Bias-Variance Paradoxes: Double-Descent and Interpolation

Jeffrey Mei

Introduction

- prediction: how will our model perform on unseen data?
- test error can be estimated from training error ("what you see is what you get")
- bias-variance trade-off is the foundation of many modern statistical techniques:
  - regularization: lasso, splines, ridge-regression
  - ensemble methods: random forest, boosting
  - cross-validation

Classical Bias-Variance Tradeoff

$$
\begin{aligned}
MSE\left(\hat{f}(x)\right) &= \mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] \\
&= \mathbb{E}\left[\left(\left(y - \mathbb{E}\left[\hat{f}(x)\right]\right) + \left(\mathbb{E}\left[\hat{f}(x)\right] - \hat{f}(x)\right)\right)^2\right] \\
&= \mathbb{E}\Big[\left(y - \mathbb{E}\left[\hat{f}(x)\right]\right)^2 + \\
&\qquad 2\left(y - \mathbb{E}\left[\hat{f}(x)\right]\right)\left(\mathbb{E}\left[\hat{f}(x)\right] - \hat{f}(x)\right) + \\
&\qquad \left(\mathbb{E}\left[\hat{f}(x)\right] - \hat{f}(x)\right)^2\Big] \\
&= \mathbb{E}\left[\left(y - \mathbb{E}\left[\hat{f}(x)\right]\right)^2\right] + \mathbb{E}\left[\left(\mathbb{E}\left[\hat{f}(x)\right] - \hat{f}(x)\right)^2\right] \\
&= Bias^2(\hat{f}(x)) + \mathbb{V}\left[\hat{f}(x)\right]
\end{aligned}
$$

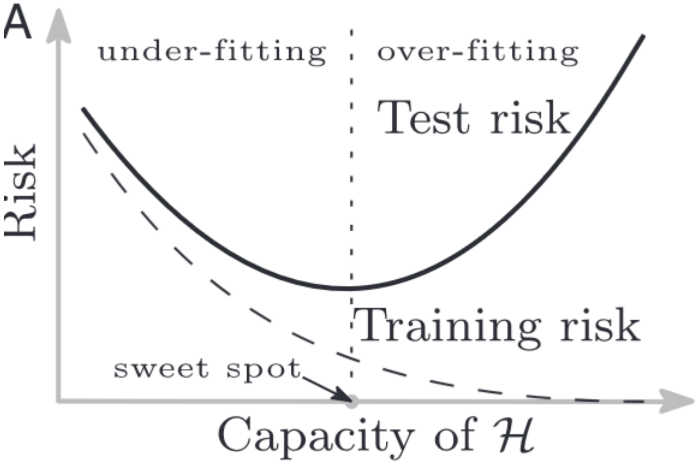


Figure: Classical Bias-Variance Tradeoff

Introduction (continued)

- "Why don't heavily parameterized neural networks overfit the data?" - Leo Breiman (Reflections after Refereeing Papers for NIPS, 1995)
- empirical evidence introduce paradoxes that defy classical explanations (Understanding Deep Learning Requires Rethinking Generalization" - Zhang 2017)
  - replace labels with noise (using CIFAR10 and ImageNet)
  - neural networks can fit training data perfectly (can memorize data set)
  - optimization is only slightly longer, despite not having a pattern to fit

Double-Descent

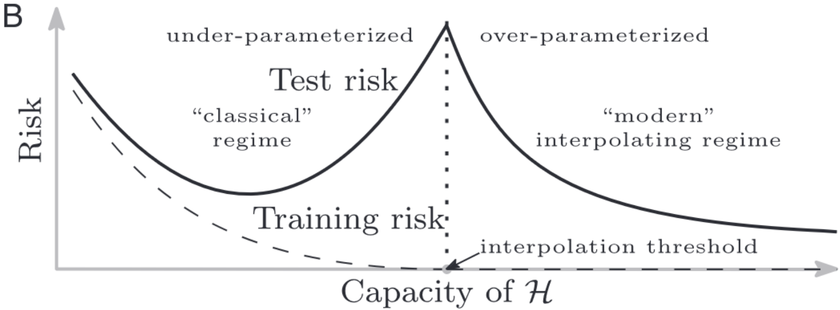


Figure: Modern Bias-Variance Tradeoff

Fully Connected Neural Networks

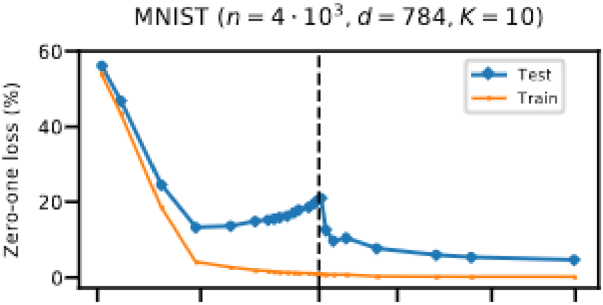


Figure: Fully connected neural networks double-descent (Belkin, 2018). Horizontal axis is number of parameters/weights ($\times 10^3$)

Random Forest

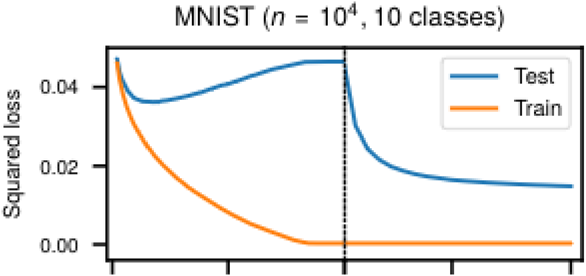


Figure: Random forest double-descent (Belkin, 2018). Horizontal axis is N_{leaf}^{max}/N_{tree}, where N_{leaf}^{max} is the maximum number of leaves on each of the N_{tree} trees.

Boosted Tree

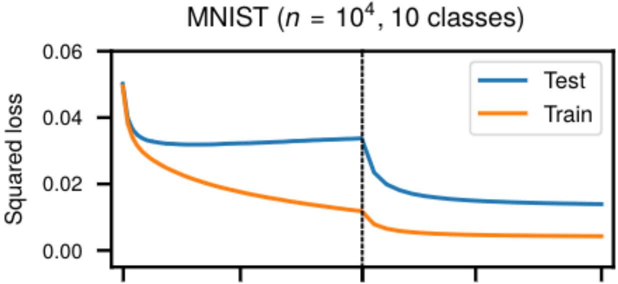
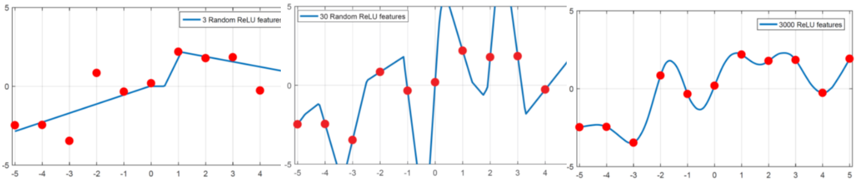


Figure: Boosted tree double-descent (Belkin, 2018). Horizontal axis is N_{tree}/N_{forest}.

Double Descent (continued)



(a) classical (b) overfit (c) overparameterized

Figure: Belkin (Fit without fear)

- RELU is piecewise linear, but (c) looks completely smooth

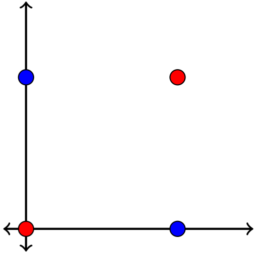
Historical Perspective: Why now?

- **classical statistics:** fixed p, linear models
- **nonparametric statistics:** rich function classes, but always regularized
- **neural networks:** interpolation peak easy to miss; training stops when estimate stops improving

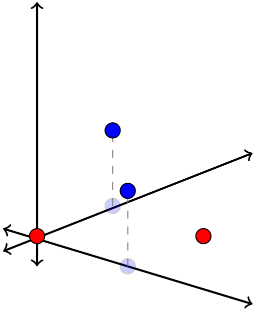
Outline

- What is double descent?
- Why does it happen?
- Why doesn't interpolation necessarily mean poor generalization?

Kernel Trick: XOR

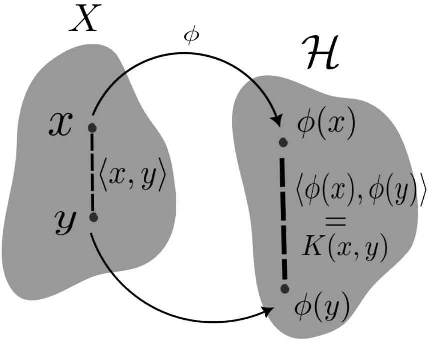


(a) XOR cannot be separated using linear function



(b) Project XOR into higher dimension to find a linear separator.

Reproducing Kernel Hilbert Space (RKHS)



- allows us to implicitly access higher dimensions cheaply through "kernel trick"
- $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ defined implies reproducing kernel $K(x, y)$ exists (and vice versa)

Kernel Trick: Polynomial Kernel

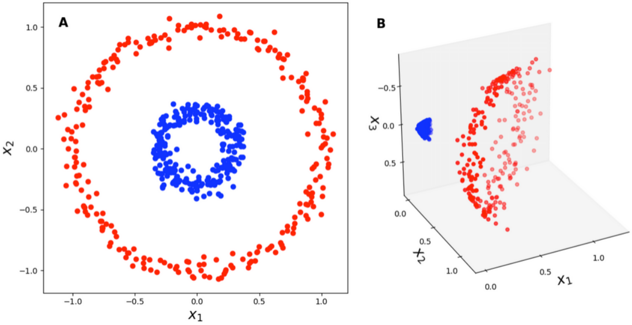


Figure: Polynomial kernel makes space linearly separable [https://gregorygundersen.com/blog/2019/12/10/kernel-trick/]

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}, \quad K(x, y) = \langle \phi(x), \phi(y) \rangle = \langle x, y \rangle^2$$

To Understand Deep Learning We Need to Understand Kernel Learning - Belkin

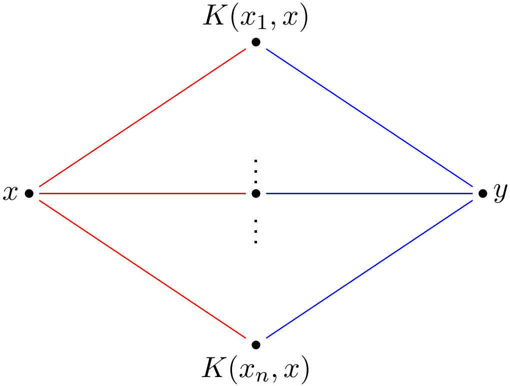
Kernel Machines

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), \qquad \alpha_i \in \mathcal{R} \tag{1}$$

where $K(x, z) = e^{-\|x-z\|^2/2}$ (gaussian kernel).

- can be interpreted as a two-layer neural network
  - first layer is fixed non-linear transformation determined by kernel
  - second layer is regression on feature space

Kernel Machine as Neural Network



X $\qquad$ $\phi(X)$ $\qquad$ $\sum_{i=1}^{n} \alpha_i K(x_i, x)$

To Understand We Need to Understand Kernel Learning - Belkin

There is a unique predictor f_{ker} that interpolates the data

$$f_{ker}(x_i) = y_i \text{ for all } i = 1, ..., n$$

Corrupting Data:
$(x_i, y_i) \sim P$ for $i = 1, ..., n$.
Let there be a probability q that a label y_i becomes mislabeled.
Even after corrupting the label data, the interpolator is nearly Bayes optimal:

$$f^*_{P_q} = f^*_P$$

To Understand Deep Learning We Need to Understand Kernel Learning - Belkin

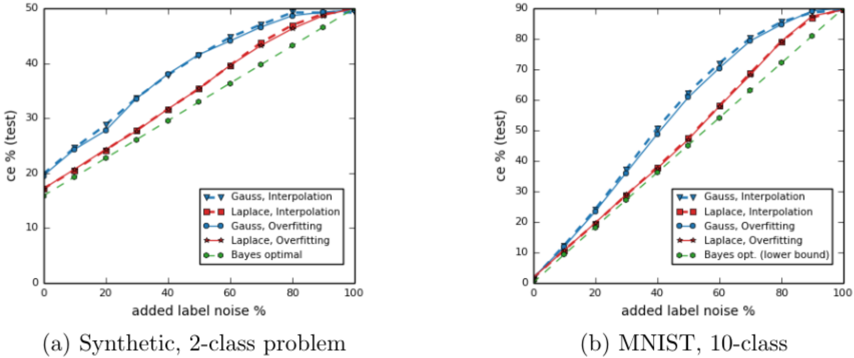


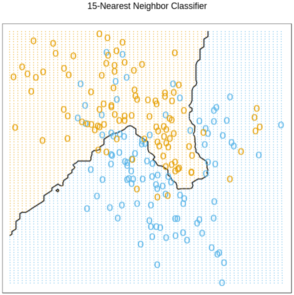
Figure: To understand deep learning we need to understand kernel learning

To Understand Deep Learning We Need to Understand Kernel Learning - Belkin

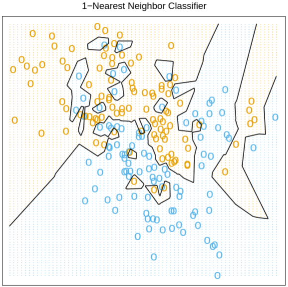
Consequences:

- we can interpolate mislabeled data and still get nearly optimal generalization performance
- must understand interpolators more deeply

Interpolation



(a) Elements of statistical learning figure (2.2)



(b) Elements of statistical learning figure (2.3)

- 1-NN is an interpolator for training data
- to improve risk, we increase K, but then KNN is no longer an interpolator
- not Bayes optimal when $K = 1$

Simplical Interpolation

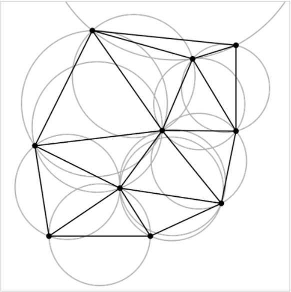
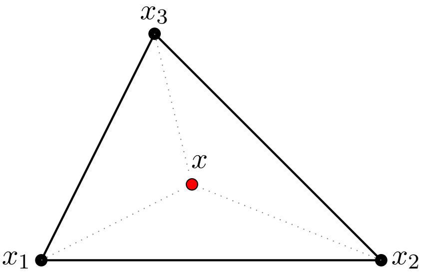


Figure: Delaunay Triangulation
https://en.wikipedia.org/wiki/Delaunay_triangulation

- partition convex hull of data into d-dimensional simplices
- find triangulation that interpolates data: $f_{simp}(x_i) = y_i$
- near-optimal risk with high dimension

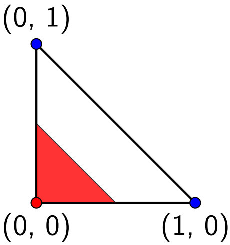
$$\mathcal{R}(f_{simp}) - \mathcal{R}(f^*) = O\left(1/\sqrt{d}\right) \tag{2}$$

Simplical Interpolation

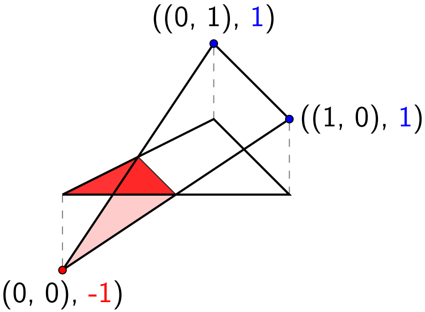


- linear interpolant: $y = \sum_{i=1}^{d+1} w_i y_i$
- $w_i : \sum_{i=1}^{d+1} w_i = 1$ are barycentric coordinates
  - reparameterize space as weighted sum of vertices

Simplical Interpolation



(a) 2D perspective



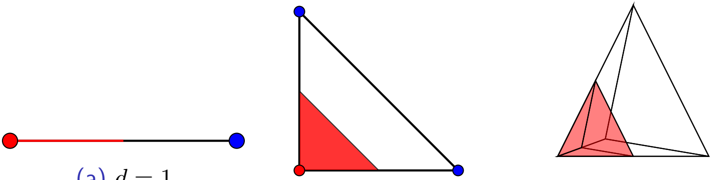
(b) 3D perspective

Misclassification region:

$$vol(s_{1/2}) = \frac{1}{2^d} vol(s_d) \tag{3}$$

- noisy points get localized in high dimensions

Simplical Interpolation



(a) $d = 1$ (b) $d = 2$ (c) d=3

- this interpolation scheme is not consistent
- gives us some insight into how interpolation may be compatible with generalization

Weighted k-NN

$$K(x, z) = \frac{1}{\|x - z\|^\alpha}, \quad \alpha > 0 \tag{4}$$

$$f_{sing}(x) = \frac{\sum_{i=1}^{k} K(x, x_i) y_i}{\sum_{i=1}^{k} K(x, x_i)} \tag{5}$$

- consistent interpolator

Weighted k-NN

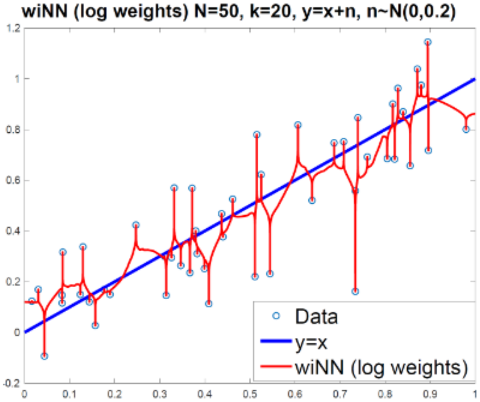


Figure: Weighted Interpolating Nearest Neighbors (fit without fear)

- actual set where points are "overfit" is relatively small

Theory

- "overfitting or perfect fitting"
- μ probability distribution on $\Omega \subset \mathbb{R}^d$ compact
- $f^*(x)$: optimal classifier
- $\hat{f}(x)$: consistent interpolating classifier
- $\mathcal{A}_n = \left\{ x \in \Omega : \hat{f}_n(x) \neq f^*(x) \right\}$ set of adversarial examples

$$\lim_{n \to \infty} \mu(\mathcal{A}_n) = 0$$

Theorem

Aribtrary closeness For any $\varepsilon > 0$ and $\delta \in (0, 1)$, there exists $N \in \mathbb{N}$, such that for all $n \geq N$, with probability $\geq \delta$, every point in Ω is within distance 2ε of the set $\mathcal{A}_n$.

Conclusion

- bias-variance trade-off does not explain success of interpolation
- theory for interpolation is severely underdeveloped
- interpolation is compatible with optimality

Outstanding Questions:

- does this apply to all data?
- how does this apply to the "deep" part of deep neural networks?

References

- Surprises in High-dimensional ridgeless least squares interpolation
- Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate
- Reconciling modern machine-learning practice and the classical bias-variance trade-off
- Does data interpolation contradict statistical optimality?
- Fit without fear
- To understand deep learning we need to understand kernel learning
- Understanding deep learning requires rethinking generalization
- On the role of optimization in double descent: a least squares study