Learning Smooth Conditional Class Probability Functions With Small Outcomes Using Deep Neural Networks: From Statistical Theory to Practice

A presentation about my MSc Statistics & Data Science thesis

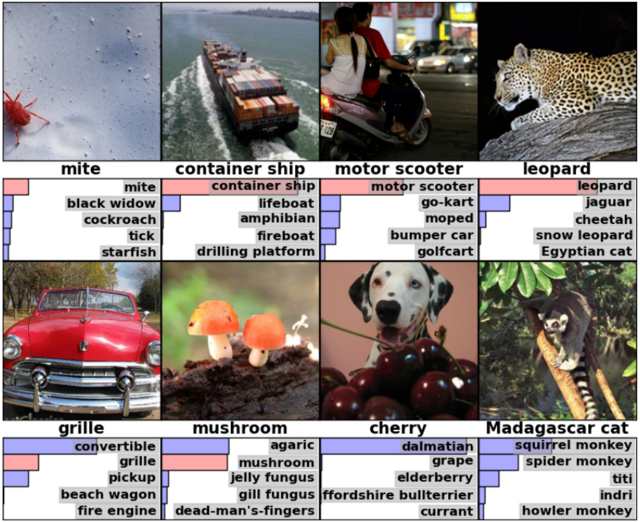
H. C. (Bram) Otten

October 25, 2021

Introduction

- Neural networks perform many machine learning tasks well.
- Practical experiments versus theoretical developments.

- Bos and Schmidt-Hieber (2021).
- Simulation study.

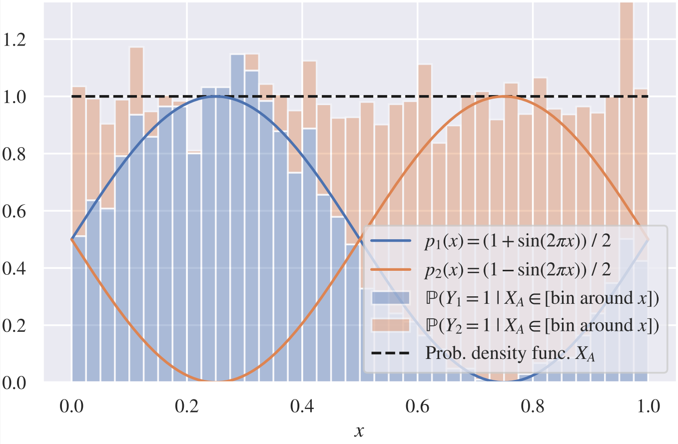


From Krizhevsky, Sutskever, and Hinton (2012)

Supervised Classification Model & Risk

- There are a distribution of input $\boldsymbol{X}$ and a conditional class probability function $\boldsymbol{p} : D^d \to [0, 1]^K$.
- Labels $\boldsymbol{Y}$ are sampled from a categorical distribution with probability vector $\boldsymbol{p}(\boldsymbol{X})$, where $p_k(\boldsymbol{x}) := \mathbb{P}_{\boldsymbol{Y}|\boldsymbol{X}}(Y_k = 1 \mid \boldsymbol{X} = \boldsymbol{x})$.

- Neural networks learn to approximate $\boldsymbol{p}$ using $\frac{1}{n} \sum_{j=1}^{n} \sum_{k=1}^{K} L(\widehat{\boldsymbol{p}}_k(\boldsymbol{X}_j), \boldsymbol{Y}_j^k)$, where L is a chosen loss function.
- *Risk* is the expected loss.



$X_{\text{A}} \sim \text{uniform}([0, 1])$, and $n = 4096$.

Almost the Main Risk Bound of Bos and Schmidt-Hieber (2021)

- Küllback-Leibler divergence risk:

$$\mathbb{E}_{\boldsymbol{X}} \operatorname{KL}(\widehat{\boldsymbol{p}}(\boldsymbol{X}), \boldsymbol{p}(\boldsymbol{X})) = \mathbb{E}_{\boldsymbol{X}}\left[\sum_{k=1}^{K} p_k(\boldsymbol{X}) \log\left(\frac{p_k(\boldsymbol{X})}{\widehat{p}_k(\boldsymbol{X})}\right)\right].$$

- Smaller $p_k \implies$ larger KL.
- For proportion of small p: α-small value bound $\exists C\, \forall k\;\; \mathbb{P}_{\boldsymbol{X}}(p_k(\boldsymbol{X}) \leq t) \leq Ct^{\alpha}$.
- Larger small value bound index $\alpha \implies$ more conditional class probabilities away from zero.

Main Risk Bound of Bos and Schmidt-Hieber (2021)

- Main risk bound:

$$\mathbb{E}_{\boldsymbol{X}}\left[\sum_{k=1}^{K} p_k(\boldsymbol{X}) \min\left\{B, \log\left(\frac{p_k(\boldsymbol{X})}{\hat{p}(\boldsymbol{X})}\right)\right\}\right] \leq C' B L\, \phi_n (\log n)^2,$$

where the rate

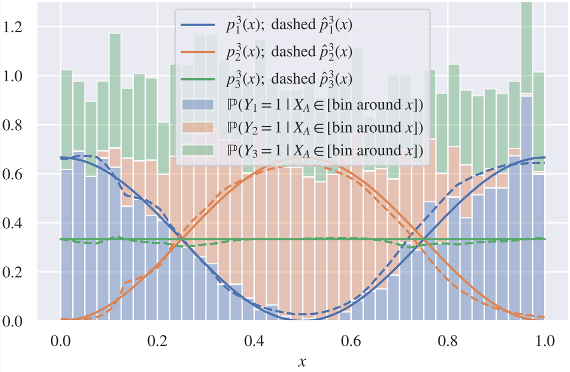
$$\phi_n = \begin{cases} K^{\frac{(1+\alpha)\beta+(3+\alpha)d}{(1+\alpha)\beta+d}} n^{-\frac{(1+\alpha)\beta}{\beta(1+\alpha)+d}}, & \text{if } \alpha \in [0,1] \\ K^{\frac{2\beta+4d}{2\beta+d}} n^{-\frac{2\beta}{2\beta+d}}, & \text{if } \alpha > 1 \end{cases}$$

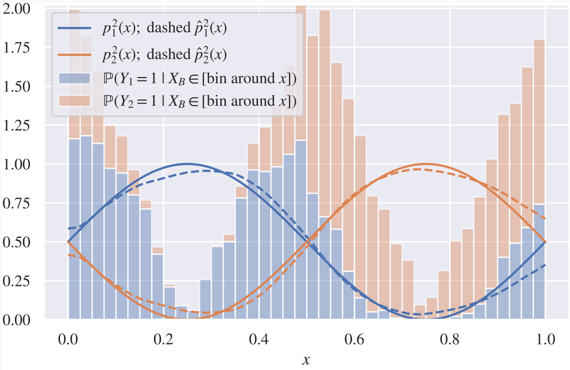
and C', B, L, d, and K are "constants" that do not matter for convergence rate.

- Larger small value bound index $\alpha \implies$ at least as fast of a convergence rate.
- Larger Hölder smoothness index $\beta \implies$ faster convergence rate.
- Arbitrarily high $\beta \implies \phi_n \asymp n^{-1}$.

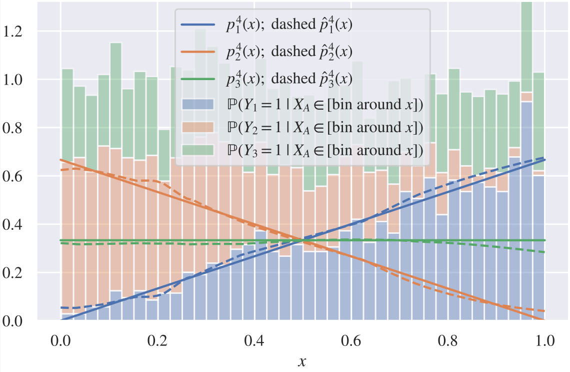
Simulation Study Setup

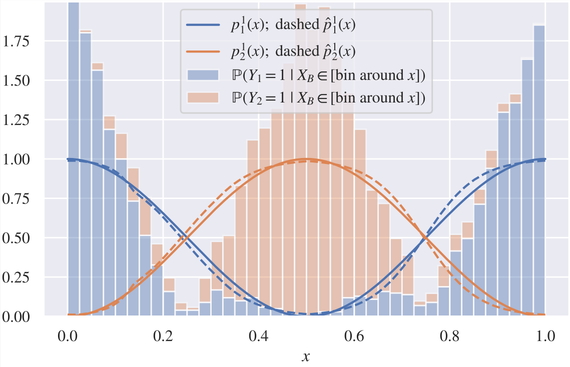
- Define illustrative scenarios (combinations of $\boldsymbol{X}$ and $\boldsymbol{p}$), all arbitrarily high β.
- Calculate relevant quantities for the main risk bound.
- Train and evaluate forty "optimal" ReLU + softmax networks per n per scenario.
- Compare convergence rates of *estimated risks* to expected n^{-1}.





[[More Scenarios]]





- Two-dimensional input scenarios C5 and C6 can not be visualized well.

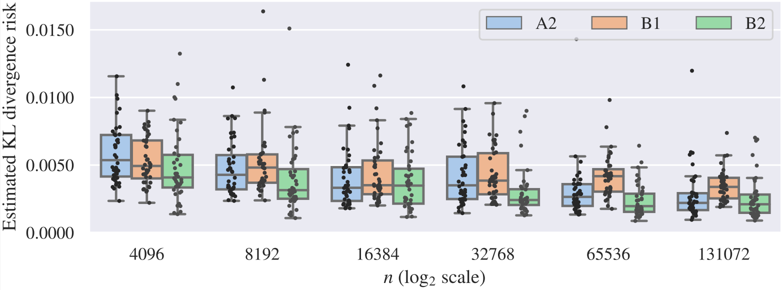
[[Deep Neural Networks]]

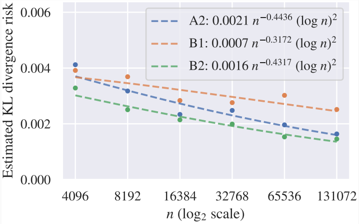
- Implementation in Keras/TensorFlow on Python.
- Rectified linear units (ReLUs) in d-dimensional input and L hidden layers.
- Softmax activation function in K-dimensional output layer.
- He normal / Glorot uniform initialization and L_1 regularization on weights.
- Adam optimizer minimizes negative log-likelihood; batch size 128.
- Early stopping when validation loss fails to decrease by more than 0.005 over 50 epochs.
- “Optimized” (hyper)parameters: learning rate, regularization penalty, as well as depth- and width-related parameters.
- $2 \times 3 \times 30$ Bayesian optimization iterations with $n \in \{8192, 65536\}$ and scenario $\in \{A3, B2, C5\}$.

[[Experimental Setup]]

- Per situation and $n \in \{4096, 8192, 16384, 32768, 65536, 131072\}$, using specific randomness seeds:
  - Generate a test set of size $m = 10^5$.
  - Forty iterations of:
    1. Generating a training set of size n and a validation set of size 10^4.
    2. Training *two* networks with the same architecture, hyperparameters, and training and validation sets.
    3. Evaluating the network with the lowest validation loss on the test set as well, without retraining.
       In this step, we obtain our estimated risks.
  - Fit $\theta_1 n^{\theta_2} (\log n)^2$ to the first quartile of the forty estimated risks, s_n.

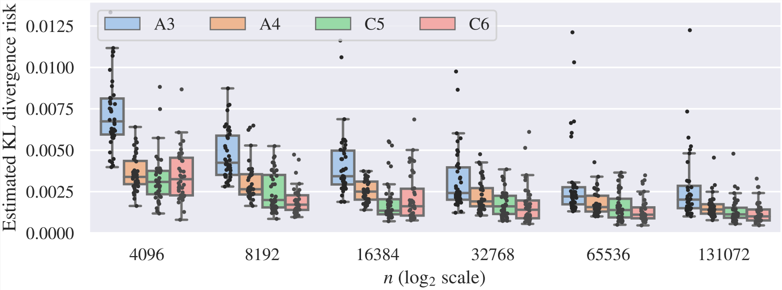
Results: Binary Classification With One-Dimensional Input Scenarios

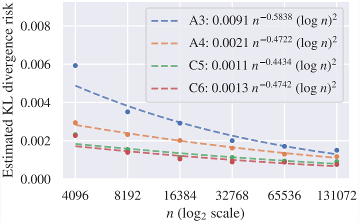




- Consider only convergence *rate*.
- Observe slower rates than n^{-1}.
- In particular in the B1 scenario (1/2-SVB).

Results: Multiclass Classification and Multi-Dimensional Input Scenarios



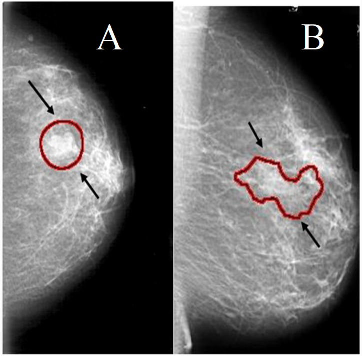


- Again observe slower rates than n^{-1}.
- Relatively fast convergence in the A3 scenario (1/2-SVB).

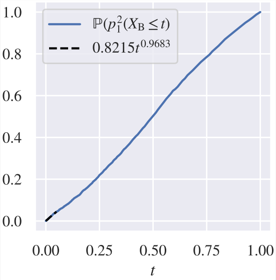
Discussion

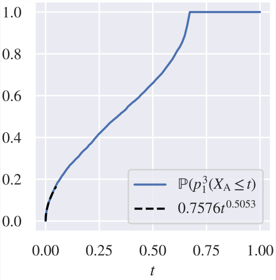
- Slower convergence rates than suggested by the main risk bound.
- Arbitrarily high $\beta \implies$ no consistent effect of α on rate.
- Bridging the gap between theory and practice is difficult.
- Future work:
  - Examine more $n, d, K;\ \alpha, \beta$.
  - [[Examine on (non-existent) datasets with empirical $\boldsymbol{p}$.]]
- `https://github.com/bramotten/DNN-Classification-Theory-In-Practice`

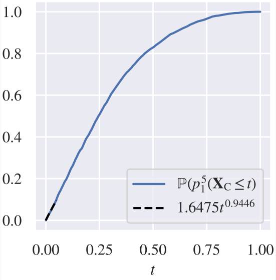
[[Pictures for (Non-existent) Datasets With Empirical p Idea.]]

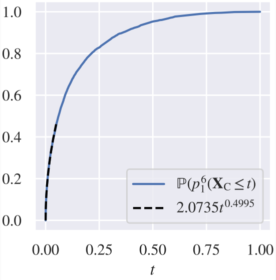


From Ragab et al. (2019)









References

Bos, T. and J. Schmidt-Hieber (2021). "Convergence rates of deep ReLU networks for multiclass classification." arXiv: 2108.00969.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in Neural Information Processing Systems* 25, pages 1097–1105. URL: https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Ragab, D. A., M. Sharkas, S. Marshall, and J. Ren (2019). "Breast cancer detection using deep convolutional neural networks and support vector machines." *PeerJ* 7. Publisher: PeerJ Inc., e6201. ISSN: 2167-8359. DOI: 10.7717/peerj.6201.