Optimization and Deep Neural Networks

Zhouchen Lin
Peking University

April 18, 2019

Outline

- Optimization for Training DNNs
- Optimization for DNN Structure Design
- Conclusions

Outline

- Optimization for Training DNNs
- Optimization for DNN Structure Design
- Conclusions

Optimization for Training DNNs

- Stochastic gradient descent is the dominant method for training DNNs
  - ✓ Low computational cost
  - ✓ Good empirical performance: can help escape local stationary points
  - ✓ Relatively easy in supporting arbitrary network topology: using automatic differentiation
  - ✗ Vanishing or blow-up gradient
  - ✗ Cannot deal with nondifferentiable activation functions directly
  - ✗ Requires much manual tuning of optimization parameters such as learning rates and convergence criteria
  - ✗ Inherent sequential: difficult to parallelize

Quoc V. Le et al., On Optimization Methods for Deep Learning, ICML 2011.
Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Optimization for Training DNNs

- Other trials, as unconstrained problem
  - Layer-wise pre-training
  - Contrastive divergence
  - Stochastic diagonal Levenberg-Marquardt
  - Hessian free
  - L-BFGS
  - Conjugate gradient

$$\min_{\{W^i\}} \ell\left(\phi(W^{n-1}\phi(\cdots\phi(W^2\phi(W^1X^1))\cdots)), L\right).$$

Quoc V. Le et al., On Optimization Methods for Deep Learning, ICML 2011.
Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Optimization for Training DNNs

- Other trials, as constrained problem, using penalty method

$$\min_{\{W^i\},\{X^i\}} \ell(X^n, L)$$
$$\text{s.t. } X^i = \phi(W^{i-1}X^{i-1}),\ i = 2, 3, \cdots, n,$$

$$\boxed{\min_{\{W^i\},\{X^i\}} \ell(X^n, L) + \frac{\mu}{2}\sum_{i=2}^{n} \|X^i - \phi(W^{i-1}X^{i-1})\|_F^2.}$$

(Carreira-Perpinan and Wang, 2014)

$$\min_{\{W^i\},\{X^i\},\{U^i\}} \ell(X^n, L)$$
$$\text{s.t. } U^i = W^{i-1}X^{i-1}, X^i = \phi(U^i),\ i = 2, 3, \cdots, n.$$

$$\boxed{\begin{aligned}&\min_{\{W^i\},\{X^i\},\{U^i\}} \ell(X^n, L)\\ &+ \frac{\mu}{2}\sum_{i=2}^{n}(\|U^i - W^{i-1}X^{i-1}\|_F^2 + \|X^i - \phi(U^i)\|_F^2).\end{aligned}}$$

(Zeng et al. 2018)

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Optimization for Training DNNs

- Other trials, as constrained problem, using ADMM

$$\min_{\{W^i\},\{X^i\},\{U^i\}} \ell(X^n, L)$$
$$\text{s.t. } U^i = W^{i-1}X^{i-1}, X^i = \phi(U^i), \ i = 2, 3, \cdots, n.$$

$$\min_{\{W^i\},\{X^i\},\{U^i\},M} \ell(U^n, L) + \frac{\beta}{2}\left\|U^n - W^{n-1}X^{n-1} + M\right\|_F^2$$
$$+ \sum_{i=2}^{n-1} \frac{\mu_i}{2}\left(\|U^i - W^{i-1}X^{i-1}\|_F^2 + \|X^i - \phi(U^i)\|_F^2\right).$$

(Taylor et al. 2016)

$$\min_{\{W^i\},\{X^i\},\{U^i\}} \ell(X^n, L)$$
$$\text{s.t. } U^{i-1} = X^{i-1}, X^i = \phi(W^{i-1}U^{i-1}), \ i = 2, 3, \cdots, n.$$

$$\min_{\{W^i\},\{X^i\},\{U^i\},\{A^i\},\{B^i\}} \ell(X^n, L)$$
$$+ \frac{\mu}{2}\sum_{i=2}^{n}\left(\left\|U^{i-1} - X^{i-1} + A^{i-1}\right\|_F^2\right.$$
$$\left. + \left\|X^i - \phi(W^{i-1}U^{i-1}) + B^{i-1}\right\|_F^2\right).$$

(Zhang, Chen, and Saligrama, 2016)

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Optimization for Training DNNs

- Other trials, as constrained problem, using lifted objective function

$$
\begin{aligned}
X^i &= \phi(W^{i-1}X^{i-1}) \\
&= \max(W^{i-1}X^{i-1}, \mathbf{0}) \\
&= \underset{U^i \geq \mathbf{0}}{\operatorname{argmin}} \|U^i - W^{i-1}X^{i-1}\|_F^2.
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
&\min_{\{W^i\},\{X^i\}} \ell(X^n, L) + \sum_{i=2}^{n} \frac{\mu_i}{2}\|X^i - W^{i-1}X^{i-1}\|_F^2 \\
&\text{s.t. } X^i \geq \mathbf{0}, \; i = 2, 3, \cdots, n.
\end{aligned}
}
$$

(Zhang and Brand, 2017)

For ReLU only!

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Lifted Proximal Operator Machines for Training DNNs

$$x = \phi(y) \Longleftrightarrow x = \operatorname*{argmin}_x h(x, y)$$

$$\min_{x,y} g(x), \text{ s.t. } x = \phi(y) \Longrightarrow \min_{x,y} g(x) + \mu h(x, y)$$

Two commonly used operations in optimization:

proximal operator

$$x = y - f'(y) \quad \text{and} \quad \boxed{x = \operatorname*{argmin}_x f(x) + \frac{1}{2}(x - y)^2}$$

$$x = \phi(y) \Longleftrightarrow x = \operatorname*{argmin}_x f(x) + \frac{1}{2}(x - y)^2$$

$$\boxed{f(x) = \int_0^x (\phi^{-1}(y) - y)dy}$$

$$0 \in (\phi^{-1}(x) - x) + (x - y) \Longleftrightarrow x = \phi(y)$$

$f(x)$ is well defined (we allow f to take value of $+\infty$) even if $\phi^{-1}(y)$ is non-unique for some y between 0 and x. Anyway, ϕ^{-1}, f, and g (to be defined later) will *not* be explicitly used in our computation.

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Lifted Proximal Operator Machines for Training DNNs

Define $f(X) = (f(X_{kl}))$.

$$X^i = \underset{X^i}{\operatorname{argmin}}\, \mathbf{1}^T f(X^i)\mathbf{1} + \frac{1}{2}\|X^i - W^{i-1}X^{i-1}\|_F^2 \iff X^i = \phi(W^{i-1}X^{i-1}).$$

$$\min_{\{W^i\},\{X^i\}} \ell(X^n, L)$$

$$\text{s.t. } X^i = \phi(W^{i-1}X^{i-1}),\ i = 2, 3, \cdots, n,$$

$$\boxed{\min_{\{W^i\},\{X^i\}} \ell(X^n, L) + \sum_{i=2}^{n} \mu_i \left(\mathbf{1}^T f(X^i)\mathbf{1} + \frac{1}{2}\|X^i - W^{i-1}X^{i-1}\|_F^2\right).}$$

So far so good. But …

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Lifted Proximal Operator Machines for Training DNNs

However, its optimality conditions for $\{X^i\}_{i=2}^{n-1}$ are:

$$\mathbf{0} \in \mu_i(\phi^{-1}(X^i) - W^{i-1}X^{i-1}) + \mu_{i+1}(W^i)^T \boxed{(W^iX^i - X^{i+1})}, \; i = 2, \cdots, n-1.$$

The equality constraints

$$X^i = \phi(W^{i-1}X^{i-1})$$

do not satisfy the above!

We need the equality constraints for fast inference on new data samples!

$$\mathbf{0} \in \mu_i(\phi^{-1}(X^i) - W^{i-1}X^{i-1}) + \mu_{i+1}(W^i)^T \boxed{(\phi(W^iX^i) - X^{i+1})}, \; i = 2, \cdots, n-1.$$

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Lifted Proximal Operator Machines for Training DNNs

$$\boxed{\begin{aligned}\min_{\{W^i\},\{X^i\}} \quad & \ell(X^n, L) \\ & + \sum_{i=2}^{n} \mu_i \left(\mathbf{1}^T f(X^i) \mathbf{1} + \frac{1}{2} \| X^i - W^{i-1} X^{i-1} \|_F^2 \right).\end{aligned}}$$

$$\begin{aligned}\min_{\{W^i\},\{X^i\}} \ell(X^n, L) + \sum_{i=2}^{n} \mu_i \Big(& \mathbf{1}^T f(X^i) \mathbf{1} \\ & \boxed{+ \mathbf{1}^T g(W^{i-1} X^{i-1}) \mathbf{1}} + \frac{1}{2} \| X^i - W^{i-1} X^{i-1} \|_F^2 \Big),\end{aligned}$$

$$\boxed{g(x) = \int_0^x (\phi(y) - y) dy.}$$

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Lifted Proximal Operator Machines for Training DNNs

Table 1. The $f(x)$ and $g(x)$ of several representative activation functions. Note that $0 < \alpha < 1$ for the leaky ReLU function and $\alpha > 0$ for the exponential linear unit (ELU) function. We only use $\phi(x)$ in our computation and do NOT explicitly use $\phi^{-1}(x)$, $f(x)$, and $g(x)$. So all these activation functions and many others can be used in LPOM.

function	$\phi(x)$	$\phi^{-1}(x)$	$f(x)$	$g(x)$
sigmoid	$\frac{1}{1+e^{-x}}$	$\log \frac{x}{1-x}$ $(0 < x < 1)$	$\begin{cases} x\log x + (1-x)\log(1-x) - \frac{x^2}{2}, & 0<x<1 \\ +\infty, & \text{otherwise} \end{cases}$	$\log(e^x+1) - \frac{x^2}{2}$
tanh	$\frac{e^x-e^{-x}}{e^x+e^{-x}}$	$\frac{1}{2}\log\frac{1+x}{1-x}$ $(-1 < x < 1)$	$\begin{cases} \frac{1}{2}[(1-x)\log(1-x) + (1+x)\log(1+x))] - \frac{x^2}{2}, & -1<x<1 \\ +\infty, & \text{otherwise} \end{cases}$	$\log(\frac{e^x+e^{-x}}{2}) - \frac{x^2}{2}$
ReLU	$\max(x,0)$	$\begin{cases} x, & x>0 \\ (-\infty,0), & x=0 \end{cases}$	$\begin{cases} 0, & x\geq 0 \\ +\infty, & \text{otherwise} \end{cases}$	$\begin{cases} 0, & x\geq 0 \\ -\frac{1}{2}x^2, & x<0 \end{cases}$
leaky ReLU	$\begin{cases} x, & x\geq 0 \\ \alpha x, & x<0 \end{cases}$	$\begin{cases} x, & x\geq 0 \\ x/\alpha, & x<0 \end{cases}$	$\begin{cases} 0, & x\geq 0 \\ \frac{1-\alpha}{2\alpha}x^2, & x<0 \end{cases}$	$\begin{cases} 0, & x\geq 0 \\ \frac{\alpha-1}{2}x^2, & x<0 \end{cases}$
ELU	$\begin{cases} x, & x\geq 0 \\ \alpha(e^x-1), & x<0 \end{cases}$	$\begin{cases} x, & x\geq 0 \\ \log(1+\frac{x}{\alpha}), & x<0 \end{cases}$	$\begin{cases} 0, & x\geq 0 \\ (\alpha+x)(\log(\frac{x}{\alpha}+1)-1) - \frac{x^2}{2}, & x<0 \end{cases}$	$\begin{cases} 0, & x\geq 0 \\ \alpha(e^x-x) - \frac{x^2}{2}, & x<0 \end{cases}$
softplus	$\log(1+e^x)$	$\log(e^x-1)$	No analytic expression	No analytic expression

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Lifted Proximal Operator Machines for Training DNNs

- Good Property of LPOM

The subproblems of penalty and ADMM methods are all **nonconvex**!

Denote the objective function of LPOM as $F(W, X)$.

Theorem 4. *Suppose $\ell(X^n, L)$ is convex in X^n and ϕ is non-decreasing. Then $F(W, X)$ is block multi-convex, i.e., convex in each X^i and W^i if all other blocks of variables are fixed.*

1. Can use Block Coordinate Descent
2. The optimal solutions for updating X^i and W^i can be obtained

Proof. $F(W, X)$ can be simplified to

$$F(W,X) = \ell(X^n, L) + \sum_{i=2}^{n} \mu_i \Big(\mathbf{1}^T \tilde{f}(X^i)\mathbf{1} + \mathbf{1}^T \tilde{g}(W^{i-1}X^{i-1})\mathbf{1} - \langle X^i, W^{i-1}X^{i-1} \rangle \Big), \tag{1}$$

where $\tilde{f}(x) = \int_0^x \phi^{-1}(y)dy$ and $\tilde{g}(x) = \int_0^x \phi(y)dy$. $\square$

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Solving LPOM

- Overall algorithm

Thanks to the block multi-convexity, LPOM can be solved by Block Coordinate Descent. Namely, we update X^i or W^i by fixing all other blocks of variables. The optimization can be performed using a mini-batch of training samples.

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Solving LPOM

Updating $\{X^i\}_{i=2}^{n-1}$

For $i = 2, \cdots, n-1$, with $\{W^i\}_{i=1}^{n-1}$ and other $\{X^j\}_{j=2, j\neq i}^{n}$ fixed, the objective function of LPOM reduces to

$$\begin{aligned}\min_{X^i} \mu_i & \left(\mathbf{1}^T f(X^i)\mathbf{1} + \frac{1}{2}\|X^i - W^{i-1}X^{i-1}\|_F^2\right) \\ + \mu_{i+1} & \left(\mathbf{1}^T g(W^i X^i)\mathbf{1} + \frac{1}{2}\|X^{i+1} - W^i X^i\|_F^2\right).\end{aligned}$$

Optimality condition:

$$\begin{aligned}\mathbf{0} \in & \mu_i(\phi^{-1}(X^i) - W^{i-1}X^{i-1}) \\ & + \mu_{i+1}((W^i)^T(\phi(W^i X^i) - X^{i+1})).\end{aligned}$$

Fix-point iteration: $\boxed{\mathbf{x} = f(\mathbf{x}) \Longrightarrow \mathbf{x}_{k+1} = f(\mathbf{x}_k)}$

$$X^{i,t+1} = \phi\left(W^{i-1}X^{i-1} - \frac{\mu_{i+1}}{\mu_i}(W^i)^T(\phi(W^i X^{i,t}) - X^{i+1})\right)$$

$\boxed{\text{Only } \phi\text{! No its inverse or derivative!}}$

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Solving LPOM

Updating X^n
LPOM reduces to

$$\min_{X^n} \ell(X^n, L) + \mu_n \left(\mathbf{1}^T f(X^i) \mathbf{1} + \frac{1}{2} \| X^n - W^{n-1} X^{n-1} \|_F^2 \right).$$

Optimality condition (assuming that the loss function is differentiable w.r.t. X^n):

$$\mathbf{0} \in \frac{\partial \ell(X^n, L)}{\partial X^n} + \mu_n (\phi^{-1}(X^n) - W^{n-1} X^{n-1}).$$

Fix-point iteration:

$$X^{n,t+1} = \phi \left(W^{n-1} X^{n-1} - \frac{1}{\mu_n} \frac{\partial \ell(X^{n,t}, L)}{\partial X^n} \right)$$

$\boxed{\text{Only } \phi\text{! No its inverse or derivative!}}$

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Solving LPOM

Updating $\{W^i\}_{i=1}^{n-1}$

$\{W^i\}_{i=1}^{n-1}$ can be updated with full parallelization. When $\{X^i\}_{i=2}^{n}$ are fixed, the objective of LPOM reduces to

$$\min_{W^i} \mathbf{1}^T g(W^i X^i)\mathbf{1} + \frac{1}{2}\|W^i X^i - X^{i+1}\|_F^2, \ i = 1, \cdots, n-1, \tag{1}$$

which can be solved in parallel. (1) can be rewritten as

$$\min_{W^i} \mathbf{1}^T \tilde{g}(W^i X^i)\mathbf{1} - \langle X^{i+1}, W^i X^i \rangle, \tag{2}$$

where $\tilde{g}(x) = \int_0^x \phi(y)dy$, as introduced before.

Suppose that $\phi(x)$ is β-Lipschitz continuous. Then $\tilde{g}(x)$ is β-smooth:

$$|\tilde{g}'(x) - \tilde{g}'(y)| = |\phi(x) - \phi(y)| \leq \beta|x - y|.$$

(2) can be solved by APG via locally linearizing $\hat{g}(W) \equiv \tilde{g}(WX^i)$.

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Solving LPOM

The iteration is:

$$W^{i,t+1} = Y^{i,t} - \frac{1}{\beta}(\phi(Y^{i,t}X^{i}) - X^{i+1})(X^{i})^{\dagger},$$

where $Y^{i,t}$ is an extrapolation of $W^{i,t}$ and $W^{i,t-1}$.

$\boxed{\text{Only } \phi\text{! No its inverse or derivative!}}$

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Parallelizing LPOM

The update of $\{W^i\}$ is already parallel. The serial update procedure of $\{X^i\}$ can be easily changed to parallel update: each X^i is updated using the latest information of other X^j's, $j \neq i$.

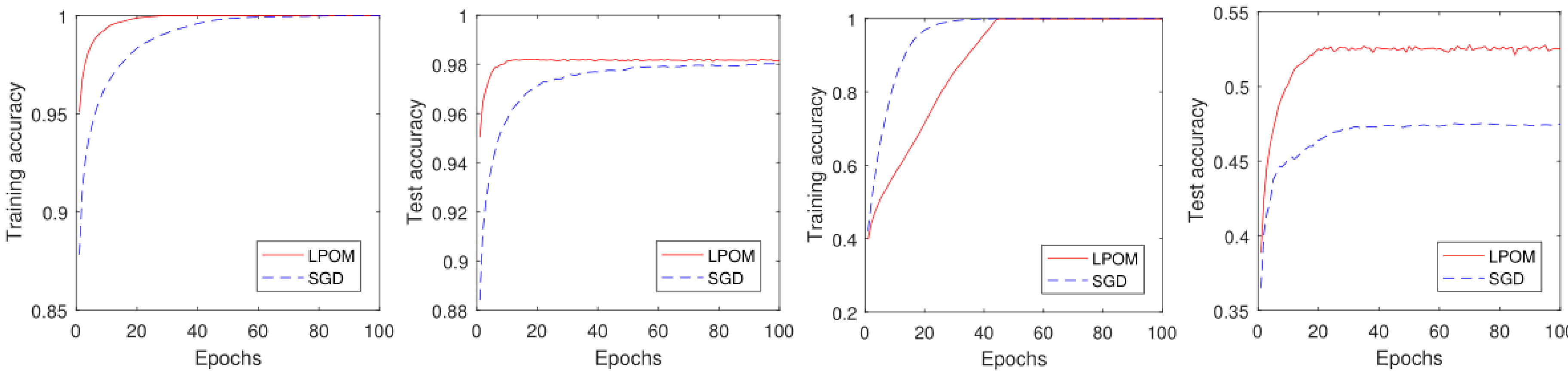
Asynchronous parallelization!

SGD can only be parallelized at the implementation level, not the algorithmic level.

Paper submitted to IJCAI 2019.

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Experiments



(a) MNIST (Training Acc.) (b) MNIST (Test Acc.) (c) CIFAR-10 (Training Acc.) (d) CIFAR-10 (Test Acc.)

Figure 1: Comparison of LPOM and SGD on the MNIST and the CIFAR-10 datasets.

Table 2: Comparison of accuracies of LPOM and (Askari et al., 2018) on the MNIST dataset using different networks.

Hidden layers	300	300-100	500-150	500-200-100	400-200-100-50
(Askari et al., 2018)	89.8%	87.5%	86.5%	85.3%	77.0%
LPOM	97.7%	96.9%	97.1%	96.2%	96.1%

Table 3: Comparison with SGD and (Taylor et al., 2016) on the SVHD dataset.

SGD	95.0%
(Taylor et al., 2016)	96.5%
LPOM	98.3%

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Advantages of LPOM

When compared with ADMM based methods, LPOM does not require Lagrange multipliers and more auxiliary variables than $\{X^i\}_{i=2}^n$.

Moreover, we have designed delicate algorithms so that no auxiliary variables are needed either when solving LPOM (to be introduced). So LPOM has much less variables than ADMM based methods and hence saves memory greatly. Actually, its memory cost equals to that of SGD.

Advantages of LPOM

When compared with the penalty methods, the optimality conditions of LPOM are simpler, e.g.:

(LPOM) $$(\phi(W^iX^i)-X^{i+1})(X^i)^T=\mathbf{0}, \quad i=1,\cdots,n-1.$$

(MAC) $$[(\phi(W^iX^i)-X^{i+1})\circ\phi'(W^iX^i)](X^i)^T=\mathbf{0}, i=1,\cdots,n-1.$$

This may imply that the solution sets of MAC and others are more complex and also "larger" than that of LPOM. So it may be easier to find good solutions of LPOM.

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

Advantages of LPOM

When compared with the convex optimization reformulation methods, LPOM can handle much more general activation functions, rather than ReLU only.

When compared with gradient based methods, such as SGD, LPOM can work with any non-decreasing Lipschitz continuous activation function without numerical difficulties, including being saturating (e.g., sigmoid and tanh) and non-differentiable (e.g., ReLU and leaky ReLU) and can update the layer-wise weights and activations in parallel. Moreover, LPOM only needs to tune the penaltys μ_i's, which is much easier than tuning the learning rates of SGD.

Jia Li, Cong Fang, and Zhouchen Lin, Lifted Proximal Operator Machines, AAAI 2019.

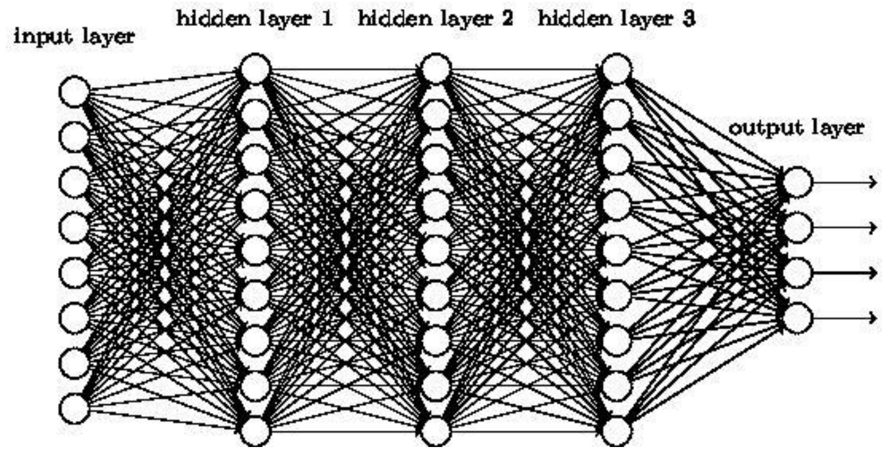
Future Work

- Extend to CNNs
  - Support more operations, e.g., pooling and batch normalization
  - Support arbitrary topology

Outline

- Optimization for Training DNNs
- Optimization for DNN Structure Design
- Conclusions

Analogy between DNN and Optimization



$$\mathbf{x}_{k+1} = \phi(\mathbf{W}_k \mathbf{x}_k)$$

$$\mathbf{x}_{k+1} = g(\mathbf{x}_k, \mathbf{x}_{k-1}, \nabla f(\mathbf{x}_k))$$

- Optimization Inspired DNNs for Image Reconstruction
- Optimization Inspired DNNs for Image Recognition

- Optimization Inspired DNNs for Image Reconstruction
- Optimization Inspired DNNs for Image Recognition

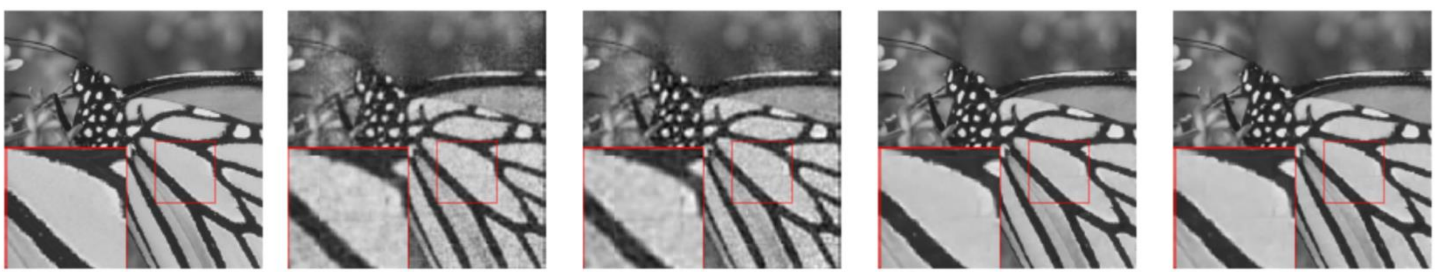
Optimization Inspired DNNs for Image Reconstruction

- ISTA-Net
- FISTA-Net
- ADMM-Net

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 + \sum_{l=1}^{L} \lambda_l g(\mathbf{D}_l \mathbf{x})$$

$$\min_{\mathbf{x},\mathbf{z}_l} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 + \sum_{l=1}^{L} \lambda_l g(\mathbf{z}_l), \quad s.t. \quad \mathbf{z}_l = \mathbf{D}_l \mathbf{x}$$

- Rewrite the iterative algorithm to solve the image reconstruction model as neural networks
- The activation functions are usually the soft thresholding operator



(a) Original (b) SDA (c) ReconNet (d) ISTA-Net (e) FISTA-Net

Jian Zhang and Bernard Ghanem, ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing, CVPR 2018.

Jian Zhang and Bernard Ghanem, ISTA-Net - Iterative Shrinkage-Thresholding Algorithm Inspired Deep Network for Image Compressive Sensing, arXiv, 2017.

Y. Yang, J. Sun, H. Li, and X. Zongben. Deep ADMM-net for compressive sensing MRI, NIPS, 2016.

- Optimization Inspired DNNs for Image Reconstruction
- Optimization Inspired DNNs for Image Recognition

Gradient Descent

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

Theorem 1. *Let f be convex and β-smooth. Then gradient descent with $\eta = \beta^{-1}$ satisfies*

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{2\beta \|\mathbf{x}_1 - \mathbf{x}^*\|^2}{t-1}.$$

Heavy-Ball Method

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1}), \quad \alpha_k > 0, \beta_k > 0.$$

Image 2: SGD without momentum

Image 3: SGD with momentum

Theorem 2. *Let f be convex and L-smooth. Then the heavy-ball method with* $\alpha_k = \dfrac{\alpha_0}{k+2}$ *and* $\beta_k = \dfrac{k}{k+2}$, *where* $\alpha_0 \in (0, L^{-1}]$, *satisfies*

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{2\alpha_0(t+1)}.$$

Heavy-Ball Method

Heaby-ball is faster than gradient descent when f is μ-strongly convex:

$$\|\mathbf{x}_k - \mathbf{x}^*\| \le q^k \|\mathbf{x}_0 - \mathbf{x}^*\|,$$

where

$$q = \begin{cases} \dfrac{L-\mu}{L+\mu}, & \text{gradient descent} \\ \dfrac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}, & \text{heavy-ball} \end{cases} \tag{1}$$

Nesterov's Accelerated Gradient Descent

Nesterov's accelerated algorithm:

$$\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}, \text{ and } \gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}.$$

(Note that $\gamma_t \leq 0$.) Now the algorithm is simply defined by the following equations, with $\mathbf{x}_1 = \mathbf{y}_1$ an arbitrary initial point,

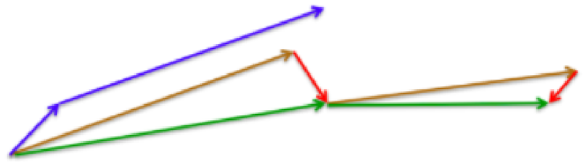


Image 4: Nesterov update

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \frac{1}{\beta}\nabla f(\mathbf{x}_t),$$

$$\mathbf{x}_{t+1} = (1-\gamma_s)\mathbf{y}_{t+1} + \gamma_t \mathbf{y}_t.$$

Theorem 3. *Let f be a convex and β-smooth function, then Nesterov's accelerated gradient descent satisfies*

$$f(\mathbf{y}_t) - f(\mathbf{x}^*) \leq \frac{2\beta\|\mathbf{x}_1 - \mathbf{x}^*\|^2}{t^2}.$$

Alternating Direction Method (ADM)

Model Problem:

$$\begin{aligned} \min_{\mathbf{x}_1,\mathbf{x}_2} \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2), \\ s.t. \quad & \mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2) = \mathbf{b}, \end{aligned}$$

where f_i are convex functions and $\mathcal{A}_i$ are linear mappings.

$$\begin{aligned} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}) \quad = \quad & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \langle \boldsymbol{\lambda}, \mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2) - \mathbf{b} \rangle \\ & + \frac{\beta}{2} \| \mathcal{A}_1(\mathbf{x}_1) + \mathcal{A}_2(\mathbf{x}_2) - \mathbf{b} \|_F^2, \end{aligned}$$

$$\begin{aligned} \mathbf{x}_1^{k+1} \quad &= \quad \arg\min_{\mathbf{x}_1} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2^k, \boldsymbol{\lambda}^k), \\ \mathbf{x}_2^{k+1} \quad &= \quad \arg\min_{\mathbf{x}_2} \mathcal{L}(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \boldsymbol{\lambda}^k), \\ \boldsymbol{\lambda}^{k+1} \quad &= \quad \boldsymbol{\lambda}^k + \beta_k [\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^{k+1}) - \mathbf{b}]. \end{aligned}$$

Assume: Easy

Update β_k

Optimization Inspired DNNs for Recognition

- genetic algorithm (Schaffer et al., 1992; Lam et al., 2003)
  - perform worse than the hand-crafted ones
- Bayesian optimization (Domhan et al. 2015)
- meta-modeling approach (Bergstra et al. 2013)
- adaptive strategy (Kwok and Yeung 1997, Ma and Khorasani 2003, Cortes et al. 2017)
- reinforcement learning (Baker et al. 2016, Zoph and Le 2017)

$\boxed{\text{Heuristic, computation intensive, domain knowledge required}}$

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Key Observation

$$\mathbf{x}_{k+1} = \Phi(\mathbf{W}_k \mathbf{x}_k). \tag{1}$$

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \nabla f(\mathbf{z}_k). \tag{2}$$

We do not consider the optimal weights during the structure design stage. Thus, we fix the matrix $\mathbf{W}_k$ as $\mathbf{W}$ to simplify the analysis.

Lemma 4. *Suppose* $\mathbf{W}$ *is a symmetric and positive definite matrix. Let* $\mathbf{U} = \mathbf{W}^{1/2}$. *Then there exists a function* $f(\mathbf{x})$ *such that (1) is equivalent to minimizing* $F(\mathbf{x}) = f(\mathbf{U}\mathbf{x})$ *using the following steps:*

1. *Define a new variable* $\mathbf{z} = \mathbf{U}\mathbf{x}$,
2. *Using (2) to minimize* $f(\mathbf{z})$,
3. *Recovering* $\mathbf{x}_k$ *from* $\mathbf{z}_k$ *via* $\mathbf{x} = \mathbf{U}^{-1}\mathbf{z}$.

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Key Observation

Table 1: The optimization objectives for the common activation functions.

	Activation function	Optimization objective $f(\mathbf{x})$
Sigmoid	$\frac{1}{1+e^{-x}}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \left[\mathbf{U}_i^T\mathbf{x} + \log\left(\frac{1}{e^{\mathbf{U}_i^T\mathbf{x}}} + 1\right)\right]$
tanh	$\frac{1-e^{-2x}}{1+e^{-2x}}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \left[\mathbf{U}_i^T\mathbf{x} + \log\left(\frac{1}{e^{2\mathbf{U}_i^T\mathbf{x}}} + 1\right)\right]$
Softplus	$\log(e^x + 1)$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \left[C - \mathrm{polylog}(2, -e^{\mathbf{U}_i^T\mathbf{x}})\right]$
Softsign	$\frac{x}{1+\|x\|}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \phi_i(\mathbf{x})$, where $\phi_i(\mathbf{x}) = \begin{cases} \mathbf{U}_i^T\mathbf{x} - \log(\mathbf{U}_i^T\mathbf{x}+1), & \text{if } \mathbf{U}_i^T\mathbf{x} > 0, \\ -\mathbf{U}_i^T\mathbf{x} - \log(\mathbf{U}_i^T\mathbf{x}-1), & \text{otherwise} \end{cases}$
ReLU	$\begin{cases} x, & \text{if } x > 0, \\ 0, & \text{if } x \le 0. \end{cases}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \phi_i(\mathbf{x})$, where $\phi_i(\mathbf{x}) = \begin{cases} \frac{(\mathbf{U}_i^T\mathbf{x})^2}{2}, & \text{if } \mathbf{U}_i^T\mathbf{x} > 0, \\ 0, & \text{otherwise} \end{cases}$
Leaky ReLU	$\begin{cases} x, & \text{if } x > 0, \\ \alpha x, & \text{if } x \le 0. \end{cases}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \phi_i(\mathbf{x})$, where $\phi_i(\mathbf{x}) = \begin{cases} \frac{(\mathbf{U}_i^T\mathbf{x})^2}{2}, & \text{if } \mathbf{U}_i^T\mathbf{x} > 0, \\ \frac{\alpha(\mathbf{U}_i^T\mathbf{x})^2}{2}, & \text{otherwise} \end{cases}$
ELU	$\begin{cases} x, & \text{if } x > 0, \\ a(e^x - 1), & \text{if } x \le 0. \end{cases}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \phi_i(\mathbf{x})$, where $\phi_i(\mathbf{x}) = \begin{cases} \frac{(\mathbf{U}_i^T\mathbf{x})^2}{2}, & \text{if } \mathbf{U}_i^T\mathbf{x} > 0, \\ a(e^{\mathbf{U}_i^T\mathbf{x}} - \mathbf{U}_i^T\mathbf{x}), & \text{otherwise} \end{cases}$
Swish	$\frac{x}{1+e^{-x}}$	$\frac{\|\mathbf{x}\|^2}{2} - \sum_i \left[\frac{(\mathbf{U}_i^T\mathbf{x})^2}{2} + \mathbf{U}_i^T\mathbf{x}\log\left(\frac{1}{e^{\mathbf{U}_i^T\mathbf{x}}} + 1\right) - \mathrm{polylog}\left(2, -\frac{1}{e^{\mathbf{U}_i^T\mathbf{x}}}\right)\right]$

Networks for image reconstruction only aim at solving:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \sum_{l=1}^{L} \lambda_l g(\mathbf{D}_l\mathbf{x})$$

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Hypothesis

- Faster algorithms may lead to better DNNs
  - DNNs are computing features
  - We want to compute features as quickly as possible and we want as shallow as possible networks
  - Faster algorithms can compute features quicker

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

From GD to Other Optimization Algorithms

The Heavy Ball Algorithm. Following the three steps, we have:

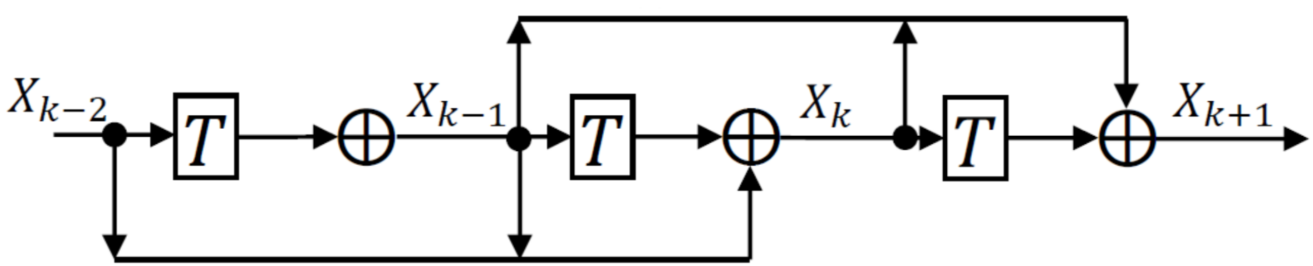
1. Variable substitution $\mathbf{z} = \mathbf{U}\mathbf{x}$.

2. Using the heavy-ball algorithm to minimize $f(\mathbf{z})$:

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \nabla f(\mathbf{z}_k) + \beta(\mathbf{z}_k - \mathbf{z}_{k-1}) = \mathbf{U}\Phi(\mathbf{U}\mathbf{z}_k) + \beta(\mathbf{z}_k - \mathbf{z}_{k-1}); \quad (1)$$

3. Recovering $\mathbf{x}$ from $\mathbf{z}$ via $\mathbf{x} = \mathbf{U}^{-1}\mathbf{z}$:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{U}^{-1}\mathbf{z}_{k+1} = \Phi(\mathbf{U}\mathbf{z}_k) + \beta(\mathbf{U}^{-1}\mathbf{z}_k - \mathbf{U}^{-1}\mathbf{z}_{k-1}) \\ &= \Phi(\mathbf{U}^2\mathbf{x}_k) + \beta(\mathbf{x}_k - \mathbf{x}_{k-1}) = \Phi(\mathbf{W}\mathbf{x}_k) + \beta(\mathbf{x}_k - \mathbf{x}_{k-1}). \end{aligned} \quad (2)$$



Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

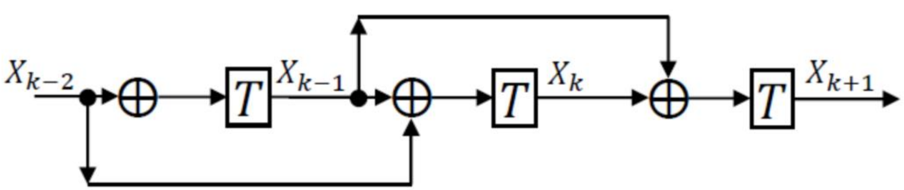
From GD to Other Optimization Algorithms

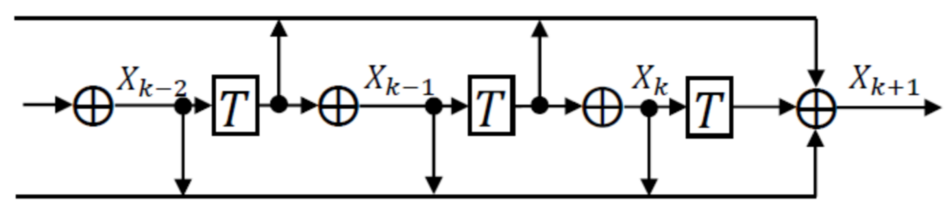
Nesterov's Accelerated Gradient Descent Algorithm. Following the same three steps, we have:

$$\mathbf{x}_{k+1} = \Phi(\mathbf{W}(\mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1}))); \tag{1}$$

where $\beta_k = \theta_k(1 - \theta_{k-1})/\theta_{k-1}$. An equivalent formulation:

$$\mathbf{x}_{k+1} = \sum_{j=0}^{k} h_{k+1;j}\Phi(\mathbf{W}\mathbf{x}_j) + \mathbf{x}_k - \sum_{j=0}^{k} h_{k+1;j}\mathbf{x}_j. \tag{2}$$



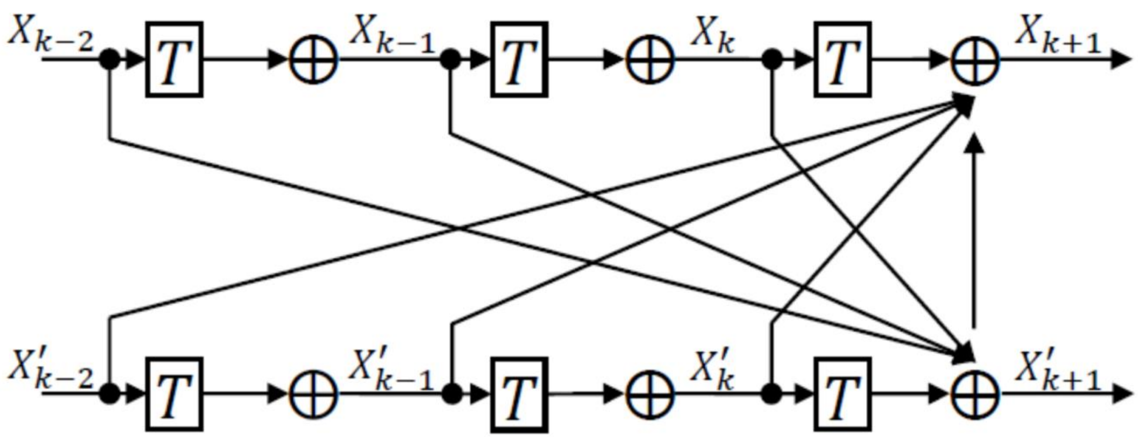


Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

From GD to Other Optimization Algorithms

ADMM.

$$
\begin{aligned}
\mathbf{x}'_{k+1} &= \frac{1}{2}\left(\Phi(\mathbf{W}\mathbf{x}'_k) + \mathbf{x}_k - \sum_{t=1}^{k}(\mathbf{x}'_t - \mathbf{x}_t)\right); \\
\mathbf{x}_{k+1} &= \frac{1}{2}\left(\Phi(\mathbf{W}\mathbf{x}_k) + \mathbf{x}'_{k+1} + \sum_{t=1}^{k}(\mathbf{x}'_t - \mathbf{x}_t)\right).
\end{aligned} \tag{1}
$$



Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Engineering Tricks

- Relax **W** to be learnable $\mathbf{W} \to \mathbf{W}_k$
- Relax Φ to be learnable and incorporate pooling and batch normalization $\Phi(\mathbf{W}\mathbf{x}) \to T(\mathbf{x})$
- Relax coefficients to be learnable

$$\mathbf{x}_{k+1} = \Phi(\mathbf{W}\mathbf{x}_k) + \beta(\mathbf{x}_k - \mathbf{x}_{k-1}) \to \mathbf{x}_{k+1} = T(\mathbf{x}_k) + \beta_1 \mathbf{x}_k + \beta_2 \mathbf{x}_{k-1}$$

$$\mathbf{x}_{k+1} = \Phi(\mathbf{W}(\mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1}))) \to \mathbf{x}_{k+1} = T(\beta_1 \mathbf{x}_k + \beta_2 \mathbf{x}_{k-1})$$

$$\mathbf{x}_{k+1} = \sum_{j=0}^{k} h_{k+1;} \Phi(\mathbf{W}\mathbf{x}_j) + \mathbf{x}_k - \sum_{j=0}^{k} h_{k+1;j} \mathbf{x}_j \to \mathbf{x}_{k+1} = \sum_{j=0}^{k} \alpha_{k+1}^{j} T(\mathbf{x}_j) + \sum_{j=0}^{k} \beta_{k+1}^{j} \mathbf{x}_j$$

$$\begin{aligned} \mathbf{x}'_{k+1} &= \frac{1}{2}\left(\Phi(\mathbf{W}\mathbf{x}'_k) + \mathbf{x}_k - \sum_{t=1}^{k}(\mathbf{x}'_t - \mathbf{x}_t)\right); \\ \mathbf{x}_{k+1} &= \frac{1}{2}\left(\Phi(\mathbf{W}\mathbf{x}_k) + \mathbf{x}'_{k+1} + \sum_{t=1}^{k}(\mathbf{x}'_t - \mathbf{x}_t)\right). \end{aligned} \to \begin{aligned} \mathbf{x}'_{k+1} &= T(\mathbf{x}'_k) + \sum_{t=1}^{k} \alpha_t \mathbf{x}'_t + \sum_{t=1}^{k} \beta_t \mathbf{x}_t; \\ \mathbf{x}_{k+1} &= T(\mathbf{x}_k) + \sum_{t=1}^{k} \alpha_t \mathbf{x}'_t + \sum_{t=1}^{k} \beta_t \mathbf{x}_t. \end{aligned}$$

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Engineering Tricks

$$\mathbf{x}_{k+1} = T(\mathbf{x}_k) + \beta_1 \mathbf{x}_k + \beta_2 \mathbf{x}_{k-1} \quad (16)$$

$$\mathbf{x}_{k+1} = \sum_{j=0}^{k} \alpha_{k+1}^{j} T(\mathbf{x}_j) + \sum_{j=0}^{k} \beta_{k+1}^{j} \mathbf{x}_j \quad (18)$$

$$\begin{aligned} \mathbf{x}'_{k+1} &= T(\mathbf{x}'_k) + \sum_{t=1}^{k} \alpha_t \mathbf{x}'_t + \sum_{t=1}^{k} \beta_t \mathbf{x}_t; \\ \mathbf{x}_{k+1} &= T(\mathbf{x}_k) + \sum_{t=1}^{k} \alpha_t \mathbf{x}'_t + \sum_{t=1}^{k} \beta_t \mathbf{x}_t. \end{aligned} \quad (20)$$

Algorithm	Network Structure	Transforming Setting
GD (1)	CNN	$\mathbf{Wx}$ $\rightarrow$convolution
HB (2)	ResNet	$\beta_2 = 0$ in (16)
AGD (4)	DenseNet	$\beta = 0$, $\alpha = 1$ in (18)
ADMM (6)	DMRNet	$\alpha_k = \beta_k = \frac{1}{2}$ in (20)

ResNet and DenseNet are special cases!

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Experiments

Model	CIFAR-10	CIFAR-100	CIFAR-10(+)	CIFAR-100(+)
ResNet ($n=9$)	10.05	39.65	5.32	26.03
HB-Net (16) ($n=9$)	10.17	38.52	5.46	26
ResNet ($n=18$)	9.17	38.13	5.06	24.71
HB-Net (16) ($n=18$)	8.66	36.4	5.04	23.93
DenseNet ($k=12, L=40$)*	7	27.55	5.24	24.42
AGD-Net (18) ($k=12, L=40$)	6.44	26.33	5.2	24.87
DenseNet ($k=12, L=52$)	6.05	26.3	5.09	24.33
AGD-Net (18) ($k=12, L=52$)	5.75	24.92	4.94	23.84

Error rates (%) on ImageNet when HB-Net and AGD-Net have the same depth as their baselines.

Model	top-1(%)	top-5(%)
ResNet-34	26.73	8.74
HB-Net-34	26.33	8.56
DenseNet-121	25.02	7.71
AGD-Net-121	24.62	7.39

Huan Li, Yibo Yang, and Zhouchen Lin, Optimization Algorithm Inspired Deep Neural Network Structure Design, ACML 2018

Discussions

- Can serve as a good initialization of AutoML and hand design.

Outline

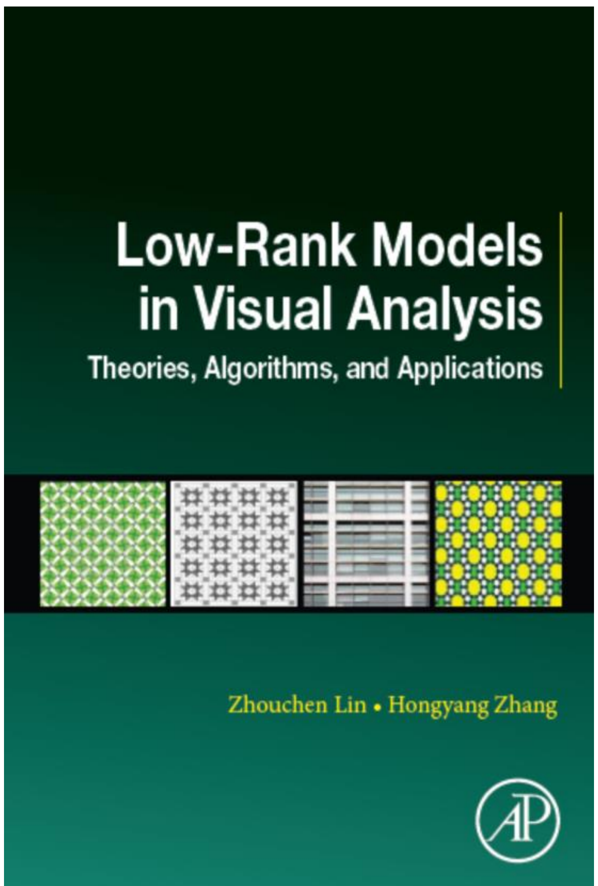
- Optimization for Training DNNs
- Optimization for DNN Structure Design
- Conclusions

Conclusions

- Optimization is an integral part of machine learning
- Optimization can not only help training DNNs, but also help structure design
- More interesting connections are yet to explore

Thanks!

- zlin@pku.edu.cn
- http://www.cis.pku.edu.cn/faculty/vision/zlin/zlin.htm



Recruitments

PKU: PostDoc (**270K** RMB/year) and Faculty

Samsung Beijing AI Lab: Researcher

之江Lab: Researcher and PostDoc

All in **machine learning** related areas

Please Google me and visit my webpage!