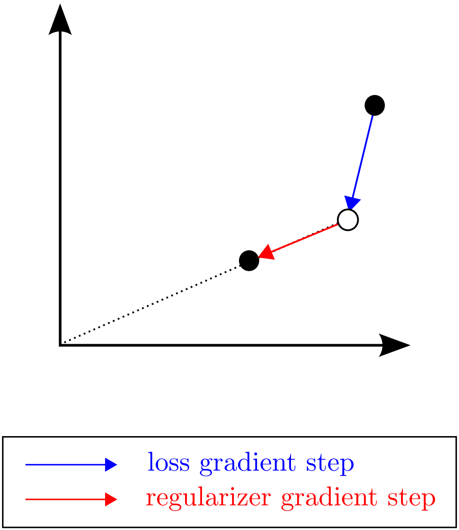
Plain SGD with ℓ_2-regularization



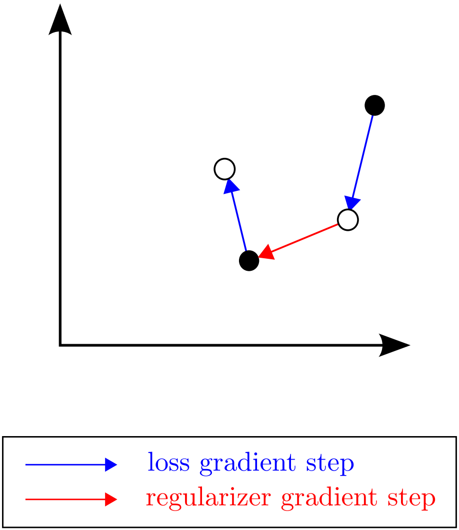
Plain SGD with ℓ_2-regularization



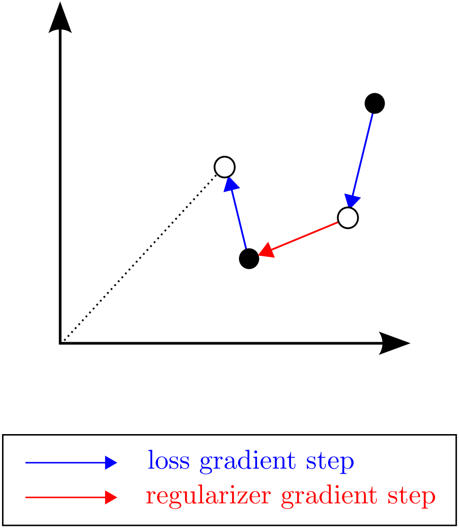
Plain SGD with ℓ_2-regularization



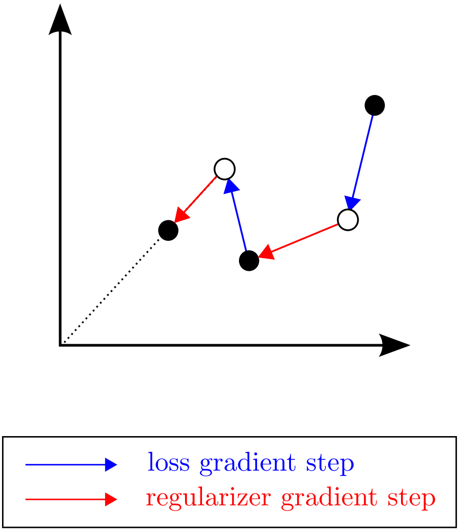
Plain SGD with ℓ_2-regularization



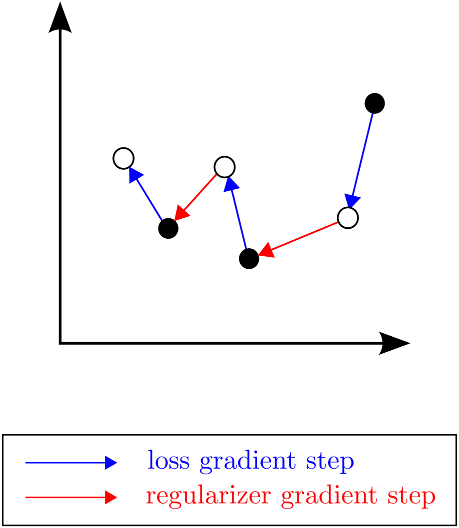
Plain SGD with ℓ_2-regularization



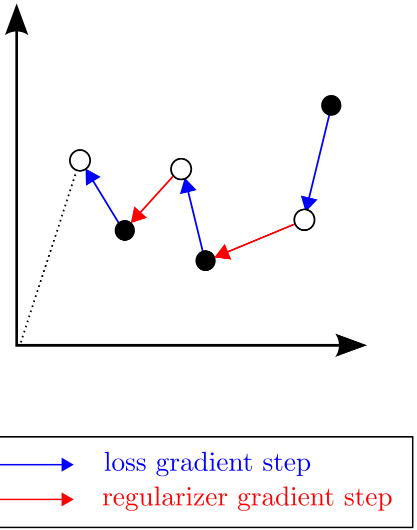
Plain SGD with ℓ_2-regularization



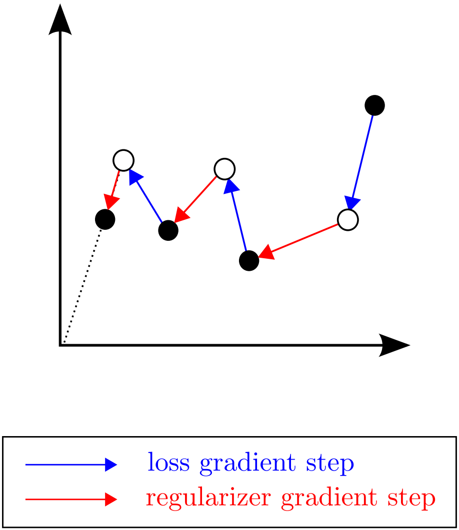
Plain SGD with ℓ_2-regularization



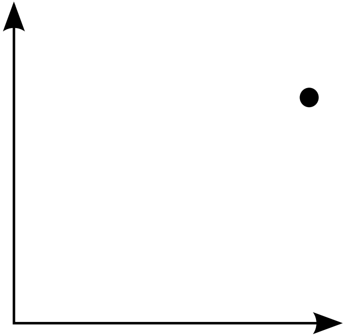
Plain SGD with ℓ_2-regularization

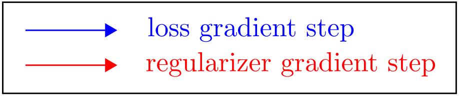


Plain SGD with ℓ_2-regularization

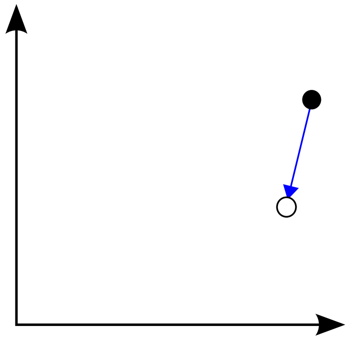


Plain SGD with ℓ_1-regularization

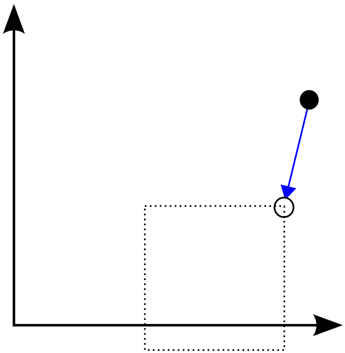


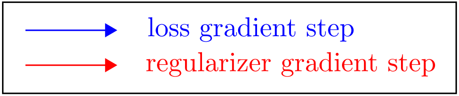


Plain SGD with ℓ_1-regularization

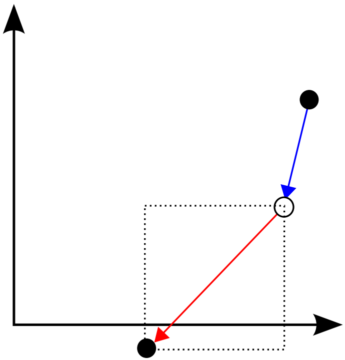


Plain SGD with ℓ_1-regularization



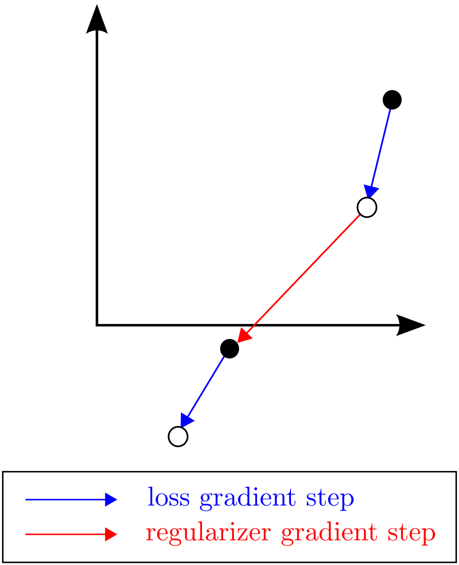


Plain SGD with ℓ_1-regularization

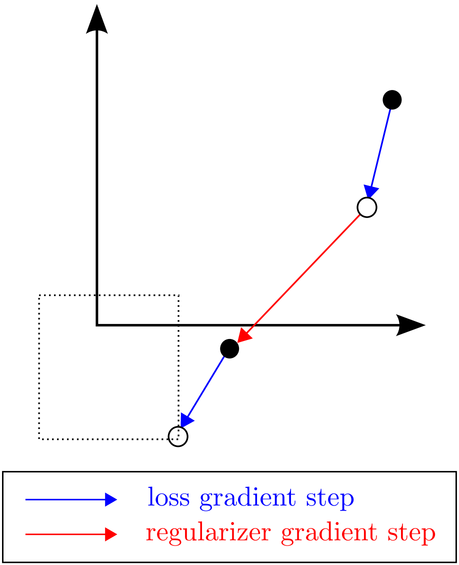


loss gradient step
regularizer gradient step

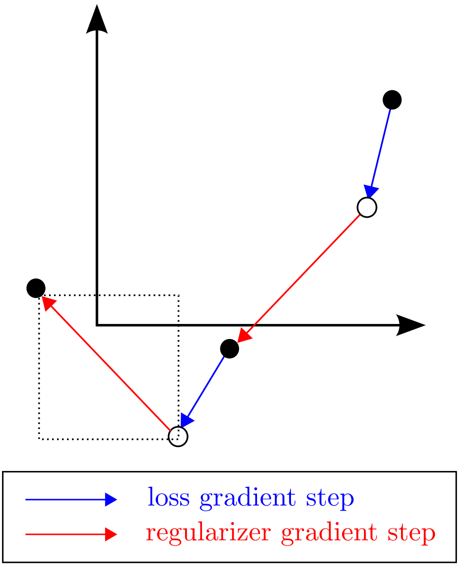
Plain SGD with ℓ_1-regularization



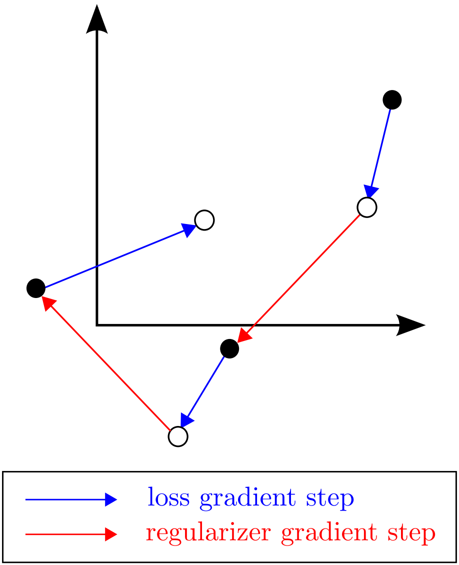
Plain SGD with ℓ_1-regularization



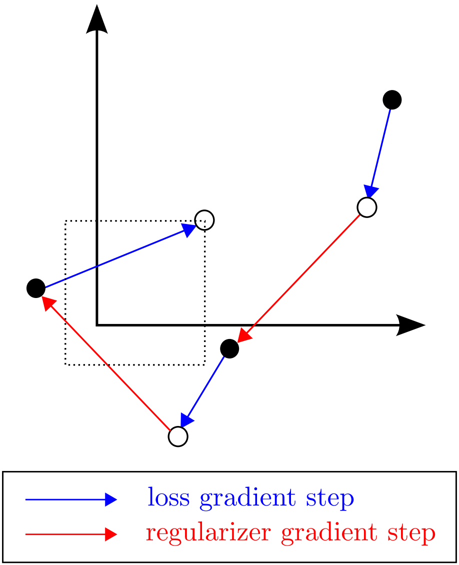
Plain SGD with ℓ_1-regularization



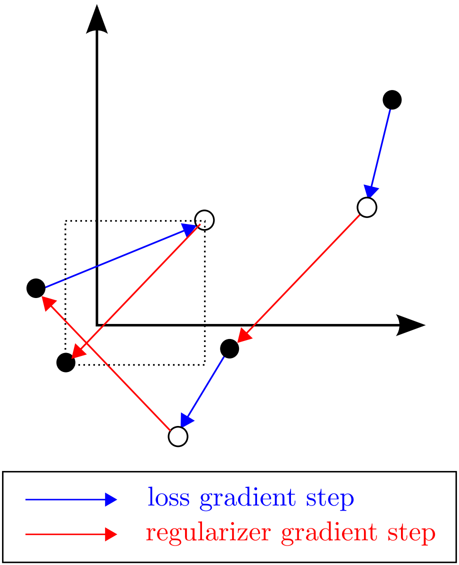
Plain SGD with ℓ_1-regularization



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)

“Sparse” Online Algorithms

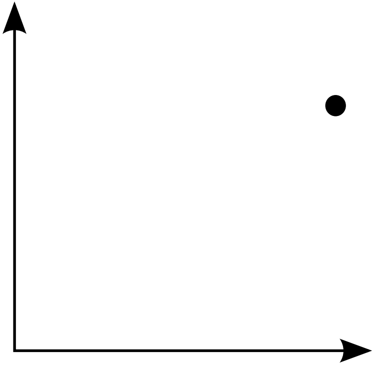
- Truncated Gradient (Langford et al., 2009)
- Online Forward-Backward Splitting (Duchi and Singer, 2009)
- Regularized Dual Averaging (Xiao, 2010)
- Online Proximal Gradient (Martins et al., 2011a)

Truncated Gradient (Langford et al., 2009)

input: laziness coefficient K, stepsize sequence $(\eta_t)_{t=1}^{T}$
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  (sub-)gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \tilde{\nabla} L(\boldsymbol{\theta}; x_t, y_t)$
  if t/K is integer **then**
    truncation step: $\mathbf{w} \leftarrow \underbrace{\mathbf{w} - \mathrm{sign}(\mathbf{w})\,(|\mathbf{w}| - \eta_t K \tau)}_{\textbf{soft-thresholding}}$
  end if
end for

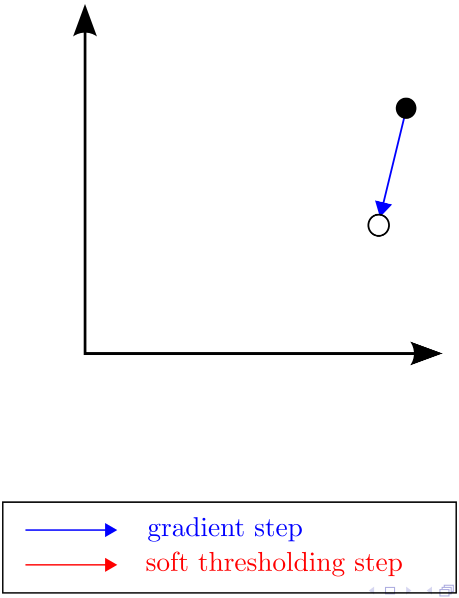
- take gradients **only with respect to the loss**
- **every K rounds:** a "lazy" soft-thresholding step
- Langford et al. (2009) also suggest other forms of truncation
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**

Truncated Gradient (Langford et al., 2009)

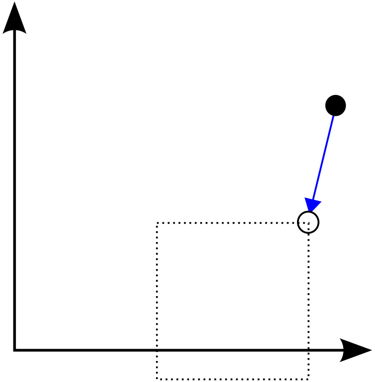


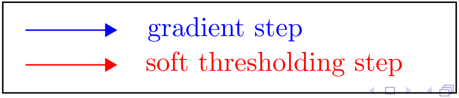
gradient step
soft thresholding step

Truncated Gradient (Langford et al., 2009)

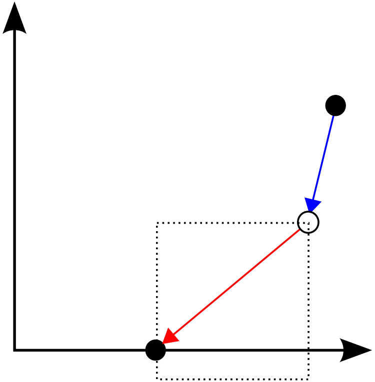


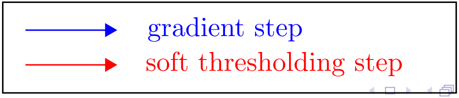
Truncated Gradient (Langford et al., 2009)



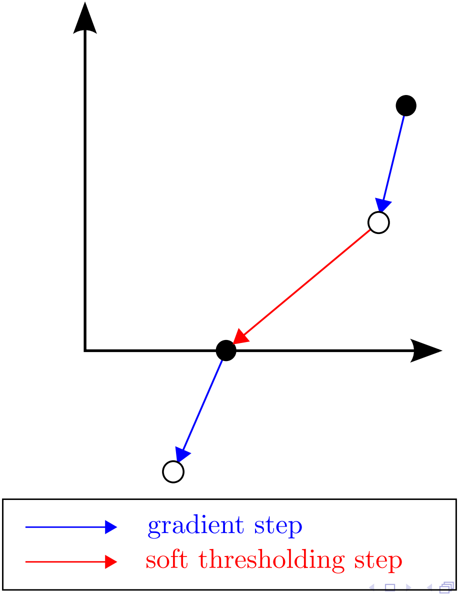


Truncated Gradient (Langford et al., 2009)

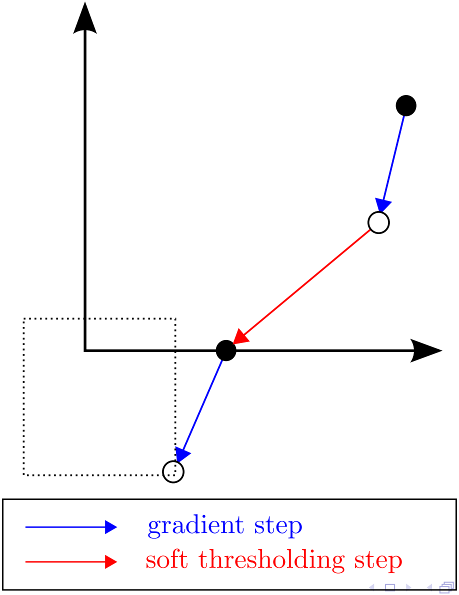




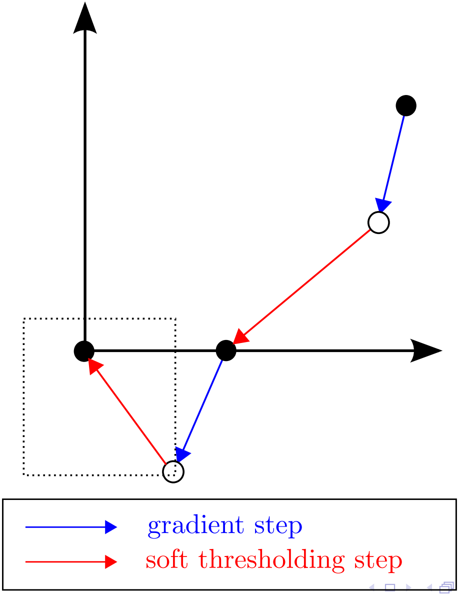
Truncated Gradient (Langford et al., 2009)



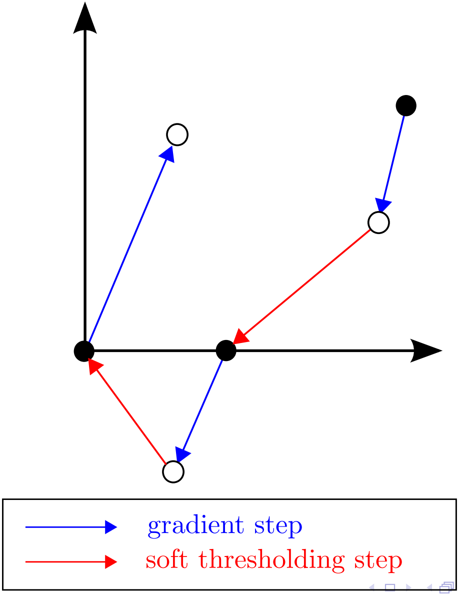
Truncated Gradient (Langford et al., 2009)



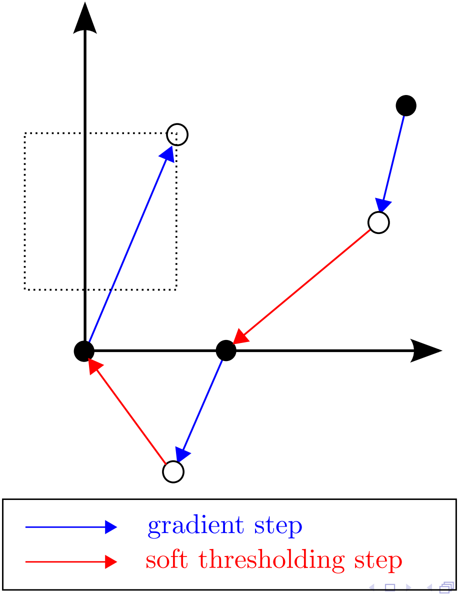
Truncated Gradient (Langford et al., 2009)



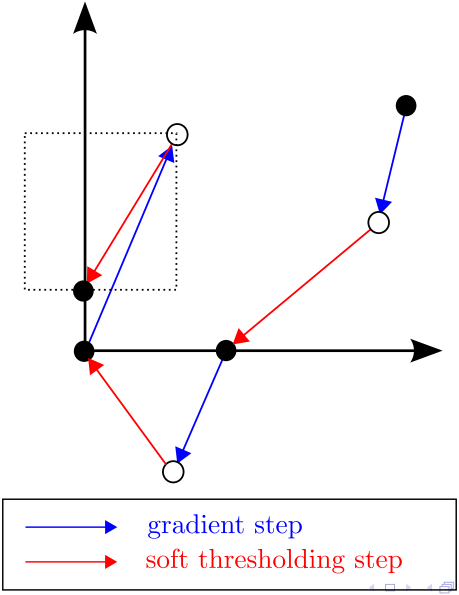
Truncated Gradient (Langford et al., 2009)



Truncated Gradient (Langford et al., 2009)



Truncated Gradient (Langford et al., 2009)



“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)

input: stepsize sequences $(\eta_t)_{t=1}^{T}$, $(\rho_t)_{t=1}^{T}$
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: $\mathbf{w} \leftarrow \text{prox}_{\rho_t \Omega}(\mathbf{w})$
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)
- **practical drawback:** without a laziness parameter, iterates are usually not very sparse
- **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)

Regularized Dual Averaging (Xiao, 2010)

input: coefficient η_0
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  gradient step: $\mathbf{s} \leftarrow \mathbf{s} + \nabla L(\mathbf{w}; x_t, y_t)$
  proximal step: $\mathbf{w} \leftarrow \eta_0 \sqrt{t} \times \mathrm{prox}_{\Omega}(-\mathbf{s}/t)$
end for

- based on the **dual averaging technique** (Nesterov, 2009)
- **in practice:** quite effective at getting sparse iterates (the proximal steps are not vanishing)
- $O(C_1/\epsilon^2 + C_2/\sqrt{\epsilon})$ **convergence**, where C_1 is a Lipschitz constant, and C_2 is the variance of the stochastic gradients
- **drawback:** requires storing two vectors ($\mathbf{w}$ and $\mathbf{s}$), and $\mathbf{s}$ is not sparse

What About Group Sparsity?

Both online forward-backward splitting (Duchi and Singer, 2009) and regularized dual averaging (Xiao, 2010) can handle groups

All that is necessary is to compute $\text{prox}_{\Omega}(\mathbf{w})$

- easy for non-overlapping and tree-structured groups
- **But what about general overlapping groups?**

Martins et al. (2011a): a prox-grad algorithm that can handle arbitrary overlapping groups

- decompose $\Omega(\mathbf{w}) = \sum_{j=1}^{J} \Omega_j(\mathbf{w})$ where each Ω_j is **non-overlapping**
- then apply prox_{Ω_j} *sequentially*
- still convergent (Martins et al., 2011a)

“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 Proximal Gradient (Martins et al., 2011a)

input: gravity sequence $(\sigma_t)_{t=1}^T$, stepsize sequence $(\eta_t)_{t=1}^T$
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla L(\boldsymbol{\theta}; x_t, y_t)$
  sequential proximal steps:
  for $j = 1, 2, \ldots$ **do**
    $\mathbf{w} \leftarrow \mathrm{prox}_{\eta_t \sigma_t \Omega_j}(\mathbf{w})$
  end for
end for

Online Proximal Gradient (Martins et al., 2011a)

input: gravity sequence $(\sigma_t)_{t=1}^T$, stepsize sequence $(\eta_t)_{t=1}^T$
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla L(\boldsymbol{\theta}; x_t, y_t)$
  sequential proximal steps:
  for $j = 1, 2, \ldots$ **do**
    $\mathbf{w} \leftarrow \mathrm{prox}_{\eta_t \sigma_t \Omega_j}(\mathbf{w})$
  end for
end for

- **PAC Convergence.** ϵ-accurate solution after $T \leq O(1/\epsilon^2)$ rounds
- **Computational efficiency.** Each gradient step is **linear** in the number of features that fire.
  Each proximal step is **linear** in the number of groups M.
  Both are **independent** of D.

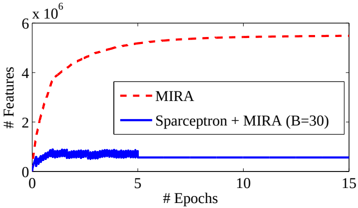
Implementation Tricks (Martins et al., 2011b)

- **Budget driven shrinkage.** Instead of a regularization constant, specify a *budget* on the number of selected groups. Each proximal step sets σ_t to meet this target.
- **Sparseptron.** Let $L(\mathbf{w}) = \mathbf{w}^\top(\mathbf{f}(x, \hat{y}) - \mathbf{f}(x, y))$ be the perceptron loss. The algorithm becomes perceptron with shrinkage.
- **Debiasing.** Run a few iterations of sparseptron to identify the relevant groups. Then run a unregularized learner at a second stage.

Implementation Tricks (Martins et al., 2011b)

- **Budget driven shrinkage.** Instead of a regularization constant, specify a *budget* on the number of selected groups. Each proximal step sets σ_t to meet this target.
- **Sparseptron.** Let $L(\mathbf{w}) = \mathbf{w}^\top(\mathbf{f}(x, \hat{y}) - \mathbf{f}(x, y))$ be the perceptron loss. The algorithm becomes perceptron with shrinkage.
- **Debiasing.** Run a few iterations of sparseptron to identify the relevant groups. Then run a unregularized learner at a second stage.
- **Memory efficiency.** Only a small active set of features need to be maintained. Entire groups can be deleted after each proximal step.

  Many irrelevant features are never instantiated.



Summary of Algorithms

	Converges?	Rate?	Sparse?	Groups?	Overlaps?
Prox-grad (IST)	✓	$O(1/\epsilon)$	✓	✓	Not easy
FISTA	✓	$O(1/\sqrt{\epsilon})$	✓	✓	Not easy
ADMM	✓	$O(1/\epsilon)$	No	✓	✓
Online subgradient	✓	$O(1/\epsilon^2)$	No	✓	No
Truncated gradient	✓	$O(1/\epsilon^2)$	✓	No	No
FOBOS	✓	$O(1/\epsilon^2)$	Sort of	✓	Not easy
RDA	✓	$O(1/\epsilon^2)$	✓	✓	Not easy
Online prox-grad	✓	$O(1/\epsilon^2)$	✓	✓	✓

Outline

Applications of Structured Sparsity in NLP

1. Non-overlapping groups by feature template
2. Tree-structured groups: coarse-to-fine
3. Arbitrarily overlapping groups

Applications of Structured Sparsity in NLP

1. Non-overlapping groups by feature template
2. Tree-structured groups: coarse-to-fine
3. Arbitrarily overlapping groups

Martins et al. (2011b): Group by Template

Feature templates provide a straightforward way to define non-overlapping groups.

To achieve group sparsity, we optimize:

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

where we use the $\ell_{2,1}$ norm:

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

for M groups/templates.

Structured Prediction Tasks (Martins et al., 2011b)

- **Chunking** (CoNLL 2000 shared task; Sang and Buchholz, 2000)
  $+0.5$ F_1 with 30 groups (out of 96)
- **NER** (CoNLL 2002/3 shared tasks on Spanish, Dutch, English; Sang, 2002; Sang and De Meulder, 2003)
  $+1$–2 F_1 with 200 groups (out of 452)
- **Dependency parsing** (CoNLL-X shared task on several languages; Buchholz and Marsi, 2006), 684 feature templates based on McDonald et al. (2005)

Which features get selected?

- Qualitative analysis of selected templates:

	Arabic	Danish	Japanese	Slovene	Spanish	Turkish
Bilexical	++	+			+	
Lex. → POS	+		+			
POS → Lex.	++	+	+		+	+
POS → POS			++	+		
Middle POS	++	++	++	++	++	++
Shape	++	++	++	++		
Direction		+	+	+	+	+
Distance	++	+	+	+	+	+

  (Empty: none or very few templates selected; +: some templates selected; ++: most or all templates selected.)
- Morphologically-rich languages with small datasets (Turkish and Slovene) avoid lexical features.
- In Japanese, contextual POS appear to be especially relevant.

Which features get selected?

- Qualitative analysis of selected templates:

	Arabic	Danish	Japanese	Slovene	Spanish	Turkish
Bilexical	++	+			+	
Lex. → POS	+		+			
POS → Lex.	++	+	+		+	+
POS → POS			++	+		
Middle POS	++	++	++	++	++	++
Shape	++	++	++	++		
Direction		+	+	+	+	+
Distance	++	+	+	+	+	+

(Empty: none or very few templates selected; +: some templates selected; ++: most or all templates selected.)

- Morphologically-rich languages with small datasets (Turkish and Slovene) avoid lexical features.
- In Japanese, contextual POS appear to be especially relevant.
- **Take this with a grain of salt:** some patterns may be properties of the datasets, not the languages!

Sociolinguistic Association Discovery (Eisenstein et al., 2011)

- Dataset:
  - geotagged tweets from 9,250 authors
  - mapping of locations to the U.S. Census' ZIP code tabulation areas (ZCTAs)
  - a ten-dimensional vector of statistics on demographic attributes
- Can we learn a compact set of terms used on Twitter that associate with demographics?

Sociolinguistic Association Discovery (Eisenstein et al., 2011)

- Setup: multi-output regression.
  - x_n is a P-dimensional vector of independent variables; matrix is $\mathbf{X} \in \mathbb{R}^{N \times P}$
  - y_n is a T-dimensional vector of dependent variables; matrix is $\mathbf{Y} \in \mathbb{R}^{N \times T}$
  - $w_{p,t}$ is the regression coefficient for the pth variable in the tth task; matrix is $\mathbf{W} \in \mathbb{R}^{P \times T}$
  - Regularized objective with squared error loss typical for regression:

$$\min_{\mathbf{W}} \Omega(\mathbf{W}) + \|\mathbf{Y} - \mathbf{X}\mathbf{W}\|_F^2$$

- Regressions are run in *both* directions.

Structured Sparsity with $\ell_{\infty,1}$

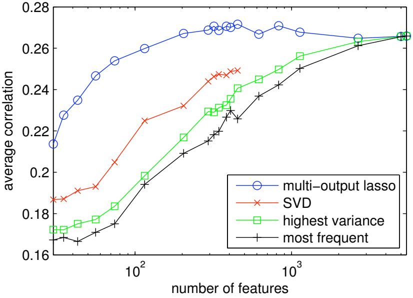
- Drive entire rows of **W** to zero (Turlach et al., 2005): "some predictors are useless for *any* task"

$$\Omega(\mathbf{W}) = \lambda \sum_{t=1}^{T} \max_{p} w_{p,t}$$

- Optimization with blockwise coordinate ascent (Liu et al., 2009) and some tricks to maintain sparsity (Eisenstein et al., 2011)
- See also: Duh et al. (2010) used multitask regression and $\ell_{2,1}$ to select features useful for reranking across many instances (application in machine translation).

Predicting Demographics from Text (Eisenstein et al., 2011)

- Predict 10-dimensional ZCTA characterization from words tweeted in that region (vocabulary is $P = 5,418$)
- Measure Pearson's correlation between prediction and correct value (average over tasks, cross-validated test sets)
- Compare with truncated SVD, greatest variance across authors, most frequent words



Predictive Words (Eisenstein et al., 2011)

	white	Afr. Am.	Hisp.	Eng. lang.	Span. lang.	other lang.	urban	family	renter	med. inc.
-_-	-		+	-	+	+	+			
;)		-	+	-	+					
:(		-								
:)		-								
:d	+	-	+	-	+					
as			-	+	-					
awesome	+	-					-		-	+
break			-	+	-	-				
campus			-	+	-	-				
dead	-	+		-	+		+		+	
hell			-	+	-	-				
shit	-								+	
train				-	+				+	
will			-	+	-					
would				+					-	
atlanta			-	+	-	-				
famu		+	-	+	-	-				-
harlem				-					+	
bbm	-	+		-		+	+		+	
lls		+	-	+	-	-				
lmaoo	-	+	+	-	+	+	+		+	
lmaooo	-	+	+	-	+	+	+		+	
lmaoooo	-	+	+	-	+	+			+	
lmfaoo	-		+	-	+	+			+	
lmfaooo	-		+	-	+	+			+	
lml	-	+	+	-	+	+	+		+	-
odee	-		+	-	+		+		+	

	white	Afr. Am.	Hisp.	Eng. lang.	Span. lang.	other lang.	urban	family	renter	med. inc.
omw	-	+	+	-	+	+	+		+	
smfh	-	+	+	-	+	+	+		+	
smh	-	+					+		+	
w|	-		+	-	+	+	+		+	
con			+	-	+				+	
la		-	+	-	+					
si		-	+	-	+					
dats	-	+		-					+	-
deadass	-	+	+	-	+	+	+		+	
haha	+	-							-	
hahah	+	-								
hahaha	+	-							-	+
ima	-		+	-	+				+	
madd	-			-		+		+		
nah	-		+	-	+	+			+	
ova	-	+		-					+	
sis	-	+							+	
skool	-	+		-		+	+		+	-
wassup	-	+	+	-	+	+	+		+	-
wat	-	+	+	-	+	+	+		+	-
ya	-	+							+	
yall	-	+								
yep			-	+	-	-	-		-	
yoo	-	+	+	-	+	+	+		+	
yooo	-	+		-	+				+	

Table: Demographically-indicative terms discovered by multi-output sparse regression. Statistically significant ($p < .05$) associations are marked (+/-).

Non-overlapping Groups for "Some" Ambiguity

Learning mappings from word types to labels (POS or semantic predicates)

- Semisupervised lexicon expansion with graph-based learning (Das and Smith, 2012)
  - Elitist lasso (squared $\ell_{1,2}$; Kowalski and Torrésani, 2009) for per-word sparsity

    $$\lambda \sum_{v} \left(\sum_{y} |w_{v,y}| \right)^2$$

    where v is a word and y is a label.
  - $+3\%$ accuracy on unknown-word frame prediction, with 35% as many lexicon entries
- Unsupervised POS tagging with posterior regularization (Graça et al., 2009)
  - Incorporates $\ell_{\infty,1}$ norm
  - $+2$–7% accuracy on 1-many POS evaluation

Applications of Structured Sparsity in NLP

1. Non-overlapping groups by feature template
2. Tree-structured groups: coarse-to-fine
3. Arbitrarily overlapping groups

Log-Linear Language Models (Nelakanti et al., 2013)

Setup: multinomial logistic regression (Della Pietra et al., 1997)

$$p(y \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{f}(\mathbf{x}))}{\sum_{v \in V} \exp(\mathbf{w}_v^\top \mathbf{f}(\mathbf{x}))}$$

Log-Linear Language Models (Nelakanti et al., 2013)

Setup: multinomial logistic regression (Della Pietra et al., 1997)

$$p(y \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{f}(\mathbf{x}))}{\sum_{v \in V} \exp(\mathbf{w}_v^\top \mathbf{f}(\mathbf{x}))}$$

Regularized objective with logistic loss:

$$\min_{\mathbf{w}} - \sum_{i=1}^{N} \log p(y_i \mid \mathbf{x}_{1:k}; \mathbf{w}) + \Omega(\mathbf{w})$$

Log-Linear Language Models (Nelakanti et al., 2013)

Setup: multinomial logistic regression (Della Pietra et al., 1997)

$$p(y \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{f}(\mathbf{x}))}{\sum_{v \in V} \exp(\mathbf{w}_v^\top \mathbf{f}(\mathbf{x}))}$$

Regularized objective with logistic loss:

$$\min_{\mathbf{w}} - \sum_{i=1}^{N} \log p(y_i \mid \mathbf{x}_{1:k}; \mathbf{w}) + \Omega(\mathbf{w})$$

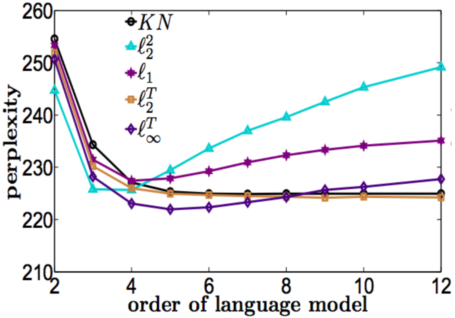
There are many choices for $\Omega(\mathbf{w})$. A key consideration is that the size of $\mathbf{w}$ increases rapidly as k gets bigger.

Log-Linear Language Models (Nelakanti et al., 2013)

- Encode history suffixes from length 0 to k in a tree; each is a feature.
- Tree-structured penalty: a longer suffix can only be included if all its shorter suffixes are included.
  - Can use $\ell_{2,1}$ or $\ell_{\infty,1}$ norm

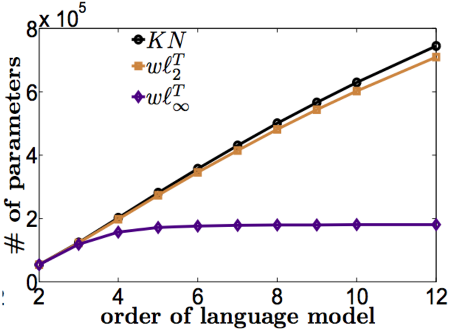
Experimental Results: AP-news

Good generalization results (perplexity):



Experimental Results: AP-news

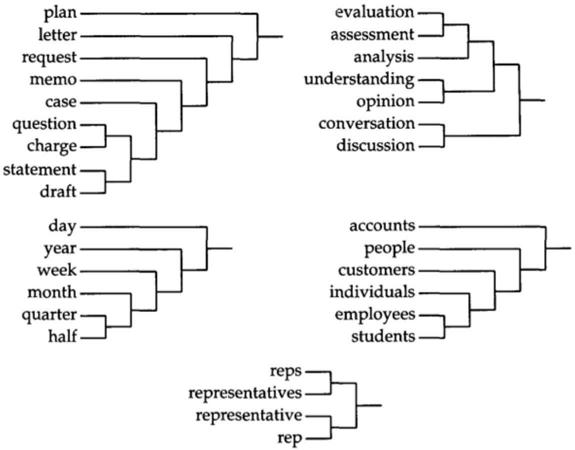
Small model size:



Groups from Word Clusters (Yogatama and Smith, 2014a)

- Task: text classification
- Model: bag-of-words logistic regression
- Hierarchical clusters from Brown et al. (1992): include the words in a cluster only if its parent cluster is included.

Brown et al. (1992) Clusters



Regularize or Add Features?

- 20-newsgroups binary tasks:

		+ Brown features			Brown
dataset	**baseline**	**lasso**	**ridge**	**elastic**	**group lasso**
science	91.90 (ridge)	86.96	90.51	91.14	**93.04**
sports	93.71 (elastic)	82.66	88.94	85.43	**93.71**
religion	92.47 (ridge)	94.98	**96.93**	**96.93**	92.89
computer	87.13 (elastic)	55.72	**96.65**	67.57	86.36

- Caveat: we ought to use more data to learn the clusters!

Applications of Structured Sparsity in NLP

1. Non-overlapping groups by feature template
2. Tree-structured groups: coarse-to-fine
3. Arbitrarily overlapping groups

Groups from Data (Yogatama and Smith, 2014b)

- Task: text classification
- Model: bag-of-words logistic regression
- Groups: one group for every sentence in every training-set document
  - Intuition: only some sentences are relevant
  - Past work used latent "relevance" variables (Yessenalina et al., 2010; Tackstrom and McDonald, 2011)
- Use ADMM to handle thousands/millions of overlapping groups.
  - Copy weights allow inspection to see which training sentences are "selected"
  - Additional ℓ_1 penalty for strong sparsity

Topic Classification (IBM vs. Mac)

Sentence	Negative	Positive
from : *anonymized*		
subject : accelerating the macplus ... ;)		(0.05)
lines : 15 we ’ re about ready to take a bold step into the 90s around here by accelerating our rather large collection of stock macplus computers .		(0.07) (0.03) (0.02) (0.02)
yes indeed , difficult to comprehend why anyone would want to accelerate a macplus, but that’s another story .		(0.06) (0.02) (0.02) (0.04)
suffuce it to say , we can get accelerators easier than new machines .		(0.01)
hey , i don ’ t make the rules ...		(0.01)
anyway , on to the purpose of this post: i ’ m looking for info on macplus acelerators .		(0.04) (0.01)
so far , i ’ ve found some lit on the novy accelerator and the micrmac multispeed accelartor .		(0.02) (0.02) (0.02) (0.04)
both look acceptable , but i would like to hear from anyone who has tried these .	(–0.01)	
also , if someone would recommend another accelerator for the macplus , i ’ d like to hear about it .		(0.06) (0.03) (0.02) (0.06)
thanks for any time and effort you expend on this !	(–0.01) (–0.01) (–0.01)	
karl		

Bars show log-odds effect of removing the sentence: **sentence**, **elastic**, **ridge**, **lasso**.

Sentiment Analysis (Amazon DVDs; Blitzer et al., 2007)

Sentence	Negative	Positive
this film is one big joke : you have all the basics elements of romance (love at first sight , great passion , etc .) and gangster flicks (brutality , dagerous machinations , the mysterious don , etc.) , but it is all done with the crudest humor .		(0.42) (0.22) (0.07) (0.48)
it ' s the kind of thing you either like viserally and immediately " get " or you don ' t .		(0.01) (0.01)
that is a matter of taste and expectations .		(0.01)
i enjoyed it and it took me back to the mid80s , when nicolson and turner were in their primes .		(0.02) (0.01)
the acting is very good , if a bit obviously tongue - in - cheek .		(0.01)

Bars show log-odds effect of removing the sentence: **sentence**, **elastic**, **ridge**, **lasso**.

Outline

Summary

- Sparsity is desirable in NLP: *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.
- We have seen examples for feature template selection, tree structures, and data-driven groups, but many more are possible!
- Small/medium scale: many batch algorithms available, with fast convergence (IST, FISTA, SpaRSA, ...)
- Large scale: distributed optimization algorithms (ADMM) or 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), grants PEst-OE/EEI/LA0008/2011 and PTDC/EEI-SII/2312/2012.
- 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, Intelligo project, contract 2012/24803.

priberam

POR LISBOA PROGRAMA OPERACIONAL REGIONAL

QREN QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL PORTUGAL 2007.2013

UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

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.

Amaldi, E. and Kann, V. (1998). On the approximation of minimizing non zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260.

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

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.

Bioucas-Dias, J. and Figueiredo, M. (2007). A new twist: two-step iterativeshrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16:2992–3004.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*.

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

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

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.

Cessie, S. L. and Houwelingen, J. C. V. (1992). Ridge estimators in logistic regression. *Journal of the Royal Statistical Society; Series C*, 41:191–201.

Chen, S. and Rosenfeld, R. (1999). A Gaussian prior for smoothing maximum entropy models. Technical report, CMU-CS-99-108.

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

References II

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

Das, D. and Smith, N. A. (2012). Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of NAACL*.

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.

Davis, G., Mallat, S., and Avellaneda, M. (1997). Greedy adaptive approximation. *Journal of Constructive Approximation*, 13:57–98.

Della Pietra, S., Della Pietra, V., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

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

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

Duh, K., Sudoh, K., Tsukada, H., Isozaki, H., and Nagata, M. (2010). *n*-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics*.

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

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.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. Technical report, Stanford University.

References III

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

Goodman, J. (2004). Exponential priors for maximum entropy models. In *Proc. of NAACL*.

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.

Hale, E., Yin, W., and Zhang, Y. (2008). Fixed-point continuation for l1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19:1107–1130.

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

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320.

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. (2011). Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334.

Kazama, J. and Tsujii, J. (2003). Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. of EMNLP*.

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

Kowalski, M. and Torrésani, B. (2009). Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3(3):251–264.

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.

Liu, H., Palatucci, M., and Zhang, J. (2009). Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 649–656. ACM.

References IV

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*.

Martins, A. F. T., Smith, N. A., Xing, E. P., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2010). Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

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

Muthukrishnan, S. (2005). *Data Streams: Algorithms and Applications*. Now Publishers, Boston, MA.

Nelakanti, A., Archambeau, C., Mairal, J., Bach, F., and Bouchard, G. (2013). Structured penalties for log-linear language models. In *Proc. of EMNLP*.

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Technical report, CORE report.

Nesterov, Y. (2009). Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259.

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.

Powell, M. J. D. (1969). A method for nonlinear constraints in minimization problems. In Fletcher, R., editor, *Optimization*, pages 283–298. Academic Press.

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

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*.

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

References V

Sang, E. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*.

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

Schaefer, R., Roi, L., and Wolfe, R. (1984). A ridge logistic estimator. *Communications in Statistical Theory and Methods*, 13:99–113.

Schmidt, M. and Murphy, K. (2010). Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proc. of AISTATS*.

Shor, N. (1985). *Minimization Methods for Non-differentiable Functions*. Springer.

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.

Tackstrom, O. and McDonald, R. (2011). Discovering fine-grained sentiment with latent variable structured prediction models. In *Proc. of ECIR*.

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.

Turlach, B. A., Venables, W. N., and Wright, S. J. (2005). Simultaneous variable selection. *Technometrics*, 47(3):349–363.

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

Williams, P. (1995). Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7:117–143.

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.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

References VI

Yessenalina, A., Yue, Y., and Cardie, C. (2010). Multi-level structured models for document sentiment classification. In *Proc. of EMNLP*.

Yogatama, D. and Smith, N. A. (2014a). Linguistic structured sparsity in text categorization. In *Proc. of ACL*.

Yogatama, D. and Smith, N. A. (2014b). Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *Proc. of ICML*.

Yuan, L., Liu, J., and Ye, J. (2011). Efficient methods for overlapping group lasso. In *Advances in Neural Information Processing Systems 24*, pages 352–360.

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.