

Fast learning with deep learning architectures for classification

Yongdai Kim ¹
jointly with
Ilsang Ohn and Dongha Kim

¹Department of Statistics, Seoul National University, South Korea

Congratulate Aad's 60th Birthday!
and
thanks the organizers for kind invitation.

Outline

- 1 Introduction
- 2 Estimation of a decision boundary: Review
- 3 Fast learning with DNN
- 4 Computational considerations (under construction)
- 5 Summary

Outline

- 1 Introduction
- 2 Estimation of a decision boundary: Review
- 3 Fast learning with DNN
- 4 Computational considerations (under construction)
- 5 Summary

Deep learning application



What is deep learning?

- A machine learning technique for recognition, generation and reconstruction of images, languages, voices etc.
- A method to construct a predictive model by learning a deep neural network (DNN).
- Four ingredients in deep learning
 - Data
 - DNN architectures (Model: fully connected DNN, CNN, RNN, DGM (deep generative model), ...)
 - Objective functions (e.g. log-likelihood, variational lower bound, cross-entropy, ...)
 - Optimization algorithms (eg. stochastic gradient descent, RMSProp, Adam, ...)

Fully connected deep neural network

- The simplest DNN architecture
- Equation for the DNN with L many layers:
 - $\mathbf{h}^{(0)} = \mathbf{x} \in \mathbb{R}^p$ and $n_0 = p$.
 - $z_j^{(l)} = \sum_{i=1}^{n_{l-1}} w_{ij}^{(l-1)} h_i^{(l-1)} + b_j^{(l)}, l = 1, \dots, L$
 - $h_j^{(l)} = \sigma(z_j^{(l)}), l = 1, \dots, L$ (hidden layer)

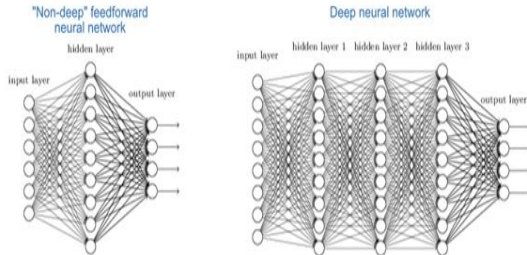
•

$$\Pr(Y = k|\mathbf{x}) = \frac{\exp(f_k(\mathbf{x}))}{\sum_{j=1}^K \exp(f_j(\mathbf{x}))},$$

where $f_k(\mathbf{x}) = \mathbf{h}^{(L)\top} \beta_k$.

- Parameters: $(\mathbf{W}, \mathbf{b}, \beta)$.
- $\sigma(\cdot)$ is the activation function such as
 - Sigmoid: $\sigma(z) = (1 + \exp(-z))^{-1}$
 - ReLU: $\sigma(z) = (z)_+$.

Shallow vs Deep



Explanation of the success of DNN 1: Complexity

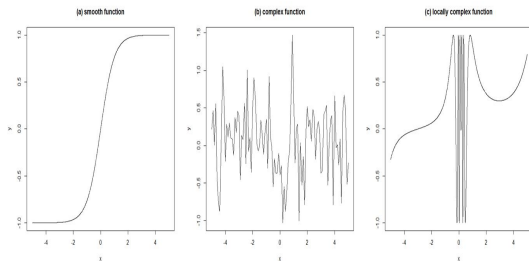
- A theoretical explanation for the superiority of DNN to shallow NN
- DNN can approximate a complex function with fewer parameters than shallow NN.
- Eldan and Shamir (2016)
 - Any shallow NN can be represented by a two layer DNN with only twice more parameters.
 - On the other hand, there is a two layer DNN which cannot be approximated by a shallow NN with polynomially many hidden nodes.
- Montufar et al. (2014)
 - Consider DNN with the ReLU activation function.
 - $f_k(\mathbf{x})$ is piecewise linear.
 - The number of linear regions (regions with different gradients) grows exponentially in L , but polynomially in n_l .

Explanation of the success of DNN 2: Invariancy

- Another explanation of the success of DNN, in particular for image recognition.
- For image recognition, a decision rule should be invariant to a small modification due to translation, rotation, illumination, occlusion and so on.
- Goodfellow et al. (2009) : Invariance of DNN increase as L increase.
- Miyato et al. (2017) : Accuracy of DNN is improved by adding a penalty to the objective function which encourages invariancy.

Motivation of this research

- Complexity and invariancy are seemingly contradictory concepts.
- Three types of functions



- Conjecture: DNN is good at approximating a locally complex function.
- In this talk, I will explain that DNN is good at estimating the classifier (i.e. the decision boundary) which is most locally complex.

Recent literatures

- Bauer and Kohler, 2019, On deep learning as a remedy for the curse of dimensionality in nonparametric regression, AoS.
- Schmidt-Hieber, 2017, Nonparametric regression using deep neural networks with ReLU activation function, Accepted by AoS
- Imaizumi and Fukumizu, 2019, Deep neural networks learn non-smooth functions effectively, AISATS.
- All of these results are about regression problems.

Outline

- 1 Introduction
- 2 Estimation of a decision boundary: Review**
- 3 Fast learning with DNN
- 4 Computational considerations (under construction)
- 5 Summary

Classification and decision boundary (classifier)

- Consider a two-class classification problem: $Y \in \{-1, 1\}$
- Input: $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^p$
- Data: $\mathcal{L} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ which are assumed to be independent realizations of $(\mathbf{X}, Y) \sim \mathbb{P}$.
- A classifier $C : \mathcal{X} \rightarrow \{-1, 1\}$.
- Optimal classifier (Bayes classifier) C^* is defined as

$$C^*(\mathbf{x}) = \operatorname{argmin}_C \mathbb{E} I\{Y \neq C(\mathbf{X})\}.$$

- It is well known that the Bayes classifier is given as

$$C^*(\mathbf{x}) = 2I\{\mathbb{P}(Y = 1|\mathbf{x}) > 0.5\} - 1.$$

Empirical Risk minimization

- A method to estimate the Bayes classifier
- (0-1) Empirical risk:

$$R_n(C) = \frac{1}{n} \sum_{i=1}^n I(y_i \neq C(\mathbf{x}_i)).$$

- Let $\mathcal{C}_n$ be a given set of classifiers (e.g. a given model, sieve)
- Empirical risk minimization method estimates C^* by

$$\hat{C} = \operatorname{argmin}_{C \in \mathcal{C}_n} R_n(C).$$

- The law of large numbers implies $R_n(C) \rightarrow R(C)$ for given C , where $R(C) = \mathbb{E}I\{Y \neq C(\mathbf{X})\}$ (i.e. population misclassification error).
 - Hence, we expect that $\hat{C} \rightarrow C^*$ provided that C^* belongs to $\mathcal{C}_n$ eventually.
- (*) We consider a sieve to avoid overfitting.

A difficulty in ERM with 0-1 loss

- In practice, it is almost impossible to minimize $R_n(\cdot)$.
- Under regularity conditions, it is known that minimizing $R_n(\cdot)$ is NP-complete.
- Difficulty in minimizing $R_n(\cdot)$ is partly due to the indicator function $I(y \neq C(\mathbf{x}))$ called the 0-1 loss, which is neither continuous nor convex.
- To resolve this problem, we use convex surrogated loss functions other than the 0-1 loss.

Surrogated loss and function estimation

- For a given convex surrogated loss ϕ , the empirical risk of a real-valued function f is defined as

$$R_{\phi,n}(f) = \frac{1}{n} \sum_{i=1}^n \phi(y_i f(\mathbf{x}_i)).$$

- Let $\mathcal{F}_n$ be a given set of (real-valued) functions.
- We estimate f by minimizing $R_{\phi,n}(\cdot)$ on $\mathcal{F}_n$.
- We construct the classifier from $\hat{f}$ by $\hat{C}(\mathbf{x}) = \text{sign}\{\hat{f}(\mathbf{x})\}$.
- Examples of surrogated losses are
 - Exponential loss: $\phi(z) = \exp(-z)$
 - Logistic loss: $\phi(z) = \log(1 + \exp(-z))$
 - Hinge loss: $\phi(z) = (1 - z)_+$.

Loss and decision boundary

- For a given surrogated loss ϕ , define f_ϕ^* as

$$f_\phi^* = \operatorname{argmin}_{f \in \mathcal{F}_\infty} \mathbb{E}(\phi(Yf(\mathbf{X}))),$$

where $\mathcal{F}_\infty$ is the set of functions such that, roughly speaking, $\mathcal{F}_n \rightarrow \mathcal{F}_\infty$.

- We expect that $\hat{f}$ converges to f_ϕ^* .
- Let $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 | \mathbf{X} = \mathbf{x})$ be the conditional class probability.
- It can be shown that
 - $f_\phi^*(\mathbf{x}) = \frac{1}{2} \log \frac{\eta(\mathbf{x})}{1-\eta(\mathbf{x})}$ for the exponential loss
 - $f_\phi^*(\mathbf{x}) = \log \frac{\eta(\mathbf{x})}{1-\eta(\mathbf{x})}$ for the logistic loss
 - $f_\phi^*(\mathbf{x}) = 2I(\eta(\mathbf{x}) \geq 1/2) - 1$ for the hinge loss.

provided that $f_\phi^* \in \mathcal{F}_\infty$.

- Note that the hinge loss estimates the classifier directly while the exponential and logistic losses estimate the probability.**

Optimal boundary estimation

- For given f , let $R(f) = \mathbb{E}I(Yf(\mathbf{x}) < 0)$ (i.e. misclassification error).
- Mammen and Tsybakov (1999) proved that estimating the classifier directly is more efficient than estimating the probability.
- Excess risk of f : $R(f) - R(C^*)$
- Roughly speaking, Mammen and Tsybakov (1999) proved that

$$O_p(1/n) \leq R(\hat{f}) - R(C^*) < O_p(1/\sqrt{n})$$

when $\hat{f}$ directly estimates the classifier while

$$R(\hat{f}) - R(C^*) \geq O_p(1/\sqrt{n})$$

when $\hat{f}$ estimates the conditional class probability.

Fast convergence rate and surrogated loss

- We say that $\hat{f}$ achieves a fast convergence rate if

$$R(\hat{f}) - R(f^*) = o_p(1/\sqrt{n}).$$

- Mammen and Tsybakov (1999) and Tsybakov (2004) proved that $\hat{f}$ directly minimizing the 0-1 empirical risk achieves fast convergence rates.
- However, as mentioned earlier, minimizing the 0-1 empirical risk is computationally demanding.
- Since the hinge loss directly estimates the decision boundary, it is a good candidate for fast convergence rate.
- In fact, Steinwart and Scovel (2007) proved that the hinge loss and the RKHS (Reproducing Kernel Hilbert Space) with a Gaussian kernel as $\mathcal{F}_n$ (i.e. SVM) achieves a fast convergence rate.
- Shen et al. (2003) proved a similar result with the ψ -loss which is nonconvex.

Choice of $\mathcal{F}_n$

- A difficulty in estimating the classifier directly lies in the choice of $\mathcal{F}_n$ such that $C^* \in \mathcal{F}_\infty$.
- Most of practically used $\mathcal{F}_n$ such as splines and RKHS consist of only smooth continuous functions.
- It would be difficult to approximate the classifier itself by a smooth continuous function since the classifier is discontinuous.
- The fast convergence rate of Steinwart and Scovel (2007) for the SVM requires stronger conditions on the true distribution and is much slower than the optimal one.
- So far, not much is known about how to choose $\mathcal{F}_n$ optimally for estimating the classifier.
- We will show that DNN is a good candidate for $\mathcal{F}_n$.

Outline

- 1 Introduction
- 2 Estimation of a decision boundary: Review
- 3 Fast learning with DNN**
- 4 Computational considerations (under construction)
- 5 Summary

Assumptions

- Tsybakov's noise condition: There exist $C > 0$ and $q \geq 0$ such that

$$\mathbb{P} \{ \mathbf{X} : |2\eta(\mathbf{X}) - 1| > t \} \leq Ct^q$$

for all t .

- $\mathcal{F}_n$ with DNN
 - L_n : the number of layers
 - N_n : the maximum number of nodes at each layer
 - S_n : the maximum number of nonzero weights (i.e. sparsity)
 - B_n : the upper bound of $w_{ij}^{(l)}$
 - F_n : the upper bound of $\sup_k \sup_{\mathbf{x} \in \mathcal{X}} |f_k(\mathbf{x})|$.
- Loss function: Hinge loss
- That is

$$\hat{f}_n = \sup_{f \in \mathcal{F}_n} \sum_{i=1}^n (1 - y_i f(\mathbf{x}_i))_+$$

Case 1: Smooth decision boundary

- $\mathcal{H}^{\alpha,r}([0, 1]^{p-1})$: be the class of α -Hölder continuous function on $[0, 1]^{p-1}$ whose Hölder norm is less than or equal to $r > 0$.
- Roughly speaking, we assume that the true decision boundary is a $p - 1$ dimensional manifold belonging to $\mathcal{H}^{\alpha,r}([0, 1]^{p-1})$.

Rigorous definition of smooth decision boundary

- *horizon function*: $\Psi_{g,j} : [0, 1]^p \rightarrow \{0, 1\}$ as

$$\Psi_{g,j}(\mathbf{x}) := I(x_j \geq g(\mathbf{x}_{-j})),$$

where $\mathbf{x}_{-j} = (x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_p)$.

- *basis piece*: $I_{g,j}$ as

$$I_{g,j} := \{\mathbf{x} \in [0, 1]^p : \Psi_{g,j}(\mathbf{x}) = 1\}.$$

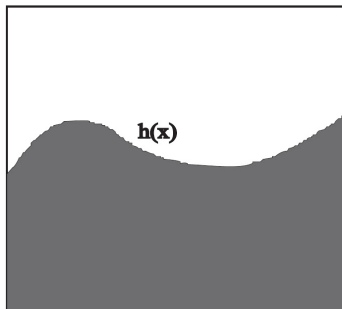
- *The set of pieces*:

$$\mathcal{A}^{\alpha,r,M} := \left\{ A \subset [0, 1]^p : A = \bigcap_{m=1}^M I_{g_m,j_m}, g_m \in \mathcal{H}^{\alpha,r}([0, 1]^{p-1}), j_m \in \{1, \dots, p\} \right\}.$$

- Let

$$\mathcal{C}^{\alpha,r,M,K} = \{C : C(\mathbf{x}) = 2 \sum_{k=1}^K I(\mathbf{x} \in A_k) - 1, A_k \in \mathcal{A}^{\alpha,r,M}, A_k \cap A_{k'} = \emptyset \text{ for some } K\}$$

Example of Horizon function



Theorem

- We can choose L_n, N_n, S_n, B_n , and F_n to have

$$\sup_{C^* \in \mathcal{C}^{\alpha, r, M, K}} \mathbb{E}_{\mathcal{L}} \left[R(\hat{f}_n) - R(C^*) \right] \lesssim \left(\frac{\log^3 n}{n} \right)^{\frac{\alpha(q+1)}{\alpha(q+2) + (p-1)(q+1)}} \quad (1)$$

- The above rate is slightly slower than the minimax lower bound (Mammen and Tsybakov, 1999) which is

$$n^{-\frac{\alpha(q+1)}{\alpha(q+2) + (p-1)q}}.$$

- It might be because we use the hinge loss instead of the 0-1 loss.

Comparison with the previous results

- Mammen and Tsybakov (1999) and Tsybakov (2004) prove that the optimal convergence rate is achievable as long as the sieve is carefully chosen.
- Tsybakov and van de Geer (2005) propose a specific sieve based on the wavelet basis expansion of the decision boundary to be able to achieve the optimal convergence rate.
- All of these results consider minimizing the 0-1 empirical risk on a given sieve of classifiers which is computationally demanding.
- **Our sieve and empirical risk, DNN for $\mathcal{F}_n$ and hinge loss, is at least continuous in parameters!**

Case 2: Smooth conditional class probability

- Assume that $\eta(\cdot) \in \mathcal{H}^{\beta,r}([0, 1]^p)$ for some $\beta > 0$ and $r > 0$.
- **Theorem:** We can choose L_n, N_n, S_n, B_n , and F_n to have

$$\sup_{\eta \in \mathcal{H}^{\beta,r}} \mathbb{E} \left[R(\hat{f}_n) - R(C^*) \right] \lesssim \left(\frac{\log^3 n}{n} \right)^{\frac{\beta(q+1)}{\beta(q+2)+p}}. \quad (2)$$

- The above rate is minimax-optimal (Audibert and Tsybakov, 2007) upto the logarithm factor.
- Audibert and Tsybakov (2007) proposes an estimator achieving the optimal convergence rate but still uses the 0-1 empirical risk.

Case 3: Large margin condition

- Let $D^* = \{\mathbf{x} : \eta(\mathbf{x}) = 1/2\}$ be the optimal decision boundary.
- Large margin condition: There exist $C > 0$, $\epsilon_0 > 0$, and $\gamma \in [1, \infty]$ such that for any $\epsilon \in (0, \epsilon_0]$,

$$\Pr(\{\mathbf{X} : \text{dist}(\mathbf{X}, D^*) \leq \epsilon\}) \leq C\epsilon^\gamma.$$

- **Theorem:** We can choose L_n, N_n, S_n, B_n , and F_n to have

$$\sup_{C^* \in \mathcal{C}^{\alpha, r, M, K}} \mathbb{E} \left[R(\hat{f}_n) - R(C^*) \right] \lesssim \left(\frac{\log^3 n}{n} \right)^{\frac{\alpha(q+1)}{\alpha(q+2) + (p-1)(q+1)/\gamma}}. \quad (3)$$

- A similar convergence rate is obtained by Steinwart and Scovel (2007) for the SVM.
- The result partly explains how the curse of dimensionality would be avoided in practice.

Empirical evidence of large margin assumption



Main Techniques of the proofs (e.g. Theorem 2)

- Note that

$$C^*(\mathbf{x}) = 2g \circ \{\eta(\mathbf{x}) - 1/2\} - 1$$

where $\eta(\mathbf{x}) = \mathbb{P}(Y = 1|\mathbf{x})$ and $g(z) = I(z > 0)$.

- There are two deep neural networks which approximate η and g well.
- For η , Yarotsky (2017) proves that DNN is good at approximating a smooth continuous function in terms of sup norm.
- For g , note that

$$\frac{1}{\epsilon} \{\sigma(z + \epsilon) - \sigma(z)\} = \frac{z}{\epsilon} I(0 < z < \epsilon) + I(z \geq \epsilon) \rightarrow g(z) \quad (4)$$

as $\epsilon \rightarrow 0$ in some sense where $\sigma(z) = (z)_+$ is the ReLU.

- Combining these two approximation results, we prove that DNN is good at approximating $C^*(\mathbf{x})$ when $\eta(\mathbf{x})$ is smooth.
 - Using standard empirical process techniques with a carefully calculated complexity of $\mathcal{F}_n$, we derive the fast convergence rates.
- (*) Petersen and Voigtlaender (2018) utilized (4) to prove that DNN is good at approximating a discontinuous function in terms of L_2 norm.

Remark on the cross-entropy

- In practice, the cross-entropy (the negative log-likelihood) is popularly used and works quite well.
- In general, the cross-entropy gives slower convergence rates than the hinge loss.
- A possible explanation of practical success of the cross-entropy is that in practice
 - $\eta(\mathbf{x})$ is close to 0 or 1 for most $\mathbf{x}$.
 - That is, most images are easily discriminable.
 - Under this situation, we can derive a fast convergence rate for the cross-entropy.

Outline

- 1 Introduction
- 2 Estimation of a decision boundary: Review
- 3 Fast learning with DNN
- 4 Computational considerations (under construction)**
- 5 Summary

Sparsity assumption

- We assume that the DNN should satisfy a certain sparsity assumption (i.e. S_n) to achieve fast convergence rates.
- In practice, it is not easy to find an appropriate sparse DNN architecture.
- A promising way would be to use a sparse penalty such as Lasso (Liu et al., 2015; Han et al., 2015; Wen et al., 2016).
- There are at least two issues:
 - Choice of the sparse penalty to yield a good convergence rate
 - Computational algorithm

Choice of sparse penalty

- l_1 penalty would not give a good convergence rate since the complexity of l_1 constraint DNNs would be too large.
- l_0 penalty would be a promising one.
- Hereafter, f denotes either the function itself or the parameters (weights and biases) in the corresponding DNN.
- We consider estimating $\hat{f}$ by minimizing

$$\sum_{i=1}^n (1 - y_i f(\mathbf{x}_i))_+ + \lambda_n \|f\|_0.$$

- We have proved that there exists $\{\lambda_n\}$ such that $\hat{f}$ achieves a fast convergence rate.
- But, such $\{\lambda_n\}$ still depends on α (or β) and q .
- We are pursuing a method to select λ_n data-adaptively.

Computational algorithm

- Using l_0 penalty might not work well due to local minimum problems.
- To ease the problem of local minima, we consider the truncated l_1 penalty (Shen et al., 2012) as

$$J_{a_n}(|z|) = I(|z| > a_n) + |z|I(|z| < a_n).$$

- We consider estimating $\hat{f}$ by minimizing

$$\sum_{i=1}^n (1 - y_i f(\mathbf{x}_i))_+ + \lambda_n \sum_t J_{c\lambda_n}(|f_t|) \quad (5)$$

for some $c > 0$.

- We have proved that there exist c and $\{\lambda_n\}$ such that $\hat{f}$ achieves a fast convergence rate.

Conex-ConCave Procedure (CCCP)

- An advantage of using the truncated l_1 penalty is that we can find a local minimum by use of the CCCP algorithm (Kim et al., 2008).
- Let $\hat{f}^{\text{curr}}$ be the current estimate of f .
- The CCCP algorithm updates $\hat{f}$ by minimizing

$$\sum_{i=1}^n (1 - y_i f(\mathbf{x}_i))_+ - \nabla(|\hat{f}^{\text{curr}}|)^T f + \lambda_n \|f\|_1, \quad (6)$$

where

$$\nabla(|z|) = I(|z| > c\lambda_n).$$

- It is known that the objective function (5) decreases at every iteration of CCCP.
- Note that (6) is convex except $f(\mathbf{x})$.

Proximal algorithm

- A standard gradient descent algorithm does not work well for minimizing (6) since it loses the sparsity soon.
- To resolve this problem, we can use the proximal algorithm as follows.
- Stochastic gradient descent algorithm with l_1 penalty
 - $\mathcal{L}_t$: a mini-batch arriving at time t .
 - Let

$$C_t(f) = \sum_{i \in \mathcal{L}_t} (1 - y_i f(\mathbf{x}_i))_+ - \nabla(|\hat{f}^{\text{curr}}|)^T f$$

- Calculate the gradient

$$\nabla_t = \left. \frac{\partial C_t(f)}{\partial f} \right|_{f=f^{(t)}}.$$

Proximal algorithm

(*) Mirror descent algorithm: Let

$$f^{(t+1)} = \operatorname{argmin}_f f^T \nabla_t + \frac{1}{2\eta_t} \|f - f^{(t)}\|^2.$$

It turns out that

$$f^{(t+1)} = f^{(t)} - \eta_t \nabla_t.$$

- Proximal algorithm (Sparse mirror descent algorithm)

$$f^{(t+1)} = \operatorname{argmin}_f f^T \nabla_t + \frac{1}{2\eta_t} \|f - f^{(t)}\|^2 + \lambda \|f\|_1.$$

Outline

- 1 Introduction
- 2 Estimation of a decision boundary: Review
- 3 Fast learning with DNN
- 4 Computational considerations (under construction)
- 5 Summary**

Summary

- We have shown that sparse DNN provides a very flexible class of functions to estimate the decision boundary well.
- Finding the optimal level of sparsity of DNN (theoretically as well as computationally) is an ongoing project.

Thank you!!

References

- Audibert, J.-Y. and Tsybakov, A. B. (2007). Fast learning rates for plug-in classifiers. *The Annals of statistics*, 35(2):608–633.
- Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940.
- Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., and Ng, A. Y. (2009). Measuring invariances in deep networks. In *Advances in neural information processing systems*, pages 646–654.
- Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143.
- Kim, Y., Choi, H., and Oh, H.-S. (2008). Smoothly clipped absolute deviation on high dimensions. *Journal of the American Statistical Association*, 103(484):1665–1673.
- Liu, B., Wang, M., Foroosh, H., Tappen, M., and Pensky, M. (2015). Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814.
- Mammen, E. and Tsybakov, A. B. (1999). Smooth discrimination analysis. *The Annals of Statistics*, 27(6):1808–1829.

References

- Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2017). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*.
- Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932.
- Petersen, P. and Voigtlaender, F. (2018). Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 108:296–330.
- Shen, X., Pan, W., and Zhu, Y. (2012). Likelihood-based selection and sharp parameter estimation. *Journal of the American Statistical Association*, 107(497):223–232.
- Shen, X., Tseng, G. C., Zhang, X., and Wong, W. H. (2003). On ψ -learning. *Journal of the American Statistical Association*, 98(463):724–734.
- Steinwart, I. and Scovel, C. (2007). Fast rates for support vector machines using gaussian kernels. *The Annals of Statistics*, 35(2):575–607.
- Tsybakov, A. B. (2004). Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166.

References

- Tsybakov, A. B. and van de Geer, S. A. (2005). Square root penalty: adaptation to the margin in classification and in edge estimation. *The Annals of Statistics*, 33(3):1203–1224.
- Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082.
- Yarotsky, D. (2017). Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114.