Bayesian neural networks for sparse coding

Danil Kuzin [1] Olga Isupova [2] Lyudmila Mihaylova [1]

[1] Department of Automatic Control and Systems Engineering, University of Sheffield, UK

[2] Machine Learning Research Group, University of Oxford, UK

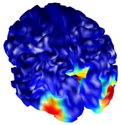
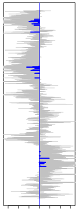
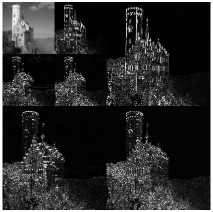
ICASSP 2019

Problem

Sparse regression

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$
$$\mathbf{y} \in \mathbb{R}^K,\ \boldsymbol{\beta} \in \mathbb{R}^D,\ K < D$$

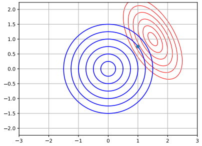
How to find sparse β given $\mathbf{y}$, $\mathbf{X}$?

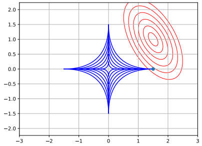
Sparse frequentist regression

Add sparsity-inducing penalty

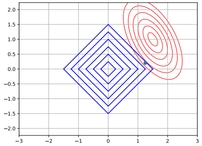
$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}} \left[||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + ||\boldsymbol{\beta}||_p^p \right]$$



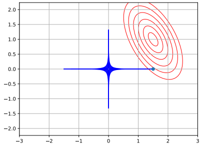
l_2-penalty



$l_{1/2}$-penalty



l_1-penalty

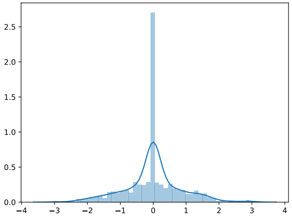


$l_{1/8}$-penalty

Sparse Bayesian regression

Add sparsity-inducing prior

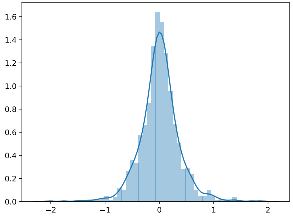
Strong sparsity



$$\beta_d \sim (1 - z_d)\mathcal{N}(0, \sigma^2) + z_d\delta_0$$
$$z_d \sim \text{Ber}(\omega)$$

- probability of exact zero
- discrete variables

Weak sparsity



$$\beta_d \sim \mathcal{N}(0, \sigma_d^2)$$
$$\sigma_d^2 \sim \text{IG}(a)$$

- continuous at zero
- continuous variables

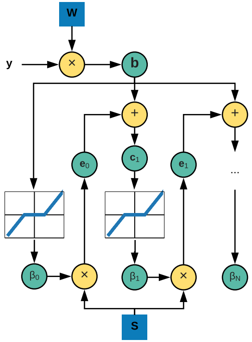
Problem

Build a model that estimates $\boldsymbol{\beta}$ from observations $\mathbf{y}$ such that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, and elements of $\boldsymbol{\beta}$ contain zeros.

Iterative Shrinkage-Thresholding Algorithm (ISTA)

Soft thresholding

$$h_\lambda(\mathbf{b}) = \mathrm{sgn}(\mathbf{b}) \max(|\mathbf{b}| - \lambda, 0),$$



Prediction

```
Require: observation $\mathbf{y}$
1: Define. weights $\mathbf{W} = \mathbf{X}^\top / E$, $E$ — the largest eigenvalue of $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{S} = \mathbf{I}_{D\times D} - \mathbf{W}\mathbf{X}$.
2: Initialisation. Dense layer $\mathbf{b} \leftarrow \mathbf{W}\mathbf{y}$
3: Initialisation. Soft-thresholding nonlinearity $\widehat{\boldsymbol{\beta}}_0 \leftarrow h_\lambda(\mathbf{b})$
4: repeat
5:     Dense layer $\mathbf{c}_l \leftarrow \mathbf{b} + \mathbf{S}\widehat{\boldsymbol{\beta}}_{l-1}$
6:     Soft-thresholding nonlinearity $\widehat{\boldsymbol{\beta}}_l \leftarrow h_\lambda(\mathbf{c}_l)$
7: until converged
8: return $\widehat{\boldsymbol{\beta}} \leftarrow \widehat{\boldsymbol{\beta}}_L$
```

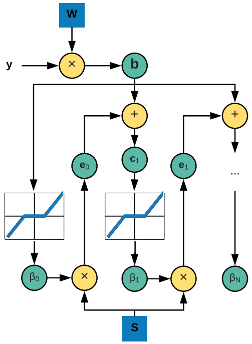
Daubechies I., Defrise M., De Mol C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on pure and applied mathematics, 2004.

Problem

Train a model on N data examples $\mathbf{B}, \mathbf{Y}$ that estimates $\boldsymbol{\beta}$ from observations $\mathbf{y}$ such that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, and elements of $\boldsymbol{\beta}$ contain zeros.

Learned ISTA (LISTA)

Learn weights $\mathbf{W}, \mathbf{S}$ based on training data



Forward pass

```
Require: observation $\mathbf{y}$, current weights $\mathbf{W}, \mathbf{S}$,
    current $\lambda$, number of layers $L$
1: Initialisation. Dense layer $\mathbf{b} \leftarrow \mathbf{W}\mathbf{y}$
2: Initialisation. Soft-thresholding nonlinearity
    $\widehat{\boldsymbol{\beta}}_0 \leftarrow h_\lambda(\mathbf{b})$
3: for $l = 1$ to $L$ do
4:     Dense layer $\mathbf{c}_l \leftarrow \mathbf{b} + \mathbf{S}\widehat{\boldsymbol{\beta}}_{l-1}$
5:     Soft-thresholding nonlinearity $\widehat{\boldsymbol{\beta}}_l \leftarrow h_\lambda(\mathbf{c}_l)$
6: end for
7: return $\widehat{\boldsymbol{\beta}} \leftarrow \widehat{\boldsymbol{\beta}}_L$
```

Backward pass

```
1: update $\mathbf{W}, \mathbf{S}, \lambda$ based on derivatives of the mean
    squared loss
```

Gregor, K., LeCun, Y. Learning fast approximations of sparse coding. ICML 2010

BayesLISTA

Prior

$$p(\mathbf{W}) = \prod_{d=1}^{D} \prod_{k=1}^{K} \mathcal{N}(w_{ij}; 0, \eta^{-1}), \quad p(\mathbf{S}) = \prod_{d'=1}^{D} \prod_{d''=1}^{D} \mathcal{N}(s_{d'd''}; 0, \eta^{-1}),$$

Forward pass

Propagate the distribution for $\widehat{\boldsymbol{\beta}}$ through layers, add Gaussian noise

$$p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{W}, \mathbf{S}, \gamma, \lambda) = \prod_{d=1}^{D} \mathcal{N}\left(\beta_d; [\widehat{\boldsymbol{\beta}}(\mathbf{y}; \mathbf{S}, \mathbf{W}, \lambda)]_d, \gamma^{-1}\right)$$

Posterior

$$p(\mathbf{W}, \mathbf{S}, \gamma, \eta|\mathbf{B}, \mathbf{Y}, \lambda) = \frac{p(\mathbf{B}|\mathbf{Y}, \mathbf{W}, \mathbf{S}, \gamma, \lambda) p(\mathbf{W}|\eta) p(\mathbf{S}|\eta) p(\eta) p(\gamma)}{p(\mathbf{B}|\mathbf{Y}, \lambda)}$$

Backward pass

Update weights with probabilistic backpropagation

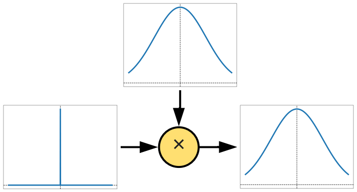
Uncertainty propagation: Initialisation

Goal

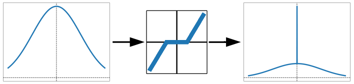
Model intermediate latent variables with parametrised distributions

Apply approximate Bayesian inference methods

1. $\mathbf{b} = \mathbf{W}\mathbf{y}$ is Gaussian-distributed

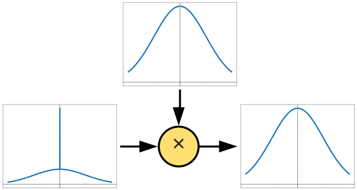


2. $\widehat{\boldsymbol{\beta}}_0 = h_\lambda(\mathbf{b})$ is approximated with the spike and slab distribution

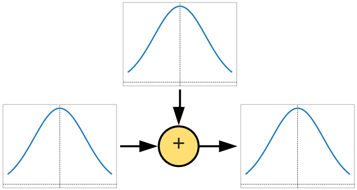


Uncertainty propagation: Iterations

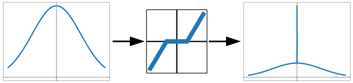
3. $\mathbf{e}_l = \mathbf{S}\widehat{\boldsymbol{\beta}}_{l-1}$ is approximated with the Gaussian distribution



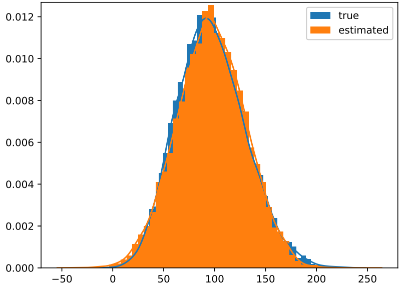
4. $\mathbf{c}_l = \mathbf{b} + \mathbf{e}_l$ is Gaussian-distributed



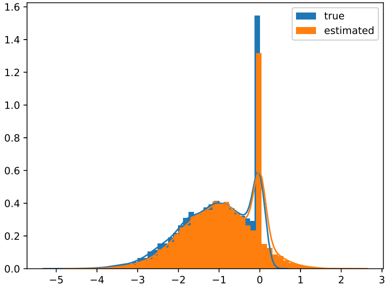
5. $\widehat{\boldsymbol{\beta}}_l = h_\lambda(\mathbf{c}_l)$ is approximated with the spike and slab distribution



Approximation quality



Approximation of the product of Gaussians



Approximation of the propagation through soft thresholding

Probabilistic backpropagation

Posterior

$$p(\mathbf{W}, \mathbf{S}, \gamma, \eta | \mathbf{B}, \mathbf{Y}, \lambda) = \frac{p(\mathbf{B}|\mathbf{Y}, \mathbf{W}, \mathbf{S}, \gamma, \lambda) p(\mathbf{W}|\eta) p(\mathbf{S}|\eta) p(\eta) p(\gamma)}{p(\mathbf{B}|\mathbf{Y}, \lambda)}$$

Approximate posterior

$$q(\mathbf{W}, \mathbf{S}, \gamma, \eta) = \prod_{d=1}^{D} \prod_{k=1}^{K} \mathcal{N}(w_{dk}; m^w_{dk}, v^w_{dk}) \prod_{d'=1}^{D} \prod_{d''=1}^{D} \mathcal{N}(s_{d'd''}; m^s_{d'd''}, v^s_{d'd''}) \\ \times \mathrm{Gam}(\gamma; a^\gamma, b^\gamma) \mathrm{Gam}(\eta; a^\eta, b^\eta)$$

Assumed density filtering (ADF)

Iteratively add factors from the posterior p one-by-one in the factorised approximating distribution q and update the approximation q.

Hernandez-Lobato, J. M., Adams, R. Probabilistic backpropagation for scalable learning of Bayesian neural networks, ICML 2015

Probabilistic backpropagation

Updates for Gaussian parameters

$$q(a) = Z^{-1} f(a)\mathcal{N}(a; m, v)$$

For each factor $a \in \{w_{dk}, s_{d'd''}\}$ new parameters are:

$$m \leftarrow m + v\frac{\partial \log Z}{\partial m},\ v \leftarrow v - v^2\left[\left(\frac{\partial \log Z}{\partial m}\right)^2 - 2\frac{\partial \log Z}{\partial v}\right]$$

$\frac{\partial \log Z}{\partial m}$, $\frac{\partial \log Z}{\partial v}$ are required

Updates for Gamma parameters

For each factor $a \in \{\eta, \gamma\}$ new parameters are:

$$\alpha \leftarrow [Z_0 Z_2 Z_1^{-2}(\alpha+1)/\alpha - 1]^{-1},\ \beta \leftarrow [Z_2 Z_1^{-1}(\alpha+1)/\beta - Z_1 Z_0^{-1}\alpha/\beta]^{-1}$$

$Z_0 = Z(\alpha, \beta)$, $Z_1 = Z(\alpha+1, \beta)$, $Z_2 = Z(\alpha+2, \beta)$ are required

Probabilistic backpropagation

The normalising constant when incorporating the likelihood factor:

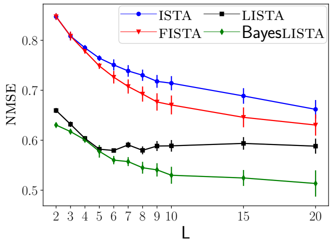
$$Z = \int \prod_{d=1}^{D} \mathcal{N}(\beta_d; [\widehat{\boldsymbol{\beta}}(\mathbf{y}; \mathbf{S}, \mathbf{W}, \lambda)]_d, \gamma^{-1}) q(\mathbf{W}, \mathbf{S}, \gamma, \eta) \mathrm{d}\mathbf{W} \mathrm{d}\mathbf{S} \mathrm{d}\gamma \mathrm{d}\eta$$

Assuming the spike and slab distribution for $\widehat{\boldsymbol{\beta}}$:

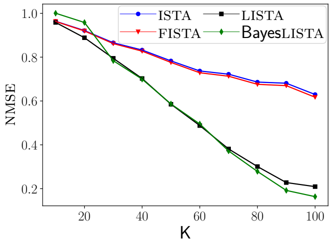
$$Z \approx \prod_{d=1}^{D} \left[\omega_d^{\widehat{\boldsymbol{\beta}}} \mathcal{T}\left(\beta_d; 0, \beta^\gamma/\alpha^\gamma, 2\alpha^\gamma\right) + \left(1 - \omega_d^{\widehat{\boldsymbol{\beta}}}\right) \mathcal{N}\left(\beta_d; m_d^{\widehat{\boldsymbol{\beta}}}, \beta^\gamma/(\alpha^\gamma - 1) + v_d^{\widehat{\boldsymbol{\beta}}}\right) \right],$$

where $\{\omega_d^{\widehat{\boldsymbol{\beta}}}, m_d^{\widehat{\boldsymbol{\beta}}}, v_d^{\widehat{\boldsymbol{\beta}}}\}$ are the parameters of the spike and slab distribution for $[\widehat{\boldsymbol{\beta}}]_d$.

Synthetic Experiments



Different depth performance



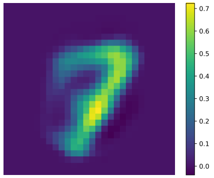
Different observation size performance

$$N_{\text{train}} = 1000,\ D = 100,\ p(\beta_d == 0) = 0.8,\ \eta^{-1} = 0.25,\ \lambda = 0.1$$

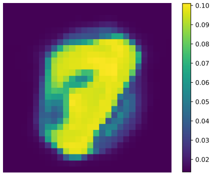
MNIST Experiments

$\mathbf{X}$ - random i.i.d, $D = 784$, $K = 100$

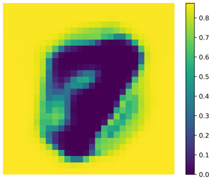
Posterior parameters for an image of digit 7



$\boldsymbol{\beta}$ posterior mean

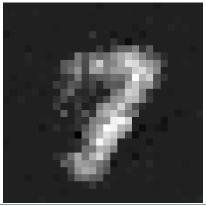


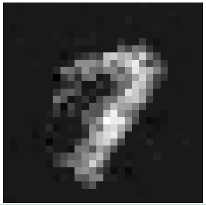
$\boldsymbol{\beta}$ posterior std

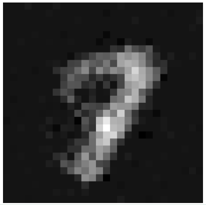


$\boldsymbol{\beta}$ posterior spike indicator

Samples from the posterior for an image of digit 7





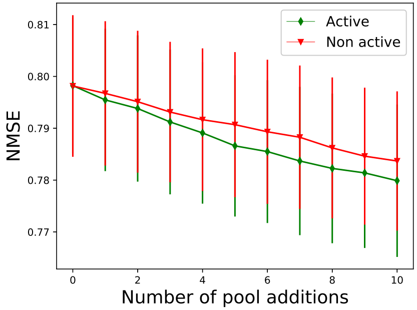


Active Learning

Idea

Use the estimated uncertainty to choose next training data with largest variance

$$N_{\text{train}} = 50,\ N_{\text{pool}} = 500,\ N_{\text{addition}} = 1$$



Contributions and future work

Key contributions

- Bayesian version of LISTA
- Uncertainty propagation to make inference feasible
- Uncertainty quantification, for example, for active learning

Future work

- The shrinkage parameter λ as a parameter of the model
- Stochastic inference

Code will be available at danilkuzin.github.io