# Stochastic Methods for L1 Regularized Loss Minimization

S. Shalev-Shwartz and *A. Tewari*

Toyota Technological Institute at Chicago

June 16, 2009

# L1 regularized loss minimization

$$\mathbf{w}^\star = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)}_{\mathcal{L}_m(\mathbf{w})} + \underbrace{\lambda\|\mathbf{w}\|_1}_{L_1 \text{ penalty}}$$

Examples:

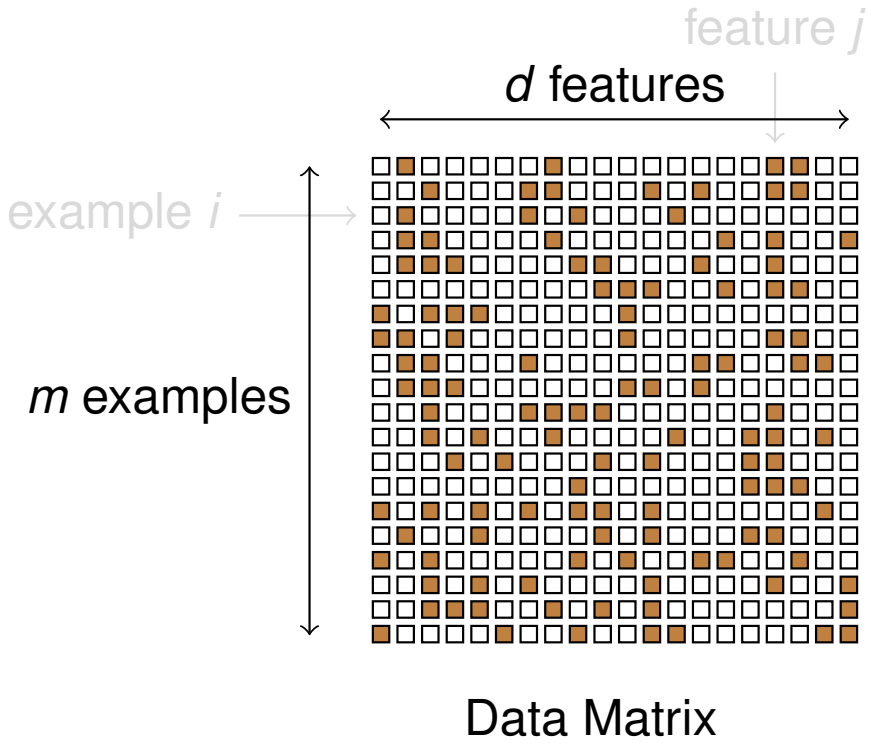
- LASSO: $L(a, y) = (a - y)^2$
- $L_1$ regularized logistic regression: $L(a, y) = \log(1 + exp(-ay))$
- $L_1$ regularized hinge-loss: $L(a, y) = \max\{0, 1 - ya\}$

# Our Contribution

- $L_1$ penalty encourages sparsity
- With $L(\cdot, y)$ convex, results in a convex optimization problem: problem solved?
- Run-time of IP methods scales typically as $\max\{d^3, m^3\}$: this is bad for large datasets
- Can live with worse dependence on $\frac{1}{\epsilon}$
- Our contribution: Two practical methods for solving L1 regularized problem for large datasets

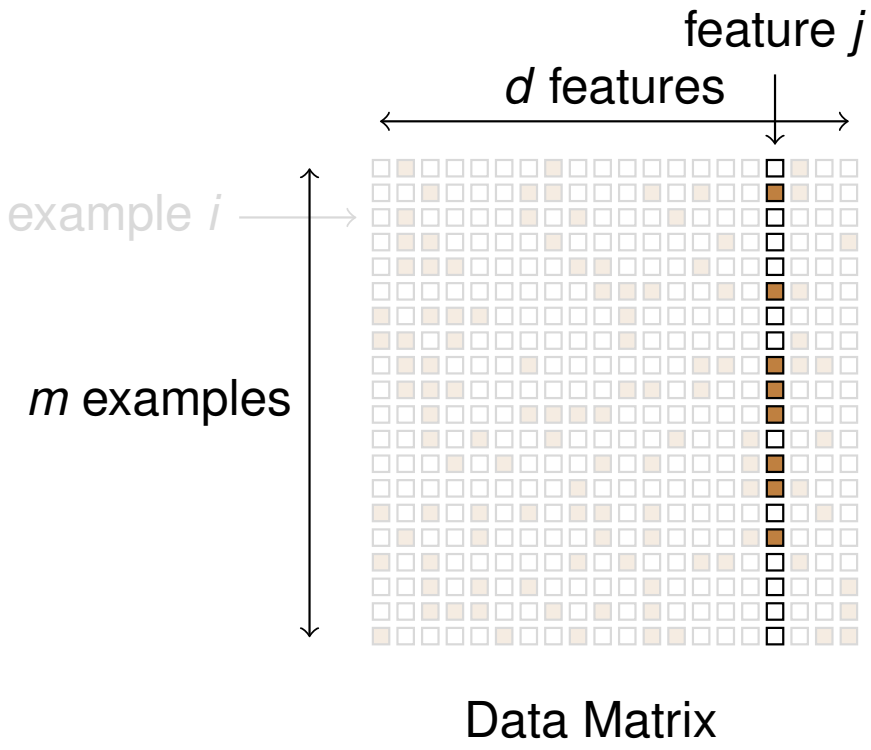
# Main Theme – Stochastic Updates

Idea: Many simple updates are more efficient than few sophisticated updates and randomness helps!



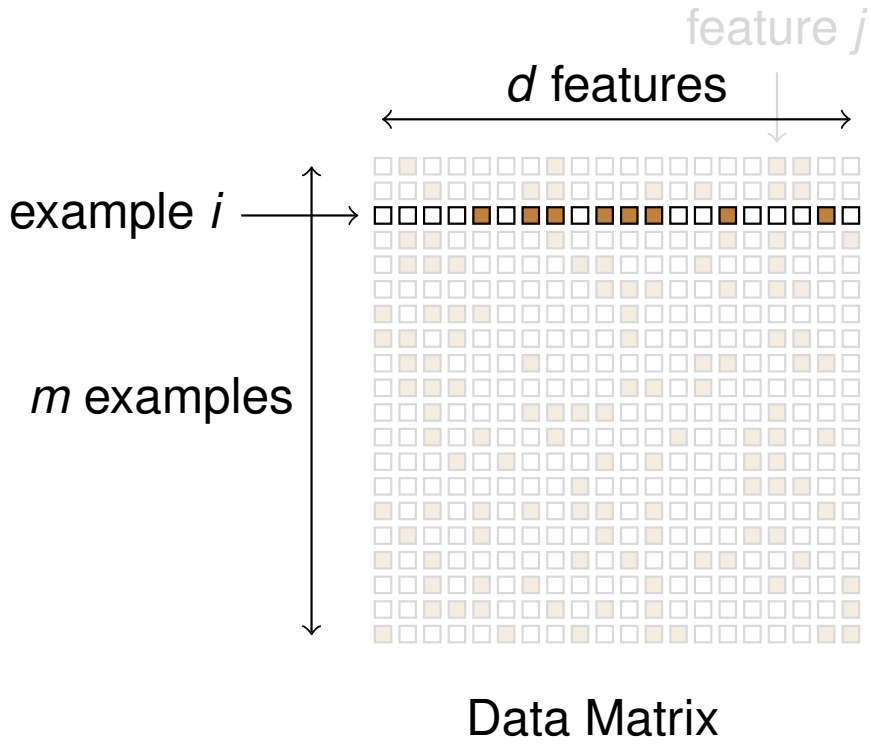
# Main Theme – Stochastic Updates

Idea: Many simple updates are more efficient than few sophisticated updates and randomness helps!



# Main Theme – Stochastic Updates

Idea: Many simple updates are more efficient than few sophisticated updates and randomness helps!




Data Matrix

# Outline

# Outline

# Outline

# Outline

# Outline

# Coordinate Descent

Goal: To minimize $R(\mathbf{w}) = R(w_1, \ldots, w_d)$

## Generic CD Algorithm

FOR $t = 1, 2, \ldots$

[SELECTION STEP]
Choose coordinate $j \in \{1, \ldots, d\}$

[UPDATE STEP]
Update $w_j$ so that $R(\mathbf{w})$ decreases (descent)

# Coordinate Descent

Goal: To minimize $R(\mathbf{w}) = R(w_1, \ldots, w_d)$

## Generic CD Algorithm

FOR $t = 1, 2, \ldots$

[SELECTION STEP]
Choose coordinate $j \in \{1, \ldots, d\}$

[UPDATE STEP]
Update $w_j$ so that $R(\mathbf{w})$ decreases (descent)

# Coordinate Descent

Goal: To minimize $R(\mathbf{w}) = R(w_1, \ldots, w_d)$

## Generic CD Algorithm

FOR $t = 1, 2, \ldots$

[SELECTION STEP]
Choose coordinate $j \in \{1, \ldots, d\}$

[UPDATE STEP]
Update $w_j$ so that $R(\mathbf{w})$ decreases (descent)

# Coordinate Descent

Goal: To minimize $R(\mathbf{w}) = R(w_1, \ldots, w_d)$

## Generic CD Algorithm

FOR $t = 1, 2, \ldots$

[SELECTION STEP]
Choose coordinate $j \in \{1, \ldots, d\}$

[UPDATE STEP]
Update $w_j$ so that $R(\mathbf{w})$ decreases (descent)

# Coordinate Descent

Goal: To minimize $R(\mathbf{w}) = R(w_1, \ldots, w_d)$

## Generic CD Algorithm

FOR $t = 1, 2, \ldots$

[SELECTION STEP]
Choose coordinate $j \in \{1, \ldots, d\}$

[UPDATE STEP]
Update $w_j$ so that $R(\mathbf{w})$ decreases (descent)

# Coordinate Descent

Goal: To minimize $R(\mathbf{w}) = R(w_1, \ldots, w_d)$

## Generic CD Algorithm

FOR $t = 1, 2, \ldots$

[SELECTION STEP]
Choose coordinate $j \in \{1, \ldots, d\}$

[UPDATE STEP]
Update $w_j$ so that $R(\mathbf{w})$ decreases (descent)

# Coordinate Descent for L1

*Coordinate-wise descent algorithms deserve more attention in convex optimization. They are simple and well-suited to large problems.*

– Friedman et al., 2007

*Our reasons for liking coordinate descent boil down to simplicity, speed and stability.*

– Wu & Lange, 2008

# Lots of Recent Activity

| | |
|---|---|
| *Genkin, Lewis, Madigan (2007)* | Minimize along 1 coordinate<br>(using quadratic approximation) |
| *Friedman et al. (2007)* | show empirically that CD is competitive<br>cycle through coordinates |
| *Wu and Lange (2008)* | cyclic & greedy coordinate selection rules<br>proof of convergence |
| *Tseng and Yun (2009)* | theoretical analysis<br>sophisticated coordinate selection rule<br>rule takes $O(d)$ time |

# A Simple Transformation

Recall we're minimizing

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i) + \lambda\|\mathbf{w}\|_1$$

Doubling dimension by letting $\hat{\mathbf{x}}_i = [\mathbf{x}_i; -\mathbf{x}_i]$, we get rid of non-differentiability of objective

## Equivalent Problem

$$\min_{\mathbf{w}\in\mathbb{R}^{2d}} R(\mathbf{w}) \quad \text{s.t.} \quad \mathbf{w} \geq \mathbf{0}$$

where

$$R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \hat{\mathbf{x}}_i\rangle, y_i) + \lambda \sum_{j=1}^{2d} w_j$$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ — simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ — $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ — to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ — simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ — $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ — to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ — simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ — $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ — to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ — simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ — $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ — to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ — simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ — $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ — to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ simple rule; takes $O(1)$ time!

[UPDATE STEP]

$g_j = (\nabla R(\mathbf{w}))_j$ $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ to keep $w_j \geq 0$

# Stochastic Coordinate Descent

## SCD Algorithm

$\beta =$ upper bound on $L''(\cdot, y)$ (e.g. $\beta = 1$ for LASSO)

FOR $t = 1, 2, \ldots$

[SELECTION STEP]

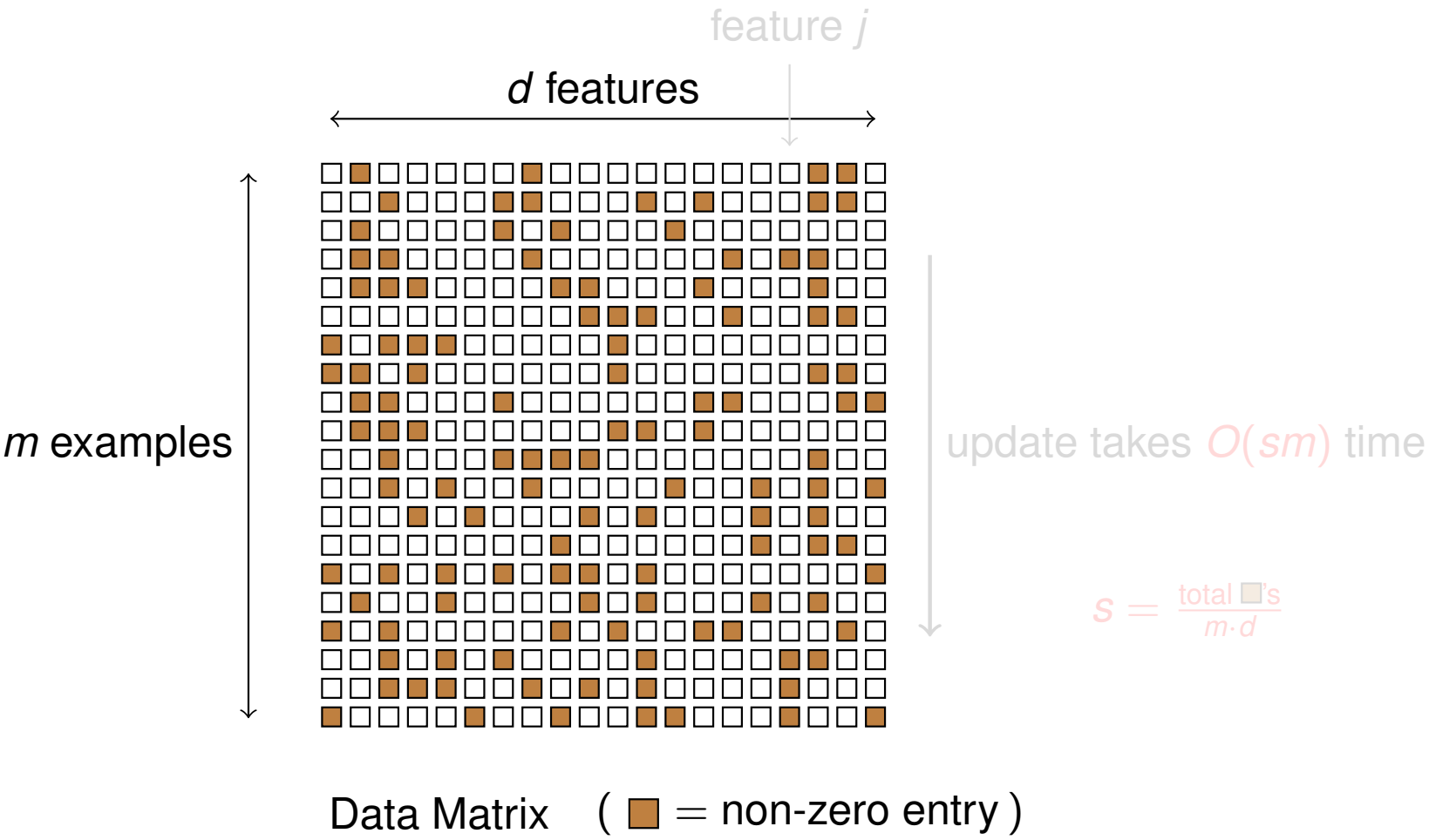
Choose $j$ *randomly* from $\{1, \ldots, 2d\}$ — simple rule; takes $O(1)$ time!

[UPDATE STEP]

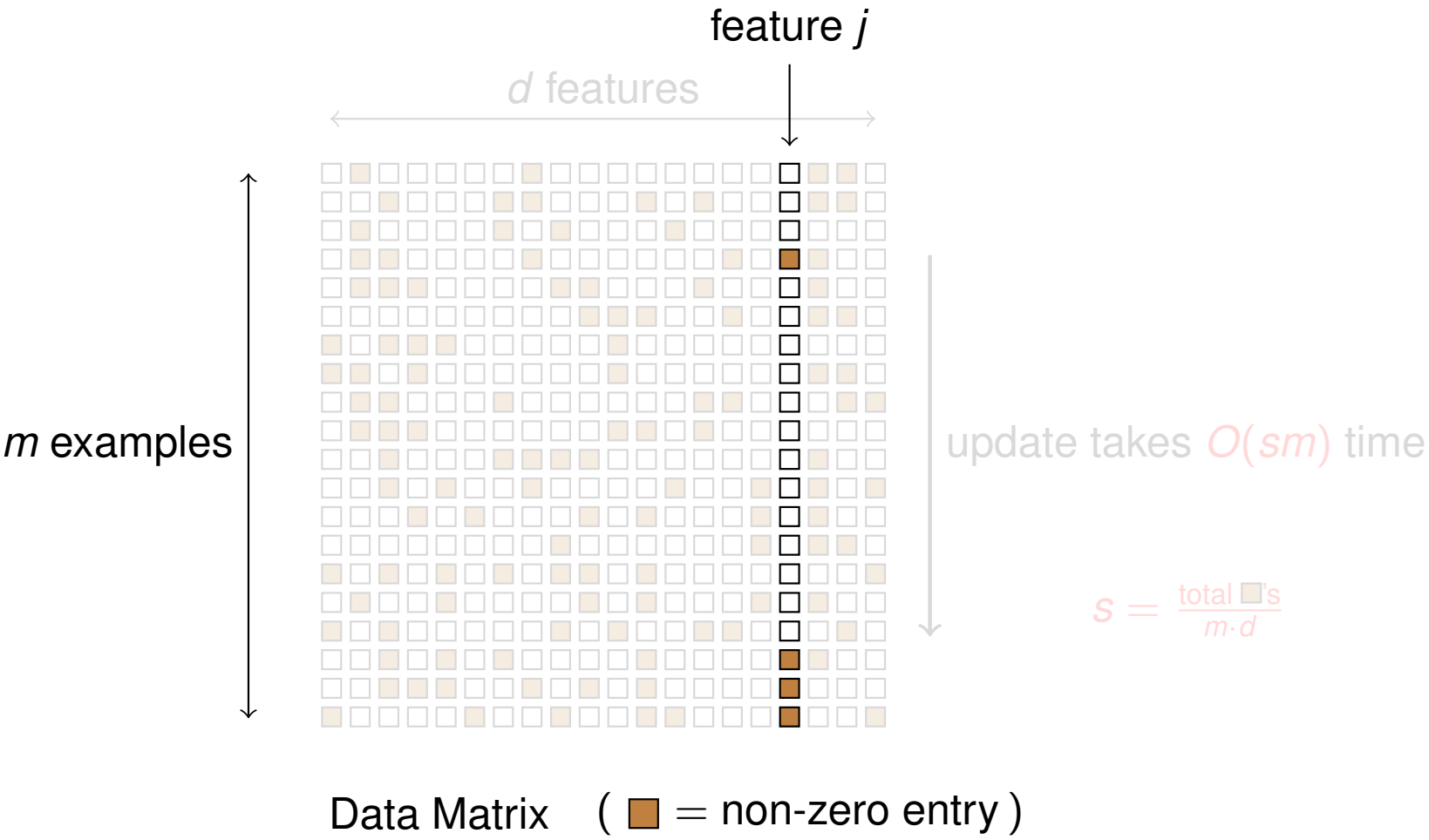
$g_j = (\nabla R(\mathbf{w}))_j$ — $j$th entry in the gradient

$w_j = \max\{0, w_j - g_j/\beta\}$ — to keep $w_j \geq 0$

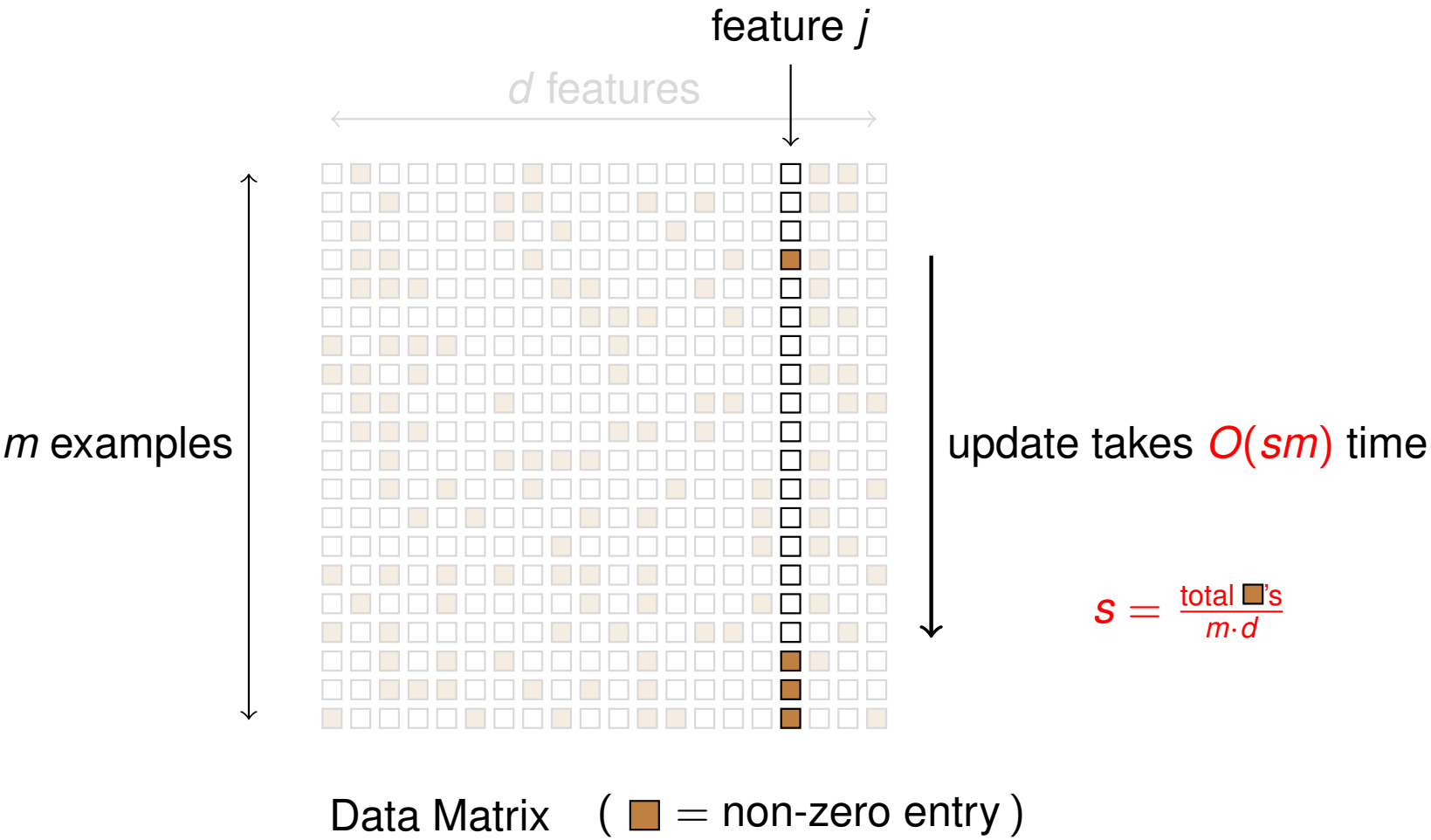
# Implementing Updates Efficiently



# Implementing Updates Efficiently



# Implementing Updates Efficiently



# Run-time Guarantee

- Simple selection rule
- Parameter free
- No Hessian computations; no line searches
- Yet, best convergence guarantee: time to achieve (expected) $\epsilon$-accuracy is

$$O\left(\frac{m\, d\, \beta\, \|\mathbf{w}^\star\|_2^2}{\epsilon}\right)$$

- We save a factor of $d$ compared to Tseng-Yun's (2009) bound

$$O\left(\frac{m\, d^2\, \beta\, \|\mathbf{w}^\star\|_2^2}{\epsilon}\right)$$

# Run-time Guarantee

- Simple selection rule
- Parameter free
- No Hessian computations; no line searches
- Yet, best convergence guarantee: time to achieve (expected) $\epsilon$-accuracy is

$$O\left(\frac{m\,d\,\beta\,\|\mathbf{w}^\star\|_2^2}{\epsilon}\right)$$

- We save a factor of $d$ compared to Tseng-Yun's (2009) bound

$$O\left(\frac{m\,d^2\,\beta\,\|\mathbf{w}^\star\|_2^2}{\epsilon}\right)$$

# Run-time Guarantee

- Simple selection rule
- Parameter free
- No Hessian computations; no line searches
- Yet, best convergence guarantee: time to achieve (expected) $\epsilon$-accuracy is

$$O\left(\frac{m\,d\,\beta\,\|\mathbf{w}^\star\|_2^2}{\epsilon}\right)$$

- We save a factor of $d$ compared to Tseng-Yun's (2009) bound

$$O\left(\frac{m\,d^2\,\beta\,\|\mathbf{w}^\star\|_2^2}{\epsilon}\right)$$

# Proof Idea

- A common proof strategy is to find a *potential* that decreases along the way
- Common potentials
  - Distance from the minimum $\|\mathbf{w} - \mathbf{w}^\star\|_2^2$
  - Objective function itself $R(\mathbf{w})$
- Analysis of SCD simplifies when we use the "double potential"

$$\frac{\beta}{2}\|\mathbf{w} - \mathbf{w}^\star\|_2^2 + R(\mathbf{w})$$

# Proof Idea

- A common proof strategy is to find a *potential* that decreases along the way
- Common potentials
  - Distance from the minimum $\|\mathbf{w} - \mathbf{w}^\star\|_2^2$
  - Objective function itself $R(\mathbf{w})$
- Analysis of SCD simplifies when we use the "double potential"

$$\frac{\beta}{2}\|\mathbf{w} - \mathbf{w}^\star\|_2^2 + R(\mathbf{w})$$

# Outline

# Stochastic Gradient Descent

$$\nabla \mathcal{L}_m(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i) \mathbf{x}_i$$

- Get unbiased estimate of gradient using one *random* example
- Use it to perform a gradient descent step
- Avoids incurring $m$ dependence
- Shown to be state-of-the-art for large-dataset regime

# SGD for L1

$$\nabla R(\mathbf{w}) = \nabla \mathcal{L}_m(\mathbf{w}) + \lambda \text{sign}(\mathbf{w})$$

- SGD applied directly fails to give sparsity
- One fix: work with constrained formulation (Duchi et al., 2008)

$$\min_{\mathbf{w}} \mathcal{L}_m(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq B$$

- Second fix: truncate $w_i$ to 0 if it crosses 0 during SGD update (Langford et al., 2009)
- SMIDAS combines Mirror Descent with Langford et al.'s *truncated gradient* idea

# SGD for L1

$$\nabla R(\mathbf{w}) = \nabla \mathcal{L}_m(\mathbf{w}) + \lambda \text{sign}(\mathbf{w})$$

- SGD applied directly fails to give sparsity
- One fix: work with constrained formulation (Duchi et al., 2008)

$$\min_{\mathbf{w}} \mathcal{L}_m(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq B$$

- Second fix: truncate $w_i$ to 0 if it crosses 0 during SGD update (Langford et al., 2009)
- SMIDAS combines Mirror Descent with Langford et al.'s *truncated gradient* idea

# SGD for L1

$$\nabla R(\mathbf{w}) = \nabla \mathcal{L}_m(\mathbf{w}) + \lambda \mathrm{sign}(\mathbf{w})$$

- SGD applied directly fails to give sparsity
- One fix: work with constrained formulation (Duchi et al., 2008)

$$\min_{\mathbf{w}} \mathcal{L}_m(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq B$$

- Second fix: truncate $w_i$ to 0 if it crosses 0 during SGD update (Langford et al., 2009)
- SMIDAS combines Mirror Descent with Langford et al.'s *truncated gradient* idea

# SGD for L1

$$\nabla R(\mathbf{w}) = \nabla \mathcal{L}_m(\mathbf{w}) + \lambda \text{sign}(\mathbf{w})$$

- SGD applied directly fails to give sparsity
- One fix: work with constrained formulation (Duchi et al., 2008)

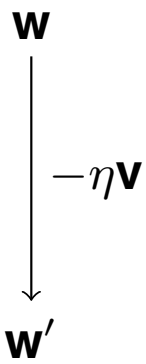
$$\min_{\mathbf{w}} \mathcal{L}_m(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq B$$

- Second fix: truncate $w_i$ to 0 if it crosses 0 during SGD update (Langford et al., 2009)
- SMIDAS combines Mirror Descent with Langford et al.'s *truncated gradient* idea

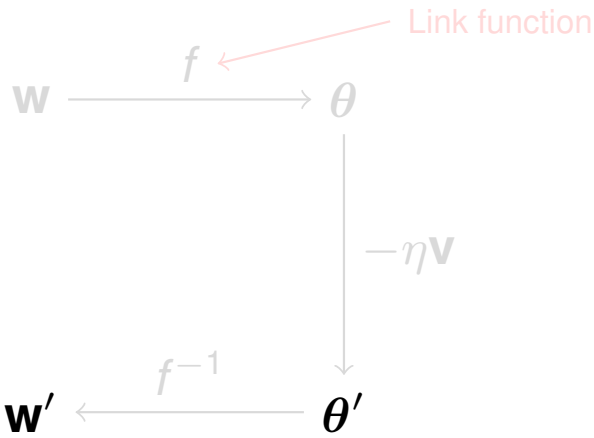
# Mirror Descent

$\mathbf{v}$ = (estimate of) gradient

*Gradient Descent*




*Mirror Descent*

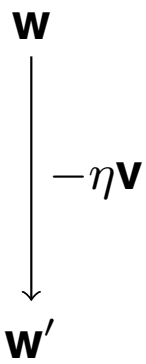



We'll use the $q$-norm link function, i.e. $f(\mathbf{w}) = \nabla ||\mathbf{w}||_q^2$

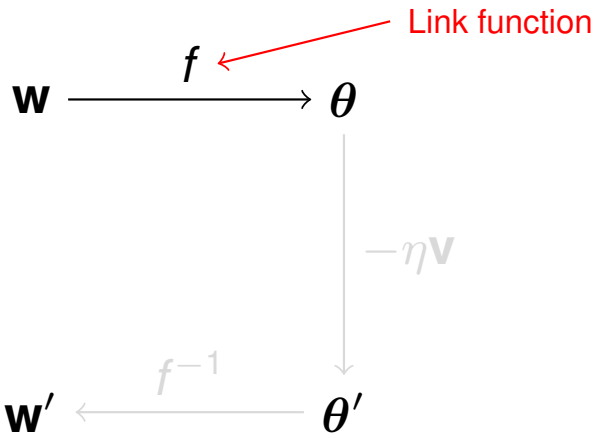
# Mirror Descent

$\mathbf{v}$ = (estimate of) gradient

*Gradient Descent*




*Mirror Descent*

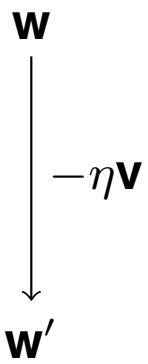



We'll use the $q$-norm link function, i.e. $f(\mathbf{w}) = \nabla \|\mathbf{w}\|_q^2$

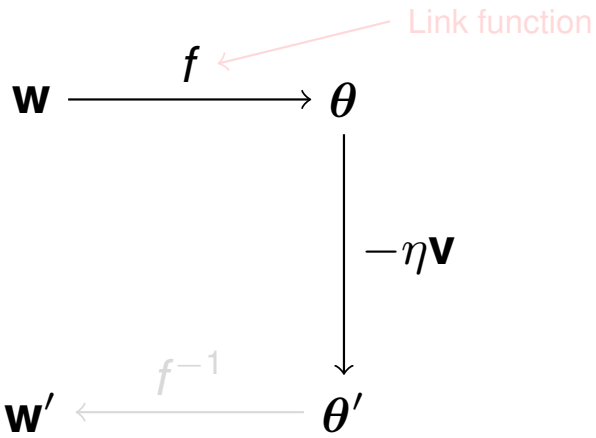
# Mirror Descent

$\mathbf{v}$ = (estimate of) gradient

*Gradient Descent*




*Mirror Descent*

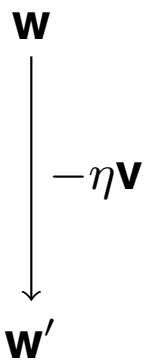



We'll use the $q$-norm link function, i.e. $f(\mathbf{w}) = \nabla \|\mathbf{w}\|_q^2$

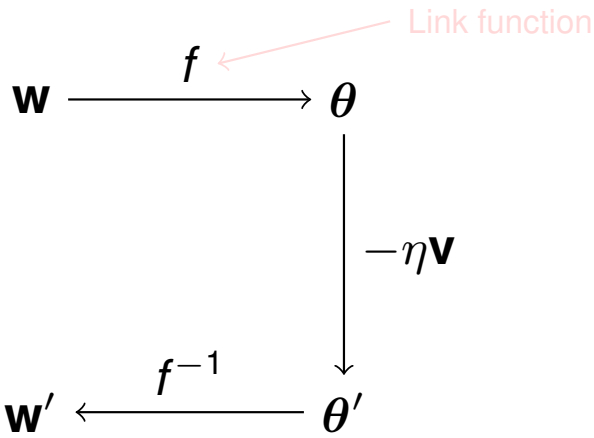
# Mirror Descent

$\mathbf{v}$ = (estimate of) gradient

*Gradient Descent*




*Mirror Descent*

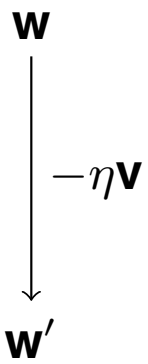



We'll use the $q$-norm link function, i.e. $f(\mathbf{w}) = \nabla \|\mathbf{w}\|_q^2$

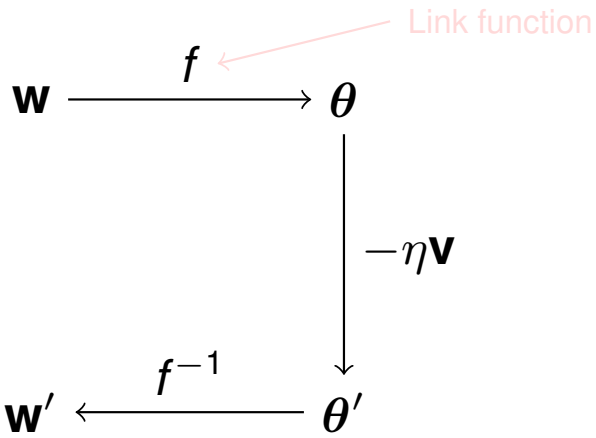
# Mirror Descent

$\mathbf{v}$ = (estimate of) gradient

*Gradient Descent*




*Mirror Descent*




We'll use the $q$-norm link function, i.e. $f(\mathbf{w}) = \nabla \|\mathbf{w}\|_q^2$

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}, \mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$
[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}, \mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$
[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}$, $\mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$

[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}$, $\mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$
[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}$, $\mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$

[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}$, $\mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$

[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

$\theta = \mathbf{0}$, $\mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$

[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

# Stochastic MIrror Descent Algorithm made Sparse

## SMIDAS

Parameter: $\eta > 0$

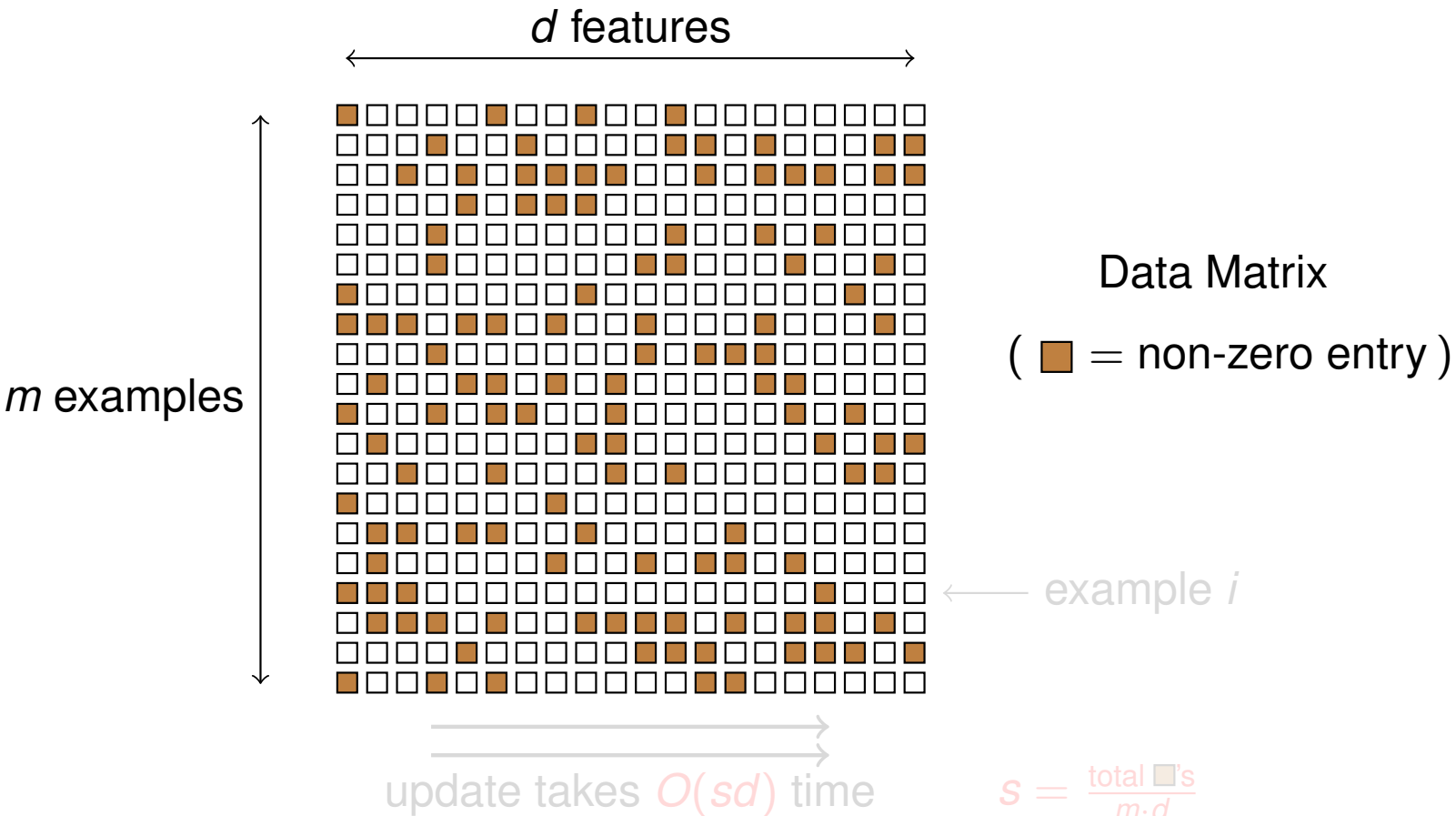
$\theta = \mathbf{0}$, $\mathbf{w} = f^{-1}(\theta)$
FOR $t = 1, 2, \ldots$

[GRADIENT ESTIMATION STEP]
Choose $i$ *randomly* from $\{1, \ldots, m\}$
$\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\mathbf{x}_i$

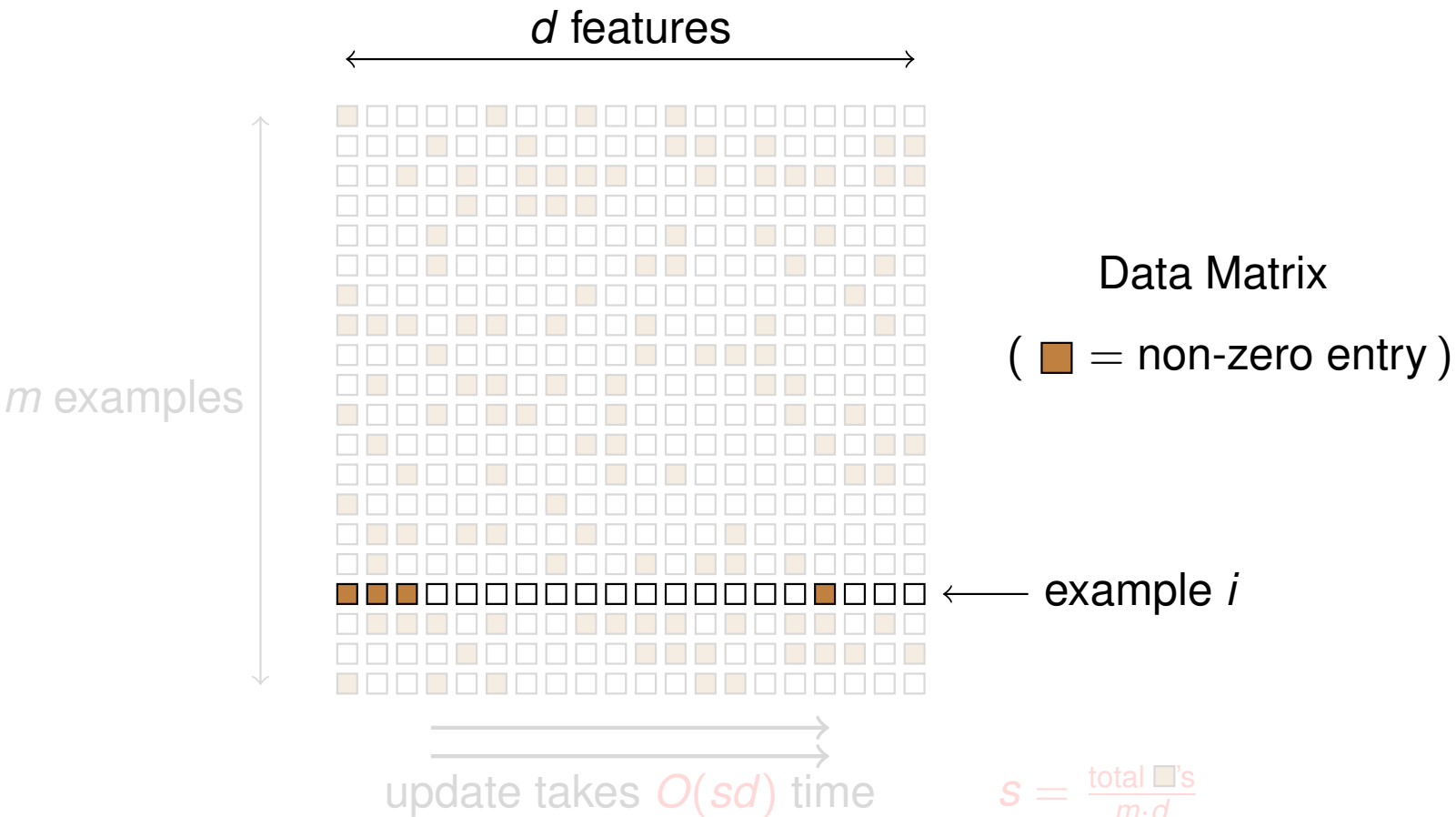
[TRUNCATED GRADIENT STEP]
$\tilde{\theta} = \theta - \eta \mathbf{v}$
$\forall j,\ \theta_j = \mathrm{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$ truncate if 0 is crossed

[MIRROR STEP]
$\mathbf{w} = f^{-1}(\theta)$ preserves sparsity pattern!

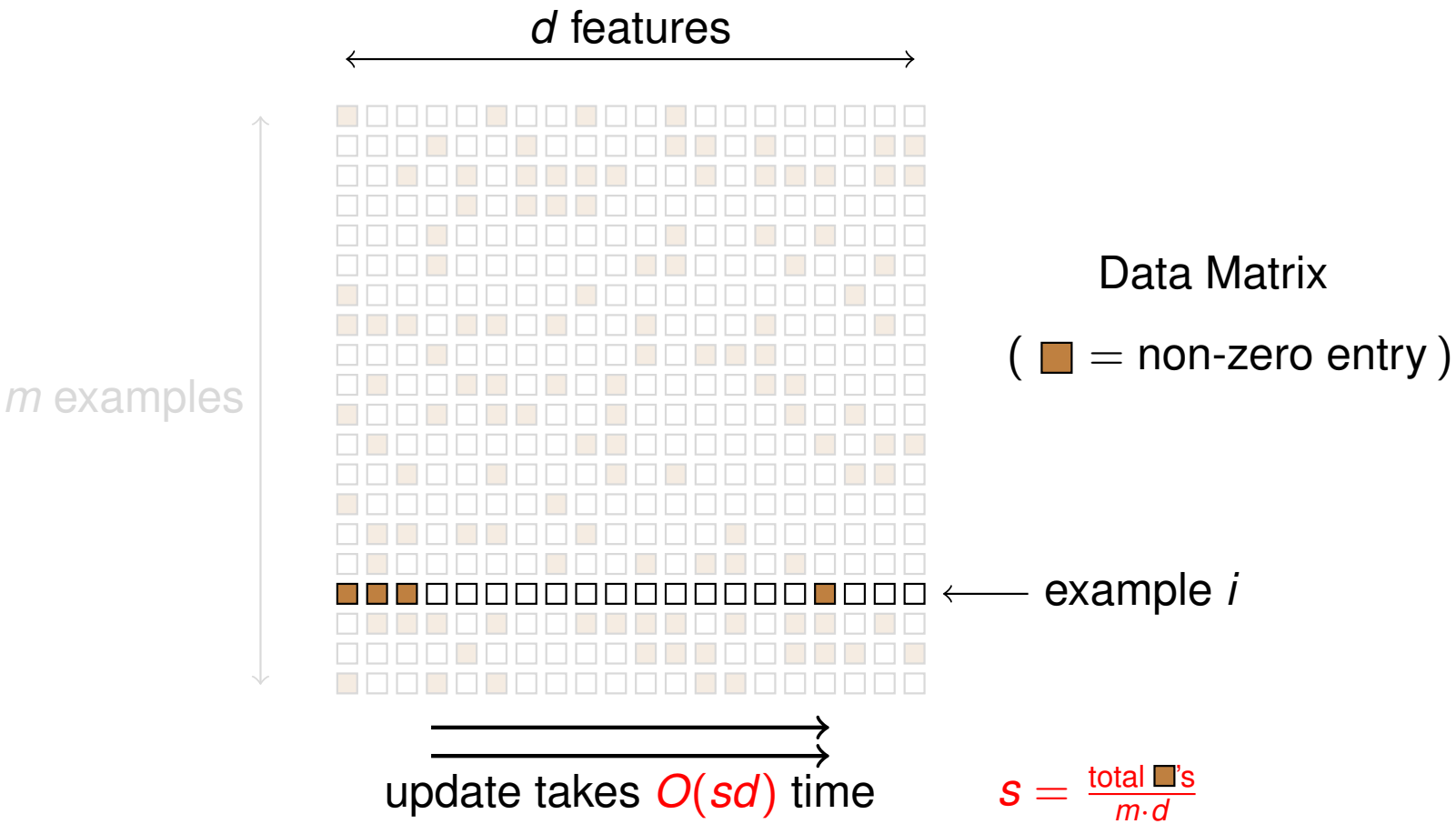
# Implementing Updates Efficiently



# Implementing Updates Efficiently



# Implementing Updates Efficiently



# Run-time Guarantee

- Time to achieve (expected) $\epsilon$-accuracy is ($\|x_i\|_\infty \leq X_\infty$)

$$O\left(\frac{d\ \log(d)\ X_\infty^2\ \|\mathbf{w}^\star\|_1^2}{\epsilon^2}\right) \qquad (B1)$$

- For Truncated Gradient (Langford et al., 2009), bound is ($X_2^2$ = average squared 2-norm of $x_i$'s)

$$O\left(\frac{d\ X_2^2\ \|\mathbf{w}^\star\|_2^2}{\epsilon^2}\right) \qquad (B2)$$

- $\mathbf{w}^\star$ sparse, $\mathbf{x}_i$'s dense: (B1) is better
- $\mathbf{x}_i$'s sparse, $\mathbf{w}^\star$ dense: (B2) is better

# Run-time Guarantee

- Time to achieve (expected) $\epsilon$-accuracy is ($\|x_i\|_\infty \leq X_\infty$)

$$O\left(\frac{d\ \log(d)\ X_\infty^2\ \|\mathbf{w}^\star\|_1^2}{\epsilon^2}\right) \qquad (B1)$$

- For Truncated Gradient (Langford et al., 2009), bound is ($X_2^2$ = average squared 2-norm of $x_i$'s)

$$O\left(\frac{d\ X_2^2\ \|\mathbf{w}^\star\|_2^2}{\epsilon^2}\right) \qquad (B2)$$

- $\mathbf{w}^\star$ sparse, $\mathbf{x}_i$'s dense: (B1) is better
- $\mathbf{x}_i$'s sparse, $\mathbf{w}^\star$ dense: (B2) is better

# Run-time Guarantee

- Time to achieve (expected) $\epsilon$-accuracy is ($\|x_i\|_\infty \leq X_\infty$)

$$O\left(\frac{d\ \log(d)\ X_\infty^2\ \|\mathbf{w}^\star\|_1^2}{\epsilon^2}\right) \qquad (B1)$$

- For Truncated Gradient (Langford et al., 2009), bound is ($X_2^2$ = average squared 2-norm of $x_i$'s)

$$O\left(\frac{d\ X_2^2\ \|\mathbf{w}^\star\|_2^2}{\epsilon^2}\right) \qquad (B2)$$

- $\mathbf{w}^\star$ sparse, $\mathbf{x}_i$'s dense: (B1) is better
- $\mathbf{x}_i$'s sparse, $\mathbf{w}^\star$ dense: (B2) is better

# Outline

# Algorithms for Experiments

- SCD: Stochastic Coordinate Descent
- DETCD (Wu & Lange): Deterministic version that chooses coordinate based on guaranteed decrease of objective
- SMIDAS
- TruncGrad (Langford et al.): Truncated Gradient (with $g = \theta = \infty$)

Last two need a parameter $\eta$. We report results for best $\eta$ (chosen over an exponentially spaced grid)

# Algorithms for Experiments

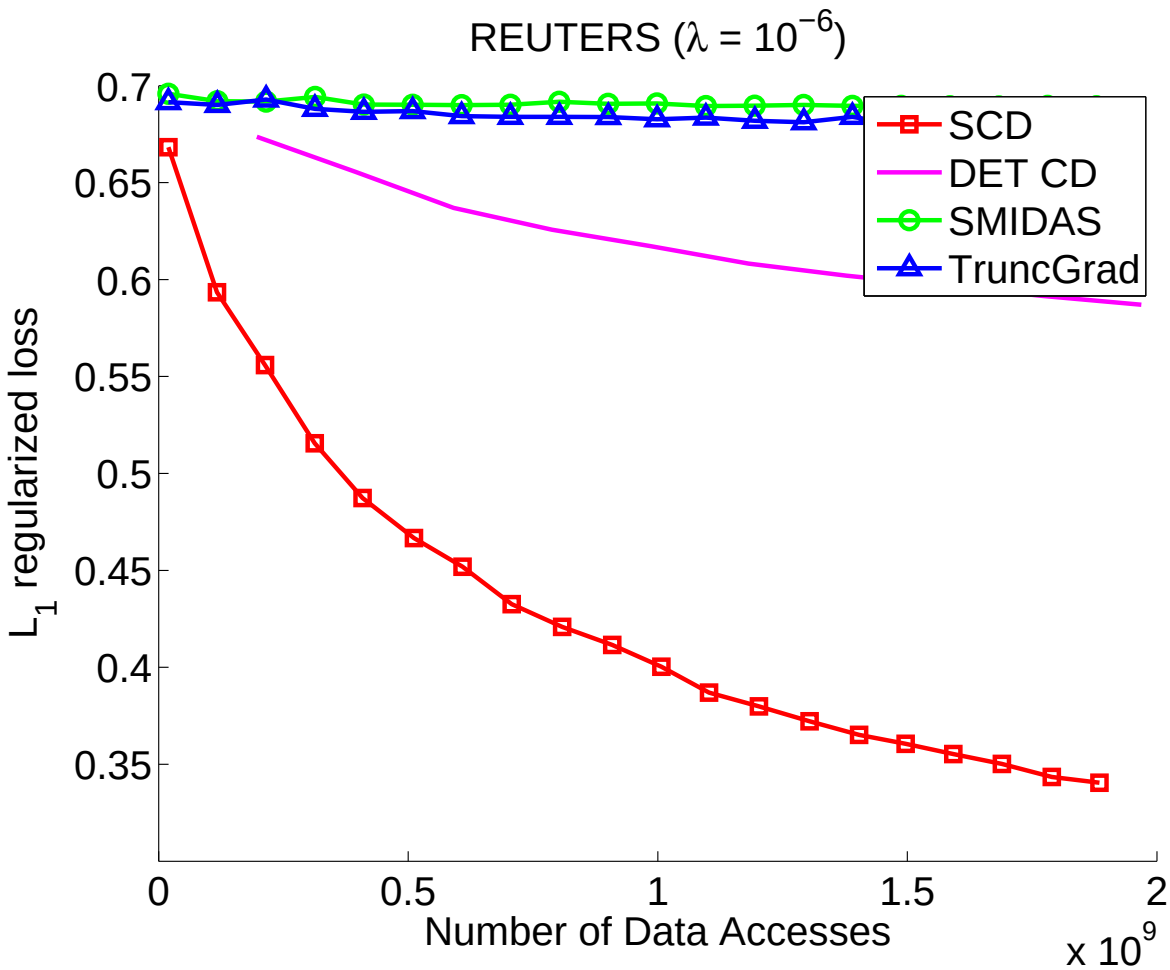
- SCD: Stochastic Coordinate Descent
- DETCD (Wu & Lange): Deterministic version that chooses coordinate based on guaranteed decrease of objective
- SMIDAS
- TruncGrad (Langford et al.): Truncated Gradient (with $g = \theta = \infty$)

Last two need a parameter $\eta$. We report results for best $\eta$ (chosen over an exponentially spaced grid)

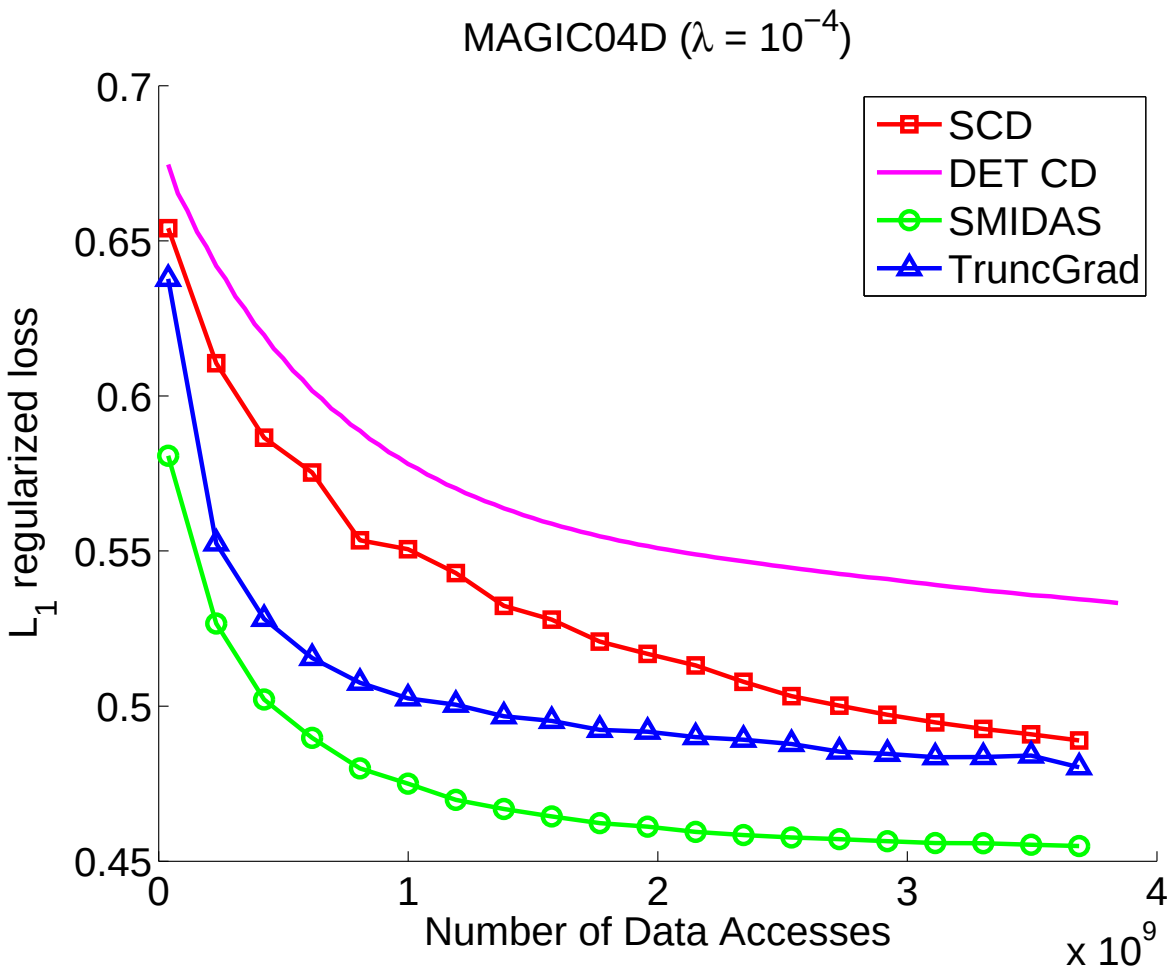
# Datasets

- Reuters
  - Dataset derived from Reuters RCV1 collection
  - 806,791 examples
  - 378,452 features
  - Sparsity level of instances is 0.03%
- Magic04D
  - Magic04 dataset from UCI + 1000 random features ($\pm 1$ with prob. 0.5)
  - 19,020 examples
  - 10+1000 = 1010 features
  - Sparsity level of instances is 100% (i.e. not sparse at all)

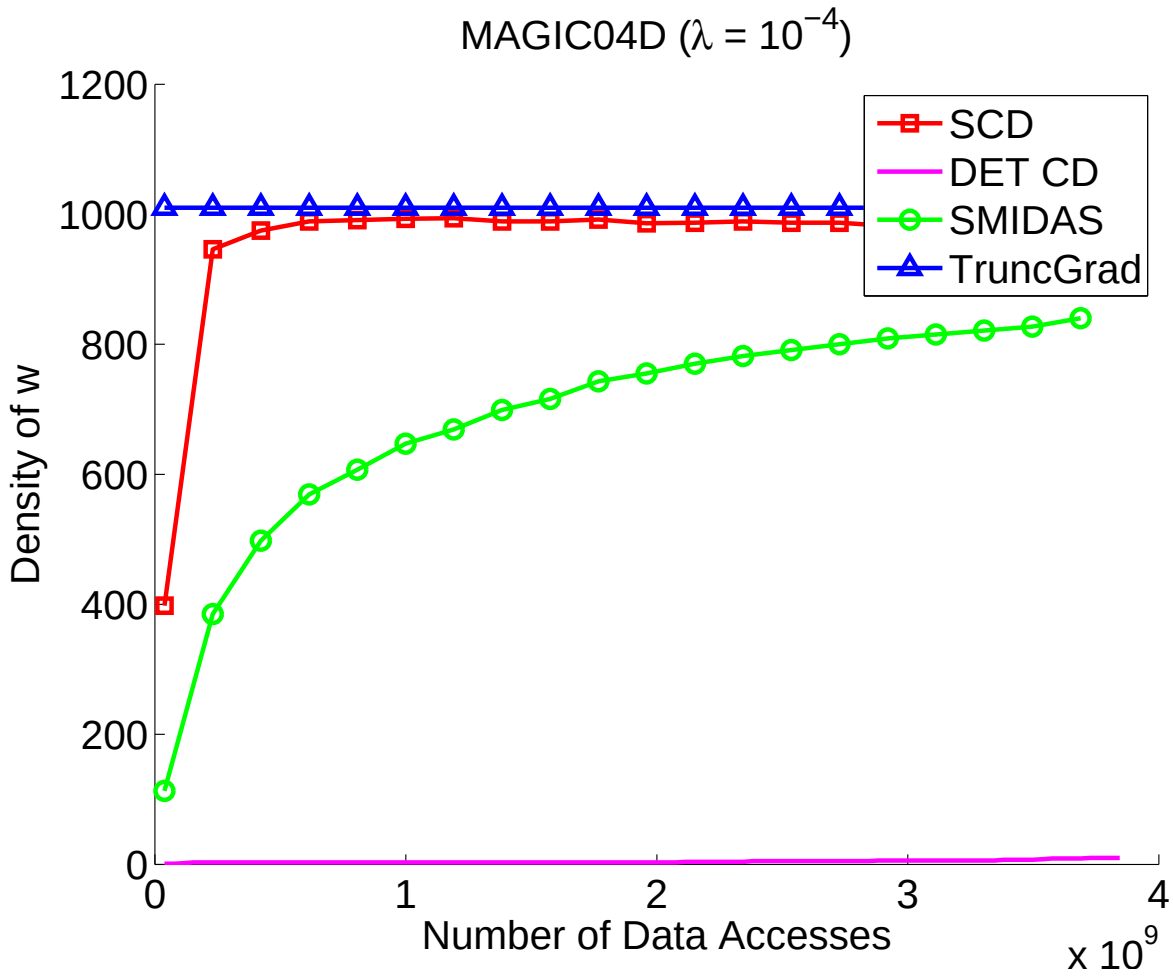
# Reuters


REUTERS (λ = 10^−6)
SCD
DET CD
SMIDAS
TruncGrad
L1 regularized loss
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0
0.5
1
1.5
2
Number of Data Accesses
x 10^9

# Magic04D


MAGIC04D (λ = 10^−4)
SCD
DET CD
SMIDAS
TruncGrad
L1 regularized loss
0.7
0.65
0.6
0.55
0.5
0.45
0
1
2
3
4
Number of Data Accesses
x 10^9

# Magic04D


MAGIC04D (λ = 10^−4)
Density of w
Number of Data Accesses
x 10^9
SCD
DET CD
SMIDAS
TruncGrad
0
200
400
600
800
1000
1200
0
1
2
3
4

# Outline

# Summary

- SCD & SMIDAS both perform lots of simple updates + randomization
- Contrast with IP methods: few but sophisticated updates
- Despite simplicity, get (provably) good convergence rates in terms of $m, d$
- Experiments suggest SCD & SMIDAS have potential to be successful in practice