南京大學 NANJING UNIVERSITY

Mining Big Data

Lijun Zhang
zlj@nju.edu.cn

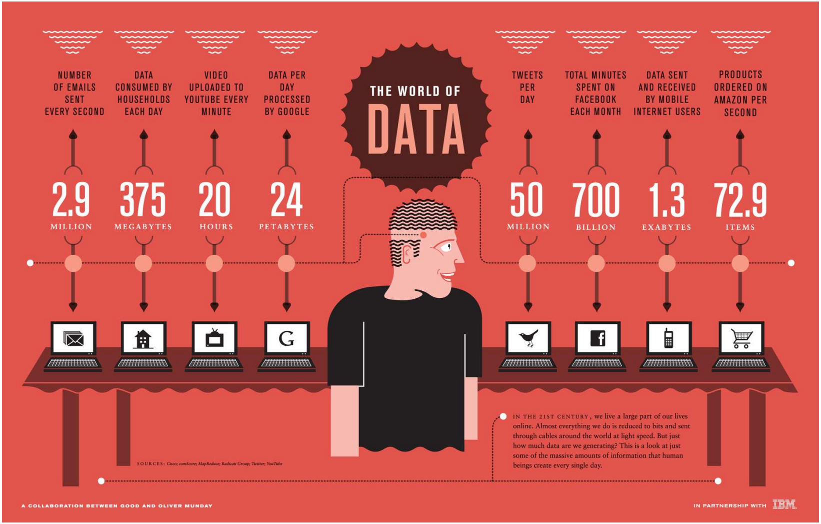
Nanjing University, China

December 23, 2015

Outline

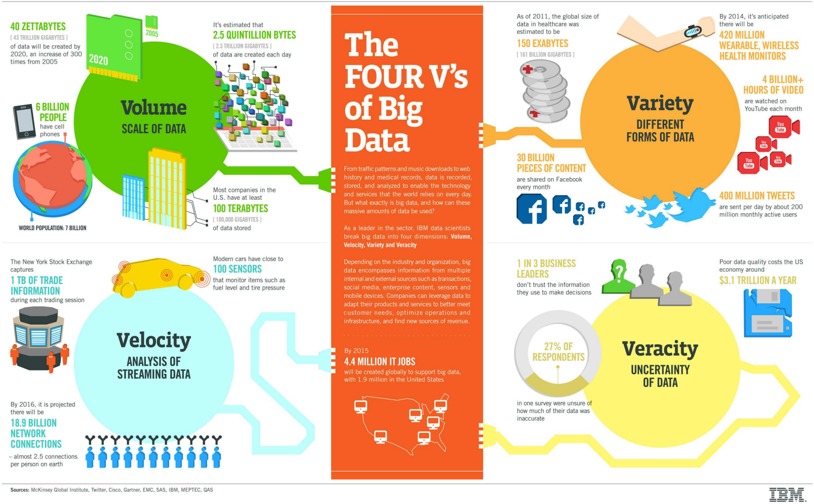
Outline

Big Data



https://infographiclist.files.wordpress.com/2011/09/world-of-data.jpeg

The Four V's of Big Data



http://www.ibmbigdatahub.com/sites/default/files/infographic_file/4-Vs-of-big-data.jpg

Outline

1. Introduction
   - Big Data
   - Supervised Learning
2. Stochastic Optimization
   - Time Reduction
   - Space Reduction
3. Distributed Optimization
   - Distributed Gradient Descent
   - ADMM
4. Online Learning
   - Full-Information Setting
   - Bandit Setting
5. Summary

Supervised Learning (I)

Training

Input

- A set of training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
  - Classification v.s. Regression
  - Batch Setting v.s. Online Setting

Output

- A function $g(\cdot)$ such that $y_i \approx g(\mathbf{x}_i)$, $\forall i$

Testing

Given a testing point $\mathbf{x}$, predict its label by $g(\mathbf{x})$

Assumption

Training and testing data are independent and identically distributed (i.i.d.)

Supervised Learning (I)

Training

Input

- A set of training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
  - Classification v.s. Regression
  - Batch Setting v.s. Online Setting

Output

- A function $g(\cdot)$ such that $y_i \approx g(\mathbf{x}_i)$, $\forall i$

Testing

Given a testing point $\mathbf{x}$, predict its label by $g(\mathbf{x})$

Assumption

Training and testing data are independent and identically distributed (i.i.d.)

Supervised Learning (I)

Training

Input

- A set of training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
  - Classification v.s. Regression
  - Batch Setting v.s. Online Setting

Output

- A function $g(\cdot)$ such that $y_i \approx g(\mathbf{x}_i)$, $\forall i$

Testing

Given a testing point $\mathbf{x}$, predict its label by $g(\mathbf{x})$

Assumption

Training and testing data are independent and identically distributed (i.i.d.)

Supervised Learning (II)

Loss Function

Measure the discrepancy between y and $g(\mathbf{x})$

- Binary loss: $\ell(u,v) = \mathbf{1}(uv < 0)$
- Hinge loss: $\ell(u,v) = \max(0, 1 - uv)$
- Squared loss: $\ell(u,v) = (u - v)^2$

Empirical Risk

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, g(\mathbf{x}_i))$$

Risk

$$\mathrm{E}_{(\mathbf{x},y)}\left[\ell(y, g(\mathbf{x}))\right]$$

Our goal is to minimize the risk instead of empirical risk.

Supervised Learning (II)

Loss Function

Measure the discrepancy between y and $g(\mathbf{x})$

- Binary loss: $\ell(u, v) = \mathbf{1}(uv < 0)$
- Hinge loss: $\ell(u, v) = \max(0, 1 - uv)$
- Squared loss: $\ell(u, v) = (u - v)^2$

Empirical Risk

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, g(\mathbf{x}_i))$$

Risk

$$\mathrm{E}_{(\mathbf{x},y)}\left[\ell(y, g(\mathbf{x}))\right]$$

Our goal is to minimize the risk instead of empirical risk.

Supervised Learning (II)

Loss Function

Measure the discrepancy between y and $g(\mathbf{x})$

- Binary loss: $\ell(u, v) = \mathbf{1}(uv < 0)$
- Hinge loss: $\ell(u, v) = \max(0, 1 - uv)$
- Squared loss: $\ell(u, v) = (u - v)^2$

Empirical Risk

$$\frac{1}{n}\sum_{i=1}^{n}\ell(y_i, g(\mathbf{x}_i))$$

Risk

$$\mathrm{E}_{(\mathbf{x},y)}\left[\ell(y, g(\mathbf{x}))\right]$$

Our goal is to minimize the risk instead of empirical risk.

Outline

What is Stochastic Optimization? (I)

Definition 1: A Special Objective [Nemirovski et al., 2009]

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \mathrm{E}_{\xi}\left[\ell(\mathbf{w},\xi)\right] = \int_{\Xi} \ell(\mathbf{w},\xi) dP(\xi)$$

where ξ is a random variable

- It is possible to generate an i.i.d. sample $\xi_1, \xi_2, \ldots$

Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \mathrm{E}_{(\mathbf{x},y)}\left[\ell(y, \mathbf{x}^\top \mathbf{w})\right]$$

$\ell(\cdot,\cdot)$ is a loss function, e.g., hinge loss $\ell(u,v) = \max(0, 1-uv)$

- Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are i.i.d.

What is Stochastic Optimization? (I)

Definition 1: A Special Objective [Nemirovski et al., 2009]

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \mathrm{E}_{\xi}\left[\ell(\mathbf{w},\xi)\right] = \int_{\Xi} \ell(\mathbf{w},\xi) dP(\xi)$$

where ξ is a random variable

- It is possible to generate an i.i.d. sample $\xi_1, \xi_2, \ldots$

Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \mathrm{E}_{(\mathbf{x},y)}\left[\ell(y, \mathbf{x}^\top \mathbf{w})\right]$$

$\ell(\cdot,\cdot)$ is a loss function, e.g., hinge loss $\ell(u,v) = \max(0, 1-uv)$

- Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are i.i.d.

What is Stochastic Optimization? (II)

Definition 2: A Special Access Model [Hazan and Kale, 2011]

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w})$$

- There exists a stochastic oracle that produces unbiased gradient $\mathbf{m}(\cdot)$

$$\mathrm{E}[\mathbf{m}(\mathbf{w})] = \nabla f(\mathbf{w})$$

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

- Sampling a $(\mathbf{x}_t, y_t)$ from $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ uniformly at random

$$\mathrm{E}[\nabla \ell(y_t, \mathbf{x}_t^\top \mathbf{w})] = \nabla \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

What is Stochastic Optimization? (II)

Definition 2: A Special Access Model [Hazan and Kale, 2011]

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w})$$

- There exists a stochastic oracle that produces unbiased gradient $\mathbf{m}(\cdot)$

$$\mathrm{E}[\mathbf{m}(\mathbf{w})] = \nabla f(\mathbf{w})$$

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

- Sampling a $(\mathbf{x}_t, y_t)$ from $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ uniformly at random

$$\mathrm{E}[\nabla \ell(y_t, \mathbf{x}_t^\top \mathbf{w})] = \nabla \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

Outline

Time Reduction (I)

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

- n is the number of training data
- d is the dimensionality

Deterministic Optimization—Gradient Descent (GD)

1: **for** $t = 1, 2, \ldots, T$ **do**

2: $\quad \mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\frac{1}{n}\sum_{i=1}^{n} \nabla \ell(y_i, \mathbf{x}_i^\top \mathbf{w}_t)\right)$

3: $\quad \mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$

4: **end for**

The Challenge

- Time complexity per iteration: $O(nd) + O(poly(d))$

Time Reduction (I)

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \mathbf{x}_i^\top\mathbf{w})$$

- n is the number of training data
- d is the dimensionality

Deterministic Optimization—Gradient Descent (GD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\frac{1}{n}\sum_{i=1}^{n} \nabla\ell(y_i, \mathbf{x}_i^\top \mathbf{w}_t)\right)$
3: $\quad \mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
4: **end for**

The Challenge

- Time complexity per iteration: $O(nd) + O(poly(d))$

Time Reduction (I)

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

- n is the number of training data
- d is the dimensionality

Deterministic Optimization—Gradient Descent (GD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\frac{1}{n}\sum_{i=1}^{n} \nabla\ell(y_i, \mathbf{x}_i^\top \mathbf{w}_t)\right)$
3: $\quad \mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
4: **end for**

The Challenge

- Time complexity per iteration: $O(nd) + O(poly(d))$

Time Reduction (II)

Empirical Risk Minimization

$$\min_{\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

Stochastic Optimization—Stochastic Gradient Descent (SGD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Sample a training example $(\mathbf{x}_t, y_t)$ uniformly at random
3: $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell(y_t, \mathbf{x}_t^\top \mathbf{w}_t)$
4: $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
5: **end for**

The Advantage—*Time Reduction*

- Time complexity per iteration: $O(d) + O(poly(d))$

Time Reduction (II)

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

Stochastic Optimization—Stochastic Gradient Descent (SGD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Sample a training example $(\mathbf{x}_t, y_t)$ uniformly at random
3: $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \nabla\ell(y_t, \mathbf{x}_t^\top \mathbf{w}_t)$
4: $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
5: **end for**

The Advantage—*Time Reduction*

- Time complexity per iteration: $O(d) + O(poly(d))$

Time Reduction (II)

Empirical Risk Minimization

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \mathbf{x}_i^\top \mathbf{w})$$

Stochastic Optimization—Stochastic Gradient Descent (SGD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Sample a training example $(\mathbf{x}_t, y_t)$ uniformly at random
3: $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \nabla\ell(y_t, \mathbf{x}_t^\top \mathbf{w}_t)$
4: $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
5: **end for**

The Advantage—*Time Reduction*

- Time complexity per iteration: $O(d) + O(poly(d))$

Outline

Space Reduction (I)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

- Matrix completion, multi-class classification
- Both m and n can be very large
- The optimal solution W_* is low-rank

Collaborative Filtering—Matrix Completion

$$\min_{W \in \mathbb{R}^{m \times n}} F(W) = \sum_{(i,j) \in \Omega} (W_{ij} - M_{ij})^2 + \lambda \|W\|_*$$

- There are m users and n items
- $M \in \mathbb{R}^{m \times n}$ is the underlying user-item rating matrix
- Ω is the set of observed indices

Space Reduction (I)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

- Matrix completion, multi-class classification
- Both m and n can be very large
- The optimal solution W_* is low-rank

Collaborative Filtering—Matrix Completion

$$\min_{W \in \mathbb{R}^{m \times n}} F(W) = \sum_{(i,j) \in \Omega} (W_{ij} - M_{ij})^2 + \lambda \|W\|_*$$

- There are m users and n items
- $M \in \mathbb{R}^{m \times n}$ is the underlying user-item rating matrix
- Ω is the set of observed indices

Space Reduction (II)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

- Matrix completion, multi-class classification
- Both m and n can be very large
- The optimal solution W_* is low-rank

Deterministic Optimization—Gradient Descent (GD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad W'_{t+1} = W_t - \eta_t \nabla F(W_t)$
3: $\quad W_{t+1} = \Pi_{\mathcal{W}}(W'_{t+1})$
4: **end for**

The Challenge

- Space complexity per iteration: $O(mn)$

Space Reduction (II)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

- Matrix completion, multi-class classification
- Both m and n can be very large
- The optimal solution W_* is low-rank

Deterministic Optimization—Gradient Descent (GD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad W'_{t+1} = W_t - \eta_t \nabla F(W_t)$
3: $\quad W_{t+1} = \Pi_{\mathcal{W}}(W'_{t+1})$
4: **end for**

The Challenge

- Space complexity per iteration: $O(mn)$

Space Reduction (II)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

- Matrix completion, multi-class classification
- Both m and n can be very large
- The optimal solution W_* is low-rank

Deterministic Optimization—Gradient Descent (GD)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad W'_{t+1} = W_t - \eta_t \nabla F(W_t)$
3: $\quad W_{t+1} = \Pi_{\mathcal{W}}(W'_{t+1})$
4: **end for**

The Challenge

- Space complexity per iteration: $O(mn)$

Space Reduction (III)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

Stochastic Proximal Gradient Descent (SPGD) [Zhang et al., 2015]

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Generate a low-rank stochastic gradient $\widehat{G}_t$ of $f(\cdot)$ at W_t

3:
$$W_{t+1} = \operatorname*{argmin}_{W \in \mathbb{R}^{m \times n}} \frac{1}{2} \left\| W - (W_t - \eta_t \widehat{G}_t) \right\|_F^2 + \eta_t \lambda \|W\|_*$$

4: **end for**

5: **return** W_{T+1}

The Advantage—*Space Reduction*

- Space complexity per iteration is low: $O((m+n)r)$

Space Reduction (III)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

Stochastic Proximal Gradient Descent (SPGD) [Zhang et al., 2015]

for $t = 1, 2, \ldots, T$ **do**

Generate a low-rank stochastic gradient $\widehat{G}_t$ of $f(\cdot)$ at W_t

$$W_{t+1} = \operatorname*{argmin}_{W \in \mathbb{R}^{m \times n}} \frac{1}{2} \left\| W - (W_t - \eta_t \widehat{G}_t) \right\|_F^2 + \eta_t \lambda \|W\|_*$$

end for

return W_{T+1}

The Advantage—*Space Reduction*

- Space complexity per iteration is low: $O((m+n)r)$

Space Reduction (III)

Nuclear Norm Regularized Optimization over Matrices

$$\min_{W \in \mathcal{W} \in \mathbb{R}^{m \times n}} F(W) = f(W) + \lambda \|W\|_*$$

Stochastic Proximal Gradient Descent (SPGD) [Zhang et al., 2015]

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Generate a low-rank stochastic gradient $\widehat{G}_t$ of $f(\cdot)$ at W_t

3:

$$W_{t+1} = \operatorname*{argmin}_{W \in \mathbb{R}^{m \times n}} \frac{1}{2} \left\| W - (W_t - \eta_t \widehat{G}_t) \right\|_F^2 + \eta_t \lambda \|W\|_*$$

4: **end for**

5: **return** W_{T+1}

The Advantage—*Space Reduction*

- Space complexity per iteration is low: $O((m+n)r)$

Limitations

Iteration Complexity

The number of iterations T to ensure

$$f(\mathbf{w}_T) - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \epsilon$$

Iteration Complexity of GD and SGD

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$	$O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\mu\epsilon}\right)$

Total Time Complexity of GD and SGD

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{n}{\sqrt{\epsilon}}\right)$	$O\left(n\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\mu\epsilon}\right)$

Limitations

Iteration Complexity

The number of iterations T to ensure

$$f(\mathbf{w}_T) - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \epsilon$$

Iteration Complexity of GD and SGD

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$	$O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\mu\epsilon}\right)$

Total Time Complexity of GD and SGD

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{n}{\sqrt{\epsilon}}\right)$	$O\left(n\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\mu\epsilon}\right)$

Limitations

Iteration Complexity

The number of iterations T to ensure

$$f(\mathbf{w}_T) - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \epsilon$$

Iteration Complexity of GD and SGD $\epsilon = 10^{-6}$

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$ 10^3	$O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ $6\sqrt{\kappa}$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$ 10^{12}	$O\left(\frac{1}{\mu\epsilon}\right)$ $\frac{10^6}{\mu}$

Total Time Complexity of GD and SGD

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{n}{\sqrt{\epsilon}}\right)$	$O\left(n\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\mu\epsilon}\right)$

Limitations

Iteration Complexity

The number of iterations T to ensure

$$f(\mathbf{w}_T) - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \epsilon$$

Iteration Complexity of GD and SGD $\epsilon = 10^{-6}$

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$ 10^3	$O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ $6\sqrt{\kappa}$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$ 10^{12}	$O\left(\frac{1}{\mu\epsilon}\right)$ $\frac{10^6}{\mu}$

Total Time Complexity of GD and SGD

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{n}{\sqrt{\epsilon}}\right)$	$O\left(n\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\mu\epsilon}\right)$

Limitations

Iteration Complexity

The number of iterations T to ensure

$$f(\mathbf{w}_T) - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \le \epsilon$$

Iteration Complexity of GD and SGD $\epsilon = 10^{-6}$

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$ 10^3	$O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ $6\sqrt{\kappa}$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$ 10^{12}	$O\left(\frac{1}{\mu\epsilon}\right)$ $\frac{10^6}{\mu}$

Total Time Complexity of GD and SGD $\epsilon = 10^{-6}$

	Convex & Smooth	Strongly Convex & Smooth
GD	$O\left(\frac{n}{\sqrt{\epsilon}}\right)$ $10^3 n$	$O\left(n\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ $6n\sqrt{\kappa}$
SGD	$O\left(\frac{1}{\epsilon^2}\right)$ 10^{12}	$O\left(\frac{1}{\mu\epsilon}\right)$ $\frac{10^6}{\mu}$

Outline

Introduction (I)

Empirical Risk Minimization in Distributed Setting

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \sum_{j=1}^{k}\sum_{i=1}^{n_j} \ell(y_i^j, \mathbf{w}^\top \mathbf{x}_i^j)$$

- $(\mathbf{x}_1^j, y_1^j) \ldots (\mathbf{x}_{n_j}^j, y_{n_j}^j)$ are training data in the j-th machine

$(\mathbf{x}_1^1, y_1^1)$
$\cdots$
$(\mathbf{x}_{n_1}^1, y_{n_1}^1)$

$(\mathbf{x}_1^2, y_1^2)$
$\cdots$
$(\mathbf{x}_{n_2}^k, y_{n_2}^k)$

$\cdots$

$(\mathbf{x}_1^k, y_1^k)$
$\cdots$
$(\mathbf{x}_{n_k}^k, y_{n_k}^k)$

Introduction (II)

General Distributed Optimization

$$\min_{\mathbf{w} \in \mathcal{W}} f(\mathbf{w}) = \sum_{j=1}^{k} f_j(\mathbf{w})$$

- For empirical risk minimization, we have

$$f_j(\mathbf{w}) = \sum_{i=1}^{n_j} \ell(y_i^j, \mathbf{w}^\top \mathbf{x}_i^j)$$

$f_1(\mathbf{w})$

$f_2(\mathbf{w})$

$\cdots$

$f_k(\mathbf{w})$

Outline

Distributed Gradient Descent

Gradient Descent

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\sum_{j=1}^{k} \nabla f_j(\mathbf{w}_t) \right)$
3: $\quad \mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
4: **end for**

Distributed Gradient Descent

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Server: send $\mathbf{w}_t$ to each worker
3: Each worker j: calculate $\nabla f_j(\mathbf{w}_t)$ and send it to server
4: Server: $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\sum_{j=1}^{k} \nabla f_j(\mathbf{w}_t) \right)$
5: Server: $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
6: **end for**

Distributed Gradient Descent

Gradient Descent

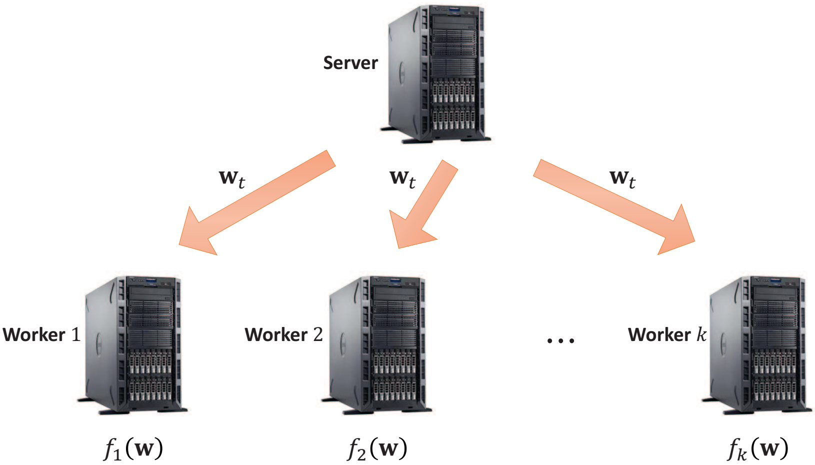
for $t = 1, 2, \ldots, T$ **do**
  $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\sum_{j=1}^{k} \nabla f_j(\mathbf{w}_t) \right)$
  $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
end for

Distributed Gradient Descent

for $t = 1, 2, \ldots, T$ **do**
  Server: send $\mathbf{w}_t$ to each worker
  Each worker j: calculate $\nabla f_j(\mathbf{w}_t)$ and send it to server
  Server: $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\sum_{j=1}^{k} \nabla f_j(\mathbf{w}_t) \right)$
  Server: $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$
end for

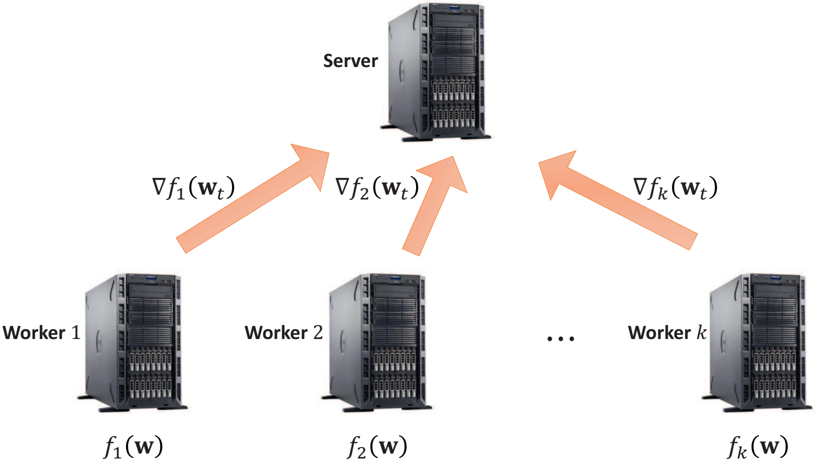
Step 1

Server: send $\mathbf{w}_t$ to each worker



Step 2

Each worker j: calculate $\nabla f_j(\mathbf{w}_t)$ and send it to server



Steps 3 and 4

- Server: $\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\sum_{j=1}^{k} \nabla f_j(\mathbf{w}_t) \right)$
- Server: $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$

Server

$$\mathbf{w}'_{t+1} = \mathbf{w}_t - \eta_t \left(\sum_{j=1}^{k} \nabla f_j(\mathbf{w}_t) \right)$$

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}(\mathbf{w}'_{t+1})$$

Outline

Reformulation of the Distributed Optimization (I)

General Distributed Optimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \sum_{j=1}^{k} f_j(\mathbf{w})$$

Global Consensus Problem

$$\min_{\mathbf{y}\in\mathcal{W},\{\mathbf{w}_j=\mathbf{y}\}_{j=1}^{k}} \sum_{j=1}^{k} f_j(\mathbf{w}_j)$$

The Lagrangian

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y} \rangle$$

Reformulation of the Distributed Optimization (I)

General Distributed Optimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \sum_{j=1}^{k} f_j(\mathbf{w})$$

Global Consensus Problem

$$\min_{\mathbf{y}\in\mathcal{W},\{\mathbf{w}_j=\mathbf{y}\}_{j=1}^{k}} \sum_{j=1}^{k} f_j(\mathbf{w}_j)$$

The Lagrangian

$$\max_{\lambda_1,\ldots,\lambda_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \lambda_j, \mathbf{w}_j - \mathbf{y}\rangle$$

Reformulation of the Distributed Optimization (I)

General Distributed Optimization

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) = \sum_{j=1}^{k} f_j(\mathbf{w})$$

Global Consensus Problem

$$\min_{\mathbf{y}\in\mathcal{W},\{\mathbf{w}_j=\mathbf{y}\}_{j=1}^{k}} \sum_{j=1}^{k} f_j(\mathbf{w}_j)$$

The Lagrangian

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle$$

Reformulation of the Distributed Optimization (II)

The Augmented Lagrangian [Roux et al., 2012]

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle + \frac{1}{2\gamma}\|\mathbf{w}_i - \mathbf{y}\|_2^2$$

Alternating Direction Method of Multipliers (ADMM)

for $t = 1, 2, \ldots, T$ **do**

Server: send $\mathbf{y}^t$ and λ_j^t to worker $j = 1, \ldots, k$

Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\lambda_j^t - \mathbf{w}\|_2^2,\ j = 1, \ldots, k$$

Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\lambda_j^t)\right)$

Server: $\lambda_j^{t+1} = \lambda_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right),\ j = 1, \ldots, k$

end for

Reformulation of the Distributed Optimization (II)

The Augmented Lagrangian [Roux et al., 2012]

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle + \frac{1}{2\gamma}\|\mathbf{w}_i - \mathbf{y}\|_2^2$$

Alternating Direction Method of Multipliers (ADMM)

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Server: send $\mathbf{y}^t$ and λ_j^t to worker $j = 1, \ldots, k$

3: Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_j^t - \mathbf{w}\|_2^2,\ j = 1, \ldots, k$$

4: Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$

5: Server: $\lambda_j^{t+1} = \lambda_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right), j = 1, \ldots, k$

6: **end for**

Reformulation of the Distributed Optimization (II)

The Augmented Lagrangian [Roux et al., 2012]

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle + \frac{1}{2\gamma}\|\mathbf{w}_i - \mathbf{y}\|_2^2$$

Alternating Direction Method of Multipliers (ADMM)

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Server: send $\mathbf{y}^t$ and λ_j^t to worker $j = 1, \ldots, k$

3: Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_j^t - \mathbf{w}\|_2^2,\ j = 1, \ldots, k$$

4: Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$

5: Server: $\lambda_j^{t+1} = \lambda_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right), j = 1, \ldots, k$

6: **end for**

Reformulation of the Distributed Optimization (II)

The Augmented Lagrangian [Roux et al., 2012]

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle + \frac{1}{2\gamma}\|\mathbf{w}_i - \mathbf{y}\|_2^2$$

Alternating Direction Method of Multipliers (ADMM)

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Server: send $\mathbf{y}^t$ and $\boldsymbol{\lambda}_j^t$ to worker $j = 1, \ldots, k$

3: Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_j^t - \mathbf{w}\|_2^2, \ j = 1, \ldots, k$$

4: Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$

5: Server: $\boldsymbol{\lambda}_j^{t+1} = \boldsymbol{\lambda}_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right), j = 1, \ldots, k$

6: **end for**

Reformulation of the Distributed Optimization (II)

The Augmented Lagrangian [Roux et al., 2012]

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle + \frac{1}{2\gamma}\|\mathbf{w}_i - \mathbf{y}\|_2^2$$

Alternating Direction Method of Multipliers (ADMM)

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Server: send $\mathbf{y}^t$ and $\boldsymbol{\lambda}_j^t$ to worker $j = 1, \ldots, k$

3: Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_j^t - \mathbf{w}\|_2^2, \; j = 1, \ldots, k$$

4: Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$

5: Server: $\boldsymbol{\lambda}_j^{t+1} = \boldsymbol{\lambda}_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right), j = 1, \ldots, k$

6: **end for**

Reformulation of the Distributed Optimization (II)

The Augmented Lagrangian [Roux et al., 2012]

$$\max_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_k} \min_{\mathbf{y}\in\mathcal{W},\mathbf{w}_1,\ldots,\mathbf{w}_k} \sum_{j=1}^{k} f_j(\mathbf{w}_j) - \langle \boldsymbol{\lambda}_j, \mathbf{w}_j - \mathbf{y}\rangle + \frac{1}{2\gamma}\|\mathbf{w}_i - \mathbf{y}\|_2^2$$

Alternating Direction Method of Multipliers (ADMM)

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Server: send $\mathbf{y}^t$ and $\boldsymbol{\lambda}_j^t$ to worker $j = 1, \ldots, k$

3: Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_j^t - \mathbf{w}\|_2^2,\ j = 1, \ldots, k$$

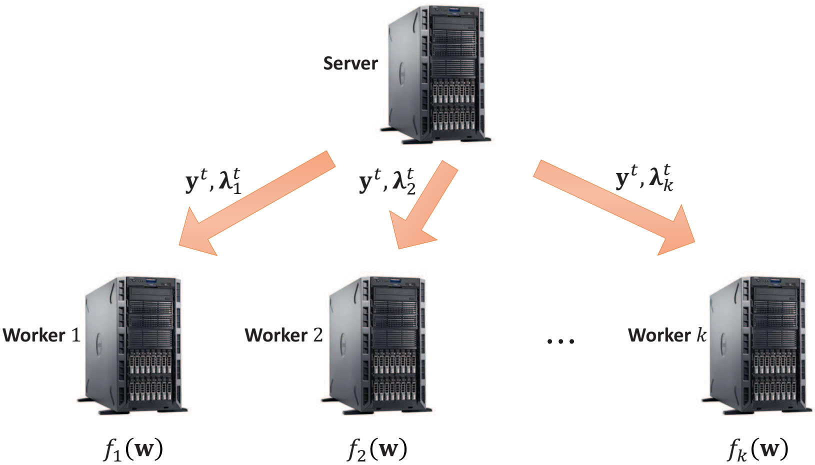
4: Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$

5: Server: $\boldsymbol{\lambda}_j^{t+1} = \boldsymbol{\lambda}_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right), j = 1, \ldots, k$

6: **end for**

Step 1

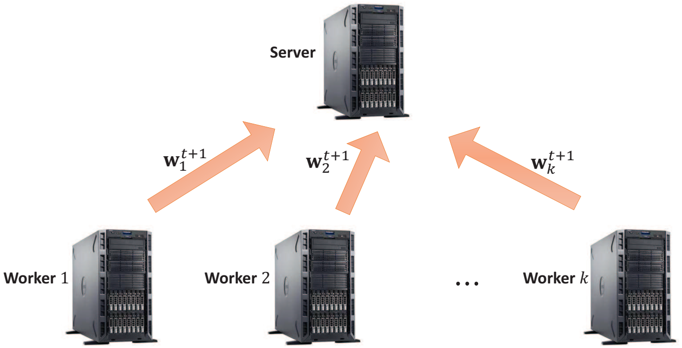
Server: send $\mathbf{y}^t$ and λ_j^t to worker $j = 1, \ldots, k$



Step 2

Each worker j: calculate $\mathbf{w}_j^{t+1}$ and send it to server

$$\mathbf{w}_j^{t+1} = \operatorname*{argmin}_{\mathbf{w}} f_j(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_j^t - \mathbf{w}\|_2^2,\ j = 1, \ldots, k$$



$$\mathbf{w}_1^{t+1} = \operatorname{argmin}_{\mathbf{w}} f_1(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_1^t - \mathbf{w}\|_2^2 \quad \mathbf{w}_k^{t+1} = \operatorname{argmin}_{\mathbf{w}} f_k(\mathbf{w}) + \frac{1}{2\gamma}\|\mathbf{y}^t + \gamma\boldsymbol{\lambda}_k^t - \mathbf{w}\|_2^2$$

Steps 3 and 4

- Server: $\mathbf{y}^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$
- Server: $\boldsymbol{\lambda}_j^{t+1} = \boldsymbol{\lambda}_j^t + \frac{1}{\gamma}\left(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}\right), j = 1, \ldots, k$

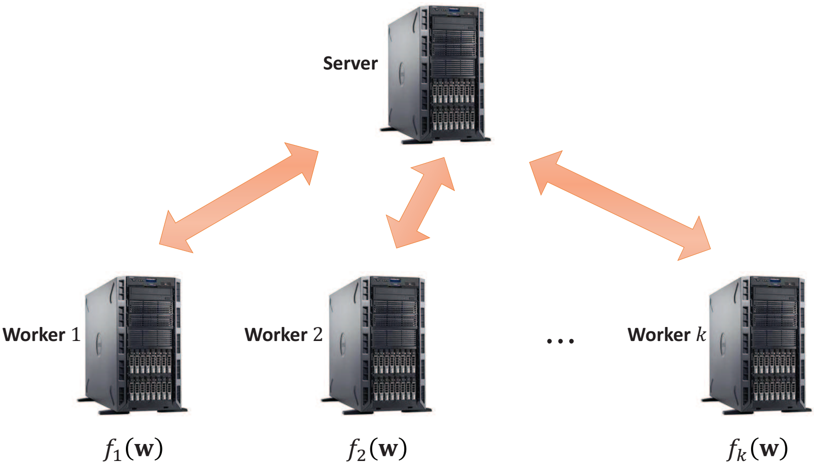
Server

$$y^{t+1} = \Pi_{\mathcal{W}}\left(\frac{1}{k}\sum_{j=1}^{k}(\mathbf{w}_j^{t+1} - \gamma\boldsymbol{\lambda}_j^t)\right)$$

$$\boldsymbol{\lambda}_j^{t+1} = \boldsymbol{\lambda}_j^t + \frac{1}{\gamma}(\mathbf{y}^{t+1} - \mathbf{w}_j^{t+1}), j = 1 \ldots, k$$

Challenges of Distributed Optimization

- Communication
- Synchronization

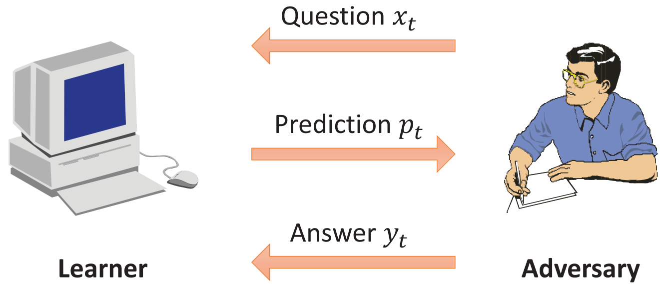


Outline

Online Learning (I)

Online Learning [Shalev-Shwartz, 2011]

for $t = 1, 2, \ldots, T$ **do**
    Receive a question $x_t \in \mathcal{X}$
    Predict $p_t \in \mathcal{D}$
    Receive true answer $y_t \in \mathcal{Y}$
    Suffer loss $\ell(y_t, p_t)$
end for



Online Learning (II)

Online Learning [Shalev-Shwartz, 2011]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Receive a question $x_t \in \mathcal{X}$
3: Predict $p_t \in \mathcal{D}$
4: Receive true answer $y_t \in \mathcal{Y}$
5: Suffer loss $\ell(y_t, p_t)$
6: **end for**

An Example—Online Classification

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Receive a question $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$
3: Predict $p_t \in \mathcal{D} = \{0, 1\}$
4: Receive true answer $y_t \in \mathcal{Y} = \{0, 1\}$
5: Suffer loss $\ell(y_t, p_t) = |y_t - p_t|$
6: **end for**

Online Learning (III)

Online Learning [Shalev-Shwartz, 2011]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Receive a question $x_t \in \mathcal{X}$
3: Predict $p_t \in \mathcal{D}$
4: Receive true answer $y_t \in \mathcal{Y}$
5: Suffer loss $\ell(y_t, p_t)$
6: **end for**

Cumulative Loss

$$\text{Cumulative Loss} = \sum_{t=1}^{T} \ell(y_t, p_t)$$

Online Learning (VI)

Online Learning [Shalev-Shwartz, 2011]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Receive a question $x_t \in \mathcal{X}$
3: Predict $p_t \in \mathcal{D}$
4: Receive true answer $y_t \in \mathcal{Y}$
5: Suffer loss $\ell(y_t, p_t)$
6: **end for**

Regret with respect to a hypothesis class $\mathcal{H}$

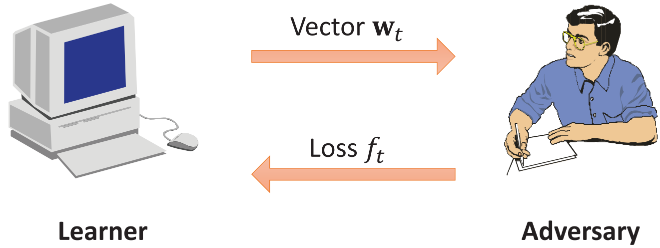
$$\mathrm{Regret} = \sum_{t=1}^{T} \ell(y_t, p_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))$$

$h : \mathcal{X} \mapsto \mathcal{D}$

Online Convex Optimization (I)

Online Convex Optimization [Shalev-Shwartz, 2011]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a convex loss function $f_t : \mathcal{W} \in \mathbb{R}$
4: Suffer loss $f_t(\mathbf{w}_t)$
5: **end for**



Online Convex Optimization (II)

Online Convex Optimization [Shalev-Shwartz, 2011]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a convex loss function $f_t : \mathcal{W} \in \mathbb{R}$
4: Suffer loss $f_t(\mathbf{w}_t)$
5: **end for**

An Example—Online SVM

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a training instance $(\mathbf{x}_t, y_t)$
4: Suffer loss $f_t(\mathbf{w}_t) = \ell(y_t, \mathbf{x}_t^\top \mathbf{w}_t) = \max(0, 1 - y_t \mathbf{x}_t^\top \mathbf{w}_t)$
5: **end for**

Online Convex Optimization (III)

Online Convex Optimization [Shalev-Shwartz, 2011]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a convex loss function $f_t : \mathcal{W} \in \mathbb{R}$
4: Suffer loss $f_t(\mathbf{w}_t)$
5: **end for**

Regret

$$\text{Regret} = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^{T} f_t(\mathbf{w})$$

Outline

Online Gradient Descent

Full-Information Setting

The learner observes not only $f_t(\mathbf{w}_t)$ but also $f_t(\cdot)$.

Online Gradient Descent [Zinkevich, 2003, Hazan et al., 2007]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a convex loss function $f_t : \mathcal{W} \in \mathbb{R}$
4: Suffer loss $f_t(\mathbf{w}_t)$
5: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t)$
6: **end for**

Regret Bound [Zinkevich, 2003, Hazan et al., 2007]

$$\text{Regret} = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^{T} f_t(\mathbf{w}) = O\left(\sqrt{T}\right) \text{ or } O(\log T)$$

Online Gradient Descent

Full-Information Setting

The learner observes not only $f_t(\mathbf{w}_t)$ but also $f_t(\cdot)$.

Online Gradient Descent [Zinkevich, 2003, Hazan et al., 2007]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a convex loss function $f_t : \mathcal{W} \in \mathbb{R}$
4: Suffer loss $f_t(\mathbf{w}_t)$
5: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t)$
6: **end for**

Regret Bound [Zinkevich, 2003, Hazan et al., 2007]

$$\text{Regret} = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{w}\in\mathcal{W}} \sum_{t=1}^{T} f_t(\mathbf{w}) = O\left(\sqrt{T}\right) \text{ or } O(\log T)$$

Online Gradient Descent

Full-Information Setting

The learner observes not only $f_t(\mathbf{w}_t)$ but also $f_t(\cdot)$.

Online Gradient Descent [Zinkevich, 2003, Hazan et al., 2007]

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a vector $\mathbf{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$
3: Receive a convex loss function $f_t : \mathcal{W} \in \mathbb{R}$
4: Suffer loss $f_t(\mathbf{w}_t)$
5: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t)$
6: **end for**

Regret Bound [Zinkevich, 2003, Hazan et al., 2007]

$$\text{Regret} = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^{T} f_t(\mathbf{w}) = O\left(\sqrt{T}\right) \text{ or } O(\log T)$$

Outline

Bandit Setting

Bandit Setting

The learner only observes not only $f_t(\mathbf{w}_t)$.

Generation Process of the Loss Functions

- Stochastic: $f_1, \ldots, f_t$ are i.i.d. sampled
- Adversarial
  - Oblivious: f_t is independent of $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$
  - Nonoblivious: f_t may depend on $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$

Types of the Loss Functions

- Convex
- Linear
- Multi-armed Bandit

Bandit Setting

Bandit Setting

The learner only observes not only $f_t(\mathbf{w}_t)$.

Generation Process of the Loss Functions

- Stochastic: $f_1, \ldots, f_t$ are i.i.d. sampled
- Adversarial
  - Oblivious: f_t is independent of $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$
  - Nonoblivious: f_t may depend on $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$

Types of the Loss Functions

- Convex
- Linear
- Multi-armed Bandit

Bandit Setting

Bandit Setting

The learner only observes not only $f_t(\mathbf{w}_t)$.

Generation Process of the Loss Functions

- Stochastic: $f_1, \ldots, f_t$ are i.i.d. sampled
- Adversarial
  - Oblivious: f_t is independent of $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$
  - Nonoblivious: f_t may depend on $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$

Types of the Loss Functions

- Convex
- Linear
- Multi-armed Bandit

Multi-armed Bandit

http://blog.mynaweb.com/wp-content/uploads/2013/02/Bandits.png

Stochastic Multi-armed Bandit

Setting

- There are K arms
- Successive pulls of arm i yield rewards $X_1^i, X_2^i, \ldots$
  - Those rewards are i.i.d. according to an unknown distribution D_i with unknown mean μ_i

The Learning Process

1: **for** $t = 1, 2, \ldots, T$ **do**
2: pull arm $i \in [K]$
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: **end for**

How to maximize the rewards?

Always pull arm $i = \operatorname{argmax}_{i \in [K]} \mu_i$

Stochastic Multi-armed Bandit

Setting

- There are K arms
- Successive pulls of arm i yield rewards $X_1^i, X_2^i, \ldots$
  - Those rewards are i.i.d. according to an unknown distribution D_i with unknown mean μ_i

The Learning Process

1: **for** $t = 1, 2, \ldots, T$ **do**
2: pull arm $i \in [K]$
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: **end for**

How to maximize the rewards?

Always pull arm $i = \operatorname{argmax}_{i \in [K]} \mu_i$

Stochastic Multi-armed Bandit

Setting

- There are K arms
- Successive pulls of arm i yield rewards $X_1^i, X_2^i, \ldots$
  - Those rewards are i.i.d. according to an unknown distribution D_i with unknown mean μ_i

The Learning Process

1: **for** $t = 1, 2, \ldots, T$ **do**
2: pull arm $i \in [K]$
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: **end for**

How to maximize the rewards?

Always pull arm $i = \operatorname{argmax}_{i \in [K]} \mu_i$

Exploitation-Exploration Dilemma

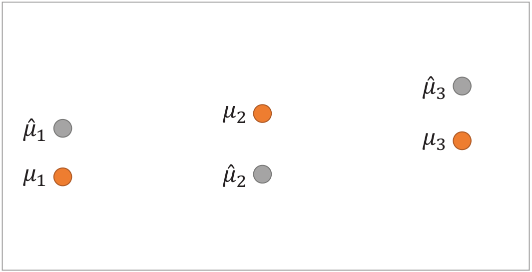
A Naive Algorithm

Exploration

1. Pull each arm $m = 100$ times
2. Calculate the estimated mean $\hat{\mu}_i$ each arm i

Exploitation

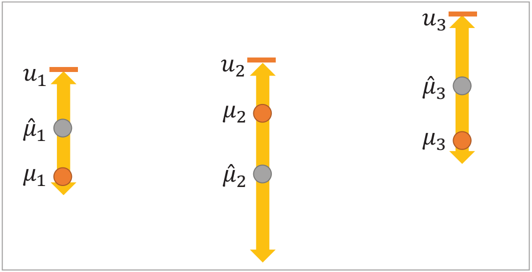
1. Always pull arm $i = \operatorname{argmax}_{i \in [K]} \hat{\mu}_i$



Upper Confidence Bounds (I)

Exploitation-Exploration Tradeoff by Upper Confidence Bounds (UCB) [Auer et al., 2002]

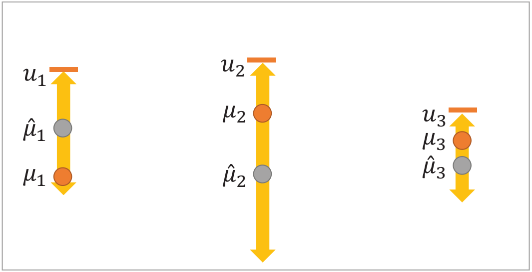
1: **for** $t = 1, 2, \ldots, T$ **do**
2: Pull arm $i = \text{argmax}_{i \in [K]} u_i$ (WHP, $u_i \geq \mu_i$)
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: Update the upper bound u_i for arm i
5: **end for**



Upper Confidence Bounds (II)

Exploitation-Exploration Tradeoff by Upper Confidence Bounds (UCB) [Auer et al., 2002]

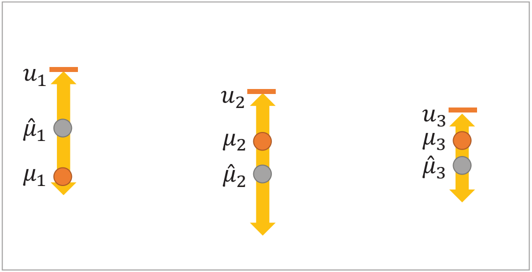
1: **for** $t = 1, 2, \dots, T$ **do**
2: Pull arm $i = \text{argmax}_{i \in [K]} u_i$ (WHP, $u_i \geq \mu_i$)
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: Update the upper bound u_i for arm i
5: **end for**



Upper Confidence Bounds (III)

Exploitation-Exploration Tradeoff by Upper Confidence Bounds (UCB) [Auer et al., 2002]

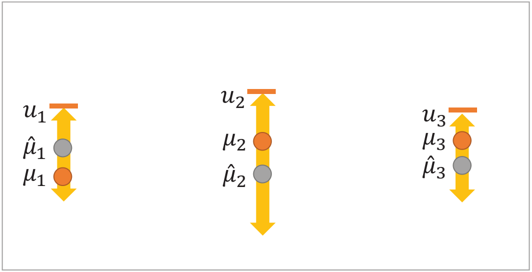
1: **for** $t = 1, 2, \ldots, T$ **do**
2: Pull arm $i = \text{argmax}_{i \in [K]} u_i$ (WHP, $u_i \geq \mu_i$)
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: Update the upper bound u_i for arm i
5: **end for**



Upper Confidence Bounds (IV)

Exploitation-Exploration Tradeoff by Upper Confidence Bounds (UCB) [Auer et al., 2002]

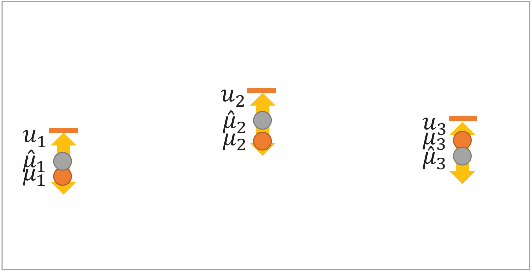
1: **for** $t = 1, 2, \ldots, T$ **do**
2: Pull arm $i = \text{argmax}_{i \in [K]} u_i$ (WHP, $u_i \geq \mu_i$)
3: Receive reward X^i sampled from distribution $\mathcal{D}_i$
4: Update the upper bound u_i for arm i
5: **end for**



Upper Confidence Bounds (V)

Exploitation-Exploration Tradeoff by Upper Confidence Bounds (UCB) [Auer et al., 2002]

for $t = 1, 2, \ldots, T$ **do**
  Pull arm $i = \mathrm{argmax}_{i \in [K]} u_i$ (WHP, $u_i \geq \mu_i$)
  Receive reward X^i sampled from distribution $\mathcal{D}_i$
  Update the upper bound u_i for arm i
end for



Outline

Summary

Stochastic Optimization

- Stochastic Gradient Descent (SGD)—*Time Reduction*
- Stochastic Proximal Gradient Descent (SPGD)—*Space Reduction*

Distributed Optimization

- Distributed Gradient Descent
- Alternating Direction Method of Multipliers (ADMM)

Online Learning

- Full-Information Setting—Online Gradient Descent
- Bandit Setting—Exploitation-Exploration Dilemma

Reference I

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002).
Finite-time analysis of the multiarmed bandit problem.
Machine Learning, 47(2-3):235–256.

Hazan, E., Agarwal, A., and Kale, S. (2007).
Logarithmic regret algorithms for online convex optimization.
Machine Learning, 69(2-3):169–192.

Hazan, E. and Kale, S. (2011).
Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization.
In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 421–436.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009).
Robust stochastic approximation approach to stochastic programming.
SIAM Journal on Optimization, 19(4):1574–1609.

Roux, N. L., Schmidt, M., and Bach, F. (2012).
A stochastic gradient method with an exponential convergence rate for finite training sets.
In *Advances in Neural Information Processing Systems 25*, pages 2672–2680.

Reference II

Shalev-Shwartz, S. (2011).
Online learning and online convex optimization.
Foundations and Trends in Machine Learning, 4(2):107¨C–194.

Zhang, L., Yang, T., Jin, R., and Zhou, Z.-H. (2015).
Stochastic proximal gradient descent for nuclear norm regularization.
ArXiv e-prints, arXiv:1511.01664.

Zinkevich, M. (2003).
Online convex programming and generalized infinitesimal gradient ascent.
In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936.