Advances in Optimization Methods for Machine Learning

(2016 - 2017)

Xiaotong Yuan

BDAT Lab
Nanjing University of Information Science and Technology

Context

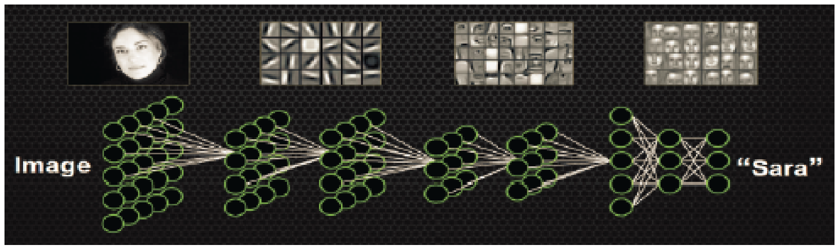
Large-Scale Machine learning

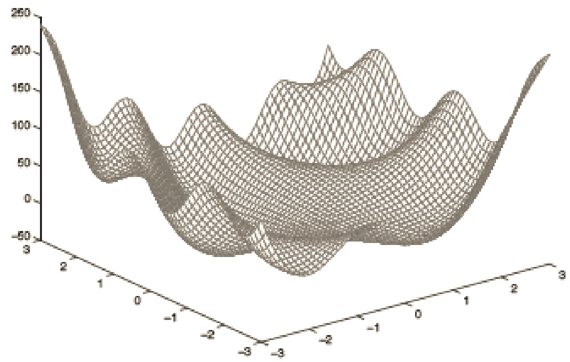
Large-scale Learning=Optimization over BIG Data

ERM: large n, large d, complex loss/penalty

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i, \theta)) \quad + \quad \lambda \Omega(\theta)$$

(x, y) = (features, labels) n=# of samples d=dimension





(Elad Hazen, tutorial 2017)

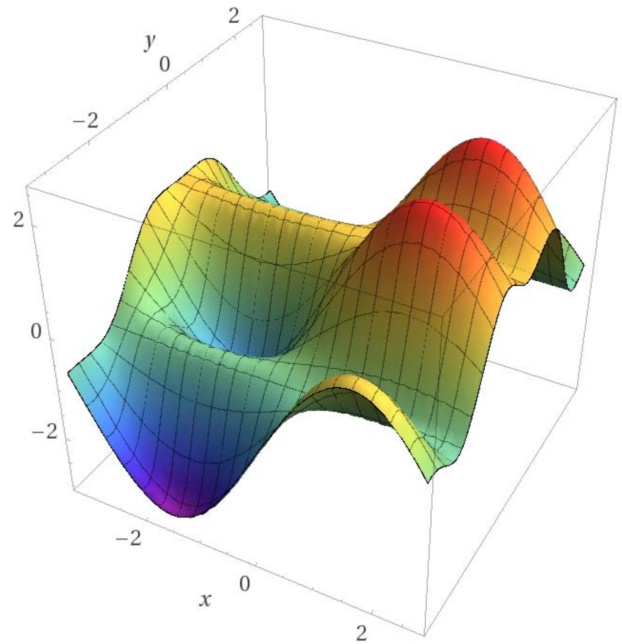
Outline

- Stochastic Variance Reduction Grad. ++
  - Nonconvex + SVRG/SAGA (Reddi *et al., ICML* 2016; Allen-Zhu & Hazan, *ICML* 2016; Reddi *et al., NIPS* 2016)
  - Sparsity + SVRG (Li *et al., ICML* 2016; Chen & Gu, *UAI* 2016)
  - Frank-Wolfe + SVRG (Hazan & Luo., *ICML* 2016)
- Distributed and parallel settings
  - Communication-efficient distributed statistical inference (Jordan *et.al*, *Arxiv* 16)
- Optimization for training deep models
  - ADMM for DNN (Taylor *et al., ICML* 16; Yang *et al., NIPS* 16)
  - Sparse DNN training (Han *et al., ICLR* 16/17; Jin *et al., Arxiv* 16)

Nonconvex SVRG

Nonconvex finite-sum problems

$$\min_{\theta \in \mathbb{R}^d} \quad g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$



Difficult to optimize, but local minima, maxima, saddle points satisfy $\nabla g(\theta) = 0$

Stationarity gap: $\mathbb{E}[\|\nabla g(\theta_t)\|^2] \leq \epsilon$

Nonconvex SVRG (Cont.)

for s=0 to S-1

$\theta_0^{s+1} \leftarrow \theta_m^s$

$\tilde{\theta}^s \leftarrow \theta_m^s$

for t=0 to m-1

Uniformly randomly pick $i(t) \in \{1, \ldots, n\}$

$$\theta_{t+1}^{s+1} = \theta_t^{s+1} - \eta_t \left[\nabla f_{i(t)}(\theta_t^{s+1}) - \nabla f_{i(t)}(\tilde{\theta}^s) + \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{\theta}^s) \right]$$

end

end

Algorithm is identical to convex SVRG

(Reddi *et al., ICML* 2016)

Nonconvex SVRG (Cont.)

Complexity results

Algorithm	Nonconvex	Nonconvex-PL
SGD	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{1}{\epsilon^2}\right)$
GD	$O\left(\frac{n}{\epsilon}\right)$	$O\left(\frac{n}{2\mu}\log\frac{1}{\epsilon}\right)$
SVRG	$O\left(n+\frac{n^{2/3}}{\epsilon}\right)$	$O\left(\left(n+\frac{n^{2/3}}{2\mu}\right)\log\frac{1}{\epsilon}\right)$
SAGA	$O\left(n+\frac{n^{2/3}}{\epsilon}\right)$	$O\left(\left(n+\frac{n^{2/3}}{2\mu}\right)\log\frac{1}{\epsilon}\right)$
MSVRG	$O\left(\min\left(\frac{1}{\epsilon^2}\right), \frac{n^{2/3}}{\epsilon}\right)$	—

(Reddi *et al., ICML* 2016; *Allen-Zhu & Hazan, ICML 2016*)

SVRG for Nonconvex Composite Opt.

Nonconvex composite optimization:

$$\min_{x \in \mathbb{R}^d} \quad F(x) := \boxed{f(x)} + h(x)$$

nonconvex convex nonsmooth

Once again variance reduction works!

Algorithm	IFO	PO	IFO (PL)	PO (PL)	Constant minibatch?
ProxSGD	$O\left(1/\epsilon^2\right)$	$O\left(1/\epsilon\right)$	$O\left(1/\epsilon^2\right)$	$O\left(1/\epsilon\right)$	?
ProxGD	$O\left(n/\epsilon\right)$	$O\left(1/\epsilon\right)$	$O\left(n\kappa \log(1/\epsilon)\right)$	$O\left(\kappa \log(1/\epsilon)\right)$	–
ProxSVRG	$O(n + (n^{2/3}/\epsilon))$	$O(1/\epsilon)$	$O((n + \kappa n^{2/3}) \log(1/\epsilon))$	$O(\kappa \log(1/\epsilon))$	√
ProxSAGA	$O(n + (n^{2/3}/\epsilon))$	$O(1/\epsilon)$	$O((n + \kappa n^{2/3}) \log(1/\epsilon))$	$O(\kappa \log(1/\epsilon))$	√

(Reddi *et al.*, *NIPS* 2016)

SVRG for Sparse Learning

Sparsity-constrained optimization

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \mathcal{F}(\boldsymbol{w}) \quad \text{s.t. } \|\boldsymbol{w}\|_0 \leq k$$

Nonconvex and NP-hard!

Gradient Hard Thresholding (GHT)

$$\boldsymbol{w}^{(t+1)} = \mathcal{H}_k\left(\boldsymbol{w}^{(t+1)} - \eta \nabla \mathcal{F}(\boldsymbol{w}^{(t+1)})\right)$$

Gradient descent + Hard truncation

SVR-GHT

Algorithm 1 Stochastic Variance Reduced Gradient Hard Thresholding Algorithm.

Input: update frequency m, step size parameter η, sparsity k, and initial solution $\widetilde{\boldsymbol{w}}^{(0)}$

for $r = 1, 2, \dots$

$\quad \widetilde{\boldsymbol{w}} = \widetilde{\boldsymbol{w}}^{(r-1)}$, $\widetilde{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\widetilde{\boldsymbol{w}})$, $\boldsymbol{w}^{(0)} = \widetilde{\boldsymbol{w}}$

$\quad$ **for** $t = 0, 1, \dots, m-1$

$\quad\quad$ (S1) Randomly sample i_t from $[n]$

$\quad\quad$ (S2) $\bar{\boldsymbol{w}}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta\left(\nabla f_{i_t}(\boldsymbol{w}^{(t)}) - \nabla f_{i_t}(\widetilde{\boldsymbol{w}}) + \widetilde{\boldsymbol{\mu}}\right)$

SVRG

$\quad\quad$ (S3) $\boldsymbol{w}^{(t+1)} = \mathcal{H}_k(\bar{\boldsymbol{w}}^{(t+1)})$

hard truncation

$\quad$ **end for**

$\quad \widetilde{\boldsymbol{w}}^{(r)} = \boldsymbol{w}^{(m)}$

end for

SVRG + Hard truncation

Exhibits similar convergence behavior to GHT!

(Li *et al.*, *ICML* 2016)

SVRG for Projection-Free Optimization

Convex constrained optimization

$$\min_{\boldsymbol{w}\in\Omega} f(\boldsymbol{w}) = \min_{\boldsymbol{w}\in\Omega} \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w})$$

Quadratic Projection: $\operatorname{argmin}_{\boldsymbol{v}\in\Omega} \|\boldsymbol{w}-\boldsymbol{v}\|^2$

Linear Projection: $\operatorname{argmin}_{\boldsymbol{v}\in\Omega} \boldsymbol{w}^\top \boldsymbol{v}$

Frank-Wolfe Algorithm: LP much faster than QP

$$\boldsymbol{v}_k = \operatorname*{argmin}_{\boldsymbol{v}\in\Omega} \nabla f(\boldsymbol{w}_{k-1})^\top \boldsymbol{v}$$

$$\boldsymbol{w}_k = (1-\gamma_k)\boldsymbol{w}_{k-1} + \gamma_k \boldsymbol{v}_k$$

SVRG-FW

Algorithm 1 Stochastic Variance-Reduced Frank-Wolfe (SVRF)

- **Input:** Objective function $f = \frac{1}{n}\sum_{i=1}^{n} f_i$.
- **Input:** Parameters γ_k, m_k and N_k.
- **Initialize:** $\boldsymbol{w}_0 = \min_{\boldsymbol{w}\in\Omega} \nabla f(\boldsymbol{x})^\top \boldsymbol{w}$ for some arbitrary $\boldsymbol{x} \in \Omega$.
- **for** $t = 1, 2, \ldots, T$ **do**
  - Take snapshot: $\boldsymbol{x}_0 = \boldsymbol{w}_{t-1}$ and compute $\nabla f(\boldsymbol{x}_0)$.
  - **for** $k = 1$ **to** N_t **do**
    - Compute $\tilde{\nabla}_k$, the average of m_k iid samples of $\tilde{\nabla} f(\boldsymbol{x}_{k-1}, \boldsymbol{x}_0)$.
    - Compute $\boldsymbol{v}_k = \min_{\boldsymbol{v}\in\Omega} \tilde{\nabla}_k^\top \boldsymbol{v}$.
    - Compute $\boldsymbol{x}_k = (1-\gamma_k)\boldsymbol{x}_{k-1} + \gamma_k \boldsymbol{v}_k$.
  - **end for**
  - Set $\boldsymbol{w}_t = \boldsymbol{x}_{N_t}$.
- **end for**

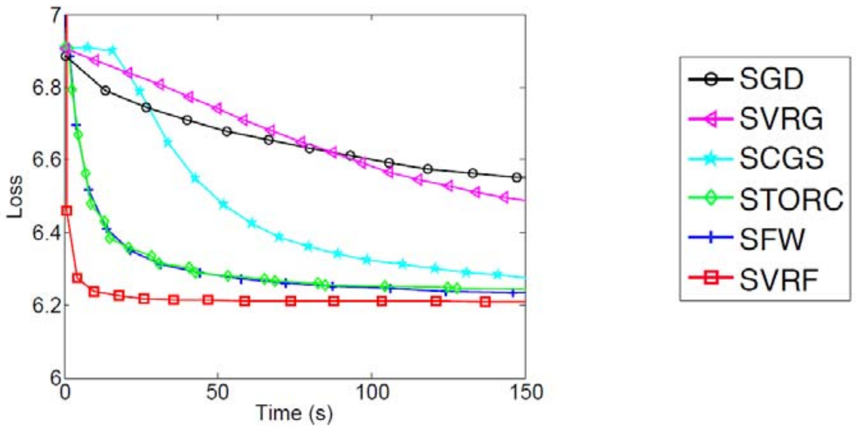
SVRG

Frank-Wolfe

SVRG + Frank-Wolfe

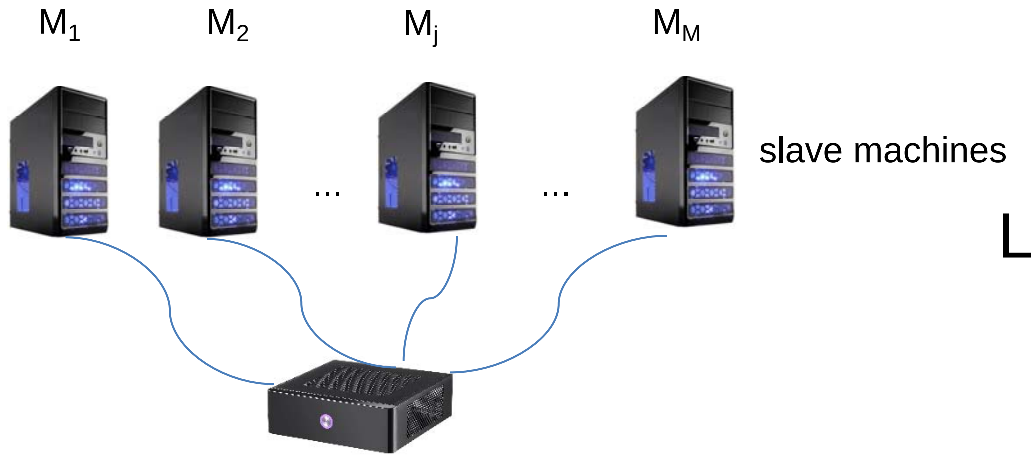
	previous work	**this work**
Smooth	$\mathcal{O}(\frac{1}{\epsilon^2})$	$\mathcal{O}(\frac{1}{\epsilon^{1.5}})$
Smooth and Strongly Convex	$\mathcal{O}(\frac{1}{\epsilon})$	$\mathcal{O}(\ln\frac{1}{\epsilon})$

Improved complexity bound



(Hazan & Luo, *ICML* 2016)

Distributed Optimization



Distributed data storage:

$$D = [D_1, D_2, \ldots, D_j, \ldots, D_M]$$

$$\forall j, \ \ D_j \text{ is stored on machine } M_j$$

Global solution

$$w^* = \arg\min_w \left\{ F(w) := \sum_{(x_i, y_i) \in D} l(w \mid x_i, y_i) \right\}$$

Local solution on M_j

$$w_j^* = \arg\min_w \left\{ F_j(w) := \sum_{(x_i, y_i) \in D_j} l(w \mid x_i, y_i) \right\}$$

Problem: How to calculate w^* based on $\{w_j^*\}_{j=1}^M$

Practical Issues:

- communication cost
- local machine processing delay

……

Communication-Efficient Distributed Statistical Inference

Global loss :

$$\mathcal{L}_N(\theta) = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{k}\mathcal{L}(\theta; z_{ij}) = \frac{1}{k}\sum_{j=1}^{k}\mathcal{L}_j(\theta)$$

Local loss of machine j:

$$\mathcal{L}_j(\theta) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\theta; z_{ij}), \quad \text{for } j \in [k],$$

Key Idea: surrogate construction based on local higher-order derivative and **global** first-order derivative

$$\widetilde{\mathcal{L}}(\theta) = \mathcal{L}_N(\overline{\theta}) + \langle \nabla \mathcal{L}_N(\overline{\theta}), \theta - \overline{\theta}\rangle + \sum_{j=2}^{\infty}\frac{1}{j!}\nabla^j \mathcal{L}_1(\overline{\theta})\,(\theta - \overline{\theta})^{\otimes j}$$

global

local

(Jordan *et.al*, *Arxiv* 16)

Initialize $\theta^{(0)} = \overline{\theta}$;

for $t = 0, 1, \ldots, T-1$ **do**

Distributed **global** gradient evaluation

Transmit the current iterate $\theta^{(t)}$ to local machines $\{\mathcal{M}_j\}_{j=1}^{k}$;

for $j = 1 : k$ **do**

Compute the local gradient $\nabla \mathcal{L}_j(\theta^{(t)})$ at machine $\mathcal{M}_j$;

Transmit the local gradient $\nabla \mathcal{L}_j(\theta^{(t)})$ to machine $\mathcal{M}_1$;

end

Calculate the global gradient $\nabla \mathcal{L}_N(\theta^{(t)}) = \frac{1}{k}\sum_{j=1}^{k} \nabla \mathcal{L}_j(\theta^{(t)})$ in Machine $\mathcal{M}_1$;

Form the surrogate function $\widetilde{\mathcal{L}}^t(\theta) = \mathcal{L}_1(\theta) - \langle \theta, \nabla \mathcal{L}_1(\theta^{(t)}) - \nabla \mathcal{L}_N(\theta^{(t)}) \rangle$;

Do one of the following in Machine $\mathcal{M}_1$:

(1). Update $\theta^{(t+1)} \in \arg\min_{\theta \in \Theta} \widetilde{\mathcal{L}}^t(\theta)$; `// Exactly minimizing surrogate function` $\widetilde{\mathcal{L}}$

(2). Update $\theta^{(t+1)} = \theta^{(t)} - \nabla^2 \mathcal{L}_1(\theta^{(t)})^{-1} \nabla \mathcal{L}_N(\theta^{(t)})$; `// One-step quadratic approximation`

Local surrogate minimization

end

return $\theta^{(T)}$

Algorithm 1: Iterative local estimation

(Jordan *et.al*, *Arxiv* 16, Wang *et al.*, *Arxiv*, 2016)

ADMM in Deep Learning

General form of DNN

$$\min_{W} \ \ell(f(a_0; W), y)$$

$W=\{W_1,W_2,\ldots,W_L\}$ - the weight matrices
a_0 - input activations
h - activation function

Constrined form of DNN:

$$\begin{aligned}
\underset{\{W_l\},\{a_l\},\{z_l\}}{\text{minimize}} \quad & \ell(z_L, y) \\
\text{subject to} \quad & z_l = W_l a_{l-1}, \text{ for } l = 1, 2, \cdots L \\
& a_l = h_l(z_l), \text{ for } l = 1, 2, \cdots L-1.
\end{aligned}$$

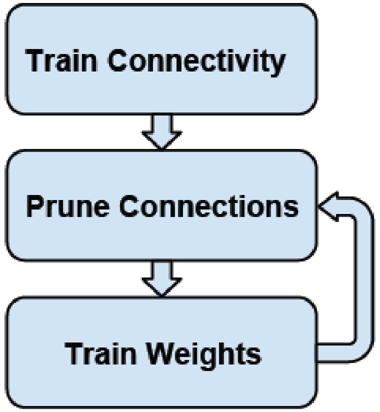
Bregman ADMM

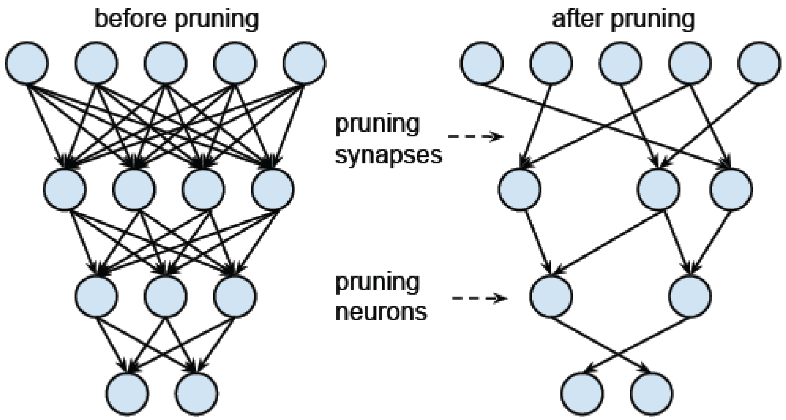
$$\begin{aligned}
\underset{\{W_l\},\{a_l\},\{z_l\}}{\text{minimize}} \quad & \ell(z_L, y) \\
& + \langle z_L, \lambda \rangle + \beta_L \|z_L - W_L a_{L-1}\|^2 \\
& + \sum_{l=1}^{L-1} \left[\gamma_l \|a_l - h_l(z_l)\|^2 + \beta_l \|z_l - W_l a_{l-1}\|^2 \right]
\end{aligned}$$

Merits:
1. Do not need gradient propagation
2. Exhibits strong scaling in the distributed computing environment

(Taylor *et al.*, *ICML* 2016)

Sparse DNN Training



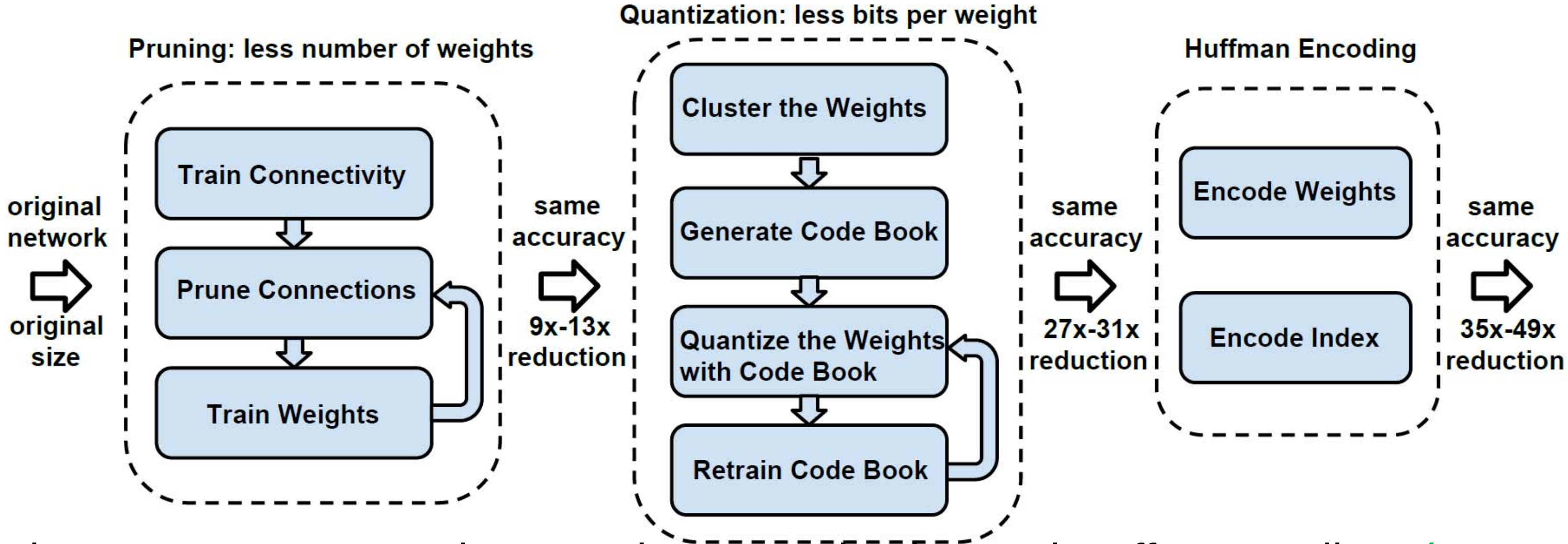


Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
Baseline Caffemodel [26]	42.78%	19.73%	61.0M	1×
Data-free pruning [28]	44.40%	-	39.6M	1.5×
Fastfood-32-AD [29]	41.93%	-	32.8M	2×
Fastfood-16-AD [29]	42.90%	-	16.4M	3.7×
Collins & Kohli [30]	44.40%	-	15.2M	4×
Naive Cut	47.18%	23.23%	13.8M	4.4×
SVD [12]	44.02%	20.56%	11.9M	5×
Network Pruning	**42.77%**	**19.67%**	**6.7M**	**9×**

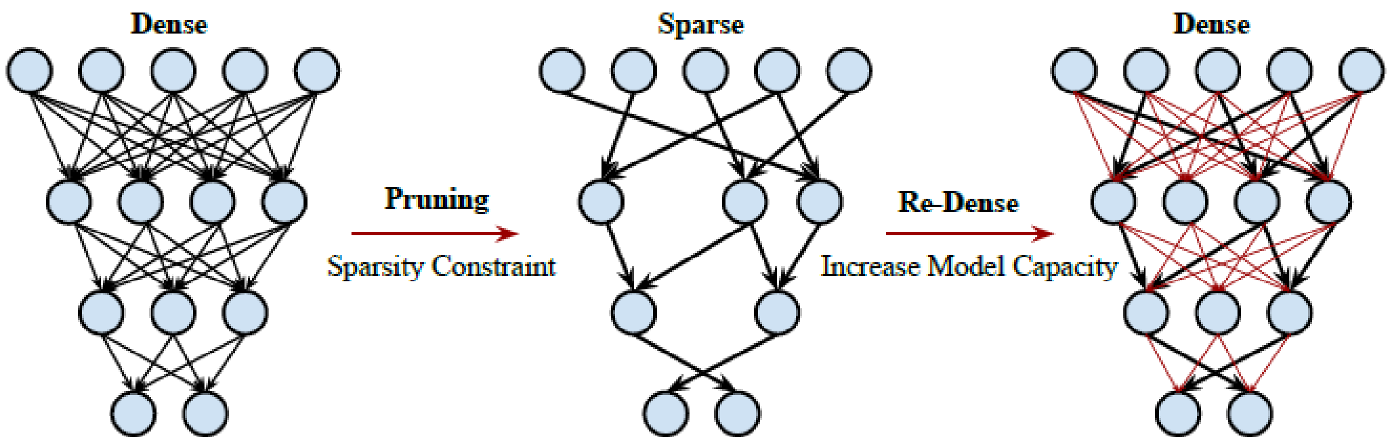
Comparison with other model reduction methods on AlexNet

(Han, *et al., NIPS* 2015)

Sparse DNN Training (Cont.)



Three stage compression: pruning, quantization and Huffman coding. (Han *et al.*, *ICLR* 2016)



Dense-Sparse-Dense Training. (Han *et al.*, *ICLR* 2017, Jin *et al.*, *Arxiv* 2016)

Other Interesting Work

Stochastic optimization methods

- Newton-type methods (Garber *et al., ICML* 2016)
- Adaptive Blockwise Frank-Wolfe (Osokin *et al., ICML* 2016)
- New insights and improvements on SDCA (Shai Shalev-Shwartz, *ICML* 2016)
- SGD without replacement sampling (Shamir., *NIPS* 2016)

Distributed and parallel optimization

- Parallel/distributed Frank-Wolfe (Wang *et al., ICML* 2016)
- Adaptive delay (Sra *et al., AISTATS* 2016)
- Distributed Kernel PCA (Balcan *et.al, KDD* 2016)
- Distributed Submodular Cover (Mirzasoleiman *et al., NIPS* 2016)

Other Interesting Work (Cont.)

Optimization for training deep models

- Improved training methods for GANs (Salimans *et al., NIPS* 2016)
- Weight/Layer normalization for acceleration (Salimans & Kingma *et.al, NIPS* 2016; *Ba et al., Arxiv* 2016)
- Distributed second-order optimization (Ba *et al., ICLR* 2017)

Fast top eigenvector computation (Garber *et al., ICML* 2016)

Convergence acceleration (Scieur *et al., NIPS* 2016)

NOT touch upon:

Online learning/optimization, Bayesian inference in graphical models, Markov-chain Monte-Carlo, Partial information and bandit algorithms,…

Perspectives

Nonconvex optimization

- Impact of non-convexity on generalization
- Information-Theoretic limits for nonconvex finite-sums
- Duality theory for nonconvex optimization
- More “tractable” nonconvex models
- Infinite dimensional nonconvex problems

Communication-efficient distributed optimization (theory and implementation)

Energy economic training of deep models

Polynomial optimization

New applications/software toolkits

and many more…

Thank you!