Coordinate descent

Alexandre Gramfort, Mathurin Massias

first.last@inria.fr

Master 2 Data Science, Univ. Paris Saclay
Optimisation for Data Science
course based on notes from Olivier Fercoq.

Table of Contents

Why coordinate descent for datascience?

You have seen 1st order methods:

- gradient descent
- proximal gradient descent
- accelerated gradient descent

You'll also see with Pierre

- Newton methods
- quasi-Newton methods

Coordinate descent (CD) has received a lot of attention in ML/stats over the last 10 years. It's state-of-the-art techniques on a number of learning problems, as CD applies in this settings (not as general as gradient descent). It's what R GLMNET package and Scikit-Learn Lasso / Elastic-Net / LinearSVC estimators use.

Coordinate wise optimization

We work in finite dimension $\mathbb{R}^n$ (think n parameters to optimize)

Coordinate descent is **extremely simple**

Idea: minimize one coordinate at a time (keeping the others fixed)

Question: Given convex, differentiable $f : \mathbb{R}^n \to \mathbb{R}$, if we are at a point $\mathbf{x}$ such that $f(\cdot)$ is minimized along each coordinate axis, have we found a global minimizer?

i.e., does $f(\mathbf{x} + d\mathbf{e}_i) \geq f(x)\ \forall d \in \mathbb{R},\ \forall i \Rightarrow f(\mathbf{x}) = \min_{\mathbf{z}} f(\mathbf{z})$?

where $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0) \in \mathbb{R}^n$ is the i-th canonical basis vector.

Coordinate wise optimization

$f(\mathbf{x} + d\mathbf{e}_i) \geq f(\mathbf{x}), \forall d \in \mathbb{R}$ implies that

$$\frac{\partial f}{\partial x^{(i)}}(\mathbf{x}) = 0$$

which implies

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x^{(1)}}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x^{(n)}}(\mathbf{x}) \right) = \mathbf{0}_n$$

OK for f smooth and convex !

Table of Contents

Exact coordinate descent

Objective: $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$

Initialisation: $\mathbf{x}_0 = (x_0^{(1)}, \ldots, x_0^{(n)})$.

Algorithm: At iteration t:

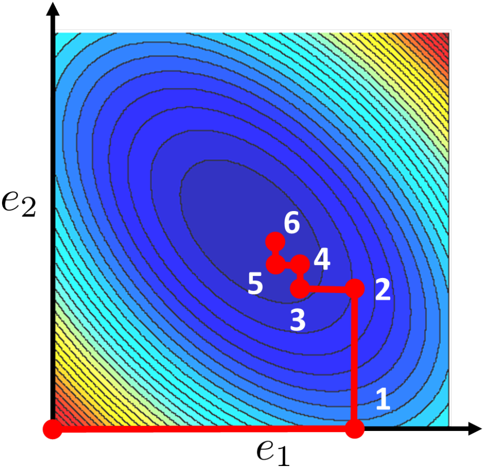
$$\text{Choose } j = (t \mod n) + 1 \quad \text{(cyclic rule)}$$

$$\begin{cases} x_{t+1}^{(i)} = x_t^{(i)} & \text{if } i \neq j \\ x_{t+1}^{(i)} = \arg\min_{z\in\mathbb{R}} f(x_t^{(1)}, \ldots, x_t^{(j-1)}, z, x_t^{(j+1)}, \ldots, x_t^{(n)}) & \text{if } i = j \end{cases}$$

Note: The order of cycle through coordinates is arbitrary, can use any permutation of $1, 2, \ldots, n$.

Note: We just have to solve 1D problems, but lots of them...

Example



Coordinate descent on a 2D problem

Example: Linear regression/OLS

Let $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$, where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the design matrix with columns $\mathbf{a}_1, \ldots, \mathbf{a}_n$ (one per feature)

Minimizing over $x^{(i)}$, with all $x^{(j)}$, $j \neq i$ fixed, yields:

$$0 = \nabla_i f(\mathbf{x}) = \mathbf{a}_i^\top (\mathbf{A}\mathbf{x} - \mathbf{y}) = \mathbf{a}_i^\top (\mathbf{a}_i x^{(i)} + \mathbf{A}_{-i}\mathbf{x}^{(-i)} - \mathbf{y})$$

i.e., we take:

$$x^{(i)} = \frac{\mathbf{a}_i^\top (\mathbf{y} - \mathbf{A}_{-i}\mathbf{x}^{(-\mathbf{i})})}{\mathbf{a}_i^\top \mathbf{a}_i}$$

Repeat these update by cycling over coordinates
$\rightarrow$ notebook

Example: Linear regression/OLS

Note that doing:

$$x^{(i)} \leftarrow \frac{\mathbf{a}_i^\top(\mathbf{y} - \mathbf{A}_{-i}\mathbf{x}^{(-\mathbf{i})})}{\mathbf{a}_i^\top \mathbf{a}_i}$$

is equivalent to:

$$x^{(i)} \leftarrow x^{(i)} + \frac{\mathbf{a}_i^\top \mathbf{r}}{\mathbf{a}_i^\top \mathbf{a}_i}$$

where $\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x}$ is the current *residual*. If current $\mathbf{r}$ is available the cost of an update is $O(m)$. Updating $\mathbf{r}$ is also $O(m)$ so full pass/epoch on coordinates is $O(mn)$, as for gradient descent.

Convergence of exact coordinate descent

Proposition (Warga (1963))

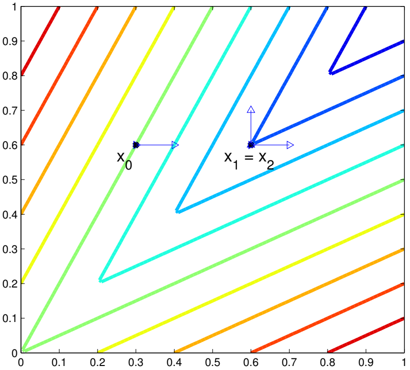
Assume that

- f *is continuously differentiable*
- f *is strictly convex*
- *there exists* $x_* \in \arg\min_{x \in \mathcal{X}} f(x)$

then the exact coordinate descent method converges to x_*.

Counter-example: convex nonsmooth

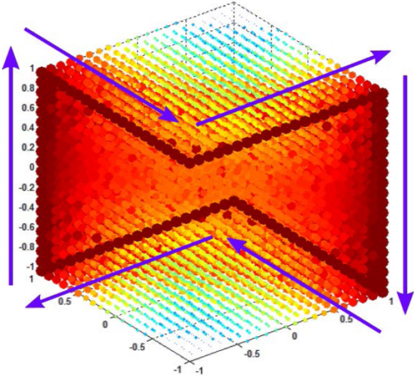
What if f is convex but non-smooth?



$$f(x^{(1)}, x^{(2)}) = |x^{(1)} - x^{(2)}| - \min(x^{(1)}, x^{(2)})$$

Counter-example: smooth nonconvex

What is f is smooth but non-convex? (Example due to Powell)



$$f(x^{(1)}, x^{(2)}, x^{(3)}) = -(x^{(1)}x^{(2)} + x^{(2)}x^{(3)} + x^{(3)}x^{(1)}) + \sum_{i=1}^{3} \max(0, |x^{(i)}| - 1)^2$$

Table of Contents

Motivation

- A 1D optimisation problem to solve at each iteration: This may be expensive
- We may solve it approximately since we've got plenty of iterations left
- We will do one single gradient step in the 1D problem

Coordinate gradient descent

Parameters: $\gamma_1, \ldots, \gamma_n > 0$

Algorithm:

Choose $i_{t+1} \in \{1, \ldots, n\}$

$$\begin{cases} x_{t+1}^{(i)} = x_t^{(i)} - \gamma_i \nabla_i f(\mathbf{x}_t) & \text{if } i = i_{t+1} \\ x_{t+1}^{(i)} = x_t^{(i)} & \text{if } i \neq i_{t+1} \end{cases}$$

Coordinate gradient descent

Parameters: $\gamma_1, \ldots, \gamma_n > 0$

Algorithm:

Choose $i_{t+1} \in \{1, \ldots, n\}$

$$\begin{cases} x_{t+1}^{(i)} = x_t^{(i)} - \gamma_i \nabla_i f(\mathbf{x}_t) & \text{if } i = i_{t+1} \\ x_{t+1}^{(i)} = x_t^{(i)} & \text{if } i \neq i_{t+1} \end{cases}$$

Choice of γ_i? coordinate-wise Lipschitz constant i.e. Lipschitz constant of

$$g_{i,x} : \mathbb{R} \to \mathbb{R}$$
$$h \mapsto f(\mathbf{x} + \mathbf{e}_i h) = f(x^{(1)}, \ldots, x^{(i-1)}, x^{(i)} + h, x^{(i+1)}, \ldots, x^{(n)})$$

We will denote $L_i = L(\nabla g_{i,x})$ this Lipschitz constant.

Convergence speed

Assume f is convex; ∇f is Lipschitz continuous; $\forall i$, $\gamma_i = \frac{1}{L_i}$.

Proposition (Beck and Tetruashvili (2013))

If $i_{t+1} = (t \mod n) + 1$, *then*

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_*) \leq 4L_{\max}(1 + n^3 L_{\max}^2 / L_{\min}^2) \frac{R^2(\mathbf{x}_0)}{t + 8/n}$$

where $R^2(\mathbf{x}_0) = \max_{\mathbf{x},\mathbf{y} \in X} \{\|\mathbf{x} - \mathbf{y}\| : f(\mathbf{y}) \leq f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$, $L_{\max} = \max_i L_i$ *and* $L_{\min} = \min_i L_i$.

Note: n^3 is usually be prohibitive. Due to pathological cases of the cyclic rule this bound is <u>very</u> pessimistic (cf. notebook example).

Convergence speed with randomization

Assume f is convex; ∇f is Lipschitz continuous; $\forall i$, $\gamma_i = \frac{1}{L_i}$.

Proposition (Nesterov (2012))

If i_{t+1} is randomly generated, independently of $i_1, \ldots, i_t$ and $\forall i \in \{1, \ldots, n\}$, $\mathbb{P}(i_{t+1} = i) = \frac{1}{n}$, then

$$\mathbb{E}[f(\mathbf{x}_{t+1}) - f(\mathbf{x}_*)] \leq \frac{n}{t+n}\Big((1 - \frac{1}{n})(f(\mathbf{x}_0) - f(\mathbf{x}_*)) + \frac{1}{2}\|\mathbf{x}_* - \mathbf{x}_0\|_L^2\Big)$$

where $\|\mathbf{x}\|_L^2 = \sum_{i=1}^n L_i \|x^{(i)}\|_2^2$.

Note: As the algorithm is stochastic, the bound is in expectation.

Comparison with gradient descent

The iteration complexity of the gradient descent method is

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_*) \leq \frac{L(\nabla f)}{2(t+1)} \|\mathbf{x}_* - \mathbf{x}_0\|_2^2$$

To get an ϵ-solution (*i.e.*, such that $f(\mathbf{x}_t) - f(\mathbf{x}_*) \leq \epsilon$), we need at most $\frac{L(\nabla f)}{2\epsilon}\|\mathbf{x}_* - \mathbf{x}_0\|_2^2$ iterations.

while for coordinate descent we need (omitting randomization)

$$\frac{n}{\epsilon}\left((1 - \frac{1}{n})(f(\mathbf{x}_0) - f(\mathbf{x}_*)) + \frac{1}{2}\|\mathbf{x}_* - \mathbf{x}_0\|_L^2\right)$$

iterations.

Comparison with gradient descent

How do the cost of iterations compare for Least Squares?

Let C = cost of one GD iteration and c = cost of one CD update.

GD: C = cost of computing $\nabla f(\mathbf{x}) = \mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$
$\to C = O(\mathrm{nnz}(A))$ or $C = O(mn)$ for a dense matrix.

CD: $\nabla_i f(\mathbf{x}) = \mathbf{e}_i^\top \mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$ and with smart residual updates
$\to c = O(\mathrm{nnz}(A))/n$ or $c = O(m)$ for a dense matrix. So

$$c \approx C/n$$

Comparison with gradient descent

Number of iterations for CD to reach a precision ϵ:

$$\frac{n}{\epsilon}\Big((1-\frac{1}{n})(f(\mathbf{x}_0)-f(\mathbf{x}_*))+\frac{1}{2}\|\mathbf{x}_*-\mathbf{x}_0\|_L^2\Big)$$

- $f(\mathbf{x}_0)-f(\mathbf{x}_*) \leq \frac{L(\nabla f)}{2}\|\mathbf{x}_0-\mathbf{x}_*\|_2^2$ and it may happen that $f(\mathbf{x}_0)-f(\mathbf{x}_*) \ll \frac{L(\nabla f)}{2}\|\mathbf{x}_0-\mathbf{x}_*\|_2^2$
- $L(\nabla f)=\lambda_{\max}(A^\top A)$ and $L_i=\|\mathbf{a}_i\|^2$. We always have $L_i \leq L(\nabla f)$ and it may happen that $L_i = O(L(\nabla f)/n)$.
- So in the quadratic case, $C_{CD} \leq C_{GD}$ and we may have $C_{CD}=O(C_{GD}/n)$.
- Explains the results in the notebook...

Table of Contents

CD for composite separable problem?

Let us consider:

$$F(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{n} g_i(x^{(i)}) \ ,$$

with

- f convex, differentiable
- each g_i convex

The non-smooth part is here separable.

Question: Does

$$F(\mathbf{x} + d\mathbf{e}_i) \geq F(\mathbf{x}) \ \forall d \in \mathbb{R}, \ \forall i \overset{?}{\Rightarrow} F(\mathbf{x}) = \min_{\mathbf{z}} F(\mathbf{z})$$

CD for composite separable problem?

$$
\begin{aligned}
F(\mathbf{y}) - F(\mathbf{x}) &\geq \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \sum_{i=1}^{n} (g_i(y^{(i)}) - g_i(x^{(i)})) \\
&\geq \sum_{i=1}^{n} \underbrace{\left[\nabla_i f(\mathbf{x})(y^{(i)} - x^{(i)}) + (g_i(y^{(i)}) - g_i(x^{(i)}))\right]}_{\geq 0} \\
&\geq 0
\end{aligned}
$$

This suggests that it should work . . .

Proximal coordinate descent

Parameters: $\gamma_1, \ldots, \gamma_n > 0$

Algorithm:

Choose $i_{t+1} \in \{1, \ldots, n\}$

$$\begin{cases} x_{t+1}^{(i)} = \mathrm{prox}_{\gamma_i, g_i}\left(x_t^{(i)} - \gamma_i \nabla_i f(\mathbf{x}_t)\right) & \text{if } i = i_{t+1} \\ x_{t+1}^{(i)} = x_t^{(i)} & \text{if } i \neq i_{t+1} \end{cases}$$

$\mathrm{prox}_{\gamma, g}(\mathbf{y}) = \arg\min_{\mathbf{x} \in \mathbb{R}^n} g(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2_{\gamma^{-1}}$
$\mathrm{prox}_{\gamma_i, g_i}(y) = \arg\min_{x \in \mathbb{R}} g_i(x) + \frac{1}{2\gamma_i}(x - y)^2$

$\rightarrow$ derive proximal operators for $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_2^2$ and $g(\mathbf{x}) = \mathbb{I}_{[0,1]}(\mathbf{x})$.

Convergence speed

We want to minimize $F = f + g$.
Assume f and g are convex; ∇f is Lipschitz continuous;
$\forall i$, $\gamma_i = \frac{1}{L_i}$.

Proposition (Richtárik and Takáč (2014))

If i_{k+1} is randomly generated, independently of $i_1, \dots, i_k$ and $\forall i \in \{1, \dots, n\}$, $\mathbb{P}(i_{k+1} = i) = \frac{1}{n}$, then

$$\mathbb{E}[F(x_{k+1}) - F(x_*)] \leq \frac{n}{k+n}\left((1-\frac{1}{n})(F(x_0) - F(x_*)) + \frac{1}{2}\|x_* - x_0\|_L^2\right)$$

Note: One obtain the same rate as for non-composite objectives.
$\rightarrow$ cf. Proof in lecture notes.

Table of Contents

Regression and classification under sparsity constraints

$$\min_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + \sum_{i=1}^{n} g_i(x^{(i)})$$

- Lasso: $f(x) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$ and $g(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_i |x^{(i)}|$
- ℓ_1 log. reg.: $f(x) = \log(\exp(-\mathbf{y} \odot \mathbf{A}\mathbf{x}) + 1)$ and $g(\mathbf{x}) = \|x\|_1$ where $\odot$ is the elementwise product (Hadamard product).
- Box-constrained regression $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$ s.t. $\|\mathbf{x}\|_\infty \leq \kappa$
- Non-negative least squares (NNLS) $f(x) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$ s.t. $x^{(i)} \geq 0$

Note: Generally the regularizer is separable non-smooth and the data fit is smooth.

$\rightarrow$ write full algorithm for NNLS and Lasso

Multi-output regression under sparsity constraints

Multi-task Lasso (k tasks):

$$\min_{\mathbf{X} \in \mathbb{R}^{n \times k}} F(\mathbf{X}) = \min_{\mathbf{X} \in \mathbb{R}^{n \times k}} \frac{1}{2} \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_{Fro}^2 + \sum_{i=1}^{n} \|\mathbf{x}^{(i,\cdot)}\|_2$$

where $\mathbf{x}^{(i,\cdot)}$ is the i-th row of matrix $\mathbf{X}$.

Note: Here the g is still separable yet blocks of coordinates are updated at each iteration (it's block proximal coordinate descent). First convergence proof due to Tseng (2001).

Support vector machines

Coordinate descent can be applied to the SVM in the dual. If the primal with ($y_i \in \{-1, 1\}$) reads:

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \max(0, 1 - y_i(z_i^\top w + b)) + \frac{1}{2}\|w\|_2^2$$

the classical dual of SVM for binary classification is given by:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2}\alpha^T Q \alpha + \mathbf{1}^\top \alpha \text{ s.t. } y^T \alpha = 0 \text{ and } 0 \leq \alpha \leq C\mathbf{1}$$

with $Q_{ij} = y_i y_j z_i^\top z_j$.

Note: Here w is the normal to the separating hyperplane and b is the intercept.

$\rightarrow$ Derive the dual from the primal writing the Lagrangian and KKT optimality conditions.

Support vector machines with SMO

The dual reads:

$$\max_{\alpha \mathbb{R}^n} -\frac{1}{2}\alpha^T Q\alpha + \mathbf{1}^\top \alpha \text{ s.t. } y^T\alpha = 0 \text{ and } 0 \leq \alpha \leq C\mathbf{1}$$

Sequential minimal optimization or SMO (Platt, 1998) is a blockwise coordinate descent in blocks of 2. Instead of cycling, it chooses the next block greedily.

Note: This does not meet separability assumptions for convergence we have just seen.

Note: This is what is implemented in Scikit-Learn SVC and SVR estimators that use internally the libsvm C++ library.

Support vector machines with SDCA

If one does not fit an intercept b the primal reads:

$$\min_{w \in \mathbb{R}^p} C \sum_{i=1}^{n} \max(0, 1 - y_i z_i^\top w) + \frac{1}{2}\|w\|_2^2$$

and a dual formulation becomes:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2}\alpha^T Q\alpha + \mathbf{1}^\top \alpha - \mathbb{I}_{[0,C]^n}(\alpha).$$

Proximal coordinate ascent applies to this problem. When using the stochastic approach this algorithm is called Stochastic Dual Coordinate Ascent (SDCA).

Note: This is what is implemented in Scikit-Learn LinearSVC when using parameter *dual=True*. It uses internally the liblinear C++ library.

$\rightarrow$ Write an implementation of SDCA.

Support vector machines with SDCA

Proposition (Shalev-Shwartz and Zhang (2013))

Let us define a primal point $w_k = Z^\top Diag(y)\,\alpha_k$, where $(\alpha_k)_{k\geq 0}$ is generated by SDCA. The duality gap satisfies for all $K \geq n$,

$$\mathbb{E}\Big[\frac{1}{K}\sum_{k=K}^{2K-1} P(w_k) - D(\alpha_k)\Big] \leq$$

$$\frac{n}{K+n}\Big((1-\frac{1}{n})(D(\alpha_*) - D(\alpha_0)) + \frac{1}{2}\|\alpha_* - \alpha_0\|_L^2\Big) + \frac{n}{2K}C^2\sum_{i=1}^{n} L_i$$

where $\forall i, L_i = y_i^2\|z_i\|^2$.

$\rightarrow$ cf. Proof in lecture notes.

Graphical Lasso

Let $A \in \mathbb{R}^{n \times p}$, where rows are independent Gaussian observations drawn from $N(0, \Sigma)$,
The graphical Lasso estimator (Banerjee et al., 2007, Friedman et al., 2007) reads:

$$\min_{\Theta \in \mathbb{R}^{p \times p}} - \log \det \Theta + \operatorname{tr} S\Theta + \lambda \|\Theta\|_1$$

where $\|\Theta\|_1 = \sum_{ij} |\Theta_{ij}|$.

It provides an estimate of Σ^{-1} (precision matrix) when $S = A^\top A/n$ is the empirical covariance.

Graphical Lasso

Stationarity conditions:

$$-\Theta^{-1} + S + \lambda\Gamma = 0$$

where $\Gamma_{ij} \in \partial|\Theta_{ij}|$. Posing $W = \Theta^{-1}$. It is possible to do a coordinate descent on W. See Friedman et al. (2007).

Note: With $\lambda = 0$ one recovers the maximum likelihood estimator.
Note: This is implemented the *GraphLasso* estimator in Scikit-Learn or in the glasso package in R.