Generalization Error in Neural Networks

Henry D. Pfister
Duke University

Machine Learning Summer School
Duke University
June 21st, 2019

Deep Neural Networks (DNNs)

- Important theoretical questions
  - With more parameters than data, why do DNNs generalize?
  - With zero training error, why don't they overfit?

Deep Neural Networks (DNNs)

- Important theoretical questions
  - With more parameters than data, why do DNNs generalize?
  - With zero training error, why don't they overfit?
- Notable advances in understanding over past 2 to 3 years
  - Over parametrization $\rightarrow$ easy optimization without overfitting!
  - Large NNs have loss landscapes with connected minima
  - Unifying themes: SGD, flat minima, connections to kernel methods
  - More work needed: multiple partial explanations must be reconciled

Deep Neural Networks (DNNs)

- Important theoretical questions
  - With more parameters than data, why do DNNs generalize?
  - With zero training error, why don't they overfit?
- Notable advances in understanding over past 2 to 3 years
  - Over parametrization $\rightarrow$ easy optimization without overfitting!
  - Large NNs have loss landscapes with connected minima
  - Unifying themes: SGD, flat minima, connections to kernel methods
  - More work needed: multiple partial explanations must be reconciled
- Experiments vs. Proofs
  - Researchers in machine learning like mathematical proofs
  - Without experiments, it's not even clear what to prove!
  - Progress driven by simple experiments exposing key properties

Outline

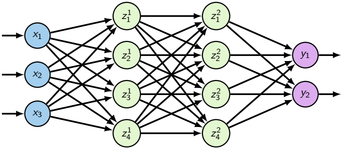
- Background
  - Neural networks and training
- Predicting Neural Network Performance
  - Model complexity and overparametrization
  - Classical bias-variance trade-off
- The Loss Landscape
  - Visualization via 1D and 2D projections
  - Connections to generalization error
  - Modern bias-variance trade-off

Outline

- Background
  - Neural networks and training
- Predicting Neural Network Performance
  - Model complexity and overparametrization
  - Classical bias-variance trade-off
- The Loss Landscape
  - Visualization via 1D and 2D projections
  - Connections to generalization error
  - Modern bias-variance trade-off
- Acknowledgment
  - This is a survey of recent results and essentially all the ideas and figures are taken (with citations) from other people's papers

Problem Setup

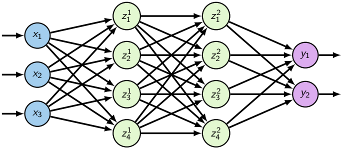
- Neural network
  - function f_θ from $\mathcal{X} = \mathbb{R}^n$ to $\mathcal{Y} = \mathbb{R}^d$
  - weights represented by $\theta \in \mathbb{R}^p$



Loss landscape taken from `https://www.cs.umd.edu/~tomg/img/landscapes/noshort.png` [LXT+18]

Problem Setup

- Neural network
  - function f_θ from $\mathcal{X} = \mathbb{R}^n$ to $\mathcal{Y} = \mathbb{R}^d$
  - weights represented by $\theta \in \mathbb{R}^p$

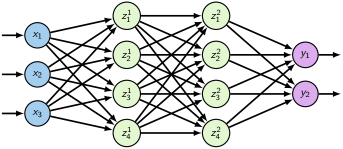


- Training set
  - Set of tuples $(x, y) \in \mathcal{X} \times \mathcal{Y}$
  - For classification into d classes, let $y \in \mathcal{Y}$ be a one-hot vector
  - Entire training set denoted $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$

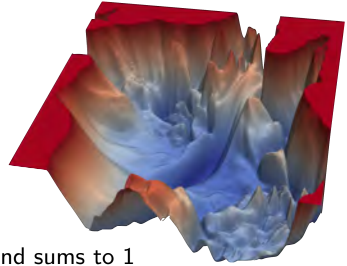
Loss landscape taken from `https://www.cs.umd.edu/~tomg/img/landscapes/noshort.png` [LXT+18]

Problem Setup

- Neural network
  - function f_θ from $\mathcal{X} = \mathbb{R}^n$ to $\mathcal{Y} = \mathbb{R}^d$
  - weights represented by $\theta \in \mathbb{R}^p$



- Training set
  - Set of tuples $(x, y) \in \mathcal{X} \times \mathcal{Y}$
  - For classification into d classes, let $y \in \mathcal{Y}$ be a one-hot vector
  - Entire training set denoted $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$



- Loss function
  - Cross entropy: $L(y, \hat{y}) \triangleq -\sum_{i=1}^{d} y_i \ln \hat{y}_i$
  - Loss for entire training set is

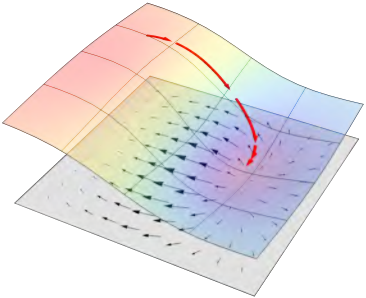
$$\mathcal{L}_{\mathcal{D}}(\theta) \triangleq \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f_\theta(x))$$

  - note: assume $f_\theta(x) \in \mathbb{R}^d$ non-negative and sums to 1

Loss landscape taken from https://www.cs.umd.edu/~tomg/img/landscapes/noshort.png [LXT+18]

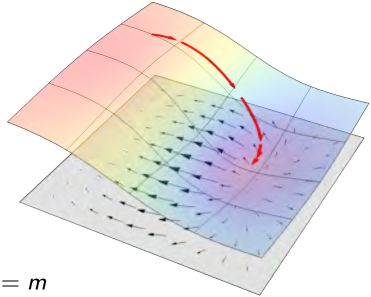
Neural Network Training

- Loss $\mathcal{L}_{\mathcal{D}}(\theta)$ landscape
  - Adjust $\theta \in \mathbb{R}^p$ to minimize loss
  - Gradient vector $\nabla \mathcal{L}_{\mathcal{D}}(\theta)$ gives θ direction of maximum increase



Neural Network Training

- Loss $\mathcal{L}_{\mathcal{D}}(\theta)$ landscape
  - Adjust $\theta \in \mathbb{R}^p$ to minimize loss
  - Gradient vector $\nabla \mathcal{L}_{\mathcal{D}}(\theta)$ gives θ direction of maximum increase

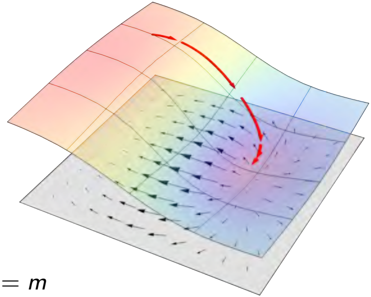


- (stochastic) Gradient descent
  - full-batch: $\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}_{\mathcal{D}}(\theta_t)$
  - mini-batch: subset $\mathcal{S}_t \subset \mathcal{D}$ with $|\mathcal{S}_t| = m$

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}_{\mathcal{S}_t}(\theta_t)$$

Neural Network Training

- Loss $\mathcal{L}_{\mathcal{D}}(\theta)$ landscape
  - Adjust $\theta \in \mathbb{R}^p$ to minimize loss
  - Gradient vector $\nabla \mathcal{L}_{\mathcal{D}}(\theta)$ gives θ direction of maximum increase



- (stochastic) Gradient descent
  - full-batch: $\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}_{\mathcal{D}}(\theta_t)$
  - mini-batch: subset $\mathcal{S}_t \subset \mathcal{D}$ with $|\mathcal{S}_t| = m$

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}_{\mathcal{S}_t}(\theta_t)$$

- Generalization
  - The test set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$ contains held-out training data
  - Actual goal is to minimize $\mathcal{L}_{\mathcal{T}}(\theta)$ without knowing $\mathcal{T}$!

Predicting Neural Network Performance

- Key questions
  - Will gradient descent reach a local minima? a global minima?
  - Error rate on the training data? on the test data?
  - Effect of network complexity? of mini-batch size?

Predicting Neural Network Performance

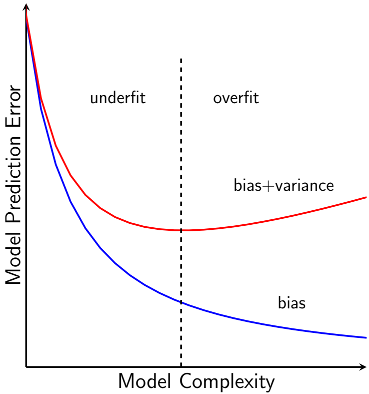
- Key questions
  - Will gradient descent reach a local minima? a global minima?
  - Error rate on the training data? on the test data?
  - Effect of network complexity? of mini-batch size?
- Overparametrized (OP) regime
  - Many more parameters than training samples (i.e., $p \gg |\mathcal{D}|$)
  - Theory predicts gradient descent achieves zero training error (many papers including [SC16, LL18, AZLL18, DLL^{+}18])
  - Kernel connection allows bounds on generalization error [ADH^{+}19]

Predicting Neural Network Performance

- Key questions
  - Will gradient descent reach a local minima? a global minima?
  - Error rate on the training data? on the test data?
  - Effect of network complexity? of mini-batch size?
- Overparametrized (OP) regime
  - Many more parameters than training samples (i.e., $p \gg |\mathcal{D}|$)
  - Theory predicts gradient descent achieves zero training error (many papers including [SC16, LL18, AZLL18, DLL$^+$18])
  - Kernel connection allows bounds on generalization error [ADH$^+$19]
- Rough answers (i.e., folklore that is proven in special cases)
  - Gradient descent typically reaches a global min with zero error
  - Stochastic gradient descent biased towards flat minima
  - Performance depends weakly on m and optimization method

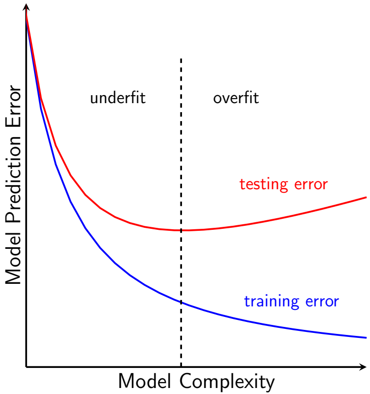
Model Complexity: Classical Perspective

- Classical bias-variance trade-off
  - Bias error due to overly simple model decreases with complexity
  - Variance error from parameter noise increases with complexity



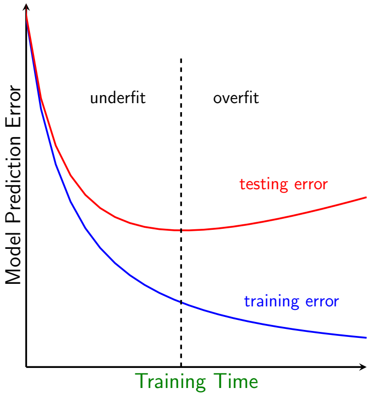
Model Complexity: Classical Perspective

- Classical bias-variance trade-off
  - Bias error due to overly simple model decreases with complexity
  - Variance error from parameter noise increases with complexity
- Overparametrized neural networks
  - Can fit random labels! [ZBH+16]
  - Thus, no classical reason to expect good generalization!
  - But, learning random labels takes more training time...

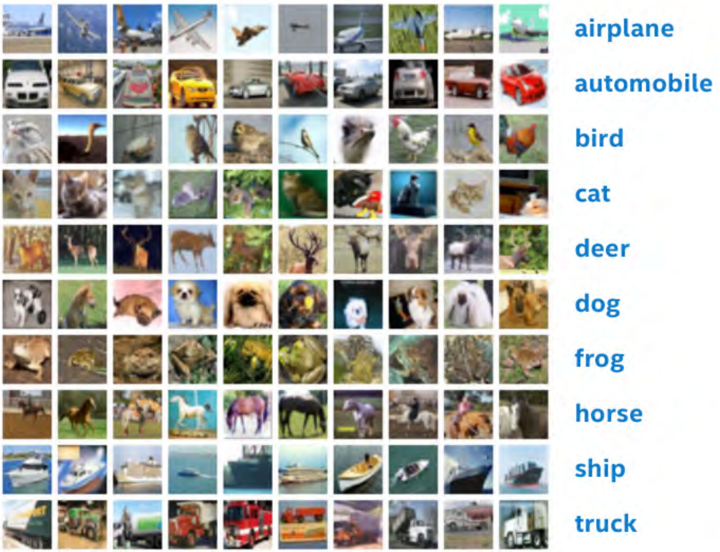


Model Complexity: Classical Perspective

- Classical bias-variance trade-off
  - Bias error due to overly simple model decreases with complexity
  - Variance error from parameter noise increases with complexity
- Overparametrized neural networks
  - Can fit random labels! [ZBH+16]
  - Thus, no classical reason to expect good generalization!
  - But, learning random labels takes more training time...
- Similar curve for NN training
  - Model complexity → training time
  - Training time is related to model complexity
  - In practice, use cross validation to pick stopping time



Example Dataset: CIFAR-10



10 classes of 32x32 RGB images with 6000 images per class

The Loss Landscape: 1D Projections

- Visualization
  - Difficult to gain insight from low-D plot of high-D functions
  - Idea: linear interpolation between weight vectors $\theta^{(0)}, \theta^{(1)}$:

$$\ell(\alpha) = \mathcal{L}_{\mathcal{D}}\big(\theta^{(0)} + \alpha(\theta^{(0)} - \theta^{(1)})\big)$$

Figure taken from [KMN$^+$17]

The Loss Landscape: 1D Projections

- Visualization
  - Difficult to gain insight from low-D plot of high-D functions
  - Idea: linear interpolation between weight vectors $\theta^{(0)}, \theta^{(1)}$:

$$\ell(\alpha) = \mathcal{L}_{\mathcal{D}}\big(\theta^{(0)} + \alpha(\theta^{(0)} - \theta^{(1)})\big)$$

- Experiments to test if "flat minima generalize" [KMN$^+$17]
  - Train a convolutional neural network (CNN) on CIFAR-10 using small/large mini-batches ($m = 256/5\text{K}$) to get $\theta^{(0)}, \theta^{(1)}$

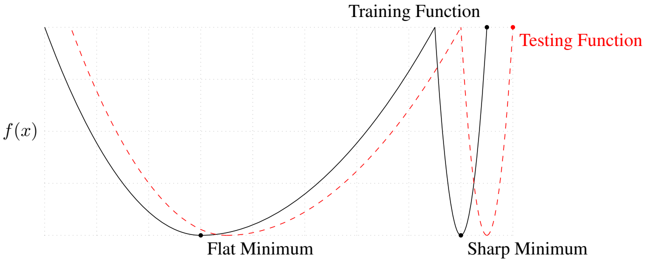


Figure taken from [KMN$^+$17]

The Loss Landscape: 1D Projections

- Visualization
  - Difficult to gain insight from low-D plot of high-D functions
  - Idea: linear interpolation between weight vectors $\theta^{(0)}, \theta^{(1)}$:

$$\ell(\alpha) = \mathcal{L}_{\mathcal{D}}\big(\theta^{(0)} + \alpha(\theta^{(0)} - \theta^{(1)})\big)$$

- Experiments to test if "flat minima generalize" [KMN$^+$17]
  - Train a convolutional neural network (CNN) on CIFAR-10 using small/large mini-batches ($m = 256/5K$) to get $\theta^{(0)}, \theta^{(1)}$

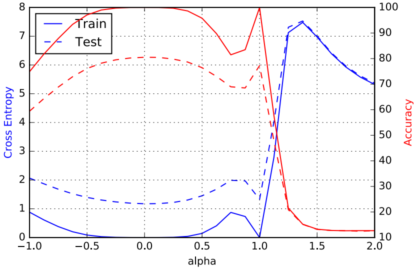
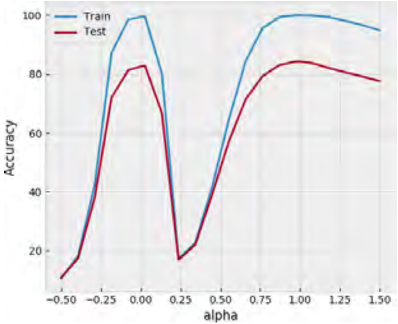
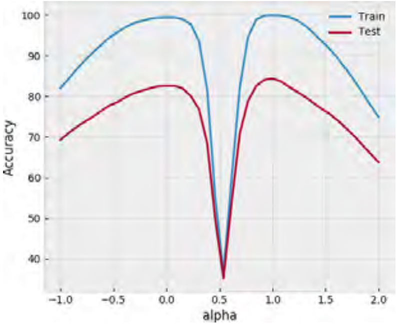


Figure taken from [KMN$^+$17]

Sharp vs. Flat vs. Scaling





In reality, the width of the minima is quite affected by:
optimizer, scaling issues, batch normalization, and norms of $\theta^{(0)}, \theta^{(1)}$

Can use batch normalization to rescale weights (except last layer) to same norm

Comparing optimizers (SGD at $\alpha = 0$, ADAM at $\alpha = 1$):
left/right figures are before/after rescaling

Interpolation of batch normalization discussed in [GIP+18]

Properties of the Loss Landscape

- Question: Does the loss landscape have isolated minima?
  - Visualization in high-D is tricky but some properties can be tested

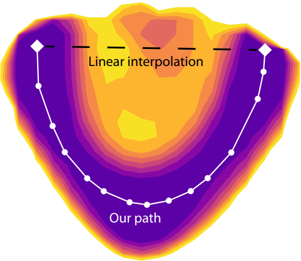
Figure taken from [DVSH18]

Properties of the Loss Landscape

- Question: Does the loss landscape have isolated minima?
  - Visualization in high-D is tricky but some properties can be tested
- Experiment from [DVSH18]
  - DenseNets with weights $\theta^{(0)}, \theta^{(1)}$ having zero loss on CIFAR-10
  - SGD used to find curved path from $\theta^{(0)}$ to $\theta^{(1)}$ with minimal loss
  - Study found low-loss paths connecting a set of trained networks

Figure taken from [DVSH18]

- Question: Does the loss landscape have isolated minima?
  - Visualization in high-D is tricky but some properties can be tested
- Experiment from [DVSH18]
  - DenseNets with weights $\theta^{(0)}, \theta^{(1)}$ having zero loss on CIFAR-10
  - SGD used to find curved path from $\theta^{(0)}$ to $\theta^{(1)}$ with minimal loss
  - Study found low-loss paths connecting a set of trained networks



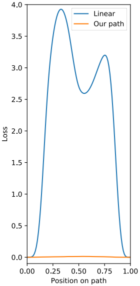


Figure taken from [DVSH18]

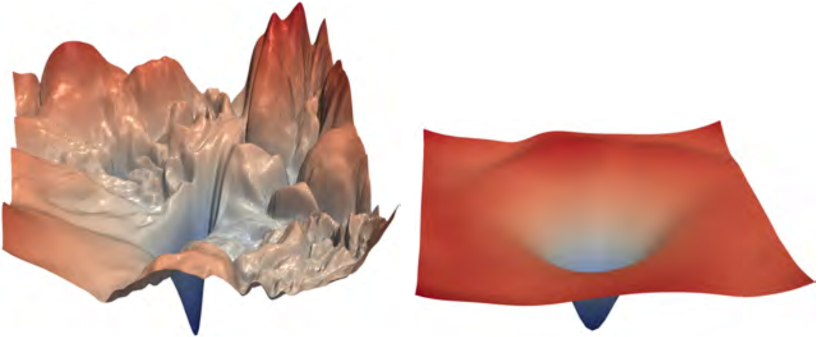
Visualizing the Loss Landscape

- People love pretty pictures but meaningful visualization is hard
  - Consider the 2D projection $g(\alpha, \beta) = \mathcal{L}_{\mathcal{D}}(\theta + \alpha\Delta_1 + \beta\Delta_2)$ where vectors $\Delta^{(0)}, \Delta^{(1)} \in \mathbb{R}^p$ are chosen randomly
  - Scaling is problematic, so novel filter normalization is used [LXT+18]

ResNet-56 loss landscape for CIFAR-10 from [LXT+18]

Visualizing the Loss Landscape

- People love pretty pictures but meaningful visualization is hard
  - Consider the 2D projection $g(\alpha, \beta) = \mathcal{L}_{\mathcal{D}}(\theta + \alpha\Delta_1 + \beta\Delta_2)$ where vectors $\Delta^{(0)}, \Delta^{(1)} \in \mathbb{R}^p$ are chosen randomly
  - Scaling is problematic, so novel filter normalization is used [LXT+18]



(a) without skip connections

(b) with skip connections

ResNet-56 Loss Landscape

ResNet-56 loss landscape for CIFAR-10 from [LXT+18]

Exploring Generalization Error with Poisoned Training

- Binary classification experiment from [HDF18]
  - Poisoned loss mixes training loss and test loss with incorrect labels:

$$\mathcal{P}_\beta(\theta) = \frac{1-\beta}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L\big(y, f_\theta(x)\big) + \frac{\beta}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} L\big(1-y, f_\theta(x)\big)$$

  - SGD on $\mathcal{P}_\beta(\theta)$ increases training accuracy but decreases test
  - note: for binary, we use $\mathcal{Y}=\mathbb{R}$ and $L(y,\hat{y}) = y \ln \frac{1}{\hat{y}} + (1-y) \ln \frac{1}{1-\hat{y}}$

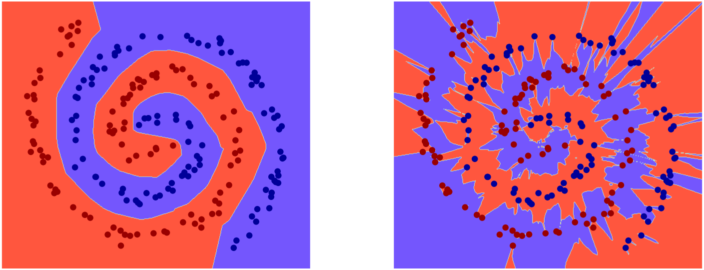
Figures taken from [HDF18]. First figure: red / blue dots show training points. Second figure: pink shows t-sne for $\beta = 0$ SGD.

Exploring Generalization Error with Poisoned Training

- Binary classification experiment from [HDF18]
  - Poisoned loss mixes training loss and test loss with incorrect labels:

$$\mathcal{P}_\beta(\theta) = \frac{1-\beta}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f_\theta(x)) + \frac{\beta}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} L\big(1-y, f_\theta(x)\big)$$

  - SGD on $\mathcal{P}_\beta(\theta)$ increases training accuracy but decreases test
  - note: for binary, we use $\mathcal{Y}=\mathbb{R}$ and $L(y,\hat{y}) = y \ln \frac{1}{\hat{y}} + (1-y) \ln \frac{1}{1-\hat{y}}$



(a) 100% train, 100% test

(b) 100% train, 7% test

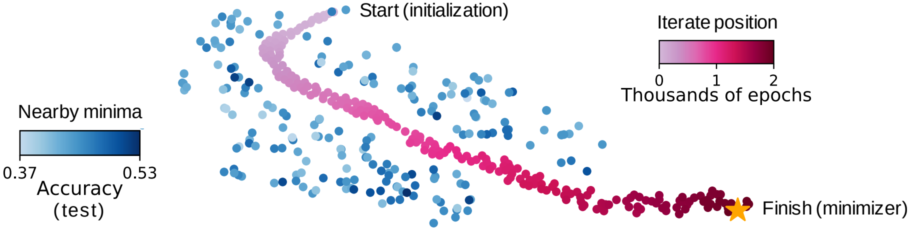
Figures taken from [HDF18]. First figure: red / blue dots show training points. Second figure: pink shows t-sne for $\beta = 0$ SGD.

Exploring Generalization Error with Poisoned Training

- Binary classification experiment from [HDF18]
  - Poisoned loss mixes training loss and test loss with incorrect labels:

$$\mathcal{P}_\beta(\theta) = \frac{1-\beta}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(y, f_\theta(x)) + \frac{\beta}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} L(1-y, f_\theta(x))$$

  - SGD on $\mathcal{P}_\beta(\theta)$ increases training accuracy but decreases test
  - note: for binary, we use $\mathcal{Y}=\mathbb{R}$ and $L(y,\hat{y}) = y \ln \frac{1}{\hat{y}} + (1-y) \ln \frac{1}{1-\hat{y}}$

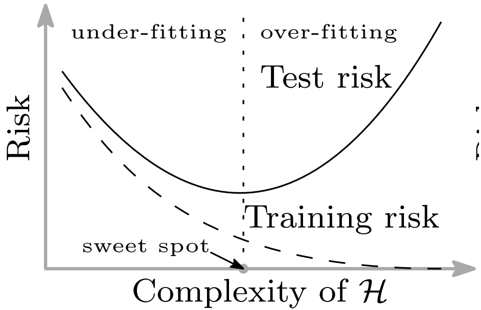


Swiss Roll Training Path

Figures taken from [HDF18]. First figure: red / blue dots show training points. Second figure: pink shows t-sne for $\beta = 0$ SGD.

The Modern Bias-Variance Trade-Off

- The classical bias-variance trade-off is almost right ;-)
  - Classically, the overparametrized regime was considered silly
  - But, gradient descent from a random start implicitly regularizes
  - Except when the #parameters $\approx$ #samples because then the zero-loss solution is almost unique

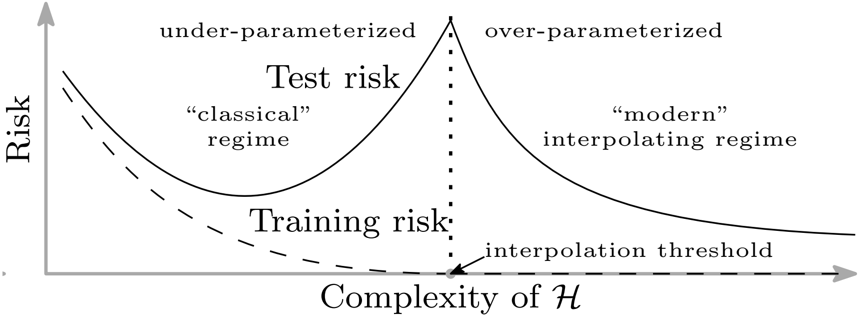


(a) U-shaped "bias-variance" risk curve

Figures from [BHMM18]. Last figure: ReLU NN with single FC hidden layer on MNIST, left = zero-one loss, right = MSE loss

The Modern Bias-Variance Trade-Off

- The classical bias-variance trade-off is almost right ;-)
  - Classically, the overparametrized regime was considered silly
  - But, gradient descent from a random start implicitly regularizes
  - Except when the #parameters $\approx$ #samples because then the zero-loss solution is almost unique

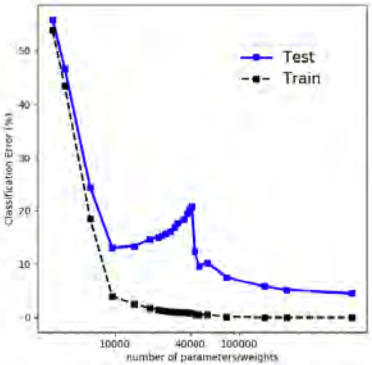


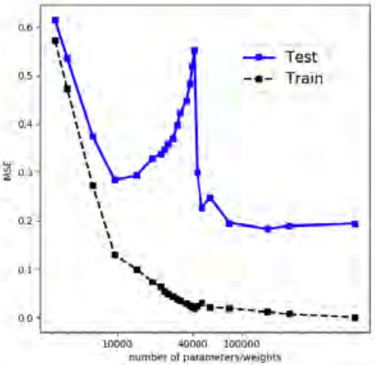
(b) "double descent" risk curve

Figures from [BHMM18]. Last figure: ReLU NN with single FC hidden layer on MNIST, left = zero-one loss, right = MSE loss

The Modern Bias-Variance Trade-Off

- The classical bias-variance trade-off is almost right ;-)
  - Classically, the overparametrized regime was considered silly
  - But, gradient descent from a random start implicitly regularizes
  - Except when the #parameters $\approx$ #samples because then the zero-loss solution is almost unique





Figures from [BHMM18]. Last figure: ReLU NN with single FC hidden layer on MNIST, left = zero-one loss, right = MSE loss

Conclusions

- Overparametrized Neural Networks
  - Can be trained quickly to zero error for any labels
  - (Stochastic) gradient descent provides implicit regularization
  - Not explainable with classical bias-variance trade-off
- The Loss Landscape
  - Visualization via 1D and 2D Projections
  - Low-loss paths exist between most minima
  - Poisoned training to visualize bad minima
- Additional Topics
  - Modern bias-variance trade-Off
  - Connections to generalization error

References I

[ADH+19] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang.
Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks.
arXiv preprint arXiv:1901.08584, 2019.

[AZLL18] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang.
Learning and generalization in overparameterized neural networks, going beyond two layers.
arXiv preprint arXiv:1811.04918, 2018.

[BHMM18] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal.
Reconciling modern machine learning and the bias-variance trade-off.
arXiv preprint arXiv:1812.11118, 2018.

[DLL+18] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai.
Gradient descent finds global minima of deep neural networks.
arXiv preprint arXiv:1811.03804, 2018.

References II

[DVSH18] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht.
Essentially no barriers in neural network energy landscape.
Intl. Conf. on Mach. Learn., 2018.
arXiv preprint arXiv:1803.00885.

[GIP+18] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson.
Loss surfaces, mode connectivity, and fast ensembling of DNNs.
In *Adv. in Neural Inform. Processing Syst.*, pages 8789–8798, 2018.

[HDF18] Eric Huang, Andrew C. Doherty, and Steven Flammia.
Performance of quantum error correction with coherent errors.
arXiv preprint arXiv:1805.08227, 2018.
[Online]. Available: https://arxiv.org/abs/1805.08227.

[KMN+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang.
On large-batch training for deep learning: Generalization gap and sharp minima.
In *Intl. Conf. on Learn. Rep.*, 2017.

References III

[LL18] Yuanzhi Li and Yingyu Liang.
Learning overparameterized neural networks via stochastic gradient descent on structured data.
In *Adv. in Neural Inform. Processing Syst.*, pages 8157–8166, 2018.

[LXT+18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein.
Visualizing the loss landscape of neural nets.
In *Adv. in Neural Inform. Processing Syst.*, pages 6389–6399, 2018.

[SC16] Daniel Soudry and Yair Carmon.
No bad local minima: Data independent training error guarantees for multilayer neural networks.
arXiv preprint arXiv:1605.08361, 2016.

[ZBH+16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
arXiv preprint arXiv:1611.03530, 2016.