A practical tour of optimization algorithms for the Lasso

Alexandre Gramfort
alexandre.gramfort@inria.fr

Inria, Parietal Team
Université Paris-Saclay

Huawei - Apr. 2017

Outline

- What is the Lasso
- Lasso with an orthogonal design
- From projected gradient to proximal gradient
- Optimality conditions and subgradients (LARS algo.)
- Coordinate descent algorithm

… *with some demos*

Lasso

$$x^* \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|b - Ax\|^2 + \lambda\|x\|_1$$

with $A \in \mathbb{R}^{n \times p}$ $\lambda > 0$ $\|x\|_1 = \sum_{i=1}^{p} |x_i|$

- Commonly attributed to [Tibshirani 96] (> 19000 citations)
- Also known as Basis Pursuit Denoising [Chen 95] (> 9000 c.)
- Convex way of promoting sparsity in high dimensional regression / inverse problems.
- Can lead to statistical guarantees even if $n \approx \log(p)$

Algorithm 0

Using CVX Toolbox

```
n = 10;
A = randn(n/2,n);
b = randn(n/2,1);
gamma = 1;

cvx_begin
  variable x(n)
  dual variable y
  minimize(0.5*norm(A*x - b, 2) + gamma * norm(x, 1))
cvx_end
```

http://cvxr.com/cvx/

Algorithm 1

Rewrite: $x_i = x_i^+ + x_i^- = \max(x_i, 0) + \min(x_i, 0)$

$$|x_i| = x_i^+ - x_i^- = \max(x_i, 0) + \max(-x_i, 0)$$

$$\|x\|_1 = x^+ - x^-$$

Leads to:

$$z^* \in \operatorname*{argmin}_{z \in \mathbb{R}_+^{2p}} \frac{1}{2} \|b - [A, -A]z\|^2 + \lambda \sum_i z_i$$

- This is a simple smooth convex optimization problem with positivity constraints (convex constraints)

Gradient Descent

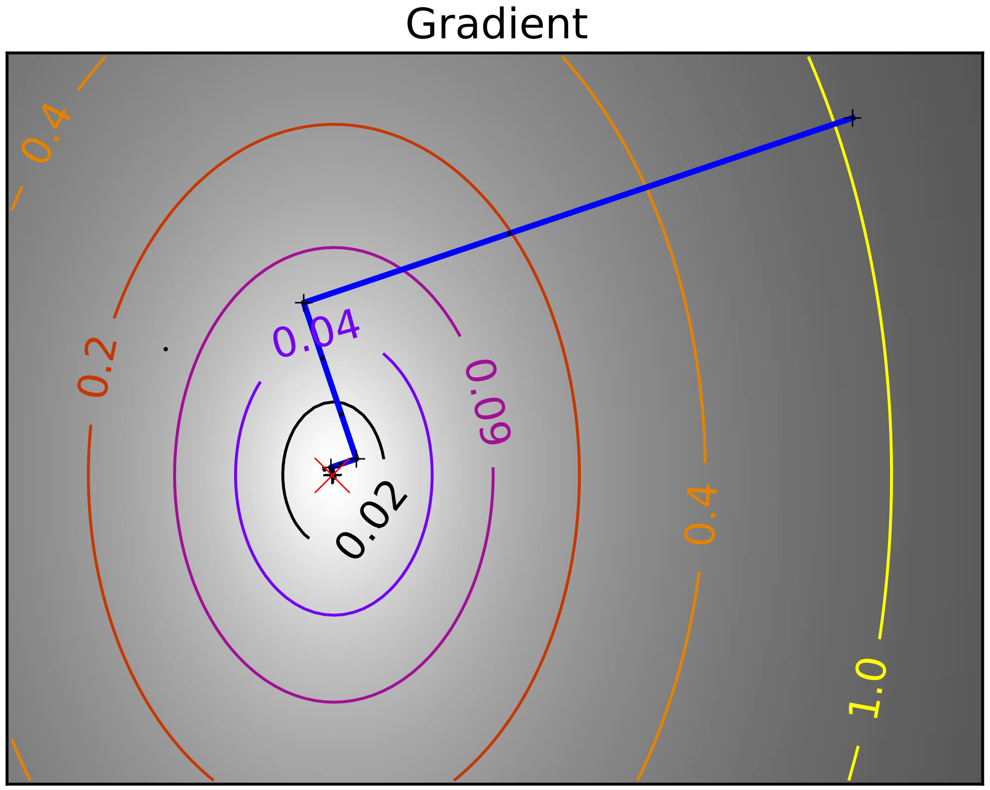
$$\min_{x \in \mathbb{R}^p} f(x)$$

With f smooth with L-Lipschitz gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

Gradient descent reads:

$$x^{k+1} = x^k - \frac{1}{L}\nabla f(x^k)$$



Projected gradient Descent

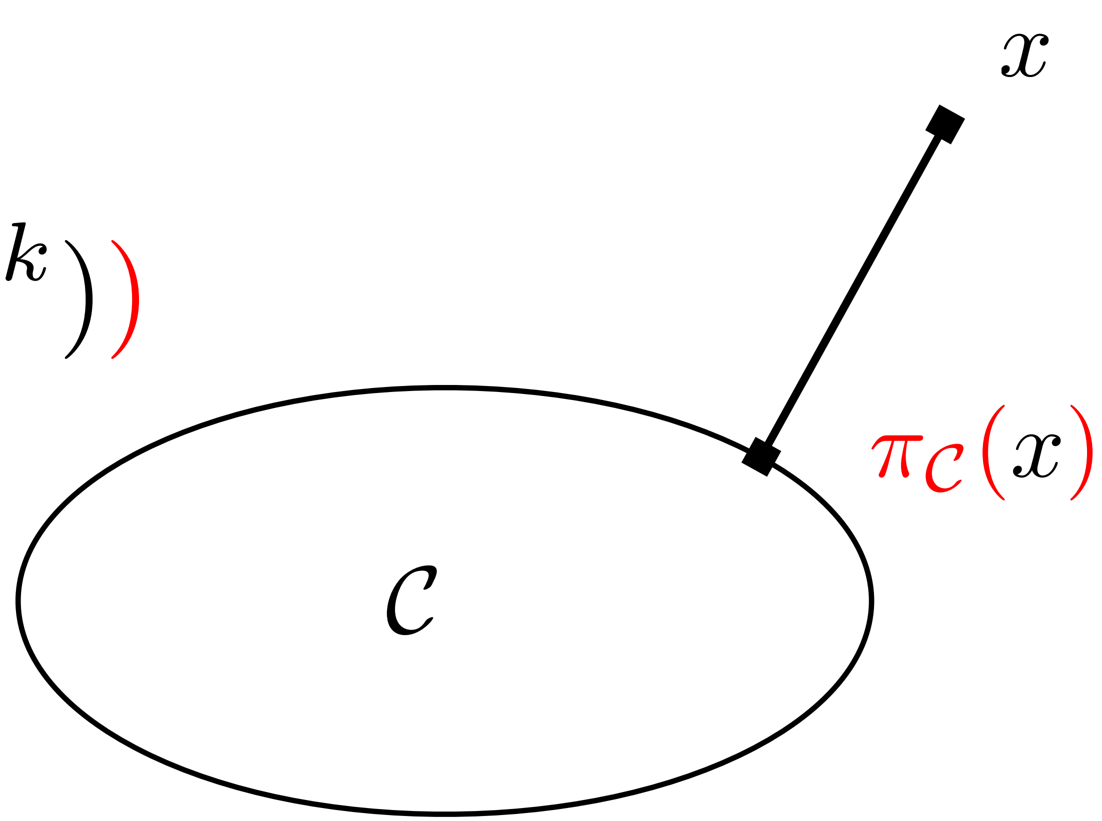
$$\min_{x \in \mathcal{C} \subset \mathbb{R}^p} f(x)$$

With $\mathcal{C}$ a convex set and f smooth with L-Lipschitz gradient

projected gradient reads:

$$x^{k+1} = \pi_{\mathcal{C}}(x^k - \frac{1}{L} \nabla f(x^k))$$

Orthogonal projection on $\mathcal{C}$



demo_grad_proj.ipynb

What if A is orthogonal?

- Let's assume we have a square orthogonal design matrix

$$A^\top A = AA^\top = I_p$$

One has: $\|b - Ax\|^2 = \|A^\top b - x\|^2$

So the Lasso boils down to minimizing:

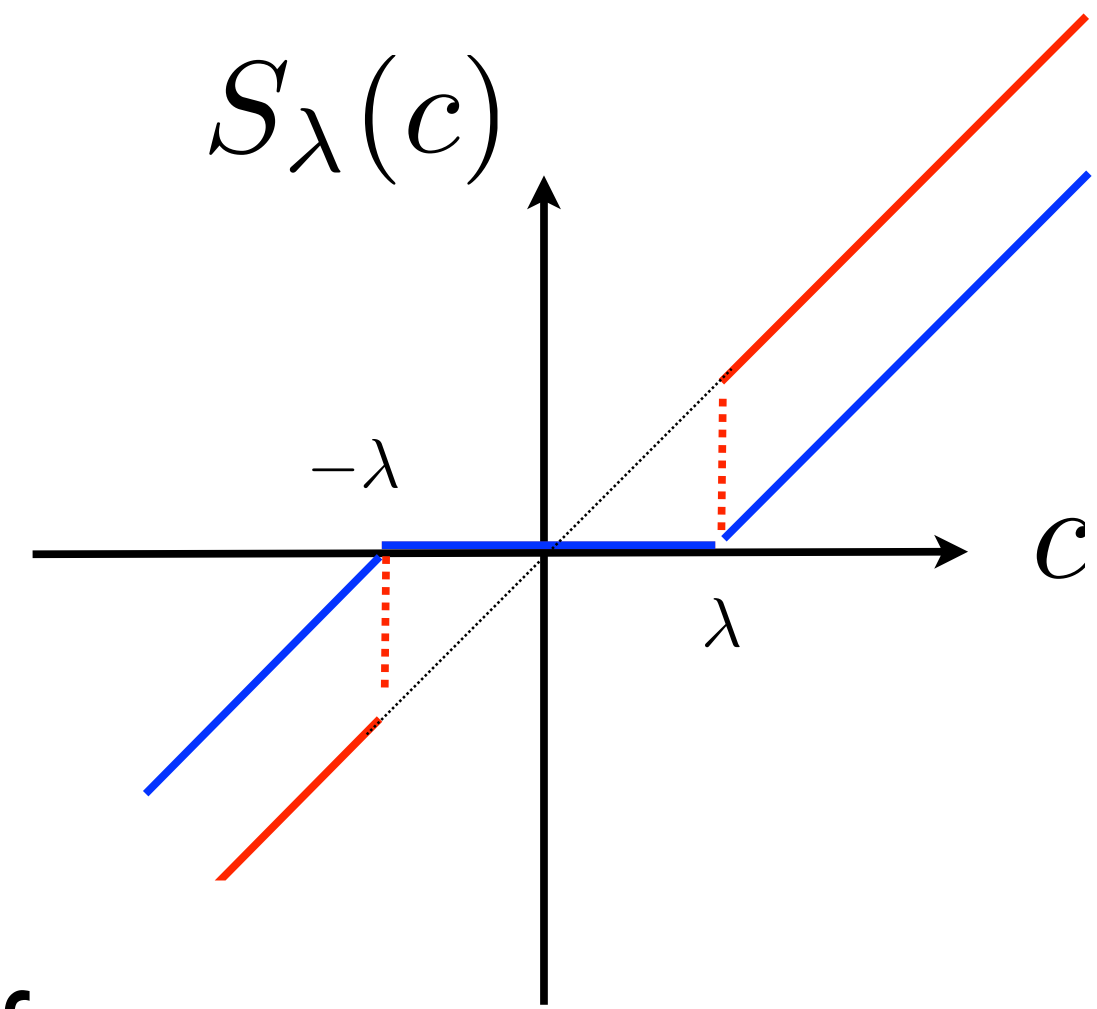
$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^p_+} \frac{1}{2}\|A^\top b - x\|^2 + \lambda \|x\|_1$$

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^p_+} \sum_{i=1}^{p} \left(\frac{1}{2}((A^\top b)_i - x_i)^2 + \lambda |x_i| \right) \quad \text{(p 1d problems)}$$

$$x^* \triangleq \operatorname{prox}_{\lambda \|\cdot\|_1}(A^\top b) \quad \text{(Definition of the proximal operator)}$$

Proximal operator of L1 norm

The soft-thresholding: $c \to \mathrm{sign}(c)(|c| - \lambda)_+$



is the solution of

$$\min_x \frac{1}{2}(c - x)^2 + \lambda \cdot |x|$$

Algorithm with A orthogonal

```
c = A.T.dot(b)
x_star = np.sign(c) * np.maximum(np.abs(c) - lambd, 0.)
```

python

What if A is NOT orthogonal?

Let us define: $f(x) = \frac{1}{2}\|b - Ax\|^2$

Leads to: $\nabla f(x) = -A^\top (b - Ax)$

The Lipschitz constant of the gradient: $L = \|A^\top A\|_2$

Quadratic upper bound of f at the previous iterate:

$$x^{k+1} = \underset{x \in \mathbb{R}^p_+}{\operatorname{argmin}} f(x^k) + (x - x^k)^\top \nabla f(x^k) + \frac{L}{2}\|x - x^k\|^2 + \lambda \|x\|_1$$

$\Leftrightarrow$

$$x^{k+1} = \underset{x \in \mathbb{R}^p_+}{\operatorname{argmin}} \frac{1}{2}\|x - (x^k - \frac{1}{L}\nabla f(x^k))\|^2 + \frac{\lambda}{L}\|x\|_1$$

Algorithm 2: Proximal gradient descent

That we can rewrite:

$$x^{k+1} = \underset{x \in \mathbb{R}^p_+}{\operatorname{argmin}} \frac{1}{2} \|x - (x^k - \frac{1}{L} \nabla f(x^k))\|^2 + \frac{\lambda}{L} \|x\|_1$$

$$= \operatorname{prox}_{\frac{\lambda}{L} \|\cdot\|_1} (x^k - \frac{1}{L} \nabla f(x^k))$$

[Daubechies et al. 2004, Combettes et al. 2005]

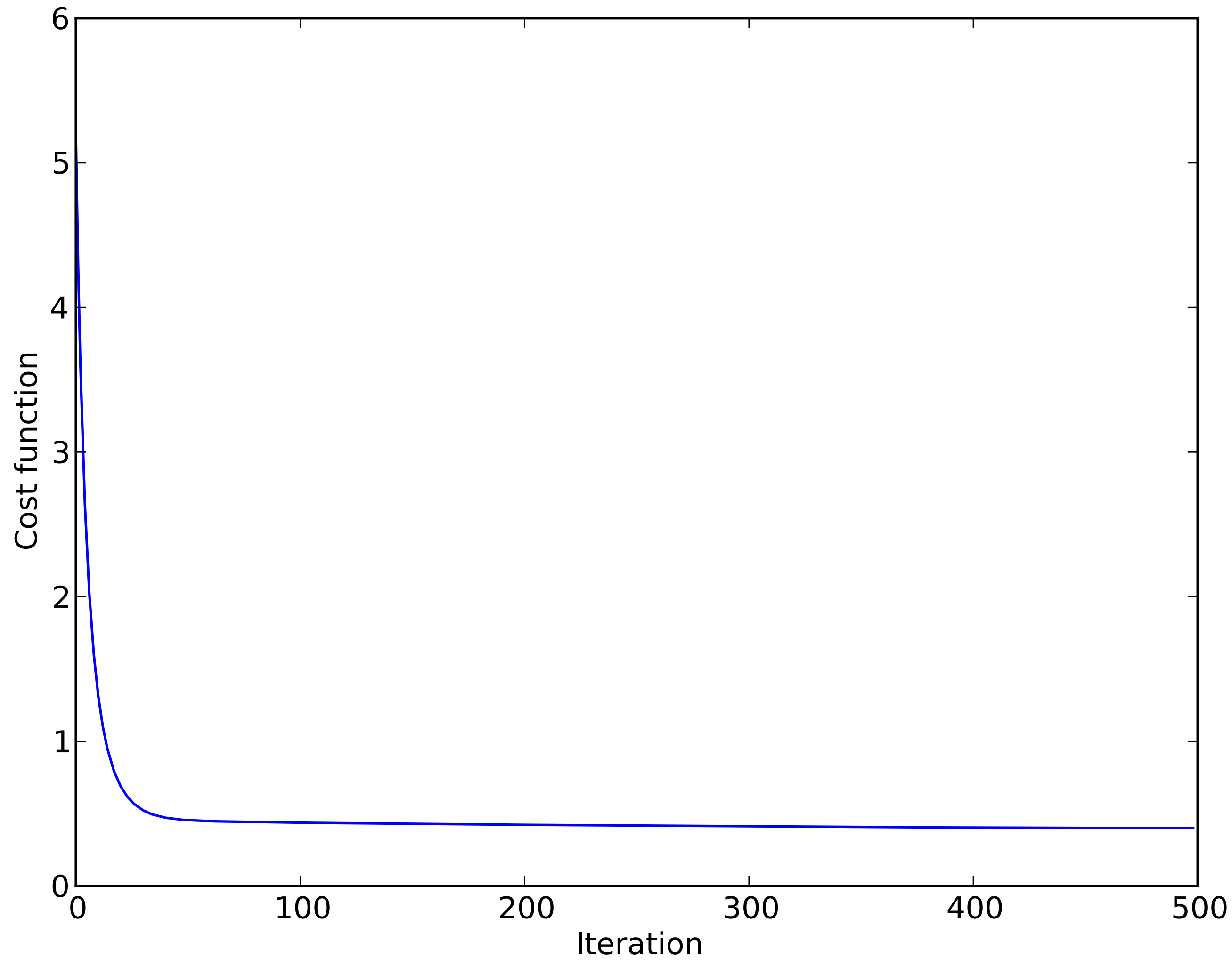
Remark: If f is not strongly convex $f(x^k) - f(x^*) = \mathcal{O}\left(\frac{1}{k}\right)$

Very far from an exponential rate of GD with strong convexity

Remark: There exist so called "accelerated" methods known as FISTA, Nesterov acceleration…

Proximal gradient

```
alpha = 0.1 # Lambda parameter
L = linalg.norm(A)**2
x = np.zeros(A.shape[1])
for i in range(max_iter):
    x += (1. / L) * np.dot(A.T, b - np.dot(A, x))
    x = np.sign(X) * np.maximum(np.abs(X) - (alpha / L), 0)
```



demo_grad_proximal.ipynb

Pros of proximal gradient

- **First order method** (only requires to compute gradients)
- Algorithms **scalable even if p is large** (needs to store A in memory)
- Great if A is an implicit linear operator (Fourier, Wavelet, MDCT, etc.) as dot products have some logarithmic complexities.

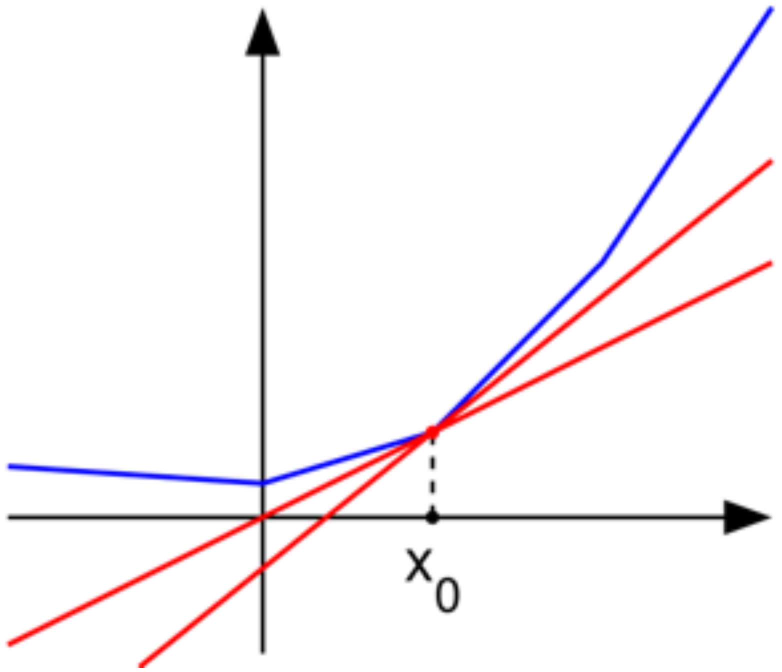
Subgradient and subdifferential

The subdifferential of f at x_0 is:

$$\partial f(x_0) = \{g \in \mathbb{R}^n / f(x) - f(x_0) \geq g^T(x - x_0), \forall x \in \mathbb{R}^n\}$$

Properties

- The subdifferential is a convex set
- x_0 is a minimizer of f if $0 \in \partial f(x_0)$



Exercise: What is $\partial |\cdot|(0) = ?$

Path of solutions

Lemma [Fuchs 97] **: Let x^* be a solution of the Lasso**

$$x^* \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|b - Ax\|^2 + \lambda\|x\|_1$$

Let the support $I = \{i \text{ s.t. } x_i \neq 0\}$

Then:

$$A_I^\top(Ax^* - b) + \lambda \operatorname{sign}(x_I^*) = 0$$

$$\|A_{I^c}^\top(Ax^* - b)\|_\infty \leq \lambda$$

And also:

$$\boxed{x_I^* = (A_I^\top A_I)^{-1}(A_I^\top b - \lambda \operatorname{sign}(x_I^*))}$$

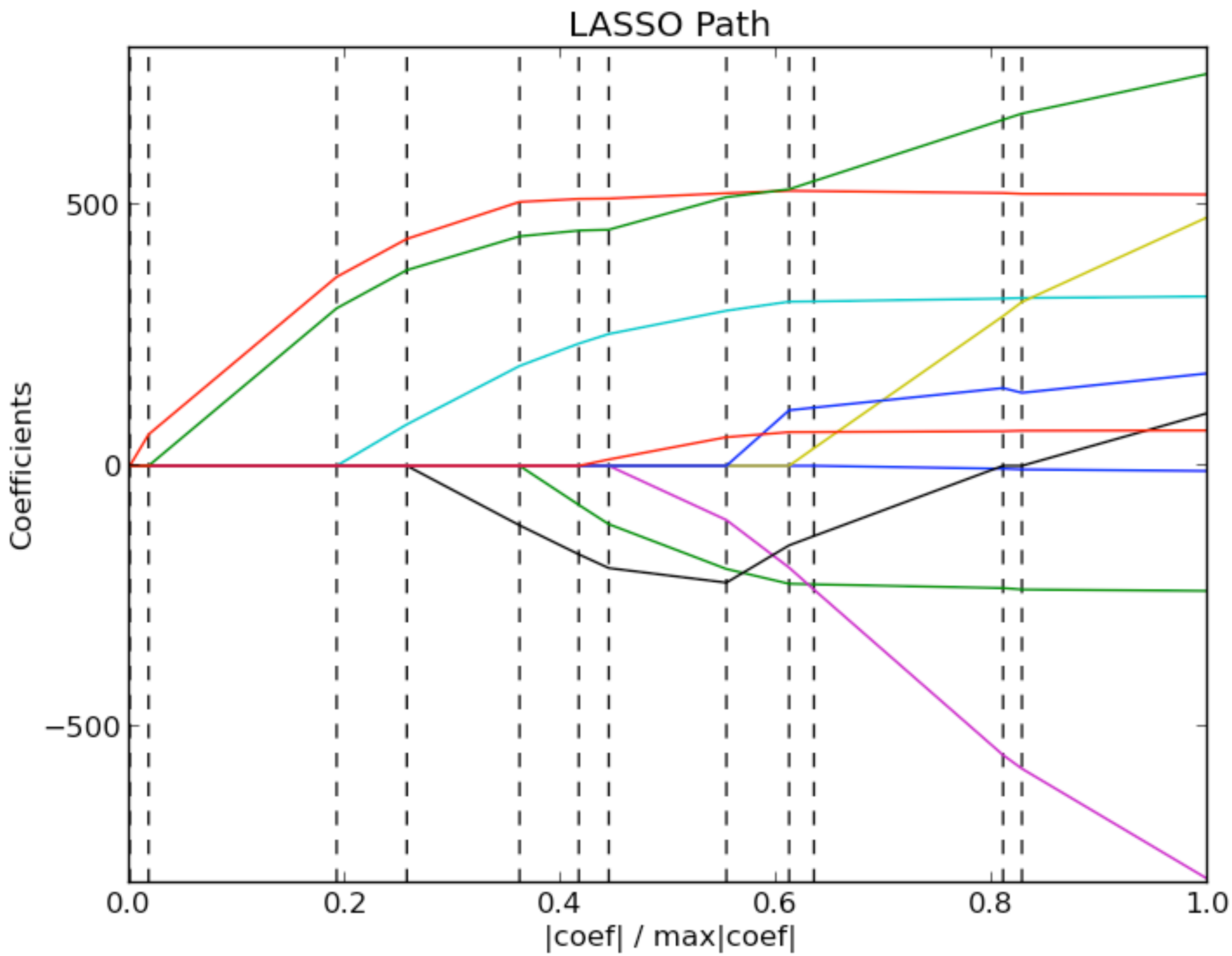
Algorithm 3: Homotopy and LARS

The idea is to compute the full path of solution noticing that for a given sparsity / sign pattern the solution if affine.

$$\boxed{x_I^* = (A_I^\top A_I)^{-1}(A_I^\top b - \lambda \operatorname{sign}(x_I^*))}$$

The LARS algorithm [Osborne 2000, Efron et al. 2004] consists if finding the breakpoints along the path.

Lasso path with LARS algorithm



Pros/Cons of LARS

Pros:

- Gives the full path of solution
- Fast with support is small and one can compute Gram matrix

Cons:

- Scales with the size of the support
- Hard to make it numerically stable
- One can have many many breakpoints [Mairal et al. 2012]

demo_lasso_lars.ipynb

Coordinate descent (CD)

Limitation of proximal gradient descent:

$$x^{k+1} = \text{prox}_{\frac{\lambda}{L}\|\cdot\|_1}(x^k - \frac{1}{L}\nabla f(x^k))$$

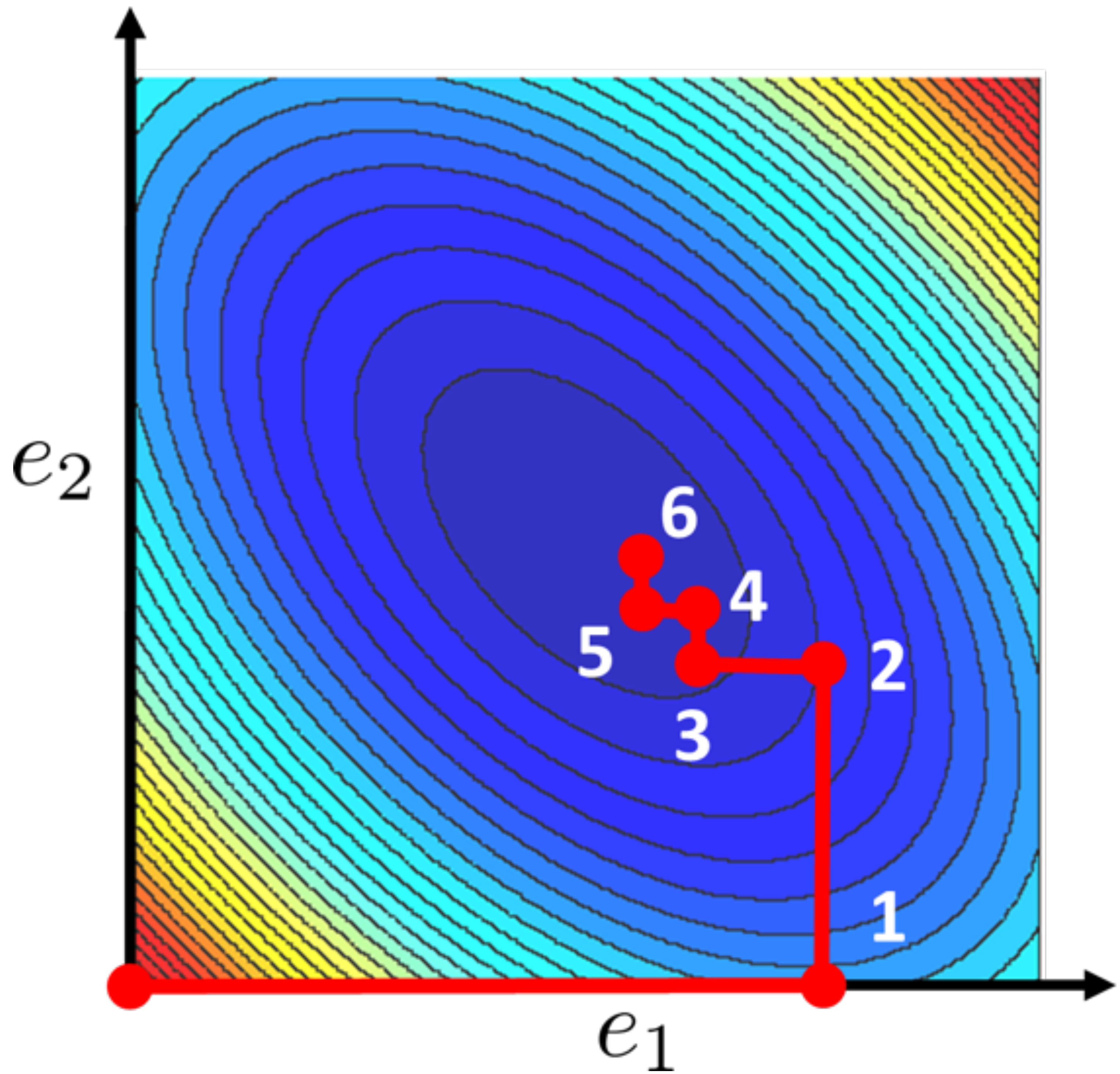
if L is big we make tiny steps !

The idea of coordinate descent (CD) is to update one coefficient at a time (also known as univariate *relaxation methods* in optimization or Gauss Seidel's method).

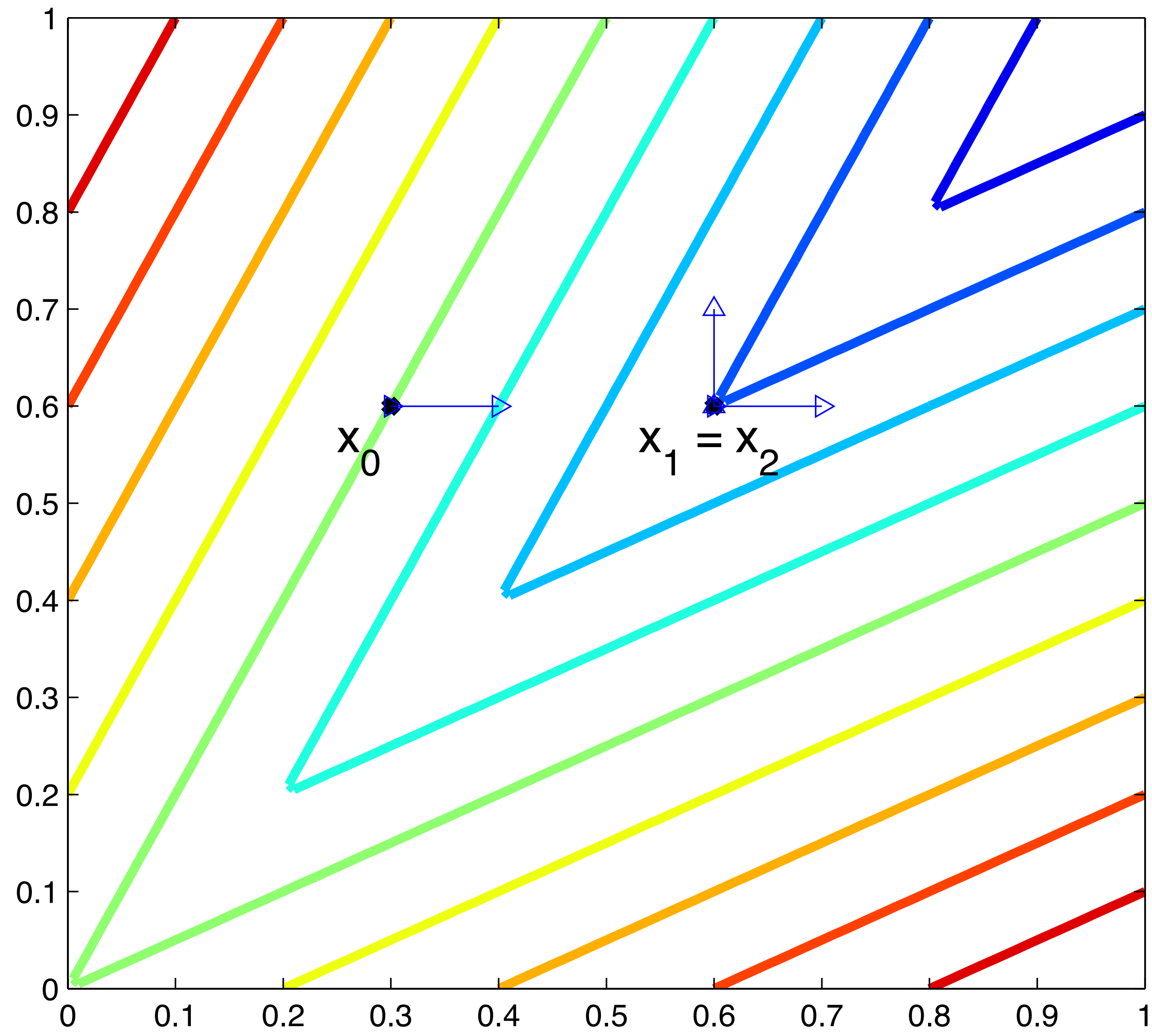
Hope: make bigger steps.

Spoiler: It is the state of the art in machine learning problems (cf. GLMNET R package, scikit-learn) [Friedman et al. 2009]

Coordinate descent (CD)



Coordinate descent (CD)



Warning: It does not always work !

Algorithm 4: Coordinate descent (CD)

Since the regularization $\|x\|_1 = \sum_{i=1}^{p} |x_i|$

is separable function CD works for the Lasso [Tseng 2001]

Proximal coordinate descent algorithm works:

$$
\begin{aligned}
&\text{for } k = 1 \ldots K \\
&\quad i = (k \bmod p) + 1 \\
&\quad x_i^{k+1} = \text{prox}_{\frac{\lambda}{L_i}}(x_i^k - \frac{1}{L_i}(\nabla f(x^k))_i)
\end{aligned}
$$

$L_i \ll L$ we make bigger steps !

Algorithm 4: Coordinate descent (CD)

- Their exist many “tricks” to make CD fast for the Lasso
- Lazy update of the residuals
- Pre-computation of certain dot products
- Active set methods
- Screening rules
- More in the next talk…

Conclusion

- What is the Lasso
- Lasso with an orthogonal design
- From projected gradient to proximal gradient
- Optimality conditions and subgradients (LARS algo.)
- Coordinate descent algorithm

Contact

http://alexandre.gramfort.net

GitHub : @agramfort

Twitter : @agramfort