Learning Neural Networks with Adaptive Regularization

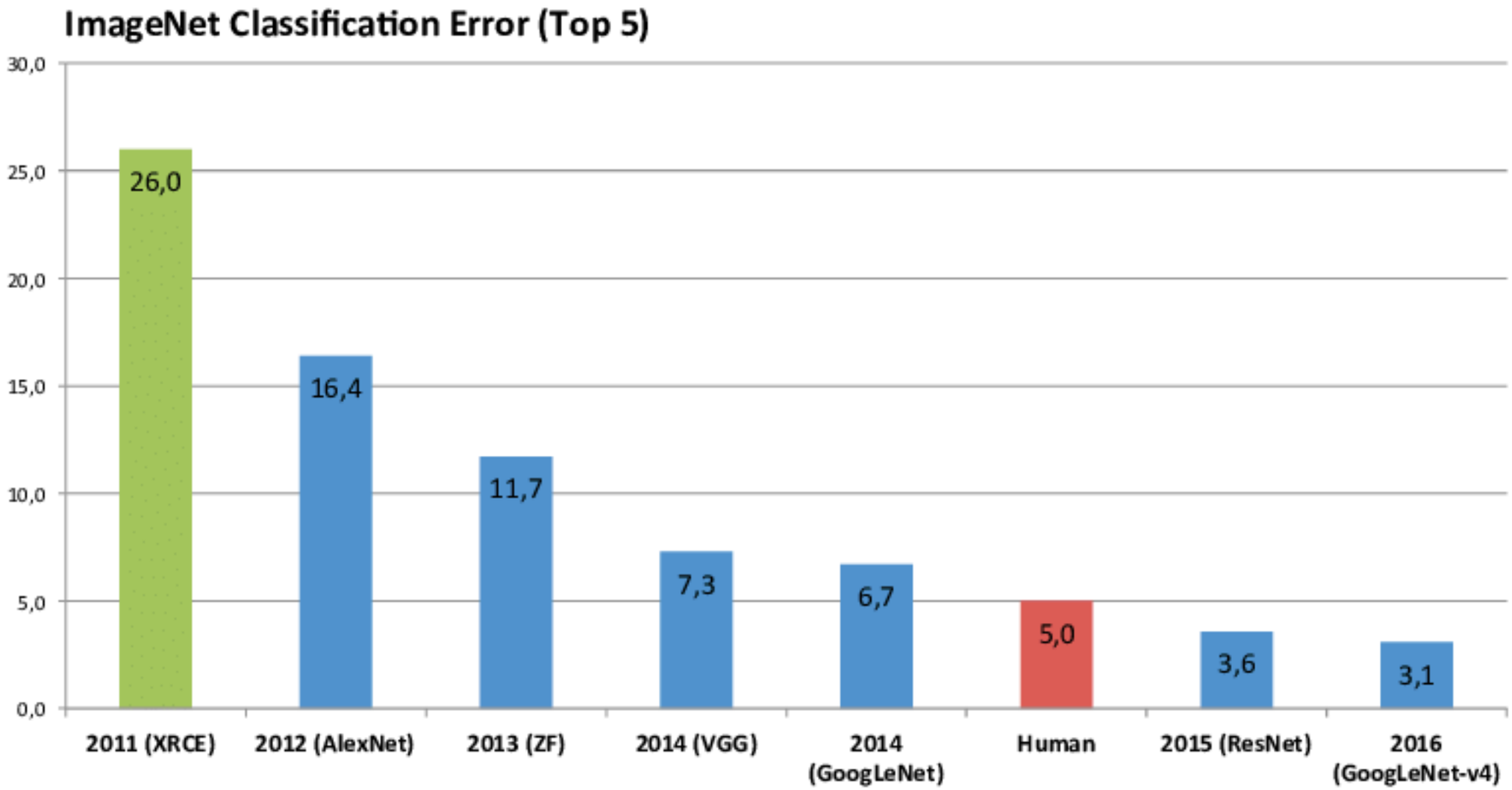
Han Zhao
han.zhao@cs.cmu.edu
Machine Learning Department, Carnegie Mellon University

Sep. 27th, 2019

Joint work with Yao-Hung Hubert Tsai, Ruslan Salakhutdinov, and Geoffrey J. Gordon

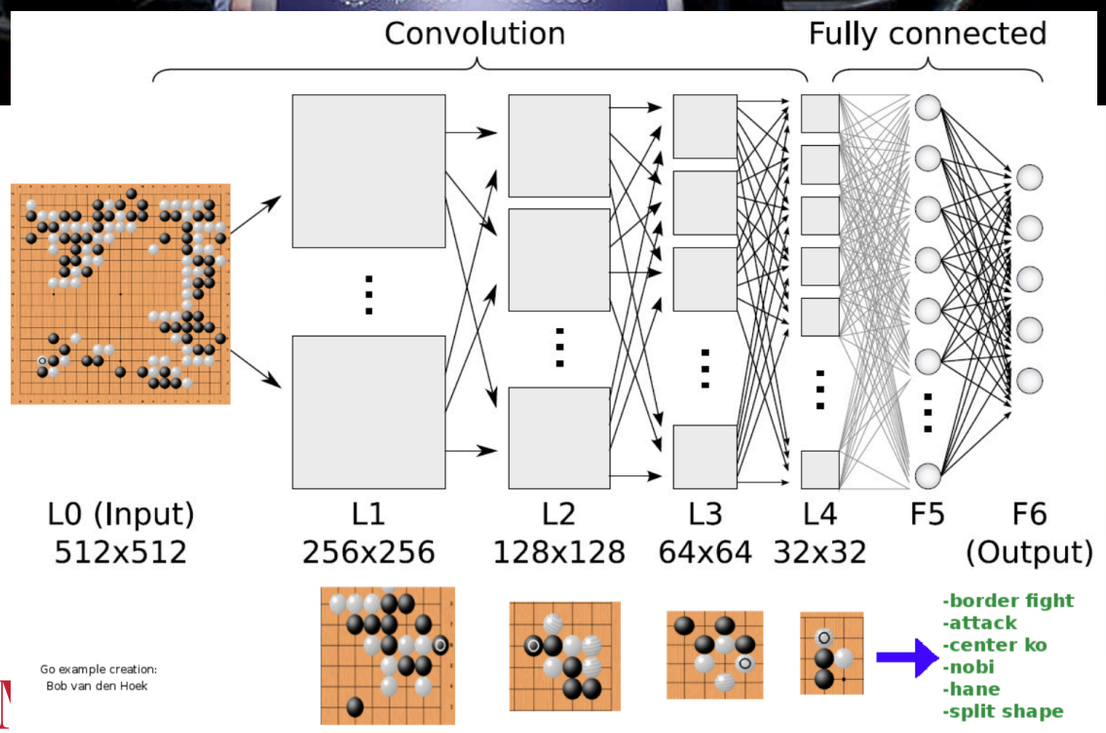
Recent Success of Deep Learning

ImageNet (Top 5) Classification Error:



Recent Success of Deep Learning

Go: AlphaGo vs Seudo Lee and Jie Ke



Go example creation:
Bob van den Hoek

Recent Success of Deep Learning

Dota 2: OpenAI Five

- Beats professional player on One-on-One game, 2017
- Beats amateur and semi-professional teams on Five-on-Five games, 2018

Recent Success of Deep Learning

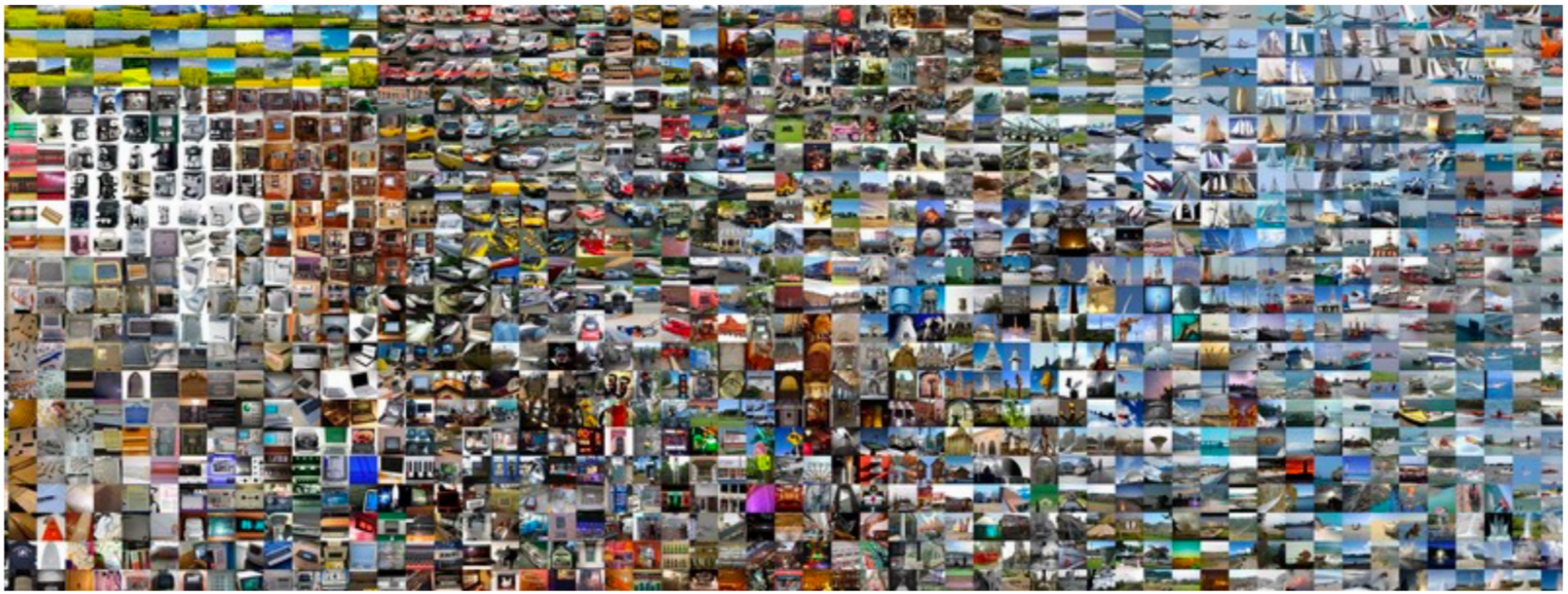
Other application domains:

- Natural Language Processing: Neural Machine Translation (2014)
- Artificial Game Playing: Atari (2015), AlphaGo (2016), StarCraft II (2018)
- Automatic Speech Recognition (2010)
- Computer Vision: Image generation (2014)
- Healthcare: Drug discovery…

Recent Success of Deep Learning

However, successful training of neural networks requires:

- Large-scale datasets
- Powerful computational resources



ImageNet: ~1M images, ~1K classes

Recent Success of Deep Learning

However, successful training of neural networks requires:

- Large-scale datasets
- Powerful computational resources

Configuration	Search threads	No. of CPU	No. of GPU	Elo rating
Single[10] p. 10–11	40	48	1	2,181
Single	40	48	2	2,738
Single	40	48	4	2,850
Single	40	48	8	2,890
Distributed	12	428	64	2,937
Distributed	24	764	112	3,079
Distributed	40	1,202	176	3,140
Distributed	64	1,920	280	3,168

Versions	Hardware	Elo rating	Matches
AlphaGo Fan	176 GPUs,[52] distributed	3,144[51]	5:0 against Fan Hui
AlphaGo Lee	48 TPUs,[52] distributed	3,739[51]	4:1 against Lee Sedol
AlphaGo Master	4 TPUs,[52] single machine	4,858[51]	60:0 against professional players; Future of Go Summit

Go: training data ~ almost infinity: simulated self-playing game

Recent Success of Deep Learning

However, successful training of neural networks requires:

- Large-scale datasets
- Powerful computational resources

How to effectively regularize the training of neural networks when there is only few data available?

Outline

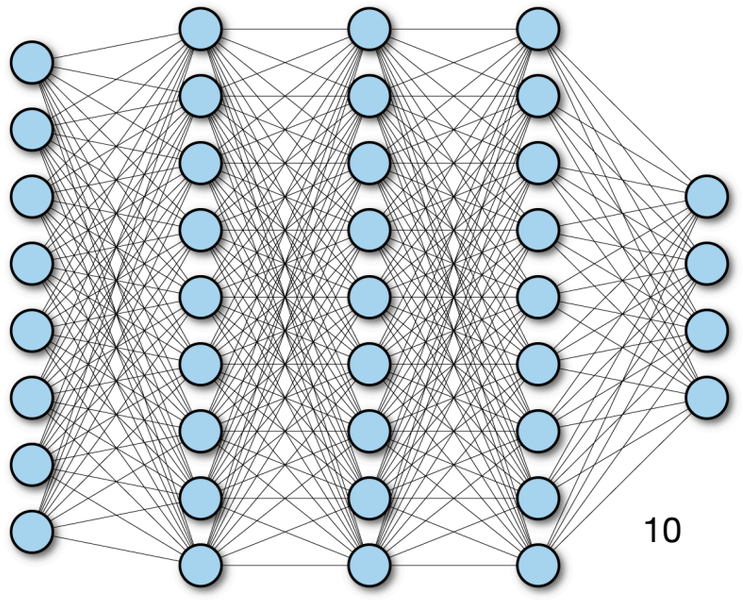
- Deep Learning Preliminary
- Learning with Adaptive Regularization
- Experiments
- Conclusion

Deep Learning Preliminary

A simple fully-connected neural network:

- Data set: $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}, \ \forall i \in [n]$
- Model:

$$f(\mathbf{x}; \theta) = \sigma(W_L g(W_{L-1} \cdots W_2 g(W_1 \mathbf{x})))$$



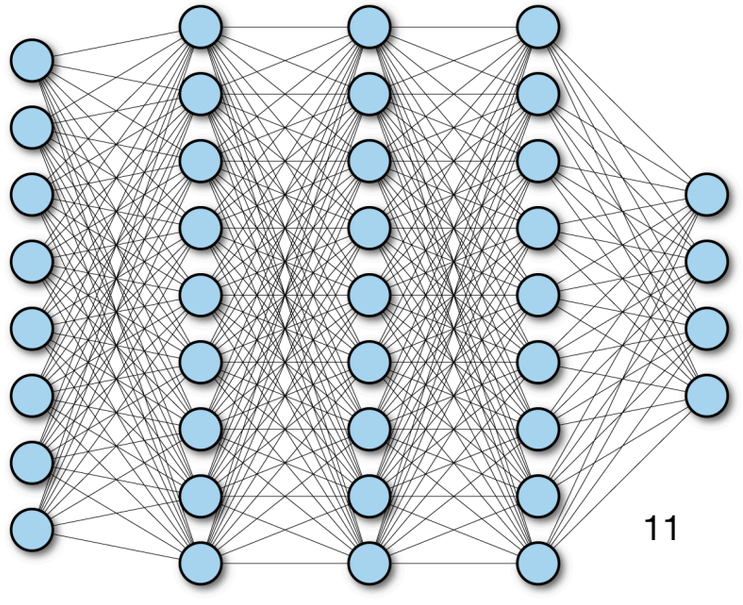
Deep Learning Preliminary

A simple fully-connected neural network:

- Data set: $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}, \ \forall i \in [n]$
- Model:

$$f(\mathbf{x}; \theta) = \sigma(W_L g(W_{L-1} \cdots W_2 g(W_1 \mathbf{x})))$$

- Output layer activation function: $\sigma(\cdot)$, the sigmoid/softmax function for classification, linear for regression



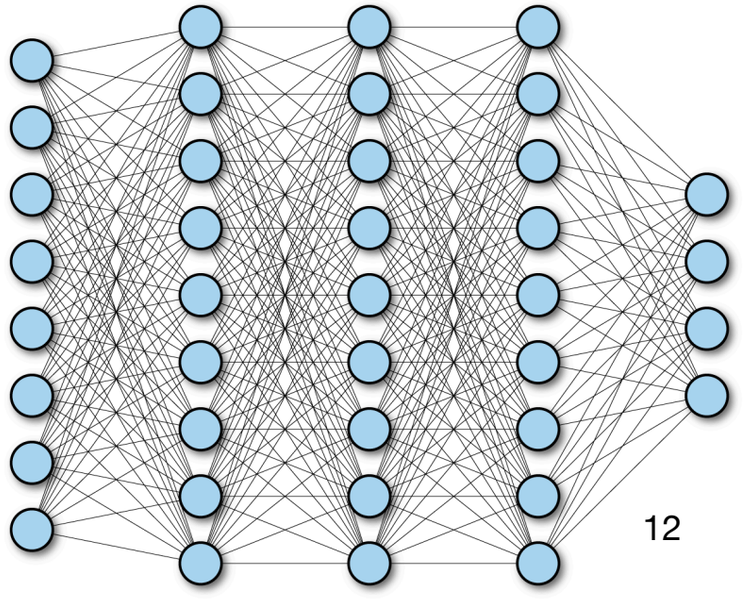
Deep Learning Preliminary

A simple fully-connected neural network:

- Data set: $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}, \ \forall i \in [n]$
- Model:

$$f(\mathbf{x}; \theta) = \sigma(W_L g(W_{L-1} \cdots W_2 g(W_1 \mathbf{x})))$$

- Output layer activation function: $\sigma(\cdot)$, the sigmoid/softmax function for classification, linear for regression
- Hidden layer activation function: $g(t) = \max\{0, t\}$, the ReLU function



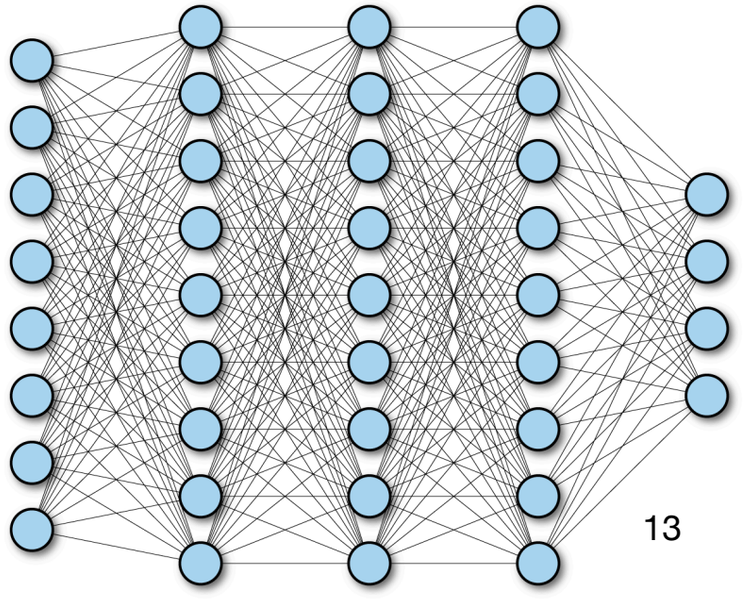
Deep Learning Preliminary

A simple fully-connected neural network:

- Data set: $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}, \ \forall i \in [n]$
- Model:

$$f(\mathbf{x}; \theta) = \sigma(W_L g(W_{L-1} \cdots W_2 g(W_1 \mathbf{x})))$$

- Output layer activation function: $\sigma(\cdot)$, the sigmoid/softmax function for classification, linear for regression
- Hidden layer activation function: $g(t) = \max\{0, t\}$, the ReLU function
- Essentially: a sequence of affine transformation + component-wise nonlinear activation

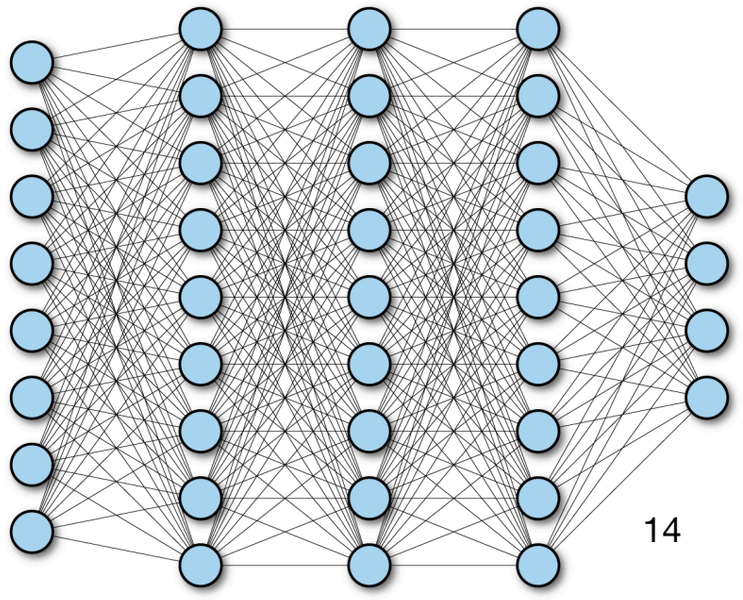


Deep Learning Preliminary

A simple fully-connected neural network:

- Loss function: Maximum Log-likelihood Estimation (MLE)

$$\hat{y} := f(\mathbf{x}; \theta) = \sigma\left(W_L g\left(W_{L-1} \cdots W_2 g\left(W_1 \mathbf{x}\right)\right)\right) \in (0, 1)$$



Deep Learning Preliminary

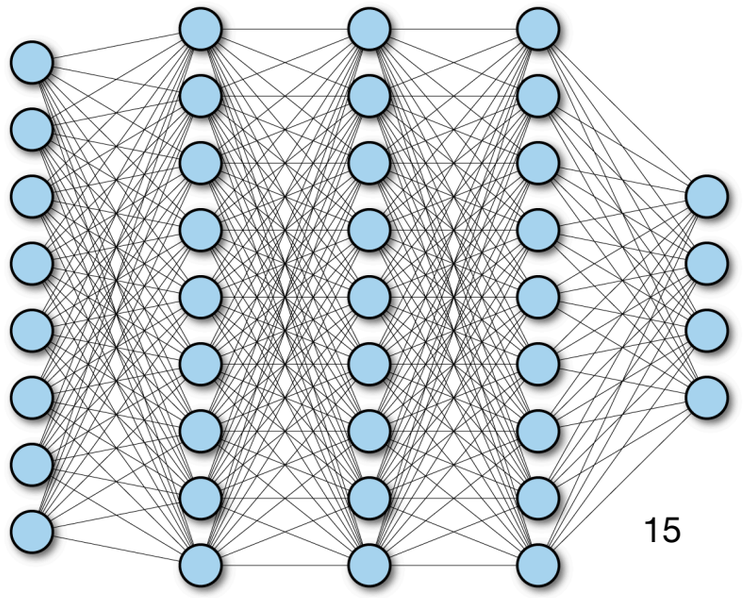
A simple fully-connected neural network:

- Loss function: Maximum Log-likelihood Estimation (MLE)

$$\hat{y} := f(\mathbf{x}; \theta) = \sigma\left(W_L g\left(W_{L-1} \cdots W_2 g\left(W_1 \mathbf{x}\right)\right)\right) \in (0, 1)$$

- Optimization: stochastic gradient descent

$$\theta \leftarrow \theta - \gamma \frac{\partial \ell(\theta)}{\partial \theta}$$



Deep Learning Preliminary

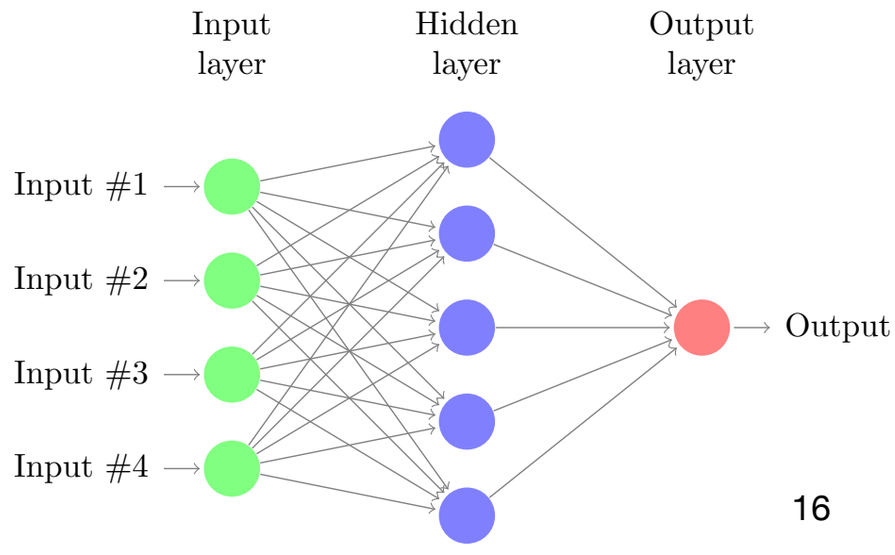
Let's consider a two-layer NN as illustration:

- Hidden layer $\mathbf{h} \in \mathbb{R}^p$ with output $\hat{y} \in \mathbb{R}$ for regression problem:

$$\hat{y} = \mathbf{a}^T \mathbf{h}, \quad \mathbf{h} = g(W\mathbf{x}), \quad W \in \mathbb{R}^{p \times d}$$

- Loss function:

$$\ell(W, \mathbf{a}) = \frac{1}{2}(\hat{y} - y)^2$$



Deep Learning Preliminary

Let's consider a two-layer NN as illustration:

- Hidden layer $\mathbf{h} \in \mathbb{R}^p$ with output $\hat{y} \in \mathbb{R}$ for regression problem:

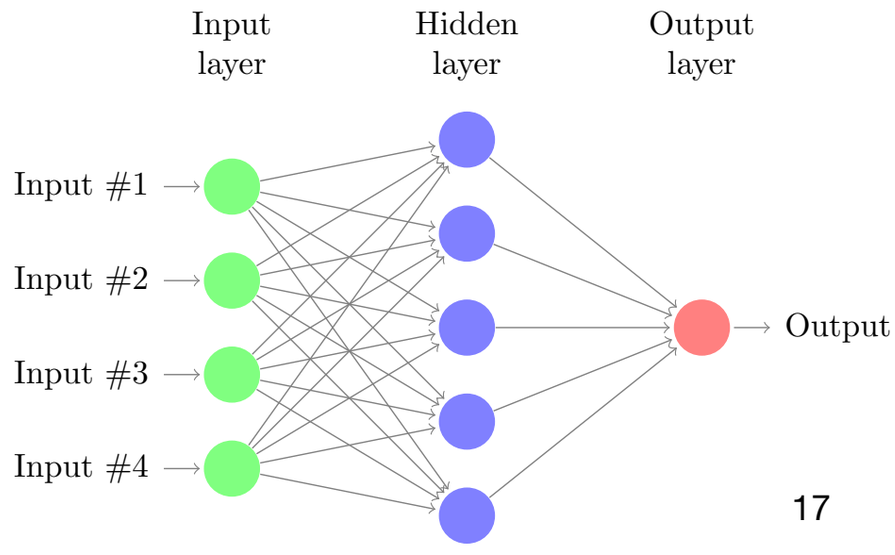
$$\hat{y} = \mathbf{a}^T \mathbf{h}, \quad \mathbf{h} = g(W\mathbf{x}), \quad W \in \mathbb{R}^{p \times d}$$

- Loss function:

$$\ell(W, \mathbf{a}) = \frac{1}{2}(\hat{y} - y)^2$$

- Consider the update formula for weight matrix W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$



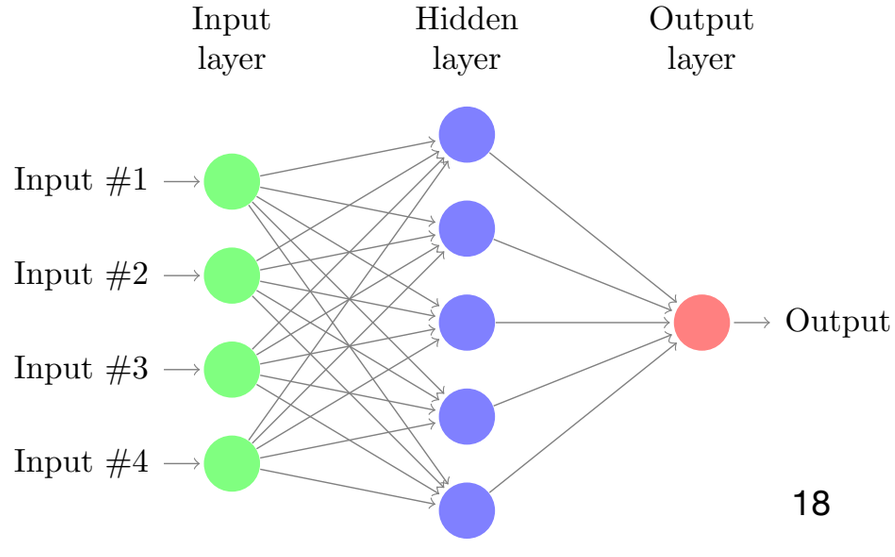
Deep Learning Preliminary

Let's consider a two-layer NN as illustration:

- Gradient update formula for weight matrix W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

- $\mathbf{h}'$: component-wise derivative of $\mathbf{h}$ w.r.t. its input argument



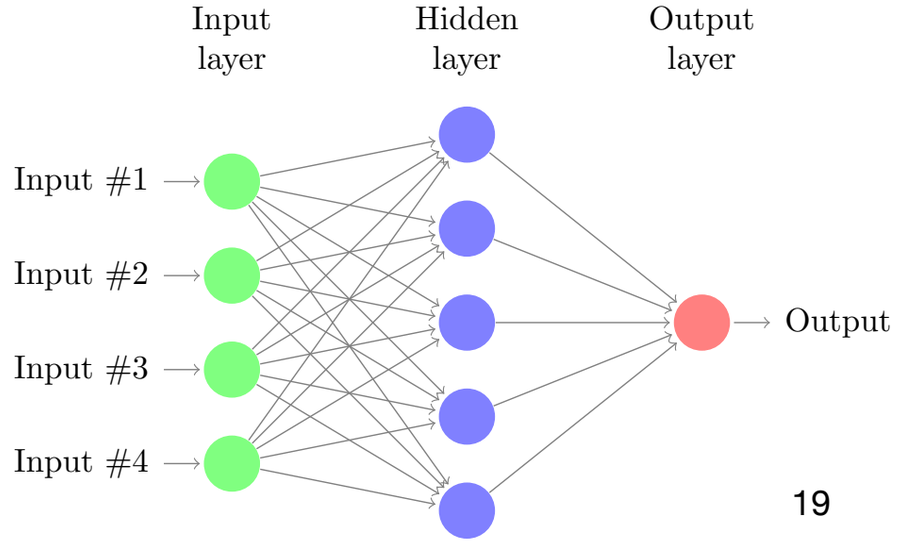
Deep Learning Preliminary

Let's consider a two-layer NN as illustration:

- Gradient update formula for weight matrix W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

- $\mathbf{h}'$: component-wise derivative of $\mathbf{h}$ w.r.t. its input argument
- The gradient w.r.t. W is always rank-1: rows/columns of W are correlated



Deep Learning Preliminary

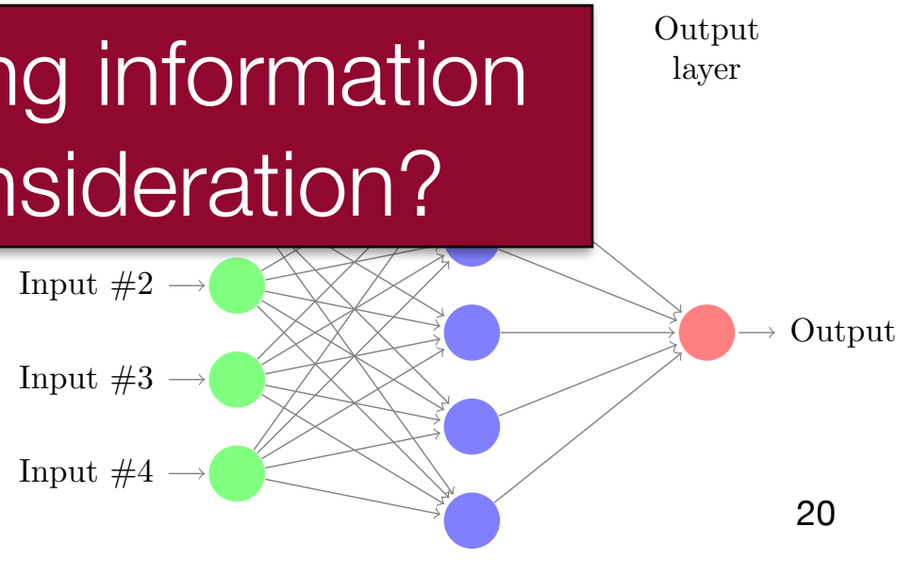
Let's consider a two-layer NN as illustration:

- Gradient update formula for weight matrix W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

- $\mathbf{h}'$: component-wise derivative of $\mathbf{h}$ w.r.t. its input argument
- The gradient w.r.t. W is always rank-1: rows/columns of W are correlated

Update each row/column by taking information from other rows/columns into consideration?



Outline

- Deep Learning Preliminary
- Learning with Adaptive Regularization
- Experiments
- Conclusion

Learning with Adaptive Regularization

Recall the gradient update formula for W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

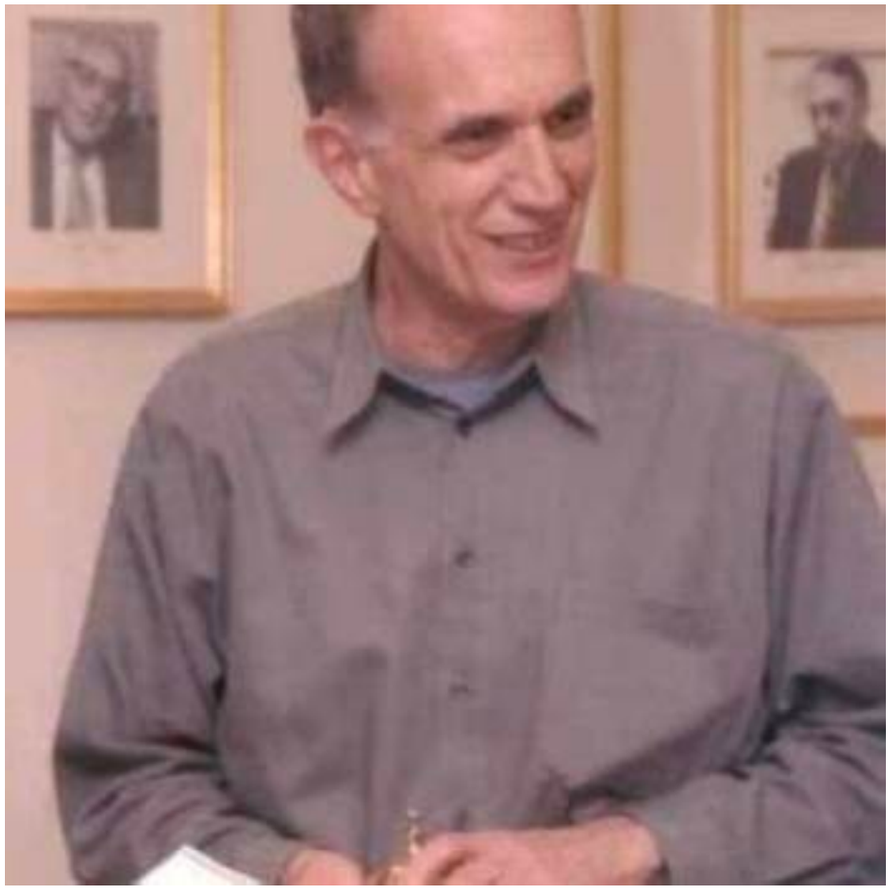
How can we make use of this structure?

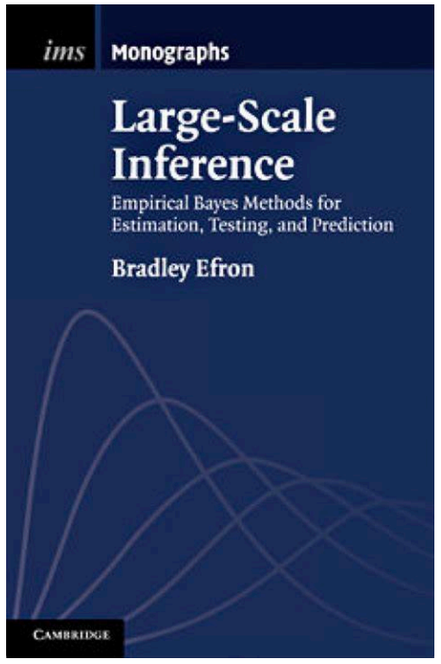
Learning with Adaptive Regularization

Recall the gradient update formula for W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

How can we make use of this structure?





"Learning from the experience of others"

— Prof. Bradley Efron

Learning with Adaptive Regularization

Recall the gradient update formula for W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

Now consider the following multivariate normal distribution:

$$W \sim \mathcal{MN}(0_{p \times d}, \Sigma_r, \Sigma_c)$$

- $\Sigma_r \in \mathbb{S}^p_{++},\ \Sigma_c \in \mathbb{S}^d_{++}$ are row and column covariance matrices

Learning with Adaptive Regularization

Recall the gradient update formula for W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

Now consider the following multivariate normal distribution:

$$W \sim \mathcal{MN}(0_{p\times d}, \Sigma_r, \Sigma_c)$$

- $\Sigma_r \in \mathbb{S}^p_{++},\ \Sigma_c \in \mathbb{S}^d_{++}$ are row and column covariance matrices
- Equivalently, this means

$$\text{vec}(W) \sim \mathcal{N}(0_{p\times d}, \Sigma_c \otimes \Sigma_r)$$

$$p(W \mid \Sigma_r, \Sigma_c) = \frac{\exp\left(-\text{Tr}(\Sigma_r^{-1} W \Sigma_c^{-1} W^T)/2\right)}{(2\pi)^{pd/2}\det(\Sigma_r)^{d/2}\det(\Sigma_c)^{p/2}}$$

Learning with Adaptive Regularization

Recall the gradient update formula for W:

$$W \leftarrow W - \gamma(\hat{y} - y)(\mathbf{a} \circ \mathbf{h}')\mathbf{x}^T$$

Now consider the following multivariate normal distribution:

$$W \sim \mathcal{MN}(0_{p\times d}, \Sigma_r, \Sigma_c)$$

- $\Sigma_r \in \mathbb{S}^p_{++},\ \Sigma_c \in \mathbb{S}^d_{++}$ are row and column covariance matrices
- Equivalently, this means

$$\text{vec}(W) \sim \mathcal{N}(0_{p\times d}, \Sigma_c \otimes \Sigma_r)$$

$$p(W \mid \Sigma_r, \Sigma_c) = \frac{\exp\left(-\text{Tr}(\Sigma_r^{-1} W \Sigma_c^{-1} W^T)/2\right)}{(2\pi)^{pd/2}\det(\Sigma_r)^{d/2}\det(\Sigma_c)^{p/2}}$$

- Or, $\quad W_{i:} \sim \mathcal{N}(\mathbf{0}_d, [\Sigma_r]_{ii} \cdot \Sigma_c), \quad W_{:j} \sim \mathcal{N}(\mathbf{0}_p, [\Sigma_c]_{jj} \cdot \Sigma_r)$

Learning with Adaptive Regularization

OK, but how to determine the parameters in the prior?

Learning with Adaptive Regularization

OK, but how to determine the parameters in the prior?

The empirical Bayes method: set the parameters of the prior from data by maximizing the marginal log-likelihood function (Efron and Morris, 1973; Efron and Hastie, 2016)

$$\widehat{\Sigma_r}, \widehat{\Sigma_c} = \underset{\Sigma_r, \Sigma_c}{\arg\max}\ p(\mathcal{D} \mid \Sigma_r, \Sigma_c) = \underset{\Sigma_r, \Sigma_c}{\arg\max} \int p(\mathcal{D} \mid W) \cdot p(W \mid \Sigma_r, \Sigma_c)\ dW$$

Learning with Adaptive Regularization

OK, but how to determine the parameters in the prior?

The empirical Bayes method: set the parameters of the prior from data by maximizing the marginal log-likelihood function (Efron and Morris, 1973; Efron and Hastie, 2016)

$$\widehat{\Sigma_r}, \widehat{\Sigma_c} = \arg\max_{\Sigma_r, \Sigma_c} \; p(\mathcal{D} \mid \Sigma_r, \Sigma_c) = \arg\max_{\Sigma_r, \Sigma_c} \int p(\mathcal{D} \mid W) \cdot p(W \mid \Sigma_r, \Sigma_c) \; dW$$

- High dimensional integration, intractable to compute, no closed form solution in general

Learning with Adaptive Regularization

OK, but how to determine the parameters in the prior?

The empirical Bayes method: set the parameters of the prior from data by maximizing the marginal log-likelihood function (Efron and Morris, 1973; Efron and Hastie, 2016)

$$\widehat{\Sigma_r}, \widehat{\Sigma_c} = \arg\max_{\Sigma_r, \Sigma_c} \; p(\mathcal{D} \mid \Sigma_r, \Sigma_c) = \arg\max_{\Sigma_r, \Sigma_c} \int p(\mathcal{D} \mid W) \cdot p(W \mid \Sigma_r, \Sigma_c) \; dW$$

- High dimensional integration, intractable to compute, no closed form solution in general
- Even with closed form solution, maximization is not necessarily tractable

Learning with Adaptive Regularization

OK, but how to determine the parameters in the prior?

The empirical Bayes method: set the parameters of the prior from data by maximizing the marginal log-likelihood function (Efron and Morris, 1973; Efron and Hastie, 2016)

$$\widehat{\Sigma_r}, \widehat{\Sigma_c} = \arg\max_{\Sigma_r, \Sigma_c} \; p(\mathcal{D} \mid \Sigma_r, \Sigma_c) = \arg\max_{\Sigma_r, \Sigma_c} \int p(\mathcal{D} \mid W) \cdot p(W \mid \Sigma_r, \Sigma_c) \; dW$$

- High dimensional integration, intractable to compute, no closed form solution in general
- Even with closed form solution, maximization is not necessarily tractable
- Example with closed-form solution: the James-Stein estimator

Learning with Adaptive Regularization

Iterative maximization of the joint distribution:

- Given weight matrix W, compute the most likely parameters Σ_r, Σ_c
- Given parameters Σ_r, Σ_c, plug in and update W

$$\max_{W} \max_{\Sigma_r, \Sigma_c} \log p(\mathcal{D} \mid W) + \log p(W \mid \Sigma_c, \Sigma_r)$$

Learning with Adaptive Regularization

Iterative maximization of the joint distribution:

- Given weight matrix W, compute the most likely parameters Σ_r, Σ_c
- Given parameters Σ_r, Σ_c, plug in and update W

$$\max_{W} \max_{\Sigma_r, \Sigma_c} \quad \log p(\mathcal{D} \mid W) + \log p(W \mid \Sigma_c, \Sigma_r)$$

Instantiate the above principle using the matrix-variate normal distribution, we arrive at the following optimization problem:

$$\min_{W, \mathbf{a}} \min_{\Omega_r, \Omega_c} \quad \frac{1}{2n} \sum_{i \in [n]} (\hat{y}(\mathbf{x}_i; W, \mathbf{a}) - y_i)^2 + \lambda ||\Omega_r^{1/2} W \Omega_c^{1/2}||_F^2 - \lambda \big(d \log\det(\Omega_r) + p \log\det(\Omega_c) \big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p, \; uI_d \preceq \Omega_c \preceq vI_d$$

Learning with Adaptive Regularization

Instantiate the above principle using the matrix-variate normal distribution, we arrive at the following optimization problem:

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- $\Omega_r := \Sigma_r^{-1},\ \Omega_c := \Sigma_c^{-1}$ are the precision matrices

Learning with Adaptive Regularization

Instantiate the above principle using the matrix-variate normal distribution, we arrive at the following optimization problem:

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2 + \lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2 - \lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- $\Omega_r := \Sigma_r^{-1},\ \Omega_c := \Sigma_c^{-1}$ are the precision matrices
- The first term corresponds to data fitting term

Learning with Adaptive Regularization

Instantiate the above principle using the matrix-variate normal distribution, we arrive at the following optimization problem:

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2 + \lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2 - \lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- $\Omega_r := \Sigma_r^{-1},\ \Omega_c := \Sigma_c^{-1}$ are the precision matrices
- The first term corresponds to data fitting term
- The second term corresponds to regularization term

Learning with Adaptive Regularization

Instantiate the above principle using the matrix-variate normal distribution, we arrive at the following optimization problem:

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- $\Omega_r := \Sigma_r^{-1},\ \Omega_c := \Sigma_c^{-1}$ are the precision matrices
- The first term corresponds to data fitting term
- The second term corresponds to regularization term
- λ is a trade-off hyper-parameter that only depends on p, d

Learning with Adaptive Regularization

Maximum-A-Posteriori estimation

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\Big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\Big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

The optimization formulation defines a sequence of MAP problems:

Learning with Adaptive Regularization

Maximum-A-Posteriori estimation

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\Big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\Big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

The optimization formulation defines a sequence of MAP problems:

- At iteration t, with $(\Omega_r^{(t)},\Omega_c^{(t)})$, the second term becomes

$$||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2=||\text{vec}(\Omega_r^{1/2}W\Omega_c^{1/2})||_2^2=||(\Omega_c^{1/2}\otimes\Omega_r^{1/2})\ \text{vec}(W)||_2^2$$

Learning with Adaptive Regularization

Maximum-A-Posteriori estimation

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\Big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\Big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

The optimization formulation defines a sequence of MAP problems:

- At iteration t, with $(\Omega_r^{(t)},\Omega_c^{(t)})$, the second term becomes

$$||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2=||\mathrm{vec}(\Omega_r^{1/2}W\Omega_c^{1/2})||_2^2=||(\Omega_c^{1/2}\otimes\Omega_r^{1/2})\ \mathrm{vec}(W)||_2^2$$

- Tikhonov regularization imposed on W with the Tikhonov matrix as $\Gamma:=\Omega_c^{1/2}\otimes\Omega_r^{1/2}$

Learning with Adaptive Regularization

Maximum-A-Posteriori estimation

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\Big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\Big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

The optimization formulation defines a sequence of MAP problems:

- At iteration t, with $(\Omega_r^{(t)},\Omega_c^{(t)})$, the second term becomes

$$||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2=||\mathrm{vec}(\Omega_r^{1/2}W\Omega_c^{1/2})||_2^2=||(\Omega_c^{1/2}\otimes\Omega_r^{1/2})\ \mathrm{vec}(W)||_2^2$$

- Tikhonov regularization imposed on W with the Tikhonov matrix as $\Gamma := \Omega_c^{1/2}\otimes\Omega_r^{1/2}$
- The Tikhonov matrix is not predefined, but learned from data adaptively

Learning with Adaptive Regularization

Volume minimization

$\log\det(\cdot)$ is a concave function over the positive-definite cone. For any pair of matrices $A_1, A_2 \in \mathbb{S}^m_{++}$, the following inequality holds:

$$\begin{aligned}\log\det(A_1) &\leq \log\det(A_2) + \langle \nabla \log\det(A_2), A_1 - A_2 \rangle \\ &= \log\det(A_2) + \mathrm{Tr}(A_2^{-1} A_1) - m\end{aligned}$$

Learning with Adaptive Regularization

Volume minimization

$\log\det(\cdot)$ is a concave function over the positive-definite cone. For any pair of matrices $A_1, A_2 \in \mathbb{S}^m_{++}$, the following inequality holds:

$$\begin{aligned}\log\det(A_1) &\leq \log\det(A_2) + \langle \nabla \log\det(A_2), A_1 - A_2 \rangle \\ &= \log\det(A_2) + \mathrm{Tr}(A_2^{-1} A_1) - m\end{aligned}$$

Apply the above inequality twice, we obtain:

$$d\log\det(W\Omega_c W^T/2d) \leq -d\log\det(\Omega_r) + \frac{1}{2}\mathrm{Tr}(\Omega_r W \Omega_c W^T) - dp,$$

$$p\log\det(W^T\Omega_r W/2p) \leq -p\log\det(\Omega_c) + \frac{1}{2}\mathrm{Tr}(\Omega_r W \Omega_c W^T) - dp.$$

Learning with Adaptive Regularization

Volume minimization

Apply the above inequality twice, we obtain:

$$d \log\det(W\Omega_c W^T/2d) \leq -d\log\det(\Omega_r) + \frac{1}{2}\operatorname{Tr}(\Omega_r W \Omega_c W^T) - dp,$$

$$p \log\det(W^T\Omega_r W/2p) \leq -p\log\det(\Omega_c) + \frac{1}{2}\operatorname{Tr}(\Omega_r W \Omega_c W^T) - dp.$$

which implies:

$$d\log\det(W\Omega_c W^T) + p\log\det(W^T\Omega_r W) \leq ||\Omega_r^{1/2} W \Omega_c^{1/2}||_F^2 - \big(d\log\det(\Omega_r) + p\log\det(\Omega_c)\big) + c$$

Learning with Adaptive Regularization

Volume minimization

Apply the above inequality twice, we obtain:

$$d \log\det(W\Omega_c W^T/2d) \leq -d\log\det(\Omega_r) + \frac{1}{2}\operatorname{Tr}(\Omega_r W \Omega_c W^T) - dp,$$

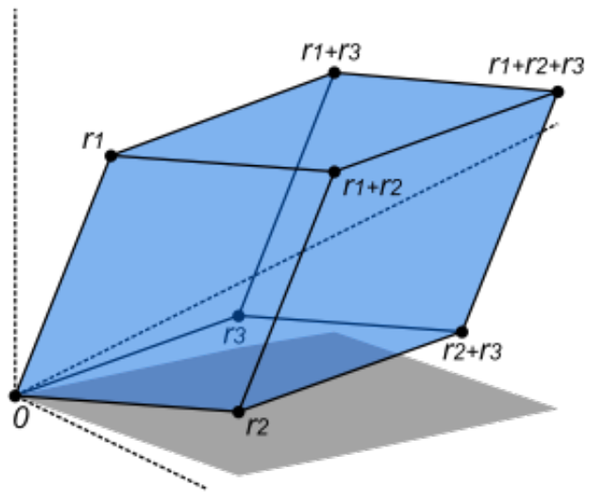
$$p \log\det(W^T\Omega_r W/2p) \leq -p\log\det(\Omega_c) + \frac{1}{2}\operatorname{Tr}(\Omega_r W \Omega_c W^T) - dp.$$

which implies:

$$d\log\det(W\Omega_c W^T) + p\log\det(W^T\Omega_r W) \leq ||\Omega_r^{1/2} W \Omega_c^{1/2}||_F^2 - \big(d\log\det(\Omega_r) + p\log\det(\Omega_c)\big) + c$$

Squared volume of the parallelepiped spanned by row/column vectors of preconditioned W

Our regularizer



Learning with Adaptive Regularization

How to optimize (Ω_r, Ω_c) ?

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n} \sum_{i\in[n]} (\hat{y}(\mathbf{x}_i; W, \mathbf{a}) - y_i)^2 + \lambda ||\Omega_r^{1/2} W \Omega_c^{1/2}||_F^2 - \lambda \big(d \log\det(\Omega_r) + p \log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

Learning with Adaptive Regularization

How to optimize (Ω_r, Ω_c) ?

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- For fixed W and Ω_c, the objective function is convex in Ω_r

Learning with Adaptive Regularization

How to optimize (Ω_r, Ω_c) ?

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- For fixed W and Ω_c, the objective function is convex in Ω_r
- For fixed W and Ω_r, the objective function is convex in Ω_c

Learning with Adaptive Regularization

How to optimize (Ω_r, Ω_c) ?

$$\min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad \frac{1}{2n}\sum_{i\in[n]}(\hat{y}(\mathbf{x}_i;W,\mathbf{a})-y_i)^2+\lambda||\Omega_r^{1/2}W\Omega_c^{1/2}||_F^2-\lambda\big(d\log\det(\Omega_r)+p\log\det(\Omega_c)\big)$$

$$\text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p,\ uI_d \preceq \Omega_c \preceq vI_d$$

- For fixed W and Ω_c, the objective function is convex in Ω_r
- For fixed W and Ω_r, the objective function is convex in Ω_c
- The objective function is symmetric w.r.t. Ω_r and Ω_c

It suffices if we focus on and solve one of them

Learning with Adaptive Regularization

For fixed W and Ω_c, consider the optimization over Ω_r:

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r), \qquad \text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p$$

Define the constraint set: $\mathcal{C} := \{A \in \mathbb{S}^p_{++} \mid uI_p \preceq A \preceq vI_p\}$ and indicator function

$$\mathbb{I}_{\mathcal{C}}(A) = \begin{cases} 0 & \text{If } A \in \mathcal{C} \\ \infty & \text{Otherwise} \end{cases}$$

Learning with Adaptive Regularization

For fixed W and Ω_c, consider the optimization over Ω_r:

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r), \qquad \text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p$$

Define the constraint set: $\mathcal{C} := \{A \in \mathbb{S}^p_{++} \mid uI_p \preceq A \preceq vI_p\}$ and indicator function

$$\mathbb{I}_{\mathcal{C}}(A) = \begin{cases} 0 & \text{If } A \in \mathcal{C} \\ \infty & \text{Otherwise} \end{cases}$$

Now we have an equivalent unconstrained convex optimization problem:

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r)$$

Learning with Adaptive Regularization

For fixed W and Ω_c, consider the optimization over Ω_r:

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r), \qquad \text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p$$

Define the constraint set: $\mathcal{C} := \{A \in \mathbb{S}^p_{++} \mid uI_p \preceq A \preceq vI_p\}$ and indicator function

$$\mathbb{I}_{\mathcal{C}}(A) = \begin{cases} 0 & \text{If } A \in \mathcal{C} \\ \infty & \text{Otherwise} \end{cases}$$

Now we have an equivalent unconstrained convex optimization problem:

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r)$$

The first-order optimality condition tells us:

$$0 \in \partial \left(\frac{1}{d} \mathrm{Tr}(\Omega_r W \Omega_c W^T) - \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r) \right) = W\Omega_c W^T/d - \Omega_r^{-1} + \mathcal{N}_{\mathcal{C}}(\Omega_r)$$

Learning with Adaptive Regularization

For fixed W and Ω_c, consider the optimization over Ω_r:

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r), \qquad \text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p$$

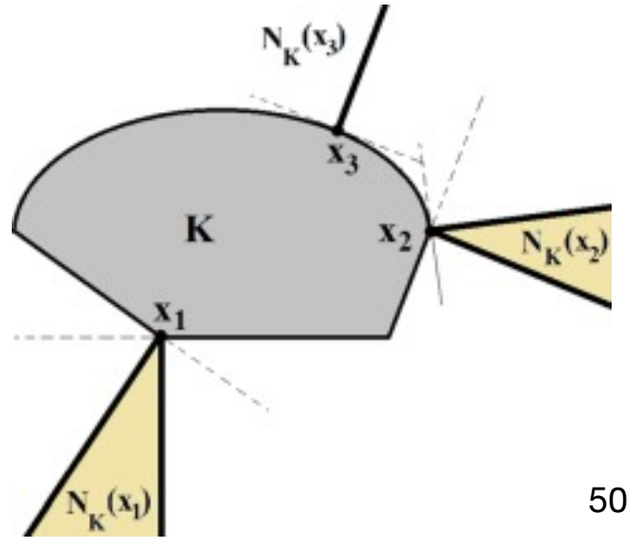
The first-order optimality condition tells us:

$$0 \in \partial \left(\frac{1}{d} \mathrm{Tr}(\Omega_r W \Omega_c W^T) - \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r) \right) = W\Omega_c W^T/d - \Omega_r^{-1} + \mathcal{N}_{\mathcal{C}}(\Omega_r)$$

where

$$\mathcal{N}_{\mathcal{C}}(A) := \{B \in \mathbb{S}^p \mid \mathrm{Tr}(B^T(Z - A)) \leq 0, \forall Z \in \mathcal{C}\}$$

is the normal cone w.r.t. the constraint set $\mathcal{C}$ at A.



Learning with Adaptive Regularization

The first-order optimality condition tells us:

$$0 \in \partial \left(\frac{1}{d} \operatorname{Tr}(\Omega_r W \Omega_c W^T) - \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r) \right) = W\Omega_c W^T/d - \Omega_r^{-1} + \mathcal{N}_{\mathcal{C}}(\Omega_r)$$

Lemma: Let $\Omega_r \in \mathcal{C}$, then $\mathcal{N}_{\mathcal{C}}(\Omega_r) = -\mathcal{N}_{\mathcal{C}}(\Omega_r^{-1})$.

Learning with Adaptive Regularization

The first-order optimality condition tells us:

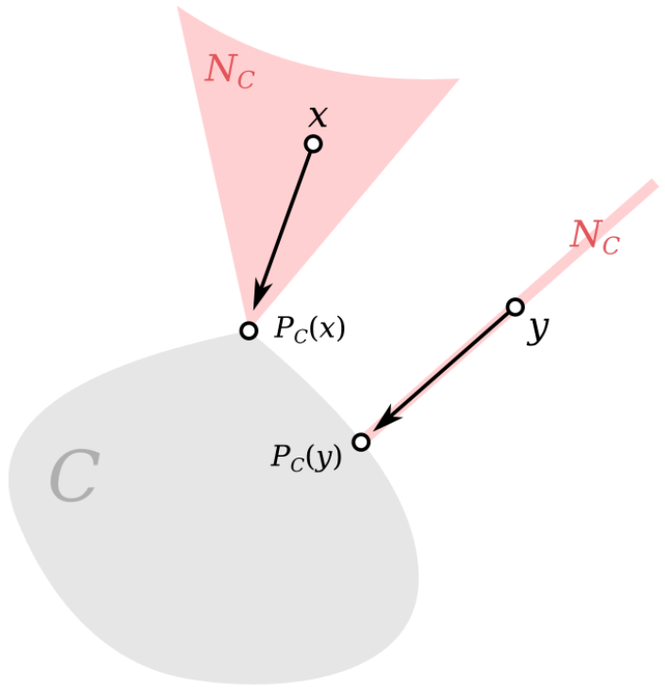
$$0 \in \partial \left(\frac{1}{d} \operatorname{Tr}(\Omega_r W \Omega_c W^T) - \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r) \right) = W\Omega_c W^T/d - \Omega_r^{-1} + \mathcal{N}_{\mathcal{C}}(\Omega_r)$$

Lemma: Let $\Omega_r \in \mathcal{C}$, then $\mathcal{N}_{\mathcal{C}}(\Omega_r) = -\mathcal{N}_{\mathcal{C}}(\Omega_r^{-1})$.

With this lemma, we have:

$$W\Omega_c W^T/d - \Omega_r^{-1} \in \mathcal{N}_{\mathcal{C}}(\Omega_r^{-1})$$

which means the optimal Ω_r^{-1} is the Euclidean projection of $W\Omega_c W^T/d$ onto $\mathcal{C}$.



Learning with Adaptive Regularization

The first-order optimality condition tells us:

$$0 \in \partial \left(\frac{1}{d} \operatorname{Tr}(\Omega_r W \Omega_c W^T) - \log\det(\Omega_r) + \mathbb{I}_{\mathcal{C}}(\Omega_r) \right) = W\Omega_c W^T/d - \Omega_r^{-1} + \mathcal{N}_{\mathcal{C}}(\Omega_r)$$

Lemma: Let $\Omega_r \in \mathcal{C}$, then $\mathcal{N}_{\mathcal{C}}(\Omega_r) = -\mathcal{N}_{\mathcal{C}}(\Omega_r^{-1})$.

With this lemma, we have:

$$W\Omega_c W^T/d - \Omega_r^{-1} \in \mathcal{N}_{\mathcal{C}}(\Omega_r^{-1})$$

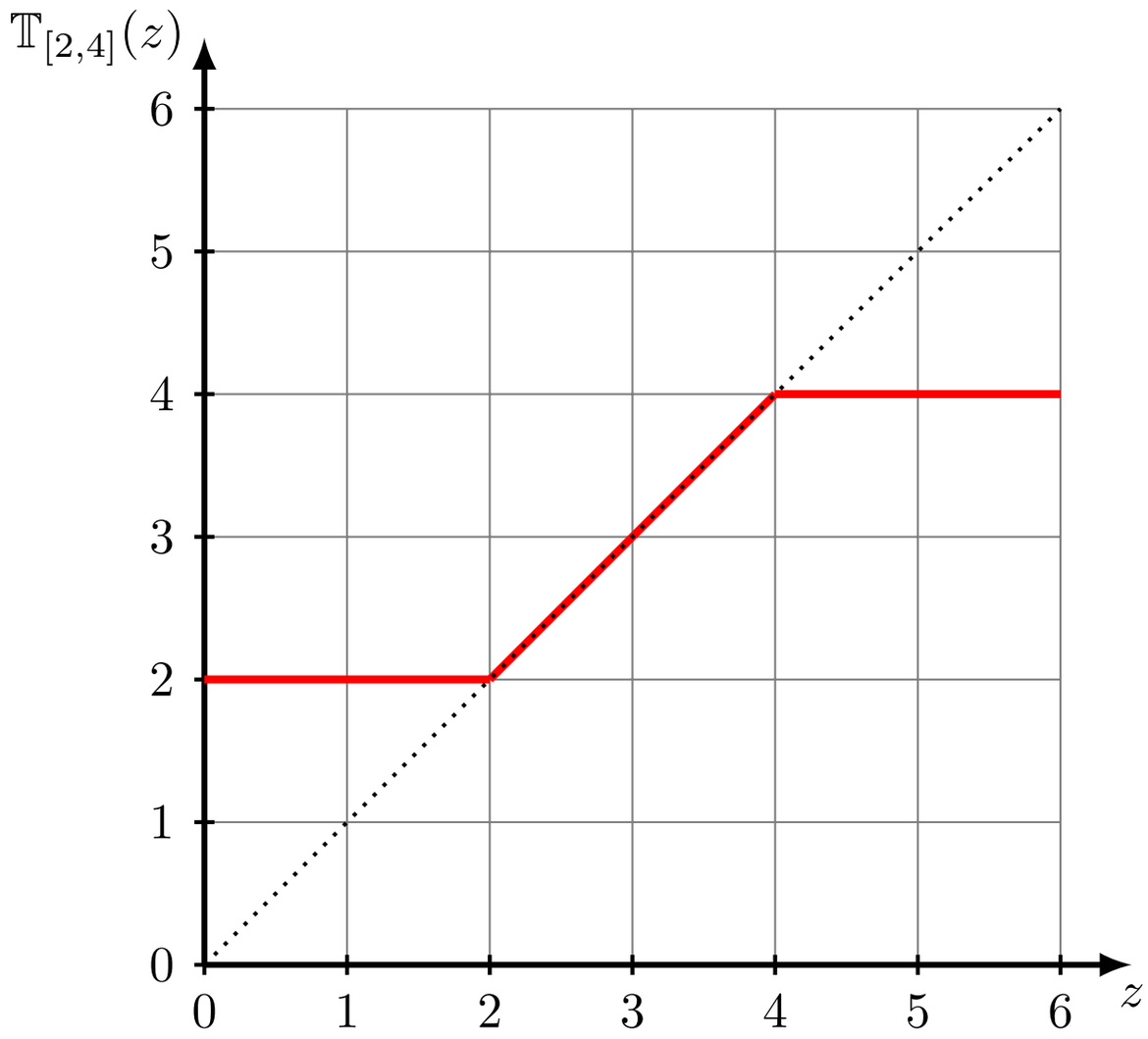
which means the optimal Ω_r^{-1} is the Euclidean projection of $W\Omega_c W^T/d$ onto $\mathcal{C}$. Hence it suffices if we can solve the following problem:

$$\min_{\Omega_r} \quad ||\Omega_r - \widetilde{\Omega_r}||_F^2, \qquad \text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p$$

Learning with Adaptive Regularization

Given a pair of constants $0 < u \leq v$, define the following thresholding operator:

$$\mathbb{T}_{[u,v]}(x) := \max\{u, \min\{v, x\}\}$$



Learning with Adaptive Regularization

Given a pair of constants $0 < u \leq v$, define the following thresholding operator:

$$\mathbb{T}_{[u,v]}(x) := \max\{u, \min\{v, x\}\}$$

The Euclidean projection onto $\mathcal{C}$ has the following closed form solution:

Theorem: Let $\widetilde{\Omega_r} \in \mathbb{S}^p$ with eigendecomposition as $\widetilde{\Omega_r} = Q\Lambda Q^T$. Then $\text{Proj}_{\mathcal{C}}(\widetilde{\Omega_r}) = Q\mathbb{T}_{[u,v]}(\Lambda)Q^T$.

Simple: Apply eigendecomposition and then threshold the spectrum

Learning with Adaptive Regularization

Given a pair of constants $0 < u \leq v$, define the following thresholding operator:

$$\mathbb{T}_{[u,v]}(x) := \max\{u, \min\{v, x\}\}$$

The Euclidean projection onto $\mathcal{C}$ has the following closed form solution:

Theorem: Let $\widetilde{\Omega_r} \in \mathbb{S}^p$ with eigendecomposition as $\widetilde{\Omega_r} = Q\Lambda Q^T$. Then $\mathrm{Proj}_{\mathcal{C}}(\widetilde{\Omega_r}) = Q\mathbb{T}_{[u,v]}(\Lambda)Q^T$.

Simple: Apply eigendecomposition and then threshold the spectrum

$$\min_{\Omega_r} \quad \mathrm{Tr}(\Omega_r W \Omega_c W^T) - d \log\det(\Omega_r), \qquad \text{subject to} \quad uI_p \preceq \Omega_r \preceq vI_p$$

Solution: $$\Omega_r^* = Q\mathbb{T}_{[u,v]}(d/\mathbf{r})Q^T$$

where $Q\mathrm{diag}(\mathbf{r})Q^T = W\Omega_c W^T$ is the spectral decomposition of $W\Omega_c W^T$

Learning with Adaptive Regularization

Block Coordinate Descent for Adaptive Regularization:

Algorithm 1 Block Coordinate Descent for Adaptive Regularization

Input: Initial value $\phi^{(0)} := \{\mathbf{a}^{(0)}, W^{(0)}\}$, $\Omega_r^{(0)} \in \mathbb{S}_{++}^p$ and $\Omega_c^{(0)} \in \mathbb{S}_{++}^d$, first-order optimization algorithm $\mathfrak{A}$.

for $t = 1, \ldots, \infty$ until convergence **do**
  Fix $\Omega_r^{(t-1)}$, $\Omega_c^{(t-1)}$, optimize $\phi^{(t)}$ by backpropagation and algorithm $\mathfrak{A}$
  $\Omega_r^{(t)} \leftarrow \textsc{InvThreshold}(W^{(t)}\Omega_c^{(t-1)}W^{(t)T}, d, u, v)$
  $\Omega_c^{(t)} \leftarrow \textsc{InvThreshold}(W^{(t)T}\Omega_r^{(t)}W^{(t)}, p, u, v)$
end for
procedure $\textsc{InvThreshold}(\Delta, m, u, v)$
  Compute SVD: $Q\mathrm{diag}(\mathbf{r})Q^T = \mathrm{SVD}(\Delta)$
  Hard thresholding $\mathbf{r}' \leftarrow \mathbb{I}_{[u,v]}(m/\mathbf{r})$
  return $Q\mathrm{diag}(\mathbf{r}')Q^T$
end procedure

- One SVD in each outer iteration to solve the matrix optimization problem
- Seamlessly work with end-to-end training with back-propagation

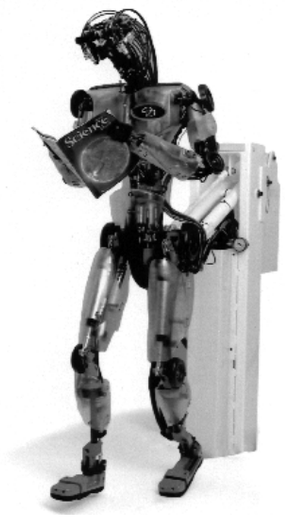
Outline

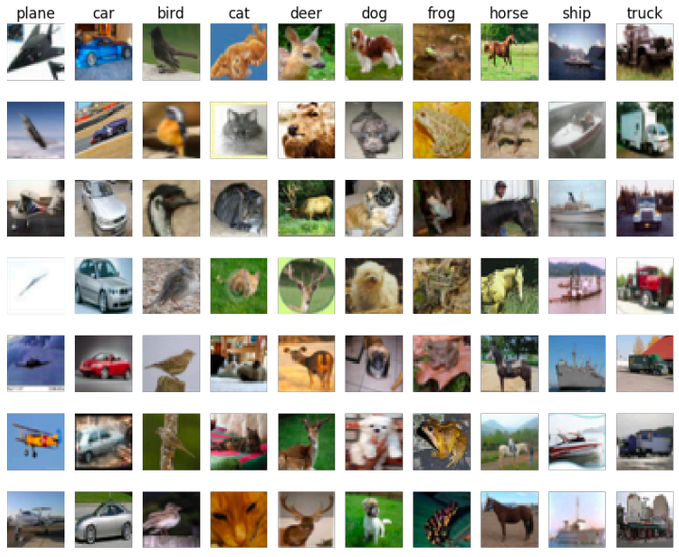
- Deep Learning Preliminary
- Learning with Adaptive Regularization
- Experiments
- Conclusion

Experiments

Multiclass Classification and Multitask Regression

- Multiclass Classification: MNIST and CIFAR10
  - MNIST: $\text{CONV}_{5\times5\times1\times10} - \text{CONV}_{5\times5\times10\times20} - \text{FC}_{320\times50} - \text{FC}_{50\times10}$
  - CIFAR10: $\text{CONV}_{5\times5\times3\times10} - \text{CONV}_{5\times5\times10\times20} - \text{FC}_{500\times500} - \text{FC}_{500\times500} - \text{FC}_{500\times10}$
- Multitask Regression: SARCOS
  - SARCOS: $\text{FC}_{21\times256} - \text{FC}_{256\times100} - \text{FC}_{100\times7}$





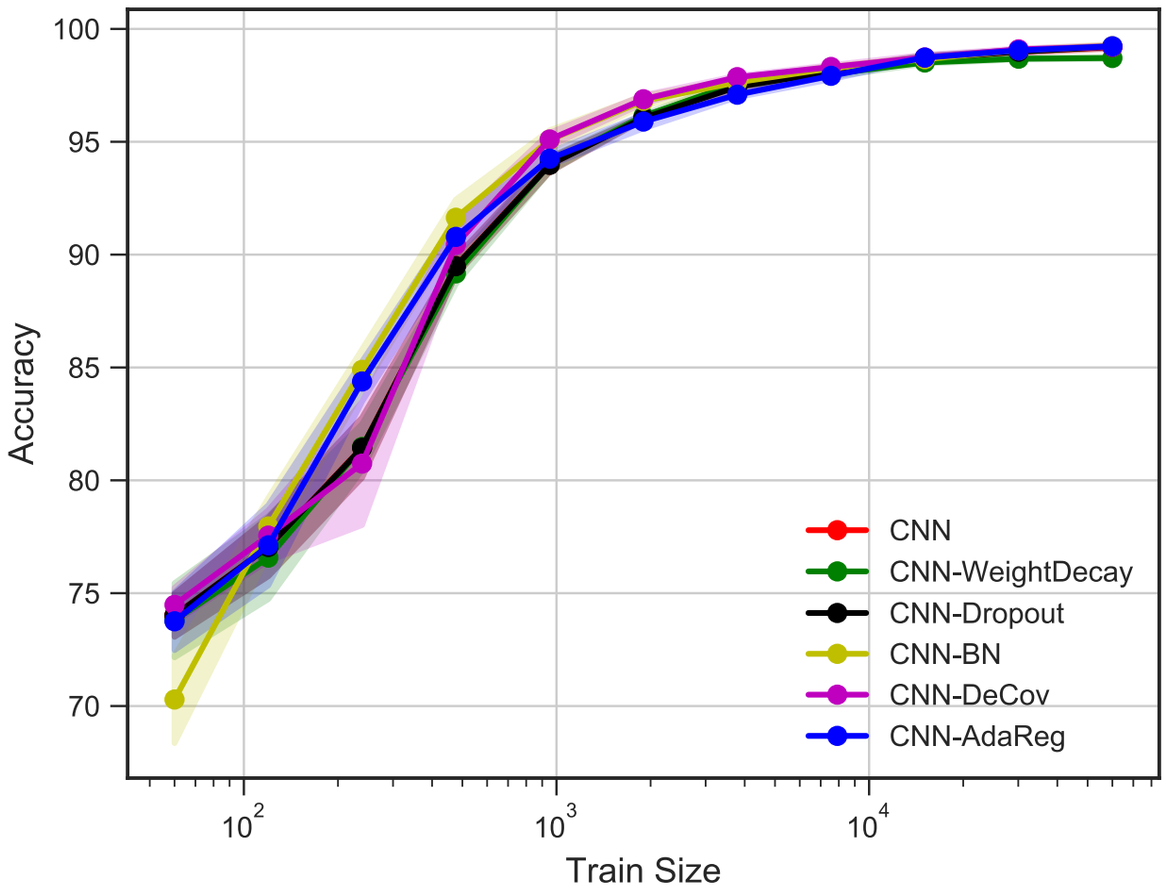
Experiments

Multiclass Classification and Multitask Regression

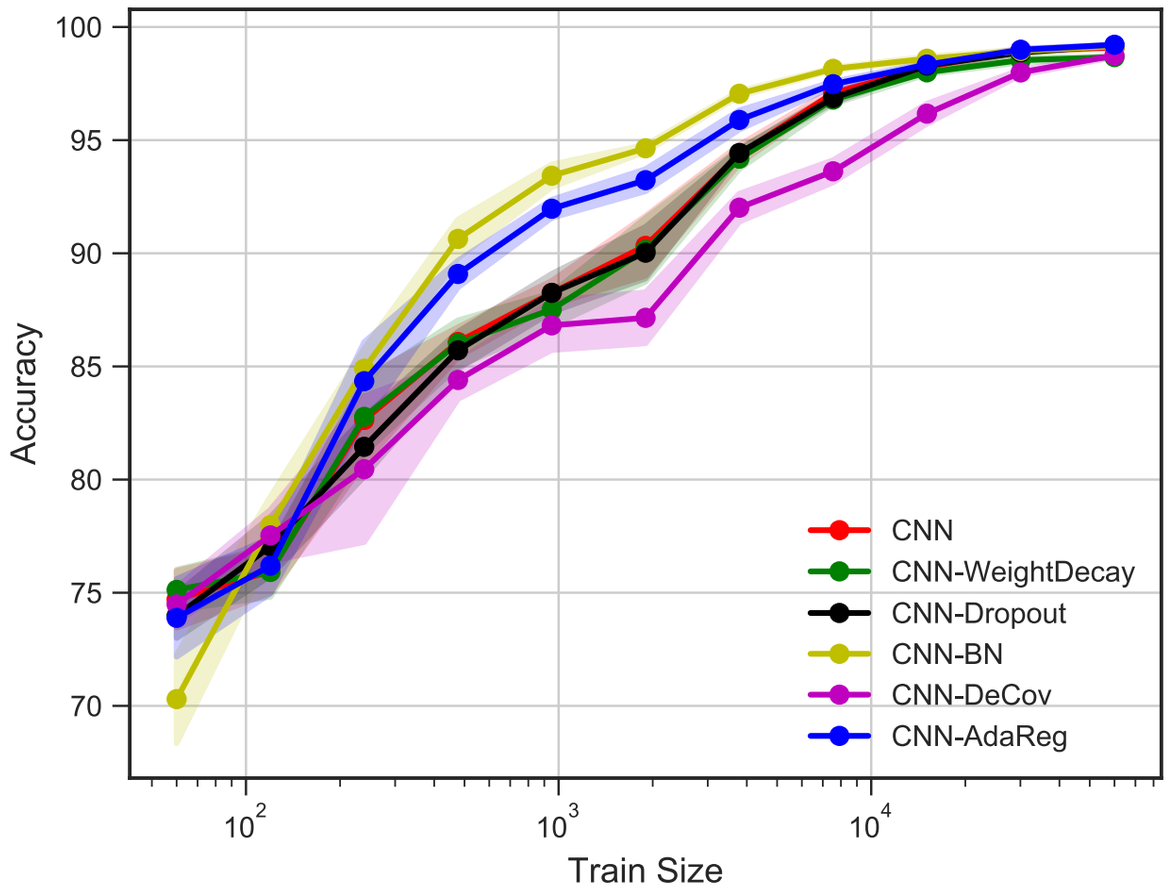
- Dropout: randomly inhibiting hidden activations
- Batch Normalization (BN): maintain and correct internal covariate shift
- Weight Decay: L2 regularization of weight matrix
- DeCov: penalize covariance matrix of hidden representations

Experiments

Generalization performance on MNIST (10 classes)



Batch size = 256

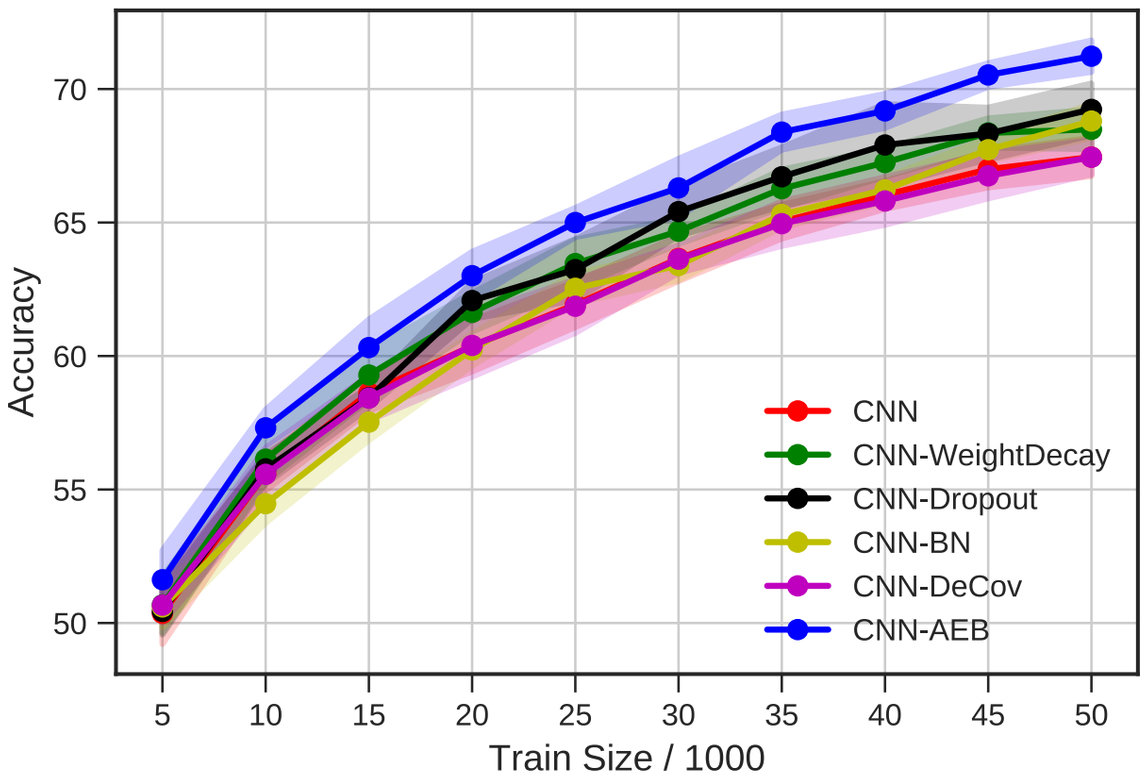


Batch size = 2048

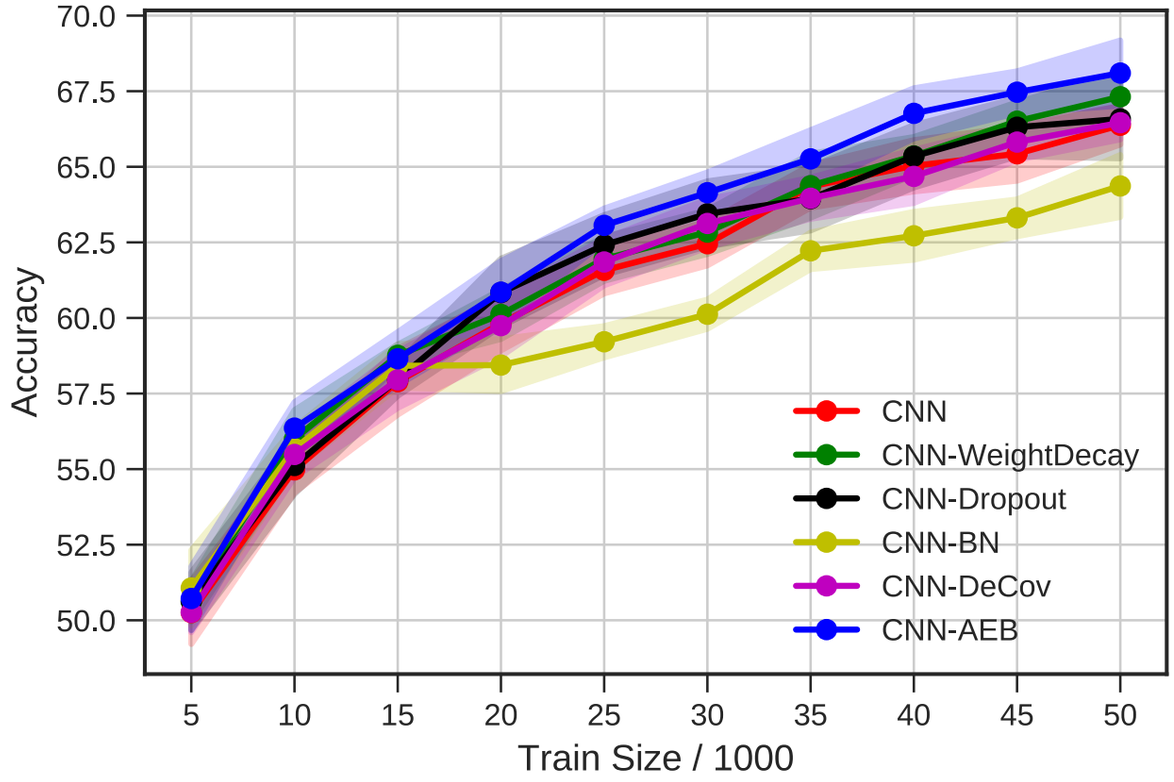
- AdaReg improves generalization with large batch size and small training size on MNIST

Experiments

Generalization performance on CIFAR10 (10 classes)



Batch size = 256

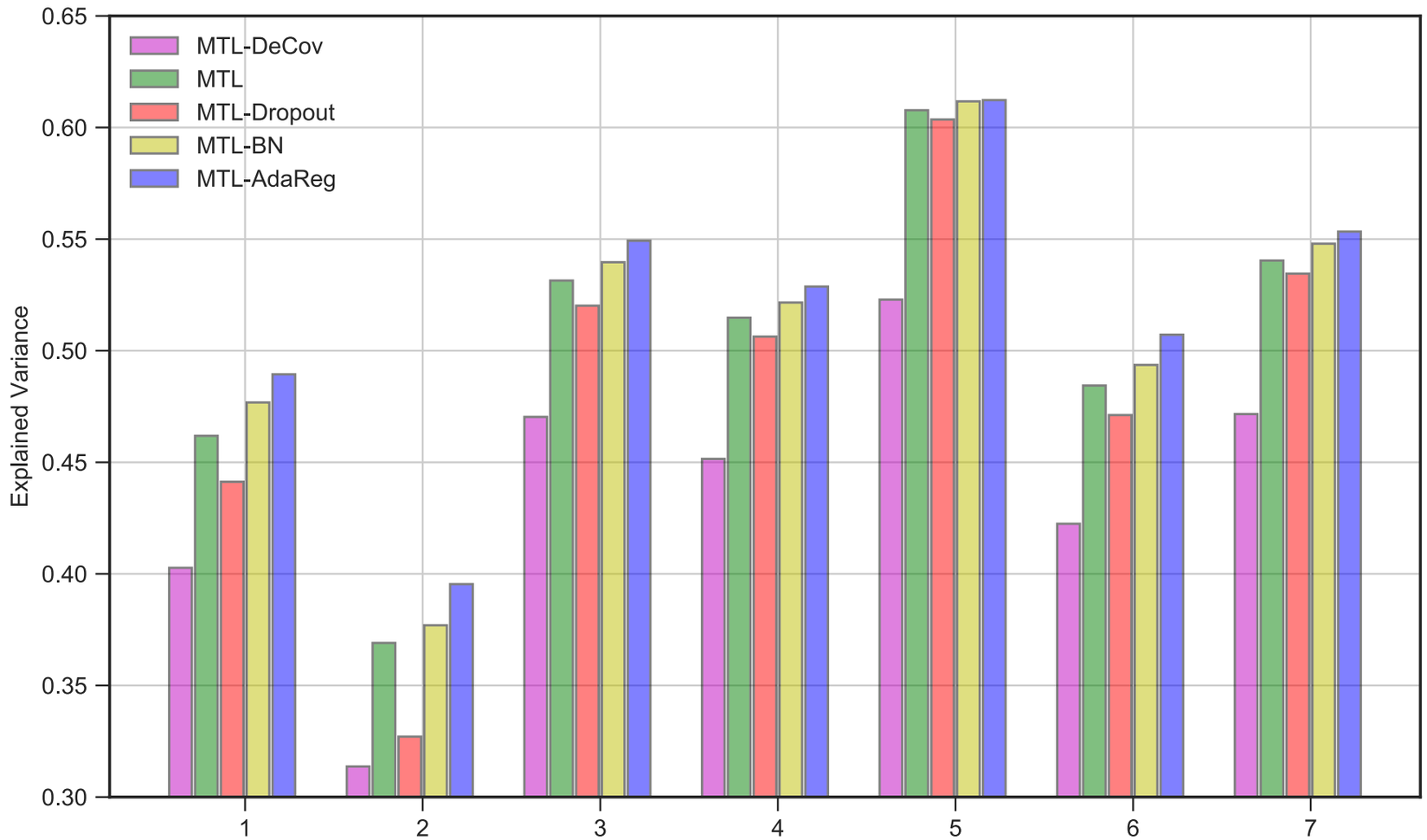


Batch size = 2048

- AdaReg consistently improves generalization on CIFAR10

Experiments

Generalization performance on SARCOS (7 tasks)

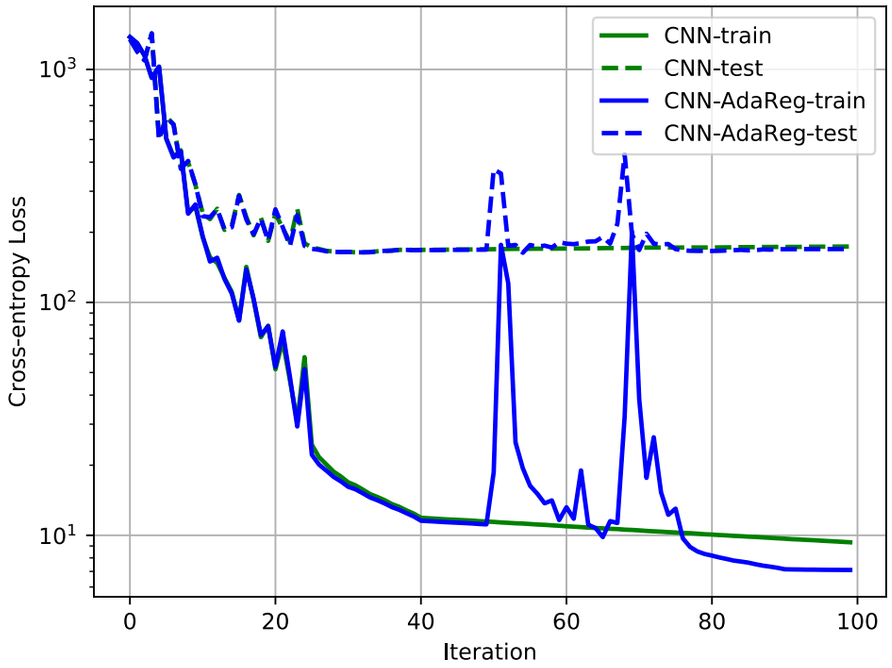


- Explained variance ~ negative MSE
- For regression tasks, AdaReg consistently outperforms other baselines in terms of generalization

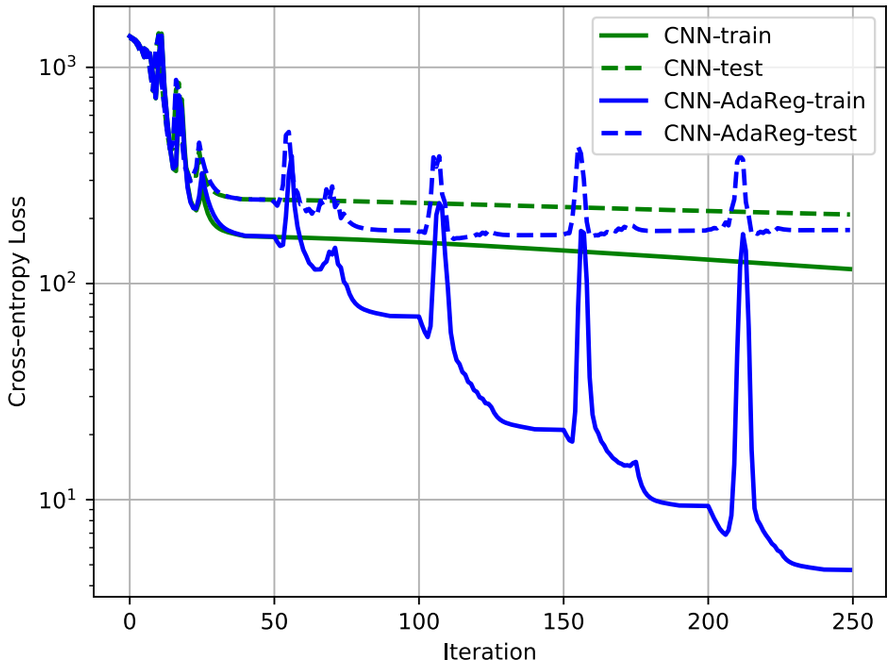
Experiments

AdaReg optimization trajectory on MNIST (train size/batch size)

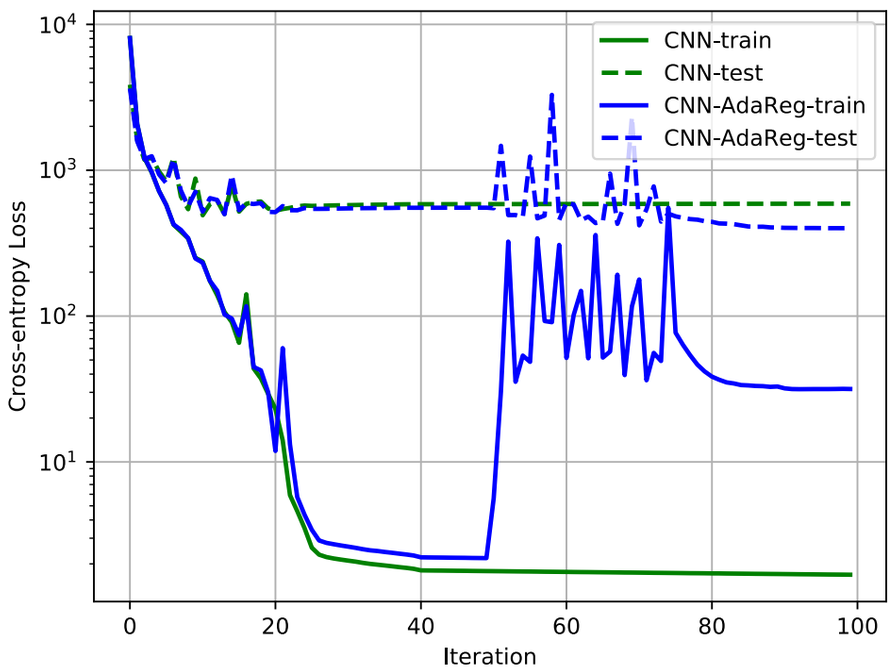
600/256



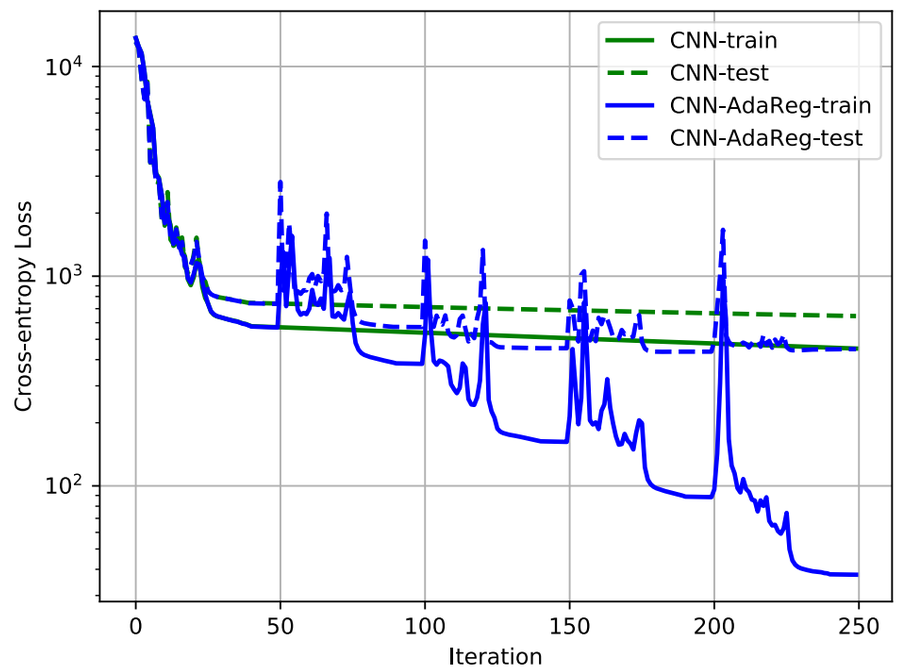
600/2048



6000/256



6000/2048



Carnegie M

Experiments

Stable Rank and Spectral Norm:

$$\text{srank}(W) := ||W||_F^2 / ||W||_2^2 \qquad 1 \leq \text{srank}(W) \leq \text{rank}(W)$$

Generalization bound of NNs using spectral norms and stable ranks (Neyshabur et al. ICLR' 17):

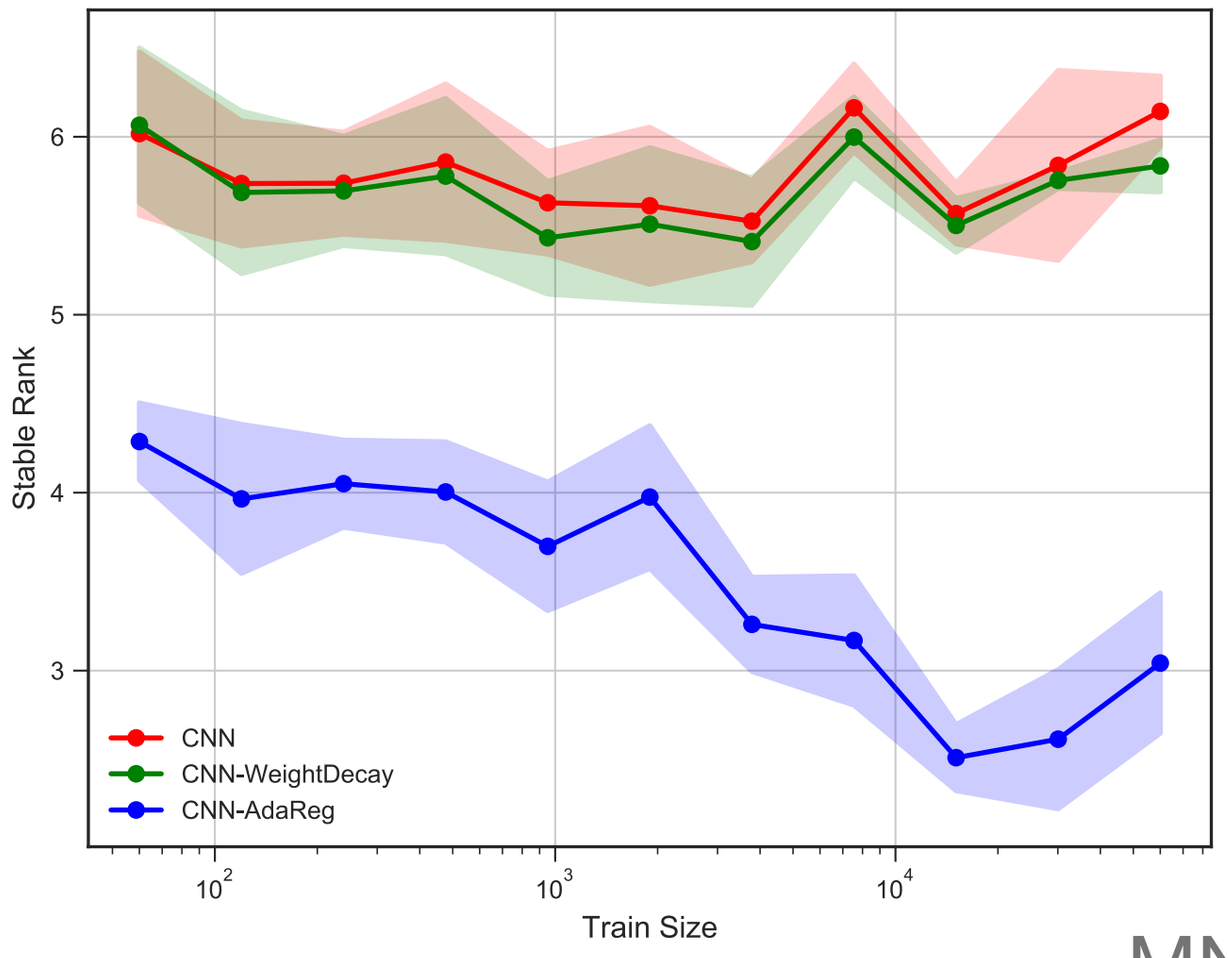
$$\text{Generalization Error} = O\left(\sqrt{\prod_{j=1}^{L} ||W_j||_2^2 \sum_{j=1}^{L} \text{srank}(W_j)/n}\right)$$

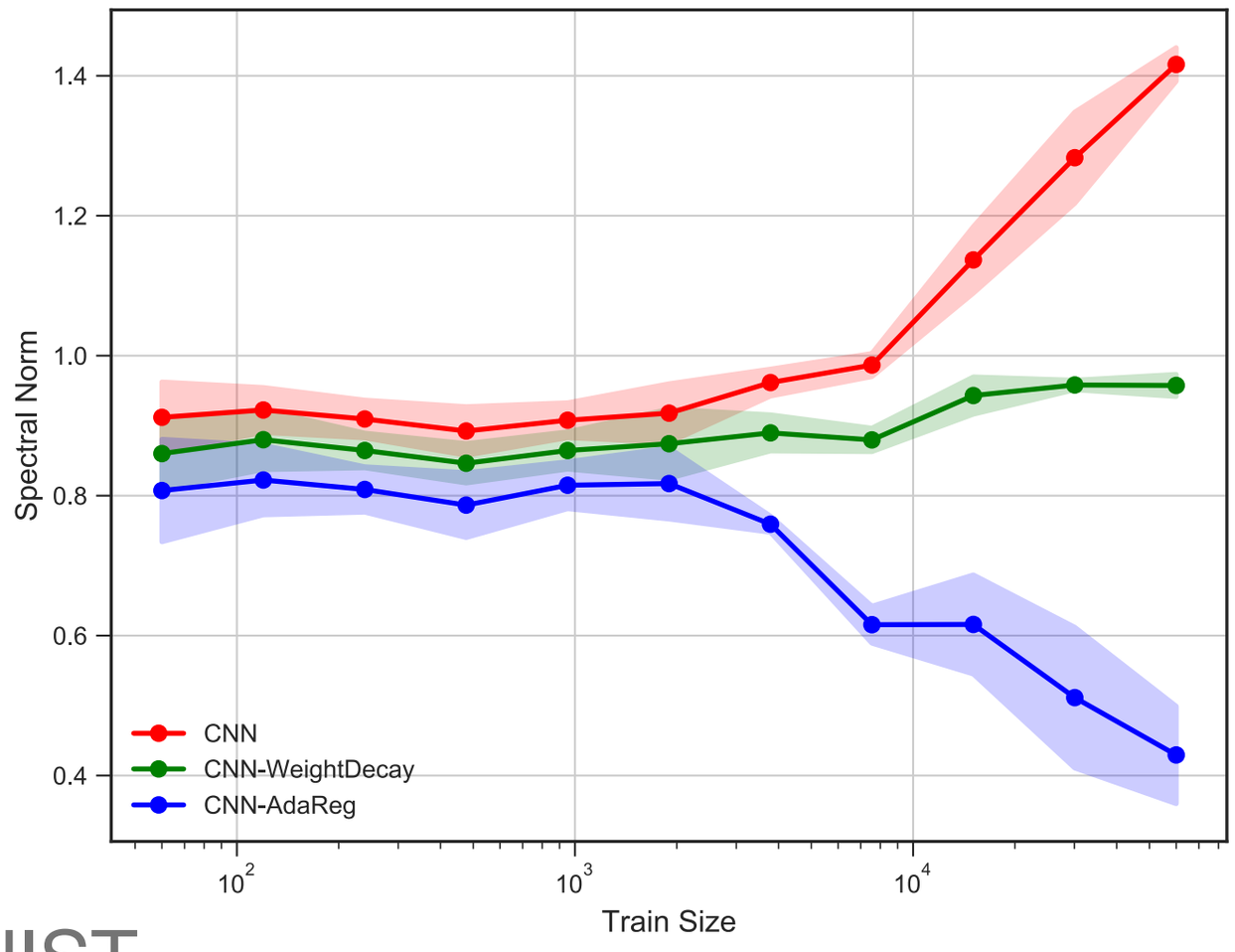
- Smaller stable rank and spectral norm implies better generalization

Experiments

AdaReg leads to smaller stable ranks and spectral norms

$$\text{Generalization Error} = O\left(\sqrt{\prod_{j=1}^{L} ||W_j||_2^2 \sum_{j=1}^{L} \text{srank}(W_j)/n}\right)$$



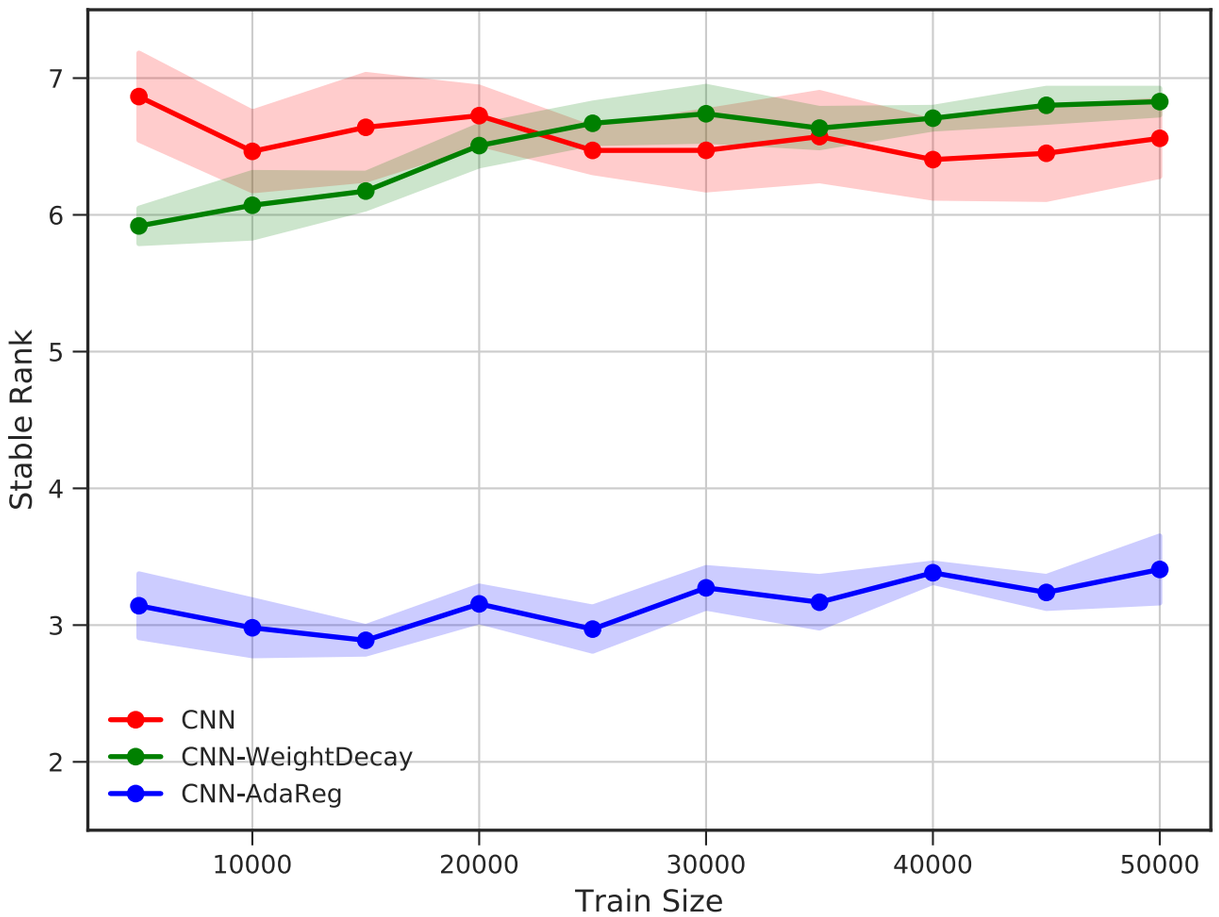


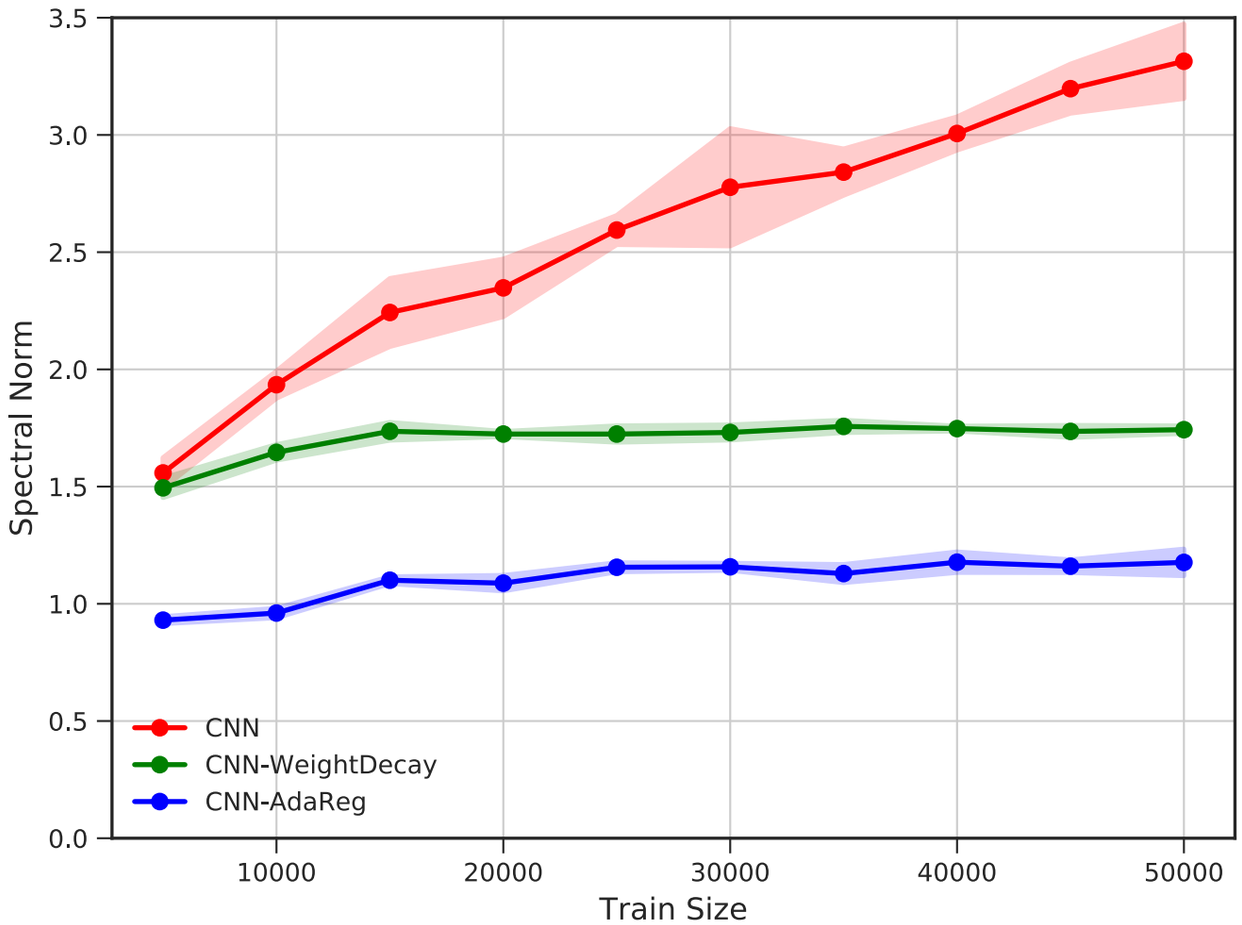
MNIST

Experiments

AdaReg leads to smaller stable ranks and spectral norms

$$\text{Generalization Error} = O\left(\sqrt{\prod_{j=1}^{L} ||W_j||_2^2 \sum_{j=1}^{L} \text{srank}(W_j)/n}\right)$$

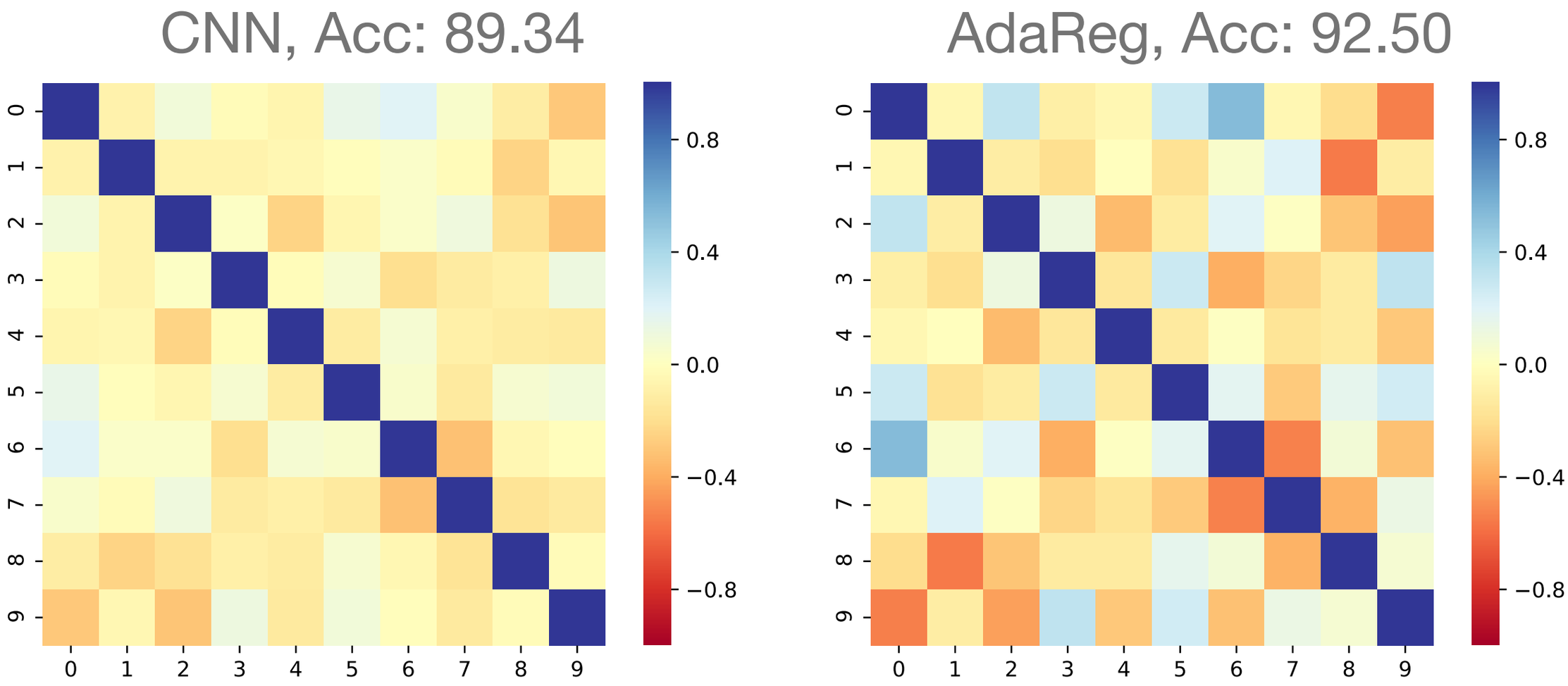




CIFAR10

Experiments

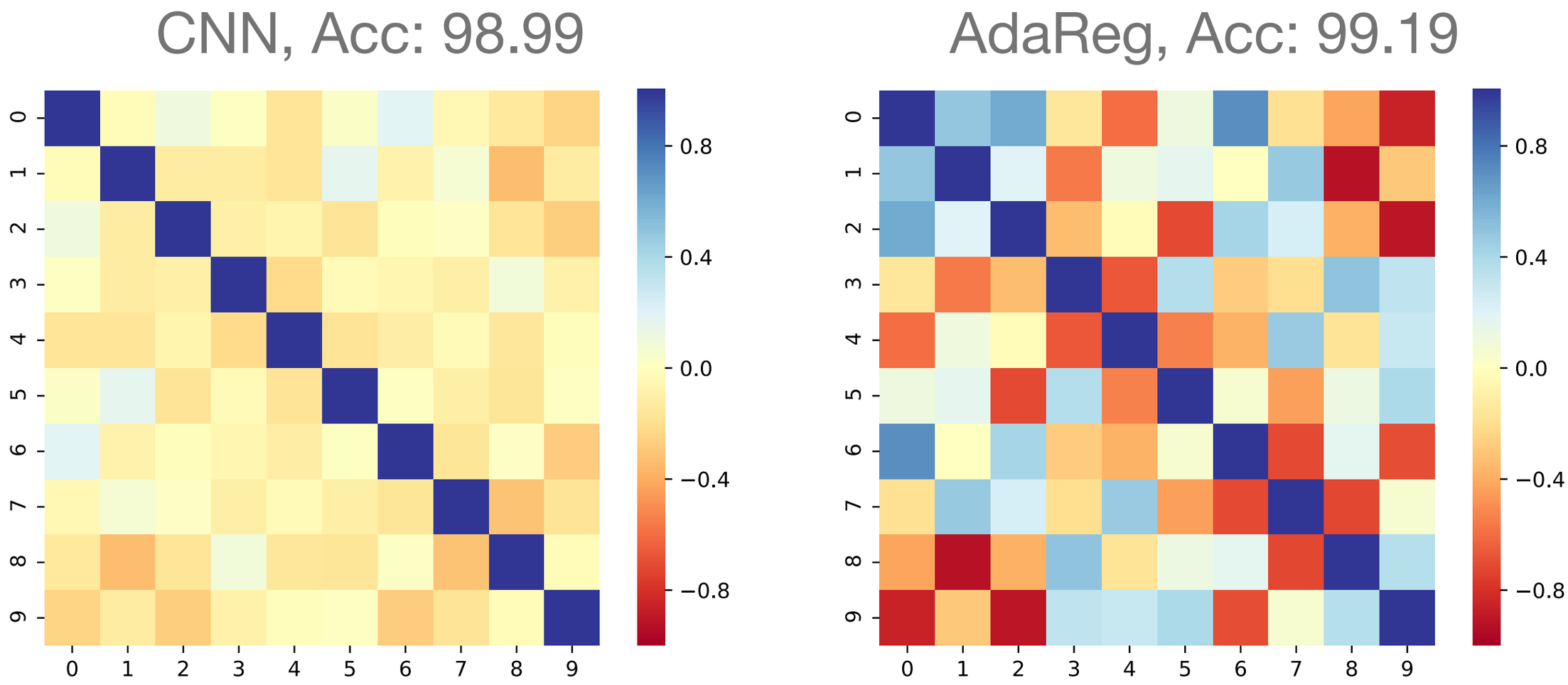
Correlations learned in the weight matrix:



- The effect of sharing statistical strength, i.e., “learning from the experience of others”

Experiments

Correlations learned in the weight matrix:



MNIST, train size = 6,000

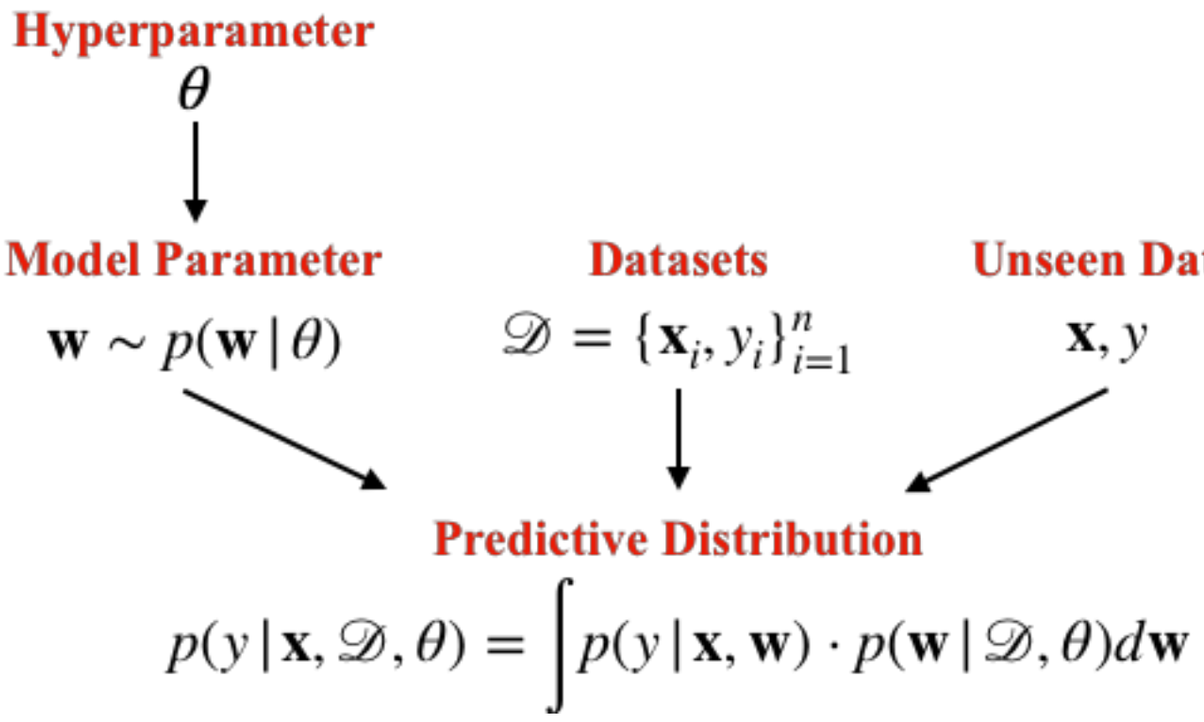
- The effect of sharing statistical strength, i.e., “learning from the experience of others”

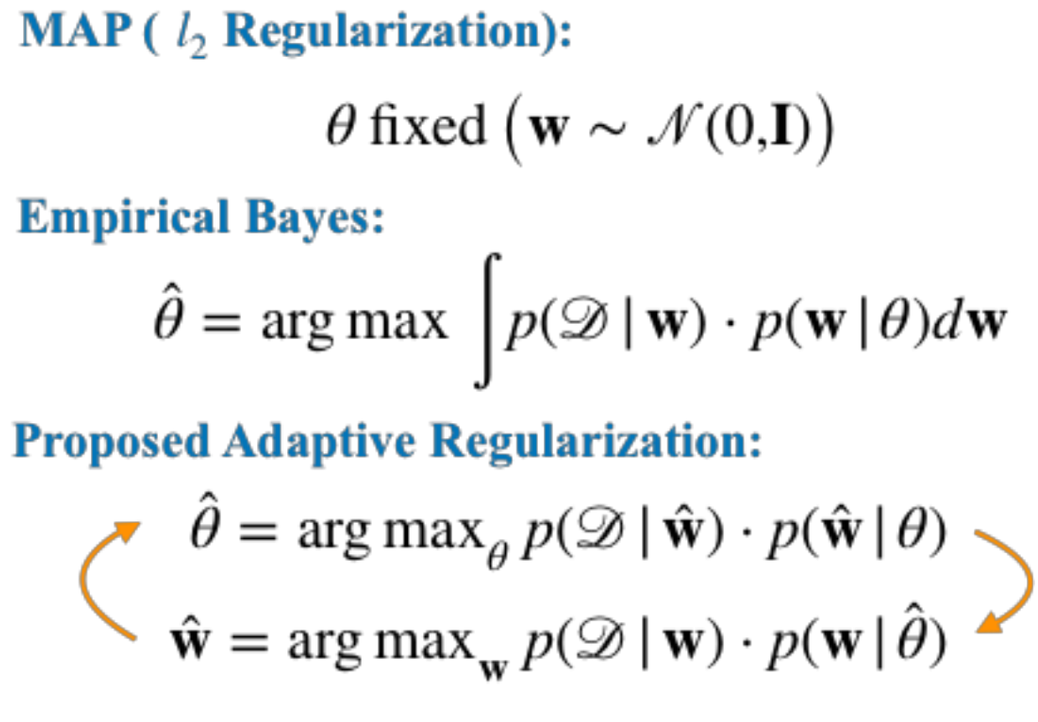
Outline

- Deep Learning Preliminary
- Learning with Adaptive Regularization
- Experiments
- Conclusion

Conclusion

- AdaReg: an approximate empirical Bayes method for regularizing NN training on small datasets





- Learn the preconditioning matrix adaptively from data

- Significant improvement in terms of spectral norm and stable rank, leading to smaller generalization error

Thanks

Q & A

Paper available at: https://arxiv.org/abs/1907.06288
To appear @ NeurIPS-19, code available @ https://github.com/yaohungt/Adaptive-Regularization-Neural-Network