Coordinate Descent and Ascent Methods

Julie Nutini

Machine Learning Reading Group
November 3rd, 2015
Motivation

• Projected-Gradient Methods
  ✓ Rewrite non-smooth problem as smooth constrained problem:

  \[
  \min_{x \in C} f(x)
  \]

  × Only handles ‘simple’ constraints, e.g., bound constraints.
  → Franke-Wolfe Algorithm: minimize linear function over \( C \).

• Proximal-Gradient Methods
  ✓ Generalizes projected-gradient:

  \[
  \min_{x} f(x) + r(x),
  \]

  where \( f \) is smooth, \( r \) is general convex function (proximable).
  × Dealing with \( r(x) = \phi(Ax) \) difficult, even when \( \phi \) simple.
  → Alternating Direction Method of Multipliers

★ TODAY: We focus on coordinate descent, which is for the case
where \( r \) is separable and \( f \) has some special structure.
Coordinate Descent Methods

• Suitable for large-scale optimization (dimension $d$ is large):
  • Certain smooth (unconstrained) problems.
  • Non-smooth problems with separable constraints/regularizers.
    • e.g., $\ell_1$-regularization, bound constraints

★ Faster than gradient descent if iterations $d$ times cheaper.
Problems Suitable for Coordinate Descent

Coordinate update \(d\) times faster than gradient update for:

\[
\begin{align*}
  h_1(x) &= f(Ax) + \sum_{i=1}^{d} g_i(x_i), \quad \text{or} \quad h_2(x) = \sum_{i \in V} g_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j)
\end{align*}
\]

- \(f\) and \(f_{ij}\) smooth, convex
- \(A\) is a matrix
- \(\{V, E\}\) is a graph
- \(g_i\) general non-degenerate convex functions

Examples \(h_1\): least squares, logistic regression, lasso, \(\ell_2\)-norm SVMs.

\[
\begin{align*}
  \text{e.g., } \min_{x \in \mathbb{R}^d} & \frac{1}{2} \|Ax - b\|^2 + \lambda \sum_{i=1}^{d} |x_i|.
\end{align*}
\]

Examples \(h_2\): quadratics, graph-based label prop., graphical models.

\[
\begin{align*}
  \text{e.g., } \min_{x \in \mathbb{R}^d} & \frac{1}{2} x^T Ax + b^T x = \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} a_{ij} x_i x_j + \sum_{i=1}^{d} b_i x_i.
\end{align*}
\]
Notation and Assumptions

We focus on the convex optimization problem

\[
\min_{x \in \mathbb{R}^d} f(x)
\]

- \( \nabla f \) coordinate-wise \( L \)-Lipschitz continuous

\[
|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L |\alpha|
\]

- \( f \) \( \mu \)-strongly convex, i.e.,

\[
x \mapsto f(x) - \frac{\mu}{2} ||x||^2
\]

is convex for some \( \mu > 0 \).

- If \( f \) is twice-differentiable, equivalent to

\[
\nabla^2_{ii} f(x) \leq L, \quad \nabla^2 f(x) \succeq \mu I.
\]
Coordinate Descent vs. Gradient Descent

\[ x^{k+1} = x^k - \frac{1}{L} \nabla_{i_k} f(x^k) e_{i_k} \]

\[ x^{k+1} = x^k - \alpha \nabla f(x^k) \]

- Global convergence rate for randomized \( i_k \) selection [Nesterov]:

\[
\mathbb{E}[f(x^{k+1})] - f(x^*) \leq \left(1 - \frac{\mu}{Ld}\right) [f(x^k) - f(x^*)]
\]

- Global convergence rate for gradient descent:

\[
f(x^{k+1}) - f(x^*) \leq \left(1 - \frac{\mu}{L_f}\right) [f(x^k) - f(x^*)]
\]

- Since \( Ld \geq L_f \geq L \), coordinate descent is slower per iteration, but \( d \) coordinate iterations are faster than one gradient iteration.
Proximal Coordinate Descent

\[
\min_{x \in \mathbb{R}^d} F(x) \equiv f(x) + \sum_i g_i(x_i)
\]

where \(f\) is smooth and \(g_i\) might be non-smooth.

- e.g., \(\ell_1\)-regularization, bound constraints

- Apply proximal-gradient style update,

\[
x^{k+1} = \text{prox} \frac{1}{L} g_{i_k} \left[ x^k - \frac{1}{L} \nabla f_i(x^k) e_{i_k} \right]
\]

where

\[
\text{prox}_{\alpha g}[y] = \arg\min_{x \in \mathbb{R}^d} \frac{1}{2} \| x - y \|^2 + \alpha g(x).
\]

- Convergence for randomized \(i_k\):

\[
\mathbb{E}[F(x^{k+1})] - F(x^*) \leq \left(1 - \frac{\mu}{dL}\right) \left[ F(x^k) - F(x^*) \right]
\]
Sampling Rules

- **Cyclic**: Cycle through $i$ in order, i.e., $i_1 = 1, i_2 = 2$, etc.
- **Uniform random**: Sample $i_k$ uniformly from $\{1, 2, \ldots, d\}$.
- **Lipschitz sampling**: Sample $i_k$ proportional to $L_i$.
- **Gauss-Southwell**: Select $i_k = \arg\max_i |\nabla_i f(x^k)|$.
- **Gauss-Southwell-Lipschitz**: Select $i_k = \arg\max_i \frac{|\nabla_i f(x^k)|}{\sqrt{L_i}}$. 

![Diagram of Coordinate Descent](image-url)
Gauss-Southwell Rules

| GSL: $\arg\max_i \frac{|\nabla_i f(x^k)|}{\sqrt{L_i}}$ | GS: $\arg\max_i |\nabla_i f(x^k)|$ |

**Intuition:** if gradients are similar, more progress if $L_i$ is small.

- Feasible for problems where $A$ is super sparse or for a graph with mean nNeighbours approximately equals maximum nNeighbours.
- Show GS and GSL up to $d$ times faster than randomized by measuring strong convexity in the 1-norm or $L$-norm, respectively.
Exact Optimization

\[ x^{k+1} = x^k - \alpha_k \nabla_{i_k} f(x^k)e_{i_k}, \quad \text{for some } i_k \]

• Exact coordinate optimization chooses the step size minimizing \( f \):

\[ f(x^{k+1}) = \min_{\alpha} \{ f(x^k - \alpha \nabla_{i_k} f(x^k)e_{i_k}) \} \]

• Alternatives:
  • Line search: find \( \alpha > 0 \) such that \( f(x^k - \alpha \nabla_{i_k} f(x^k)e_{i_k}) < f(x^k) \).
  • Select step size based on global knowledge of \( f \), e.g., \( 1/L \).
Stochastic Dual Coordinate Ascent

- Suitable for large-scale supervised learning (large \( n \) loss functions):
  - Primal formulated as sum of convex loss functions.
  - Operates on the dual.
- Achieves faster linear rate than SGD for smooth loss functions.
- Theoretically equivalent to SSG for non-smooth loss functions.
The Big Picture...

- **Stochastic Gradient Descent (SGD):**
  - ✓ Strong theoretical guarantees.
  - × Hard to tune step size (requires $\alpha \to 0$).
  - × No clear stopping criterion (Stochastic Sub-Gradient method (SSG)).
  - × Converges fast at first, then slow to more accurate solution.

- **Stochastic Dual Coordinate Ascent (SDCA):**
  - ✓ Strong theoretical guarantees that are comparable to SGD.
  - ✓ Easy to tune step size (line search).
  - ✓ Terminate when the duality gap is sufficiently small.
  - ✓ Converges to accurate solution faster than SGD.
Primal Problem

\[
(P) \min_{w \in \mathbb{R}^d} P(w) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(w^T x_i) + \frac{\lambda}{2} \|w\|^2
\]

where \(x_1, \ldots, x_n\) vectors in \(\mathbb{R}^d\), \(\phi_1, \ldots, \phi_n\) sequence of scalar convex functions, \(\lambda > 0\) regularization parameter.

**Examples:** (for given labels \(y_1, \ldots, y_n \in \{-1, 1\}\))

- **SVMs:** \(\phi_i(a) = \max\{0, 1 - y_ia\}\) (\(L\)-Lipschitz)
- **Regularized logistic regression:** \(\phi_i(a) = \log(1 + \exp(-y_ia))\)
- **Ridge regression:** \(\phi_i(a) = (a - y_i)^2\) (smooth)
- **Regression:** \(\phi_i(a) = |a - y_i|\)
- **Support vector regression:** \(\phi_i(a) = \max\{0, |a - y_i| - \nu\}\)
Dual Problem

\[(P) \quad \min_{w \in \mathbb{R}^d} P(w) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(w^T x_i) + \frac{\lambda}{2} \|w\|^2\]

\[(D) \quad \max_{\alpha \in \mathbb{R}^n} D(\alpha) = \frac{1}{n} \sum_{i=1}^{n} -\phi_i^*(-\alpha_i) - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i \right\|^2\]

where \(\phi_i^*(u) = \max_z (zu - \phi_i(z))\) is the convex conjugate of \(\phi_i\).

- Different dual variable associated with each example in training set.
Duality Gap

\( (P) \quad \min_{w \in \mathbb{R}^d} P(w) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(w^T x_i) + \frac{\lambda}{2} \|w\|^2 \)

\( (D) \quad \max_{\alpha \in \mathbb{R}^n} D(\alpha) = \frac{1}{n} \sum_{i=1}^{n} -\phi_i^*(-\alpha_i) - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i \right\|^2 \)

- Define \( w(\alpha) = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i \), then it is known that \( w(\alpha^*) = w^* \).
- \( P(w^*) = D(\alpha^*) \), which implies \( P(w) \geq D(\alpha) \) for all \( w, \alpha \).
- Duality gap is defined by \( P(w(\alpha)) - D(\alpha) \):
  - Upper bound on the primal sub-optimality: \( P(w(\alpha)) - P(w^*) \).
SDCA Algorithm

(1) Select a training example \( i \) at random.

(2) Do exact line search in the dual, i.e., find \( \Delta \alpha_i \):

\[
\text{maximize } -\phi_i^*(- (\alpha_i^{(t-1)} + \Delta \alpha_i)) - \frac{\lambda n}{2} \| w^{(t-1)} + (\lambda n)^{-1} \Delta \alpha_i x_i \|^2
\]

(3) Update the dual variable \( \alpha^{(t)} \) and the primal variable \( w^{(t)} \):

\[
\alpha^{(t)} \leftarrow \alpha^{(t-1)} + \Delta \alpha_i e_i
\]

\[
w^{(t)} \leftarrow w^{(t-1)} + (\lambda n)^{-1} \Delta \alpha_i x_i
\]

\* Terminate when duality gap is sufficiently small.

\* There are ways to get the rate without a line search that use the primal gradient/subgradient directions.
SGD vs. SDCA

• Alternative to SGD/SSG.
• If primal is smooth, get faster linear rate on duality gap than SGD.
• If primal is non-smooth, get sublinear rate on duality gap.
  → SDCA has similar update to SSG on primal.
  ✗ SSG sensitive to step-size.
  ✓ Do line search in the dual with coordinate ascent.

• SDCA may not perform as well as SGD for first few epochs (full pass)
  • SGD takes larger step size than SDCA earlier on, helps performance.
  • Using modified SGD on first epoch followed by SDCA obtains faster convergence when regularization parameter $\lambda \gg \log(n)/n$. 
### Comparison of Rates

**Lipschitz loss function** (e.g., hinge-loss, \( \phi_i(a) = \max\{0, 1 - y_ia\} \)):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>convergence type</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>primal</td>
<td>( \tilde{O}(1/(\lambda \varepsilon_p)) )</td>
</tr>
<tr>
<td>online EG (Collins et al., 2008) (for SVM)</td>
<td>dual</td>
<td>( \tilde{O}(n/\varepsilon_d) )</td>
</tr>
<tr>
<td>Stochastic Frank-Wolfe (Lacoste-Julien et al., 2012)</td>
<td>primal-dual</td>
<td>( \tilde{O}(n + 1/(\lambda \varepsilon)) )</td>
</tr>
<tr>
<td>SDCA</td>
<td>primal-dual</td>
<td>( \tilde{O}(n + 1/(\lambda \varepsilon)) ) or faster</td>
</tr>
</tbody>
</table>

**Smooth loss function** (e.g., ridge-regression, \( \phi_i(a) = (a - y_i)^2 \)):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>convergence type</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>primal</td>
<td>( \tilde{O}(1/(\lambda \varepsilon_p)) )</td>
</tr>
<tr>
<td>online EG (Collins et al., 2008) (for LR)</td>
<td>dual</td>
<td>( \tilde{O}((n + 1/\lambda) \log(1/\varepsilon_d)) )</td>
</tr>
<tr>
<td>SAG (Le Roux et al., 2012) (assuming ( n \geq 8/(\lambda \gamma) ))</td>
<td>primal</td>
<td>( \tilde{O}((n + 1/\lambda) \log(1/\varepsilon_p)) )</td>
</tr>
<tr>
<td>SDCA</td>
<td>primal-dual</td>
<td>( \tilde{O}((n + 1/\lambda) \log(1/\varepsilon)) )</td>
</tr>
</tbody>
</table>

**Even if** \( \alpha \) **is** \( \varepsilon_d \)-sub-optimal in the dual, i.e.,

\[
D(\alpha) - D(\alpha^*) \leq \varepsilon_d,
\]

the primal solution \( w(\alpha) \) might be far from optimal.

**Bound on duality-gap is upper bound on primal sub-optimality.**

**Recent results have shown improvements upon some of the rates in the above tables.**
Accelerated Coordinate Descent

- Inspired by Nesterov’s accelerated gradient method.
- Uses multi-step strategy, carries momentum from previous iterations.
- For accelerated randomized coordinate descent:
  - e.g., for a convex function: $O(1/k^2)$ rate, instead of $O(1/k)$. 
Block Coordinate Descent

\[ x^{k+1} = x^k - \frac{1}{L} \nabla_{b_k} f(x^k)e_{b_k}, \text{ for some block of indices } b_k \]

- Search along coordinate hyperplane.
- Fixed blocks, adaptive blocks.
- Randomized/proximal CD easily extended to the block case.
  - For proximal case, choice of block must be consistent with block-separable structure of regularization function \( g \).
Parallel Coordinate Descent

- **Synchronous parallelism:**
  - Divide iterate updates between processors (block), followed by synchronization step.

- **Asynchronous parallelism:**
  - Each processor:
    - Has access to $x$.
    - Chooses an index $i$, loads components of $x$ that are needed to compute the gradient component $\nabla_i f(x)$, then updates the $i$th component $x_i$.
    - **No attempt to coordinate or synchronize with other processors.**
    - Always using ‘stale’ $x$: convergence results restrict how stale.
Discussion

- **Coordinate Descent:**
  - Suitable for large-scale optimization (when $d$ is large).
  - Operates on the **primal** objective.
  - Faster than gradient descent if iterations $d$ times cheaper.

- **Stochastic Dual Coordinate Ascent:**
  - Suitable for large-scale optimization (when $n$ is large).
  - Operates on the **dual** objective.
  - If primal is smooth, obtains faster linear rate on duality gap than SGD.
  - If primal is non-smooth, obtain sublinear rate on duality gap.
    - Do line search in the dual with coordinate ascent.
  - Outperforms SGD when relatively high solution accuracy is required.
  - Terminate when duality-gap is sufficiently small.

- **Variations:** acceleration, block, parallel.