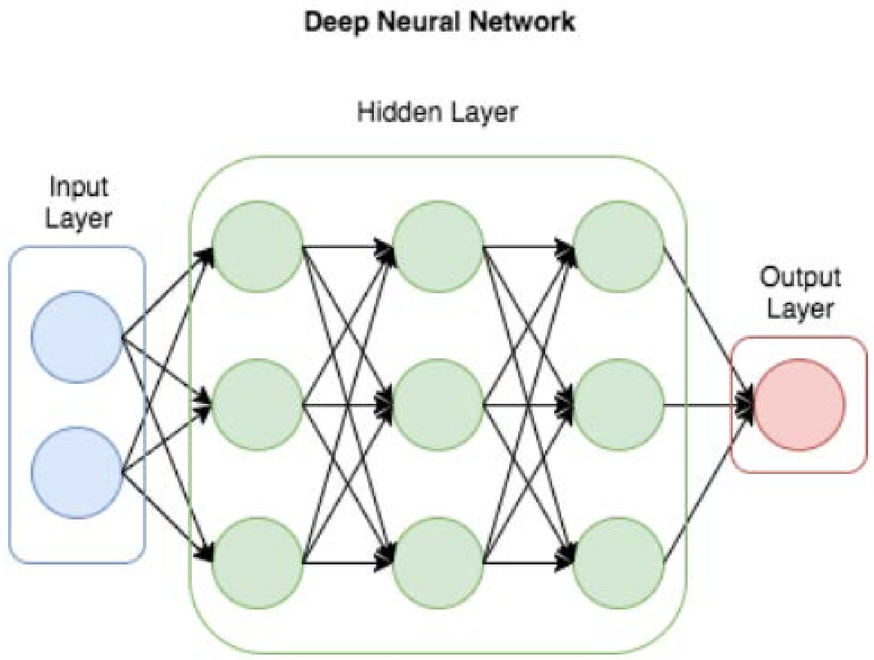
Optimization on deep learning

MATH818 RESEARCH IN APPLIED MATH

2019020356 JAEHEUN JUNG

DNN Architecture



- Universal function approximator
- Combination of weighted sum and activation
- Parametrized with matrices

Loss function

- Measurement for the difference between model and real data

MSE loss (Mean-Squared Error)

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i^{true} - y_i^{pred})^2$$

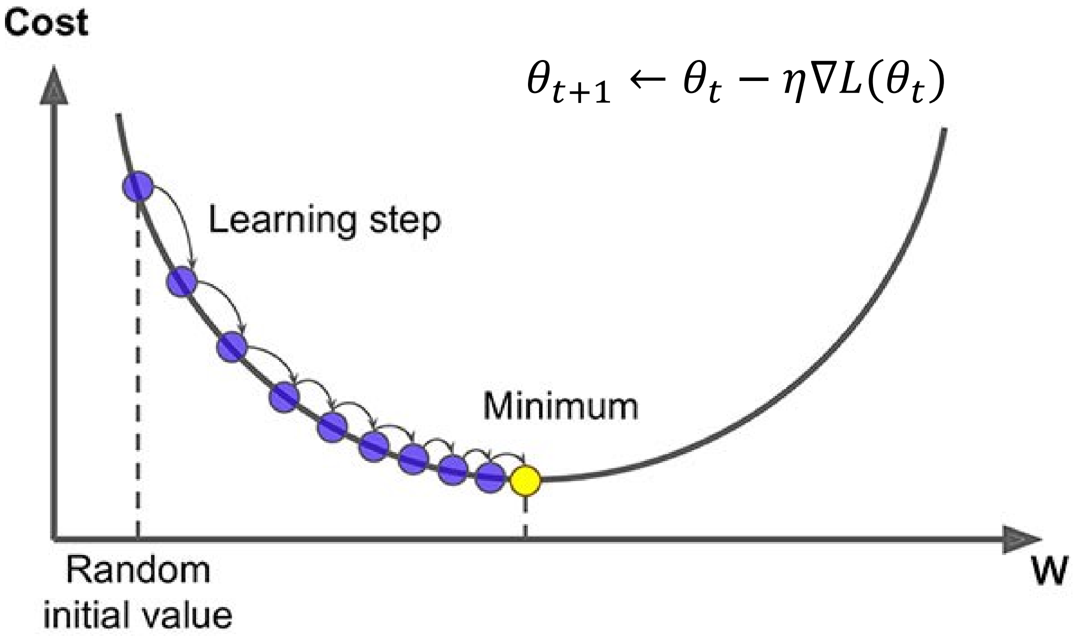
Cross-entropy loss

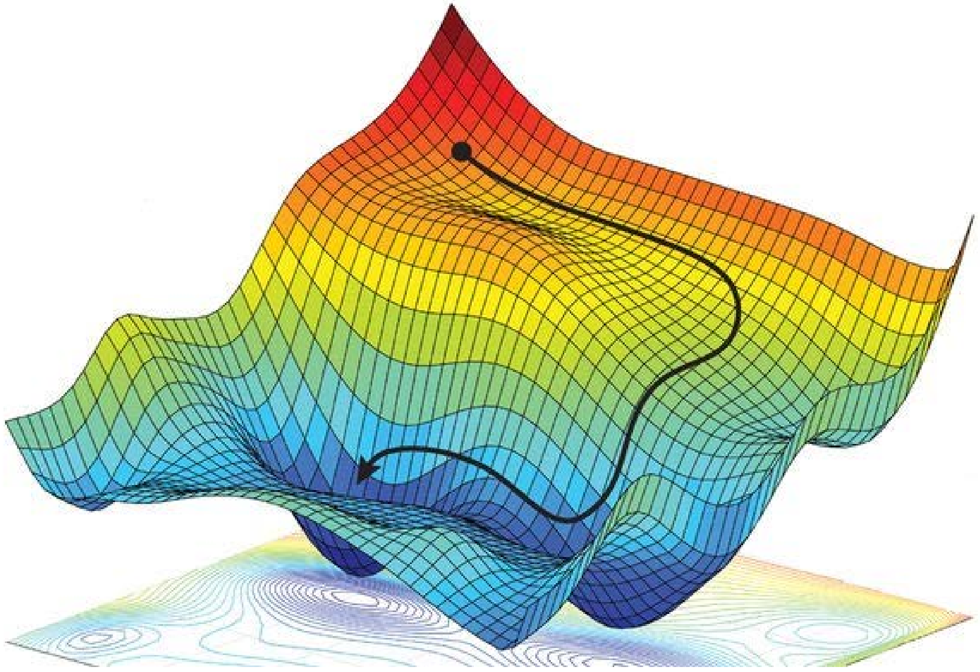
$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} -y_{ij}^{true} log(y_{ij}^{pred})$$

Gradient descent

- First-order iterative optimization

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla L(\theta_t)$$



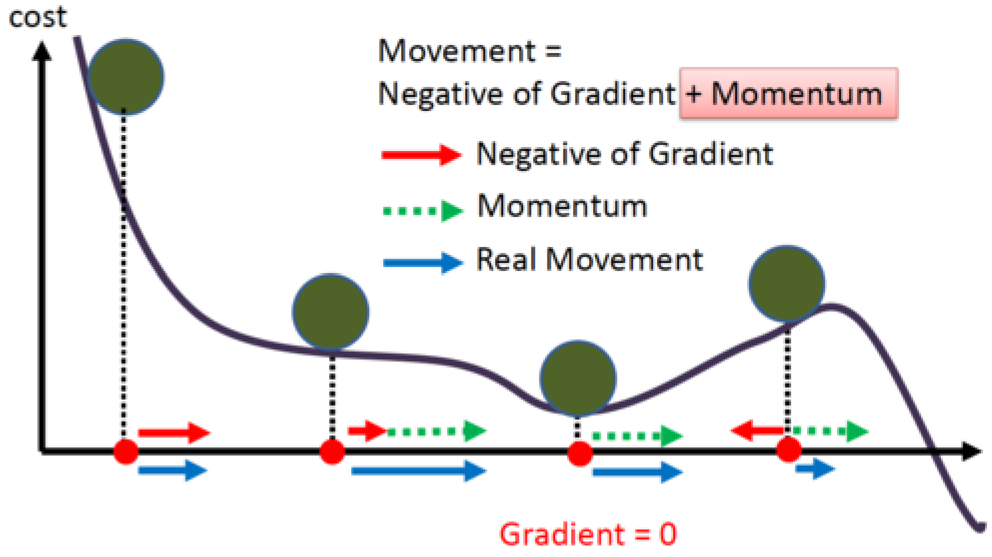


Optimizers

- Momentum

$$v_{t+1} = \gamma v_t - \eta \nabla L(\theta_t)$$

$$\theta_{t+1} = \theta_t + v_{t+1}$$



optimizers

- Adaptive learning rate

- Adagrad

$$G_\theta^t = \sum_{i=0}^{t} (\nabla L(\theta_i))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_\theta^t + \epsilon}} \nabla L(\theta_t)$$

- RMSprop

$$G_\theta^t = \gamma G_\theta^{t-1} + (1-\gamma)(\nabla L(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_\theta^t + \epsilon}} \nabla L(\theta^t)$$

optmimizers

- Adam

$$\hat{m}_\theta = \frac{m_\theta^{t+1}}{1-\beta_1^{t+1}} \text{ where } m_\theta^{t+1} = \beta_1 m_\theta^t + (1-\beta_1)\nabla L(\theta^t)$$

$$\hat{G}_\theta = \frac{G_\theta^{t+1}}{1-\beta_2^{t+1}} \text{ where } G_\theta^{t+1} = \beta_2 G_\theta^t + (1-\beta_2)(\nabla L(\theta^t))^2$$

$$\theta^{t+1} = \theta^t - \eta \frac{\hat{m}_\theta}{\sqrt{\hat{G}_\theta} + \epsilon}$$

Optimizers

- RAdam

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla L(\theta^t))^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \text{ where } m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(\theta_t)$$

$$\rho_t = \rho_\infty - 2t\frac{\beta_2^t}{1-\beta_2^t} \text{ where } \rho_\infty = \frac{2}{1-\beta_2} - 1$$

if $\quad \rho_t > 4$ then

$$\ell_t = \sqrt{\frac{1-\beta_2^t}{v_t}}$$

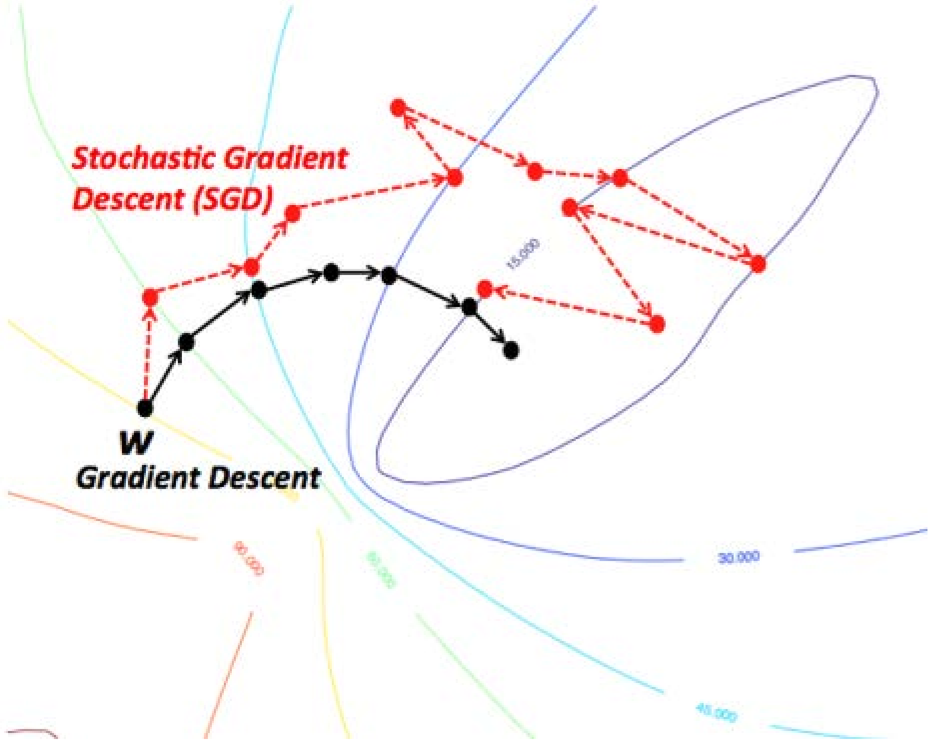
$$r_t = \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}}$$

$$\theta_t = \theta_{t-1} - \alpha_t r_t \ell_t \hat{m}_t \text{ where } \alpha_t \text{ is step size}$$

else $\theta_t = \theta_{t-1} - \alpha_t \hat{m}_t$

Batch size

of datapoints for single update



Stochastic Gradient descent

- Single datapoint

Batch Gradient descent

- All datapoints

Data parallelism

- Multiple workers compute gradient

- Parameter server collects gradients and update the parameter

- Update Rules:
  - SSGD and ASGD

Algorithm 2: worker m

Input dataset $\mathcal{X}$, minibatch size $\mathcal{B}$

for $t = 0, 1, \cdots$ **do**

  Wait to read $\theta^{(t)}$ from parameter server;

  $G_m^{(t)} := 0$;

  for $i = 1, \cdots, \mathcal{B}$ **do**

    Sample data $x_{\tilde{k},i}$ from $\mathcal{X}$;

    $G_m^{(t)} \longleftarrow G_m^{(t)} + \frac{1}{\mathcal{B}} \nabla L(x_{\tilde{k},i}, \theta^{(t)})$

  end

  Send $G_m^{(t)}$ to parameter server

end

Data parallelism

SYNCHRONOUS UPDATE

Algorithm 3: SSGD parameter server

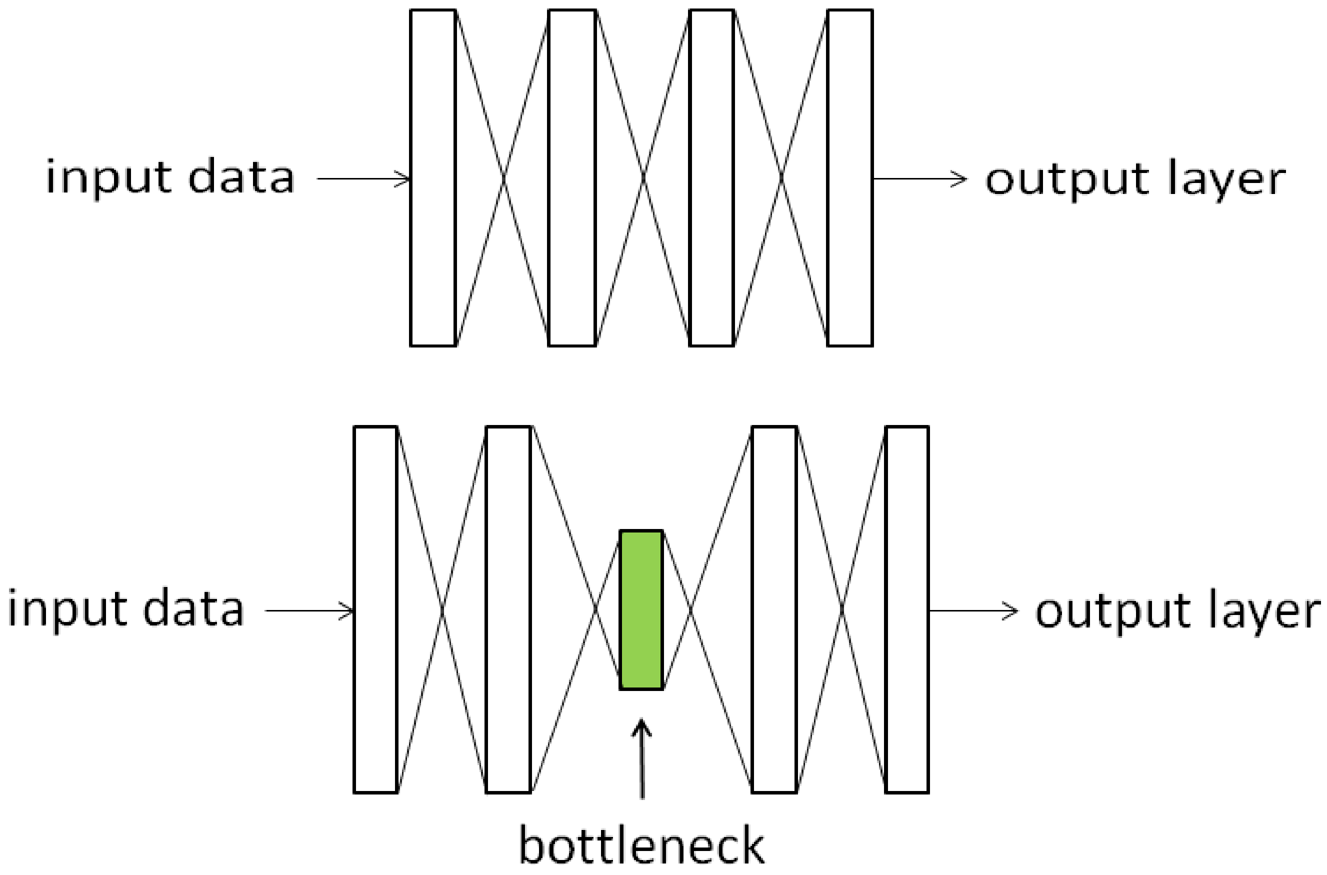
Input learning rate μ_t and number of workers M;
initialize $t \longleftarrow 0$;
initialize model $\theta^{(0)}$;
repeat
  Send $\theta^{(t)}$ to each workers;
  Wait for $G_1^{(t)}, \cdots G_M^{(t)}$ from each workers;
  $\theta^{(t+1)} \longleftarrow \theta^{(t)} - \frac{\mu_t}{N} \sum_{l=1}^{M} G_l^{(t)}$;
  $t \longleftarrow t + 1$
until *converges*;

ASYNCHRONOUS UPDATE

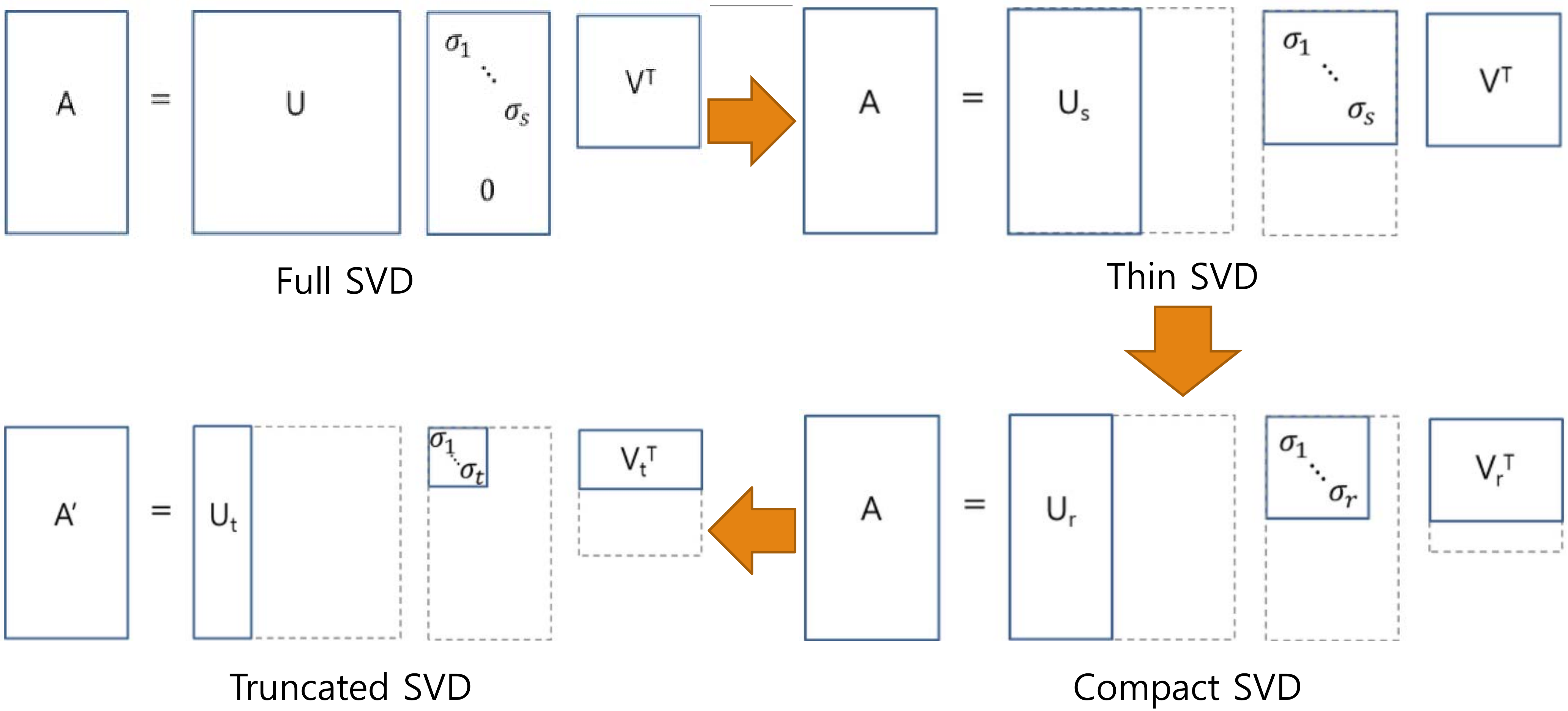
Algorithm 5: ASGD parameter server

Input learning rate μ_t and number of workers M;
initialize $t \longleftarrow 0$;
initialize model $\theta^{(0)}$;
repeat
  Wait for g_m from any worker;
  $\theta^{(t+1)} \longleftarrow \theta^{(t)} - \mu_t g_m$;
  $t \longleftarrow t + 1$
until *converges*;

Matrix rank constraint



Truncated svd



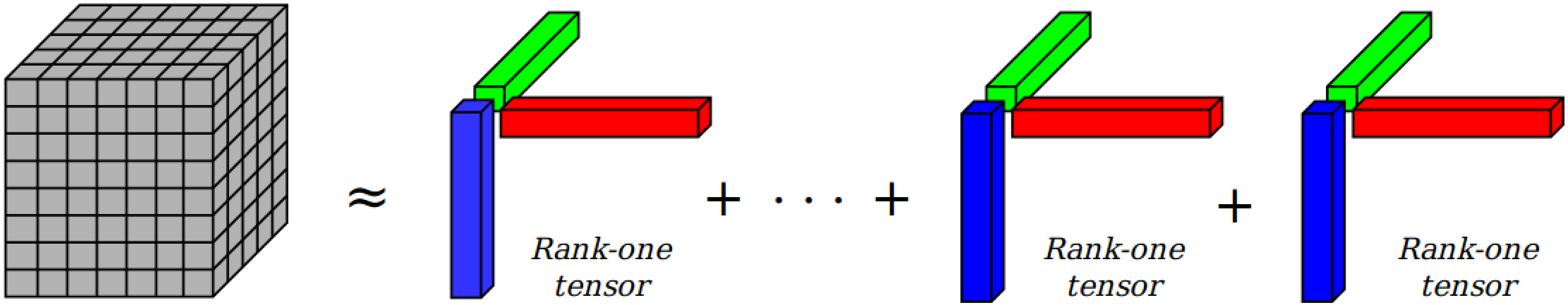
Truncated SVD

Original image: rank 402

Truncated svd: rank 80

Canonical Polyadic decomposition

Weight for Convolutional layer: 4-tensor



# of parameters: $T_1 T_2 T_3 T_4 \rightarrow R(T_1 + T_2 + T_3 + T_4)$

Low rank approximation

Model	TOP-5 Accuracy	Speed-up	Compression Rate
AlexNet	80.03%	1.	1.
BN Low-rank	80.56%	1.09	4.94
CP Low-rank	79.66%	1.82	5.
VGG-16	90.60%	1.	1.
BN Low-rank	90.47%	1.53	2.72
CP Low-rank	90.31%	2.05	2.75
GoogleNet	92.21%	1.	1.
BN Low-rank	91.88%	1.08	2.79
CP Low-rank	91.79%	1.20	2.84