Nearest Neighbor based Coordinate Descent

Pradeep Ravikumar, UT Austin

Joint with Inderjit Dhillon, Ambuj Tewari

Simons Workshop on Information Theory, Learning and Big Data, 2015

Modern Big Data

- Across modern applications {fMRI images, gene expression profiles, social networks}
  - many^many variables in system, not enough observations
- Curse of dimensionality
  - To train the system or model, number of observations have to be much larger than variables in system (scaling exponentially in non-parametric models, polynomially in parametric models)

Modern Big Data

- Research over the last decade and a half:
  - Finesse curse of dimensionality when there is some intrinsic “low-dimensional structure” such as (group) sparsity, low rank, etc.

Modern Big Data

- Research over the last decade and a half:
  - Finesse curse of dimensionality when there is some intrinsic "low-dimensional structure" such as (group) sparsity, low rank, etc.
  - See Negahban, Ravikumar, Wainwright, Yu, 2012; Chandrasekharan, Recht, Willsky, 2012 for a general linear-algebraic notion of structure

Modern Big Data

- Research over the last decade and a half:
  - Finesse curse of dimensionality when there is some intrinsic "low-dimensional structure" such as (group) sparsity, low rank, etc.
  - See Negahban, Ravikumar, Wainwright, Yu, 2012; Chandrasekharan, Recht, Willsky, 2012 for a general linear-algebraic notion of structure
- Under such structure, we know how to obtain estimators whose **statistical or sample complexity** depends weakly on problem dimension "p"
  — typically scaling as log(p)

Modern Big Data

- Research over the last decade and a half:
  - Finesse curse of dimensionality when there is some intrinsic "low-dimensional structure" such as (group) sparsity, low rank, etc.
  - See Negahban, Ravikumar, Wainwright, Yu, 2012; Chandrasekharan, Recht, Willsky, 2012 for a general linear-algebraic notion of structure
- Under such structure, we know how to obtain estimators whose **statistical or sample complexity** depends weakly on problem dimension "p"
  — typically scaling as log(p)
- Can we achieve similar weak dependence on "p" in **computational complexity**?

Convex Optimization

- Optimization Problem:

$$\min_{w \in \mathbb{R}^p} \mathcal{L}(w).$$

  - Loss $\mathcal{L}$ is *convex and smooth*:

$$\|\nabla \mathcal{L}(w) - \nabla \mathcal{L}(v)\|_\infty \leq \kappa_1 \cdot \|w - v\|_1$$

  - *Sparse minimizer* w^*: $\|w^*\|_0 = s$, $\|w^*\|_\infty \leq B$

Coordinate Descent

- Optimization Problem:

$$\min_{w \in \mathbb{R}^p} \mathcal{L}(w).$$

Algorithm Cyclic coordinate Descent

Initialize: Set the initial value of w^0.

for $n = 1, \ldots$ **do**

$j = t \bmod p$.

$w_j^t \in \arg\min_{\alpha} \mathcal{L}(w^{t-1} + \alpha e_j)$

$w_l^t = w_l^{t-1}$, for $l \neq j$.

end for

Coordinate Descent (CD)

- Coordinate Descent
  - Optimize only a single coordinate per step

Coordinate Descent (CD)

- Coordinate Descent
  - Optimize only a single coordinate per step
    - Small computation per step; well suited for high-dimensional problems

Coordinate Descent (CD)

- Coordinate Descent
    - Optimize only a single coordinate per step
        - Small computation per step; well suited for high-dimensional problems
    - Recently shown to enjoy good empirical performance

Coordinate Descent (CD)

- Coordinate Descent
  - Optimize only a single coordinate per step
    - Small computation per step; well suited for high-dimensional problems
  - Recently shown to enjoy good empirical performance
  - But at least linear (or worse) dependence of comp. complexity on **p!**

Coordinate Descent (CD)

- Coordinate Descent
  - Optimize only a single coordinate per step
    - Small computation per step; well suited for high-dimensional problems
  - Recently shown to enjoy good empirical performance
  - But at least linear (or worse) dependence of comp. complexity on **p!**
- Suppose the optimal solution is sparse (very few coordinates are non-zero)

Coordinate Descent (CD)

- Coordinate Descent
  - Optimize only a single coordinate per step
    - Small computation per step; well suited for high-dimensional problems
  - Recently shown to enjoy good empirical performance
  - But at least linear (or worse) dependence of comp. complexity on **p!**
- Suppose the optimal solution is sparse (very few coordinates are non-zero)
  - If CD judiciously chooses coordinate to optimize at each step, can it be expected to leverage potential sparsity of optimum?

Greedy Coordinate Descent (GCD)

Optimization Problem: $\min_{w \in \mathbb{R}^p} \mathcal{L}(w).$

Algorithm Greedy Coordinate Gradient Descent

Initialize: Set the initial value of w^0.

for $t = 1, \ldots$ **do**

$j = \arg\max_l |\nabla_l \mathcal{L}(w^t)|.$

$w^t = w^{t-1} - \frac{1}{\kappa_1} \nabla_j \mathcal{L}(w^t) e_j.$

end for

Greedy Coordinate Descent: Analysis

- Loss $\mathcal{L}$ is *convex and smooth*:

$$\|\nabla\mathcal{L}(w) - \nabla\mathcal{L}(v)\|_\infty \leq \kappa_1 \cdot \|w - v\|_1$$

- *Sparse minimizer* w^*: $\|w^*\|_0 = s$, $\|w^*\|_\infty \leq B$

Greedy Coordinate Descent

Guarantee:

$$\mathcal{L}(w^t) - \mathcal{L}(w^*) \leq \frac{\kappa_1}{2} \frac{\|w^0 - w^*\|_1^2}{t} = \frac{\kappa_1}{2} \frac{s^2\, B^2}{t}$$

Greedy CD

- PRO: No. of iterations avoids costly dependence on dimension "p"
- CON: Each GCD iteration (naively implemented) takes $\Omega(p)$ time

Greedy CD

- PRO: No. of iterations avoids costly dependence on dimension “p”
- CON: Each GCD iteration (naively implemented) takes $\Omega(p)$ time
- Solution: Perform approximate greedy steps **via** reduction to Approximate Nearest Neighbor (ANN)

Greedy CD

- PRO: No. of iterations avoids costly dependence on dimension “p”
- CON: Each GCD iteration (naively implemented) takes $\Omega(p)$ time
- Solution: Perform approximate greedy steps **via** reduction to Approximate Nearest Neighbor (ANN)
  - allows us to use recent advances in **sublinear time** ANN search: e.g. locality sensitive hashing (LSH)

Fast Greedy and Nearest Neighbor

- Common objective in statistical learning:

$$\mathcal{L}(w) = \sum_{i=1}^{n} \ell(w^T x^i, y^i)$$

Fast Greedy and Nearest Neighbor

- Common objective in statistical learning:

$$\mathcal{L}(w) = \sum_{i=1}^{n} \ell(w^T x^i, y^i)$$

- $\nabla_j \mathcal{L}(w) = \langle x_j, r(w) \rangle$ is an *inner product* between feature j and "residual" $r(w) = (\ell'(w^T x^i, y^i))_{i=1}^{n}$

Fast Greedy and Nearest Neighbor

- Common objective in statistical learning:

$$\mathcal{L}(w) = \sum_{i=1}^{n} \ell(w^T x^i, y^i)$$

- $\nabla_j \mathcal{L}(w) = \langle x_j, r(w) \rangle$ is an *inner product* between feature j and "residual" $r(w) = (\ell'(w^T x^i, y^i))_{i=1}^{n}$
- Greedy step needs to compute (assuming $\|x_j\|_2 = 1$)

$$\arg\max_{j \in [p]} | \left\langle x_j, r(w^t) \right\rangle | \equiv \arg\min_{j \in [2p]} \|\bar{x}_j - r(w^t)\|_2^2$$

Fast Greedy and Nearest Neighbor

- Common objective in statistical learning:

$$\mathcal{L}(w) = \sum_{i=1}^{n} \ell(w^T x^i, y^i)$$

- $\nabla_j \mathcal{L}(w) = \langle x_j, r(w) \rangle$ is an *inner product* between feature j and "residual" $r(w) = (\ell'(w^T x^i, y^i))_{i=1}^{n}$
- Greedy step needs to compute (assuming $\|x_j\|_2 = 1$)

$$\arg\max_{j \in [p]} |\, \langle x_j, r(w^t) \rangle \,| \equiv \arg\min_{j \in [2p]} \|\bar{x}_j - r(w^t)\|_2^2$$

- Leverage state-of-the-art in NN search to do this in $o(p)$ time

Approximate Greedy CD: Analysis

- If greedy step has *multiplicative approximation factor* $(1+\epsilon_{\text{nn}})$ then:

$$\mathcal{L}(w^t) - \mathcal{L}(w^\star) \leq \frac{1+\epsilon_{\text{nn}}}{\epsilon_{\text{nn}}(1/\epsilon)+1} \cdot \frac{\kappa_1 \|w^0 - w^\star\|_1^2}{t}$$

- In summary, convergence rate is $K \cdot \frac{\kappa_1 s^2}{t}$

Fast Greedy: Computational Complexity

- If *each greedy step* costs $C_t(n, p, \epsilon_{\text{nn}})$, *overall cost* C_G to accuracy ϵ is:

$$C_G = C_t(n, p, \epsilon_{\text{nn}}) \cdot \frac{K\kappa_1 s^2}{\epsilon}$$

- *Preprocessing time* $C_-(n, p, \epsilon_{\text{nn}})$ can be amortized.

Fast Greedy: Computational Complexity

- **Locality Sensitive Hashing:** Uses random projections to hash data points such that distant points are unlikely to collide. It gives: $(\rho = 1/(1+\epsilon_{\mathrm{nn}}) < 1)$

$$C_t = O\left(np^{\rho}\right) \qquad C_- = O\left(n\,p^{1+\rho}\,\epsilon_{\mathrm{nn}}^{-2}\right)$$

Fast Greedy: Computational Complexity

- **Locality Sensitive Hashing:** Uses random projections to hash data points such that distant points are unlikely to collide. It gives: $(\rho = 1/(1+\epsilon_{\text{nn}}) < 1)$
  $$C_t = O\left(np^{\rho}\right) \qquad C_- = O\left(n\, p^{1+\rho}\, \epsilon_{\text{nn}}^{-2}\right)$$
- **Ailon & Chazelle (2006)'s method:** Uses multiple lookup tables after random projections. It gives:
  $$C_t = O\left(n \log n + \epsilon_{\text{nn}}^{-3} \log^2 p\right) \qquad C_- = O\left(p^{\epsilon_{\text{nn}}^{-2}}\right)$$

Fast Greedy: Computational Complexity

- **Locality Sensitive Hashing:** Uses random projections to hash data points such that distant points are unlikely to collide. It gives: $(\rho = 1/(1+\epsilon_{\mathrm{nn}}) < 1)$
  $$C_t = O\left(np^{\rho}\right) \qquad C_- = O\left(n\,p^{1+\rho}\,\epsilon_{\mathrm{nn}}^{-2}\right)$$
- **Ailon & Chazelle (2006)'s method:** Uses multiple lookup tables after random projections. It gives:
  $$C_t = O\left(n\log n + \epsilon_{\mathrm{nn}}^{-3}\log^2 p\right) \qquad C_- = O\left(p^{\epsilon_{\mathrm{nn}}^{-2}}\right)$$
- **Quad Trees+Random Projections:** Under *mutual incoherence*, using simple quad tree with random Gaussian projections, we obtain:
  $$C_t = O\left(p^{\epsilon_{\mathrm{nn}}^{-2}}\right) \qquad C_- = O\left(n\,p\,\log p\,\epsilon_{\mathrm{nn}}^{-2}\right)$$
  Mutual incoherence ($\mu = \max_{i\neq j}\langle x_i, x_j\rangle < 1$) plays important role in *statistical complexity* for sparse parameter recovery. Here it is also related to *computational complexity*.

Non-smooth Objectives

- Smooth plus separable *composite objective*:
  (think of $\mathcal{R} = \lambda \| \cdot \|_1$)

$$\min_{w \in \mathbb{R}^p} \mathcal{L}(w) + \mathcal{R}(w)$$

- *Separable regularizer*: $\mathcal{R}(w) = \sum_j \mathcal{R}_j(w_j)$

Non-smooth Objectives

- Smooth plus separable *composite objective*: (think of $\mathcal{R} = \lambda\|\cdot\|_1$)

$$\min_{w\in\mathbb{R}^p} \mathcal{L}(w) + \mathcal{R}(w)$$

- *Separable regularizer*: $\mathcal{R}(w) = \sum_j \mathcal{R}_j(w_j)$
- If we updated coordinate j:

$$w_j^{t+1} = \arg\min_w g_j^t(w - w_j^t) + \frac{\kappa_1}{2}(w - w_j^t)^2 + R_j(w)$$

- *Guaranteed descent* in objective is $\frac{\kappa_1}{2}|\eta_j^t|^2$ where $\eta_j^t = w_j^{t+1} - w_j^t$

Modified Greedy for non-smooth objectives

Modified greedy algorithm

(chooses j with maximum guaranteed descent)

Initialize: $w^0 \leftarrow \mathbf{0}$.
for $t = 1, \ldots$ **do**
  $j_t \leftarrow \arg\max_{j \in [p]} \left| \eta_j^t \right|$
  $w^{t+1} \leftarrow w^t + \eta_{j_t}^t \mathbf{e}_{j_t}$
end for

Guarantee:

$$\mathcal{L}(w^t) + \mathcal{R}(w^t) - \mathcal{L}(w^*) - \mathcal{R}(w^*) \leq \frac{\kappa_1}{2} \frac{\|w^0 - w^*\|_1^2}{t}$$

Experiments

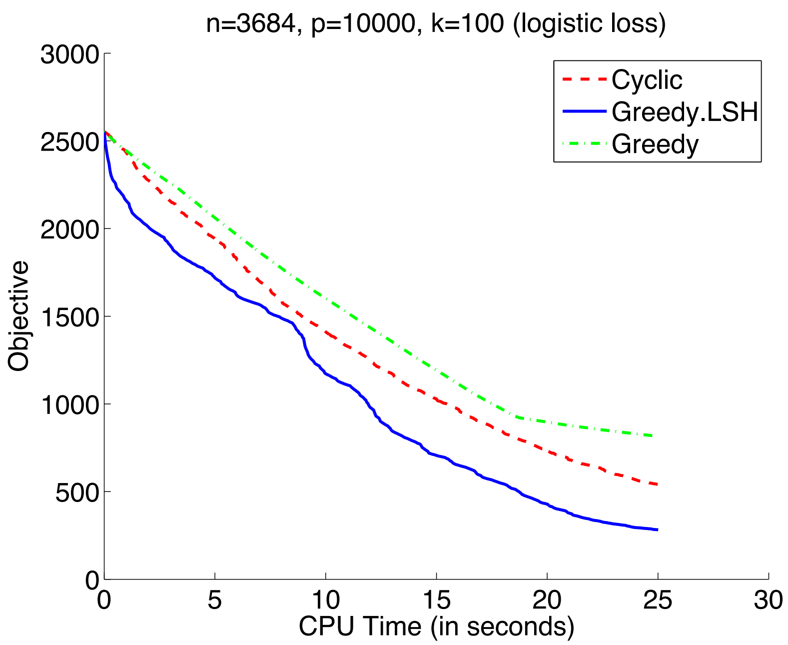
Three algos. (cyclic CD, greedy CD, greedy CD+LSH)

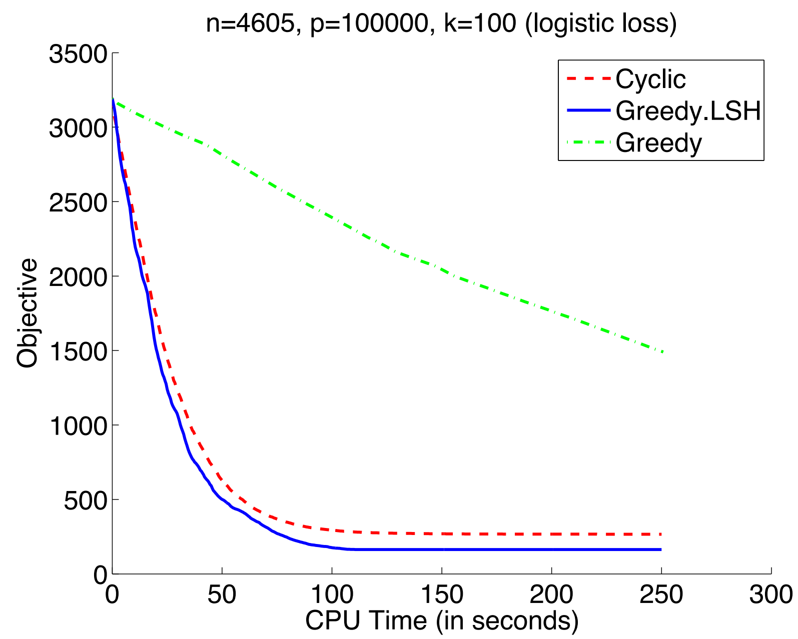
Two loss functions (logistic, squared), ℓ_1 regularization

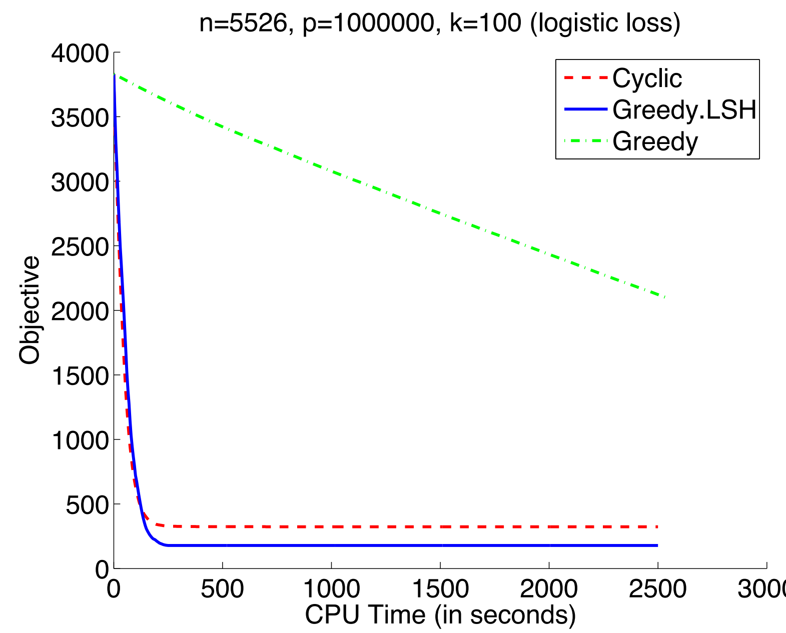
X: standard Gaussian with normalized columns

$Y = Xw_{\mathrm{tr}}$ with w_{tr} 100-sparse, $n = \lfloor 400 \log(p) \rfloor$

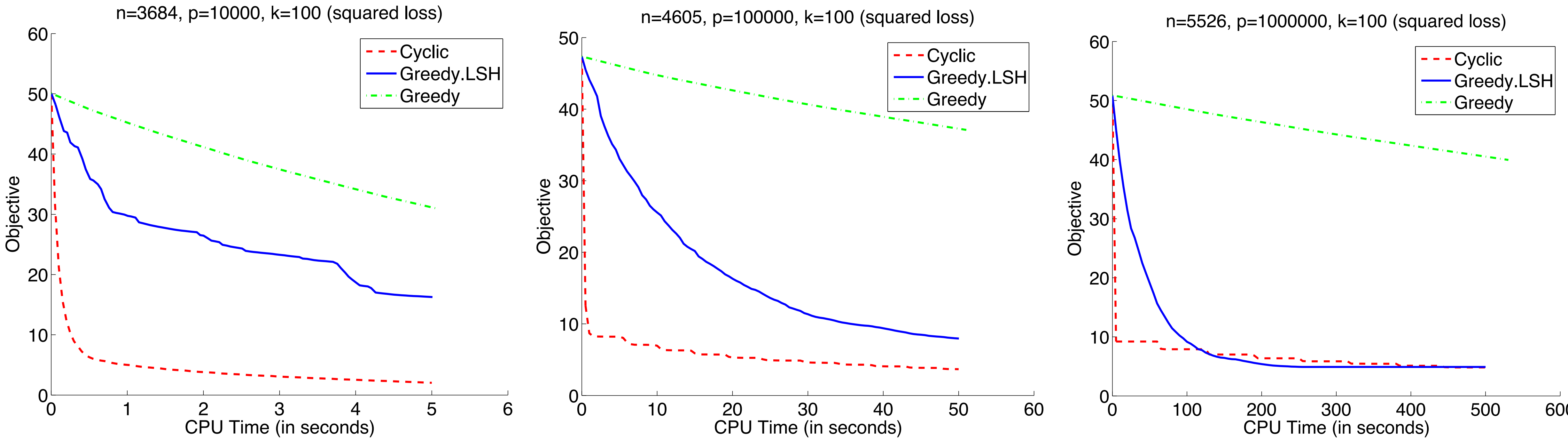
Experiments: Logistic Loss







Experiments: Squared Loss



Summary

- Optimization Method with **sub-linear** dependence on **p**!
- New connections between computational geometry and first order optimization
- Interplay between statistical and computational efficiency: mutual incoherence ⇒ very simple data structure works for ANN
- New greedy algorithm for composite objectives