Generalisation Bounds for Neural Networks

Pascale Gourdeau

University of Oxford

15 November 2018

Overview

Overview

What is generalisation?

The ability to *perform well* on unseen data.

What is generalisation?

The ability to *perform well* on unseen data.

- Assumption: the data (both for the training and testing) comes i.i.d. from a distribution D.
- Usually work in a *distribution-agnostic* setting.

What are generalisation bounds?

- *Classification setting*: input space $\mathcal{X}$ and output space $\mathcal{Y} := \{1, \ldots, k\}$ with a distribution D on $\mathcal{X} \times \mathcal{Y}$.

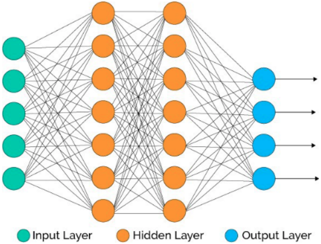
What are generalisation bounds?

- *Classification setting*: input space $\mathcal{X}$ and output space $\mathcal{Y} := \{1, \dots, k\}$ with a distribution D on $\mathcal{X} \times \mathcal{Y}$.
- Goal: to learn a function $f : \mathcal{X} \to \mathcal{Y}$ from a sample $S := \{(x_i, y_i)\}_{i=1}^{m} \subseteq \mathcal{X} \times \mathcal{Y}$.

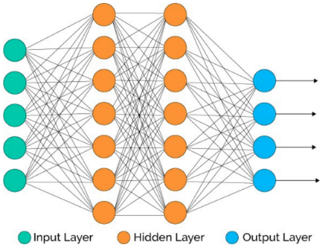
What are generalisation bounds?

- *Classification setting*: input space $\mathcal{X}$ and output space $\mathcal{Y} := \{1, \dots, k\}$ with a distribution D on $\mathcal{X} \times \mathcal{Y}$.
- Goal: to learn a function $f : \mathcal{X} \to \mathcal{Y}$ from a sample $S := \{(x_i, y_i)\}_{i=1}^m \subseteq \mathcal{X} \times \mathcal{Y}$.
- Generalisation bounds: bounding the difference between the expected and empirical losses of f with high probability over S.

What are generalisation bounds?



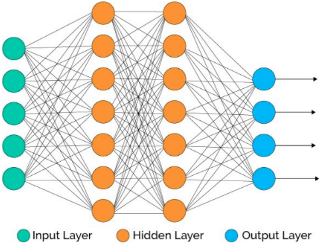
What are generalisation bounds?



For neural networks, we use the expected classification loss:

$$L_0(f) := \mathbb{P}_{(x,y)\sim D}\left(f(x)_y \leq \max_{y' \neq y} f(x)_{y'}\right) \;,$$

What are generalisation bounds?



For neural networks, we use the expected classification loss:

$$L_0(f) := \mathbb{P}_{(x,y)\sim D}\left(f(x)_y \leq \max_{y' \neq y} f(x)_{y'} \right) ,$$

and the empirical margin loss:

$$\widehat{L}_\gamma(f) := \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\left[f(x)_y \leq \gamma + \max_{y' \neq y} f(x)_{y'} \right] .$$

Why are generalisation bounds useful?

- They allow us to quantify a given model's expected generalisation performance.

Why are generalisation bounds useful?

- They allow us to quantify a given model's expected generalisation performance.
  - *E.g.: With probability 95% over the training sample, the error is at most 1%.*

Why are generalisation bounds useful?

- They allow us to quantify a given model's expected generalisation performance.
    - *E.g.: With probability 95% over the training sample, the error is at most 1%.*
- They can also:

Why are generalisation bounds useful?

- They allow us to quantify a given model's expected generalisation performance.
  - *E.g.: With probability 95% over the training sample, the error is at most 1%.*
- They can also:
  - Provide insight on the ability of a model to generalise.

Why are generalisation bounds useful?

- They allow us to quantify a given model's expected generalisation performance.
  - *E.g.: With probability 95% over the training sample, the error is at most 1%.*
- They can also:
  - Provide insight on the ability of a model to generalise.
    - This is of particular interest for us: neural networks have many counter-intuitive properties.

Why are generalisation bounds useful?

- They allow us to quantify a given model's expected generalisation performance.
    - *E.g.: With probability 95% over the training sample, the error is at most 1%.*
- They can also:
    - Provide insight on the ability of a model to generalise.
        - This is of particular interest for us: neural networks have many counter-intuitive properties.
    - Inspire new algorithms or regularisation techniques.

Overview

General Strategies

Generalisation bounds (GB) for neural networks are usually obtained by

1. Defining a class H of functions computed by neural networks with certain properties (e.g., weight matrices with bounded norms, number of layers, etc.),

General Strategies

Generalisation bounds (GB) for neural networks are usually obtained by

1. Defining a class H of functions computed by neural networks with certain properties (e.g., weight matrices with bounded norms, number of layers, etc.),
2. Deriving a generalisation bound in terms of a complexity measure $M(H)$ (e.g. size of H, Rademacher complexity),

General Strategies

Generalisation bounds (GB) for neural networks are usually obtained by

1. Defining a class H of functions computed by neural networks with certain properties (e.g., weight matrices with bounded norms, number of layers, etc.),
2. Deriving a generalisation bound in terms of a complexity measure $M(H)$ (e.g. size of H, Rademacher complexity),
3. Upper bounding $M(H)$ in terms of model parameters (e.g., norm of weight matrices, number of layers, etc.).

General Strategies: Rademacher Complexity

Definition (Rademacher complexity)

Let G be a family of functions from a set $\mathcal{Z}$ to $\mathbb{R}$. Let $\sigma_1, \ldots, \sigma_m$ be Rademacher variables: $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$. The *empirical Rademacher complexity* of G w.r.t. to a sample $S = \{z_i\}_{i=1}^m$ is

Definition (Rademacher complexity)

Let G be a family of functions from a set $\mathcal{Z}$ to $\mathbb{R}$. Let $\sigma_1, \ldots, \sigma_m$ be Rademacher variables: $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$. The *empirical Rademacher complexity* of G w.r.t. to a sample $S = \{z_i\}_{i=1}^m$ is

$$\mathcal{R}_S(G) = \mathbb{E}_\sigma \left[\sup_{g \in G} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(z_i) \right] \quad .$$

General Strategies: Rademacher Complexity

Definition (Rademacher complexity)

Let G be a family of functions from a set $\mathcal{Z}$ to $\mathbb{R}$. Let $\sigma_1, \ldots, \sigma_m$ be Rademacher variables: $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$. The *empirical Rademacher complexity* of G w.r.t. to a sample $S = \{z_i\}_{i=1}^m$ is

$$\mathcal{R}_S(G) = \mathbb{E}_\sigma \left[\sup_{g \in G} \frac{1}{m} \sum_{i=1}^m \sigma_i g(z_i) \right] \quad .$$

Intuition: How much G correlates with random noise on S.

General Strategies: Rademacher Complexity

Definition (Rademacher complexity)

Let G be a family of functions from a set $\mathcal{Z}$ to $\mathbb{R}$. Let $\sigma_1, \ldots, \sigma_m$ be Rademacher variables: $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$. The *empirical Rademacher complexity* of G w.r.t. to a sample $S = \{z_i\}_{i=1}^m$ is

$$\mathcal{R}_S(G) = \mathbb{E}_\sigma \left[\sup_{g \in G} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(z_i) \right] .$$

Intuition: How much G correlates with random noise on S. Simple examples...

General Strategies: Rademacher Complexity

Theorem

Let G be a family of functions from Z to $[0,1]$, and let S be a sample of size m drawn from Z according to D. Let $L(g) = \mathbb{E}_{z \sim D}[g(z)]$ and $\widehat{L}_(g) = \frac{1}{m}\sum_{i=1}^{m} g(z_i)$. Then for any $\delta > 0$, with probability at least $1-\delta$ over S, for all functions $g \in G$,

$$L(g) \leq \widehat{L}(g) + 2\mathcal{R}_S(G) + O\left(\sqrt{\frac{\log(1/\delta)}{m}}\right) .$$

General Strategies: Rademacher Complexity

- Computing the empirical Rademacher complexity (RC) of a given H is usually hard or impractical.
- One usually derives Rademacher complexity upper bounds, for example by using the Dudley entropy integral.

Overview

Generalisation Bounds for Neural Networks

- VC-dimension-based bounds, which usually amount to parameter counting [Goldberg and Jerrum, 1995, Bartlett et al., 1999, Bartlett et al., 2017b].

Generalisation Bounds for Neural Networks

- VC-dimension-based bounds, which usually amount to parameter counting [Goldberg and Jerrum, 1995, Bartlett et al., 1999, Bartlett et al., 2017b].
- Bounds that depend on the norm of the linear transformations [Bartlett, 1997].

Generalisation Bounds for Neural Networks

- VC-dimension-based bounds, which usually amount to parameter counting [Goldberg and Jerrum, 1995, Bartlett et al., 1999, Bartlett et al., 2017b].
- Bounds that depend on the norm of the linear transformations [Bartlett, 1997].
- Spectrally-normalised margin-based bounds [Bartlett et al., 2017a].

Generalisation Bounds for Neural Networks

- VC-dimension-based bounds, which usually amount to parameter counting [Goldberg and Jerrum, 1995, Bartlett et al., 1999, Bartlett et al., 2017b].
- Bounds that depend on the norm of the linear transformations [Bartlett, 1997].
- Spectrally-normalised margin-based bounds [Bartlett et al., 2017a].
- PAC-Bayesian approach to margin-based bounds [Neyshabur et al., 2017].

Overview

Compression Approach: Overview

Paper:

Stronger generalization bounds for deep nets via a compression approach.

Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Compression Approach: Overview

Paper:

Stronger generalization bounds for deep nets via a compression approach.

Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Two methods, both based on compressing a network by representing its weight matrices with fewer parameters.

Compression Approach: Overview

Paper:

Stronger generalization bounds for deep nets via a compression approach.

Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Two methods, both based on compressing a network by representing its weight matrices with fewer parameters.

1. Define *compressibility* of a function f via G, a (finite) set of functions, and derive a generalisation bound that relates the losses of f and G.

Compression Approach: Overview

Paper:

Stronger generalization bounds for deep nets via a compression approach.

Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Two methods, both based on compressing a network by representing its weight matrices with fewer parameters.

1. Define *compressibility* of a function f via G, a (finite) set of functions, and derive a generalisation bound that relates the losses of f and G.
    - f: a neural network; G: class of neural networks that have less parameters and that can approximate f.

Compression Approach: Overview

Paper:

Stronger generalization bounds for deep nets via a compression approach.

Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Two methods, both based on compressing a network by representing its weight matrices with fewer parameters.

1. Define *compressibility* of a function f via G, a (finite) set of functions, and derive a generalisation bound that relates the losses of f and G.
   - f: a neural network; G: class of neural networks that have less parameters and that can approximate f.
   - Results in the same bound as in [Neyshabur et al., 2017].

Compression Approach: Overview

Paper:

Stronger generalization bounds for deep nets via a compression approach.

Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Two methods, both based on compressing a network by representing its weight matrices with fewer parameters.

1. Define *compressibility* of a function f via G, a (finite) set of functions, and derive a generalisation bound that relates the losses of f and G.
   - f: a neural network; G: class of neural networks that have less parameters and that can approximate f.
   - Results in the same bound as in [Neyshabur et al., 2017].
2. A different compression framework based on random projections, together with noise stability properties of the network, gives tighter generalisation bounds than the first method.

Compression Approach: Overview

Paper:
Stronger generalization bounds for deep nets via a compression approach.
Sanjeev Arora, Rong Ge, Behnam Neyshabur and Yi Zhang.

Two methods, both based on compressing a network by representing its weight matrices with fewer parameters.

1. Define *compressibility* of a function f via G, a (finite) set of functions, and derive a generalisation bound that relates the losses of f and G.
    - f: a neural network; G: class of neural networks that have less parameters and that can approximate f.
    - Results in the same bound as in [Neyshabur et al., 2017].
2. A different compression framework based on random projections, together with noise stability properties of the network, gives tighter generalisation bounds than the first method.
    - Can be adapted to convolutional neural networks.

Compressed networks: Method 1

Compression framework Define a notion of *compressibility* with respect to an approximation parameter $\gamma > 0$ and sample S.

Compressed networks: Method 1

Compression framework Define a notion of *compressibility* with respect to an approximation parameter $\gamma > 0$ and sample S.

Definition

- $f : \mathbb{R}^d \to \mathbb{R}^k$.
- $G_{\mathcal{A}} := \left\{ g_A : \mathbb{R}^d \to \mathbb{R}^k \mid A \in \mathcal{A} \right\}$, where A is a set of parameters.

Compressed networks: Method 1

Compression framework Define a notion of *compressibility* with respect to an approximation parameter $\gamma > 0$ and sample S.

Definition

- $f : \mathbb{R}^d \to \mathbb{R}^k$.
- $G_{\mathcal{A}} := \{ g_A : \mathbb{R}^d \to \mathbb{R}^k \mid A \in \mathcal{A} \}$, where A is a set of parameters.

We say that f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for all x in the sample S,

$$\| f(x) - g_A(x) \|_\infty \leq \gamma \ .$$

Compressed networks: Method 1

Definition

f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for all x in the sample S,

$$\|f(x) - g_A(x)\|_\infty \leq \gamma \ .$$

Compressed networks: Method 1

Definition

f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for all x in the sample S,

$$\|f(x) - g_A(x)\|_\infty \leq \gamma \ .$$

Theorem

Let $G_{\mathcal{A}} := \{g_A \mid A \in \mathcal{A}\}$, where A is a set of q parameters, each of which can have at most r discrete values.

Compressed networks: Method 1

Definition

f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for all x in the sample S,

$$\|f(x) - g_A(x)\|_\infty \leq \gamma \ .$$

Theorem

Let $G_{\mathcal{A}} := \{g_A \mid A \in \mathcal{A}\}$, where A is a set of q parameters, each of which can have at most r discrete values. Let S be a training set of m samples. For any margin $\gamma > 0$, if f is (γ, S)-compressible via $G_{\mathcal{A}}$, then there exists $A \in \mathcal{A}$ such that w.h.p. over S,

$$L_0(g_A) \leq \widehat{L}_\gamma(f) + \mathcal{O}\left(\sqrt{\frac{q \log r}{m}}\right) \ .$$

Compressed networks: Method 1

How do we compress a neural network and apply this theorem?

Compressed networks: Method 1

How do we compress a neural network and apply this theorem?
Compression scheme:

- Low-rank approximation for the weight matrices
  $\Longrightarrow$ the weight matrices can be represented using less parameters.

Compressed networks: Method 1

How do we compress a neural network and apply this theorem?
Compression scheme:

- Low-rank approximation for the weight matrices
  $\Longrightarrow$ the weight matrices can be represented using less parameters.
  - The choice of the reconstruction error ensures that the compressed network approximates the original network.

Compressed networks: Method 1

How do we compress a neural network and apply this theorem?
Compression scheme:

- Low-rank approximation for the weight matrices
  $\implies$ the weight matrices can be represented using less parameters.
  - The choice of the reconstruction error ensures that the compressed network approximates the original network.
- Discretise weights and define the class $G_{\mathcal{A}}$.

Compressed networks: Method 1

How do we compress a neural network and apply this theorem?
Compression scheme:

- Low-rank approximation for the weight matrices
  $\Longrightarrow$ the weight matrices can be represented using less parameters.
  - The choice of the reconstruction error ensures that the compressed network approximates the original network.
- Discretise weights and define the class $G_{\mathcal{A}}$.

Compressed networks: Method 1

How do we compress a neural network and apply this theorem?
Compression scheme:

- Low-rank approximation for the weight matrices
  $\Longrightarrow$ the weight matrices can be represented using less parameters.
  - The choice of the reconstruction error ensures that the compressed network approximates the original network.
- Discretise weights and define the class $G_{\mathcal{A}}$.

Theorem

Let $S \sim D^m$ and let $\gamma > 0$. A neural network of depth L with linear transformations $A_1, \ldots, A_L$. Then with high probability over S,

$$L_0(f) \leq \widehat{L}_\gamma(f) + \tilde{\mathcal{O}}\left(\sqrt{\frac{hL^2 \max_{x\in S}\|x\| \prod_{i=1}^L \|A_i\|_2^2 \sum_{i=1}^L \frac{\|A_i\|_F^2}{\|A_i\|_2^2}}{\gamma^2 m}}\right).$$

Compressed networks: Method 1

Theorem

Let $S \sim D^m$ and let $\gamma > 0$. A neural network of depth L with linear transformations $A_1, \ldots, A_L$. Then with high probability over S,

$$L_0(f) \leq \widehat{L}_\gamma(f) + \tilde{\mathcal{O}}\left(\sqrt{\frac{hL^2 \max_{x \in S} \|x\| \prod_{i=1}^{L} \|A_i\|_2^2 \sum_{i=1}^{L} \frac{\|A_i\|_F^2}{\|A_i\|_2^2}}{\gamma^2 m}}\right).$$

Some remarks:

- γ is used both as the margin for the loss, and the approximation parameter for compressibility.
- Although the framework bounds the expected loss of the *compressed network* g_A by the empiricial loss of the *original network* f, one can show that g_A approximates f on the *whole input space* and not just S. This thus gives a generalisation bound for f.

Compressed networks: Method 2

Two main ideas:

1. Define neural network properties, which are related to *noise stability* and *empirical observations*.

Compressed networks: Method 2

Two main ideas:

1. Define neural network properties, which are related to *noise stability* and *empirical observations*.
2. Randomly project the linear transformations onto lower-dimensional subspace (Johnson-Lindenstrauss transformation).

Compressed networks: Method 2

Two main ideas:

1. Define neural network properties, which are related to *noise stability* and *empirical observations*.
2. Randomly project the linear transformations onto lower-dimensional subspace (Johnson-Lindenstrauss transformation).
3. Use (1) and (2) to derive a tighter generalisation bound.

Compressed networks: Method 2

Examples of neural network properties:

- μ_i (layer cushion) : $\approx$ reciprocal of noise sensitivity.

Compressed networks: Method 2

Examples of neural network properties:

- μ_i (layer cushion) : $\approx$ reciprocal of noise sensitivity.
- c (activation contraction): relates to the percentage of ReLU units that are activated (in practice $\approx 1/2$).

Compressed networks: Method 2

Examples of neural network properties:

- μ_i (layer cushion) : $\approx$ reciprocal of noise sensitivity.
- c (activation contraction): relates to the percentage of ReLU units that are activated (in practice $\approx 1/2$).
- *These properties relate to noise sensitivity and empirical observations.*

Compressed networks: Method 2

General idea for the random projections:

- Perturb the weight matrices by random projection on a lower-dimensional subspace.

Compressed networks: Method 2

General idea for the random projections:

- Perturb the weight matrices by random projection on a lower-dimensional subspace.
- Prove that the output of the network isn't changed much.

Compressed networks: Method 2

General idea for the random projections:

- Perturb the weight matrices by random projection on a lower-dimensional subspace.
- Prove that the output of the network isn't changed much.
  - Result of the noise stability properties mentioned on the previous slide, and the Johnson-Lindenstrauss transformation.

Compressed networks: Method 2

General idea for the random projections:

- Perturb the weight matrices by random projection on a lower-dimensional subspace.
- Prove that the output of the network isn't changed much.
    - Result of the noise stability properties mentioned on the previous slide, and the Johnson-Lindenstrauss transformation.
- Can represent the network with much fewer parameters.

Compressed networks: Method 2

General idea for the random projections:

- Perturb the weight matrices by random projection on a lower-dimensional subspace.
- Prove that the output of the network isn't changed much.
    - Result of the noise stability properties mentioned on the previous slide, and the Johnson-Lindenstrauss transformation.
- Can represent the network with much fewer parameters.
- Use standard tools to get a generalisation bound:

Compressed networks: Method 2

General idea for the random projections:

- Perturb the weight matrices by random projection on a lower-dimensional subspace.
- Prove that the output of the network isn't changed much.
  - Result of the noise stability properties mentioned on the previous slide, and the Johnson-Lindenstrauss transformation.
- Can represent the network with much fewer parameters.
- Use standard tools to get a generalisation bound:
  - Dudley entropy integral to bound the empirical Rademacher complexity of the margin loss function on the compressed network.

Compressed networks: Method 2

Theorem

For any fully connected network f_A with $\rho_\delta \geq 3L$, and any margin $\gamma > 0$, the random projection algorithm generates weights $\tilde{A}$ s.t. with high probability over the training set,

$$L_0(f_{\tilde{A}}) \leq \widehat{L}_\gamma(f_A) + \tilde{O}\left(\sqrt{\frac{c^2 L^2 \max_{x \in S} \|f_A(x)\|_2^2 \sum_{i=1}^{L} \frac{1}{\mu_i^2 \mu_{i\rightarrow}^2}}{\gamma^2 m}}\right).$$

Overview

Conclusion

- Two different frameworks to compress neural networks and get better generalisation bounds.

Conclusion

- Two different frameworks to compress neural networks and get better generalisation bounds.
- One was able to recover the result from [Neyshabur et al., 2017].

Conclusion

- Two different frameworks to compress neural networks and get better generalisation bounds.
- One was able to recover the result from [Neyshabur et al., 2017].
- The other is a tighter bound and performs well in practice.

Conclusion

- Two different frameworks to compress neural networks and get better generalisation bounds.
- One was able to recover the result from [Neyshabur et al., 2017].
- The other is a tighter bound and performs well in practice.
- Can be extended to convolutional neural networks.

Research Directions

- Other compression approaches: weight pruning, computational unit pruning, etc.

Research Directions

- Other compression approaches: weight pruning, computational unit pruning, etc.
- Current and future work:

Research Directions

- Other compression approaches: weight pruning, computational unit pruning, etc.
- Current and future work:
    - Can we get better bounds? Currently: not useful in practice.

Research Directions

- Other compression approaches: weight pruning, computational unit pruning, etc.
- Current and future work:
  - Can we get better bounds? Currently: not useful in practice.
  - Can we develop notion and guarantees for adversarial generalisation? [Yin et al., 2018, Cullina et al., 2018]

Research Directions

- Other compression approaches: weight pruning, computational unit pruning, etc.
- Current and future work:
  - Can we get better bounds? Currently: not useful in practice.
  - Can we develop notion and guarantees for adversarial generalisation? [Yin et al., 2018, Cullina et al., 2018]
  - Can *algorithmic stability* offer better bounds and explanations? [Bousquet and Elisseeff, 2002],[Hardt et al., 2016].

References I

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018).
Stronger generalization bounds for deep nets via a compression approach.
arXiv preprint arXiv:1802.05296.

Bartlett, P. L. (1997).
For valid generalization the size of the weights is more important than the size of the network.
In *Advances in neural information processing systems*, pages 134–140.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017a).
Spectrally-normalized margin bounds for neural networks.
In *Advances in Neural Information Processing Systems*, pages 6240–6249.

References II

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2017b).
Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks.
arXiv preprint arXiv:1703.02930.

Bartlett, P. L., Maiorov, V., and Meir, R. (1999).
Almost linear vc dimension bounds for piecewise polynomial networks.
In *Advances in Neural Information Processing Systems*, pages 190–196.

Bousquet, O. and Elisseeff, A. (2002).
Stability and generalization.
Journal of machine learning research, 2(Mar):499–526.

Cullina, D., Bhagoji, A. N., and Mittal, P. (2018).
Pac-learning in the presence of evasion adversaries.
Advances in Neural Information Processing Systems.

References III

- Goldberg, P. W. and Jerrum, M. R. (1995).
  Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers.
  Machine Learning, 18(2-3):131–148.

- Hardt, M., Recht, B., and Singer, Y. (2016).
  Train faster, generalize better: stability of stochastic gradient descent.
  In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 1225–1234. JMLR. org.

- Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017).
  A pac-bayesian approach to spectrally-normalized margin bounds for neural networks.
  arXiv preprint arXiv:1707.09564.

References IV

Yin, D., Ramchandran, K., and Bartlett, P. (2018).
Rademacher complexity for adversarially robust generalization.
arXiv preprint arXiv:1810.11914.