Fast gradient descent for drifting least squares regression: Non-asymptotic bounds and application to bandits

Prashanth L A[†]

Joint work with Nathaniel Korda[♯] and Rémi Munos[†]

[†]INRIA Lille - Team SequeL [♯]MLRG - Oxford University

November 26, 2014

Complacs News Recommendation Platform

- NOAM database: 17 million articles from 2010
- Task: Find the best among 2000 news feeds
- Reward: Relevancy score of the article
- Feature dimension: 80000 (approx)

[1] In collaboration with Nello Cristianini and Tom Welfare at University of Bristol

Complacs News Recommendation Platform

- NOAM database: 17 million articles from 2010
- Task: Find the best among 2000 news feeds
- Reward: Relevancy score of the article
- Feature dimension: 80000 (approx)

[1] In collaboration with Nello Cristianini and Tom Welfare at University of Bristol

Complacs News Recommendation Platform

- NOAM database: 17 million articles from 2010
- Task: Find the best among 2000 news feeds
- Reward: Relevancy score of the article
- Feature dimension: 80000 (approx)

[1] In collaboration with Nello Cristianini and Tom Welfare at University of Bristol

Complacs News Recommendation Platform

- NOAM database: 17 million articles from 2010
- Task: Find the best among 2000 news feeds
- Reward: Relevancy score of the article
- Feature dimension: 80000 (approx)

[1]In collaboration with Nello Cristianini and Tom Welfare at University of Bristol

More on relevancy score

Problem: Find the best news feed for Crime stories

Sample scores:

Five dead in Finnish mall shooting	Score: 1.93
Holidays provide more opportunities to drink	Score: -0.48
Russia raises price of vodka	Score: 2.67
Why Obama Care Must Be Defeated	Score: 0.43
University closure due to weather	Score: -1.06

More on relevancy score

Problem: Find the best news feed for Crime stories

Sample scores:

Five dead in Finnish mall shooting	Score: 1.93
Holidays provide more opportunities to drink	Score: −0.48
Russia raises price of vodka	Score: 2.67
Why Obama Care Must Be Defeated	Score: 0.43
University closure due to weather	Score: −1.06

More on relevancy score

Problem: Find the best news feed for Crime stories

Sample scores:

Five dead in Finnish mall shooting	Score: 1.93
Holidays provide more opportunities to drink	Score: -0.48
Russia raises price of vodka	Score: 2.67
Why Obama Care Must Be Defeated	Score: 0.43
University closure due to weather	Score: -1.06

More on relevancy score

Problem: Find the best news feed for Crime stories

Sample scores:

Five dead in Finnish mall shooting	Score: 1.93
Holidays provide more opportunities to drink	Score: -0.48
Russia raises price of vodka	Score: 2.67
Why Obama Care Must Be Defeated	Score: 0.43
University closure due to weather	Score: -1.06

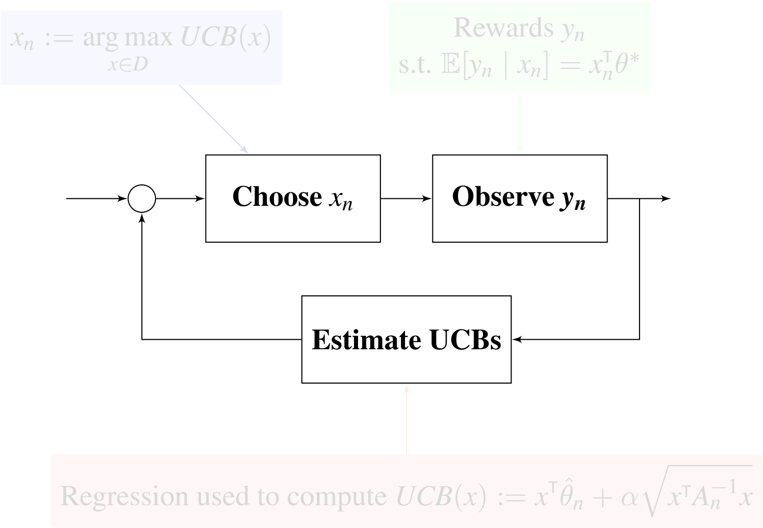
More on relevancy score

Problem: Find the best news feed for Crime stories

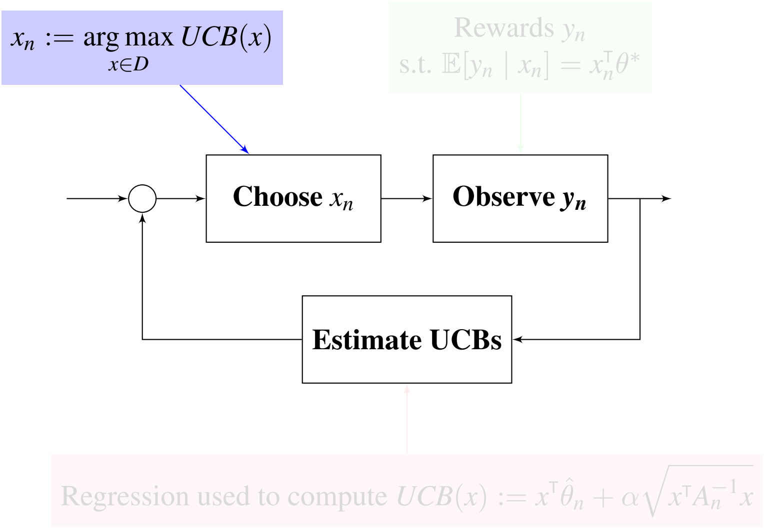
Sample scores:

Five dead in Finnish mall shooting	Score: 1.93
Holidays provide more opportunities to drink	Score: -0.48
Russia raises price of vodka	Score: 2.67
Why Obama Care Must Be Defeated	Score: 0.43
University closure due to weather	Score: -1.06

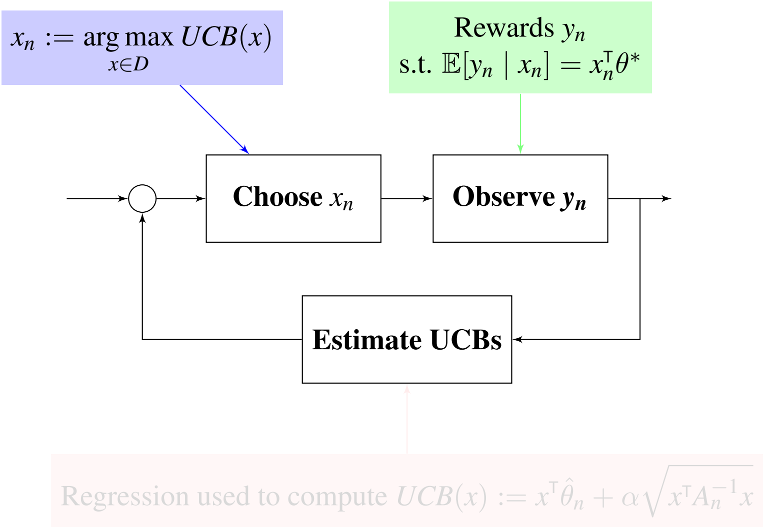
A linear bandit algorithm



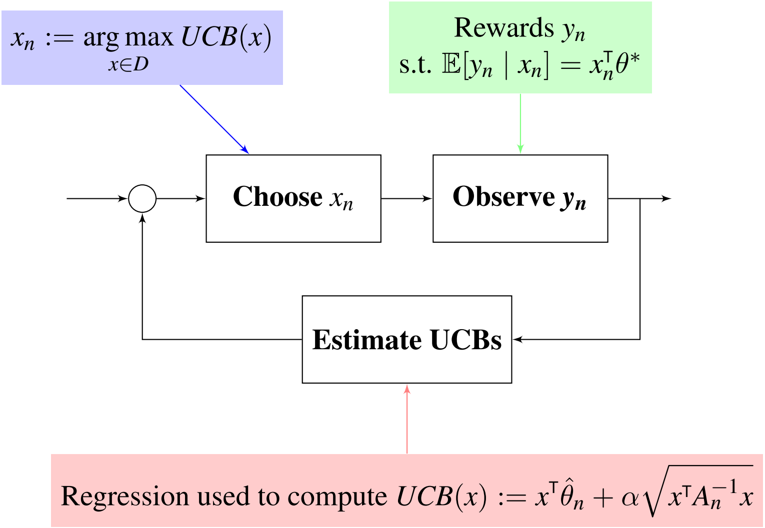
A linear bandit algorithm



A linear bandit algorithm



A linear bandit algorithm



UCB values

- Mean-reward estimate

$$UCB(x) = \hat{\mu}(x) + \alpha \, \hat{\sigma}(x)$$

- Confidence width

At each round t, select a tap. Optimize the quality of n selected beers

UCB values

- Mean-reward estimate

$$UCB(x) = \hat{\mu}(x) + \alpha\, \hat{\sigma}(x)$$

- Confidence width

At each round t, select a tap. Optimize the quality of n selected beers

UCB values

- Mean-reward estimate

$$UCB(x) = \hat{\mu}(x) + \alpha \hat{\sigma}(x)$$

- Confidence width

At each round t, select a tap. Optimize the quality of n selected beers

UCB values

Linearity $\Rightarrow$ No need to estimate mean-reward of all arms, estimating θ^* is enough

- Regression $\hat{\theta}_n = A_n^{-1} b_n$

$$UCB(x) = \hat{\mu}(x) + \alpha\, \hat{\sigma}(x)$$

- Mahalanobis distance of x from A_n: $\sqrt{x^\mathsf{T} A_n^{-1} x}$

Optimize the beer you drink, before you get drunk

UCB values

Linearity $\Rightarrow$ No need to estimate mean-reward of all arms, estimating θ^* is enough

- Regression $\hat{\theta}_n = A_n^{-1} b_n$

$$UCB(x) = \hat{\mu}(x) + \alpha \hat{\sigma}(x)$$

- Mahalanobis distance of x from A_n: $\sqrt{x^\mathsf{T} A_n^{-1} x}$

Optimize the beer you drink, before you get drunk

UCB values

Linearity $\Rightarrow$ No need to estimate mean-reward of all arms, estimating θ^* is enough

- Regression $\hat{\theta}_n = A_n^{-1} b_n$

$$UCB(x) = \hat{\mu}(x) + \alpha \, \hat{\sigma}(x)$$

- Mahalanobis distance of x from A_n: $\sqrt{x^\mathsf{T} A_n^{-1} x}$

Optimize the beer you drink, before you get drunk

Performance measure

Best arm: $x^* = \arg\min_x \{x^\mathsf{T}\theta^*\}.$

Regret: $R_T = \sum_{i=1}^{T} (x^* - x_i)^\mathsf{T}\theta^*$

Goal: ensure R_T grows sub-linearly with T

Linear bandit algorithms ensure sub-linear regret!

Performance measure

$$\text{Best arm: } x^* = \arg\min_x \{x^\mathsf{T}\theta^*\}.$$

$$\text{Regret: } R_T = \sum_{i=1}^{T} (x^* - x_i)^\mathsf{T}\theta^*$$

Goal: ensure R_T grows sub-linearly with T

Linear bandit algorithms ensure sub-linear regret!

Complexity of Least Squares Regression

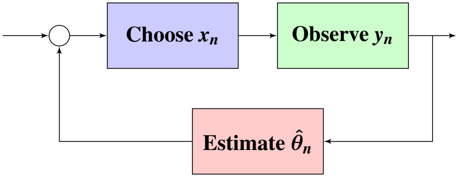


Figure : Typical ML algorithm using Regression

Regression Complexity

- $O(d^2)$ using the Sherman-Morrison lemma or
- $O(d^{2.807})$ using the Strassen algorithm or $O(d^{2.375})$ the Coppersmith-Winograd algorithm

Problem: Complacs News feed platform has **high-dimensional features** ($d \sim 10^5$) $\Rightarrow$ solving OLS is computationally costly

Complexity of Least Squares Regression

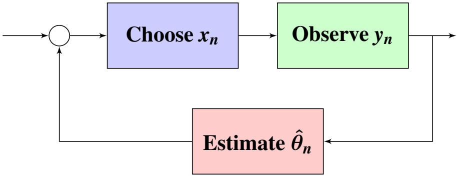


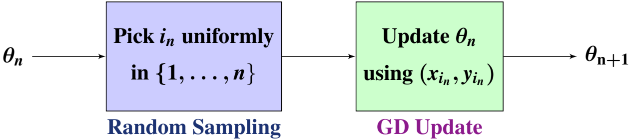
Figure : Typical ML algorithm using Regression

Regression Complexity

- $O(d^2)$ using the Sherman-Morrison lemma or
- $O(d^{2.807})$ using the Strassen algorithm or $O(d^{2.375})$ the Coppersmith-Winograd algorithm

Problem: Complacs News feed platform has **high-dimensional features** ($d \sim 10^5$) $\Rightarrow$ solving OLS is computationally costly

Fast GD for Regression



Solution: Use fast (online) gradient descent (GD)

- Efficient with complexity of only $O(d)$ (**Well-known**)
- High probability bounds with explicit constants can be derived (**not fully known**)

Bandits+GD for News Recommendation

LinUCB: a well-known **contextual bandit** algorithm that employs **regression** in each iteration

Fast GD: provides good approximation to regression (**with low computational cost**)

Strongly-Convex Bandits: no loss in regret except log-factors **Proved!**

Non Strongly-Convex Bandits: Encouraging empirical results for **linUCB+fast GD**] on two news feed platforms

Bandits+GD for News Recommendation

LinUCB: a well-known **contextual bandit** algorithm that employs **regression** in each iteration

Fast GD: provides good approximation to regression (**with low computational cost**)

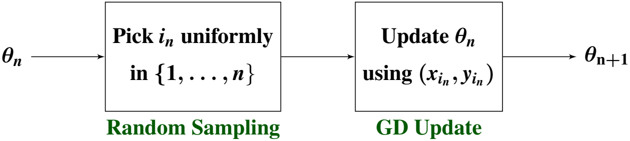
Strongly-Convex Bandits: no loss in regret except log-factors **Proved!**

Non Strongly-Convex Bandits: Encouraging empirical results for **linUCB+fast GD**] on two news feed platforms

Outline

1. Strongly convex bandits
2. Non-strongly convex bandits
3. News recommendation application

fast GD

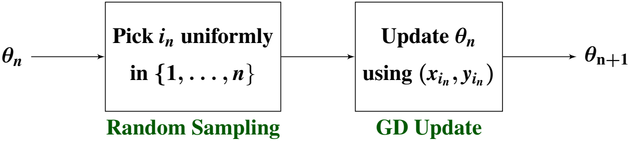


- Step-sizes

$$\theta_n = \theta_{n-1} + \gamma_n \left(y_{i_n} - \theta_{n-1}^{\mathsf{T}} x_{i_n}\right) x_{i_n}$$

- Sample gradient

fast GD

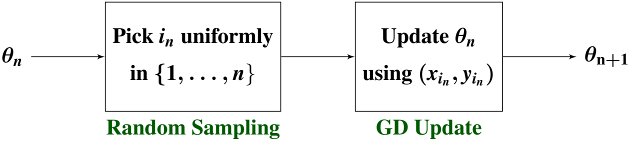


- Step-sizes

$$\theta_n = \theta_{n-1} + \gamma_n \left(y_{i_n} - \theta_{n-1}^\mathsf{T} x_{i_n} \right) x_{i_n}$$

- Sample gradient

fast GD



- Step-sizes

$$\theta_n = \theta_{n-1} + \gamma_n \left(y_{i_n} - \theta_{n-1}^\mathsf{T} x_{i_n}\right) x_{i_n}$$

- Sample gradient

Assumptions

Setting: $y_n = x_n^\mathsf{T}\theta^* + \xi_n$, where ξ_n is i.i.d. zero-mean

(A1) $\sup_n \|x_n\|_2 \le 1.$

Bounded features

(A2) $|\xi_n| \le 1, \forall n.$

Bounded noise

(A3) $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\mathsf{T}\right) \ge \mu.$

Strongly convex co-variance matrix (for each n)!

Assumptions

Setting: $y_n = x_n^\mathsf{T}\theta^* + \xi_n$, where ξ_n is i.i.d. zero-mean

(A1) $\sup_n \|x_n\|_2 \leq 1$.

(A2) $|\xi_n| \leq 1, \forall n$.

(A3) $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\mathsf{T}\right) \geq \mu$.

Bounded features

Bounded noise

Strongly convex co-variance matrix (for each n)!

Assumptions

Setting: $y_n = x_n^\mathsf{T}\theta^* + \xi_n$, where ξ_n is i.i.d. zero-mean

(A1) $\sup_n \|x_n\|_2 \leq 1.$ Bounded features

(A2) $|\xi_n| \leq 1, \forall n.$ Bounded noise

(A3) $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\mathsf{T}\right) \geq \mu.$ Strongly convex co-variance matrix (for each n)!

Assumptions

Setting: $y_n = x_n^\mathsf{T}\theta^* + \xi_n$, where ξ_n is i.i.d. zero-mean

(A1) $\sup_n \|x_n\|_2 \leq 1$.

Bounded features

(A2) $|\xi_n| \leq 1, \forall n$.

Bounded noise

(A3) $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\mathsf{T}\right) \geq \mu$.

Strongly convex co-variance matrix (for each n)!

Why deriving error bounds is difficult?

$$\begin{aligned}\theta_n - \hat{\theta}_n =& \theta_n - \hat{\theta}_{n-1} + \hat{\theta}_{n-1} - \hat{\theta}_n \\ =& \theta_{n-1} - \hat{\theta}_{n-1} + \hat{\theta}_{n-1} - \hat{\theta}_n + \gamma_n (y_{i_n} - \theta_{n-1}^\mathsf{T} x_{i_n}) x_{i_n} \\ =& \underbrace{\Pi_n(\theta_0 - \theta^*)}_{\text{Initial Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \Pi_n \Pi_k^{-1} \Delta \tilde{M}_k}_{\text{Sampling Error}} - \underbrace{\sum_{k=1}^{n} \Pi_n \Pi_k^{-1} (\hat{\theta}_k - \hat{\theta}_{k-1})}_{\text{Drift Error}},\end{aligned}$$

Present in earlier SGD works
and can be handled easily

Consequence of changing target
Hard to control!

Note: $\bar{A}_n = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\mathsf{T}$, $\Pi_n := \prod_{k=1}^{n} (I - \gamma_k \bar{A}_k)$, and $\Delta \tilde{M}_k$ is a martingale difference.

Why deriving error bounds is difficult?

$$
\begin{aligned}
\theta_n - \hat{\theta}_n =& \theta_n - \hat{\theta}_{n-1} + \hat{\theta}_{n-1} - \hat{\theta}_n \\
=& \theta_{n-1} - \hat{\theta}_{n-1} + \hat{\theta}_{n-1} - \hat{\theta}_n + \gamma_n (y_{i_n} - \theta_{n-1}^\mathsf{T} x_{i_n}) x_{i_n} \\
=& \underbrace{\Pi_n(\theta_0 - \theta^*)}_{\text{Initial Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \Pi_n \Pi_k^{-1} \Delta\tilde{M}_k}_{\text{Sampling Error}} - \underbrace{\sum_{k=1}^{n} \Pi_n \Pi_k^{-1} (\hat{\theta}_k - \hat{\theta}_{k-1})}_{\text{Drift Error}},
\end{aligned}
$$

Present in earlier SGD works
and can be handled easily

Consequence of changing target
Hard to control!

Note: $\bar{A}_n = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\mathsf{T}$, $\Pi_n := \prod_{k=1}^{n}\left(I - \gamma_k \bar{A}_k\right)$, and $\Delta\tilde{M}_k$ is a martingale difference.

Why deriving error bounds is difficult?

$$
\begin{aligned}
\theta_n - \hat{\theta}_n &= \theta_n - \hat{\theta}_{n-1} + \hat{\theta}_{n-1} - \hat{\theta}_n \\
&= \theta_{n-1} - \hat{\theta}_{n-1} + \hat{\theta}_{n-1} - \hat{\theta}_n + \gamma_n (y_{i_n} - \theta_{n-1}^\mathsf{T} x_{i_n}) x_{i_n} \\
&= \underbrace{\Pi_n(\theta_0 - \theta^*)}_{\text{Initial Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \Pi_n \Pi_k^{-1} \Delta \tilde{M}_k}_{\text{Sampling Error}} - \underbrace{\sum_{k=1}^{n} \Pi_n \Pi_k^{-1} (\hat{\theta}_k - \hat{\theta}_{k-1})}_{\text{Drift Error}},
\end{aligned}
$$

Present in earlier SGD works
and can be handled easily

Consequence of changing target
Hard to control!

Note: $\bar{A}_n = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\mathsf{T}$, $\Pi_n := \prod_{k=1}^{n} (I - \gamma_k \bar{A}_k)$, and $\Delta \tilde{M}_k$ is a martingale difference.

Handling Drift Error

Note $F_n(\theta) := \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta^\intercal x_i)^2$ and $\bar{A}_n = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\intercal$. Also, $\mathbb{E}[y_n \mid x_n] = x_n^\intercal \theta^*$.

To control the drift error, we observe that

$$\left(\nabla F_n(\hat{\theta}_n) = 0 = \nabla F_{n-1}(\hat{\theta}_{n-1})\right)$$
$$\Longrightarrow \left(\hat{\theta}_{n-1} - \hat{\theta}_n = \xi_n A_{n-1}^{-1} x_n - (x_n^\intercal(\hat{\theta}_n - \theta^*)) A_{n-1}^{-1} x_n\right).$$

Thus, drift is controlled by the convergence of $\hat{\theta}_n$ to θ^*

Key: confidence ball result[1]

[1] Dani, Varsha, Thomas P. Hayes, and Sham M. Kakade, (2008) "Stochastic Linear Optimization under Bandit Feedback." In: COLT

Handling Drift Error

Note $F_n(\theta) := \dfrac{1}{2}\sum_{i=1}^{n}(y_i - \theta^\intercal x_i)^2$ and $\bar{A}_n = \dfrac{1}{n}\sum_{i=1}^{n} x_i x_i^\intercal$. Also, $\mathbb{E}[y_n \mid x_n] = x_n^\intercal \theta^*$.

To control the drift error, we observe that

$$\left(\nabla F_n(\hat{\theta}_n) = 0 = \nabla F_{n-1}(\hat{\theta}_{n-1})\right)$$
$$\Longrightarrow \left(\hat{\theta}_{n-1} - \hat{\theta}_n = \xi_n A_{n-1}^{-1} x_n - (x_n^\intercal(\hat{\theta}_n - \theta^*))A_{n-1}^{-1} x_n\right).$$

Thus, drift is controlled by the convergence of $\hat{\theta}_n$ to θ^*

Key: confidence ball result[1]

[1] Dani, Varsha, Thomas P. Hayes, and Sham M. Kakade, (2008) "Stochastic Linear Optimization under Bandit Feedback." In: COLT

Handling Drift Error

Note $F_n(\theta) := \dfrac{1}{2}\sum_{i=1}^{n}(y_i - \theta^\intercal x_i)^2$ and $\bar{A}_n = \dfrac{1}{n}\sum_{i=1}^{n} x_i x_i^\intercal$. Also, $\mathbb{E}[y_n \mid x_n] = x_n^\intercal \theta^*$.

To control the drift error, we observe that

$$\left(\nabla F_n(\hat{\theta}_n) = 0 = \nabla F_{n-1}(\hat{\theta}_{n-1})\right)$$
$$\Longrightarrow \left(\hat{\theta}_{n-1} - \hat{\theta}_n = \xi_n A_{n-1}^{-1} x_n - (x_n^\intercal(\hat{\theta}_n - \theta^*))A_{n-1}^{-1} x_n\right).$$

Thus, drift is controlled by the convergence of $\hat{\theta}_n$ to θ^*

Key: confidence ball result[1]

[1] Dani, Varsha, Thomas P. Hayes, and Sham M. Kakade, (2008) "Stochastic Linear Optimization under Bandit Feedback." In: COLT

Error bound

With $\gamma_n = c/(4(c+n))$ and $\mu c/4 \in (2/3, 1)$ we have:

High prob. bound For any $\delta > 0$,

$$P\left(\left\| \theta_n - \hat{\theta}_n \right\|_2 \leq \sqrt{\frac{K_{\mu,c}}{n} \log \frac{1}{\delta}} + \frac{h_1(n)}{\sqrt{n}} \right) \geq 1 - \delta.$$

Optimal rate $O\left(n^{-1/2}\right)$

Bound in expectation

$$\mathbb{E} \left\| \theta_n - \hat{\theta}_n \right\|_2 \leq \frac{\left\| \theta_0 - \hat{\theta}_n \right\|_2}{n^{\mu c}} + \frac{h_2(n)}{\sqrt{n}}.$$

- Initial error
- Sampling error

[1] $K_{\mu,c}$ is a constant depending on μ and c and $h_1(n)$, $h_2(n)$ hide log factors.

[2] By iterate-averaging, the dependency of c on μ can be removed.

Error bound

With $\gamma_n = c/(4(c+n))$ and $\mu c/4 \in (2/3, 1)$ we have:

High prob. bound For any $\delta > 0$,

$$P\left(\left\|\theta_n - \hat{\theta}_n\right\|_2 \leq \sqrt{\frac{K_{\mu,c}}{n}\log\frac{1}{\delta}} + \frac{h_1(n)}{\sqrt{n}}\right) \geq 1 - \delta.$$

Optimal rate $O\left(n^{-1/2}\right)$

Bound in expectation

$$\mathbb{E}\left\|\theta_n - \hat{\theta}_n\right\|_2 \leq \frac{\left\|\theta_0 - \hat{\theta}_n\right\|_2}{n^{\mu c}} + \frac{h_2(n)}{\sqrt{n}}.$$

- Initial error
- Sampling error

[1] $K_{\mu,c}$ is a constant depending on μ and c and $h_1(n)$, $h_2(n)$ hide log factors.

[2] By iterate-averaging, the dependency of c on μ can be removed.

Error bound

With $\gamma_n = c/(4(c+n))$ and $\mu c/4 \in (2/3, 1)$ we have:

High prob. bound For any $\delta > 0$,

$$P\left(\left\| \theta_n - \hat{\theta}_n \right\|_2 \le \sqrt{\frac{K_{\mu,c}}{n} \log \frac{1}{\delta}} + \frac{h_1(n)}{\sqrt{n}} \right) \ge 1 - \delta.$$

Optimal rate $O\left(n^{-1/2}\right)$

Bound in expectation

$$\mathbb{E} \left\| \theta_n - \hat{\theta}_n \right\|_2 \le \frac{\left\| \theta_0 - \hat{\theta}_n \right\|_2}{n^{\mu c}} + \frac{h_2(n)}{\sqrt{n}}.$$

- Initial error
- Sampling error

[1] $K_{\mu,c}$ is a constant depending on μ and c and $h_1(n)$, $h_2(n)$ hide log factors.

[2] By iterate-averaging, the dependency of c on μ can be removed.

PEGE Algorithm[1]

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

- Pull each of the d basis arms once
- Using losses, compute OLS
- Use OLS estimate to compute a greedy decision
- Pull the greedy arm m times

For each cycle $m = 1, 2, \ldots$ **do**

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

$$\hat{\theta}_{md} = \frac{1}{m}\left(\sum_{i=1}^{d} b_i b_i^{\mathsf{T}}\right)^{-1} \sum_{i=1}^{m}\sum_{j=1}^{d} b_i y_j(i).$$

Exploitation Phase

Find $x = \arg\min_{x \in D}\{\hat{\theta}_{md}^{\mathsf{T}} x\}$

Choose arm x m times consecutively.

[1] P. Rusmevichientong and J,N. Tsitsiklis, (2010) Linearly Parameterized Bandits. In: Math. Oper. Res.

PEGE Algorithm[1]

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

For each cycle $m = 1, 2, \ldots$ **do**

- Pull each of the d basis arms once

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

- Using losses, compute OLS

$$\hat{\theta}_{md} = \frac{1}{m}\left(\sum_{i=1}^{d} b_i b_i^{\mathsf{T}}\right)^{-1} \sum_{i=1}^{m}\sum_{j=1}^{d} b_i y_j(i).$$

- Use OLS estimate to compute a greedy decision

Exploitation Phase

Find $x = \arg\min_{x\in D}\{\hat{\theta}_{md}^{\mathsf{T}} x\}$

- Pull the greedy arm m times

Choose arm x m times consecutively.

[1] P. Rusmevichientong and J,N. Tsitsiklis, (2010) Linearly Parameterized Bandits. In: Math. Oper. Res.

PEGE Algorithm[1]

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

For each cycle $m = 1, 2, \ldots$ **do**

- Pull each of the d basis arms once

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

- Using losses, compute OLS

$$\hat{\theta}_{md} = \frac{1}{m}\left(\sum_{i=1}^{d} b_i b_i^{\mathsf{T}}\right)^{-1} \sum_{i=1}^{m}\sum_{j=1}^{d} b_i y_j(i).$$

- Use OLS estimate to compute a greedy decision

Exploitation Phase

Find $x = \arg\min_{x \in D}\{\hat{\theta}_{md}^{\mathsf{T}} x\}$

- Pull the greedy arm m times

Choose arm x m times consecutively.

[1] P. Rusmevichientong and J,N. Tsitsiklis, (2010) Linearly Parameterized Bandits. In: Math. Oper. Res.

PEGE Algorithm[1]

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

For each cycle $m = 1, 2, \ldots$ **do**

- Pull each of the d basis arms once

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

- Using losses, compute OLS

$$\hat{\theta}_{md} = \frac{1}{m} \left(\sum_{i=1}^{d} b_i b_i^{\mathsf{T}} \right)^{-1} \sum_{i=1}^{m} \sum_{j=1}^{d} b_i y_j(i).$$

- Use OLS estimate to compute a greedy decision

Exploitation Phase

Find $x = \arg\min_{x \in D} \{\hat{\theta}_{md}^{\mathsf{T}} x\}$

- Pull the greedy arm m times

Choose arm x m times consecutively.

[1] P. Rusmevichientong and J,N. Tsitsiklis, (2010) Linearly Parameterized Bandits. In: Math. Oper. Res.

PEGE Algorithm with fast GD

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

- Pull each of the d basis arms once
- Using losses, update fast GD iterate
- Use fast GD iterate to compute a greedy decision
- Pull the greedy arm m times

For each cycle $m = 1, 2, \ldots$ **do**

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

Update fast GD iterate θ_{md}

Exploitation Phase

Find $x = \arg\min_{x \in D} \{\theta_{md}^{\mathsf{T}} x\}$

Choose arm x m times consecutively.

PEGE Algorithm with fast GD

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

- Pull each of the d basis arms once
- Using losses, update fast GD iterate
- Use fast GD iterate to compute a greedy decision
- Pull the greedy arm m times

For each cycle $m = 1, 2, \ldots$ **do**

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

Update fast GD iterate θ_{md}

Exploitation Phase

Find $x = \arg\min_{x \in D} \{\theta_{md}^{\mathsf{T}} x\}$

Choose arm x m times consecutively.

PEGE Algorithm with fast GD

Input A basis $\{b_1, \dots, b_d\} \in D$ for $\mathbb{R}^d$.

- Pull each of the d basis arms once
- Using losses, update fast GD iterate
- Use fast GD iterate to compute a greedy decision
- Pull the greedy arm m times

For each cycle $m = 1, 2, \dots$ **do**

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

Update fast GD iterate θ_{md}

Exploitation Phase

Find $x = \arg\min_{x \in D} \{\theta_{md}^{\mathsf{T}} x\}$

Choose arm x m times consecutively.

PEGE Algorithm with fast GD

Input A basis $\{b_1, \ldots, b_d\} \in D$ for $\mathbb{R}^d$.

- Pull each of the d basis arms once
- Using losses, update fast GD iterate
- Use fast GD iterate to compute a greedy decision
- Pull the greedy arm m times

For each cycle $m = 1, 2, \ldots$ **do**

Exploration Phase

For $i = 1$ **to** d
- Choose arm b_i
- Observe $y_i(m)$.

Update fast GD iterate θ_{md}

Exploitation Phase

Find $x = \arg\min_{x \in D} \{\theta_{md}^{\mathsf{T}} x\}$

Choose arm x m times consecutively.

Regret bound for PEGE+fast GD

(Strongly Convex Arms):
(A3) The function $G : \theta \to \arg\min_{x \in \mathcal{D}} \{\theta^\intercal x\}$ is J-Lipschitz.

Theorem

Under (A1)-(A3), regret $R_T := \sum_{i=1}^{T} x_i^\intercal \theta^* - \min_{x \in \mathcal{D}} x^\intercal \theta^*$ *satisfies*

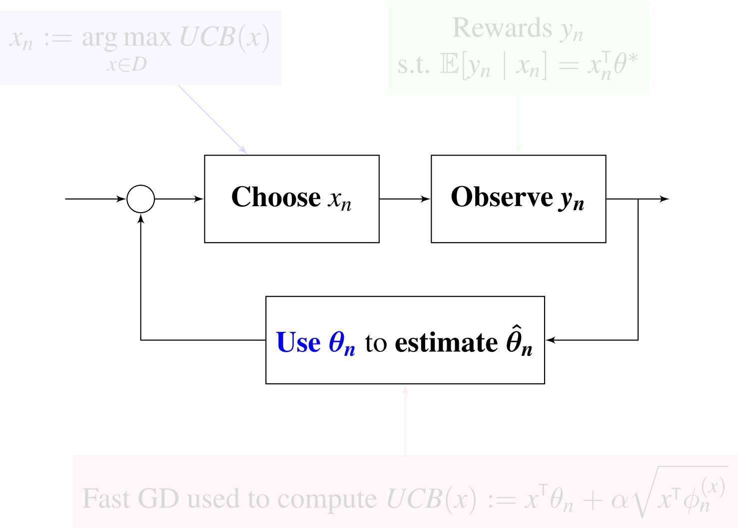
$$R_T \leq \boxed{CK_1(n)^2 d^{-1}(\|\theta^*\|_2 + \|\theta^*\|_2^{-1})\sqrt{T}}$$

The bound is worse than that for PEGE by only a factor of $O(\log^4(n))$

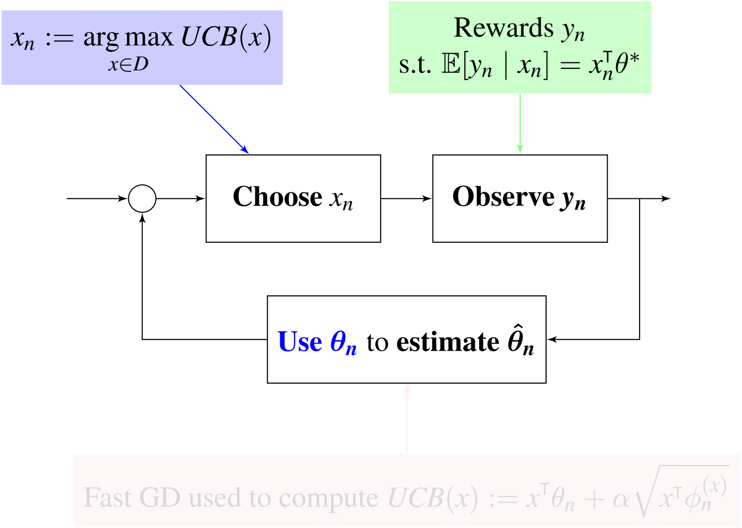
Outline

1. Strongly convex bandits
2. Non-strongly convex bandits
3. News recommendation application

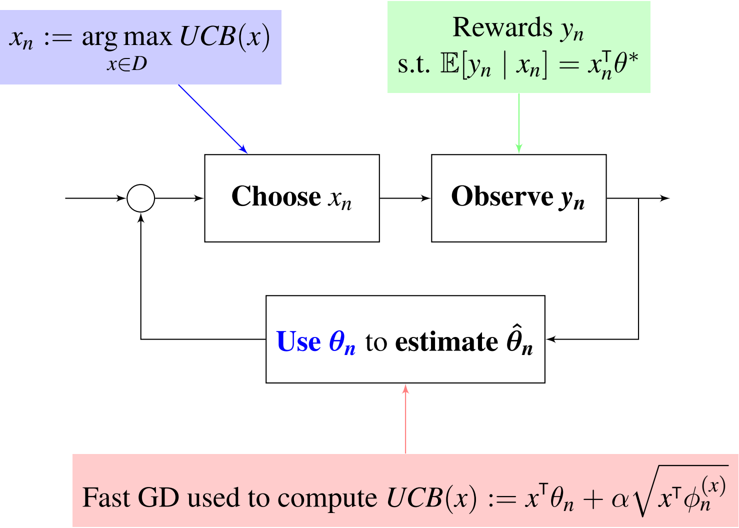
Fast linUCB


$x_n := \arg\max_{x \in D} UCB(x)$
Rewards y_n
s.t. $\mathbb{E}[y_n \mid x_n] = x_n^\mathsf{T}\theta^*$
Choose x_n
Observe y_n
Use θ_n to estimate $\hat{\theta}_n$
Fast GD used to compute $UCB(x) := x^\mathsf{T}\theta_n + \alpha\sqrt{x^\mathsf{T}\phi_n^{(x)}}$

Fast linUCB


$x_n := \arg\max_{x \in D} UCB(x)$
Rewards y_n
s.t. $\mathbb{E}[y_n \mid x_n] = x_n^\intercal \theta^*$
Choose x_n
Observe y_n
Use θ_n to **estimate $\hat{\theta}_n$**
Fast GD used to compute $UCB(x) := x^\intercal \theta_n + \alpha \sqrt{x^\intercal \phi_n^{(x)}}$

Fast linUCB


$x_n := \arg\max_{x \in D} UCB(x)$
Rewards y_n
s.t. $\mathbb{E}[y_n \mid x_n] = x_n^\mathsf{T}\theta^*$
Choose x_n
Observe $\boldsymbol{y_n}$
Use $\boldsymbol{\theta_n}$ to **estimate** $\boldsymbol{\hat{\theta}_n}$
Fast GD used to compute $UCB(x) := x^\mathsf{T}\theta_n + \alpha\sqrt{x^\mathsf{T}\phi_n^{(x)}}$

Adaptive regularization

Problem: In many settings, $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\top\right) \geq \mu$ may not hold.

Solution: Adaptively regularize with λ_n

$$\tilde{\theta}_n := \arg\min_\theta \frac{1}{2n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2 + \lambda_n \|\theta\|^2$$

GD update:

$$\theta_n = \theta_{n-1} + \gamma_n\left((y_{i_n} - \theta_{n-1}^\top x_{i_n})x_{i_n} - \lambda_n \theta_{n-1}\right)$$

Adaptive regularization

Problem: In many settings, $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\top\right) \geq \mu$ may not hold.

Solution: Adaptively regularize with λ_n

$$\tilde{\theta}_n := \arg\min_\theta \frac{1}{2n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2 + \lambda_n \|\theta\|^2$$

θ_n → Pick i_n uniformly in $\{1,\ldots,n\}$ (Random Sampling) → Update θ_n using (x_{i_n}, y_{i_n}) (GD Update) → θ_{n+1}

GD update:

$$\theta_n = \theta_{n-1} + \gamma_n\left((y_{i_n} - \theta_{n-1}^\top x_{i_n})x_{i_n} - \lambda_n\theta_{n-1}\right)$$

Adaptive regularization

Problem: In many settings, $\lambda_{\min}\left(\dfrac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\intercal\right) \geq \mu$ may not hold.

Solution: Adaptively regularize with λ_n

$$\tilde{\theta}_n := \arg\min_\theta \frac{1}{2n}\sum_{i=1}^{n}(y_i - \theta^\intercal x_i)^2 + \lambda_n \|\theta\|^2$$

θ_n → Pick i_n uniformly in $\{1,\ldots,n\}$ (Random Sampling) → Update θ_n using (x_{i_n}, y_{i_n}) (GD Update) → θ_{n+1}

GD update:

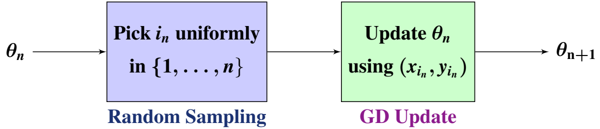
$$\theta_n = \theta_{n-1} + \gamma_n\big((y_{i_n} - \theta_{n-1}^\intercal x_{i_n})x_{i_n} - \lambda_n \theta_{n-1}\big)$$

Adaptive regularization

Problem: In many settings, $\lambda_{\min}\left(\frac{1}{n}\sum_{i=1}^{n-1} x_i x_i^\top\right) \geq \mu$ may not hold.

Solution: Adaptively regularize with λ_n

$$\tilde{\theta}_n := \arg\min_\theta \frac{1}{2n} \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2 + \lambda_n \|\theta\|^2$$



GD update:

$$\theta_n = \theta_{n-1} + \gamma_n((y_{i_n} - \theta_{n-1}^\top x_{i_n})x_{i_n} - \lambda_n \theta_{n-1})$$

Why deriving error bounds is "really" difficult here?

$$\theta_n - \tilde{\theta}_n = \underbrace{\tilde{\Pi}_n(\theta_0 - \theta^*)}_{\text{Initial Error}} - \underbrace{\sum_{k=1}^{n} \tilde{\Pi}_n \tilde{\Pi}_k^{-1} (\tilde{\theta}_k - \tilde{\theta}_{k-1})}_{\text{Drift Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \tilde{\Pi}_n \tilde{\Pi}_k^{-1} \Delta \tilde{M}_k}_{\text{Sampling Error}}, \qquad (1)$$

Need $\sum_{k=1}^{n} \gamma_k \lambda_k \to \infty$ to bound the initial error

Set $\gamma_n = O(n^{-\alpha})$ (forcing $\lambda_n = \Omega(n^{-(1-\alpha)})$)

Bad news:

This choice when plugged into (1) results in only a constant error bound!

Note: $\tilde{\Pi}_n := \prod_{k=1}^{n} \left(I - \gamma_k(\bar{A}_k + \lambda_k I)\right)$ and $\tilde{\theta}_{n-1} - \tilde{\theta}_n = \Omega(n^{-1})$, whenever $\alpha \in (0, 1)$

Why deriving error bounds is "really" difficult here?

$$\theta_n - \tilde{\theta}_n = \underbrace{\tilde{\Pi}_n(\theta_0 - \theta^*)}_{\text{Initial Error}} - \underbrace{\sum_{k=1}^{n} \tilde{\Pi}_n \tilde{\Pi}_k^{-1} (\tilde{\theta}_k - \tilde{\theta}_{k-1})}_{\text{Drift Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \tilde{\Pi}_n \tilde{\Pi}_k^{-1} \Delta \tilde{M}_k}_{\text{Sampling Error}}, \tag{1}$$

Need $\sum_{k=1}^{n} \gamma_k \lambda_k \to \infty$ to bound the initial error

Set $\gamma_n = O(n^{-\alpha})$ (forcing $\lambda_n = \Omega(n^{-(1-\alpha)})$)

Bad news:

This choice when plugged into (1) results in only a constant error bound!

Note: $\tilde{\Pi}_n := \prod_{k=1}^{n} \left(I - \gamma_k(\bar{A}_k + \lambda_k I)\right)$ and $\tilde{\theta}_{n-1} - \tilde{\theta}_n = \Omega(n^{-1})$, whenever $\alpha \in (0, 1)$

Why deriving error bounds is “really” difficult here?

$$\theta_n - \tilde{\theta}_n = \underbrace{\tilde{\Pi}_n(\theta_0 - \theta^*)}_{\text{Initial Error}} - \underbrace{\sum_{k=1}^{n} \tilde{\Pi}_n \tilde{\Pi}_k^{-1} (\tilde{\theta}_k - \tilde{\theta}_{k-1})}_{\text{Drift Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \tilde{\Pi}_n \tilde{\Pi}_k^{-1} \Delta \tilde{M}_k}_{\text{Sampling Error}}, \qquad (1)$$

Need $\sum_{k=1}^{n} \gamma_k \lambda_k \to \infty$ to bound the initial error

Set $\gamma_n = O(n^{-\alpha})$ (forcing $\lambda_n = \Omega(n^{-(1-\alpha)})$)

Bad news:

This choice when plugged into (1) results in only a constant error bound!

Note: $\tilde{\Pi}_n := \prod_{k=1}^{n} \left(I - \gamma_k(\bar{A}_k + \lambda_k I)\right)$ and $\tilde{\theta}_{n-1} - \tilde{\theta}_n = \Omega(n^{-1})$, whenever $\alpha \in (0, 1)$

Why deriving error bounds is "really" difficult here?

$$\theta_n - \tilde{\theta}_n = \underbrace{\tilde{\Pi}_n(\theta_0 - \theta^*)}_{\text{Initial Error}} - \underbrace{\sum_{k=1}^{n} \tilde{\Pi}_n \tilde{\Pi}_k^{-1}(\tilde{\theta}_k - \tilde{\theta}_{k-1})}_{\text{Drift Error}} + \underbrace{\sum_{k=1}^{n} \gamma_k \tilde{\Pi}_n \tilde{\Pi}_k^{-1} \Delta \tilde{M}_k}_{\text{Sampling Error}}, \qquad (1)$$

Need $\sum_{k=1}^{n} \gamma_k \lambda_k \to \infty$ to bound the initial error

Set $\gamma_n = O(n^{-\alpha})$ (forcing $\lambda_n = \Omega(n^{-(1-\alpha)})$)

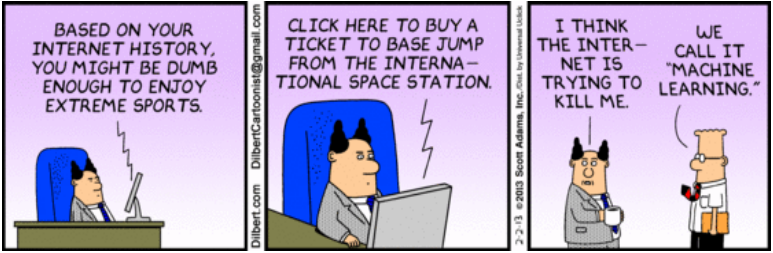
Bad news:
This choice when plugged into (1) results in only a constant error bound!

Note: $\tilde{\Pi}_n := \prod_{k=1}^{n} \left(I - \gamma_k(\bar{A}_k + \lambda_k I)\right)$ and $\tilde{\theta}_{n-1} - \tilde{\theta}_n = \Omega(n^{-1})$, whenever $\alpha \in (0,1)$

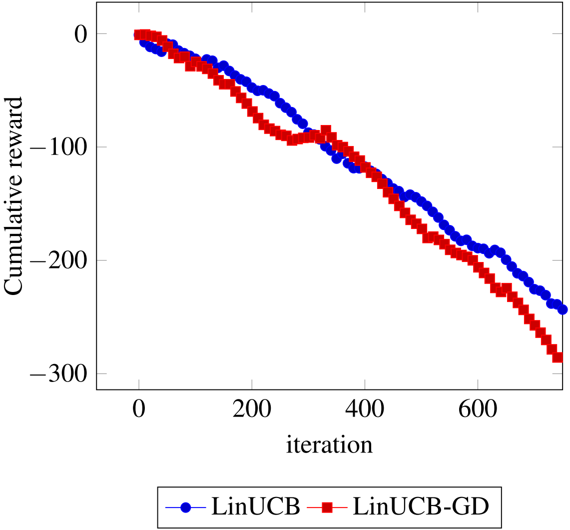
Outline

1. Strongly convex bandits
2. Non-strongly convex bandits
3. News recommendation application

Dilbert's boss on news recommendation (and ML)



Preliminary Results on Complacs News Feed Platform



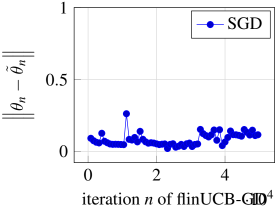
Experiments on Yahoo! Dataset [1]

Figure : The *Featured* tab in Yahoo! Today module

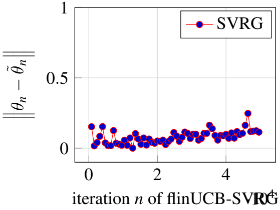
[1] Yahoo User-Click Log Dataset given under the Webscope program (2011)

Tracking Error

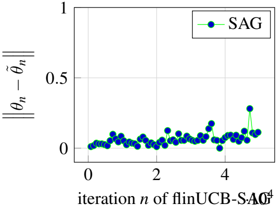
Tracking error: SGD



Tracking error: SVRG[1]



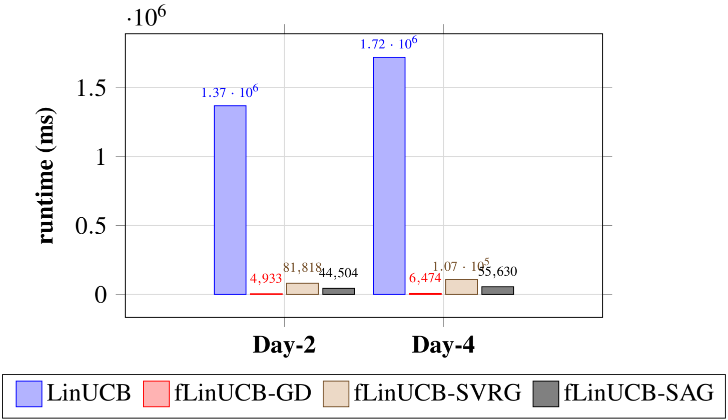
Tracking error: SAG[2]



[1] Johnson, R., and Zhang, T. (2013) "Accelerating stochastic gradient descent using predictive variance reduction". In: NIPS

[2] Roux, N. L., Schmidt, M. and Bach, F. (2012) "A stochastic gradient method with an exponential convergence rate for finite training sets." arXiv preprint arXiv:1202.6258.

Runtime Performance on two days of the Yahoo! dataset



References I

Nathaniel Korda, Prashanth L.A. and Rémi Munos,

Fast gradient descent for least squares regression: Non-asymptotic bounds and application to bandits.

AAAI, 2015.

Dilbert's boss (again) on big data!