Structured Sparsity in Natural Language Processing:

Models, Algorithms, and Applications

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

[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

NAACL 2012: Tutorials
Montréal, Québec, June 3, 2012
Slides online at http://tiny.cc/ssnlp

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 5 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

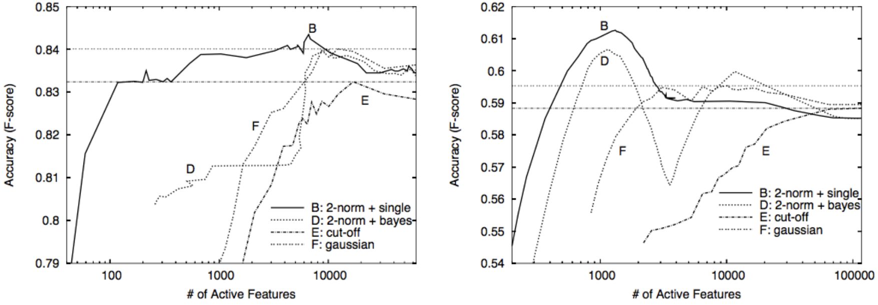
Computational savings.

- $w_d = 0$ 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.



(a) Reuters

(b) OHSUMED

Experimental results from Kazama and Tsujii (2003): F_1 on two text categorization tasks as the number of features varies.

(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

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

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

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:

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

- Total loss:

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

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 $A_{ij} = f_j(x_i)$.

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 $A_{ij} = f_j(x_i)$.
- Response vector: $\mathbf{y} = [y_1, \ldots, y_N]^\top$.

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 $A_{ij} = f_j(x_i)$.
- 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):

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

Loss functions (II)

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

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

- Partition function:

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

Loss functions (II)

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

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

- Partition function:

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

- 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.)

All these losses are particular cases of general family (Martins et al., 2010).

Regularization

Regularized parameter estimate:

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

where $\Omega(\mathbf{w}) \geq 0$ and $\lim_{\|\mathbf{w}\| \to \infty} \Omega(\mathbf{w}) = \infty$ (**coercive** function).

Why regularize?

- Improve generalization by avoiding over-fitting.

Regularization

Regularized parameter estimate:

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

where $\Omega(\mathbf{w}) \geq 0$ and $\lim_{\|\mathbf{w}\| \to \infty} \Omega(\mathbf{w}) = \infty$ (**coercive** function).

Why regularize?

- Improve generalization by avoiding over-fitting.
- The total loss may not be coercive (*e.g.*, logistic loss on separable data), thus having no minima.

Regularization

Regularized parameter estimate:

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

where $\Omega(\mathbf{w}) \geq 0$ and $\lim_{\|\mathbf{w}\| \to \infty} \Omega(\mathbf{w}) = \infty$ (**coercive** function).

Why regularize?

- Improve generalization by avoiding over-fitting.
- The total loss may not be coercive (*e.g.*, logistic loss on separable data), thus having no minima.
- Express prior knowledge about $\mathbf{w}$.

Regularization

Regularized parameter estimate:

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

where $\Omega(\mathbf{w}) \geq 0$ and $\lim_{\|\mathbf{w}\| \to \infty} \Omega(\mathbf{w}) = \infty$ (**coercive** function).

Why regularize?

- Improve generalization by avoiding over-fitting.
- The total loss may not be coercive (*e.g.*, logistic loss on separable data), thus having no minima.
- Express prior knowledge about $\mathbf{w}$.
- Select relevant features (via sparsity-inducing regularization).

Regularization

Regularized parameter estimate:

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

where $\Omega(\mathbf{w}) \geq 0$ and $\lim_{\|\mathbf{w}\| \to \infty} \Omega(\mathbf{w}) = \infty$ (**coercive** function).

Why regularize?

- Improve generalization by avoiding over-fitting.
- The total loss may not be coercive (*e.g.*, logistic loss on separable data), thus having no minima.
- Express prior knowledge about $\mathbf{w}$.
- Select relevant features (via sparsity-inducing regularization).

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

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

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 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: $L_{\text{LR}}(\mathbf{w}; x_n, y_n) = -\log P\left(y_n | x_n; \mathbf{w}\right)$.

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: $L_{\text{SE}}(\mathbf{w}; x_n, y_n) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2 = -\log \mathcal{N}(y | \mathbf{w}^\top \mathbf{f}(x), 1)$

Norms: A Quick Review

Before focusing on regularizers, a quick review about norms.

Norms: A Quick Review

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \to \mathbb{R}$ is a norm if if satisfies:

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

Norms: A Quick Review

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \to \mathbb{R}$ is a norm if if satisfies:

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

Norms: A Quick Review

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \to \mathbb{R}$ is a norm if if satisfies:

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

Norms: A Quick Review

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \to \mathbb{R}$ is a norm if if satisfies:

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

Examples of norms:

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

Norms: A Quick Review

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \to \mathbb{R}$ is a norm if if satisfies:

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

Examples of norms:

- $\|\mathbf{w}\|_p = \left(\sum_i (w_i)^p \right)^{1/p}$ (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

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \to \mathbb{R}$ is a norm if if satisfies:

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

Examples of norms:

- $\|\mathbf{w}\|_p = \left(\sum_i (w_i)^p \right)^{1/p}$ (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

Before focusing on regularizers, a quick review about norms.

Some function $p : \mathbb{R} \rightarrow \mathbb{R}$ is a norm if if satisfies:

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

Examples of norms:

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

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

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 $p(\mathbf{w}) \propto \exp\left(-\lambda\|\mathbf{w}\|_2^2\right)$

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 $p(\mathbf{w}) \propto \exp\left(-\lambda\|\mathbf{w}\|_2^2\right)$
- **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 $p(\mathbf{w}) \propto \exp\left(-\lambda\|\mathbf{w}\|_2^2\right)$
- **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 $p(\mathbf{w}) \propto \exp\left(-\lambda\|\mathbf{w}\|_2^2\right)$
- **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 $p(\mathbf{w}) \propto \exp\left(-\lambda\|\mathbf{w}\|_2^2\right)$
- **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 $p(\mathbf{w}) \propto \exp\left(-\lambda\|\mathbf{w}\|_2^2\right)$
- **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: 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 $p(w_i) \propto \exp\left(-\lambda |w_i|\right)$

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),...

C… so

Regularized parame… $x_n, y_n) + \Omega(\mathbf{w})$

The new classic is … $|w_i|$.

- Corresponds t… $\exp\left(-\lambda |w_i|\right)$
- Best known as… **lection operator (Lasso)** (Tibs…
- Used earlier in… uir, 1973; Taylor et al., 1979) r…

- Geology/geophysics
  - Claerbout and Muir (1973)
  - Taylor et al. (1979)
  - Levy and Fullager (1981)
  - Oldenburg et al. (1983)
  - Santosa and Symes (1988)
- Radio astronomy
  - Högbom (1974)
  - Schwarz (1978)
- Fourier transform spectroscopy
  - Kawata et al. (1983)
  - Mammone (1983)
  - Minami et al. (1985)
- NMR spectroscopy
  - Barkhuijsen (1985)
  - Newman (1988)
- Medical ultrasound
  - Papoulis and Chamzas (1979)

from (Goyal et al, 2010)

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).

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 $p(w_i) \propto \exp\left(-\lambda|w_i|\right)$
- 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.

The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest case:

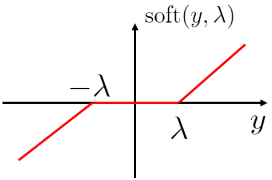
$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w - y)^2 + \lambda|w| = \text{soft}(y, \lambda) = \begin{cases} y - \lambda & \Leftarrow \quad y > \lambda \\ 0 & \Leftarrow \quad |y| \leq \lambda \\ y + \lambda & \Leftarrow \quad y < -\lambda \end{cases}$$

The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest case:

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

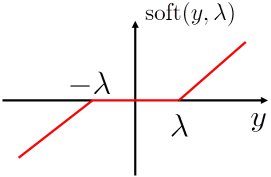


The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest case:

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



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

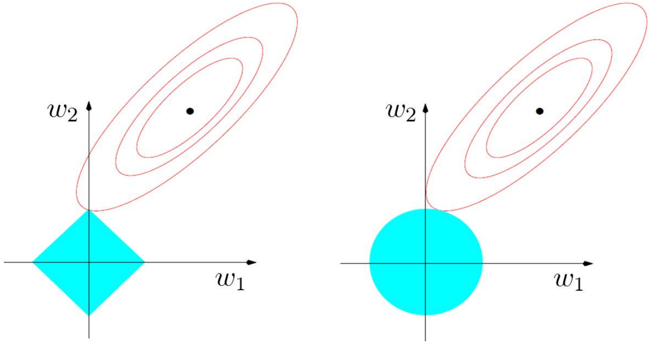
$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \frac{\lambda}{2} w^2 = \frac{1}{1+\lambda} y$$

The Lasso and Sparsity (II)

Why does the Lasso yield sparsity?

The Lasso and Sparsity (II)

Why does the Lasso yield sparsity?



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

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

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...

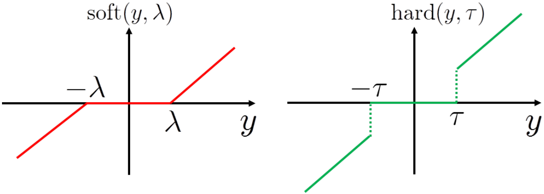
$$\widehat{w} = \arg\min_w \frac{1}{2}(w-y)^2 + \lambda|w|_0 = \mathrm{hard}(y, \sqrt{2\lambda}) = \begin{cases} y & \Leftarrow \quad |y| > \sqrt{2\lambda} \\ 0 & \Leftarrow \quad |y| \leq \sqrt{2\lambda} \end{cases}$$

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...

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

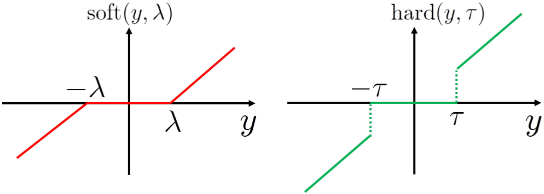


Relationship Between ℓ_1 and ℓ_0

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

Not convex, but...

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



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

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

subject to $\|\mathbf{w}\|_0 \leq \tau$ (limit the number of features)

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).

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

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

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

Relationship Between ℓ_1 and ℓ_0 (II)

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

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

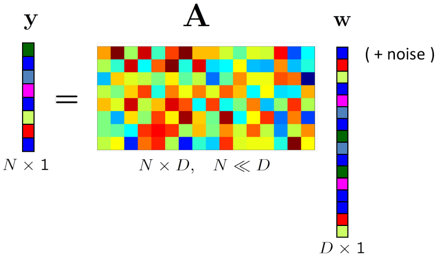
A closely related problem, also NP-hard (Muthukrishnan, 2005).

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

In some cases, one may replace ℓ_0 with ℓ_1 and obtain "similar" results:

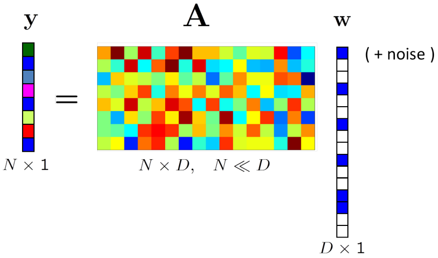
central issue in compressive sensing (CS) (Candès et al., 2006a; Donoho, 2006).

Compressive Sensing in One Slide



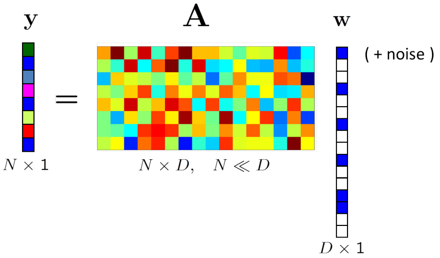
Even in the noiseless case, it seems impossible to recover **w** from **y**

Compressive Sensing in One Slide



Even in the noiseless case, it seems impossible to recover **w** from **y**
...unless, **w** is **sparse** and **A** has some properties.

Compressive Sensing in One Slide



Even in the noiseless case, it seems impossible to recover **w** from **y**
...unless, **w** is **sparse** and **A** has some properties.

If **w** is sparse enough and **A** has certain properties, then **w** is stably recovered via (Haupt and Nowak, 2006)

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{w}\|_0 \quad \text{subject to } \|\mathbf{A}\mathbf{w} - \mathbf{y}\| \leq \delta$$

NP-hard!

...OK, in Two Slides

Under some conditions on $\mathbf{A}$ (e.g., the restricted isometry property (RIP)), ℓ_0 can be replaced with ℓ_1 (Candès et al., 2006b):

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{w}\|_1$$
$$\text{subject to } \|\mathbf{A}\mathbf{w} - \mathbf{y}\| \leq \delta$$

convex problem

...OK, in Two Slides

Under some conditions on $\mathbf{A}$ (e.g., the restricted isometry property (RIP)), ℓ_0 can be replaced with ℓ_1 (Candès et al., 2006b):

$$\begin{aligned} \widehat{\mathbf{w}} \;=\; & \arg\min_{\mathbf{w}} \|\mathbf{w}\|_1 \\ & \text{subject to } \|\mathbf{A}\mathbf{w} - \mathbf{y}\| \leq \delta \end{aligned} \qquad \textbf{convex problem}$$

Matrix $\mathbf{A}$ satisfies the RIP of order k, with constant $\delta_k \in (0, 1)$, if

$$\|\mathbf{w}\|_0 \leq k \;\Rightarrow\; (1 - \delta_k)\|\mathbf{w}\|_2^2 \leq \|\mathbf{A}\mathbf{w}\| \leq (1 + \delta_k)\|\mathbf{w}\|_2^2$$

...i.e., for k-sparse vectors, $\mathbf{A}$ is approximately an isometry.

...OK, in Two Slides

Under some conditions on $\mathbf{A}$ (e.g., the restricted isometry property (RIP)), ℓ_0 can be replaced with ℓ_1 (Candès et al., 2006b):

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \|\mathbf{w}\|_1 \\ &\quad \text{subject to } \|\mathbf{A}\mathbf{w} - \mathbf{y}\| \leq \delta \end{aligned}$$

convex problem

Matrix $\mathbf{A}$ satisfies the RIP of order k, with constant $\delta_k \in (0, 1)$, if

$$\|\mathbf{w}\|_0 \leq k \;\Rightarrow\; (1 - \delta_k)\|\mathbf{w}\|_2^2 \leq \|\mathbf{A}\mathbf{w}\| \leq (1 + \delta_k)\|\mathbf{w}\|_2^2$$

...i.e., for k-sparse vectors, $\mathbf{A}$ is approximately an isometry.

Other properties (spark and null space property (NSP)) can be used; checking RIP, NSP, spark is **NP-hard** (Tillmann and Pfetsch, 2012).

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

- Nearly all CS results assume linear observations $\mathbf{y} = \mathbf{A}\mathbf{w} + \text{noise}$; recent exceptions: Blumensath (2012); Plan and Vershynin (2012).

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

- Nearly all CS results assume linear observations $\mathbf{y} = \mathbf{A}\mathbf{w} + \text{noise}$; recent exceptions: Blumensath (2012); Plan and Vershynin (2012).
- "Good" matrices with RIP or NSP are randomly constructed.

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

- Nearly all CS results assume linear observations $\mathbf{y} = \mathbf{A}\mathbf{w} + \text{noise}$; recent exceptions: Blumensath (2012); Plan and Vershynin (2012).
- "Good" matrices with RIP or NSP are randomly constructed.
- What is missing: results for (multinomial) logistic loss, not based on RIP or NSP.

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

- Nearly all CS results assume linear observations $\mathbf{y} = \mathbf{A}\mathbf{w} + \text{noise}$; recent exceptions: Blumensath (2012); Plan and Vershynin (2012).
- "Good" matrices with RIP or NSP are randomly constructed.
- What is missing: results for (multinomial) logistic loss, not based on RIP or NSP.

Other types of results for ℓ_1-regularized logistic regression:

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

- Nearly all CS results assume linear observations $\mathbf{y} = \mathbf{A}\mathbf{w} + \text{noise}$; recent exceptions: Blumensath (2012); Plan and Vershynin (2012).
- "Good" matrices with RIP or NSP are randomly constructed.
- What is missing: results for (multinomial) logistic loss, not based on RIP or NSP.

Other types of results for ℓ_1-regularized logistic regression:

- PAC-Bayesian bounds (generalization improves with sparsity): Krishnapuram et al. (2005)

Relevant Theory?

Are sparsity-related compressed sensing **(CS)** results relevant for NLP?

- Nearly all CS results assume linear observations $\mathbf{y} = \mathbf{A}\mathbf{w} + \text{noise}$; recent exceptions: Blumensath (2012); Plan and Vershynin (2012).
- "Good" matrices with RIP or NSP are randomly constructed.
- What is missing: results for (multinomial) logistic loss, not based on RIP or NSP.

Other types of results for ℓ_1-regularized logistic regression:

- PAC-Bayesian bounds (generalization improves with sparsity): Krishnapuram et al. (2005)
- Oracle (van de Geer, 2008) and consistency (Negahban et al., 2012) results.

Take-Home Messages

- Sparsity is desirable for interpretability, computational savings, and generalization
- ℓ_1-regularization gives an embedded method for feature selection
- Another view of ℓ_1: a convex surrogate for direct penalization of cardinality (ℓ_0)
- Under some conditions, ℓ_1 guarantees exact support recovery
- However: the currently known sufficient conditions are too strong and not met in typical NLP problems
- Yet: a lot of theory is still missing
- 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*

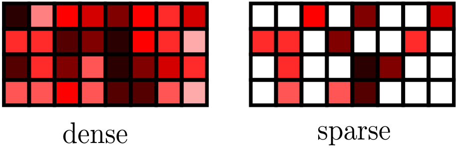
Leads to statistical gains if the prior assumptions are correct (Stojnic et al., 2009)

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)

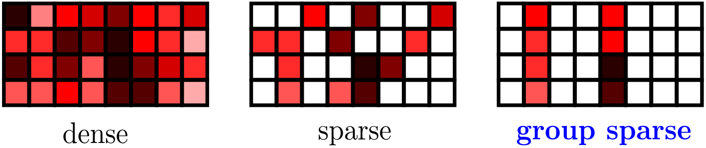
“Grid” Sparsity

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



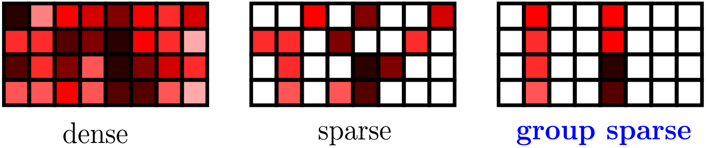
"Grid" Sparsity

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



"Grid" Sparsity

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



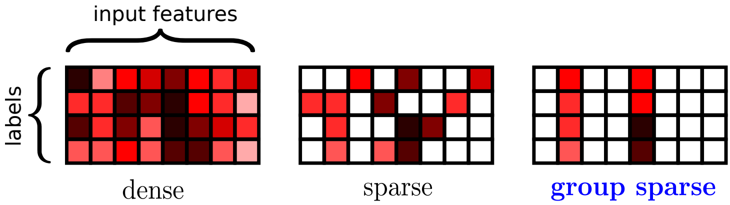
Goal: push *entire columns* to have zero weights

The groups are the columns of the grid

Example 1: Sparsity with Multiple Classes

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

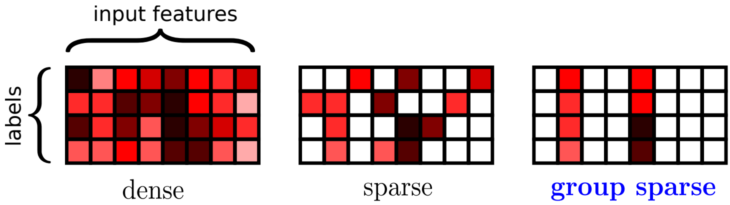
In words: we're conjoining each input feature with each output class



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



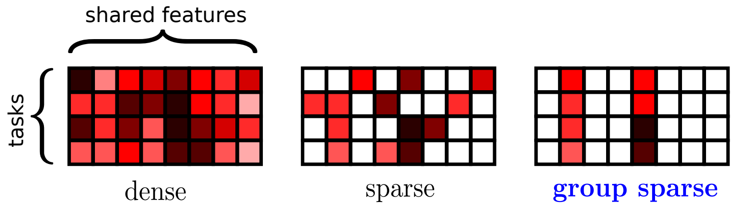
"Standard" sparsity is wasteful—we still need to hash all the input features

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

Solution: one group per *input* feature

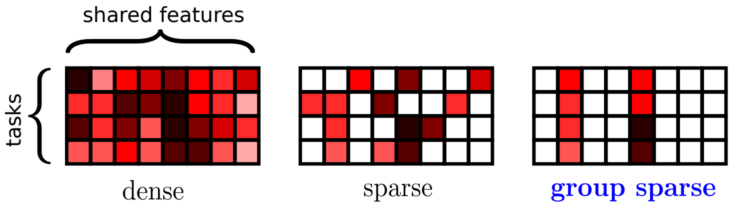
Example 2: Multi-Task Learning (Caruana, 1997; Obozinski et al., 2010)

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**

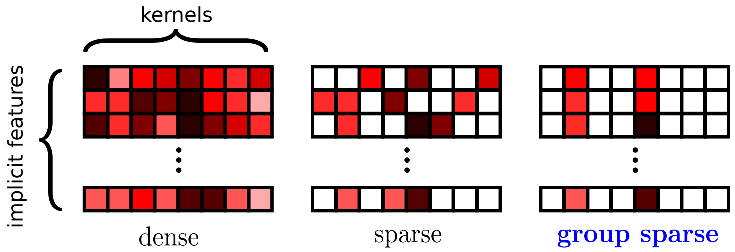


What we want: discard features that are irrelevant for *all* tasks

Solution: one group per feature

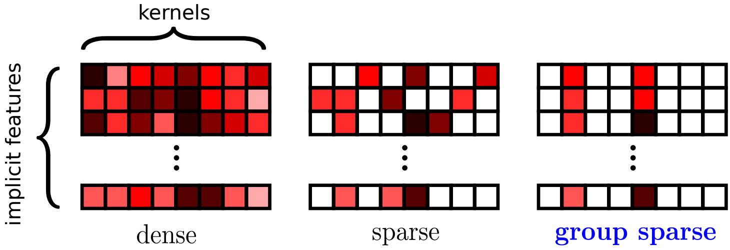
Example 3: Multiple Kernel Learning (Lanckriet et al., 2004)

Same thing, except now columns are **kernel functions** $\{K_m\}_{m=1}^{M}$



Example 3: Multiple Kernel Learning (Lanckriet et al., 2004)

Same thing, except now columns are **kernel functions** $\{K_m\}_{m=1}^{M}$

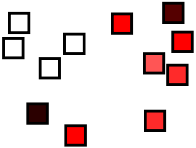


Goal: a new kernel which is a sparse combination of the given kernels

$$K((x,y),(x',y')) = \sum_{m=1}^{M} \alpha_m K_m((x,y),(x',y')), \qquad \boldsymbol{\alpha} \text{ is sparse}$$

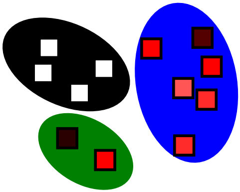
Solution: make each group be a *kernel* K_j

Group Sparsity



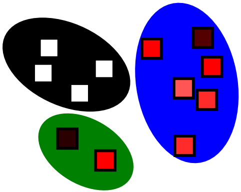
- D features

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

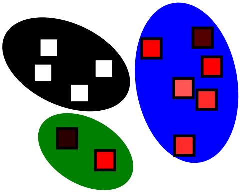


- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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

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

Group Sparsity



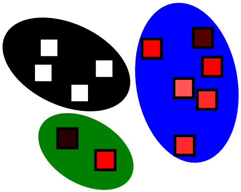
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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

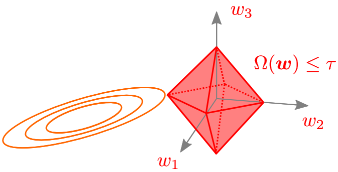
$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n) \quad \text{subject to } \Omega(\mathbf{w}) \leq \tau$$

- Morozov regularization

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

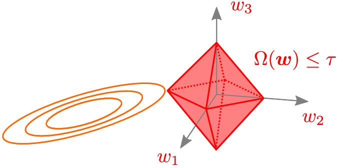
Equivalent, under mild conditions (namely convexity).

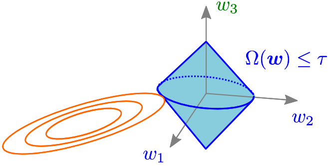
Lasso versus group-Lasso



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

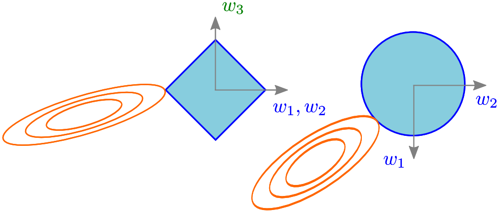
Lasso versus group-Lasso





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

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



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

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

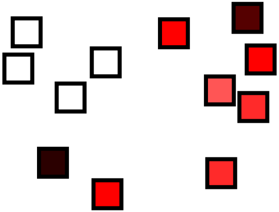
Trivial choices of groups recover *unstructured* regularizers:

- 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:

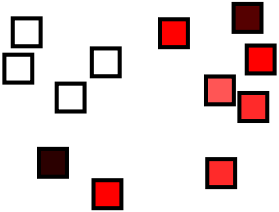
- label-based groups (groups are columns of a matrix)
- template-based groups (next)

Example: Feature Template Selection



Example: Feature Template Selection

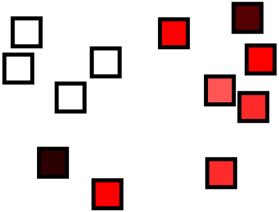
Input:	We	want	to	explore	the	feature	space
	PRP	VBP	TO	VB	DT	NN	NN
Output:	B-NP	B-VP	I-VP	I-VP	B-NP	I-NP	I-NP



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

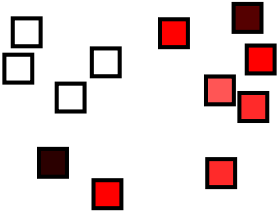
- Goal: Select relevant **feature templates**



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

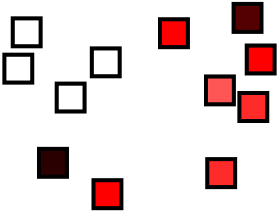
- Goal: Select relevant **feature templates**



Example: Feature Template Selection

				$\triangledown$			
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

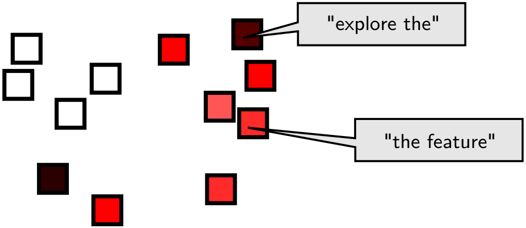
- Goal: Select relevant **feature templates**



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

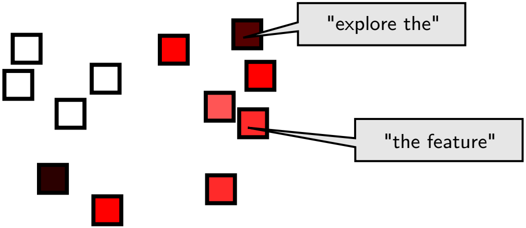
- Goal: Select relevant **feature templates**



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

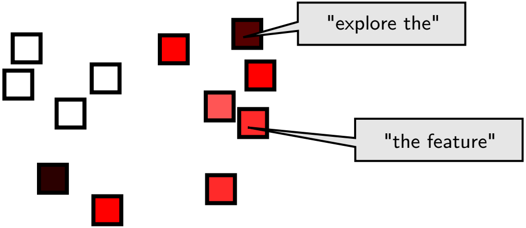
- Goal: Select relevant **feature templates**



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

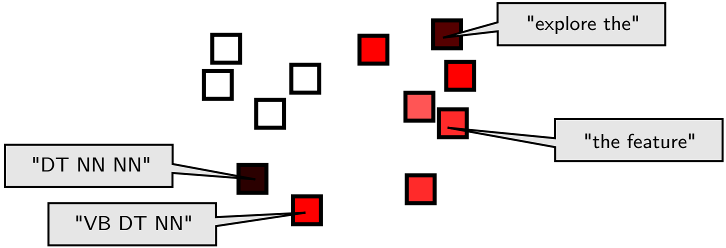
- Goal: Select relevant **feature templates**



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**

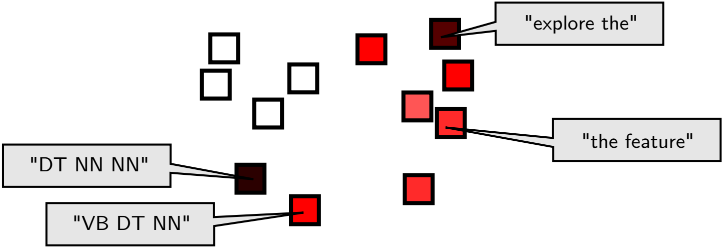


Example: Feature Template Selection

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

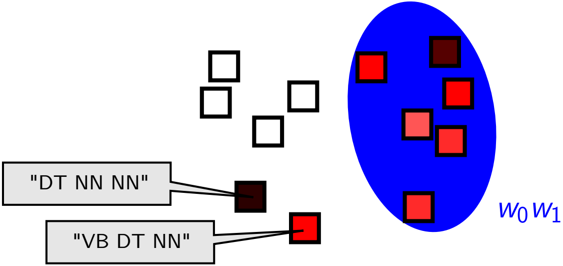


Example: Feature Template Selection

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**

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

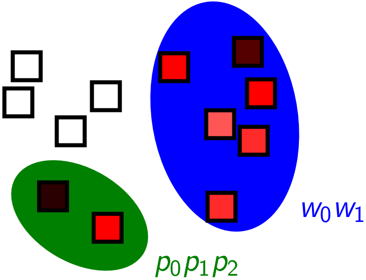


Example: Feature Template Selection

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**

  ⇒ Make each group correspond to a feature template

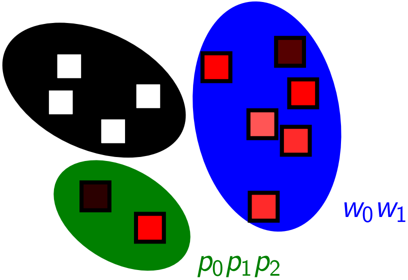


Example: Feature Template Selection

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

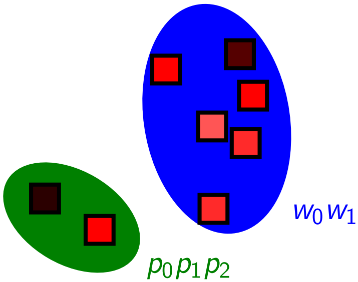


Example: Feature Template Selection

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**

  ⇒ 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

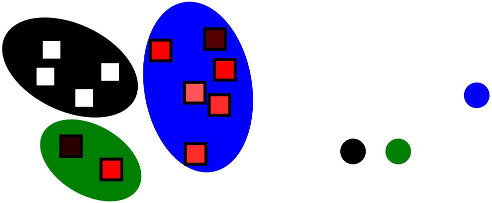
Assumption: if two groups overlap, one is contained in the other

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

Tree-Structured Groups

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

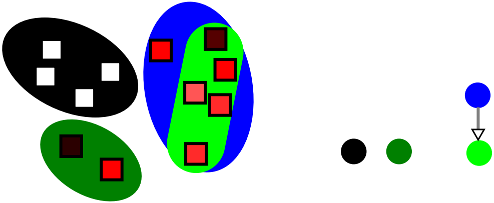
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

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

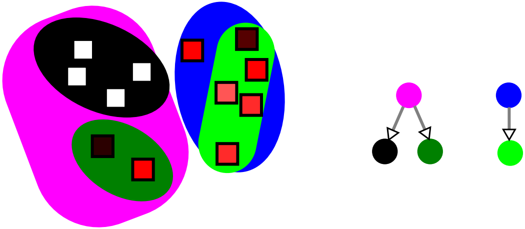
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

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

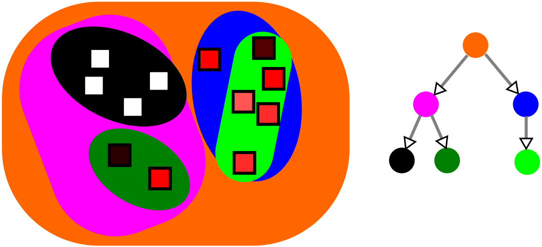
⇒ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

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

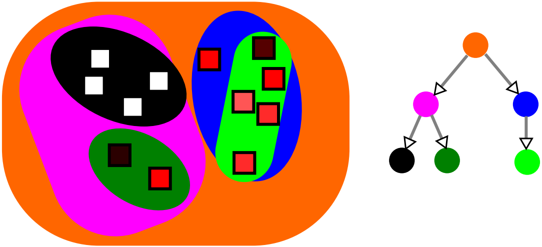
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



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)

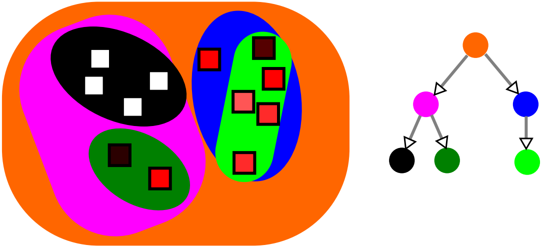


- What is the **sparsity pattern**?

Tree-Structured Groups

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

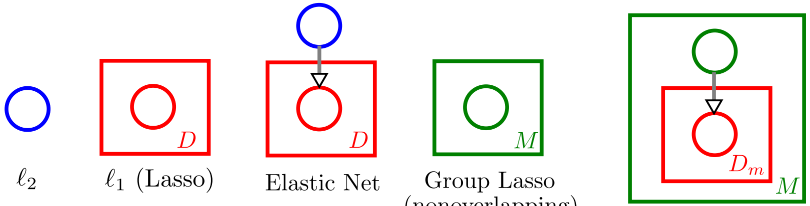
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



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

Plate Notation

Typically used for graphical models, but also works here for representing the Hasse diagram of tree-structured groups



ℓ_2 ℓ_1 (Lasso) Elastic Net Group Lasso (nonoverlapping) Sparse Group Lasso

Three Scenarios

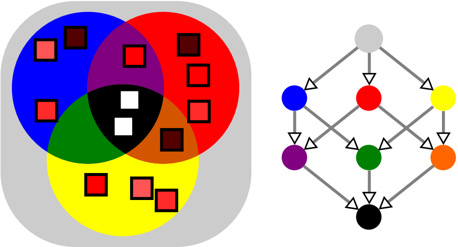
- Non-overlapping Groups
- Tree-structured Groups
- Graph-structured Groups

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**

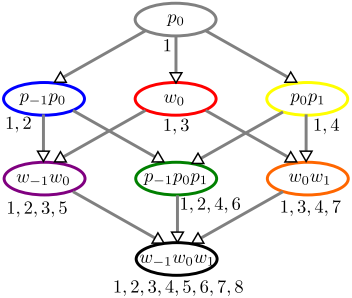


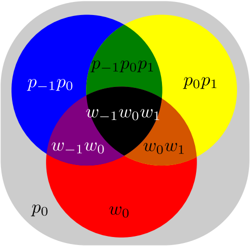
- set inclusion induces a **partial order** on groups (Jenatton et al., 2009)
- feature space becomes a **poset**
- **sparsity patterns**: given by this poset

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

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

Learning the Model

Recall that learning involves solving

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

We'll address two kinds of optimization algorithms:

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

Learning the Model

Recall that learning involves solving

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

We'll address two kinds of optimization algorithms:

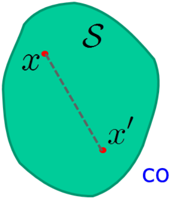
- batch algorithms (attacks the complete problem);
- online algorithms (uses the training examples one by one)

Before that: we'll review some key concepts of convex analysis

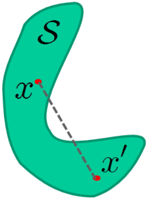
Outline

Key Concepts in Convex Analysis: Convex Sets

$\mathcal{S}$ is convex if $x, x' \in \mathcal{S} \Rightarrow \forall \lambda \in [0,1] \;\; \lambda x + (1-\lambda) x' \in \mathcal{S}$

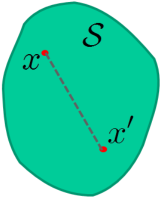


convex

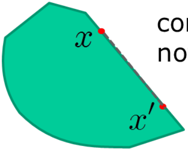


non-convex

$\mathcal{S}$ is strictly convex if $x, x' \in \mathcal{S} \Rightarrow \forall \lambda \in (0,1) \;\; \lambda x + (1-\lambda) x' \in \mathrm{int}(\mathcal{S})$



strictly convex



convex, but not strictly

Key Concepts in Convex Analysis: Convex Functions

Extended real valued function: $f : \mathbb{R}^N \to \bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$

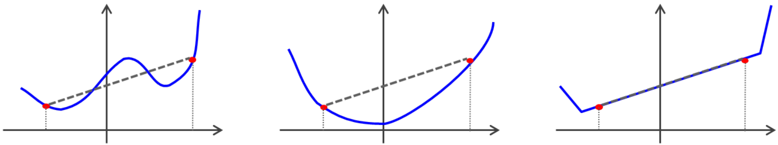
Domain of a function: $\mathrm{dom}(f) = \{x : f(x) \neq +\infty\}$

f is a convex function if

$$\forall \lambda \in [0,1], x, x' \in \mathrm{dom}(f) \;\; f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$

f is a strictly convex function if

$$\forall \lambda \in (0,1), x, x' \in \mathrm{dom}(f) \;\; f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda) f(x')$$



non-convex

convex
strictly convex

convex, not strictly

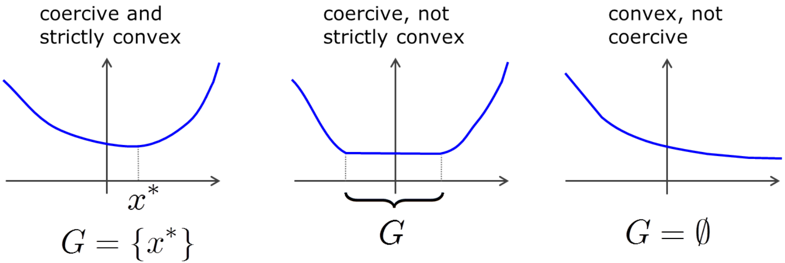
Key Concepts in Convex Analysis: Minimizers

$$f : \mathbb{R}^N \rightarrow \bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$$

f is coercive if $\lim_{\|x\| \rightarrow +\infty} f(x) = +\infty$

if f is coercive, then $G \equiv \arg\min_x f(x)$ is a non-empty set

if f is strictly convex, then G has at most one element



Key Concepts in Convex Analysis: Subgradients

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

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

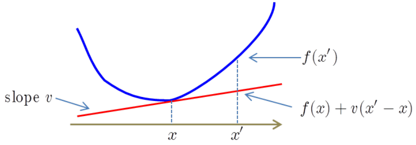
Key Concepts in Convex Analysis: Subgradients

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 in Convex Analysis: Subgradients

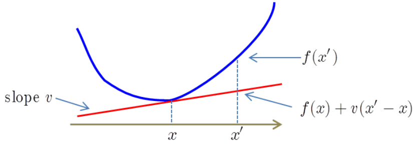
Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

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 in Convex Analysis: Subgradients

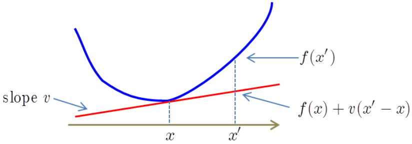
Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

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

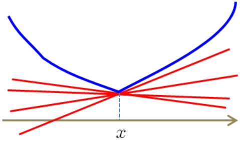
$$\boxed{\mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x} \text{ if } \; f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})}$$

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 in Convex Analysis: Subgradients

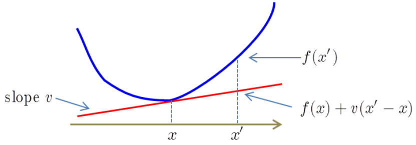
Convexity $\Rightarrow$ continuity; convexity $\not\Rightarrow$ differentiability (e.g., $f(\mathbf{w}) = \|w\|_1$).

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

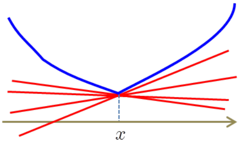
$$\boxed{\mathbf{v} \text{ is a subgradient of } f \text{ at } \mathbf{x} \text{ if } f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})}$$

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



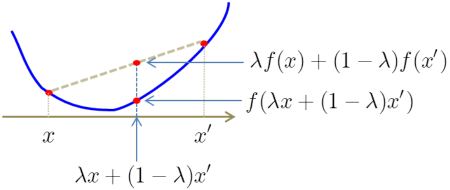
non-differentiable case

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

Key Concepts in Convex Analysis: Strong Convexity

Recall the definition of convex function: $\forall \lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda)f(x')$$



convexity

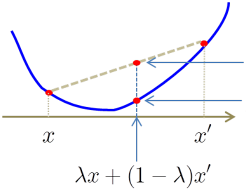
Key Concepts in Convex Analysis: Strong Convexity

Recall the definition of convex function: $\forall \lambda \in [0,1]$,

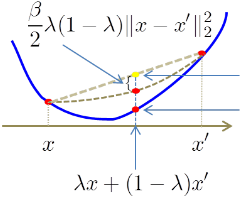
$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$

A $\beta-$strongly convex function satisfies a stronger condition: $\forall \lambda \in [0,1]$

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x') - \frac{\beta}{2}\lambda(1-\lambda)\|x - x'\|_2^2$$



convexity



strong convexity

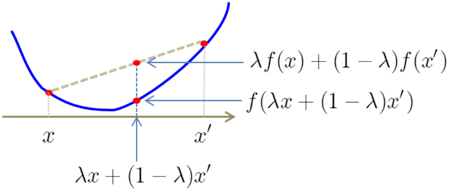
Key Concepts in Convex Analysis: Strong Convexity

Recall the definition of convex function: $\forall \lambda \in [0,1]$,

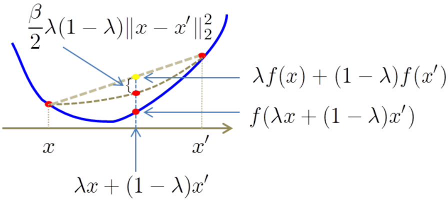
$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$

A $\beta-$strongly convex function satisfies a stronger condition: $\forall \lambda \in [0,1]$

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x') - \frac{\beta}{2}\lambda(1-\lambda)\|x-x'\|_2^2$$



convexity



strong convexity

Strong convexity $\underset{\not\Leftarrow}{\Rightarrow}$ strict convexity.

Proximity Operators

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

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.

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**

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

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.

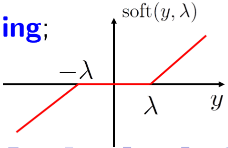
Classical examples:

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

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

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

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



Proximity Operators (II)

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

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**

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

Proximity Operators (II)

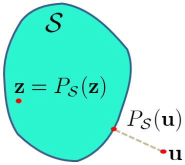
$$\boxed{prox_{\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|_2^2 + \Omega(\mathbf{u})}$$

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

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

- 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

Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_j(\mathbf{w}_{G_m})$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_{G_m}$ is a sub-vector of $\mathbf{w}$.

Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_j(\mathbf{w}_{G_m})$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_{G_m}$ is a sub-vector of $\mathbf{w}$.

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

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

Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_j(\mathbf{w}_{G_m})$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_{G_m}$ is a sub-vector of $\mathbf{w}$.

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

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

- **Tree-structured groups:** (two groups are either non-overlapping or one contais the other) prox_Ω can be computed recursively (Jenatton et al., 2011).

Proximity Operators (III)

Group regularizers: $\Omega(\mathbf{w}) = \sum_{m=1}^{M} \Omega_j(\mathbf{w}_{G_m})$

Groups: $G_m \subset \{1, 2, ..., D\}$. $\mathbf{w}_{G_m}$ is a sub-vector of $\mathbf{w}$.

- **Non-overlapping groups** ($G_m \cap G_n = \emptyset$, for $m \neq n$): separable prox operator
$$[\mathrm{prox}_\Omega(\mathbf{w})]_{G_m} = \mathrm{prox}_{\Omega_m}(\mathbf{w}_{G_m})$$
- **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 $\Omega_j(\mathbf{w}_{G_m}) = \|\mathbf{w}_{G_m}\|_2$: solved via convex smooth optimization (Yuan et al., 2011).
  - Sequential proximity steps (Martins et al., 2011a) (more later).

Outline

Subgradient Methods

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

Subgradient Methods

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

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

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

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 ℓ_1 regularizers.

(Block-)Coordinate Descent

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

(Block-)Coordinate Descent

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

Update one (block of) component(s) of $\mathbf{w}$ at a time:

$$w_i^{\text{new}} \leftarrow \arg\min_{w_i} \Omega([w_1, ..., w_i, ...w_D]) + \Lambda([w_1, ..., w_i, ...w_D])$$

(Genkin et al., 2007; Krishnapuram et al., 2005; Liu et al., 2009; Shevade and Keerthi, 2003; Tseng and Yun, 2009; Yun and Toh, 2011)

(Block-)Coordinate Descent

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

Update one (block of) component(s) of **w** at a time:

$$w_i^{\text{new}} \leftarrow \arg\min_{w_i} \Omega([w_1, ..., w_i, ...w_D]) + \Lambda([w_1, ..., w_i, ...w_D])$$

(Genkin et al., 2007; Krishnapuram et al., 2005; Liu et al., 2009; Shevade and Keerthi, 2003; Tseng and Yun, 2009; Yun and Toh, 2011)

Squared error loss: closed-form solution. Other losses (*e.g.*, logistic):

- solve numerically, *e.g.*, Newton steps (Shevade and Keerthi, 2003).
- use local quadratic approximation/bound (Krishnapuram et al., 2005; Tseng and Yun, 2009; Yun and Toh, 2011).

(Block-)Coordinate Descent

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

Update one (block of) component(s) of **w** at a time:

$$w_i^{\text{new}} \leftarrow \arg\min_{w_i} \Omega([w_1, ..., w_i, ...w_D]) + \Lambda([w_1, ..., w_i, ...w_D])$$

(Genkin et al., 2007; Krishnapuram et al., 2005; Liu et al., 2009; Shevade and Keerthi, 2003; Tseng and Yun, 2009; Yun and Toh, 2011)

Squared error loss: closed-form solution. Other losses (*e.g.*, logistic):

- solve numerically, *e.g.*, Newton steps (Shevade and Keerthi, 2003).
- use local quadratic approximation/bound (Krishnapuram et al., 2005; Tseng and Yun, 2009; Yun and Toh, 2011).

Shown to converge; competitive with state-of-the-art (Yun and Toh, 2011).

Has been used in NLP: Sokolovska et al. (2010); Lavergne et al. (2010).

Projected Gradient

Instead of $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$, tackle $\boxed{\min_{\mathbf{w}} \Lambda(\mathbf{w}) \; \text{ subject to } \Omega(\mathbf{w}) \leq \tau}$.

Projected Gradient

Instead of $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$, tackle $\boxed{\min_{\mathbf{w}} \Lambda(\mathbf{w}) \;\text{ subject to } \Omega(\mathbf{w}) \leq \tau}$.

Building blocks:

- loss gradient $\tilde{\nabla}\Lambda(\mathbf{w})$
- Euclidean projection $P_{\mathcal{S}}(\mathbf{w})$, where $\mathcal{S} = \{\mathbf{w} : \Omega(\mathbf{w}) \leq \tau\}$

Projected Gradient

Instead of $\boxed{\min_{\mathbf{w}} \; \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$, tackle $\boxed{\min_{\mathbf{w}} \Lambda(\mathbf{w}) \;\; \text{subject to } \Omega(\mathbf{w}) \leq \tau}$.

Building blocks:

- loss gradient $\tilde{\nabla}\Lambda(\mathbf{w})$
- Euclidean projection $P_{\mathcal{S}}(\mathbf{w})$, where $\mathcal{S} = \{\mathbf{w} : \Omega(\mathbf{w}) \leq \tau\}$

$$\boxed{\mathbf{w} \leftarrow P_{\mathcal{S}}(\mathbf{w} - \eta\tilde{\nabla}\Lambda(\mathbf{w}))}$$

...maybe using line search to adjust the step length.

Projected Gradient

Instead of $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$, tackle $\boxed{\min_{\mathbf{w}} \Lambda(\mathbf{w}) \ \text{ subject to } \Omega(\mathbf{w}) \leq \tau}$.

Building blocks:

- loss gradient $\tilde{\nabla}\Lambda(\mathbf{w})$
- Euclidean projection $P_{\mathcal{S}}(\mathbf{w})$, where $\mathcal{S} = \{\mathbf{w} : \Omega(\mathbf{w}) \leq \tau\}$

$$\boxed{\mathbf{w} \leftarrow P_{\mathcal{S}}(\mathbf{w} - \eta\tilde{\nabla}\Lambda(\mathbf{w}))}$$

...maybe using line search to adjust the step length.

Example: for $\mathcal{S} = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq \tau\}$, projection $P_{\mathcal{S}}(\mathbf{w})$ has $O(D \log D)$ cost (Duchi et al., 2008).

Viable and competitive alternative, which has been used in machine learning and NLP (Duchi et al., 2008; Quattoni et al., 2009).

Projected Gradient

Instead of $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$, tackle $\boxed{\min_{\mathbf{w}} \Lambda(\mathbf{w}) \ \text{ subject to } \Omega(\mathbf{w}) \leq \tau}$.

Building blocks:

- loss gradient $\tilde{\nabla}\Lambda(\mathbf{w})$
- Euclidean projection $P_{\mathcal{S}}(\mathbf{w})$, where $\mathcal{S} = \{\mathbf{w} : \Omega(\mathbf{w}) \leq \tau\}$

$$\boxed{\mathbf{w} \leftarrow P_{\mathcal{S}}(\mathbf{w} - \eta\tilde{\nabla}\Lambda(\mathbf{w}))}$$

...maybe using line search to adjust the step length.

Example: for $\mathcal{S} = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq \tau\}$, projection $P_{\mathcal{S}}(\mathbf{w})$ has $O(D \log D)$ cost (Duchi et al., 2008).

Viable and competitive alternative, which has been used in machine learning and NLP (Duchi et al., 2008; Quattoni et al., 2009).

Shown later: **projected gradient** is a particular instance of the more general **proximal gradient** methods.

From Gradient to Hessian: Newton's Method

Assume $F(\mathbf{w}) = \Omega(\mathbf{w}) + \Lambda(\mathbf{w})$ is twice-differentiable.

Second order (quadratic) Taylor expansion around $\mathbf{w}'$:

$$F(\mathbf{w}) \approx F(\mathbf{w}') + \underbrace{\nabla F(\mathbf{w}')}_{\textbf{Gradient}}{}^{\top}(\mathbf{w} - \mathbf{w}') + \frac{1}{2}(\mathbf{w} - \mathbf{w}')^{\top} \underbrace{\mathbf{H}(\mathbf{w}')}_{\textbf{Hessian:}}(\mathbf{w} - \mathbf{w}')$$

From Gradient to Hessian: Newton's Method

Assume $F(\mathbf{w}) = \Omega(\mathbf{w}) + \Lambda(\mathbf{w})$ is twice-differentiable.

Second order (quadratic) Taylor expansion around $\mathbf{w}'$:

$$F(\mathbf{w}) \approx F(\mathbf{w}') + \underbrace{\nabla F(\mathbf{w}')}_{\textbf{Gradient}}^\top (\mathbf{w} - \mathbf{w}') + \frac{1}{2}(\mathbf{w} - \mathbf{w}')^\top \underbrace{\mathbf{H}(\mathbf{w}')}_{\textbf{Hessian:}}(\mathbf{w} - \mathbf{w}')$$

Use the direction that minimizes this quadratic approximation:

$$\mathbf{w} \;\leftarrow\; \mathbf{w} - \alpha(\mathbf{H}(\mathbf{w}))^{-1}\nabla F(\mathbf{w})$$

with stepsize α usually determined by line search.

From Gradient to Hessian: Newton's Method

Assume $F(\mathbf{w}) = \Omega(\mathbf{w}) + \Lambda(\mathbf{w})$ is twice-differentiable.

Second order (quadratic) Taylor expansion around $\mathbf{w}'$:

$$F(\mathbf{w}) \approx F(\mathbf{w}') + \underbrace{\nabla F(\mathbf{w}')}_{\text{Gradient}}{}^\top (\mathbf{w} - \mathbf{w}') + \frac{1}{2}(\mathbf{w} - \mathbf{w}')^\top \underbrace{\mathbf{H}(\mathbf{w}')}_{\text{Hessian:}}(\mathbf{w} - \mathbf{w}')$$

Use the direction that minimizes this quadratic approximation:

$$\mathbf{w} \;\leftarrow\; \mathbf{w} - \alpha(\mathbf{H}(\mathbf{w}))^{-1}\nabla F(\mathbf{w})$$

with stepsize α usually determined by line search.

Drawback: may be costly (or impossible) to compute and invert the Hessian! $O(D^3)$ for a naïve approach.

From Gradient to Hessian: Newton's Method

Assume $F(\mathbf{w}) = \Omega(\mathbf{w}) + \Lambda(\mathbf{w})$ is twice-differentiable.

Second order (quadratic) Taylor expansion around $\mathbf{w}'$:

$$F(\mathbf{w}) \approx F(\mathbf{w}') + \underbrace{\nabla F(\mathbf{w}')}_{\textbf{Gradient}}^{\top}(\mathbf{w} - \mathbf{w}') + \frac{1}{2}(\mathbf{w} - \mathbf{w}')^{\top} \underbrace{\mathbf{H}(\mathbf{w}')}_{\textbf{Hessian:}}(\mathbf{w} - \mathbf{w}')$$

Use the direction that minimizes this quadratic approximation:

$$\mathbf{w} \;\leftarrow\; \mathbf{w} - \alpha(\mathbf{H}(\mathbf{w}))^{-1}\nabla F(\mathbf{w})$$

with stepsize α usually determined by line search.

Drawback: may be costly (or impossible) to compute and invert the Hessian! $O(D^3)$ for a naïve approach.

Quasi-Newton methods, namely **L-BFGS**, *approximate* the inverse Hessian directly from past gradient information.

Orthant-Wise Limited-memory Quasi Newton

OWL-QN: clever adaptation of L-BFGS to ℓ_1-regularization (Andrew and Gao, 2007; Gao et al., 2007)

Orthant-Wise Limited-memory Quasi Newton

OWL-QN: clever adaptation of L-BFGS to ℓ_1-regularization (Andrew and Gao, 2007; Gao et al., 2007)

input: stepsize sequence $(\eta_t)_{t=1}^T$
initialize $\mathbf{w} = 0$
for $t = 1, 2, \ldots$ **do**
  compute a particular subgradient $\mathbf{g}_t := \tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla}\Lambda(\mathbf{w})$
  compute **inverse Hessian approximation** $\mathbf{S}_t$ *"a la L-BFGS"*
  compute descent direction $\mathbf{d}_t = -(\mathbf{S}_t)\,\mathbf{g}_t$
  do line search for α, and update $\mathbf{w} \leftarrow \mathbf{w} + \alpha\mathbf{d}_t$
  clip w if necessary to stay in the same orthant
end for

Orthant-Wise Limited-memory Quasi Newton

OWL-QN: clever adaptation of L-BFGS to ℓ_1-regularization (Andrew and Gao, 2007; Gao et al., 2007)

input: stepsize sequence $(\eta_t)_{t=1}^T$
initialize $\mathbf{w} = 0$
for $t = 1, 2, \ldots$ **do**
  compute a particular subgradient $\mathbf{g}_t := \tilde{\nabla}\Omega(\mathbf{w}) + \tilde{\nabla}\Lambda(\mathbf{w})$
  compute **inverse Hessian approximation** $\mathbf{S}_t$ *"a la L-BFGS"*
  compute descent direction $\mathbf{d}_t = -(\mathbf{S}_t)\,\mathbf{g}_t$
  do line search for α, and update $\mathbf{w} \leftarrow \mathbf{w} + \alpha\mathbf{d}_t$
  clip w if necessary to stay in the same orthant
end for

- **Pros:** provably convergent; updates are sparse due to the clipping.
- **Cons:** not applicable to group-regularizers.

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".

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

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.

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 $\text{prox}_{\Omega} = P_{\mathcal{S}}$.

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 $\mathrm{prox}_{\Omega} = P_{\mathcal{S}}$.

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).

Majorization-Minimization Derivation

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

Separable 2nd order approximation of $\Lambda(\mathbf{w})$ around $\mathbf{w}_t$

$$\Lambda(\mathbf{w}') + (\mathbf{w} - \mathbf{w}_t)^\top \nabla\Lambda(\mathbf{w}') + \frac{1}{2\eta_t}\|\mathbf{w} - \mathbf{w}'\|^2 = Q(\mathbf{w}, \mathbf{w}_t)$$

Majorization-Minimization Derivation

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

Separable 2nd order approximation of $\Lambda(\mathbf{w})$ around $\mathbf{w}_t$

$$\Lambda(\mathbf{w}') + (\mathbf{w} - \mathbf{w}_t)^\top \nabla\Lambda(\mathbf{w}') + \frac{1}{2\eta_t}\|\mathbf{w} - \mathbf{w}'\|^2 = Q(\mathbf{w}, \mathbf{w}_t) \geq \Lambda(\mathbf{w})$$

if $\eta_t \leq 1/L$, with equality for $\mathbf{w} = \mathbf{w}_t$.

Majorization-Minimization Derivation

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

Separable 2nd order approximation of $\Lambda(\mathbf{w})$ around $\mathbf{w}_t$

$$\Lambda(\mathbf{w}') + (\mathbf{w} - \mathbf{w}_t)^\top \nabla\Lambda(\mathbf{w}') + \frac{1}{2\eta_t}\|\mathbf{w} - \mathbf{w}'\|^2 = Q(\mathbf{w}, \mathbf{w}_t) \geq \Lambda(\mathbf{w})$$

if $\eta_t \leq 1/L$, with equality for $\mathbf{w} = \mathbf{w}_t$.

Consequently, if $\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} Q(\mathbf{w}, \mathbf{w}_t) + \Omega(\mathbf{w})$,

$$\begin{aligned} \Lambda(\mathbf{w}_{t+1}) + \Omega(\mathbf{w}_{t+1}) &\leq Q(\mathbf{w}_{t+1}, \mathbf{w}_t) + \Omega(\mathbf{w}_{t+1}) \\ &\leq Q(\mathbf{w}_t, \mathbf{w}_t) + \Omega(\mathbf{w}_1) = \Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t) \end{aligned}$$

Majorization-Minimization Derivation

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

Separable 2nd order approximation of $\Lambda(\mathbf{w})$ around $\mathbf{w}_t$

$$\Lambda(\mathbf{w}') + (\mathbf{w} - \mathbf{w}_t)^\top \nabla\Lambda(\mathbf{w}') + \frac{1}{2\eta_t}\|\mathbf{w} - \mathbf{w}'\|^2 = Q(\mathbf{w}, \mathbf{w}_t) \geq \Lambda(\mathbf{w})$$

if $\eta_t \leq 1/L$, with equality for $\mathbf{w} = \mathbf{w}_t$.

Consequently, if $\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} Q(\mathbf{w}, \mathbf{w}_t) + \Omega(\mathbf{w})$,

$$\begin{aligned} \Lambda(\mathbf{w}_{t+1}) + \Omega(\mathbf{w}_{t+1}) &\leq Q(\mathbf{w}_{t+1}, \mathbf{w}_t) + \Omega(\mathbf{w}_{t+1}) \\ &\leq Q(\mathbf{w}_t, \mathbf{w}_t) + \Omega(\mathbf{w}_1) = \Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t) \end{aligned}$$

Easy to show that

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

Thus, with $\eta_t \leq 1/L$: objective monotonically decreases.

Monotonicity and Convergence

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

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

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):

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

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)

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

Monotonicity and Convergence

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

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

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)

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

Important: monotonicity doesn't imply convergence of $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_t, \ldots$.

Convergence (even with inexact steps) proved for $\eta_t \leq 2/L$ (Combettes and Wajs, 2006).

Accelerating IST: SpaRSA

The step sizes $\eta_t \leq 2/L$ (guarantees convergence) and $\eta_t \leq 1/L$ (monotonicity) are too conservative.

Accelerating IST: SpaRSA

The step sizes $\eta_t \leq 2/L$ (guarantees convergence) and $\eta_t \leq 1/L$ (monotonicity) are too conservative.

A bolder choice: let η_t mimic a Newton step (Barzilai and Borwein, 1988),

$$\frac{1}{\eta_t}\mathbf{I} \sim \mathbf{H}(\mathbf{w}_t) \quad \text{(Hessian)}$$

Accelerating IST: SpaRSA

The step sizes $\eta_t \leq 2/L$ (guarantees convergence) and $\eta_t \leq 1/L$ (monotonicity) are too conservative.

A bolder choice: let η_t mimic a Newton step (Barzilai and Borwein, 1988),

$$\frac{1}{\eta_t}\mathbf{I} \sim \mathbf{H}(\mathbf{w}_t) \quad \text{(Hessian)}$$

Approximation in the mean squared sense over the previous step:

$$\frac{1}{\eta_t} = \arg\min_{\alpha} \|\alpha(\mathbf{w}_t - \mathbf{w}_{t-1}) - (\nabla\Lambda(\mathbf{w}_t) - \nabla\Lambda(\mathbf{w}_{t-1}))\|^2$$

Accelerating IST: SpaRSA

The step sizes $\eta_t \leq 2/L$ (guarantees convergence) and $\eta_t \leq 1/L$ (monotonicity) are too conservative.

A bolder choice: let η_t mimic a Newton step (Barzilai and Borwein, 1988),

$$\frac{1}{\eta_t}\mathbf{I} \sim \mathbf{H}(\mathbf{w}_t) \quad \text{(Hessian)}$$

Approximation in the mean squared sense over the previous step:

$$\frac{1}{\eta_t} = \arg\min_{\alpha} \|\alpha(\mathbf{w}_t - \mathbf{w}_{t-1}) - (\nabla\Lambda(\mathbf{w}_t) - \nabla\Lambda(\mathbf{w}_{t-1}))\|^2$$

Resulting algorithm: SpaRSA (sparse reconstruction by separable approximation); shown to converge (with a safeguard) and to be fast Wright et al. (2009).

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):

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

Convergence of objective value (Beck and Teboulle, 2009)

$$\big(\Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)\big) - \big(\Lambda(\mathbf{w}^*) + \Omega(\mathbf{w}^*)\big) = O\left(\frac{1}{t^2}\right) \quad \text{(vs } O(1/t) \text{ for IST)}$$

Convergence of iterates has not been shown.

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):

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

Convergence of objective value (Beck and Teboulle, 2009)

$$\big(\Lambda(\mathbf{w}_t) + \Omega(\mathbf{w}_t)\big) - \big(\Lambda(\mathbf{w}^*) + \Omega(\mathbf{w}^*)\big) = O\left(\frac{1}{t^2}\right) \quad \text{(vs } O(1/t) \text{ for IST)}$$

Convergence of iterates has not been shown.

Another two-step method: TwIST (two-step IST) (Bioucas-Dias and Figueiredo, 2007).

Least Angle Regression (LARS)

LARS only applies to $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \; \lambda\|\mathbf{w}\|_1 + \|\mathbf{A}\mathbf{w} - \mathbf{y}\|^2}$

Least Angle Regression (LARS)

LARS only applies to $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \; \lambda\|\mathbf{w}\|_1 + \|\mathbf{A}\mathbf{w} - \mathbf{y}\|^2}$

Key ideas (Efron et al., 2004; Osborne et al., 2000)

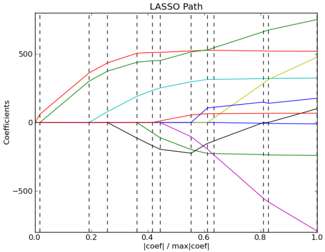
- "regularization path" $\hat{\mathbf{w}}(\lambda)$ is piecewise linear (Markowitz, 1952);
- the cusps can be identified in closed form;
- simply jump from one cusp to the next.

Least Angle Regression (LARS)

LARS only applies to $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \ \lambda\|\mathbf{w}\|_1 + \|\mathbf{A}\mathbf{w} - \mathbf{y}\|^2}$

Key ideas (Efron et al., 2004; Osborne et al., 2000)

- "regularization path" $\hat{\mathbf{w}}(\lambda)$ is piecewise linear (Markowitz, 1952);
- the cusps can be identified in closed form;
- simply jump from one cusp to the next.

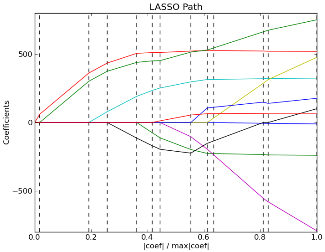


Least Angle Regression (LARS)

LARS only applies to $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \ \lambda\|\mathbf{w}\|_1 + \|\mathbf{A}\mathbf{w} - \mathbf{y}\|^2}$

Key ideas (Efron et al., 2004; Osborne et al., 2000)

- "regularization path" $\hat{\mathbf{w}}(\lambda)$ is piecewise linear (Markowitz, 1952);
- the cusps can be identified in closed form;
- simply jump from one cusp to the next.



Cons: doesn't apply to group regularizers; exponential worst case complexity (Mairal and Yu, 2012).

Homotopy/Continuation Methods

LARS is related to a more general family: homotopy/continuation methods.

Consider $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \; \lambda\bar{\Omega}(\mathbf{w}) + \Lambda(\mathbf{w})}$

Homotopy/Continuation Methods

LARS is related to a more general family: homotopy/continuation methods.

Consider $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \; \lambda\bar{\Omega}(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key ideas

- start with high value of λ, such that $\hat{\mathbf{w}}(\lambda)$ is easy (*e.g.*, zero);
- slowly decrease λ while "tracking" the solution;
- "tracking" means: use the previous $\hat{\mathbf{w}}(\lambda)$ to "warm start" the solver for the next problem.

Homotopy/Continuation Methods

LARS is related to a more general family: homotopy/continuation methods.

Consider $\boxed{\hat{\mathbf{w}}(\lambda) = \arg\min_{\mathbf{w}} \; \lambda\bar{\Omega}(\mathbf{w}) + \Lambda(\mathbf{w})}$

Key ideas

- start with high value of λ, such that $\hat{\mathbf{w}}(\lambda)$ is easy (*e.g.*, zero);
- slowly decrease λ while "tracking" the solution;
- "tracking" means: use the previous $\hat{\mathbf{w}}(\lambda)$ to "warm start" the solver for the next problem.

It's a meta-algorithm of general applicability when using "warm startable" solvers (Figueiredo et al., 2007; Hale et al., 2008; Osborne et al., 2000).

Some Stuff We Didn't Talk About

- shooting method (Fu, 1998);
- grafting (Perkins et al., 2003) and grafting-light (Zhu et al., 2010);
- forward stagewise regression (Hastie et al., 2007);
- alternating direction method of multipliers (ADMM) (Figueiredo and Bioucas-Dias, 2011).

Some Stuff We Didn't Talk About

- shooting method (Fu, 1998);
- grafting (Perkins et al., 2003) and grafting-light (Zhu et al., 2010);
- forward stagewise regression (Hastie et al., 2007);
- alternating direction method of multipliers (ADMM) (Figueiredo and Bioucas-Dias, 2011).

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

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

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 $\boxed{\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_1} \Longrightarrow \tilde{\nabla}\Omega(\mathbf{w}) = \lambda \text{sign}(\mathbf{w})$

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

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

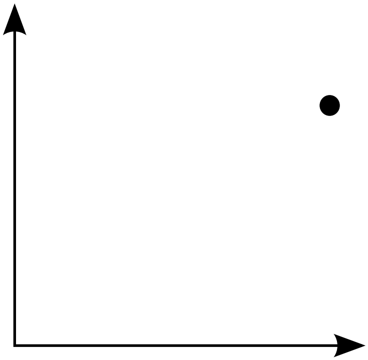
$$\mathbf{w} \leftarrow \underbrace{(1 - \eta_t\lambda)\mathbf{w}}_{\text{scaling}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

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

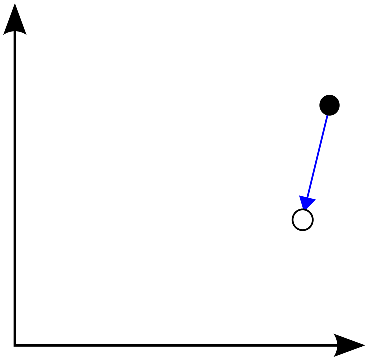
$$\mathbf{w} \leftarrow \underbrace{\mathbf{w} - \eta_t\lambda \text{sign}(\mathbf{w})}_{\text{constant penalty}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

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

Plain SGD with ℓ_2-regularization

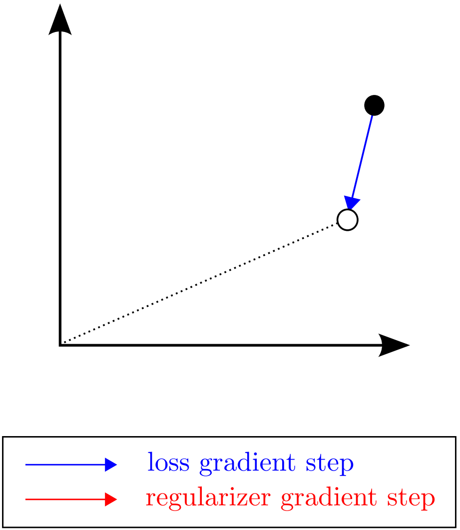


Plain SGD with ℓ_2-regularization

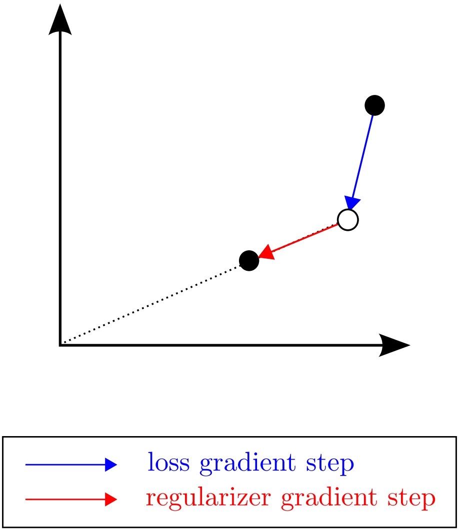


→	loss gradient step
→	regularizer gradient step

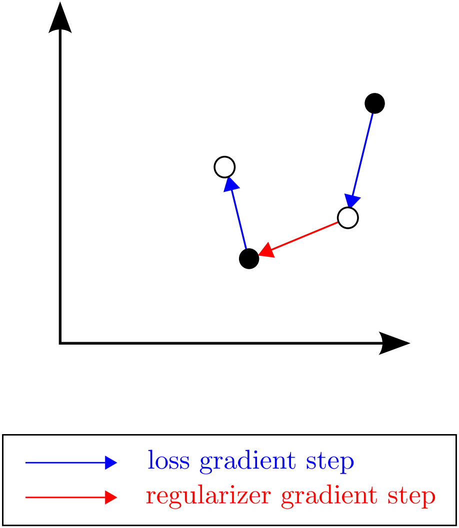
Plain SGD with ℓ_2-regularization



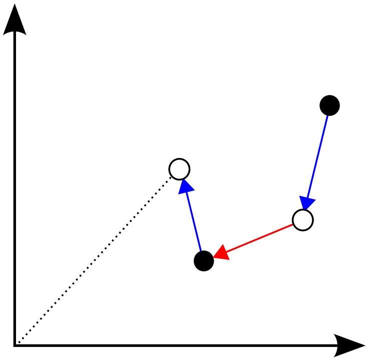
Plain SGD with ℓ_2-regularization

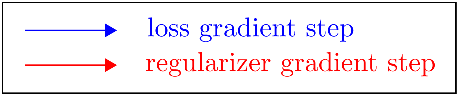


Plain SGD with ℓ_2-regularization

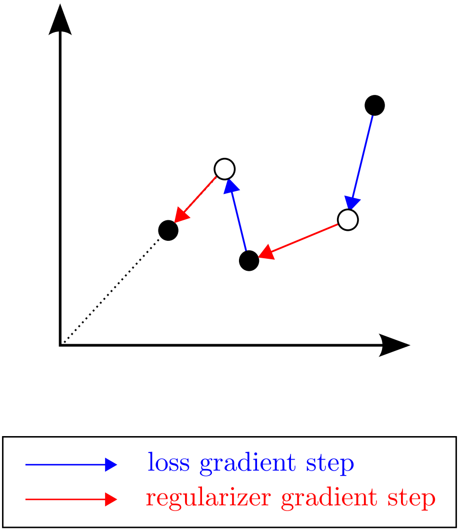


Plain SGD with ℓ_2-regularization

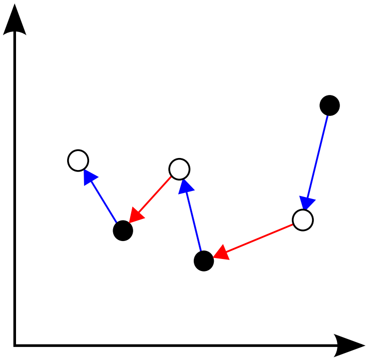




Plain SGD with ℓ_2-regularization

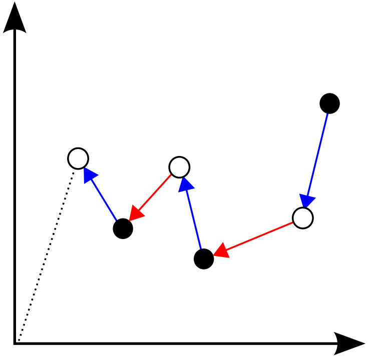


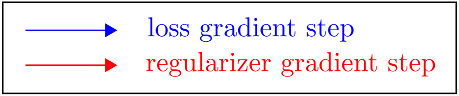
Plain SGD with ℓ_2-regularization



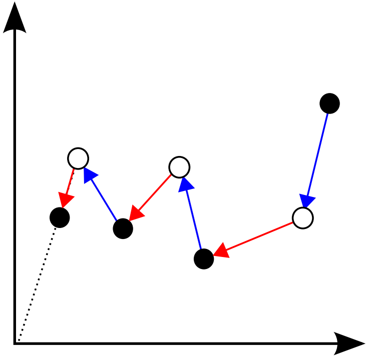
→	loss gradient step
→	regularizer gradient step

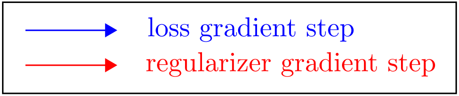
Plain SGD with ℓ_2-regularization



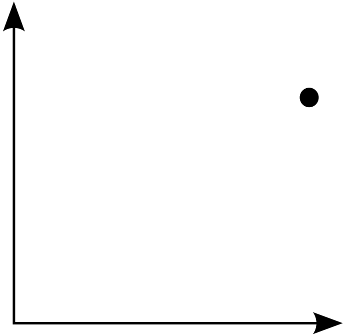


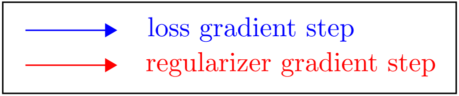
Plain SGD with ℓ_2-regularization



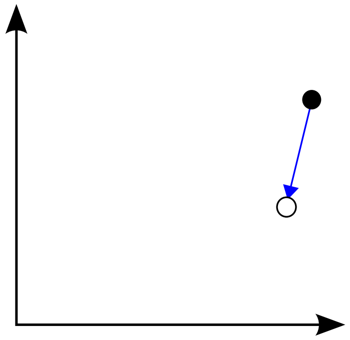


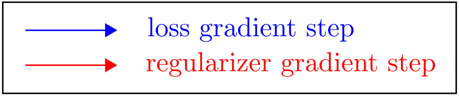
Plain SGD with ℓ_1-regularization



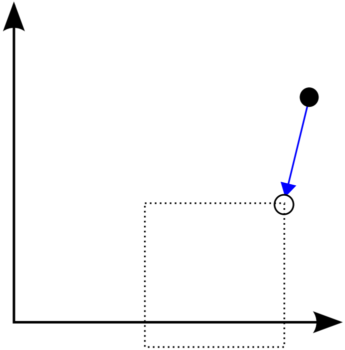


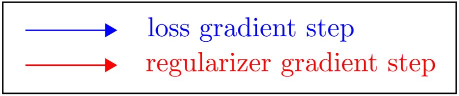
Plain SGD with ℓ_1-regularization



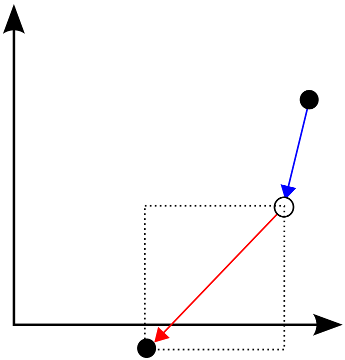


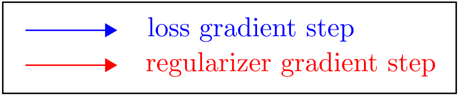
Plain SGD with ℓ_1-regularization



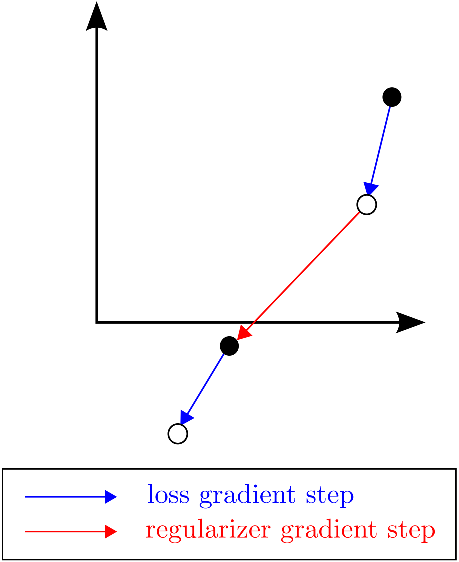


Plain SGD with ℓ_1-regularization

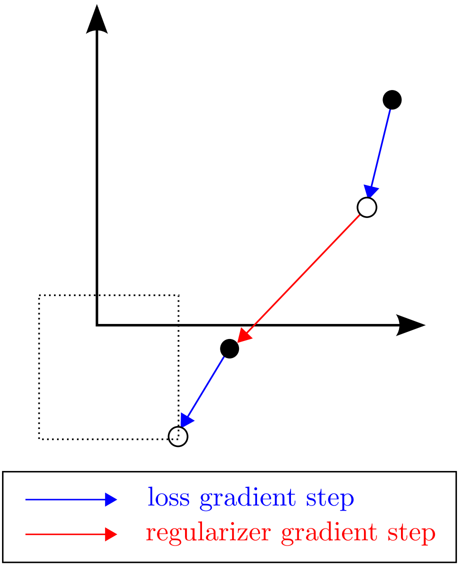




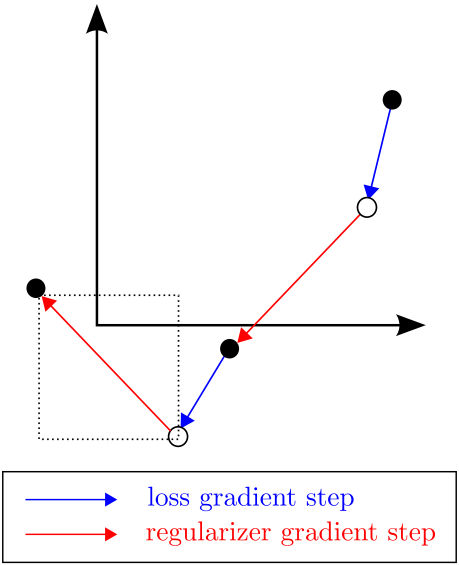
Plain SGD with ℓ_1-regularization



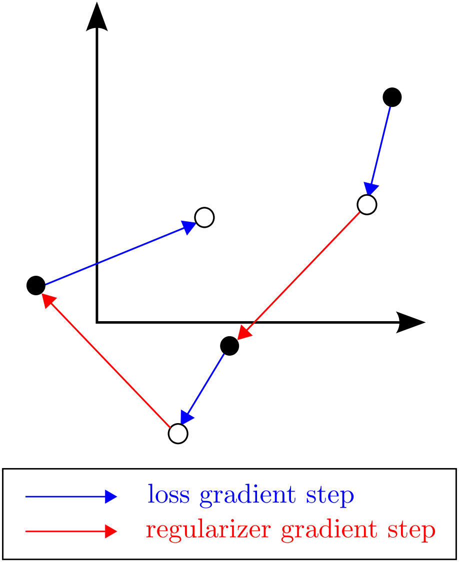
Plain SGD with ℓ_1-regularization



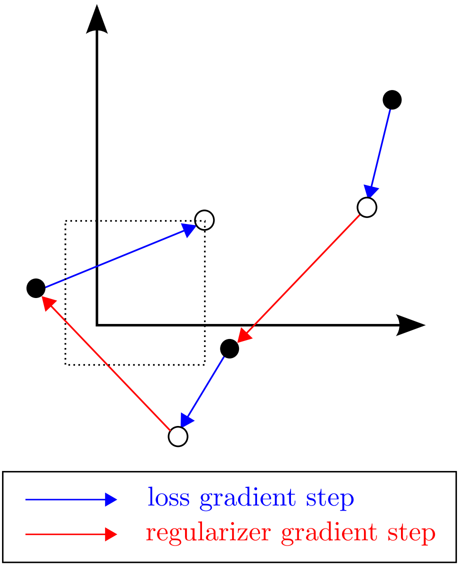
Plain SGD with ℓ_1-regularization



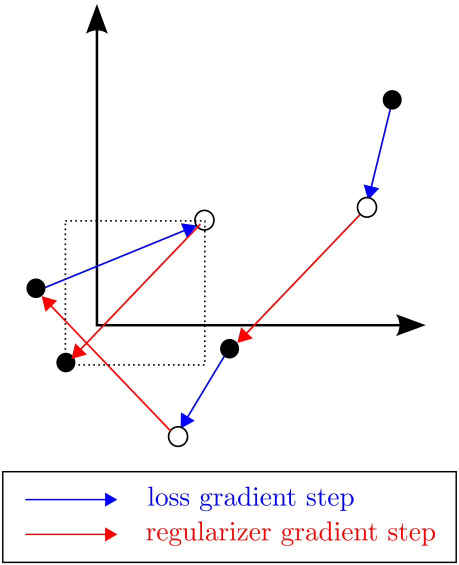
Plain SGD with ℓ_1-regularization



Plain SGD with ℓ_1-regularization



Plain SGD with ℓ_1-regularization



“Sparse” Online Algorithms

- SGD with Cumulative Penalty (Tsuruoka et al., 2009)
- 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)

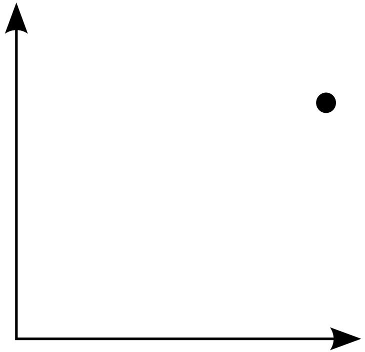
“Sparse” Online Algorithms

- SGD with Cumulative Penalty (Tsuruoka et al., 2009)
- 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)

Cumulative Penalties (Tsuruoka et al., 2009)

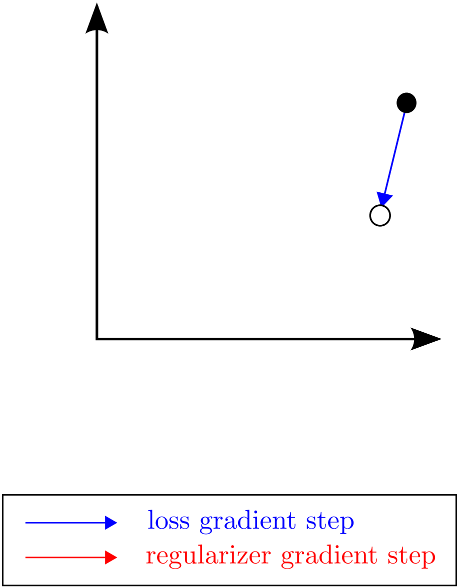
- an attempt to reconcile SGD and ℓ_1 regularizarion, maintaining algorithmic efficiency
- **computational trick:** accumulates the penalties, and applies them all at once when a feature fires (due to Carpenter (2008))
- **clipping:** if the total penalty is greater than the magnitude of the feature weight w_j, clip w_j to zero
  - **but store the amount of clipping for future use.**
- **leads to very sparse models**
- **however: no proof of convergence**

Cumulative Penalties (Tsuruoka et al., 2009)

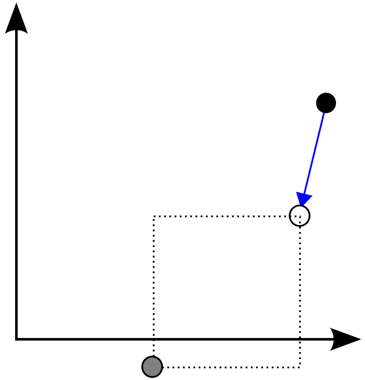


→	loss gradient step
→	regularizer gradient step

Cumulative Penalties (Tsuruoka et al., 2009)

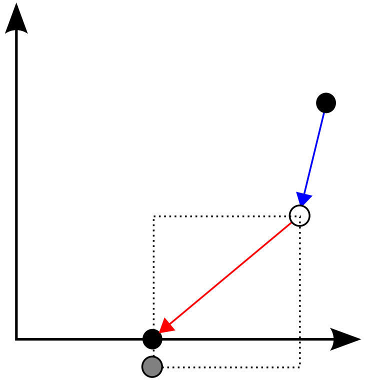


Cumulative Penalties (Tsuruoka et al., 2009)



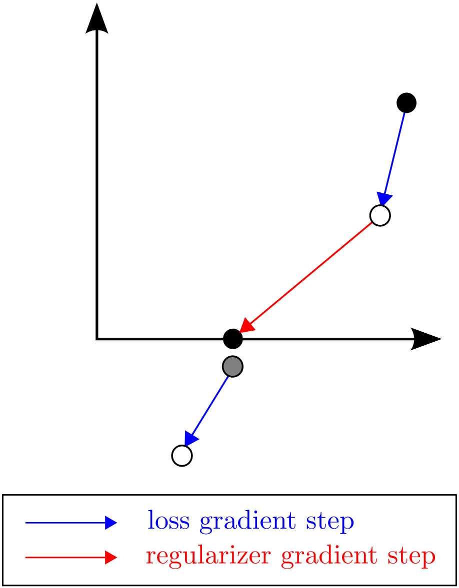
→	loss gradient step
→	regularizer gradient step

Cumulative Penalties (Tsuruoka et al., 2009)

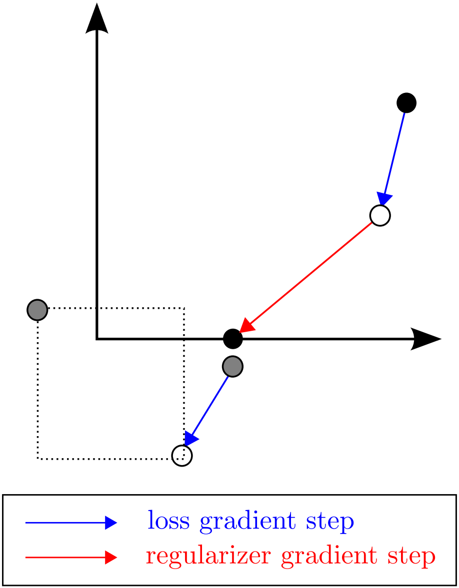


→	loss gradient step
→	regularizer gradient step

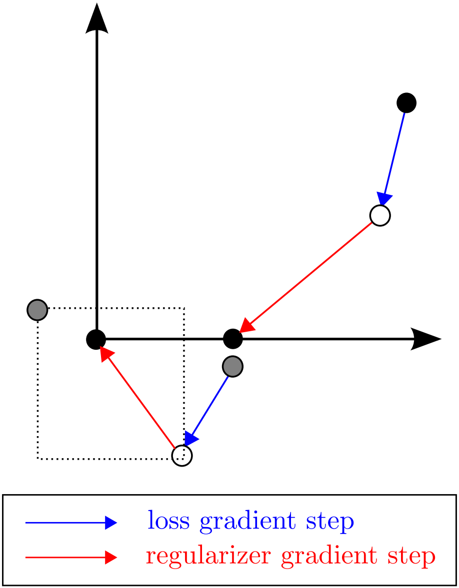
Cumulative Penalties (Tsuruoka et al., 2009)



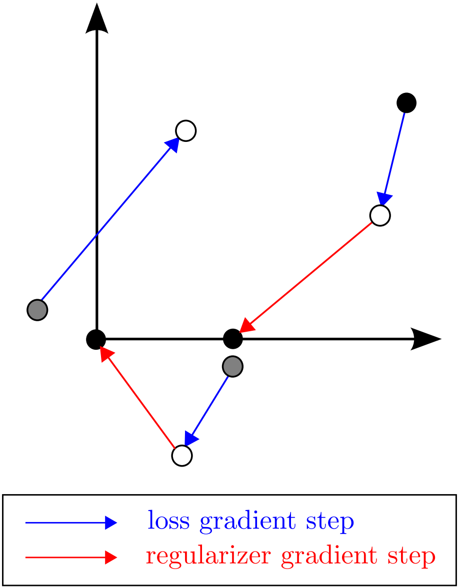
Cumulative Penalties (Tsuruoka et al., 2009)



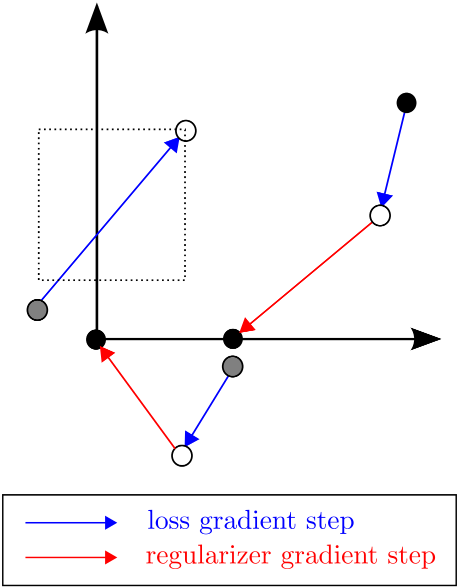
Cumulative Penalties (Tsuruoka et al., 2009)



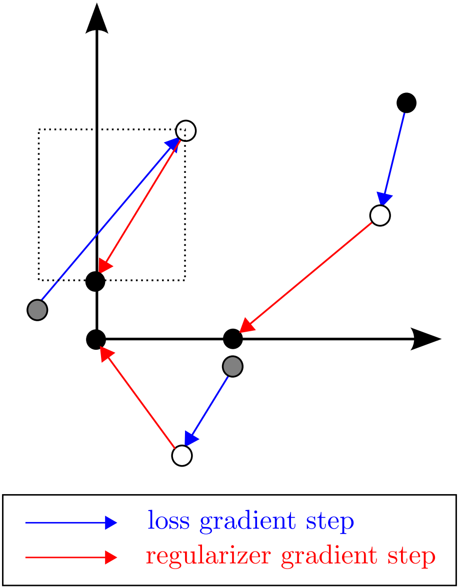
Cumulative Penalties (Tsuruoka et al., 2009)



Cumulative Penalties (Tsuruoka et al., 2009)



Cumulative Penalties (Tsuruoka et al., 2009)



"Sparse" Online Algorithms

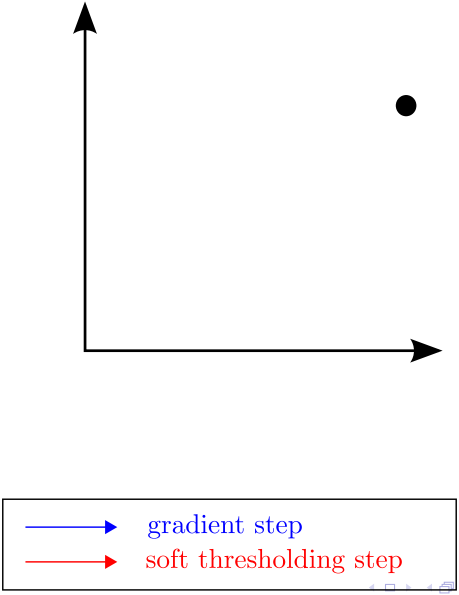
- SGD with Cumulative Penalty (Tsuruoka et al., 2009)
- Truncated Gradient (Langford et al., 2009)
- Online Forward-Backward Splitting (Duchi and Singer, 2009)
- Regularized Dual Averaging (Xiao, 2010)
- Online Proximal Gradient (Martins et al., 2011a)

Truncated Gradient (Langford et al., 2009)

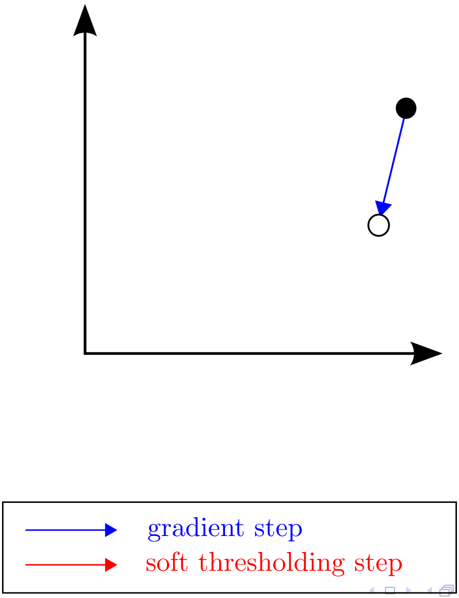
input: laziness coefficient K, stepsize sequence $(\eta_t)_{t=1}^{T}$
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  (sub-)gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \tilde{\nabla} L(\boldsymbol{\theta}; x_t, y_t)$
  if t/K is integer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

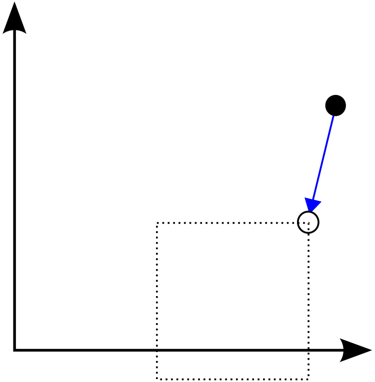
Truncated Gradient (Langford et al., 2009)

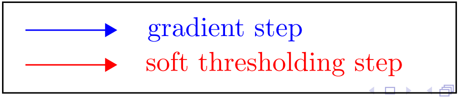


Truncated Gradient (Langford et al., 2009)

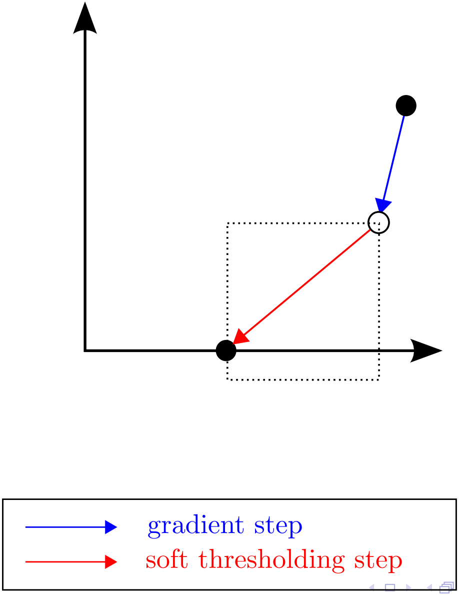


Truncated Gradient (Langford et al., 2009)

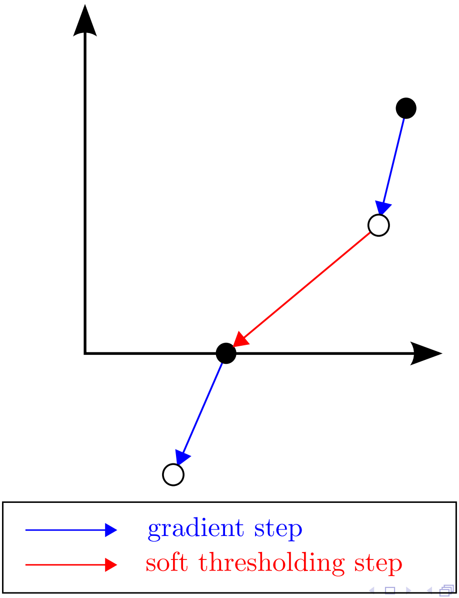




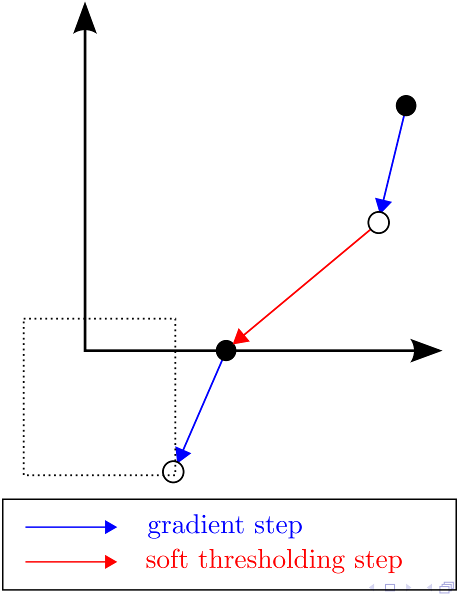
Truncated Gradient (Langford et al., 2009)



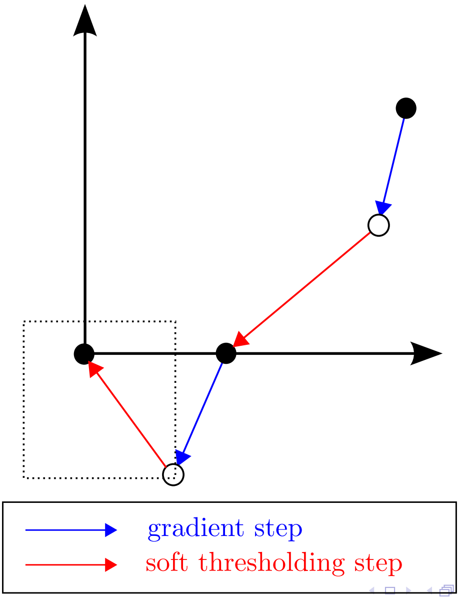
Truncated Gradient (Langford et al., 2009)



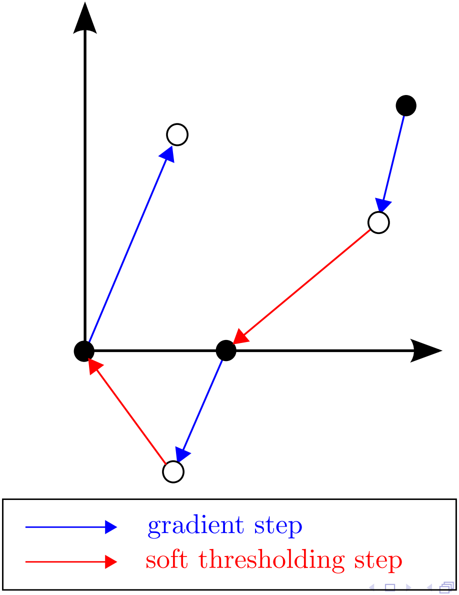
Truncated Gradient (Langford et al., 2009)



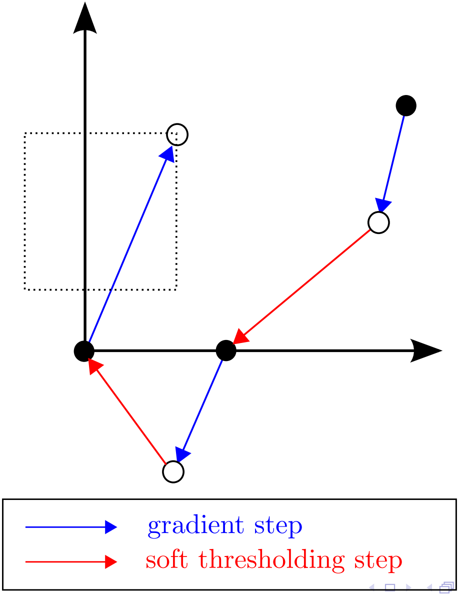
Truncated Gradient (Langford et al., 2009)



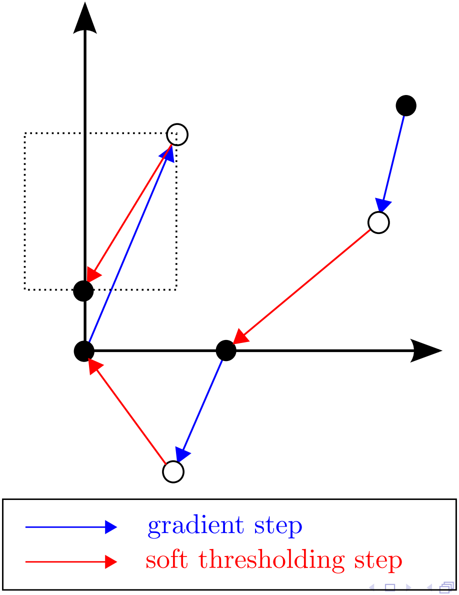
Truncated Gradient (Langford et al., 2009)



Truncated Gradient (Langford et al., 2009)



Truncated Gradient (Langford et al., 2009)



"Sparse" Online Algorithms

- SGD with Cumulative Penalty (Tsuruoka et al., 2009)
- 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

- SGD with Cumulative Penalty (Tsuruoka et al., 2009)
- 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: $\mathbf{w} \leftarrow \eta_0 \sqrt{t} \times \text{prox}_{\Omega}(-\mathbf{s}/t)$
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 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

- SGD with Cumulative Penalty (Tsuruoka et al., 2009)
- 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

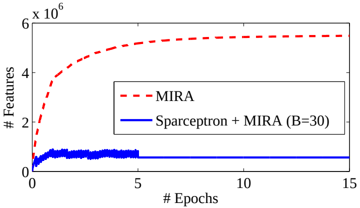
- **PAC Convergence.** ϵ-accurate solution after $T \leq O(1/\epsilon^2)$ rounds
- **Computational efficiency.** Each gradient step is **linear** in the number of features that fire.
  Each proximal step is **linear** in the number of groups M.
  Both are **independent** of D.

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?
Coordinate descent	✓	?	✓	Maybe	No
Prox-grad (IST)	✓	$O(1/\epsilon)$	Yes/No	✓	Not easy
OWL-QN	✓	?	Yes/No	No	No
SpaRSA	✓	$O(1/\epsilon)$	Yes/No	✓	Not easy
FISTA	✓	$O(1/\sqrt{\epsilon})$	Yes/No	✓	Not easy
ADMM (C-SALSA)	✓	?	No	✓	✓
Online subgradient	✓	$O(1/\epsilon^2)$	No	✓	No
Cumulative penalty	?	?	✓	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

Relatively few to date (but this list may not be exhaustive).

1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

Applications of Structured Sparsity in NLP

Relatively few to date (but this list may not be exhaustive).

1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

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:

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

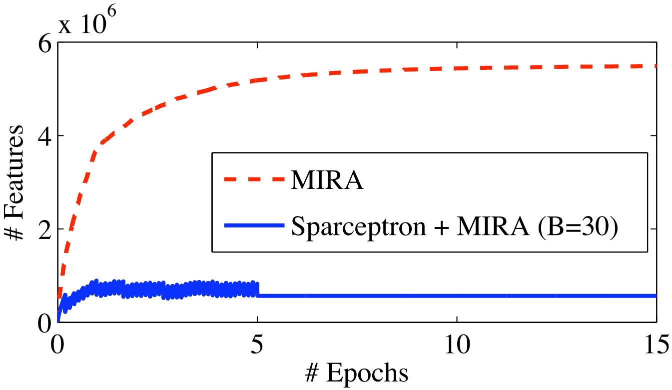
for M groups/templates.

Chunking

- CoNLL 2000 shared task (Sang and Buchholz, 2000)
- Unigram features: 96 feature templates using POS tags, words, and word shapes, with various context sizes
- Bigram features: 1 template indicating the label bigram
- **Baseline**: L_2-regularized MIRA, 15 epochs, all features, cross-validation to choose regularization strength
- **Template-based group lasso**: 5 epochs of sparseptron + 10 of MIRA

Chunking Experiments

	Baseline	Template-based group lasso			
# templates	96	10	20	30	40
model size	5,300,396	71,075	158,844	389,065	662,018
F_1 (%)	93.10	92.99	93.28	**93.59**	93.42



Memory requirement of sparseptron is $<$ 7.5% of that of the baseline.
(Oscillations are due to proximal steps after every 1,000 instances.)

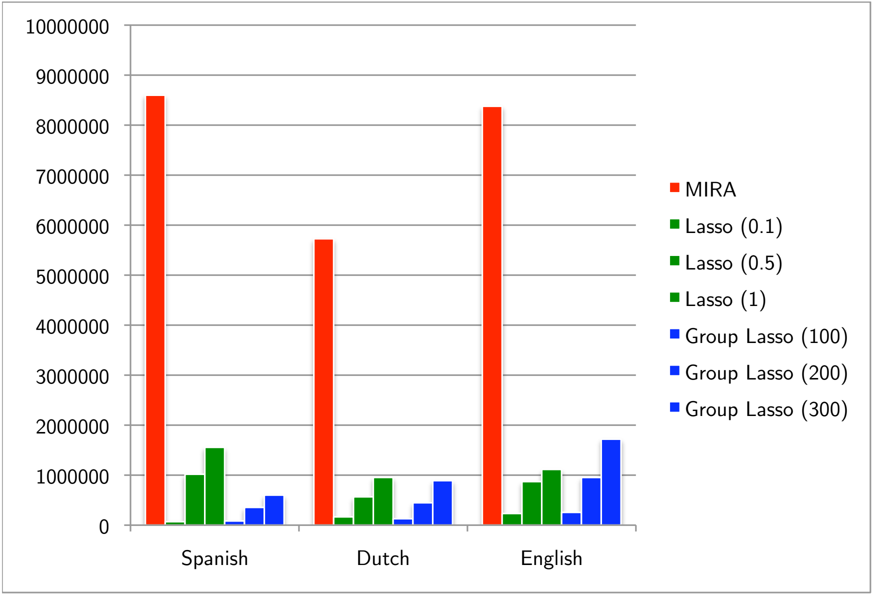
Applications of Structured Sparsity in NLP

Relatively few to date (but this list may not be exhaustive).

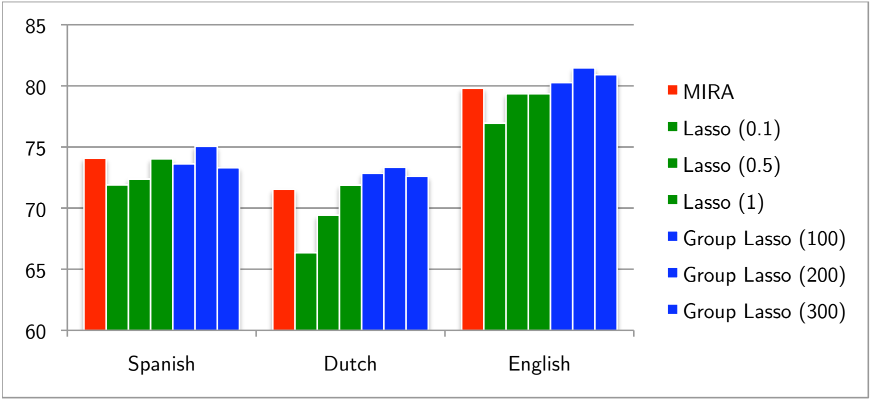
1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

Named Entity Recognition

- CoNLL 2002/2003 shared tasks (Sang, 2002; Sang and De Meulder, 2003): Spanish, Dutch, and English
- Unigram features: 452 feature templates using POS tags, words, word shapes, prefixes, suffixes, and other string features, all with various context sizes
- Bigram features: 1 template indicating the label bigram
- **Baselines**:
  - L_2-regularized MIRA, 15 epochs, all features, cross-validation to choose regularization strength
  - sparseptron with lasso, different values of C
- **Template-based group lasso**: 5 epochs of sparseptron + 10 of MIRA



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



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

Applications of Structured Sparsity in NLP

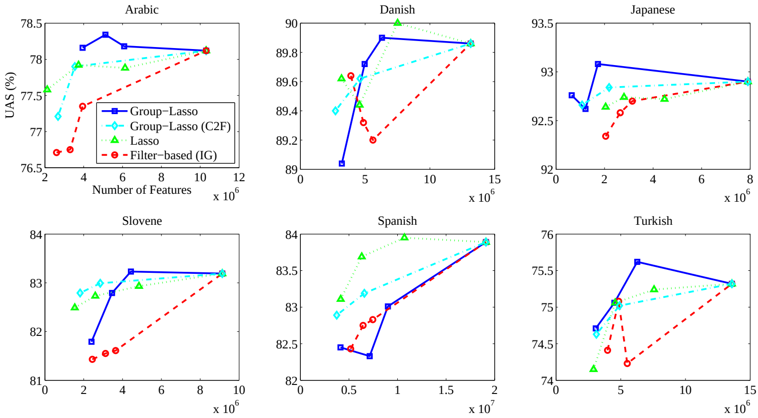
Relatively few to date (but this list may not be exhaustive).

1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

Non-projective Dependency Parsing

- CoNLL-X shared task (Buchholz and Marsi, 2006): Arabic, Danish, Dutch, Japanese, Slovene, and Spanish
- Arc-factored models (McDonald et al., 2005)
- 684 feature templates by conjoining words, shapes, lemmas, and POS of the head and the modifier, contextual POS, distance and attachment direction
- **Baselines**:
  - MIRA with all features
  - filter-based template selection (information gain)
  - standard lasso
- **Our methods**: template-based group lasso; coarse-to-fine regularization
- Budget sizes: 200, 300, and 400

Non-projective Dependency Parsing (c'ed)



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

Which features get selected?

- Qualitative analysis of selected templates:

	Arabic	Danish	Japanese	Slovene	Spanish	Turkish
Bilexical	++	+			+	
Lex. $\rightarrow$ POS	+		+			
POS $\rightarrow$ Lex.	++	+	+		+	+
POS $\rightarrow$ 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!

Applications of Structured Sparsity in NLP

Relatively few to date (but this list may not be exhaustive).

1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

Lexicon Expansion (Das and Smith, 2012)

- Desired: mapping from words (types) to categories (e.g., POS or semantic predicates)
- Allow ambiguity, but not too much!
- Given some words' mappings and a large corpus

Lexicon Expansion (Das and Smith, 2012)

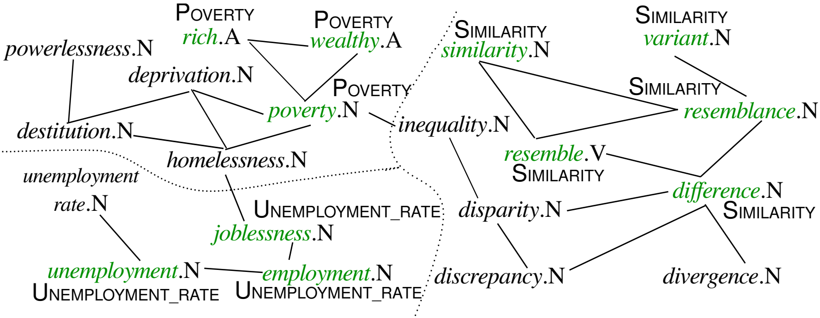
Approach:

1. Calculate distributional vectors for words
2. Construct a graph with words as vertices; edges from a word to the k-most similar distributional vectors
3. "Link" known words to empirical distributions
4. "Propagate" label distributions throughout the graph (Corduneanu and Jaakkola, 2003; Zhu et al., 2003; Subramanya and Bilmes, 2008, 2009; Talukdar and Crammer, 2009)

Known as **graph-based semisupervised learning**.

See Noah's talk about this work on Wednesday!

Example (Das and Smith, 2011)



Green words are observed in FrameNet data, each with a single frame (category); other words come from a larger, unlabeled corpus.

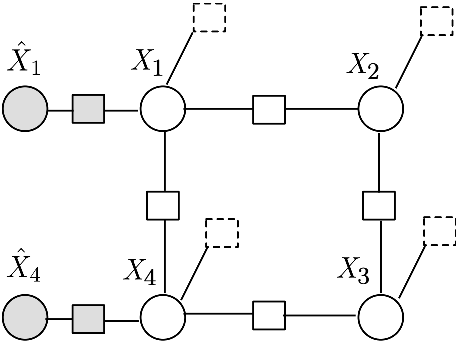
Graph-Based SSL

- Here we reason about types, not tokens (instances).
- Regularized empirical risk minimization doesn't quite describe this setting.
- Instead, think of maximum *a posteriori* inference in a factor graph $\mathcal{G} = (\mathbf{V}, \mathbf{F}, \mathbf{E})$:

$$p(\{v\}_{v \in \mathbf{V}}) = \frac{1}{Z} \prod_{f \in \mathbf{F}} \phi_f(\{v\}_{v \in \mathbf{V}:(v,f) \in \mathbf{E}})$$

where $\mathbf{V}$ are random variables, $\mathbf{F}$ are factors, and $\mathbf{E}$ are edges.

Factor Graph Representation of Graph-Based SSL



Shaded variables $\hat{X}_1$ and $\hat{X}_4$ take the values of empirical distributions over categories for words 1 and 4. Shaded factors encourage inferred distributions X_1 and X_4 to be similar to them. Solid white factors encourage smoothness across the graph, and dashed unary factors can be used to encourage sparsity.

Unary Factors

Here, $q_n(y)$ is an *unnormalized* distribution over categories given the word associated with the nth vertex.

Three unary factor conditions:

- Uniform squared ℓ_2: $-\lambda \sum_n \sum_y \left(q_n(y) - \frac{1}{|\mathcal{Y}|} \right)^2$
  - Used in past work (Subramanya et al., 2010); with quadratic pairwise penalties and normalized q, generalizes Zhu et al. (2003)
- Lasso (ℓ_1) for global sparsity: $-\lambda \sum_n \sum_y |q_n(y)|$
- Elitist lasso (squared $\ell_{1,2}$; Kowalski and Torrésani, 2009) for per-vertex sparsity):

$$-\lambda \sum_n \left(\sum_y |q_n(y)| \right)^2$$

Experimental Results: Expanding the FrameNet Lexicon

- Vertices: lemmatized, coarse POS-tagged word types
- Each $q_n(\cdot)$ is a(n unnormalized) distribution over 877 semantic frames
- 9,263 vertices with labels, 55,217 unlabeled
- Accuracy is for unknown predicates, partial match score (SemEval 2007)

	accuracy	lexicon size
supervised	46.62	–
Das and Smith (2011) (normalized q_n, squared ℓ_2-uniform)	62.35	128,960
squared ℓ_2-uniform	62.81	128,232
ℓ_1	62.43	128,771
squared $\ell_{1,2}$	**65.28**	**45,554**

Applications of Structured Sparsity in NLP

Relatively few to date (but this list may not be exhaustive).

1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

Unsupervised Tagging (Graça et al., 2009)

- **Posterior regularization** (Ganchev et al., 2010): penalize probabilistic models based on properties of their posterior distributions.
- One such property: for each token, the number of labels with nonzero probability.
  - Related idea: Ravi and Knight (2009) directly minimized the number of tag bigram types allowed by the model (using ILP).

Understanding Posterior Regularization

- Begin with a generative model $p_{\mathbf{w}}(X, Y)$. We are unsupervised here, so Y is always hidden.
- This leads to log marginal likelihood as loss: $L(\mathbf{w}; x_n) = -\log \sum_{y \in \mathcal{Y}} p_{\mathbf{w}}(x_n, y)$.
- For a given x, define a set of distributions $\mathcal{Q}_{x,\boldsymbol{\xi}} = \{q(Y \mid x) \mid \mathbb{E}_q[\mathbf{f}(x, Y)] - \mathbf{b} \le \boldsymbol{\xi}\}$.
- For a model distribution $p_{\mathbf{w}}(X, Y)$, define a regularizer:

$$\Omega(\mathbf{w}, \boldsymbol{\xi}) = \sigma \|\boldsymbol{\xi}\|_\beta + \sum_{n=1}^{N} \min_{q \in \mathcal{Q}_{x_n, \boldsymbol{\xi}}} \mathrm{KL}(q(\cdot) \| p_{\mathbf{w}}(\cdot \mid x_n))$$

Optimization for Posterior Regularization

An iterative EM-like algorithm can be used to locally optimize the regularized loss:

- E-step: calculate posteriors $p_{\mathbf{w}}(Y \mid x_n), \forall n$. (Hinges on local factorization of $p_{\mathbf{w}}$.)
- Project onto $\mathcal{Q}_{x_n}$: for each n:

$$\forall y, q(y \mid x_n) \propto p_{\mathbf{w}}(y \mid x_n) e^{-\boldsymbol{\lambda}_n^{*\top} \mathbf{f}(x_n, y)}$$

  where $\boldsymbol{\lambda}^*$ are the solution to the dual of the optimization problem inside Ω. (Hinges on local factorization of $\mathbf{f}$.)
- M-step: solve quasi-supervised problem using q to fill in the distribution over Y:

$$\min_{\mathbf{w}} \sum_{n=1}^{N} \sum_{y} -q(y \mid x_n) \log p_{\mathbf{w}}(x_n, y))$$

Posterior Sparsity (Graça et al., 2009)

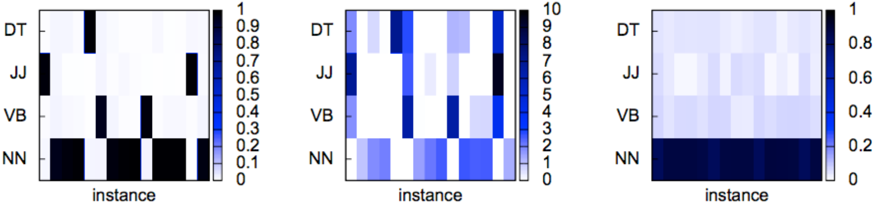
Define indicator features:

$$
\begin{aligned}
f_{i,w,t}(x_n, y) &= \begin{cases} 1 & \text{if } x_n \text{ contains the } i\text{th occurrence of } w \text{ at position } j \\ & \quad \text{and } y[j] = t \\ 0 & \text{otherwise} \end{cases} \\
b_{i,w,t} &= 0
\end{aligned}
$$

Regularize with the $\ell_{\infty,1}$ norm ($\boldsymbol{\xi}$ is solved for and substituted):

$$
\Omega(\mathbf{w}) = \sigma \underbrace{\sum_{w,t} \underbrace{\max_i \mathbb{E}_{p_\mathbf{w}}[f_{i,w,t}]}_{\ell_\infty}}_{\ell_1} + \sum_{n=1}^{N} \min_{q \in \mathcal{Q}_{x_n}} \mathrm{KL}(q(\cdot) \| p_\mathbf{w}(\cdot \mid x_n))
$$

The dual form of the optimization problem is solvable with projected gradient descent.



From Ganchev et al. (2010), figure 11. (Left) initial tag distributions for 20 instances of a word. (Middle) Optimal dual variables $\boldsymbol{\lambda}$; each row sums to $\sigma = 20$. (Right) q concentrates posteriors for all instances on the NN tag, reducing the $\ell_{\infty,1}$ norm from ≈ 4 to ≈ 1.

Unsupervised Tagging Results (Graça et al., 2009)

Estimator	PT-Conll		BG		PTB17	
	1-Many	1-1	1-Many	1-1	1-Many	1-1
EM	64.0(1.2)	40.4(3.0)	59.4(2.2)	42.0(3.0)	67.5(1.3)	46.4(2.6)
VEM(10^{-1})	60.4(0.6)	**51.1(2.3)**	54.9(3.1)	46.4(3.0)	68.2(0.8)*	**52.8(3.5)**
VEM(10^{-4})	63.2(1.0)*	48.1(2.2)	56.1(2.8)	43.3(1.7)*	67.3(0.8)*	49.6(4.3)
Sparse (10)	68.5(1.3)	43.3(2.2)	65.1(1.0)	48.0(3.3)	69.5(1.6)	50.0(3.5)
Sparse (32)	**69.2(0.9)**	43.2(2.9)	**66.0(1.8)**	48.7(2.2)	**70.2(2.2)**	49.5(2.0)
Sparse (100)	68.3(2.1)	44.5(2.4)	65.9(1.6)	**48.9(2.8)**	68.7(1.1)	47.8(1.5)*

Average accuracy (standard deviation in parentheses) over 10 different runs (random seeds identical across models) for 200 iterations. Sparse PR constraint strength is give in parentheses.

Applications of Structured Sparsity in NLP

Relatively few to date (but this list may not be exhaustive).

1. Martins et al. (2011b):
   - Phrase chunking
   - Named entity recognition
   - Dependency parsing
2. Semisupervised lexicon expansion (Das and Smith, 2012)
3. Unsupervised tagging (Graça et al., 2009)
4. Sociolinguistic association discovery (Eisenstein et al., 2011)

Sociolinguistic Association Discovery

- 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:

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

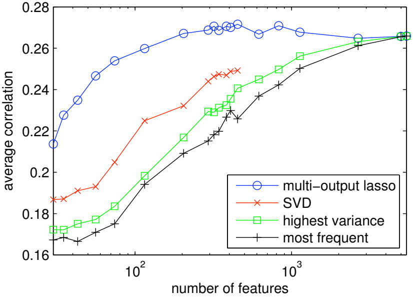
- Regressions are run in *both* directions.

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

- Drive entire rows of $\mathbf{W}$ 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:
  - Scale $\mathbf{X}$ to achieve variance 1 for each predictor
  - Precompute $\mathbf{C} = \mathbf{X}^\top \mathbf{Y} - N\bar{\mathbf{x}}^\top \bar{\mathbf{y}}$, where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are mean row vectors for $\mathbf{X}$, $\mathbf{Y}$, respectively
  - Precompute $\mathbf{D} = \mathbf{X}^\top \mathbf{X} - N\bar{\mathbf{x}}^\top \bar{\mathbf{x}}$
  - More regression tricks in Eisenstein et al. (2011)
- Related work: 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			-	+	-	-				

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

- Embed the model in a feature induction outer loop: "screen and clean" (Wu et al., 2010)
- Compare language model perplexity of models with no demographic features, raw demographic features (10), and 37 discovered conjunctive features.
  - Significant reduction compared to both baselines.

Predictive Demographic Features (Eisenstein et al., 2011)

	feature			positive terms	negative terms
1	geo: Northeast			m2 brib mangoville soho odeee	fasho #ilovefamu foo coo fina
2	geo: NYC			mangoville lolss m2 brib wordd	bahaha fasho goofy #ilovefamu tacos
4	geo: South+Midwest	renter ≤ 0.615	white ≤ 0.823	hme muthafucka bae charlotte tx	odeee m2 lolss diner mangoville
7	Afr. Am. > 0.101	renter > 0.615	Span. lang. > 0.063	dhat brib odeee lolss wassupp	bahaha charlotte california ikr enter
8	Afr. Am. ≤ 0.207	Hispanic > 0.119	Span. lang. > 0.063	les ahah para san donde	bmore ohio #lowkey #twitterjail nahhh
9	geo: NYC	Span. lang. ≤ 0.213		mangoville thatt odeee lolss buzzin	landed rodney jawn wiz golf
12	Afr. Am. > 0.442	geo: South+Midwest	white ≤ 0.823	#ilovefamu panama midterms willies #lowkey	knoe esta pero odeee hii
15	geo: West Coast	other lang. > 0.110		ahah fasho san koo diego	granted pride adore phat pressure
17	Afr. Am. > 0.442	geo: NYC	other lang. ≤ 0.110	lolss iim buzzin qonna qood	foo tender celebs pages pandora
20	Afr. Am. ≤ 0.207	Span. lang. > 0.063	white > 0.823	del bby cuando estoy muscle	knicks becoming uncomfortable large granted
23	Afr. Am. ≤ 0.050	geo: West	Span. lang. ≤ 0.106	leno it'd 15th hacked government	knicks liquor uu hunn homee
33	Afr. Am. > 0.101	geo: SF Bay	Span. lang. > 0.063	hella aha california bay o.o	aj everywhere phones shift regardless
36	Afr. Am. ≤ 0.050	geo: DC/Philadelphia	Span. lang. ≤ 0.106	deh opens stuffed yaa bmore	hmmmmm dyin tea cousin hella

Selected demographic features and words with high and low log-odds associated with each.

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, grid sparsity, and elite discovery, but many more are possible!
- Small/medium scale: many batch algorithms available, with fast convergence (IST, FISTA, SpaRSA, ...)
- Large scale: online proximal-gradient algorithms suitable to explore large feature spaces

Thank you!

- Questions?

Acknowledgments

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

References I

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.

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

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

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

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

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.

Blumensath, T. (2012). Compressed sensing with nonlinear observations and related nonlinear optimisation problems. Technical report, arXiv/1205.1650.

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

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

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

Candès, E., Romberg, J., and Tao, T. (2006b). Stable signal recovery from incomplete and inaccurate measurements. *Communications in Pure and Applied Mathematics*, 59:1207–1223.

Carpenter, B. (2008). Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, Technical report, Alias-i.

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.

Corduneanu, A. and Jaakkola, T. (2003). On information regularization. In *Proc. of UAI*.

Das, D. and Smith, N. A. (2011). Semi-supervised frame-semantic parsing for unknown predicates. In *Proc. of ACL*.

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

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

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.

References III

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

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

Ganchev, K., Graça, J., Gillenwater, J., and Taskar, B. (2010). Posterior regularization for structured latent variable models. *JMLR*, 11:2001–2049.

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

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

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.

Haupt, J. and Nowak, R. (2006). Signal reconstruction from noisy random projections. *IEEE Transactions on Information Theory*, 52:4036–4048.

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*.

References IV

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.

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

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

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

Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale CRFs. In *Proc. of ACL*.

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.

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

Mairal, J. and Yu, B. (2012). Complexity analysis of the lasso regularization path. Technical report, arXiv:1205.0079.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 7:77–91.

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.

Negahban, S., Ravikumar, P., Wainwright, M., and Yu, B. (2012). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. Technical report, Department of EECS, UC Berkeley.

References V

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.

Plan, Y. and Vershynin, R. (2012). Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. Technical report, arXiv/1202.1212.

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*.

Ravi, S. and Knight, K. (2009). Minimized models for unsupervised part-of-speech tagging. In *Proc. of ACL*.

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

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*.

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

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

Sokolovska, N., Lavergne, T., Cappé, O., and Yvon, F. (2010). Efficient learning of sparse conditional random fields for supervised sequence labelling. *IEEE Journal of Selected Topics in Signal Processing*, 4(6):953–964.

References VI

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.

Subramanya, A. and Bilmes, J. (2008). Soft-supervised learning for text classification. In *Proc. of EMNLP*.

Subramanya, A. and Bilmes, J. (2009). Entropic graph regularization in non-parametric semi-supervised classification. In *Proc. of NIPS*.

Subramanya, A., Petrov, S., and Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proc. of EMNLP*.

Talukdar, P. P. and Crammer, K. (2009). New regularized algorithms for transductive learning. In *Proc. of the ECML-PKDD*.

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.

Tillmann, A. and Pfetsch, M. (2012). The computational complexity of RIP, NSP, and related concepts in compressed sensing. Technical report, arXiv/1205.2081.

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

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. (2009). Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proc. of ACL*.

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

van de Geer, S. (2008). High-dimensional generalized linear models and the lasso. *The Annals of Statistics*, 36:614–645.

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.

Wu, J., Devlin, B., Ringquist, S., Trucco, M., and Roeder, K. (2010). Screen and clean: A tool for identifying interactions in genome-wide association studies. *Genetic Epidemiology*, 34(3):275–285.

References VII

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*.

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.

Yun, S. and Toh, K.-C. (2011). A coordinate gradient descent method for L1-regularized convex minimization. *Computational Optimization and Applications*, 48:273–307.

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.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proc. of ICML*.