Empirical Risk Minimization

- Assume that there is a joint probability distribution *P(x,y)* over X and Y and the training set consists of *n* data points (x_i, y_i)
- Our goal is to learn a function h that minimizes the risk R(h) where L is the loss function. We do not know the true distribution P(x,y), so we cannot know the true risk. However we can compute an approximation, called the empirical risk, using the training set.
- The hope is that by minimizing the empirical risk, we can learn a good hypothesis function h

$$R(h) = \mathbf{E}[L(h(x), y)] = \int L(h(x), y)\, dP(x, y).$$

$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i).$$

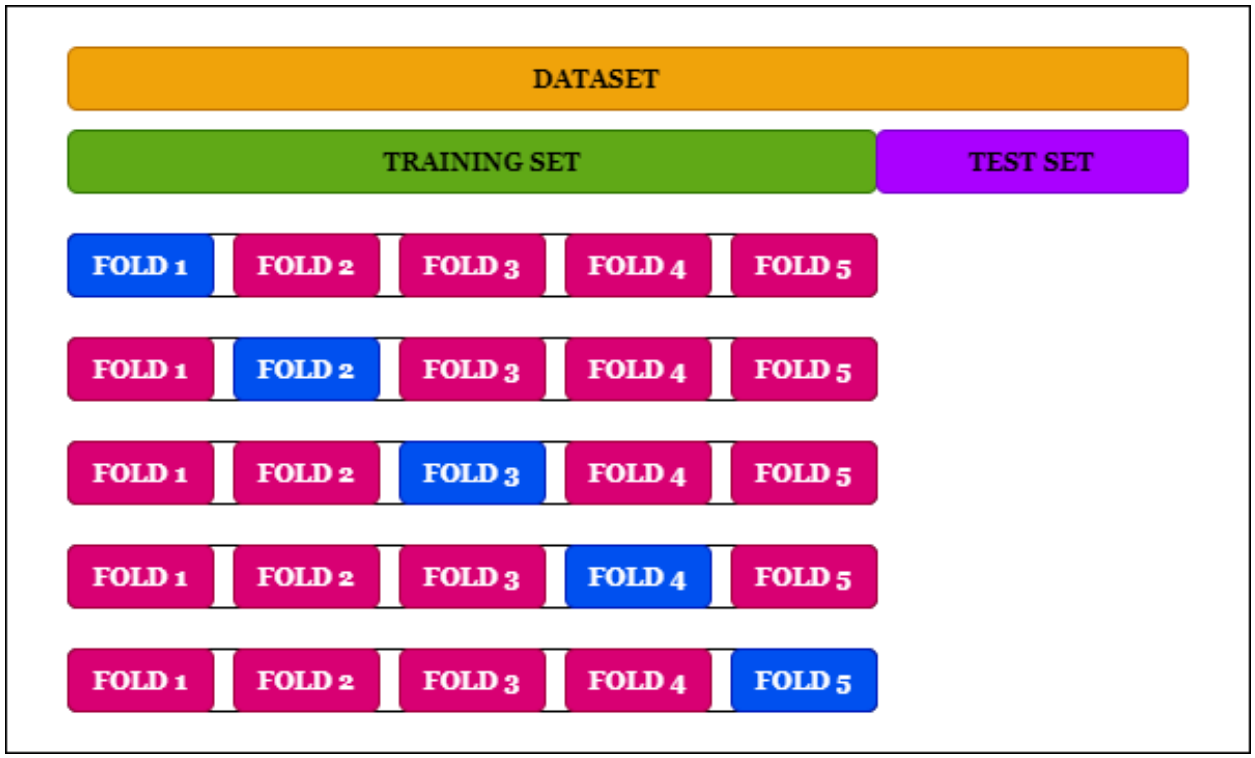
Choosing the hypothesis class

- The function *h* is picked from some hypothesis class *H* as a minimizer of the empirical risk
- Suppose *H* is a neural network. We still have to pick the number of layers, the number of neurons in each layer etc. This is part of the specification of *H*.
- Generically we talk about the "capacity" of *H*. Roughly speaking, bigger networks have higher capacity than smaller networks.
- How should we choose capacity given a certain size of training data set?

Choosing capacity of *H*

- Traditional Statistics views this as a bias-variance tradeoff
- Modern neural network practice doesn't treat this as a tradeoff – go as high capacity as you can (e.g. networks like GPT-3 push the boundary of current computational hardware)
- Resolving the apparent inconsistency between these two views is the subject of much current research

Test sets, validation sets & cross-validation



https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538

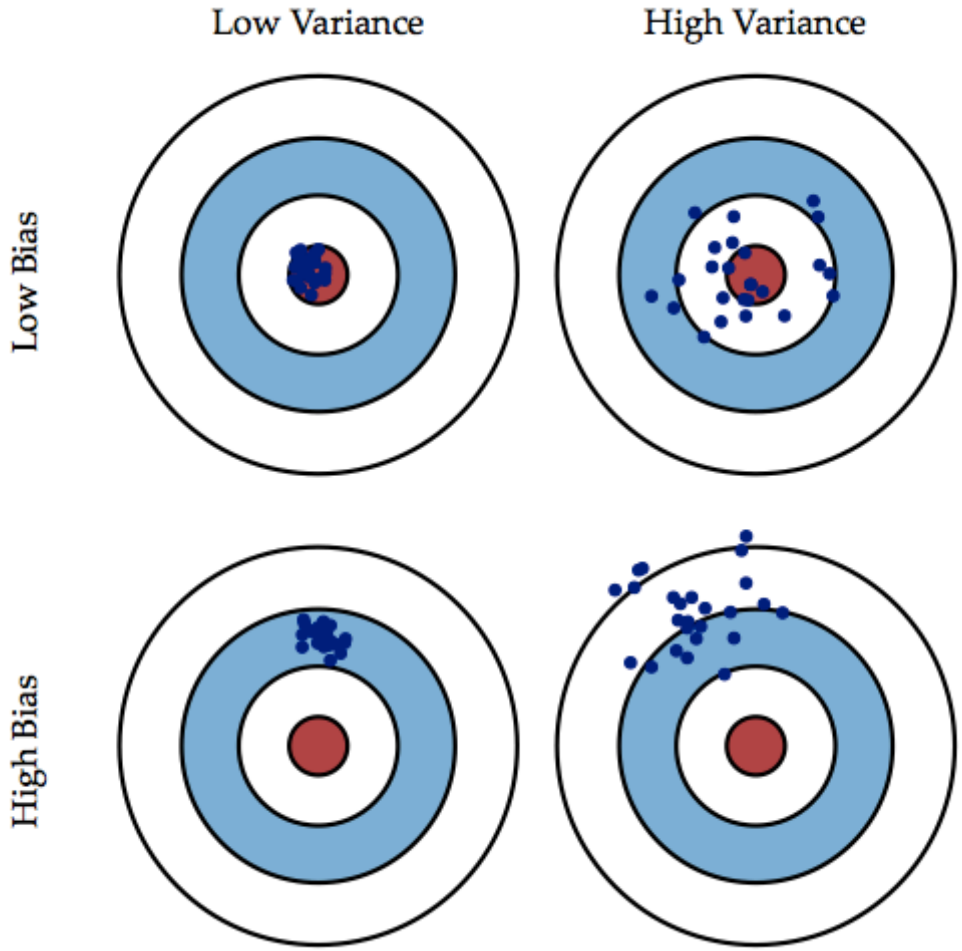
- Do not use test set to set hyper-parameters. That is a grievous sin. The test set should be used once at the very end to report your final numbers
- While developing the approach use a validation set (subset of trianing set). If you are short on data but have plenty of compute, use cross validation

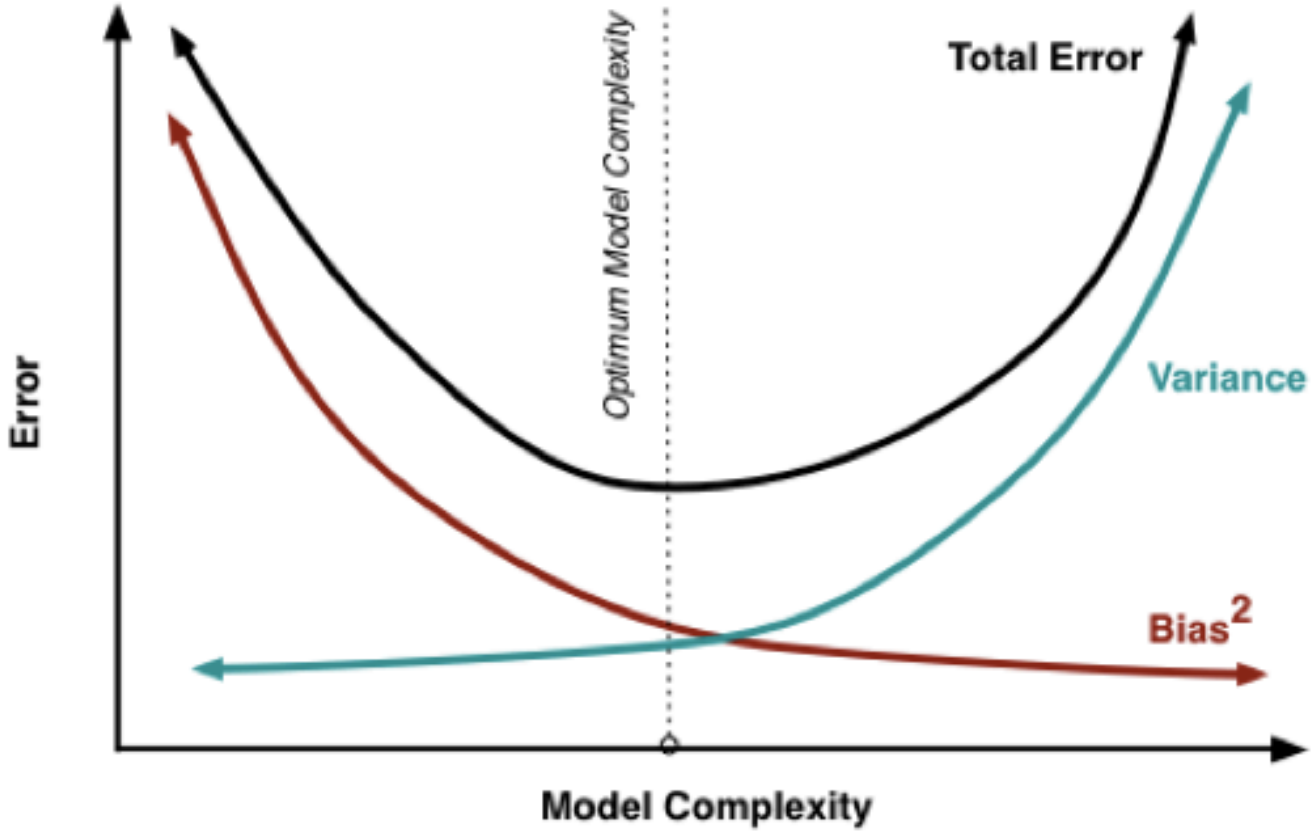
Bias-Variance tradeoff from classical statistics

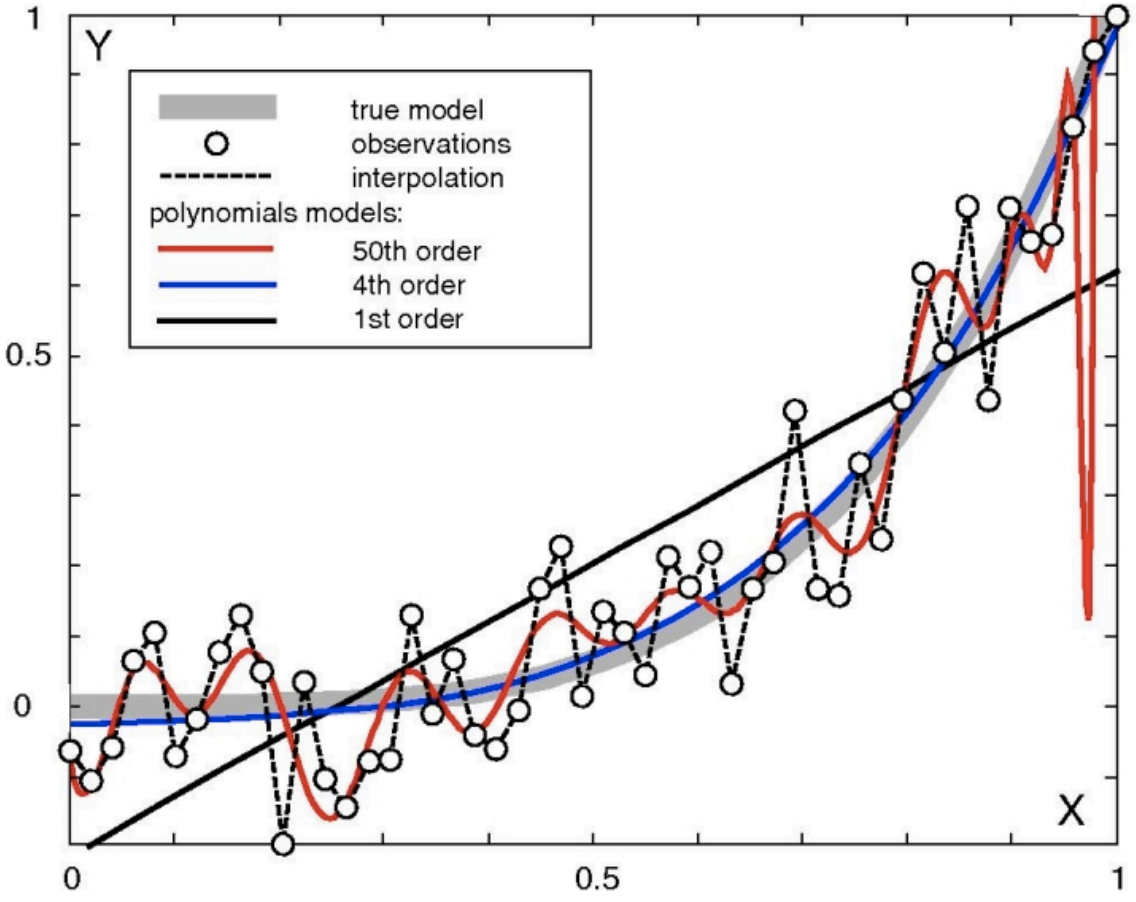
References:

1. Hastie et al, Elements of Stat. Learning Theory, Ch. 7
2. Geman, Bienenstock & Doursat (1992)
3. scott.fortmann-roe.com/docs/BiasVariance.html

Error = $(\text{Bias})^2$ + Variance + Irreducible noise




Error
Optimum Model Complexity
Total Error
Variance
Bias^2
Model Complexity


Y
X
1
0.5
0
0
0.5
1
true model
observations
interpolation
polynomials models:
50th order
4th order
1st order

Bayes Optimal Classifier

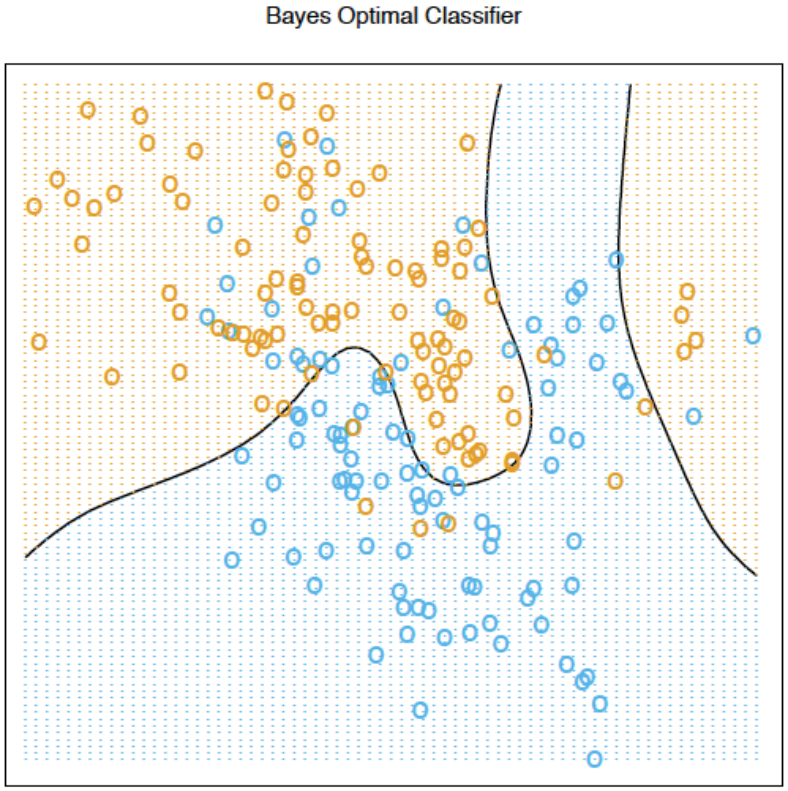
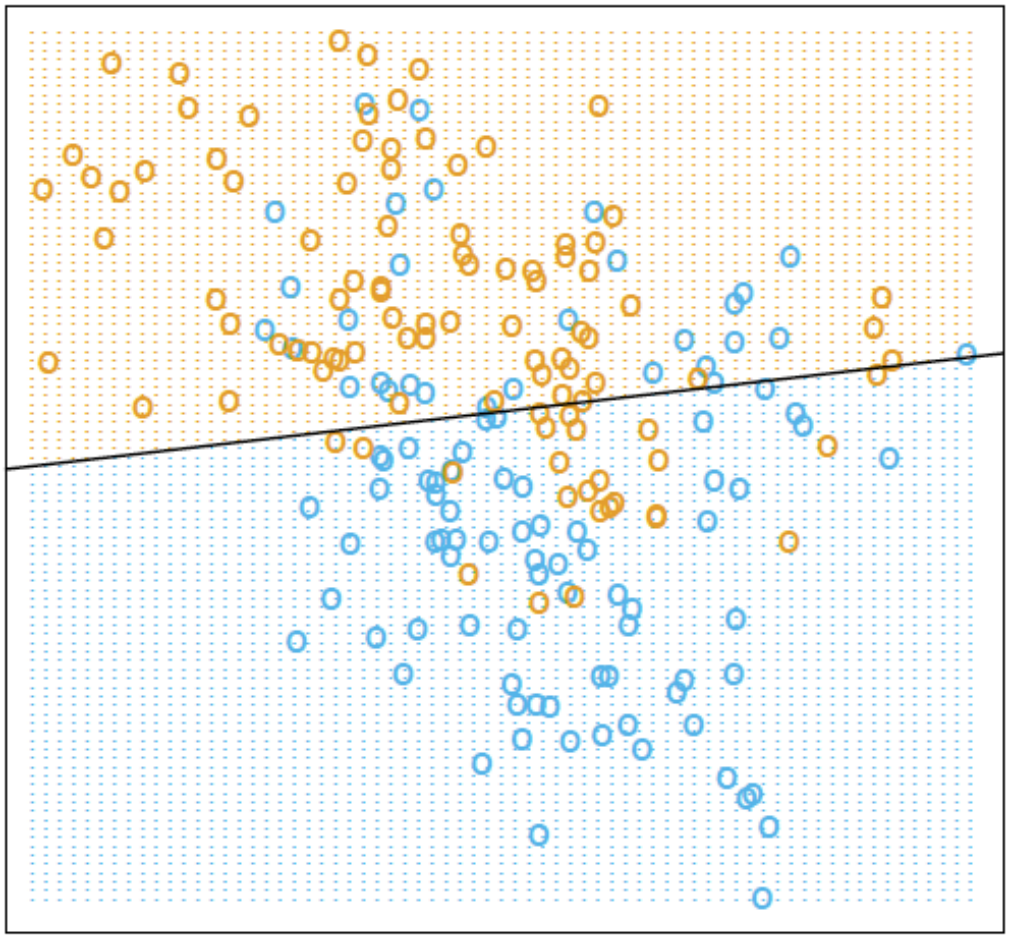


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

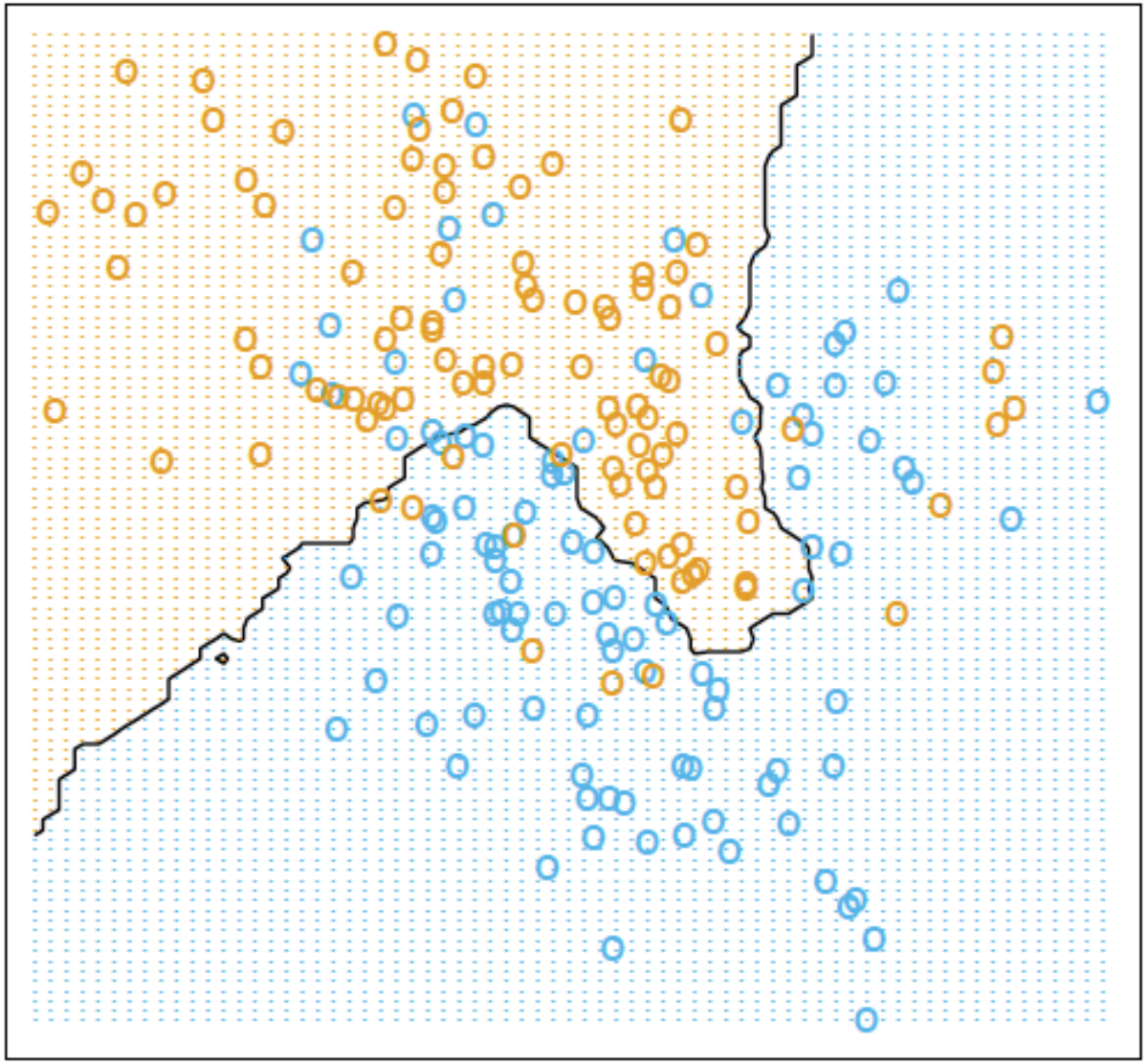
How the data was generated (ESLI, Chapter 2)

tween the two, but closer to Scenario 2. First we generated 10 means m_k from a bivariate Gaussian distribution $N((1,0)^T, \mathbf{I})$ and labeled this class BLUE. Similarly, 10 more were drawn from $N((0,1)^T, \mathbf{I})$ and labeled class ORANGE. Then for each class we generated 100 observations as follows: for each observation, we picked an m_k at random with probability $1/10$, and then generated a $N(m_k, \mathbf{I}/5)$, thus leading to a mixture of Gaussian clusters for each class. Figure 2.4 shows the results of classifying 10,000 new observations generated from the model. We compare the results for least squares and those for k-nearest neighbors for a range of values of k.

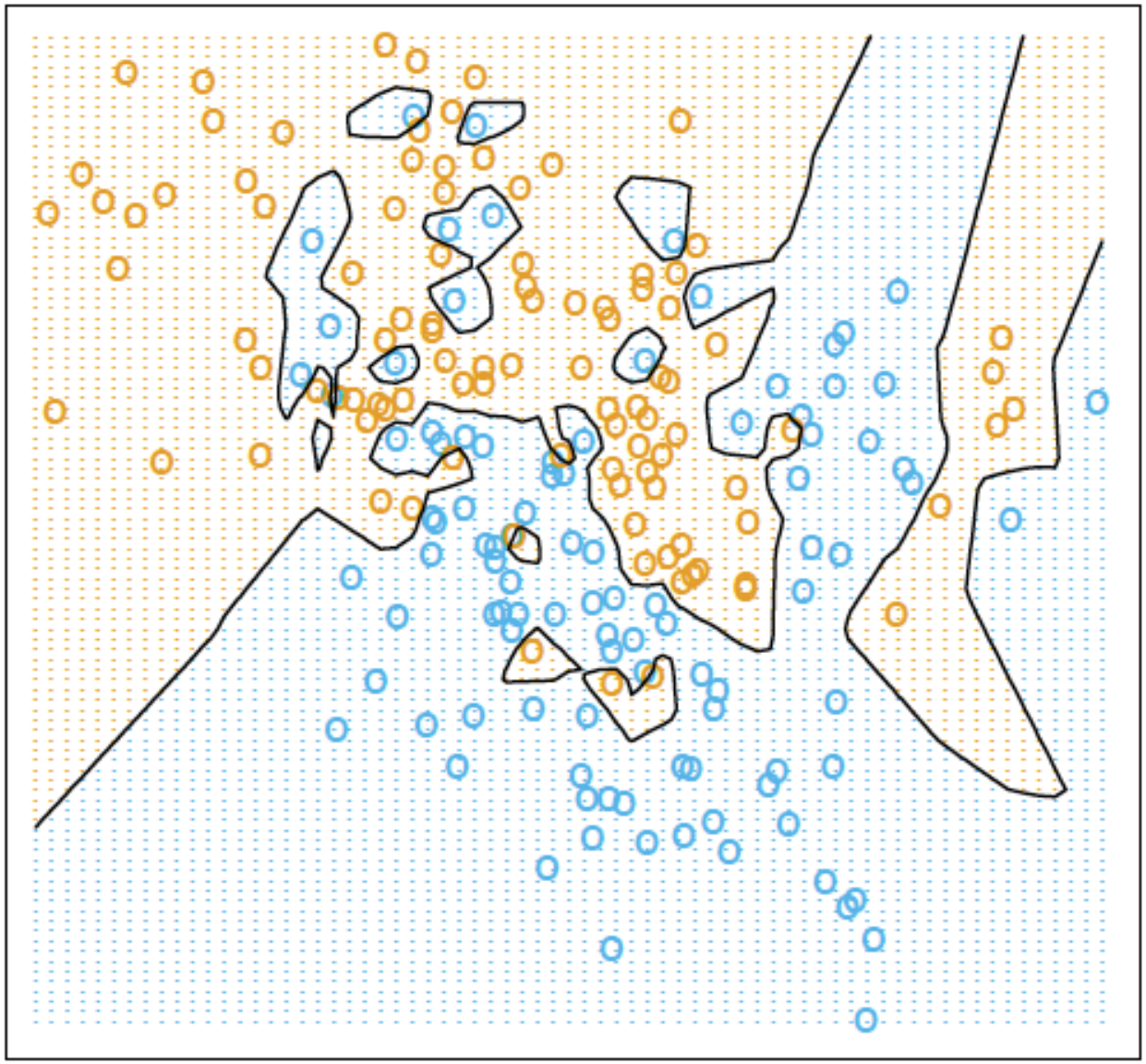
Linear Classifier



15-Nearest Neighbor



1-Nearest Neighbor



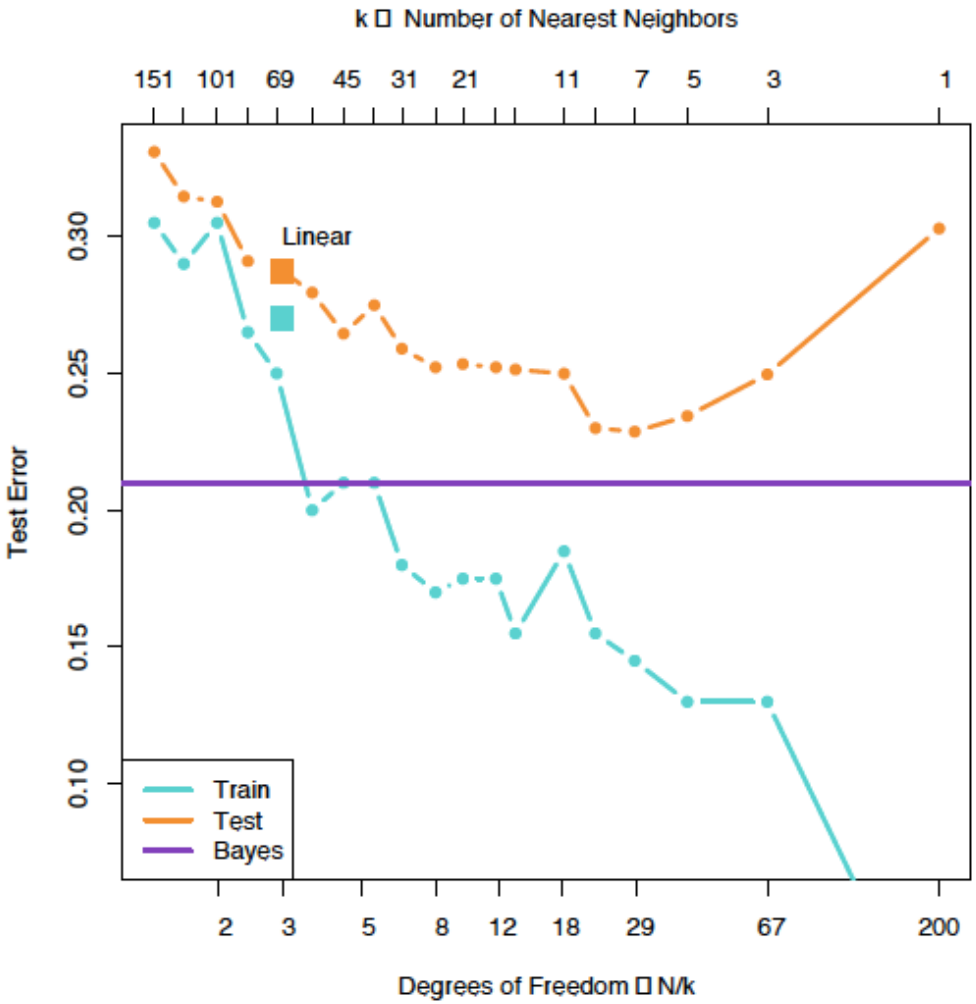


FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* 10,000. *The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.*

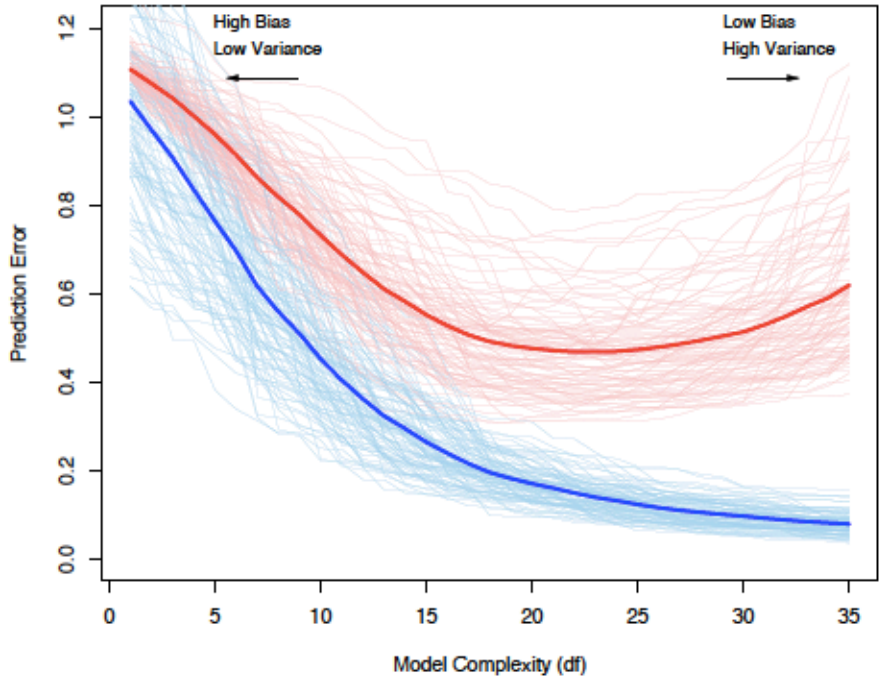


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Suggested Recipe

- The “capacity” of the model is a hyper-parameter
- We choose this based on the error on a ”validation set” (subset of training set put aside for this purpose)
- We expect that as we increase capacity, we will hit a “sweet spot” that we discover using performance on the validation set.
- When we are using capacity smaller than optimal, we are “underfitting”, when we use capacity larger than optimal, we are “overfitting”.

But who do deep learning practitioners actually do?

- Train on the largest over-parameterized model that one can.
- Totally happy with training set error becoming very small

- We don’t fear overfitting !

DOI:10.1145/3446

Understanding Deep Learning (Still) Requires Rethinking Generalization

By Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals

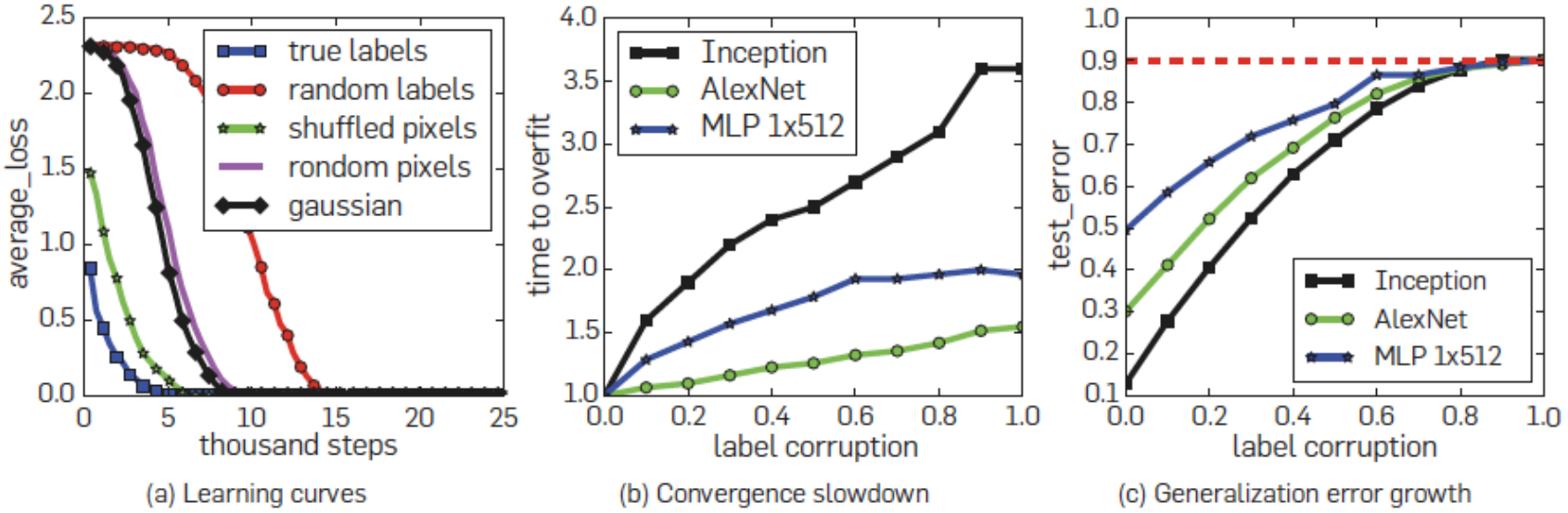
First published at ICLR 2017.
This version in CACM March 2021

The randomization test

- In their primary experiment, they create a copy of the training data where they replace each label independently by a random label chosen from the set of valid labels. A dog picture labeled “dog” might thus become a dog picture labeled “airplane”.
- They then run the learning algorithm on a mixture of the natural data and the randomized data with identical settings and model choice.

Deep Neural Nets can fit random labels

Figure 1. Fitting random labels and random pixels on CIFAR10. (a) The training loss of various experiment settings decaying with the training steps. (b) The relative convergence time with different label corruption ratio. (c) The test error (also the generalization error since training error is 0) under different label corruptions.



(a) Learning curves (b) Convergence slowdown (c) Generalization error growth

Optimization can succeed even when there is no generalization !

Reconciling modern machine-learning practice and the classical bias–variance trade-off

Mikhail Belkin[a,b,1], Daniel Hsu[c], Siyuan Ma[a], and Soumik Mandal[a]

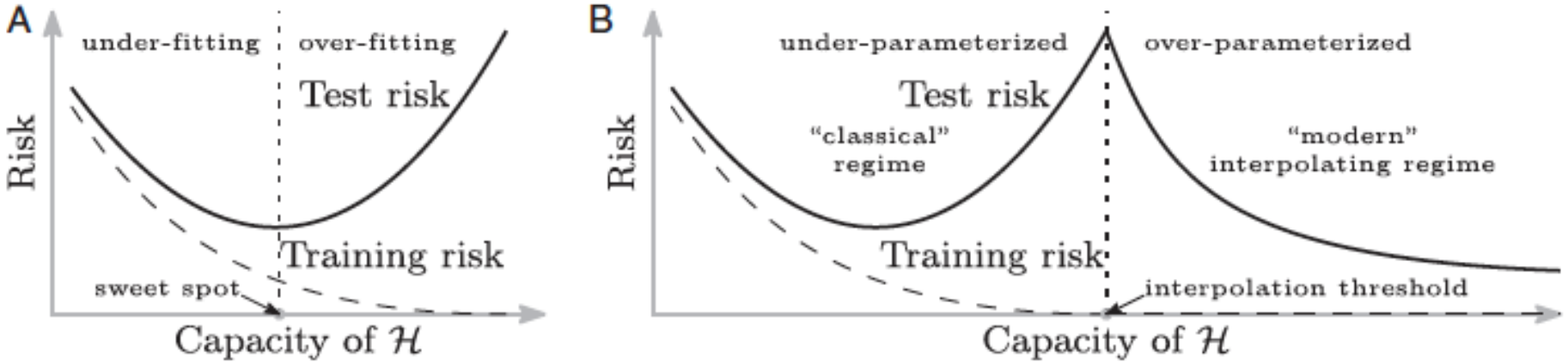


Fig. 1. Curves for training risk (dashed line) and test risk (solid line). (*A*) The classical U-shaped risk curve arising from the bias–variance trade-off. (*B*) The double-descent risk curve, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high-capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Proc. Of National Academy of Sciences, Aug 6, 2019

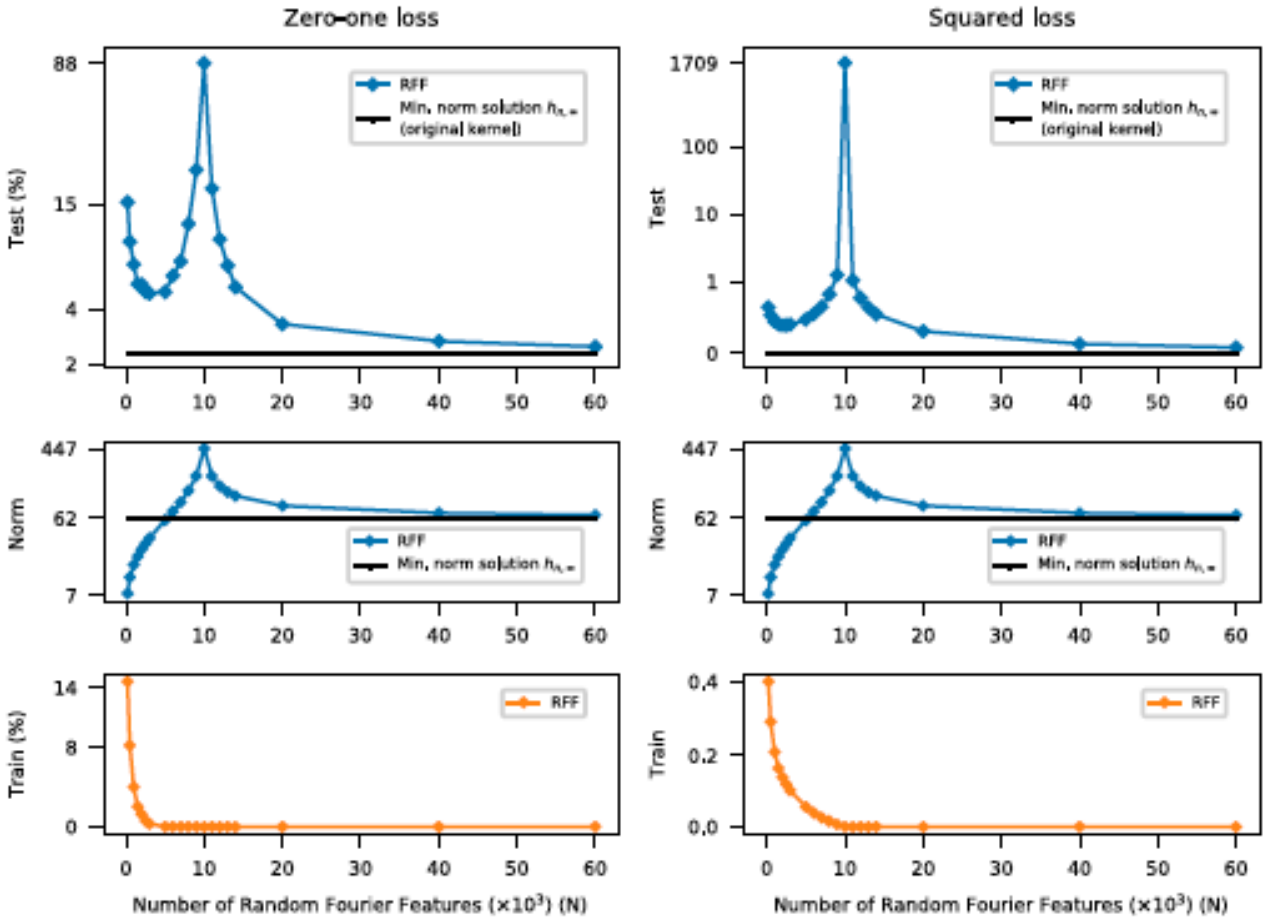


Fig. 2. Double-descent risk curve for the RFF model on MNIST. Shown are test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes). The interpolation threshold is achieved at $N = 10^4$.

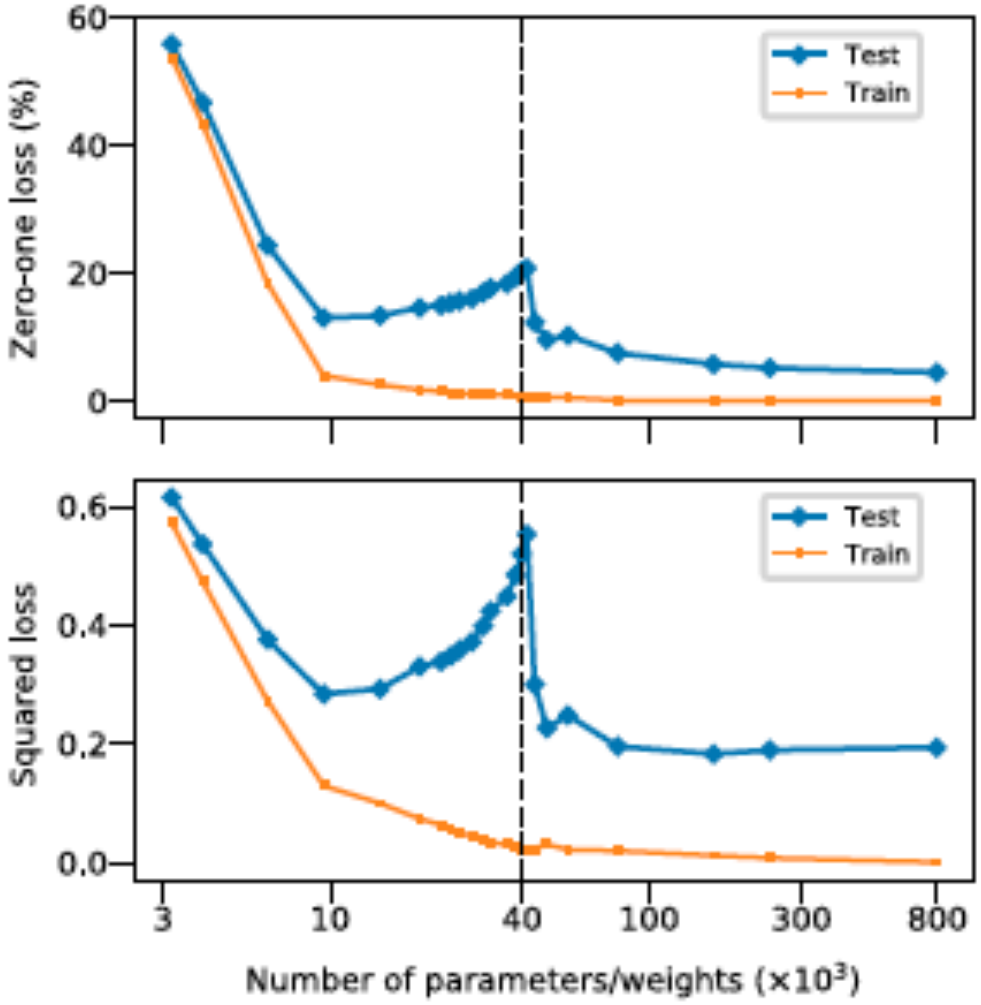


Fig. 3. Double-descent risk curve for a fully connected neural network on MNIST. Shown are training and test risks of a network with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \cdot 10^3$, $d = 784$, $K = 10$ classes). The number of parameters is $(d + 1) \cdot H + (H + 1) \cdot K$. The interpolation threshold (black dashed line) is observed at $n \cdot K$.