Machine Learning Theory 2022
Lecture 7

Tim van Erven

Download these slides now from elo.mastermath.nl!

- Complexity of classification vs regression
- Neural networks
- Bias-variance trade-off and double descent
- Towards an explanation

homework roulette
in the break

Binary Classification

- Sample complexity of agnostic PAC-learnability **determined by VC-dimension**:

$$m_{\mathcal{H}}(\epsilon, \delta) \approx \frac{\text{VCdim}(\mathcal{H}) + \ln(1/\delta)}{\epsilon^2}$$

- For some (not all!) hypothesis classes, $\text{VCdim}(\mathcal{H}) =$ **nr. of parameters**:
  - **Linear predictors**: $\mathcal{H} = \{h_{\boldsymbol{w}}(\boldsymbol{X}) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{X} \rangle) : \boldsymbol{w} \in \mathbb{R}^d\}$
  - Axis-aligned rectangles
  - ...

Regression

$$\mathcal{H}_1^B = \{h_{\boldsymbol{w}}(\boldsymbol{X}) = \langle \boldsymbol{w}, \boldsymbol{X} \rangle : \boldsymbol{w} \in \mathbb{R}^d, \|\boldsymbol{w}\|_1 \leq B\}.$$

Theorem (Lasso Estimator)

Consider linear regression with $\ell(h, \boldsymbol{X}, Y) = \frac{1}{2}(Y - \langle \boldsymbol{w}, \boldsymbol{X} \rangle)^2$ *for* $\boldsymbol{X} \in [-1, +1]^d$, $Y \in [-1, +1]$.
Then $\mathcal{H}_1^B$ *is agnostically PAC-learnable by ERM with sample complexity*

$$m(\epsilon, \delta) \leq c_B \frac{\ln(2d) + \ln(2/\delta)}{\epsilon^2}$$

for some constant $c_B > 0$ *that depends only on* B.

General pattern for regression tasks:

- **Complexity** of hypothesis class **depends on bound** B on norm $\|\boldsymbol{w}\|$ of parameters
- (and sometimes weakly on number of parameters d)

Difference between Linear Regression and Linear Classification

Linear Classification:

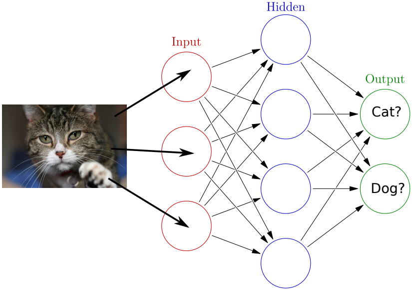
- **Not Lipschitz in w**: tiny change in $\boldsymbol{w}$ can flip prediction $h_{\boldsymbol{w}}(\boldsymbol{X})$
- Measure of complexity: **number of parameters** d

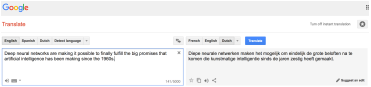
Linear Regression:

- **Lipschitz in w**: tiny change in $\boldsymbol{w}$ implies tiny change in $h_{\boldsymbol{w}}(\boldsymbol{X})$
- Main measure of complexity: **norm constraint** B

Deep Learning / Neural Networks

(Deep) Neural Networks



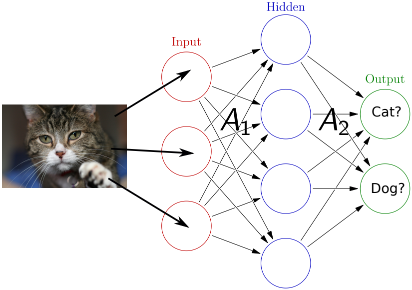


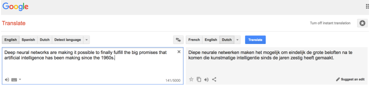
Machine translation

Speech recognition

Self-driving cars

(Deep) Neural Networks





Machine translation

Speech recognition

Self-driving cars

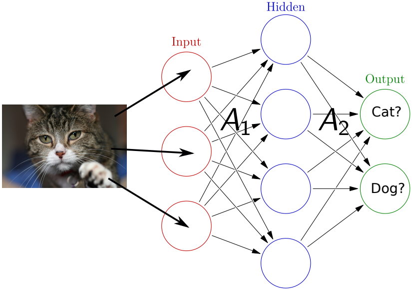
Class of **non-convex** functions parametrized by matrices $\boldsymbol{w} = (A_1, \ldots, A_m)$:

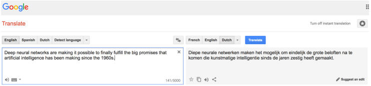
Fully connected network: $$\mathcal{H} = \{h_{\boldsymbol{w}}(\boldsymbol{X}) = A_m \sigma A_{m-1} \cdots \sigma A_1 \boldsymbol{X} : \boldsymbol{w} \in \mathcal{W}\},$$

with **activation function** $\sigma(z)$ applied component-wise to vectors. E.g.

- Rectified linear unit (ReLU): $\sigma(z) = \max\{0, z\}$
- Sigmoid: $\sigma(z) = 1/(1 + e^{-z})$

(Deep) Neural Networks





Machine translation

Speech recognition

Self-driving cars

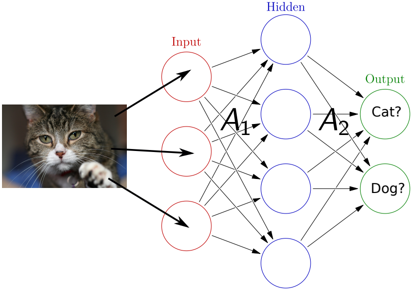
Class of **non-convex** functions parametrized by matrices $\boldsymbol{w} = (A_1, \dots, A_m) \in \mathbb{R}^d$:

Fully connected network: $\mathcal{H} = \{h_{\boldsymbol{w}}(\boldsymbol{X}) = A_m \sigma A_{m-1} \cdots \sigma A_1 \boldsymbol{X} : \boldsymbol{w} \in \mathbb{R}^d\}$,

with **activation function** $\sigma(z)$ applied component-wise to vectors. E.g.

- Rectified linear unit (ReLU): $\sigma(z) = \max\{0, z\}$
- Sigmoid: $\sigma(z) = 1/(1 + e^{-z})$

(Deep) Neural Networks



VC-dimension dependence on nr. of parameters d:

ReLU:	$\tilde{\Theta}(d)$	[Bartlett et al., 2017]
Sigmoid:	$\Theta(d^2)$	[Anthony and Bartlett, 1999]

Conclusion: need sample size $m \gg$ **nr. of parameters** to learn

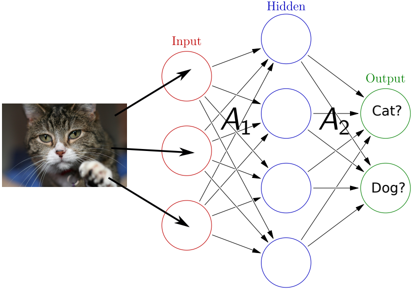
Class of **non-convex** functions parametrized by matrices $\boldsymbol{w} = (A_1, \ldots, A_m) \in \mathbb{R}^d$:

Fully connected network: $\mathcal{H} = \{h_{\boldsymbol{w}}(\boldsymbol{X}) = A_m \sigma A_{m-1} \cdots \sigma A_1 \boldsymbol{X} : \boldsymbol{w} \in \mathbb{R}^d\}$,

with **activation function** $\sigma(z)$ applied component-wise to vectors. E.g.

- Rectified linear unit (ReLU): $\sigma(z) = \max\{0, z\}$
- Sigmoid: $\sigma(z) = 1/(1 + e^{-z})$

(Deep) Neural Networks



VC-dimension dependence on nr. of parameters d:

ReLU: $\tilde{\Theta}(d)$ [Bartlett et al., 2017]
Sigmoid: $\Theta(d^2)$ [Anthony and Bartlett, 1999]

Conclusion: need sample size $m \gg$ **nr. of parameters** to learn

Class of **non-convex** functions parametrized by matrices $\boldsymbol{w} = (A_1, \ldots, A_m) \in \mathbb{R}^d$:

Fully c … $X : \boldsymbol{w} \in \mathbb{R}^d\}$,

with **a** … E.g.

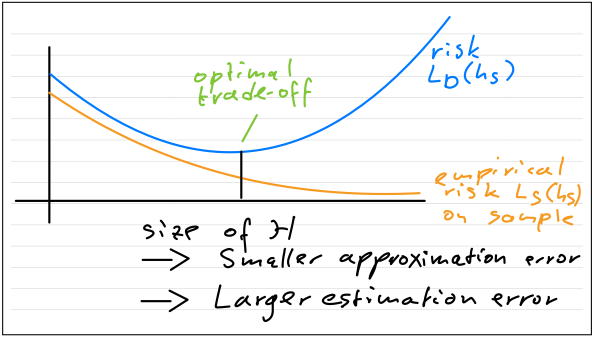
- R…
- Sigmoid: $\sigma(z) = 1/(1 + e^{\ldots})$

A First Glimpse of a Mystery:

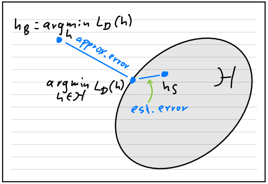
- In theory: need sample size $m \gg$ nr. parameters d
- In practise: sample size $m \ll$ nr. parameters d

Bias-Variance Trade-off and the Double Descent Phenomenon

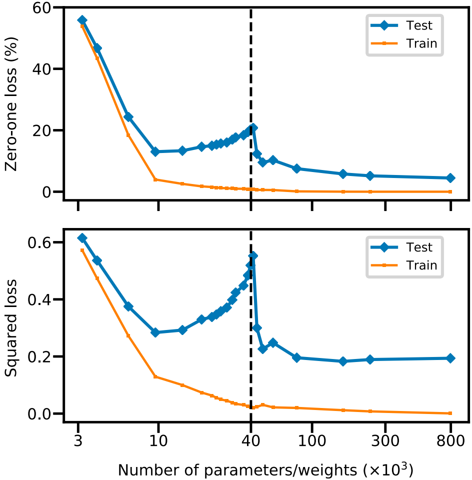
Classical Bias-Variance Trade-off



- Approximation error (bias): $\inf_{h \in \mathcal{H}} L_\mathcal{D}(h) - \inf_h L_\mathcal{D}(h)$
- Estimation error (variance): $L_\mathcal{D}(h_S) - \inf_{h \in \mathcal{H}} L_\mathcal{D}(h)$



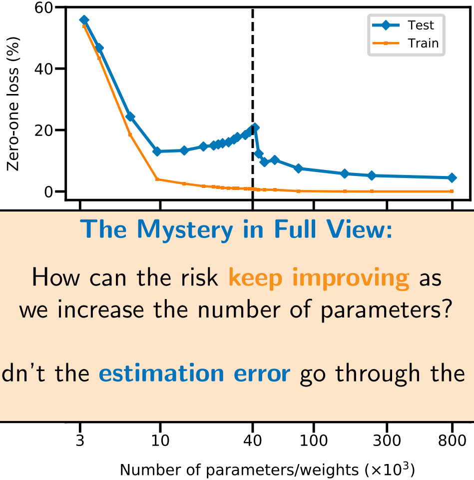
Double Descent Phenomenon



[Belkin, Hsu, Ma, Mandal, 2019]

- Varying the number of hidden units in a two-layer neural network
- Classification: MNIST hand-written digits data with 10 classes

Double Descent Phenomenon



The Mystery in Full View:

How can the risk **keep improving** as
we increase the number of parameters?

Shouldn't the **estimation error** go through the roof?

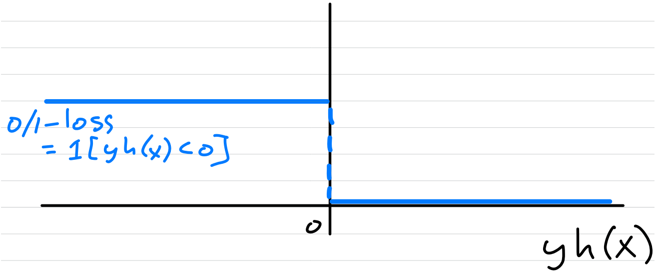
[Belkin, Hsu, Ma, Mandal, 2019]

- Varying the number of hidden units in a two-layer neural network
- Classification: MNIST hand-written digits data with 10 classes

Towards an Explanation

1. **Large margins turn classification into regression**
2. Explaining double descent

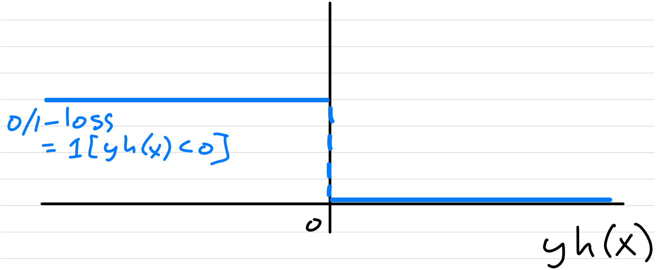
Classifiers as Real-valued Functions



NB Real-valued classifiers. E.g. $h_{\boldsymbol{w}}(\boldsymbol{X}) = \langle \boldsymbol{w}, \boldsymbol{X} \rangle$.
Prediction is sign($h(\boldsymbol{X})$)

- **Margin** $= Yh(\boldsymbol{X})$, where $Y \in \{-1, +1\}$
- Larger margin > 0: more confident correct classification

Classifiers as Real-valued Functions



NB Real-valued classifiers. E.g. $h_{\boldsymbol{w}}(\boldsymbol{X}) = \langle \boldsymbol{w}, \boldsymbol{X} \rangle$.
Prediction is $\text{sign}(h(\boldsymbol{X}))$

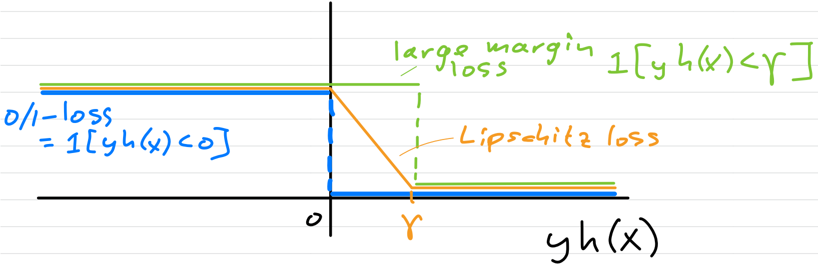
- **Margin** $= Yh(\boldsymbol{X})$, where $Y \in \{-1, +1\}$
- Larger margin > 0: more confident correct classification
- Common loss functions encourage finding large margin solutions:

$$\text{logistic loss: } \ln(1 + e^{-Yh(\boldsymbol{X})})$$

$$\text{squared loss for classification: } (Y - h(\boldsymbol{X}))^2 = (1 - Yh(X))^2$$

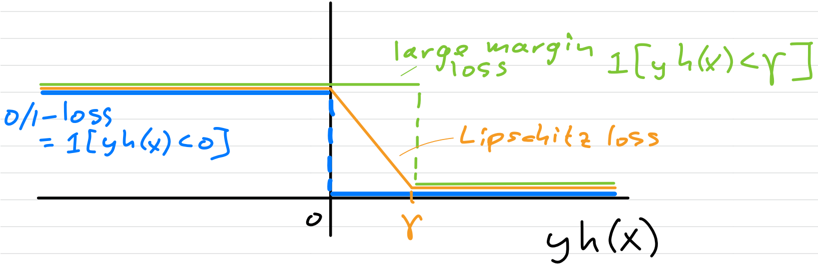
Large Margins 1

[Anthony and Bartlett, 1999]



0/1-loss $\leq$ γ-Lipschitz loss $\leq$ γ-large margin loss

Large Margins 1 [Anthony and Bartlett, 1999]



0/1-loss $\leq$ γ-Lipschitz loss $\leq$ γ-large margin loss

$$L_{\mathcal{D}}^{0/1}(h_S) \leq L_{\mathcal{D}}^{\text{Lipschitz}}(h_S)$$

$$\leq L_S^{\text{Lipschitz}}(h_S) + 2\,\mathbb{E}[\mathcal{R}(\ell^{\text{Lipschitz}}, \mathcal{H}, S)] + \sqrt{\frac{\ln(4/\delta)}{2m}} \quad \text{w.p.} \geq 1 - \delta$$

$$\leq L_S^{\text{large margin}}(h_S) + 2\,\mathbb{E}[\mathcal{R}(\ell^{\text{Lipschitz}}, \mathcal{H}, S)] + \sqrt{\frac{\ln(4/\delta)}{2m}}$$

Large Margins 2 [Anthony and Bartlett, 1999]

Theorem

Let $h_S \in \mathcal{H}$ be the output of a learning algorithm. Then, with probability at least $1 - \delta$,

$$L_{\mathcal{D}}^{0/1}(h_S) \leq L_S^{\gamma\text{-large margin}}(h_S) + 2\,\mathbb{E}[\mathcal{R}(\ell^{\gamma\text{-Lipschitz}}, \mathcal{H}, S)] + \sqrt{\frac{\ln(4/\delta)}{2m}}.$$

Large Margins 2 [Anthony and Bartlett, 1999]

Theorem

Let $h_S \in \mathcal{H}$ be the output of a learning algorithm. Then, with probability at least $1-\delta$,

$$L_{\mathcal{D}}^{0/1}(h_S) \leq L_S^{\gamma\text{-large margin}}(h_S) + 2\,\mathbb{E}[\mathcal{R}(\ell^{\gamma\text{-Lipschitz}}, \mathcal{H}, S)] + \sqrt{\frac{\ln(4/\delta)}{2m}}.$$

1. If h_S has **margin** $\geq \gamma$ on (most of) S, then $L_S^{\gamma\text{-large margin}}(h_S)$ is small

Large Margins 2 [Anthony and Bartlett, 1999]

Theorem

Let $h_S \in \mathcal{H}$ be the output of a learning algorithm. Then, with probability at least $1-\delta$,

$$L_{\mathcal{D}}^{0/1}(h_S) \leq L_S^{\gamma\text{-large margin}}(h_S) + 2\,\mathbb{E}[\mathcal{R}(\ell^{\gamma\text{-Lipschitz}}, \mathcal{H}, S)] + \sqrt{\frac{\ln(4/\delta)}{2m}}.$$

1. If h_S has **margin $\geq \gamma$** on (most of) S, then $L_S^{\gamma\text{-large margin}}(h_S)$ is small
2. Lipschitz loss is $\frac{1}{\gamma}$-Lipschitz, so can apply **contraction lemma**:

$$\mathcal{R}(\ell^{\text{Lipschitz}}, \mathcal{H}, S) \leq \frac{1}{\gamma}\mathcal{R}\Big(\big\{(h(\boldsymbol{X}_1), \ldots, h(\boldsymbol{X}_m)) : h \in \mathcal{H}\big\}\Big)$$

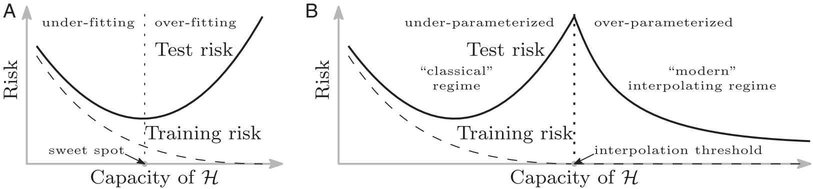
- So small changes in h imply small changes in loss
- We have **turned the classification problem into a regression task**!
- Complexity of $\mathcal{H}$ can be controlled by some norm on h.

Towards an Explanation

1. Large margins turn classification into regression
2. **Explaining double descent**

A Potential Explanation

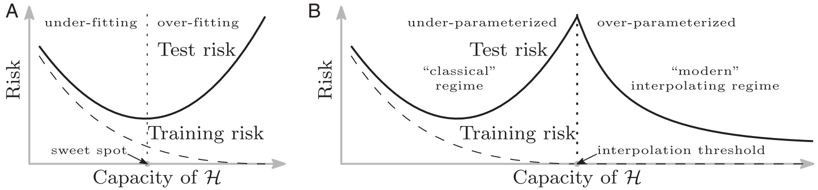
[Belkin, Hsu, Ma, Mandal, 2019]



[Belkin et al., 2019] Double Descent

A Potential Explanation

[Belkin, Hsu, Ma, Mandal, 2019]



[Belkin et al., 2019] Double Descent

Proposed explanation: suppose learning alg roughly behaves as

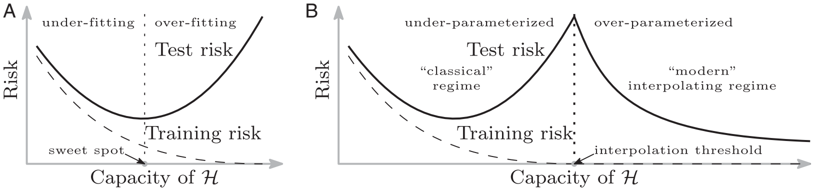
among **ERM solutions** $h_S \in \arg\min_{h \in \mathcal{H}} L_S(h)$

choose solution with **smallest norm** $\|h_S\|_{??}$

Below int. threshold: ERM unique $\rightarrow$ classical bias-variance trade-off
Above int. threshold: larger $\mathcal{H}$ $\rightarrow$ more ERM solutions $\rightarrow$ smaller $\|h_S\|_{??}$

A Potential Explanation

[Belkin, Hsu, Ma, Mandal, 2019]



[Belkin et al., 2019] Double Descent

Proposed explanation: suppose learning alg roughly behaves as

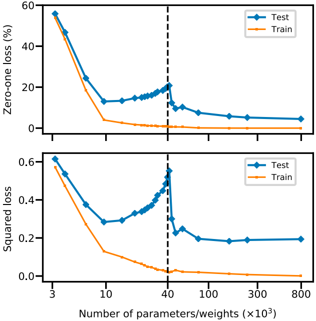
among **ERM solutions** $h_S \in \arg\min_{h \in \mathcal{H}} L_S(h)$

choose solution with **smallest norm** $\|h_S\|_{??}$

Below int. threshold: ERM unique $\rightarrow$ classical bias-variance trade-off
Above int. threshold: larger $\mathcal{H}$ $\rightarrow$ more ERM solutions $\rightarrow$ smaller $\|h_S\|_{??}$

- L_S for e.g. logistic or squared loss (encouraging large margin)
- Different norm depending on manifestation of double descent

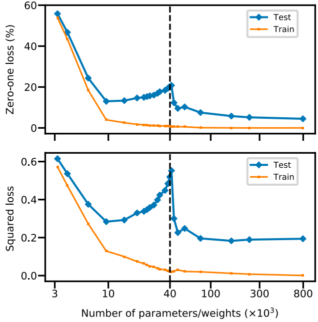
Double Descent for Neural Networks Again



[Belkin, Hsu, Ma, Mandal, 2019]

- Classification: CIFAR-10 32x32 images from 10 classes, e.g. airplanes, cats, dogs

Double Descent for Neural Networks Again



[Belkin, Hsu, Ma, Mandal, 2019]

- Classification: CIFAR-10 32x32 images from 10 classes, e.g. airplanes, cats, dogs

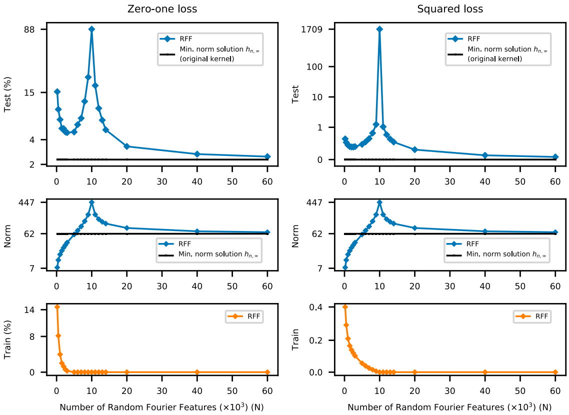
Which norm $\|h_S\|_{??}$?

Implicitly **induced by optimization algorithm!**

- Exist proposals in the literature to characterize norm.
  E.g. using neural tangent kernel [Jacot, Gabriel, Hongler, 2018]

Double Descent: Not Just for Neural Networks

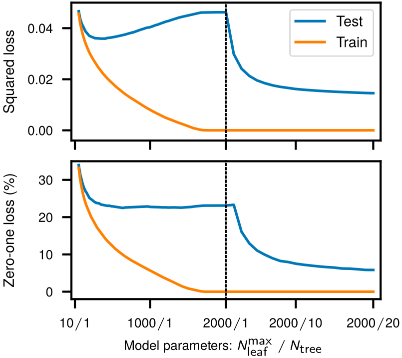
[Belkin et al., 2019] reproduce double descent phenomenon on e.g. MNIST:



Random Fourier features: linear model over N randomly generated basis functions that approximate a certain (reproducing kernel) Hilbert space as $N \to \infty$

Double Descent: Not Just for Neural Networks

[Belkin et al., 2019] reproduce double descent phenomenon on e.g. MNIST:



Random forests: ensembles of decision trees

- Complexity controlled by number of leaves per tree and by number of trees

Conclusion

- Exciting new attempts to understand the **double descent phenomenon** observed in deep learning, random Fourier features, random forests, etc.
- Using tools like **Rademacher complexity** that you have learned in this course.
- Whether proposed explanation holds up and can be fully formalized remains to be seen...
- In any case, it has already **radically changed our view** of the classical bias-variance trade-off.