

# Machine Learning Theory 2026

## Lecture 7

**Tim van Erven**

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

# 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,  $\text{VCdim}(\mathcal{H}) = \text{nr. of parameters}$ :
  - ▶ **Linear predictors**:  $\mathcal{H} = \{h_{\mathbf{w}}(\mathbf{X}) = \text{sign}(\langle \mathbf{w}, \mathbf{X} \rangle) : \mathbf{w} \in \mathbb{R}^d\}$
  - ▶ Axis-aligned rectangles
  - ▶ ...

# Regression

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

## Theorem (Lasso Estimator)

Consider linear regression with  $\ell(h, \mathbf{X}, Y) = \frac{1}{2}(Y - \langle \mathbf{w}, \mathbf{X} \rangle)^2$  for  $\mathbf{X} \in [-1, +1]^d$ ,  $Y \in [-1, +1]$ .

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  $\|\mathbf{w}\|$  of parameters
- ▶ (and sometimes weakly on number of parameters  $d$ )

# Difference between Linear Regression and Linear Classification

## Linear Classification:

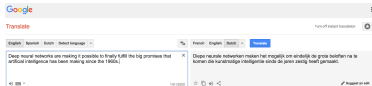
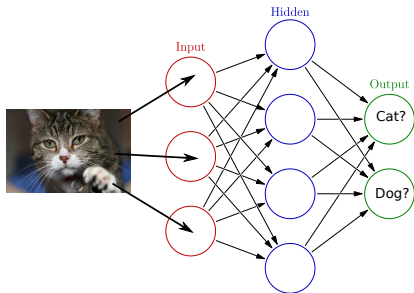
- ▶ **Not Lipschitz in  $w$** : tiny change in  $w$  can flip prediction  $h_w(\mathbf{X})$
- ▶ Measure of complexity: **number of parameters  $d$**

## Linear Regression:

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

# Deep Learning / Neural Networks

# (Fully Connected) Neural Networks



Machine translation

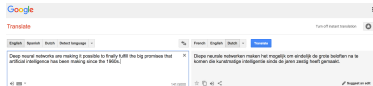
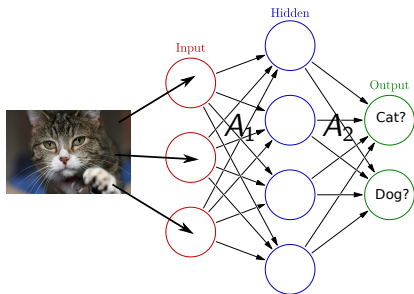


Speech recognition



Self-driving cars

# (Fully Connected) Neural Networks



Machine translation



Speech recognition



Self-driving cars

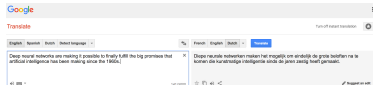
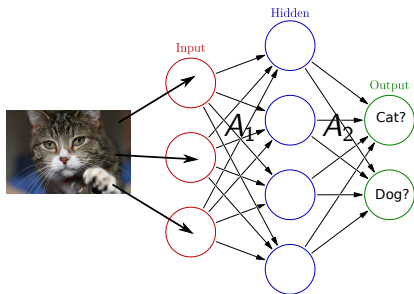
Class of **non-convex** functions parametrized by matrices  $w = (A_1, \dots, A_m)$ :

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

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

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

# (Fully Connected) Neural Networks



Machine translation



Speech recognition



Self-driving cars

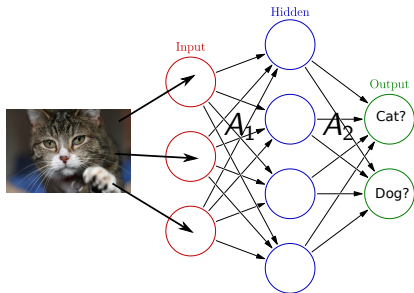
Class of **non-convex** functions parametrized by matrices  $\mathbf{w} = (A_1, \dots, A_m) \in \mathbb{R}^d$ :

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

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

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

# (Fully Connected) 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

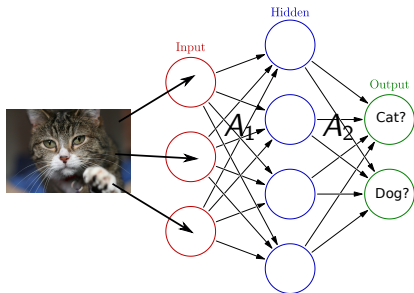
Class of **non-convex** functions parametrized by matrices  
 $\mathbf{w} = (A_1, \dots, A_m) \in \mathbb{R}^d$ :

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

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

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

# (Fully Connected) 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  $w = (A_1, \dots, A_m) \in \mathbb{R}^d$ :

Fully c

## A First Glimpse of a Mystery:

with a

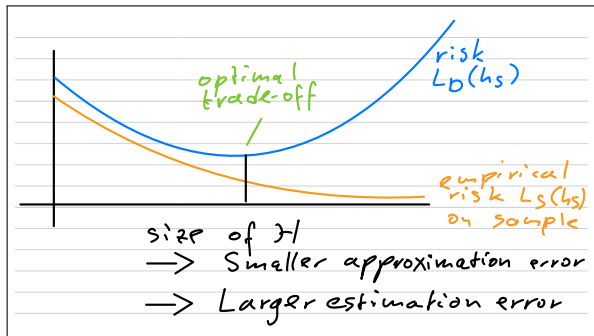
- ▶ In theory: need sample size  $m \gg$  nr. parameters  $d$
- ▶ In practise: sample size  $m \ll$  nr. parameters  $d$
- ▶ Sigmoid:  $\sigma(z) = 1/(1 + e^{-z})$

$\mathcal{X} : w \in \mathbb{R}^d$

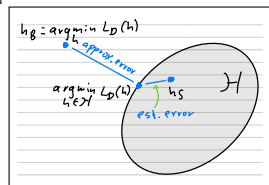
E.g.

# Bias-Variance Trade-off and the Double Descent Phenomenon

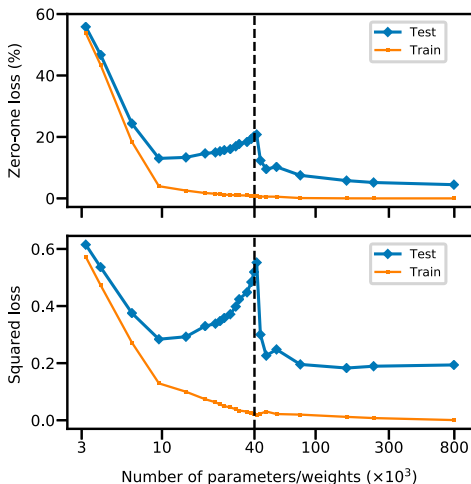
# Classical Bias-Variance Trade-off



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



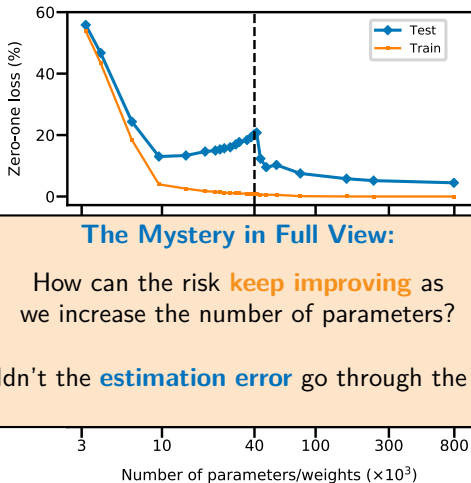
# Double Descent Phenomenon



[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?

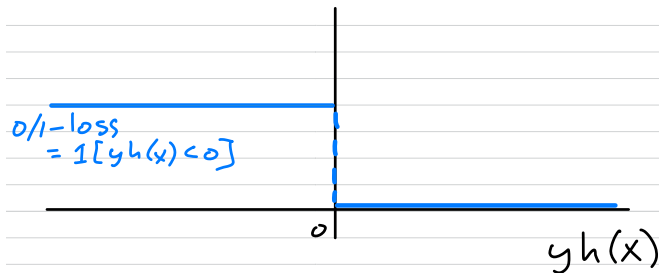
[Belkin, Hsu, Ma, Mandal, 2019]

- ▶ 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

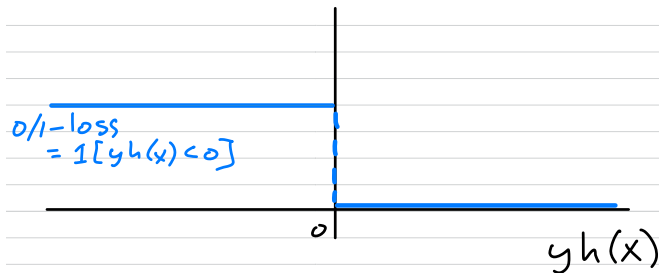
# Classifiers as Real-valued Functions



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

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

# Classifiers as Real-valued Functions



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

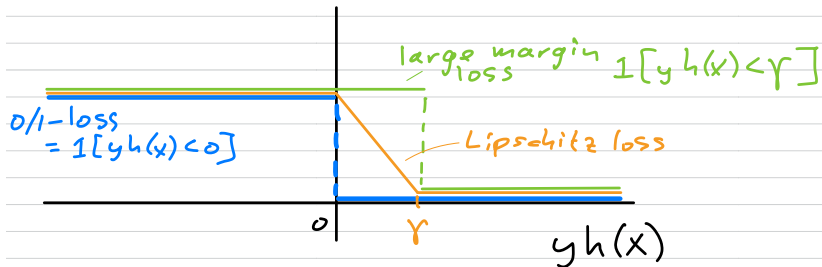
- ▶ **Margin** =  $Yh(\mathbf{X})$ , where  $Y \in \{-1, +1\}$
- ▶ Larger margin  $> 0$ : more confident correct classification
- ▶ Common loss functions encourage finding large margin solutions:

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

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

# Large Margins 1

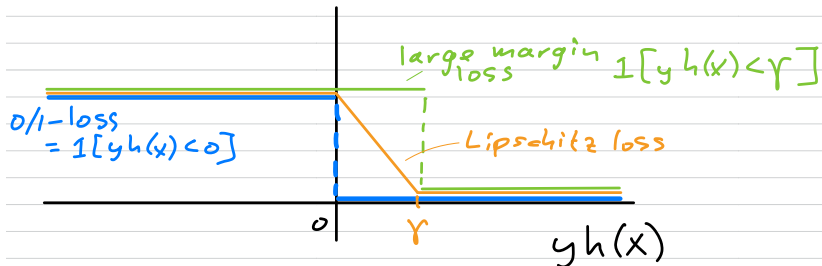
[Anthony and Bartlett, 1999]



$$0/1\text{-loss} \leq \gamma\text{-Lipschitz loss} \leq \gamma\text{-large margin loss}$$

# Large Margins 1

[Anthony and Bartlett, 1999]



$$0/1\text{-loss} \leq \gamma\text{-Lipschitz loss} \leq \gamma\text{-large margin loss}$$

$$L_D^{0/1}(h_S) \leq L_D^{\text{Lipschitz}}(h_S)$$

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

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

## Theorem

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

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

## 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

## 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**:

$$\mathcal{R}(\ell^{\text{Lipschitz}}, \mathcal{H}, S) \leq \frac{1}{\gamma} \mathcal{R}\left(\{(h(\mathbf{X}_1), \dots, h(\mathbf{X}_m)) : h \in \mathcal{H}\}\right)$$

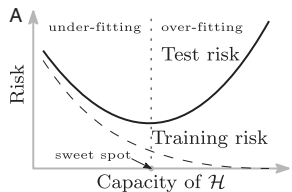
- ▶ So small changes in  $h$  imply small changes in loss
- ▶ We have **turned the classification problem into a regression task!**
- ▶ Complexity of  $\mathcal{H}$  can be controlled by some norm on  $h$ .

## Towards an Explanation

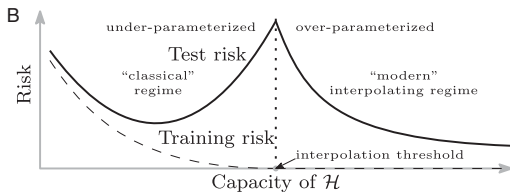
1. Large margins turn classification into regression
2. **Explaining double descent**

# A Potential Explanation

[Belkin, Hsu, Ma, Mandal, 2019]



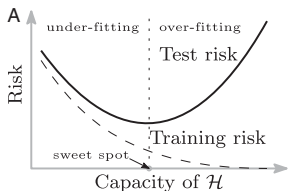
[Belkin et al., 2019]



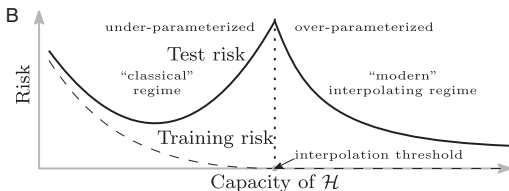
Double Descent

# 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**  $h_S \in \arg \min_{h \in \mathcal{H}} L_S(h)$

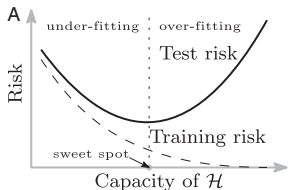
choose solution with **smallest norm**  $\|h_S\|??$

Below int. threshold: ERM unique  $\rightarrow$  classical bias-variance trade-off

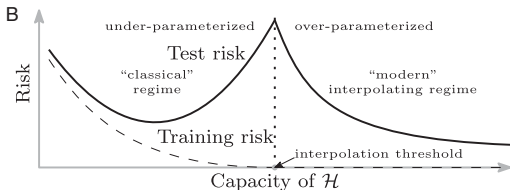
Above int. threshold: larger  $\mathcal{H} \rightarrow$  more ERM solutions  $\rightarrow$  smaller  $\|h_S\|??$

# 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**  $h_S \in \arg \min_{h \in \mathcal{H}} L_S(h)$

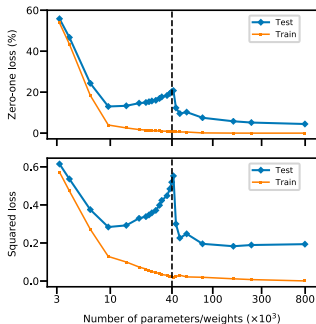
choose solution with **smallest norm**  $\|h_S\|_{??}$

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

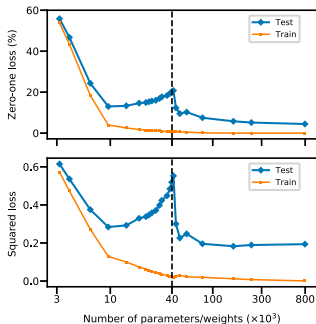
# Double Descent for Neural Networks Again



[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

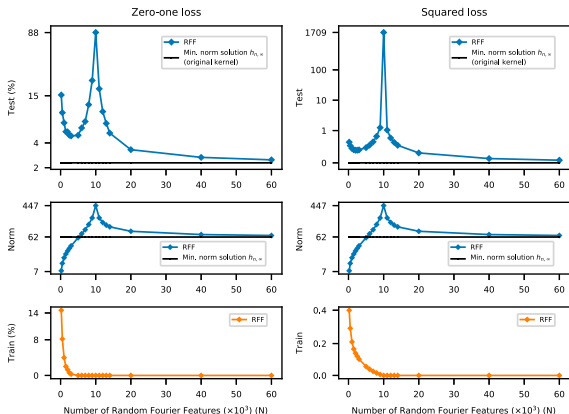
Which norm  $\|h_S\|_{??}$ ?

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:

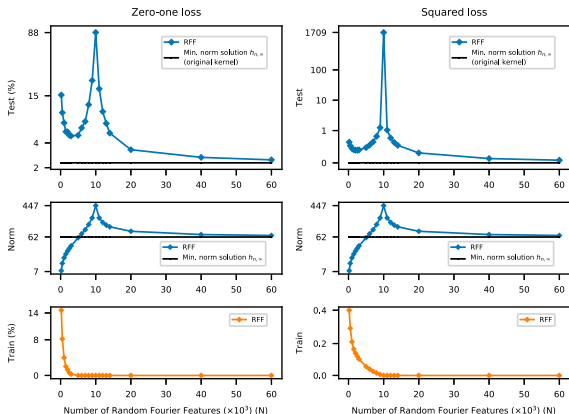


## Random Fourier features:

- ▶ Linear model over  $N$  randomly generated basis functions that approximate a certain (reproducing kernel) Hilbert space as  $N \rightarrow \infty$ .
- ▶ Training method: as proposed explanation with norm  $\|w\|_2$  for model parameters  $w$ .

# Double Descent: Not Just for Neural Networks

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

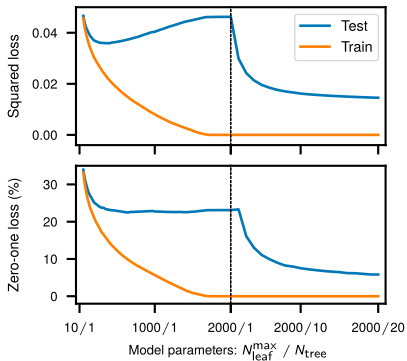


## Random Fourier features:

- ▶ Linear model over  $N$  randomly generated basis functions that approximate a certain (reproducing kernel) Hilbert space as  $N \rightarrow \infty$ .
- ▶ Training method: as proposed explanation with norm  $\|w\|_2$  for model parameters  $w$ .

# 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
- ▶ Training method: standard variant of Breiman's algorithm

# Recent Alternative Explanation [Curth, Jeffares, v.d. Schaar, 2023]:

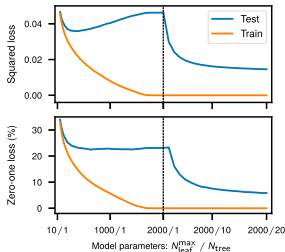
## Need More Careful Parameter Counting

In all non-deep learning experiments by [Belkin et al., 2019]:

- ▶ Below interpolation threshold  $m$ : increase model complexity along dimension 1
- ▶ Above interpolation threshold  $m$ : increase model complexity along dimension 2

### Examples:

- ▶ Random forests: [Belkin et al., 2019] increase depth of single tree up to  $m$  (**complexity dimension 1**). Then average additional trees above  $m$  (**complexity dimension 2**).



# Recent Alternative Explanation [Curth, Jeffares, v.d. Schaar, 2023]: Need More Careful Parameter Counting

In all non-deep learning experiments by [Belkin et al., 2019]:

- ▶ Below interpolation threshold  $m$ : increase model complexity along dimension 1
- ▶ Above interpolation threshold  $m$ : increase model complexity along dimension 2

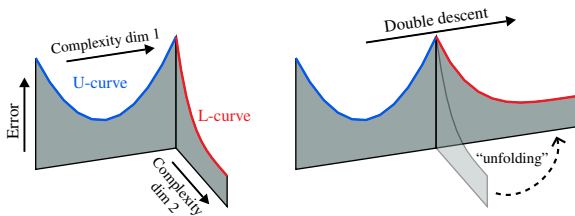
## Examples:

- ▶ Random forests: [Belkin et al., 2019] increase depth of single tree up to  $m$  (**complexity dimension 1**). Then average additional trees above  $m$  (**complexity dimension 2**).
- ▶  $N$  Random Fourier features: equivalent to least squares on basis with dimension  $\min(m, N)$ , obtained by unsupervised dimensionality reduction. [Curth, Jeffares, v.d. Schaar, 2023]
  - ▶ Nr. of least squares parameters is  $\min(m, N)$  (**complexity dimension 1**).
  - ▶ Quality of dimensionality reduction improves with  $m$  (**complexity dimension 2**).

# Recent Alternative Explanation [Curth, Jeffares, v.d. Schaar, 2023]: Need More Careful Parameter Counting

In all non-deep learning experiments by [Belkin et al., 2019]:

- ▶ Below interpolation threshold  $m$ : increase model complexity along dimension 1
- ▶ Above interpolation threshold  $m$ : increase model complexity along dimension 2



Double descent happens because experiments  
**stitch together two independent U-curves!**

# Conclusion

- ▶ Exciting new attempts to understand the **double descent phenomenon** observed in deep learning, random Fourier features, random forests, etc.
- ▶ Crucial to understand true model complexity rather than counting parameters.
- ▶ Analysis involves tools like **Rademacher complexity** that you have learned in this course.
- ▶ Whether proposed explanations can be fully formalized for deep learning remains to be seen...
- ▶ In any case, the role of optimization algorithms in determining effective model complexity provides a **fascinating new frontier** for understanding the classical bias-variance trade-off!