Approximate Empirical Bayes for Deep Neural Networks

Han Zhao*, Yao-Hung Hubert Tsai*, Ruslan Salakhutdinov and Geoff Gordon
Machine Learning Department, Carnegie Mellon University

Summary

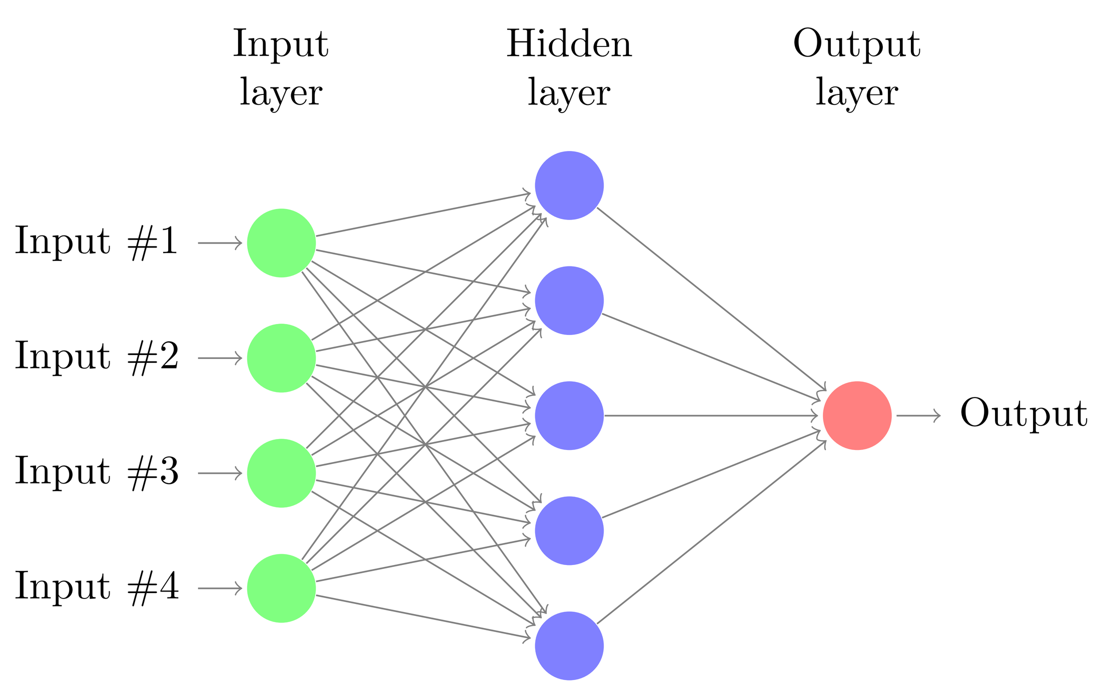
Learning Parameters in Neural Networks:

- Maximum likelihood estimation: rich neural networks overfit on small datasets.
- Regularizations: Weight decay, early stopping, dropout, DeCov, spectral regularization, etc.

Contributions:

- We propose an approximate empirical Bayes (AEB) framework for learning neural networks.
- We give a block coordinate ascent algorithm to optimize the weight matrix.

Motivating Example: A simple two-layer feed-forward neural network:



Forward propagation: $\hat{y} = \mathbf{a}^T\mathbf{h}, \quad \mathbf{h} = \sigma(W\mathbf{x})$.

- Input $\mathbf{x}$, hidden layer $\mathbf{h}$, a single output $y \in \mathbb{R}$.
- Weights: W and $\mathbf{a}$.
- ReLU activation $\sigma(\cdot)$, ℓ_2 loss as objective function.

Backward propagation:

- $W \leftarrow W - \alpha(\hat{y} - y)(\mathbf{a} \odot \mathbf{h}')\mathbf{x}^T$.
- The gradient matrix is rank one.
- Rows/Columns of the updates are correlated with each other.

It is beneficial to learn from the experience of others.

– Bradley Efron

The Model and Algorithm

Approximate Empirical Bayes:

Introduce a matrix-variate normal distribution over the weight matrix:

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

- $\Sigma_r \in \mathbb{S}^p_{++}$: Covariance matrix over row vectors.
- $\Sigma_c \in \mathbb{S}^d_{++}$: Covariance matrix over column vectors.
- $\mathcal{MN}(A, \Sigma_r, \Sigma_c)$ is a matrix-variate normal with mean $\mathrm{vec}(A)$ and covariance $\Sigma_r \otimes \Sigma_c$.

The true empirical Bayes approach is intractable for neural networks, hence we approximate it with the following optimization formulation:

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

Plugging in the prior distribution leads to a constrained optimization problem:

$$\begin{aligned} \min_{W,\mathbf{a}} \min_{\Omega_r,\Omega_c} \quad & \frac{1}{2}|\hat{y}(W,\mathbf{a}) - y|^2 + \lambda||\Omega_r^{1/2} W \Omega_c^{1/2}||_F^2 \\ & - \lambda\left(d \log\det(\Omega_r) + p\log\det(\Omega_c)\right) \\ \text{subject to} \quad & uI_p \preceq \Omega_r \preceq vI_p, \; uI_d \preceq \Omega_c \preceq vI_d \end{aligned}$$

- $\Omega_r := \Sigma_c^{-1}$ and $\Omega_c := \Sigma_c^{-1}$ are the corresponding precision matrices.

Block Coordinate Ascent Algorithm:

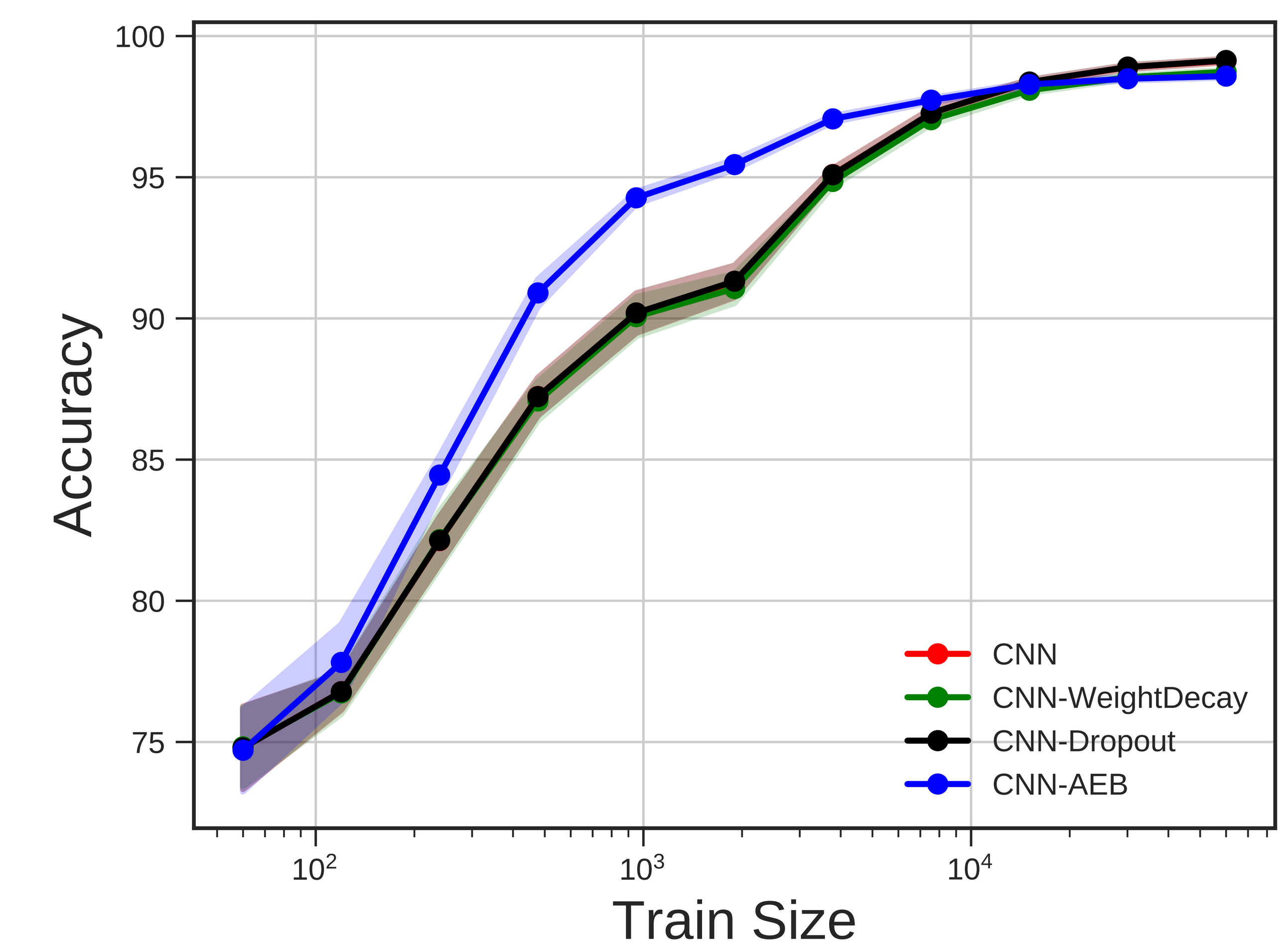
Input: Initial value $\mathbf{w}^0 := \{\mathbf{a}^{(0)}, W^{(0)}\}$, $\Omega_r^{(0)}$ and $\Omega_c^{(0)}$, first-order optimization algorithm $\mathfrak{A}$, constants $0 < u \leq v$.

```
for t = 1, ..., ∞ until convergence do
    Fix Ω_r^(t-1), Ω_c^(t-1), optimize w^(t) by backpropagation and algorithm 𝔄
    Ω_r^(t) ← InvThresholding(W^(t) Ω_c^(t-1) W^(t)T, d, u, v)
    Ω_c^(t) ← InvThresholding(W^(t)T Ω_r^(t) W^(t), p, u, v)
end for
-----------------------------------------------------------
procedure InvThresholding(Δ, m, u, v)
    Compute SVD: Q diag(r) Q^T = SVD(Δ)
    Threshold r' ← 𝕋_[u,v](m/r)
    return Q diag(r') Q^T
end procedure
```

Experiments

Multi-class classification: MNIST dataset.

Network structure: $\mathsf{CONV}_{5\times5\times1\times10}$-$\mathsf{CONV}_{5\times5\times10\times20}$-$\mathsf{FC}_{320\times50}$-$\mathsf{FC}_{50\times10}$. Only impose prior distribution on the weight matrix of the last layer.



Multi-task regression: SARCOS dataset.

Goal: Map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. The training set and test set contain 44,484 and 4,449 examples. Network structure: $\mathsf{FC}_{21\times256}$-$\mathsf{FC}_{256\times100}$-$\mathsf{FC}_{100\times7}$.