Sparsity and Structured Sparsity in Machine Learning:

Models, Algorithms, and Applications

Mário A. T. Figueiredo

Instituto de Telecomunicações
Instituto Superior Técnico, Lisboa, Portugal

Joint work with:

André T. Martins, Priberam and Instituto de Telecomunicações, Lisboa, Portugal

Noah A. Smith, Language Technologies Institute, Carnegie Mellon University, USA

February 2013

Outline

Notation

Many machine learning (ML) problems involve learning a mapping from one 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 $h_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$

Notation

Many machine learning (ML) problems involve learning a mapping from one 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 $h_{\mathbf{w}} : \mathcal{X} \rightarrow \mathcal{Y}$

$h_{\mathbf{w}}\Big($  $\Big) \rightarrow \{\text{coast}, \text{mountain}\}$

Notation

Many machine learning (ML) problems involve learning a mapping from one 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 $h_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$

$h_{\mathbf{w}}\Big($  $\Big) \to \{\text{coast, mountain}\}$

x	Only RB	France NNP	and CC	Britain NNP	backed VBD	Fischler NNP	's POS	proposal NN
y = $h_{\mathbf{w}}$(x)	O	I-LOC	O	I-LOC	O	I-PER	O	O

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, unsupervised feature learning, etc.
- $\mathbf{w} \in \mathbb{R}^D$ are the weights that parameterize the mapping.

Often, *e.g.*, in *natural language processing* (NLP) D is very large ($> 10^6$).

In some cases (*e.g.*, in NLP), the maximization above is itself challenging.

Learning Linear Models

- We observe a collection of examples $\{(x_n, y_n)\}_{n=1}^{N}$.

$y = \text{mountain}$

$y = \text{coast}$

- **Learn** $\mathbf{w}$ from the data, via regularized empirical risk minimization,

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

Logistic regression, perceptron, conditional random fields, SVM, some supervised generative models, all fit the linear modeling framework.

What is Sparsity? Why Is It Important?

- The **sparsity hypothesis**: some/most dimensions of $\mathbf{f}$ are not needed for a good $h_{\mathbf{w}}$; those dimensions can be zero, leading to a sparse $\mathbf{w}$.

What is Sparsity? Why Is It Important?

- The **sparsity hypothesis**: some/most dimensions of $\mathbf{f}$ are not needed for a good $h_{\mathbf{w}}$; those dimensions can be zero, leading to a sparse $\mathbf{w}$.

- The **bet on sparsity** (Friedman et al., 2004): it's often correct; if not, there's no good solution anyway!

What is Sparsity? Why Is It Important?

- The **sparsity hypothesis**: some/most dimensions of $\mathbf{f}$ are not needed for a good $h_{\mathbf{w}}$; those dimensions can be zero, leading to a sparse $\mathbf{w}$.
- The **bet on sparsity** (Friedman et al., 2004): it's often correct; if not, there's no good solution anyway!
- Models with just a few features are:
  - easier to explain/interpret
  - cheaper to implement

What is Sparsity? Why Is It Important?

- The **sparsity hypothesis**: some/most dimensions of $\mathbf{f}$ are not needed for a good $h_{\mathbf{w}}$; those dimensions can be zero, leading to a sparse $\mathbf{w}$.
- The **bet on sparsity** (Friedman et al., 2004): it's often correct; if not, there's no good solution anyway!
- Models with just a few features are:
  - easier to explain/interpret
  - cheaper to implement
- Generalization:
  - the goal of ML is to generalize well to new examples
  - encouraging the use of few features discourages overfitting

(Automatic) Feature Selection

Domain experts are often good at engineering features.

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

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

1. filters
2. wrappers
3. embedded methods

(Automatic) Feature Selection

Domain experts are often good at engineering features.

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

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

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

(Automatic) Feature Selection

Domain experts are often good at engineering features.

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

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

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

(Automatic) Feature Selection

Domain experts are often good at engineering features.

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

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

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

Embedded Methods for Feature Selection

Formulate the learning problem as a trade-off between

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

Embedded Methods for Feature Selection

Formulate the learning problem as a trade-off between

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

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

Embedded Methods for Feature Selection

Formulate the learning problem as a trade-off between

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

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

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

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

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

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: $A_{ij} = f_j(x_i)$.
- Response vector: $\mathbf{y} = [y_1, ..., y_N]^\top$.
- The most/best studied loss function (statistics, machine learning, signal processing); dates back to Legendre and Gauss (early 1800s).

Loss functions (II)

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

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

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):

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

- 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.)

The log-linear, hinge, and perceptron 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).

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 $\lim_{\|\mathbf{w}\| \rightarrow \infty} \Omega(\mathbf{w}) = \infty$ (**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 **w**.
- Select relevant features (via sparsity-inducing regularization).

Regularization Formulations

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

Regularization Formulations

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

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

Regularization Formulations

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

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

- Morozov regularization

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

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

- ... and the squared error (SE) loss: $L_{\mathrm{SE}}(\mathbf{w}; x_n, y_n) = \frac{1}{2}\left(y - \mathbf{w}^\top \mathbf{f}(x)\right)^2 = -\log \mathcal{N}(y|\mathbf{w}^\top \mathbf{f}(x), 1)$

Norms: A Quick Review

Before focusing on regularizers, a quick review about ℓ_p norms.

Norms: A Quick Review

Before focusing on regularizers, a quick review about ℓ_p norms.

Examples of norms:

Norms: A Quick Review

Before focusing on regularizers, a quick review about ℓ_p norms.

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 ℓ_p norms.

Examples of norms:

- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- Notable cases:
  - the famous ℓ_1 norm: $\|\mathbf{w}\|_1 = \sum_i |w_i|$
  - the ℓ_∞ norm: $\|\mathbf{w}\|_\infty = \lim_{p\rightarrow\infty} \|\mathbf{w}\|_p = \max\{|w_i|, \; i = 1, ..., D\}$

Norms: A Quick Review

Before focusing on regularizers, a quick review about ℓ_p norms.

Examples of norms:

- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- Notable cases:
  - the famous ℓ_1 norm: $\|\mathbf{w}\|_1 = \sum_i |w_i|$
  - the ℓ_∞ norm: $\|\mathbf{w}\|_\infty = \lim_{p\rightarrow\infty} \|\mathbf{w}\|_p = \max\{|w_i|, \; i = 1, ..., D\}$

Norms: A Quick Review

Before focusing on regularizers, a quick review about ℓ_p norms.

Examples of norms:

- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- Notable cases:
  - the famous ℓ_1 norm: $\|\mathbf{w}\|_1 = \sum_i |w_i|$
  - the ℓ_∞ norm: $\|\mathbf{w}\|_\infty = \lim_{p\rightarrow\infty} \|\mathbf{w}\|_p = \max\{|w_i|, \, i = 1, ..., D\}$

Fact: all norms are convex.

Norms: A Quick Review

Before focusing on regularizers, a quick review about ℓ_p norms.

Examples of norms:

- $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$ (called ℓ_p norm, for $p \geq 1$).
- Notable cases:
  - the famous ℓ_1 norm: $\|\mathbf{w}\|_1 = \sum_i |w_i|$
  - the ℓ_∞ norm: $\|\mathbf{w}\|_\infty = \lim_{p\to\infty} \|\mathbf{w}\|_p = \max\{|w_i|,\ i = 1, ..., D\}$

Fact: all norms are convex.

- The infamous ℓ_0 "norm": $\|\mathbf{w}\|_0 = \lim_{p\to 0} \|\mathbf{w}\|_p^p = |\{i : w_i \neq 0\}|$

  (not a norm).

Classical Regularizers: Ridge

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

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

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 (1962), Hoerl and Kennard (1970).

Classical Regularizers: Ridge

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

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 (1962), Hoerl and Kennard (1970).
- Closely related to Tikhonov (1943) and Wiener (1949).

Classical Regularizers: Ridge

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

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 (1962), Hoerl and Kennard (1970).
- Closely related to Tikhonov (1943) and Wiener (1949).
- Used early in computer vision (Bertero et al., 1988), (Poggio et al., 1985).

Classical Regularizers: Ridge

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

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 (1962), Hoerl and Kennard (1970).
- Closely related to Tikhonov (1943) and Wiener (1949).
- Used early in computer vision (Bertero et al., 1988), (Poggio et al., 1985).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in ML: Chen and Rosenfeld (1999).

Classical Regularizers: Ridge

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

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 (1962), Hoerl and Kennard (1970).
- Closely related to Tikhonov (1943) and Wiener (1949).
- Used early in computer vision (Bertero et al., 1988), (Poggio et al., 1985).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in ML: Chen and Rosenfeld (1999).
- **Pros**: smooth and convex, thus benign for optimization.

Classical Regularizers: Ridge

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

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 (1962), Hoerl and Kennard (1970).
- Closely related to Tikhonov (1943) and Wiener (1949).
- Used early in computer vision (Bertero et al., 1988), (Poggio et al., 1985).
- **Ridge logistic regression**: Schaefer et al. (1984), Cessie and Houwelingen (1992); in ML: Chen and Rosenfeld (1999).
- **Pros**: smooth and convex, thus benign for optimization.
- **Cons**: doesn't promote sparsity (no explicit feature selection).

Classical Regularizers: Lasso

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

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

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).

Classical Regularizers: Lasso

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

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

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

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),...
- ...also in NLP: Kazama and Tsujii (2003); Goodman (2004).

Classical Regularizers: Lasso

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

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),...
- ...also in NLP: Kazama and Tsujii (2003); Goodman (2004).
- **Pros**: encourages sparsity: embedded feature selection.

Classical Regularizers: Lasso

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

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),...
- ...also in NLP: Kazama and Tsujii (2003); Goodman (2004).
- **Pros**: encourages sparsity: embedded feature selection.
- **Cons**: convex, but non-smooth: more challenging optimization.

The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest (1D) case:

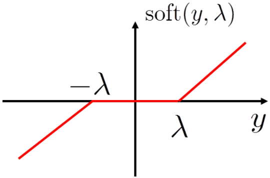
$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \lambda|w| = \text{soft}(y,\lambda) = \begin{cases} y-\lambda & \Leftarrow \quad y > \lambda \\ 0 & \Leftarrow \quad |y| \leq \lambda \\ y+\lambda & \Leftarrow \quad y < -\lambda \end{cases}$$

The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest (1D) case:

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

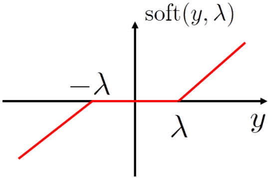


The Lasso and Sparsity

Why does the Lasso yield sparsity?

The simplest (1D) 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):

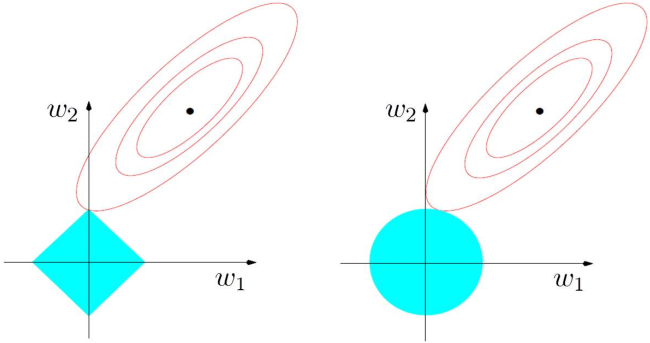
$$\widehat{w} = \arg\min_{w} \frac{1}{2}(w-y)^2 + \frac{\lambda}{2}w^2 = \frac{1}{1+\lambda}\,y$$

The Lasso and Sparsity (II)

Why does the Lasso yield sparsity?

The Lasso and Sparsity (II)

Why does the Lasso yield sparsity?



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

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

Relevant Theory?

Theoretical results for ℓ_1-regularized logistic regression:

- PAC-Bayesian bounds (generalization improves with sparsity): Krishnapuram et al. (2005)
- Logarithmic sample complexity (versus linear, for ridge) Ng (2004)
- Oracle bounds (van de Geer, 2008)
- Consistency (Negahban et al., 2012)

Outline

Models

ℓ_1 regularization promotes **sparse models**

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

Models

ℓ_1 regularization promotes **sparse models**

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

Main question: how to promote less trivial sparsity patterns?

Structured Sparsity and Groups

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

Structured Sparsity and Groups

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

Group sparsity: discard entire *groups* of features

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

Structured Sparsity and Groups

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

Group sparsity: discard entire *groups* of features

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

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

Structured Sparsity and Groups

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

Group sparsity: discard entire *groups* of features

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

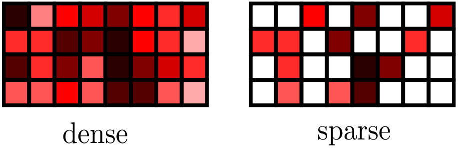
Yields statistical gains if prior assumptions are correct (Stojnic et al., 2009)

Many applications:

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

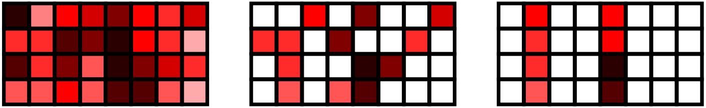
"Grid" Sparsity

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



"Grid" Sparsity

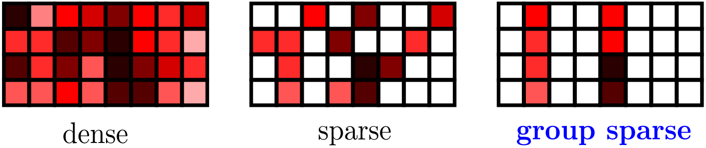
For feature spaces that can be arranged as a grid (examples next)



dense sparse **group sparse**

"Grid" Sparsity

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



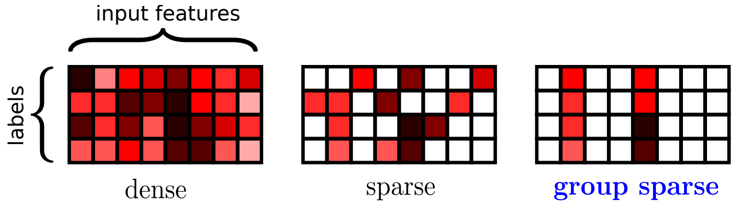
Goal: push *entire columns* to have zero weights

The groups are the columns of the grid

Example 1: Sparsity with Multiple Classes

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

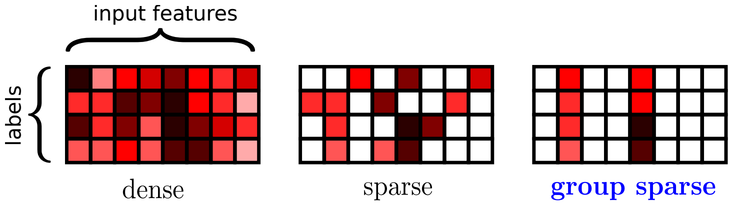
In words: we're conjoining each input feature with each output class



Example 1: Sparsity with Multiple Classes

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

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



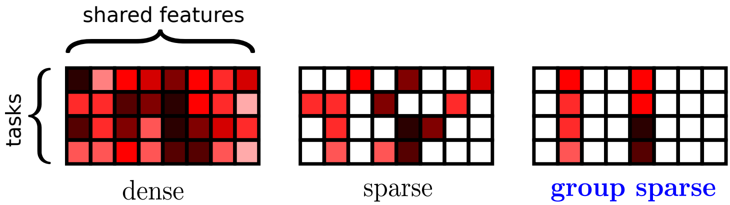
"Standard" sparsity is wasteful—we may still need all the input features

What we want: discard some input features

Solution: one group per *input* feature

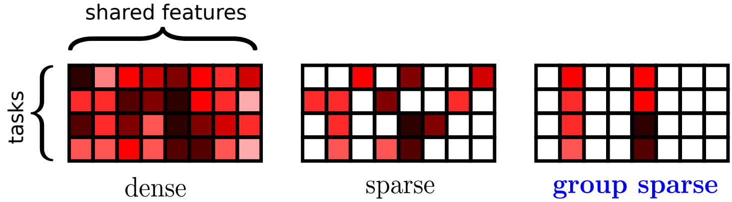
Example 2: Multi-Task Learning (Caruana, 1997; Obozinski et al., 2010)

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



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

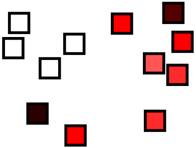
Same thing, except now rows are **tasks** and columns are **features**



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

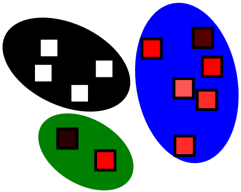
Solution: one group per feature

Group Sparsity



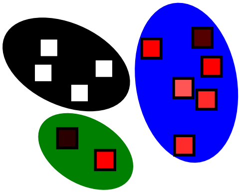
- D features

Group Sparsity



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

Group Sparsity

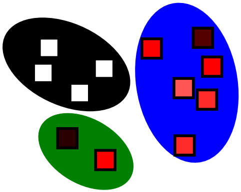


- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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

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

Group Sparsity



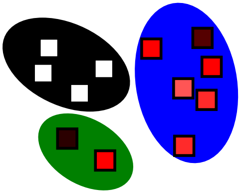
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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

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

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

Group Sparsity



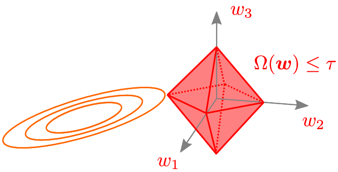
- D features
- M groups $G_1, \ldots, G_M$, each $G_m \subseteq \{1, \ldots, D\}$
- parameter subvectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$

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

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

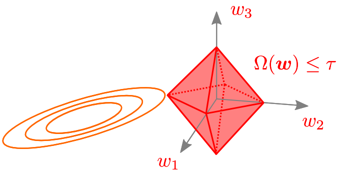
- Intuitively: the ℓ_1 **norm** of the ℓ_2 **norms**
- Technically, still a norm (called a *mixed* norm, denoted $\ell_{2,1}$)
- λ_m: prior weight for group G_m (different groups have different sizes)

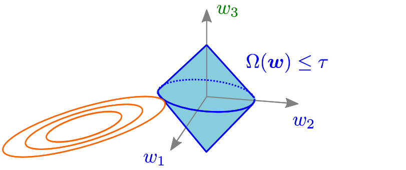
Lasso versus group-Lasso



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

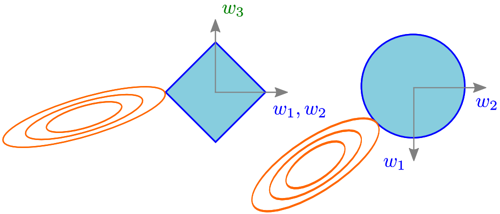
Lasso versus group-Lasso





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

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



Three Scenarios

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

Three Scenarios

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

Non-overlapping Groups

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

$\Rightarrow$ Each feature belongs to exactly one group

Non-overlapping Groups

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

$\Rightarrow$ Each feature belongs to exactly one group

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

Trivial choices of groups recover *unstructured* regularizers:

Non-overlapping Groups

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

$\Rightarrow$ Each feature belongs to exactly one group

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

Trivial choices of groups recover *unstructured* regularizers:

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

Non-overlapping Groups

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

$\Rightarrow$ Each feature belongs to exactly one group

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

Trivial choices of groups recover *unstructured* regularizers:

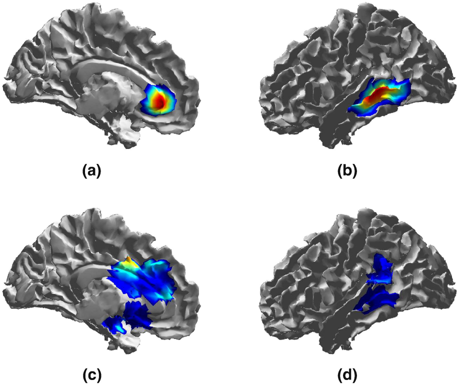
- ℓ_2-regularization: one large group $G_1 = \{1, \ldots, D\}$
- ℓ_1-regularization: D singleton groups $G_d = \{d\}$

Examples of non-trivial groups:

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

Example: Magnetoencephalographic (MEG) Reconstruction

Group: localized cortex area at localized time period (Bolstad et al., 2009)



Three Scenarios

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

Three Scenarios

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

Tree-Structured Groups

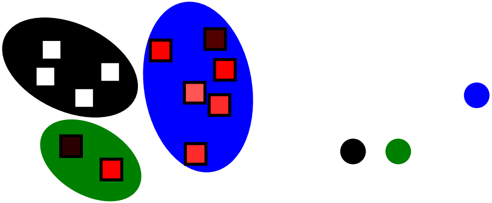
Assumption: if two groups overlap, one contains the other

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

Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

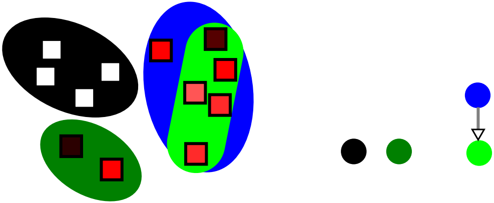
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

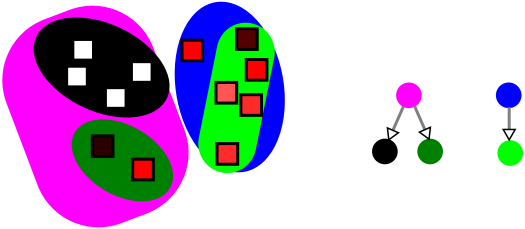
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

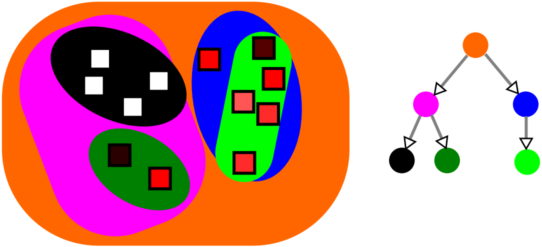
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

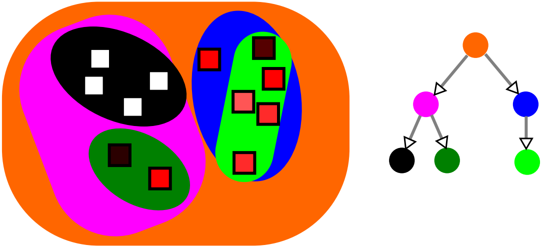
$\Rightarrow$ **hierarchical** structure (Kim and Xing, 2010; Mairal et al., 2010)



Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

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

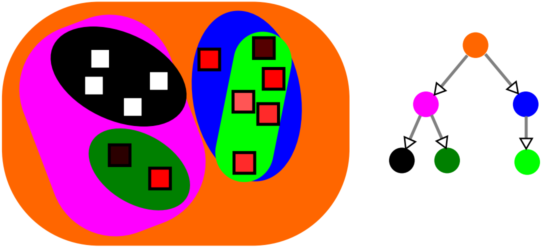


- What is the **sparsity pattern**?

Tree-Structured Groups

Assumption: if two groups overlap, one contains the other

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



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

Main messages:

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

Main messages:

- **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

Main messages:

- **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

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

Two kinds of optimization algorithms:

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

Outline

A Key Ingredient: Proximity Operators

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

A Key Ingredient: Proximity Operators

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

The **Ω-proximity operator** is the $\mathbb{R}^D \rightarrow \mathbb{R}^D$ map (Moreau, 1962)

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

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

A Key Ingredient: Proximity Operators

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

The **Ω-proximity operator** is the $\mathbb{R}^D \to \mathbb{R}^D$ map (Moreau, 1962)

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

...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**

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

A Key Ingredient: Proximity Operators

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

The **Ω-proximity operator** is the $\mathbb{R}^D \rightarrow \mathbb{R}^D$ map (Moreau, 1962)

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

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

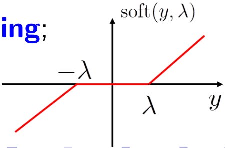
Classical examples:

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

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

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

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



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**

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

Proximity Operators (II)

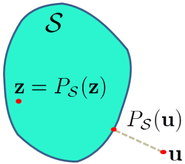
$$\boxed{prox_{\color{red}\Omega}(\mathbf{w}) = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|_2^2 + {\color{red}\Omega}(\mathbf{u})}$$

- ℓ_2 regularization, ${\color{red}\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_2}$: **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, ${\color{red}\Omega(\mathbf{w}) = \iota_{\mathcal{S}}(\mathbf{w})} = \begin{cases} 0 & \Leftarrow \quad \mathbf{w} \in \mathcal{S} \\ +\infty & \Leftarrow \quad \mathbf{w} \notin \mathcal{S} \end{cases}$

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



Euclidean projection

Proximity Operators (III)

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

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

Proximity Operators (III)

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

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

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

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

Proximity Operators (III)

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

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

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

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

- **Tree-structured:** (two groups are either disjoint or one contais the other) prox_{Ω} can be computed recursively (Jenatton et al., 2011).

Proximity Operators (III)

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

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

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

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

- **Tree-structured:** (two groups are either disjoint or one contais the other) prox_{Ω} can be computed recursively (Jenatton et al., 2011).
- **Arbitrary groups:**
  - For $\Omega_m(\mathbf{w}_m) = \|\mathbf{w}_m\|_2$: solved by convex smooth optimization (Yuan et al., 2011).
  - Sequential proximity steps (Martins et al., 2011a) (more later).

Outline

(Block-)Coordinate Descent

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

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

Projected Gradient

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

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 $\boxed{\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \Lambda(\mathbf{w})}$, address $\boxed{\min_{\mathbf{w}} \Lambda(\mathbf{w}) \ \text{ subject to } \Omega(\mathbf{w}) \leq \tau}$

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

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 method, which has been used in machine learning, including in NLP (Duchi et al., 2008; Quattoni et al., 2009).

Projected Gradient

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

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 method, which has been used in machine learning, including in 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 $\mathbf{w}'$:

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

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.

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.

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.

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

Proximal Gradient

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

Proximal Gradient

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

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

Proximal Gradient

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

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

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

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

Proximal Gradient

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

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

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

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

Proximal Gradient

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

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

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

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".

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

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

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

Can be derived with different tools:

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

Monotonicity and Convergence

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

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

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 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):

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

Convergence of objective value (Beck and Teboulle, 2009)

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

Convergence of iterates has not been shown.

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)

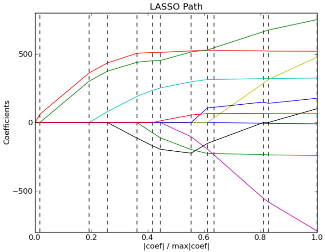
- "regularization path" $\hat{\mathbf{w}}(\lambda)$ is piecewise linear (Markowitz, 1952);
- the cusps can be identified in closed form;
- simply jump from one cusp to the next.

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.

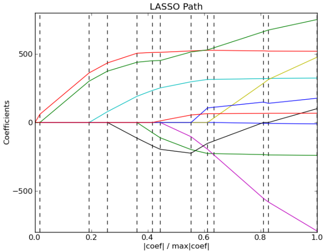


Least Angle Regression (LARS)

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)
- orthant-wise limited-memory quasi-Newton (OWL-QN) (Andrew and Gao, 2007; Gao et al., 2007); doesn't handle overlapping groups
- alternating direction method of multipliers (ADMM); handles overlapping groups (Afonso et al., 2010; Figueiredo and Bioucas-Dias, 2011).
- forward stagewise regression (Hastie et al., 2007);

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

Outline

Why Online?

1. Suitable for large datasets

Why Online?

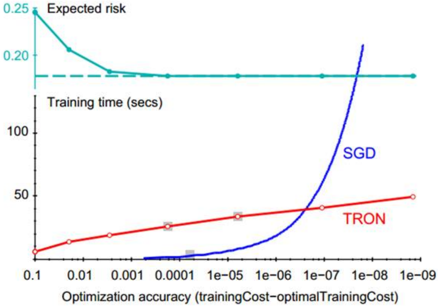
1. Suitable for large datasets
2. Suitable for structured prediction

Why Online?

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

Why Online?

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



Plain Stochastic (Sub-)Gradient Descent

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

Plain Stochastic (Sub-)Gradient Descent

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

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

SGD with Sparsity?

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

SGD with Sparsity?

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

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

SGD with Sparsity?

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

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

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

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

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

SGD with Sparsity?

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

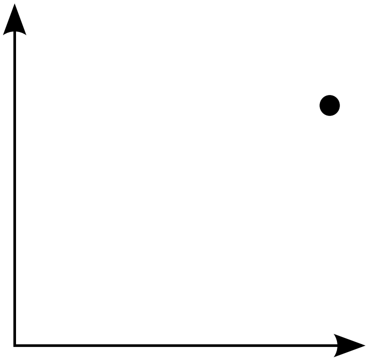
$$\mathbf{w} \;\leftarrow\; \underbrace{(1-\eta_t\lambda)\mathbf{w}}_{\text{scaling}} - \eta_t \tilde{\nabla} L(\mathbf{w}; x_t, y_t)$$

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

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

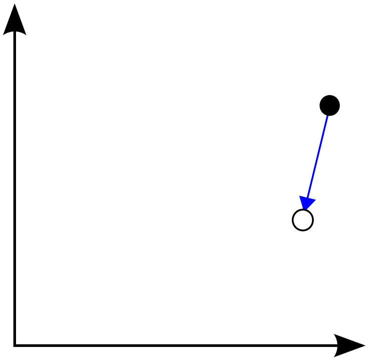
- **Problem: iterates are never sparse!**

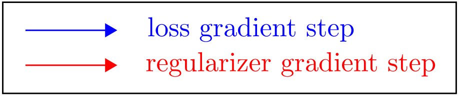
Plain SGD with ℓ_2-regularization



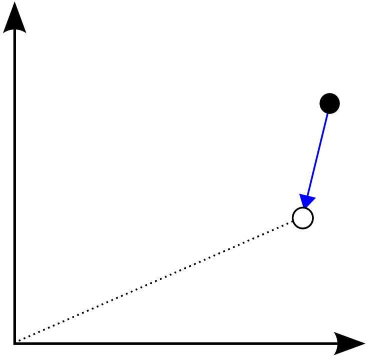
→	loss gradient step
→	regularizer gradient step

Plain SGD with ℓ_2-regularization



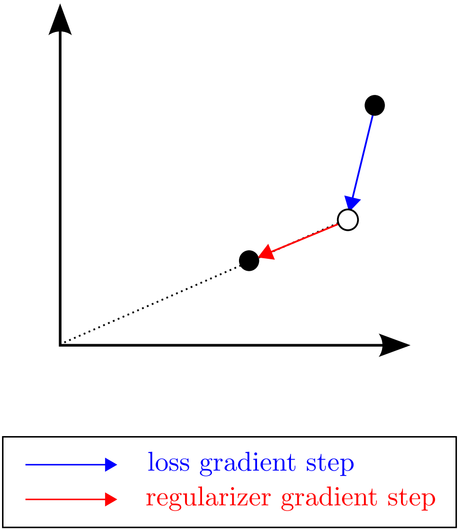


Plain SGD with ℓ_2-regularization

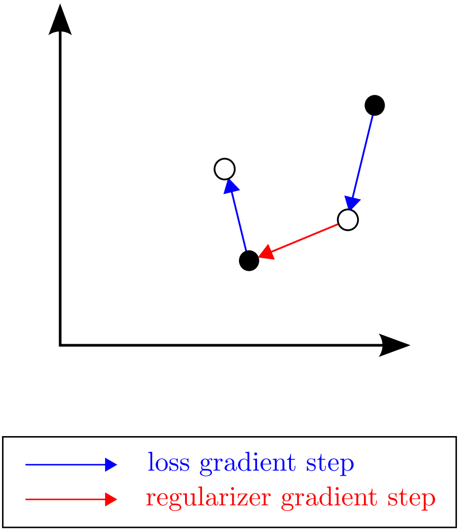


→	loss gradient step
→	regularizer gradient step

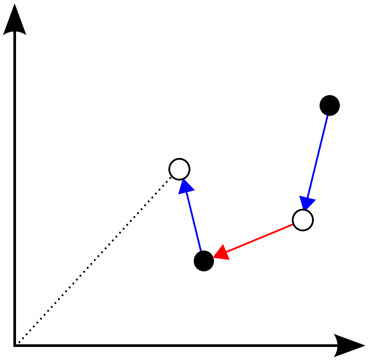
Plain SGD with ℓ_2-regularization

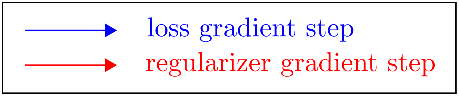


Plain SGD with ℓ_2-regularization

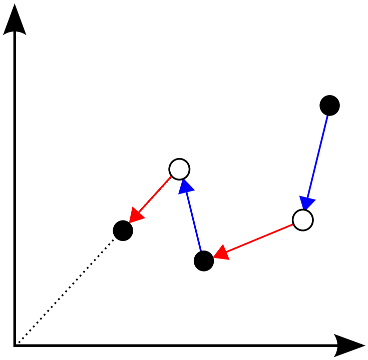


Plain SGD with ℓ_2-regularization



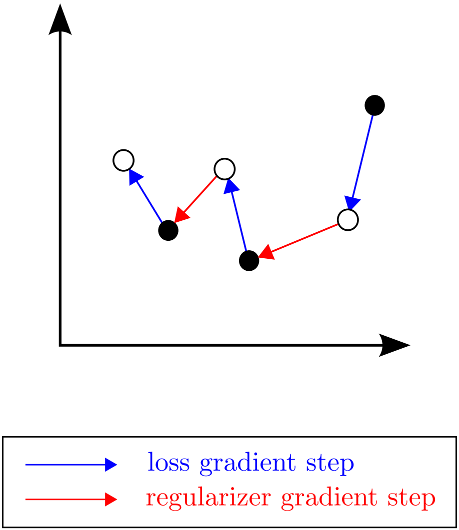


Plain SGD with ℓ_2-regularization

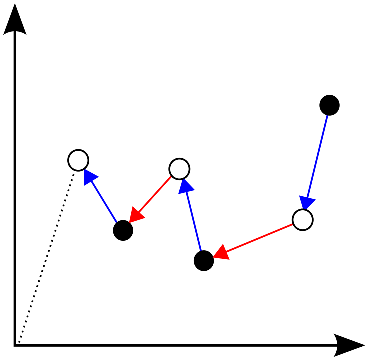


loss gradient step
regularizer gradient step

Plain SGD with ℓ_2-regularization

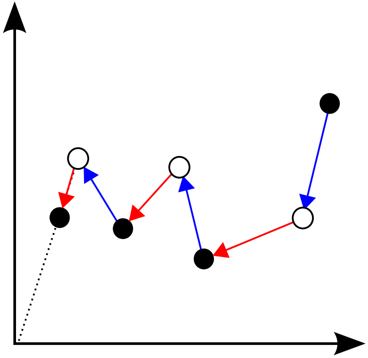


Plain SGD with ℓ_2-regularization



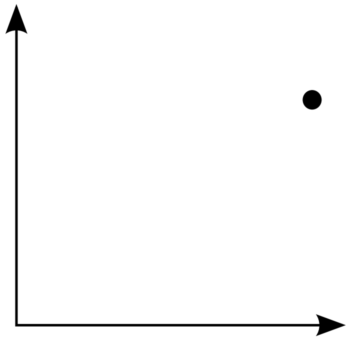
loss gradient step
regularizer gradient step

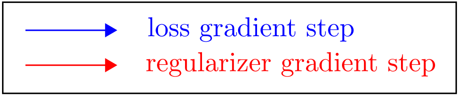
Plain SGD with ℓ_2-regularization



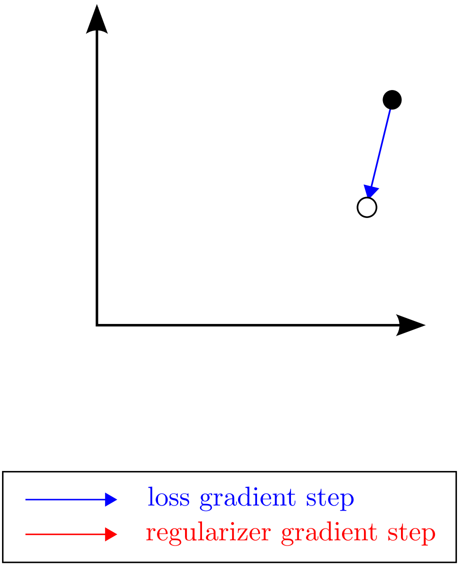
→	loss gradient step
→	regularizer gradient step

Plain SGD with ℓ_1-regularization

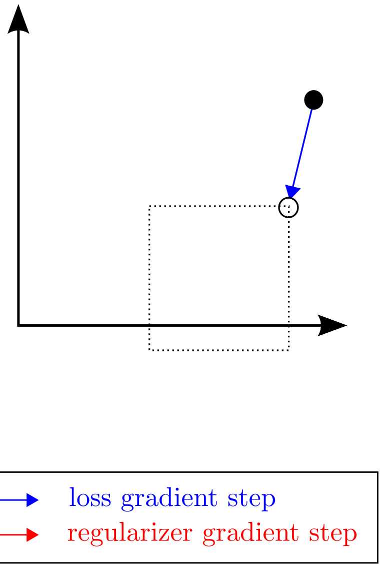




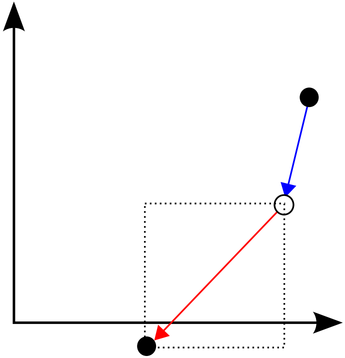
Plain SGD with ℓ_1-regularization

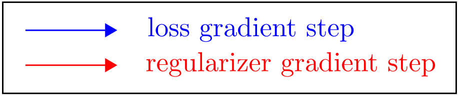


Plain SGD with ℓ_1-regularization

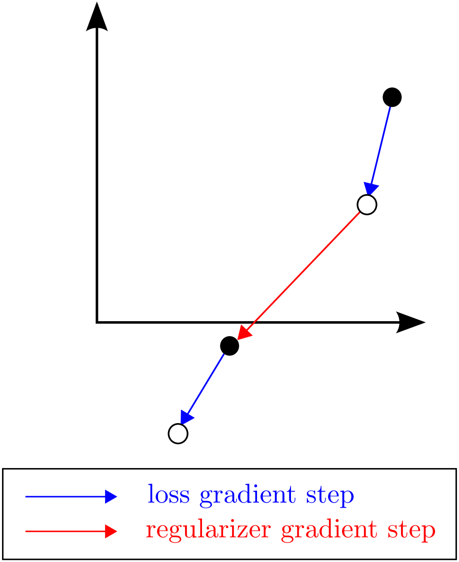


Plain SGD with ℓ_1-regularization

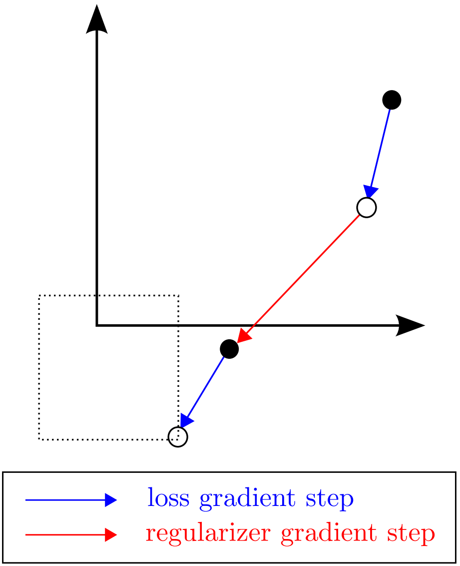




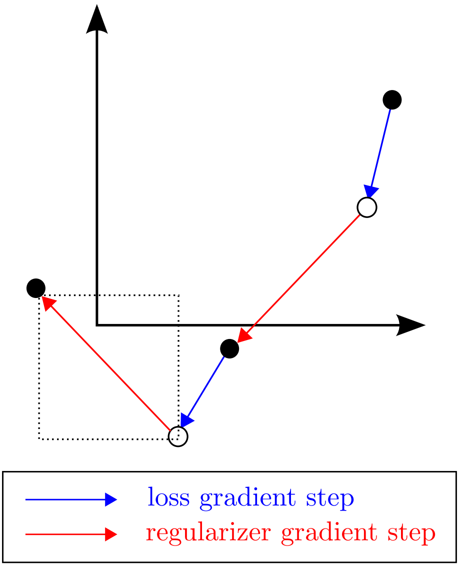
Plain SGD with ℓ_1-regularization



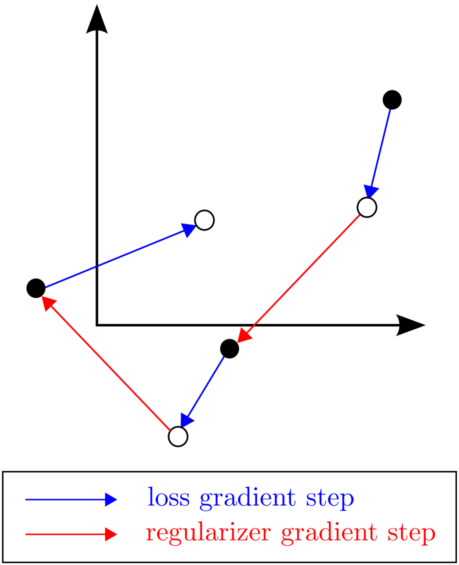
Plain SGD with ℓ_1-regularization



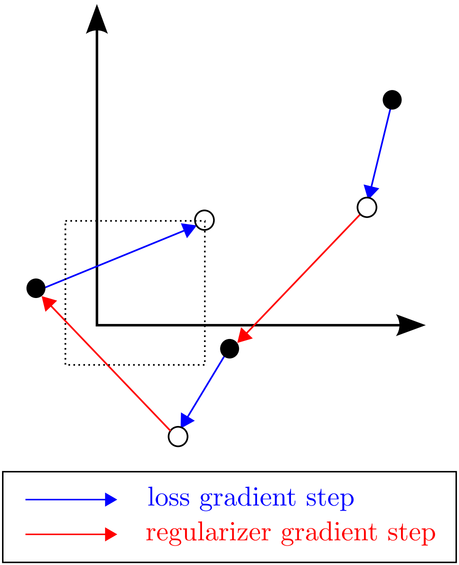
Plain SGD with ℓ_1-regularization



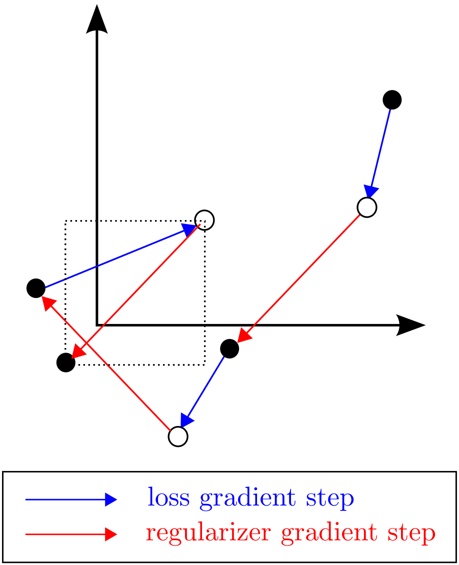
Plain SGD with ℓ_1-regularization



Plain SGD with ℓ_1-regularization



Plain SGD with ℓ_1-regularization

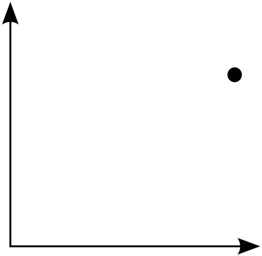


“Sparse” Online Algorithms

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

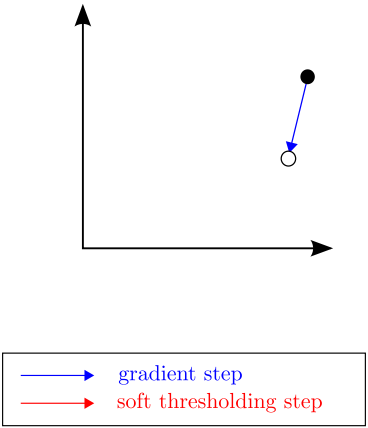
Truncated Gradient (Langford et al., 2009)

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



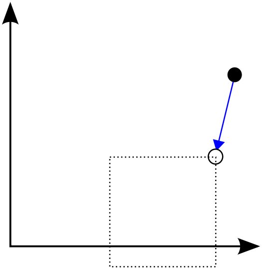
Truncated Gradient (Langford et al., 2009)

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



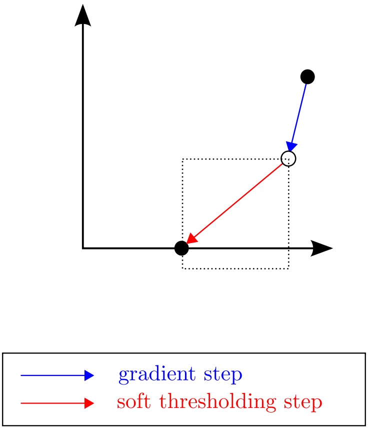
Truncated Gradient (Langford et al., 2009)

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



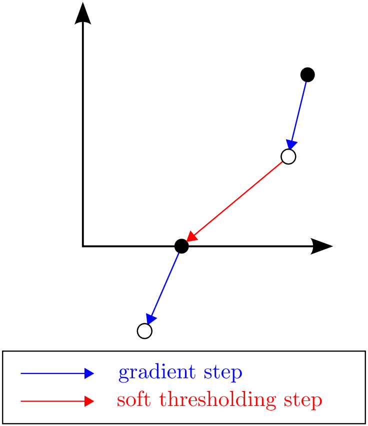
Truncated Gradient (Langford et al., 2009)

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



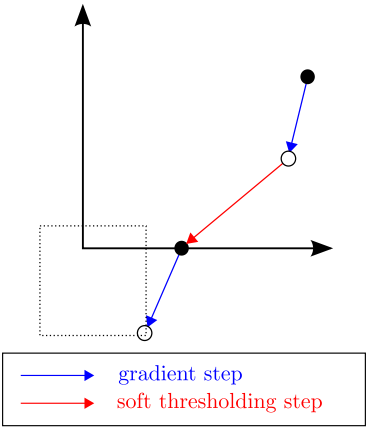
Truncated Gradient (Langford et al., 2009)

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



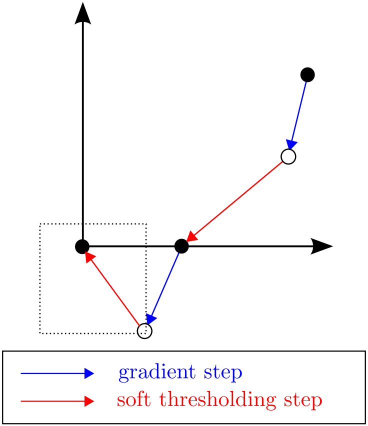
Truncated Gradient (Langford et al., 2009)

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



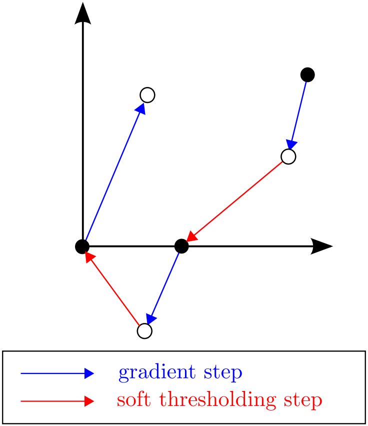
Truncated Gradient (Langford et al., 2009)

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



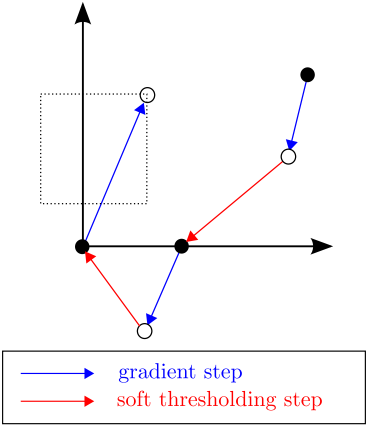
Truncated Gradient (Langford et al., 2009)

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



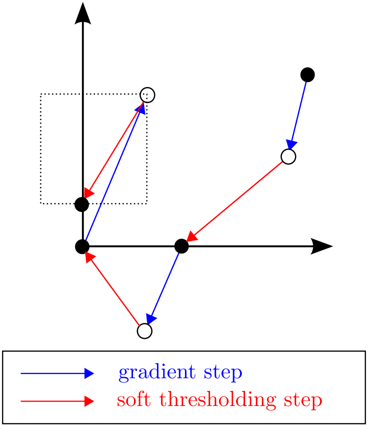
Truncated Gradient (Langford et al., 2009)

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



Truncated Gradient (Langford et al., 2009)

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



"Sparse" Online Algorithms

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

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

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:** iterates are usually not very sparse
- **converges to ϵ-accurate objective after $O(1/\epsilon^2)$ iterations**

What About Group Sparsity?

Online forward-backward splitting (Duchi and Singer, 2009) handles groups.

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

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

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

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

“Sparse” Online Algorithms

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

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

input: gravity sequence $(\sigma_t)_{t=1}^{T}$, stepsize sequence $(\eta_t)_{t=1}^{T}$
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  take training pair (x_t, y_t)
  gradient step: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla L(\theta; x_t, y_t)$
  sequential proximal steps:
  for $j = 1, 2, \ldots$ **do**
    $\mathbf{w} \leftarrow \operatorname{prox}_{\eta_t \sigma_t \Omega_j}(\mathbf{w})$
  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: $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla L(\theta; x_t, y_t)$
  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.
  Each proximal step is **linear** in the number of groups M.

Summary of Algorithms

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

Outline

Applications of Structured Sparsity in ML

Relatively few to date; we will focus on several recent NLP applications (Martins et al., 2011b):

- Phrase chunking
- Named entity recognition
- Dependency parsing

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 (group Lasso):

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

for M groups/templates.

Chunking

$x =$ "He reckons the current account deficit will narrow to only \$ 1.8 billion in September"

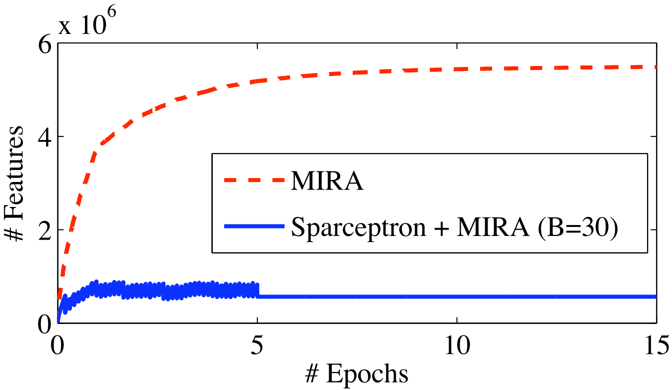
[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September]

- 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

Chunking



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 ML

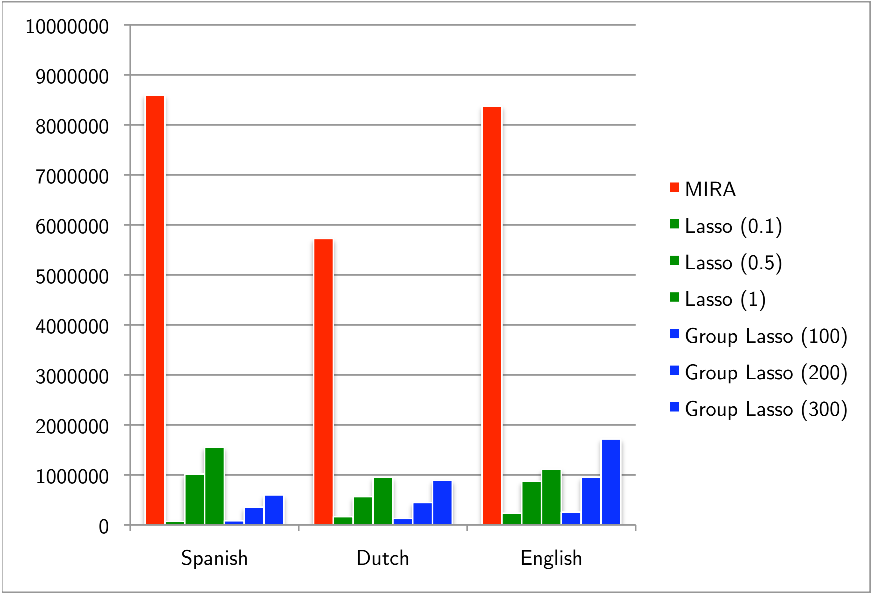
Relatively few to date; we will focus on several recent NLP applications (Martins et al., 2011b):

- Phrase chunking
- Named entity recognition
- Dependency parsing

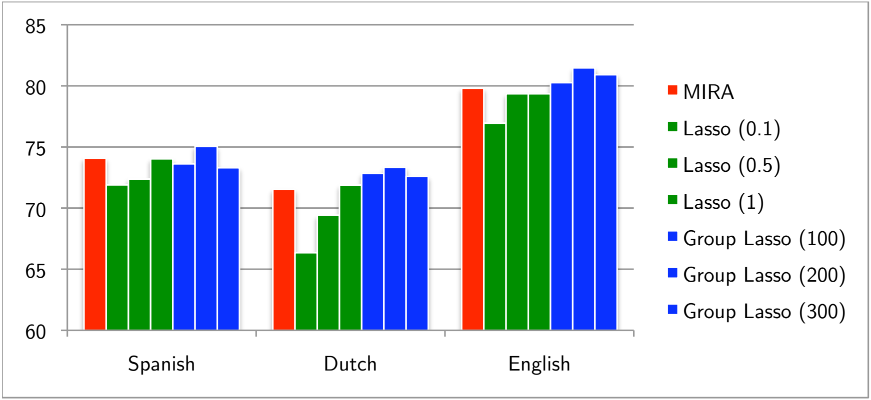
Named Entity Recognition

x	Only	France	and	Britain	backed	Fischler	's	proposal
	RB	NNP	CC	NNP	VBD	NNP	POS	NN
y = h_w(x)	O	I-LOC	O	I-LOC	O	I-PER	O	O

- 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 ML

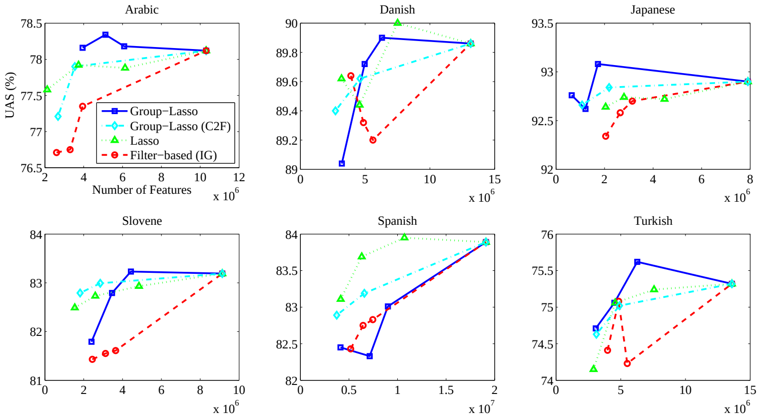
Relatively few to date; we will focus on several recent NLP applications (Martins et al., 2011b):

- Phrase chunking
- Named entity recognition
- Dependency parsing

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**.

Outline

Summary

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

Thank you!

- Questions?

Acknowledgments

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

References I

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

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

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

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

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

Bertero, M., Poggio, T., and Torre, V. (1988). Ill-posed problems in early vision. *Proceedings of the IEEE*, 76:869–889.

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

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

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

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.

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

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

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.

References II

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

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

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

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

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

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

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

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

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

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.

Hoerl, A. (1962). Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59.

Hoerl, A. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42:80–86.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2011). Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334.

References III

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

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

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*.

Moreau, J. (1962). Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sér. A Math*, 255:2897–2899.

References IV

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.

Ng, A. (2004). Feature selection, *l*1 vs. *l*2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning*.

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.

Poggio, T., Torre, V., and Koch, C. (1985). Computational vision and regularization theory. *Nature*, 317:314–319.

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

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

Sang, E. and 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.

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 V

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

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

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

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

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

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.

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

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.

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.