# Tutorial
# Generalization and Overfitting: Basic Concepts and Open Questions

Sasha Rakhlin

Jan 7, 2019

# Outline

# Outline

Feed-forward neural network:

$$f(x) = W^L \sigma(W^{L-1} \sigma(\dots \sigma(W^1 x) \dots))$$

where $\sigma$ is applied coordinate-wise. E.g. ReLU $\sigma(a) = \max\{a, 0\}$.

Empirical loss

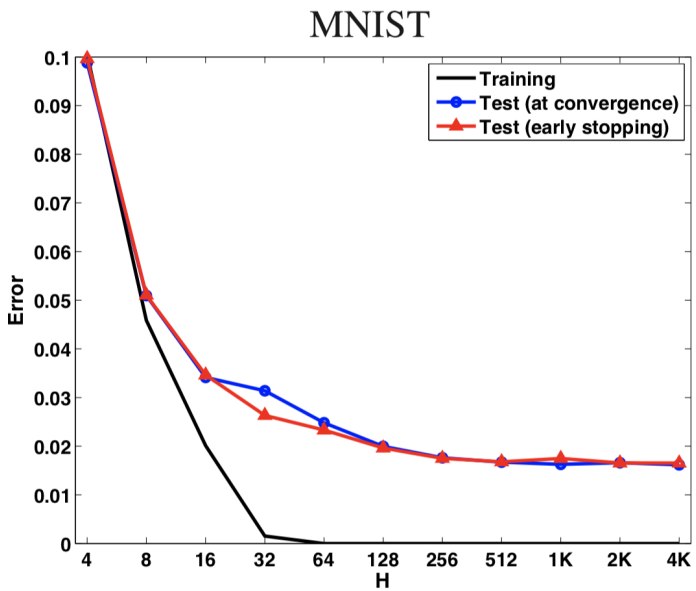
$$\widehat{\mathbf{L}}(f) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp\{-Y_i f(X_i)\})$$

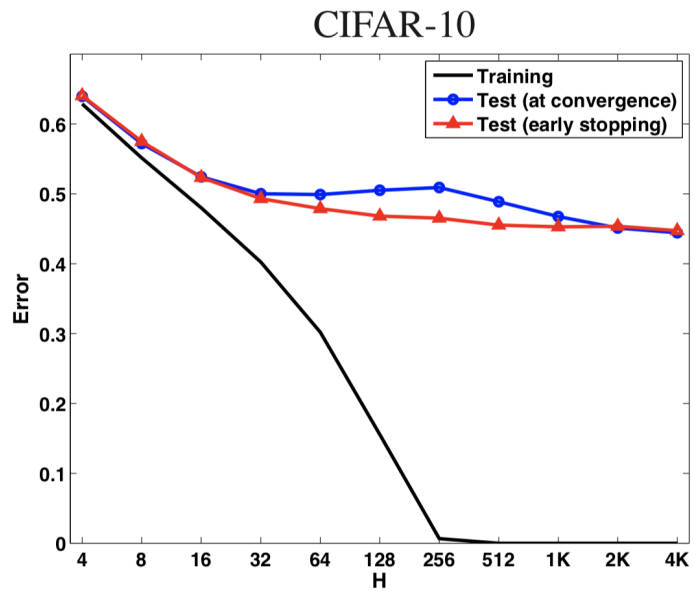
- Generalization
- Optimization
- Expressive power

Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset. Performance with and without data augmentation and weight decay are compared. The results of fitting random labels are also included.

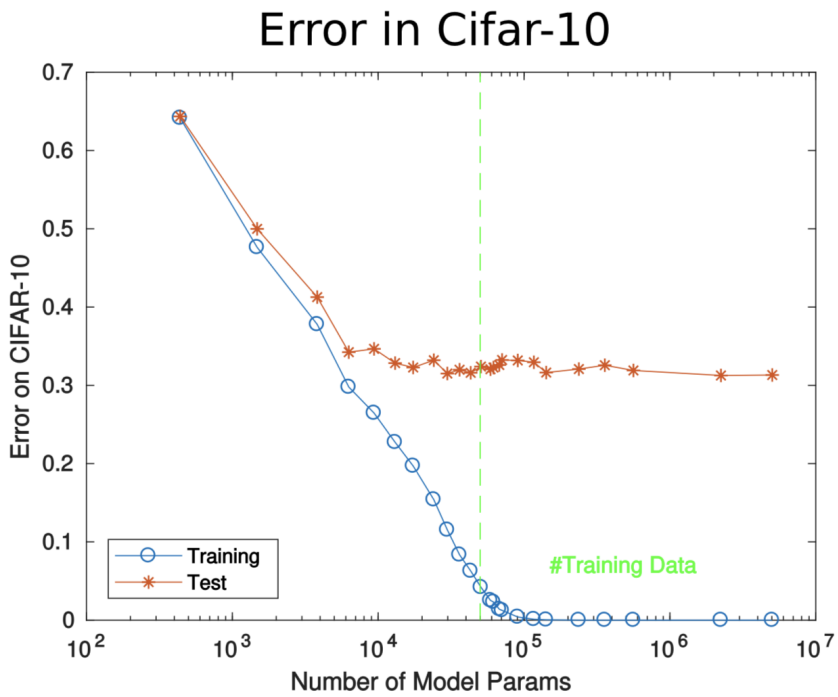
| model | # params | random crop | weight decay | train accuracy | test accuracy |
|---|---|---|---|---|---|
| Inception | 1,649,402 | yes | yes | 100.0 | 89.05 |
| | | yes | no | 100.0 | 89.31 |
| | | no | yes | 100.0 | 86.03 |
| | | no | no | 100.0 | 85.75 |
| (fitting random labels) | | no | no | 100.0 | 9.78 |
| Inception w/o BatchNorm | 1,649,402 | no | yes | 100.0 | 83.00 |
| | | no | no | 100.0 | 82.00 |
| (fitting random labels) | | no | no | 100.0 | 10.12 |
| Alexnet | 1,387,786 | yes | yes | 99.90 | 81.22 |
| | | yes | no | 99.82 | 79.66 |
| | | no | yes | 100.0 | 77.36 |
| | | no | no | 100.0 | 76.07 |
| (fitting random labels) | | no | no | 99.82 | 9.86 |
| MLP 3x512 | 1,735,178 | no | yes | 100.0 | 53.35 |
| | | no | no | 100.0 | 52.39 |
| (fitting random labels) | | no | no | 100.0 | 10.48 |
| MLP 1x512 | 1,209,866 | no | yes | 99.80 | 50.39 |
| | | no | no | 100.0 | 50.51 |
| (fitting random labels) | | no | no | 99.34 | 10.61 |

(Zhang et al 2017)




(Neyshabur et al 2015)




(Poggio et al 2017)

Unclear if worse performance for smaller networks is due to inability to find a good solution in the empirical landscape, or the model is not rich enough.

# Outline

Supervised Learning: data $\mathcal{S} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ are i.i.d. from ***unknown*** distribution $P$ on $\mathcal{X} \times \mathcal{Y}$.

*Learning algorithm*: a mapping $\{(X_1, Y_1), \ldots, (X_n, Y_n)\} \longmapsto \widehat{f}_n \in \mathcal{Y}^{\mathcal{X}}$

***Important***: we are not necessarily modeling the distribution of $(X, Y)$.

Goals:

- **Prediction:** small expected loss

$$\mathbf{L}(\widehat{f}_n) = \mathbb{E}\left[\ell(Y, \widehat{f}_n(X)) \,\middle|\, \mathcal{S}\right].$$

  Prediction on a random example from same population. Approximated by average on fresh test sample.

- **Estimation:** small $\left\|\widehat{f}_n - f^*\right\|$, or $\left\|\widehat{\theta} - \theta^*\right\|$, where $f^*$ or $\theta^*$ are parameters or functionals of $P$ (e.g. regression function $f^*(x) = \mathbb{E}[Y|X = x]$, or $f^*(x) = \langle \theta^*, x \rangle$, etc).

Why not estimate the underlying distribution $P$ (or, say, class-conditional $P(X|Y = y)$) first? Easier to directly estimate decision boundary.

# The Gold Standard

*Bayes optimal* function

$$f^* \in \arg \min_{f:\mathcal{X}\to\mathcal{Y}} \mathbf{L}(f).$$

*Bayes error*:

$$\mathbf{L}(f^*) = \inf_{f:\mathcal{X}\to\mathcal{Y}} \mathbf{L}(f)$$

Of course, we cannot calculate any of these quantities since $P$ is unknown.

# Bayes Optimal Function

Bayes optimal function $f^*$ takes on the following forms:

- Binary classification ($\mathcal{Y} = \{0, 1\}$) with the indicator loss:

$$f^*(x) = \mathbf{I}\{\eta(x) \geq 1/2\}, \quad \text{where} \quad \eta(x) = \mathbb{E}[Y|X = x]$$

- Regression ($\mathcal{Y} = \mathbb{R}$) with squared loss:

$$f^*(x) = \eta(x), \quad \text{where} \quad \eta(x) = \mathbb{E}[Y|X = x]$$

# Consistency and rates

Consistency:

$$\lim_{n\to\infty} \mathbf{L}(\widehat{f}_n) = \mathbf{L}(f^*) \qquad \text{almost surely}$$

**Good news:** consistency is possible under minimal assumptions.
**Bad news:** no nontrivial rate is possible unless we place strong(er) assumptions on the distribution.

**Goal:** weak and "reasonable" assumptions on underlying distribution that still yield "rates" (in terms of $n$) for

$$\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*).$$

Keep in mind: in neural networks, model grows with $n$ too.

# Generalization

"Generalization" in the literature may refer to one of

- Small $\mathbf{L}(\widehat{f}_n)$
- Small $\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*)$
- Small $\mathbf{L}(\widehat{f}_n) - \widehat{\mathbf{L}}(\widehat{f}_n)$ ($a$ *posteriori* interpretation)

# Outline

Suppose $\|x_t\| = 1$, $y_t \in \pm 1$, and indicator loss $\ell(y, y') = \mathbf{I}\{y \neq y'\}$.

Margin w.r.t. $(x_1, y_1), \ldots, (x_T, y_T)$:

$$\gamma = \left[ \max_{\|w\|=1} \min_{i \in [T]} y_i \langle w, x_i \rangle \right] \vee 0.$$

Claim: several procedures $\widehat{f}_n$ enjoy

$$\mathbb{E}\mathbf{L}_{01}(\widehat{f}_n) - \underbrace{\mathbf{L}_{01}(w^*)}_{0} \leq \frac{1}{n} \times \mathbb{E}\left[\gamma^{-2}\right]$$

Remarks: (a) dim-independent, (b) $\exists$ extension to "small" nonzero $\mathbf{L}_{01}(w^*)$.

Assumption is not about parametric or nonparametric form of $\mathsf{P}$, but rather on what happens at the boundary.

# Perceptron

$(x_1, y_1), \ldots, (x_T, y_T) \quad \in \mathcal{X} \times \{\pm 1\}$ ($T$ may or may not be same as $n$)

Maintain a hypothesis $w_t \in \mathbb{R}^d$ (initialize $w_1 = \mathbf{0}$).

On round $t$,

- Consider $(x_t, y_t)$
- Form prediction $\widehat{y}_t = \text{sign}(\langle w_t, x_t \rangle)$
- If $\widehat{y}_t \neq y_t$, update

$$w_{t+1} = w_t + y_t x_t$$

else

$$w_{t+1} = w_t$$

**Theorem (Novikoff '62):** Perceptron makes at most $\gamma^{-2}$ mistakes (and corrections) on **any** sequence of examples with margin $\gamma$.

# Consequence for i.i.d. data (I)

*One pass* on i.i.d. sequence $(X_1, Y_1), \ldots, (X_n, Y_n)$ (i.e. $T = n$).

**Claim:** For $\tau \sim \text{unif}(1, \ldots, n)$,

$$\mathbb{E}\mathbf{L}_{01}(w_\tau) \leq \frac{1}{n} \times \mathbb{E}\left[\gamma^{-2}\right].$$

# Consequence for i.i.d. data (II)

Cycle through data $(X_1, Y_1), \ldots, (X_n, Y_n)$ repeatedly until no more mistakes (i.e. $T \leq n\gamma^{-2}$).

Then **final** hyperplane of Perceptron separates the data perfectly, i.e. finds empirical risk minimizer (ERM), $\widehat{\mathbf{L}}_{01}(w_T) = 0$.

**Claim:**

$$\mathbb{E}\mathbf{L}_{01}(w_T) \leq \frac{1}{n} \times \mathbb{E}\left[\gamma^{-2}\right]$$

**Proof:** Shortcuts: $z = (x, y)$ and $\ell(w, z) = \mathbf{I}\{y \langle w, x \rangle \leq 0\}$. Then

$$\mathbb{E}_S \mathbb{E}_Z \ell(w_T, Z) = \mathbb{E}_{S, Z_{n+1}} \left[ \frac{1}{n+1} \sum_{t=1}^{n+1} \ell(w^{(-t)}, Z_t) \right]$$

where $w^{(-t)}$ is Perceptron's final hyperplane after cycling through data $Z_1, \ldots, Z_{t-1}, Z_{t+1}, \ldots, Z_{n+1}$.

That is, *leave-one-out* is unbiased estimate of expected loss.

Now consider cycling Perceptron on $Z_1, \ldots, Z_{n+1}$ until no more errors. Let $i_1, \ldots, i_m$ be indices on which Perceptron errs in *any* of the cycles. We know $m \leq \gamma^{-2}$. However, if index $t \notin \{i_1, \ldots, i_m\}$, then whether or not $Z_t$ was included does not matter, and $Z_t$ is correctly classified by $\mathbf{w}^{(-t)}$. Claim follows.

# SGD vs Multi-Pass

Perceptron update can be seen as gradient descent step with respect to loss

$$\max\left\{-Y_t\left\langle w, X_t\right\rangle, 0\right\}$$

and step size 1.

The two consequences presented earlier can be viewed, resp., as SGD on

$$\mathbb{E}\max\left\{-Y\left\langle w, X\right\rangle, 0\right\}$$

and multi-pass cycling "SGD" on

$$\frac{1}{n}\sum_{t=1}^{n}\max\left\{-Y_t\left\langle w, X_t\right\rangle, 0\right\}.$$

The first optimizes expected loss, the second optimizes empirical loss.

# A few observations

Much hype in ML community is about surprising performance of deep networks *despite* (a) overparametrization and (b) fitting data perfectly.

(a) We already see from the Perceptron example that number of parameters is not necessarily the only complexity notion (dimension $\mathtt{d}$ never appears and we can take $\mathtt{d} = \infty$).

(b) In high enough dimension, any $\mathtt{n}$ points in general position are linearly separable (hence, zero empirical loss for multi-cycle Perceptron). Perfectly fitting the data does not necessarily contradict good generalization.

# Outline

# Bias-Variance Tradeoff

Goal:

$$\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*)$$

In simple problems we might get away with having no bias-variance decomposition (e.g. as we just saw in linearly separable case).

For more complex problems (weak assumptions on $P$, nonparametrics), we need to stratify the space according to some "complexity" and resort to a bias-variance decomposition.

# Bias-Variance in Stat/ML

Bias-variance decomposition often studied in ML:

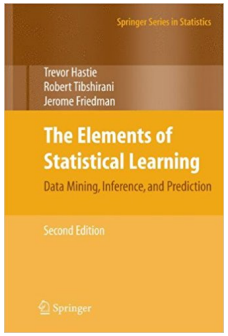
$$\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*) = \underbrace{\mathbf{L}(\widehat{f}_n) - \inf_{f \in \mathcal{F}_n} \mathbf{L}(f)}_{\text{Estimation Error}} + \underbrace{\inf_{f \in \mathcal{F}_n} \mathbf{L}(f) - \mathbf{L}(f^*)}_{\text{Approximation Error}}$$

Bias-variance decomposition often studied in Statistics (square loss):

$$\mathbb{E}\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*) = \mathbb{E}\left\|\widehat{f}_n - f^*\right\|^2_{L_2(P_X)} = \mathbb{E}\left\|\widehat{f}_n - \mathbb{E}\widehat{f}_n\right\|^2_{L_2(P_X)} + \mathbb{E}\left\|\mathbb{E}\widehat{f}_n - f^*\right\|^2_{L_2(P_X)}$$

Trade-off is parametrized by tunable parameter(s) of the procedure.

# Classical picture: U-shaped curve



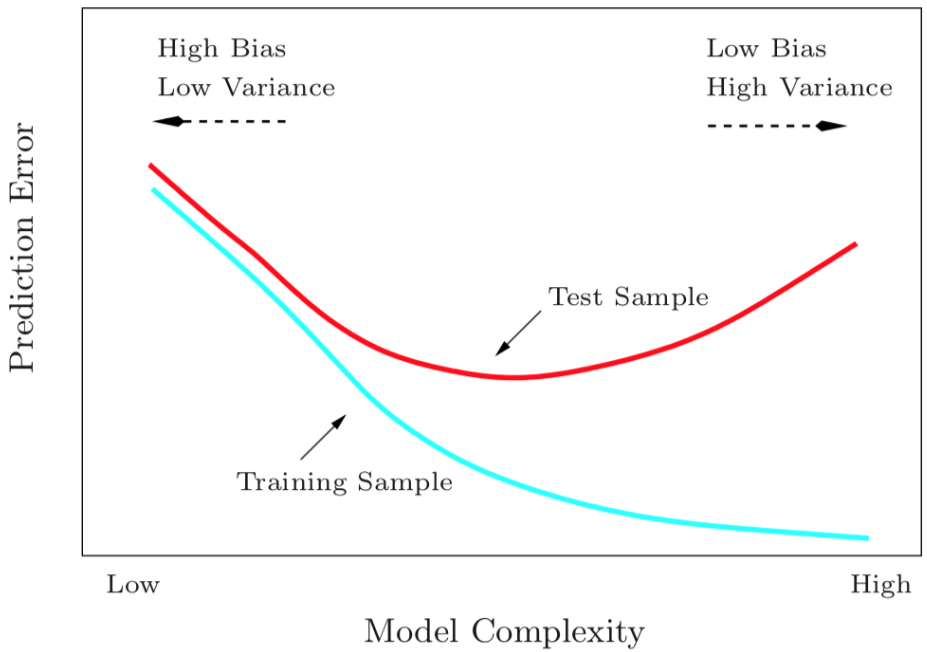




**FIGURE 2.11.** *Test and training error as a function of model complexity.*

Remark: bias-variance curve for neural networks appears to be more complex.

# Estimation error and inductive bias

If we guessed correctly and $f^* \in \mathcal{F}$, then

$$\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*) = \mathbf{L}(\widehat{f}_n) - \inf_{f \in \mathcal{F}} \mathbf{L}(f)$$

$\mathcal{F}$ as *inductive bias*: one hopes that prior knowledge about the problem ensures that approximation error $\inf_{f \in \mathcal{F}} \mathbf{L}(f) - \mathbf{L}(f^*)$ is small.

Most emphasis is on

$$\mathbf{L}(\widehat{f}_n) - \inf_{f \in \mathcal{F}_n} \mathbf{L}(f)$$

Typically:

$$\mathcal{F}_n = \{f_\theta : \mathrm{comp}(\theta) \leq B_n\}$$

Remark: not too clear what $\mathrm{comp}(\theta)$ should be for neural nets.

# Outline

For Empirical Risk Minimizer

$$\widehat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} \widehat{\mathbf{L}}(f),$$

it holds that

$$\mathbb{E}\mathbf{L}(\widehat{f}_n) - \inf_{f \in \mathcal{F}} \mathbf{L}(f) \leq \mathbb{E} \sup_{f \in \mathcal{F}} \left[ \mathbf{L}(f) - \widehat{\mathbf{L}}(f) \right].$$

Shorthand: $z = (x, y)$, $g(z) = \ell(y, f(x))$, $\mathcal{G} = \ell \circ \mathcal{F}$.

Empirical process:

$$g \mapsto \frac{1}{\sqrt{n}} \sum_{i=1}^{n} [g(Z_i) - \mathbb{E}g]$$

Rademacher/Bernoulli process:

$$g \mapsto \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \epsilon_i g(Z_i)$$

where $\epsilon_i$ independent symmetric $\pm 1$ random variables.

It holds that

$$\mathbb{E} \sup_{g \in \mathcal{G}} \sum_{i=1}^{n} [g(Z_i) - \mathbb{E}g] \leq 2\mathbb{E} \sup_{g \in \mathcal{G}} \sum_{i=1}^{n} \epsilon_i g(Z_i)$$

and the other direction holds (with additive factor if $\mathcal{G}$ is not symmetric).

Crucially, we can study the Rademacher process conditionally on $Z_i$'s.

# Vapnik-Chervonenkis

If $\mathcal{G} \subseteq \{\pm 1\}^{\mathcal{Z}}$, empirical Rademacher averages

$$\widehat{\mathscr{R}}_n(\mathcal{G}) \triangleq \mathbb{E}\left[\sup_{g \in \mathcal{G}} \sum_{i=1}^{n} \epsilon_i g(Z_i) \;\middle|\; Z_{1:n}\right] \lesssim \sqrt{\frac{\mathrm{VCdim}(\mathcal{G})}{n}}$$

# Example: Neural Nets

Fix architecture of a neural network with $L$ layers, $K$ parameters, and ReLU activation (this defines $\mathcal{F}$). (Bartlett et al '17): VC dimension of $\text{sign}(\mathcal{F})$ is $O(KL \log K)$, tight up to logs.

VC bound is vacuous when number of parameters is large. Not a good notion of complexity (see Bartlett et al '98)

Margin analysis comes to the rescue.

# Margin analysis (e.g. Koltchinskii-Panchenko'02)

Fix $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$. With probability at least $1-\delta$, for all $f \in \mathcal{F}$,

$$\mathbf{L}_{01}(f) \leq \underbrace{\frac{1}{n}\sum_{i=1}^{n} \mathbf{I}\{Y_i f(X_i) \leq \gamma\}}_{\gamma\text{-margin mistakes}} + \frac{c}{\gamma}\widehat{\mathscr{R}}_n(\mathcal{F}) + O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right)$$

Focus lifted from understanding sign patterns to understanding real-valued behavior. Example: for $\mathcal{F}_{\text{lin}} = \{x \mapsto \langle w, x\rangle : \|w\| \leq B\}$ and $\|x_i\| \leq 1$,

$$\widehat{\mathscr{R}}_n(\mathcal{F}_{\text{lin}}) \leq \frac{B}{\sqrt{n}}$$

- Dimension-independent for $\mathcal{F}_{\text{lin}}$.
- Union bound over shells of increasing complexity.
- *A posteriori* interpretation.

# Complexity for Neural Nets?

Just as in the case of $\mathcal{F}_{\text{lin}}$, we would like to define a "ball" in the space of neural networks:

$$\{f_\theta : \text{comp}(\theta) \leq B\}$$

for some notion of complexity comp.

Example:

$$\text{comp}(\theta) = \sum_{j=1}^{L} \left\| W^j \right\|_F$$

Not invariant under layer scaling. Next try:

$$\text{comp}(\theta) = \prod_{j=1}^{L} \left\| W^j \right\|_F$$

See (Neyshabur and Srebro '15), (Neyshabur at al '17), (Bartlett et al '17), and references therein for more variations.

# Complexity for Neural Nets?

(Golowich et al '18): For a class of neural networks with $\prod_{j=1}^{L} \|W^j\|_F \leq B$, Rademacher averages bounded by

$$\tilde{O}\left(\min\left\{ \frac{B}{n^{1/4}} \ , \ \frac{B\sqrt{L}}{n^{1/2}} \right\}\right) .$$

As close to Perceptron analogue as we can get at this point.

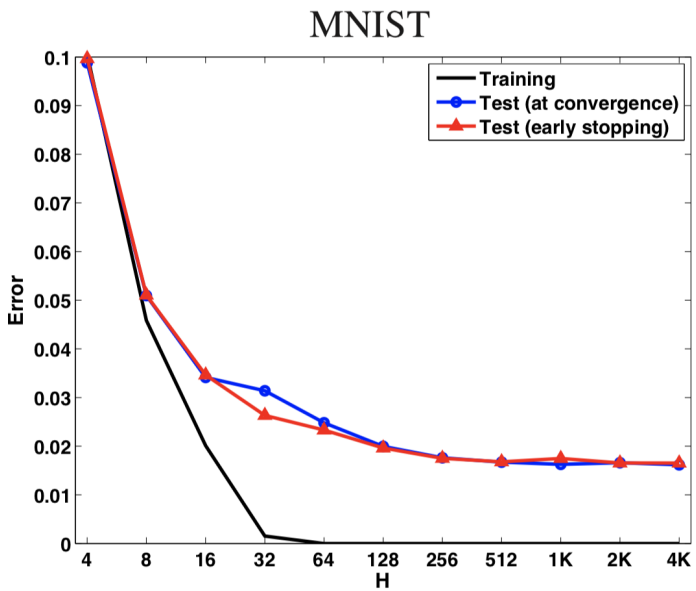
But does SGD on neural networks control this measure of complexity?

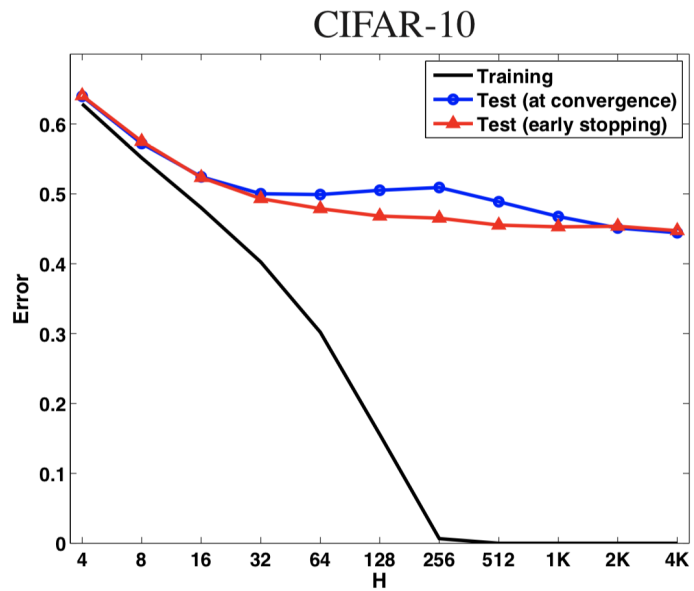
# Outline

Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset. Performance with and without data augmentation and weight decay are compared. The results of fitting random labels are also included.

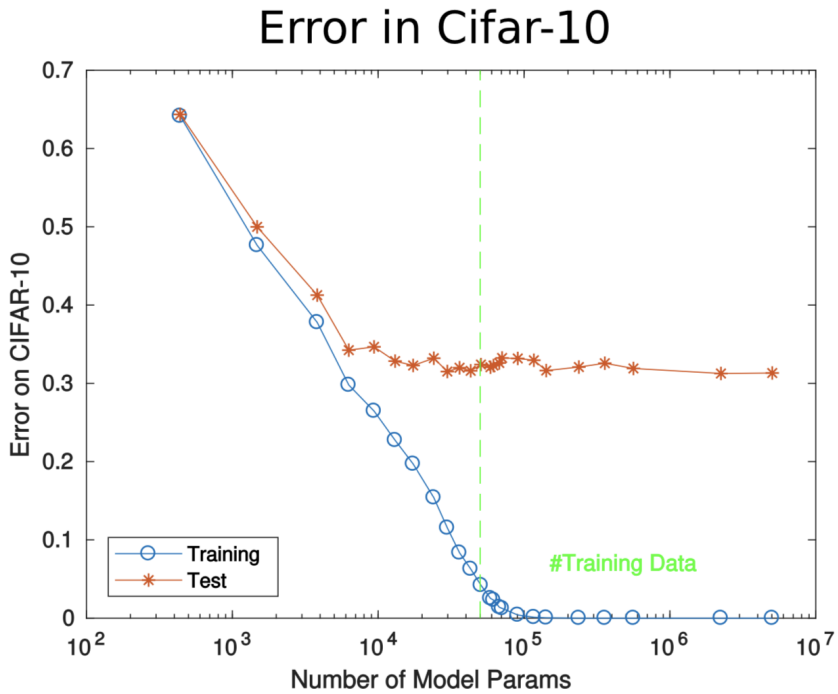
| model | # params | random crop | weight decay | train accuracy | test accuracy |
|---|---|---|---|---|---|
| Inception | 1,649,402 | yes | yes | 100.0 | 89.05 |
| | | yes | no | 100.0 | 89.31 |
| | | no | yes | 100.0 | 86.03 |
| | | no | no | 100.0 | 85.75 |
| (fitting random labels) | | no | no | 100.0 | 9.78 |
| Inception w/o BatchNorm | 1,649,402 | no | yes | 100.0 | 83.00 |
| | | no | no | 100.0 | 82.00 |
| (fitting random labels) | | no | no | 100.0 | 10.12 |
| Alexnet | 1,387,786 | yes | yes | 99.90 | 81.22 |
| | | yes | no | 99.82 | 79.66 |
| | | no | yes | 100.0 | 77.36 |
| | | no | no | 100.0 | 76.07 |
| (fitting random labels) | | no | no | 99.82 | 9.86 |
| MLP 3x512 | 1,735,178 | no | yes | 100.0 | 53.35 |
| | | no | no | 100.0 | 52.39 |
| (fitting random labels) | | no | no | 100.0 | 10.48 |
| MLP 1x512 | 1,209,866 | no | yes | 99.80 | 50.39 |
| | | no | no | 100.0 | 50.51 |
| (fitting random labels) | | no | no | 99.34 | 10.61 |

(Zhang et al 2017)




(Neyshabur et al 2015)

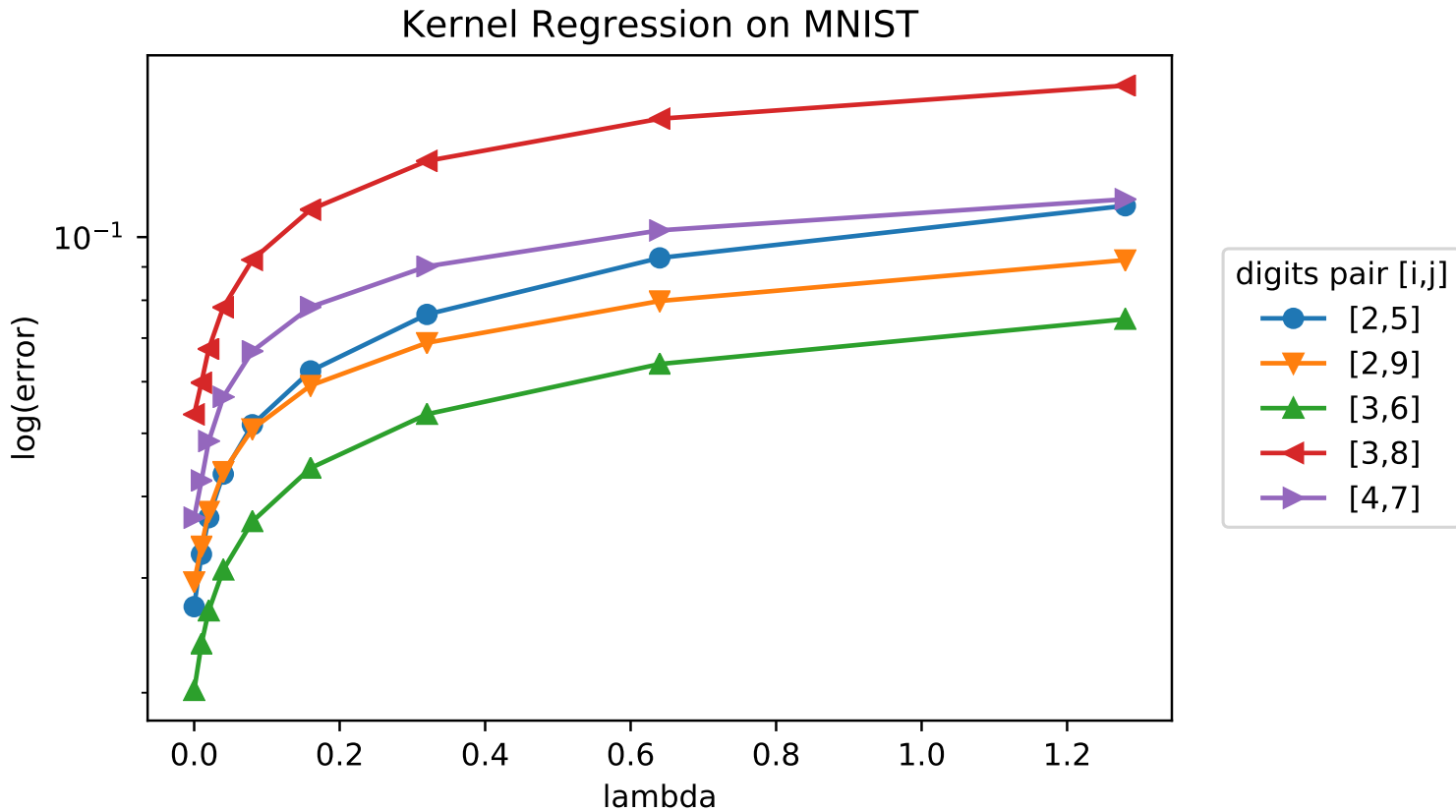



(Poggio et al 2017)

Can a learning method be successful if it memorizes the training data?

Linear separators with margin is one example. However, it is still in the "uniform convergence" regime. To avoid "uniform convergence" arguments, we consider the second style of bias-variance trade-off.

# An empirical example




$\lambda = 0$: the interpolated solution, perfect fit on training data.

# Isolated phenomenon? No

## UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION


**Chiyuan Zhang***
Massachusetts Institute of Technology
chiyuan@mit.edu

**Samy Bengio**
Google Brain
bengio@google.com

**Moritz Hardt**
Google Brain
mrtz@google.com

**Benjamin Recht**†
University of California, Berkeley
brecht@berkeley.edu

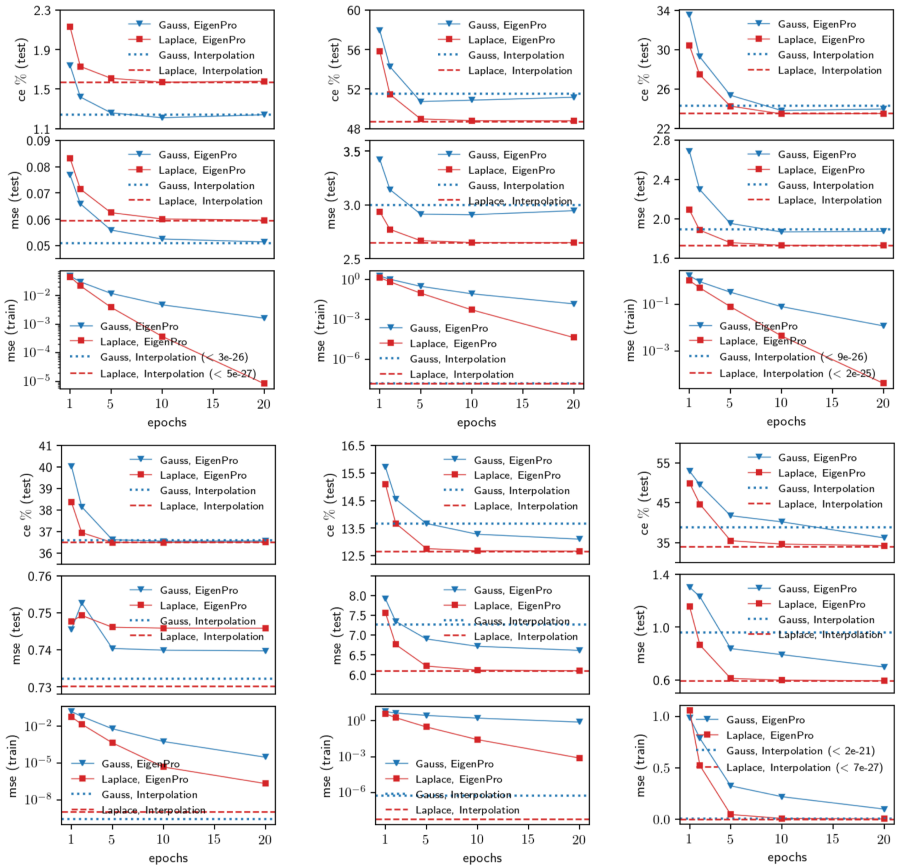
**Oriol Vinyals**
Google DeepMind
vinyals@google.com


Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset. Performance with and without data augmentation and weight decay are compared. The results of fitting random labels are also included.

| model | # params | random crop | weight decay | train accuracy | test accuracy |
|---|---|---|---|---|---|
| Inception | 1,649,402 | yes | yes | 100.0 | 89.05 |
| | | yes | no | 100.0 | 89.31 |
| | | no | yes | 100.0 | 86.03 |
| | | no | no | 100.0 | 85.75 |
| (fitting random labels) | | no | no | 100.0 | 9.78 |
| Inception w/o BatchNorm | 1,649,402 | no | yes | 100.0 | 83.00 |
| | | no | no | 100.0 | 82.00 |
| (fitting random labels) | | no | no | 100.0 | 10.12 |
| Alexnet | 1,387,786 | yes | yes | 99.90 | 81.22 |
| | | yes | no | 99.82 | 79.66 |
| | | no | yes | 100.0 | 77.36 |
| | | no | no | 100.0 | 76.07 |
| (fitting random labels) | | no | no | 99.82 | 9.86 |
| MLP 3x512 | 1,735,178 | no | yes | 100.0 | 53.35 |
| | | no | no | 100.0 | 52.39 |
| (fitting random labels) | | no | no | 100.0 | 10.48 |
| MLP 1x512 | 1,209,866 | no | yes | 99.80 | 50.39 |
| | | no | no | 100.0 | 50.51 |
| (fitting random labels) | | no | no | 99.34 | 10.61 |

To Understand Deep Learning We Need to Understand Kernel Learning


Mikhail Belkin, Siyuan Ma, Soumik Mandal
Department of Computer Science and Engineering
Ohio State University
*{mbelkin, masi}@cse.ohio-state.edu, mandal.32@osu.edu*


(d) TIMIT ($5 \cdot 10^4$ subsamples) (e) HINT-S ($2 \cdot 10^4$ subsamples) (f) 20 Newsgroups

- AdaBoost, random forests (Schapire et al. 1998, Wyner et al. 2017)
- Deep learning, kernel learning (Zhang et al. 2016, Belkin, Ma & Mandal 2018)
- Datasets: MNIST, CIFAR-10, etc (Belkin, Ma & Mandal 2018)

# History: interpolating rules in Stat/ML

Very few existing results (especially in regression).

Recent progress on local/direct interpolation schemes:

- Geometric simplicial interpolation and weighted kNN (Belkin, Hsu & Mitra 2018)
- Nonparametric Nadaraya-Watson estimator with singular kernels (Shepard 1968, Devroye et al. 1998, Belkin, Rakhlin & Tsybakov 2018)

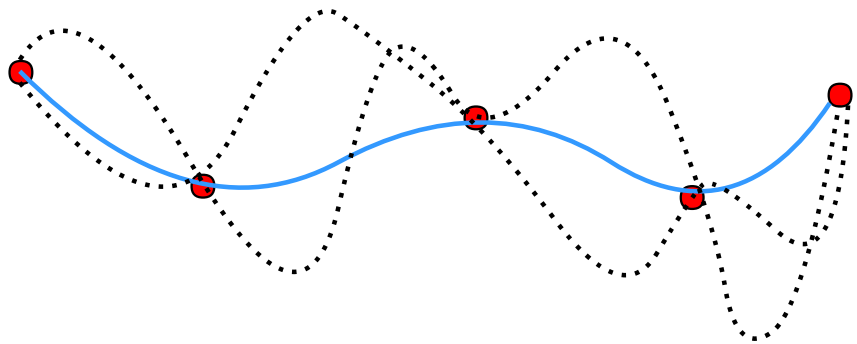
Recent progress on inverse interpolation schemes:

- Kernel ridgeless regression (Liang & Rakhlin 2018)

Lots of interpolation work in *approximation theory*, but results are "up to noise level."

Min-norm interpolation:

$$\widehat{f}_n := \operatorname*{argmin}_{f \in \mathcal{H}} \|f\|_{\mathcal{H}}, \quad \text{s.t.} \quad f(x_i) = y_i, \ \forall i \leq n \ .$$



Note: GD/SGD started from 0 converges to minimum-norm solution.

# Implicit regularization

**Geometric properties of the data design** $X$**, high dimensionality**, and **curvature of the kernel** $\Rightarrow$ interpolated solution generalizes.

# Closed-form solution

By the Representer Theorem, min-norm interpolant is

$$\widehat{f}_n(x) = K(x, X)K(X, X)^{-1}Y$$

when $K(X, X) \in \mathbb{R}^{n \times n}$ is invertible.

Difficulty in analyzing bias/variance of $\widehat{f}_n$: it is not clear how the random matrix $K^{-1}$ "aggregates" $Y$'s between data points.

# Effective dimension

Let $\lambda_1, \ldots,$ be eigenvalues of $XX^*$. Define

$$\dim(XX^*) = \sum_j \frac{\lambda_j\left(\frac{XX^*}{d}\right)}{\left[\gamma + \lambda_j\left(\frac{XX^*}{d}\right)\right]^2}$$

Compare with regularized least squares as low pass filter:

$$\dim(XX^*) = \sum_j \frac{\lambda_j\left(\frac{XX^*}{d}\right)}{\lambda + \lambda_j\left(\frac{XX^*}{d}\right)}$$

# Lower-bounds: high-dim phenomenon (work with X. Zhai)

Take Laplace kernel

$$K_\sigma(x, x') = \sigma^{-d} \exp\{-\|x - x'\| / \sigma\}$$

$\widehat{f}_n$ is minimum norm interpolation, as before.

Theorem: for odd dimension $d$, with probability $1 - O(n^{-1/2})$, for any choice of $\sigma$,

$$\mathbb{E} \left\| \widehat{f}_n - f^* \right\|^2 \geq \Omega_d(1).$$

Hence, interpolation with Laplace kernel does not work in constant $d$.

# Outline

What is *overfitting*?

- Fitting data too well?
  - In-sample loss is much smaller than out of sample?
- Model too complex?
  - Bias low, variance high?

Key takeaway: we should not conflate these.

# Overfitting

From Wikipedia, the free encyclopedia

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(August 2017)*
*(Learn how and when to remove this template message)*

In statistics, **overfitting** is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably".[1] An **overfitted model** is a statistical model that contains more parameters than can be justified by the data.[2] The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e. the noise) as if that variation represented underlying model structure.[3]:45

**Underfitting** occurs when a statistical model cannot adequately capture the underlying structure of the data. An **underfitted model** is a model where some parameters or terms that would appear in a correctly specified model are missing.[2] Underfitting would occur, for example, when fitting a linear model to non-linear data. Such a model will tend to have poor predictive performance.

Overfitting and underfitting can occur in machine learning, in particular. In machine learning, the phenomena are sometimes called "overtraining" and "undertraining".

The possibility of overfitting exists because the criterion used for selecting the model is not the same as the criterion used to judge the suitability of a model. For example, a model might be selected by maximizing its performance on some set of training data, and yet its suitability might be determined by its ability to perform well on unseen data; then overfitting occurs when a model begins to "memorize" training data rather than "learning" to generalize from a trend.

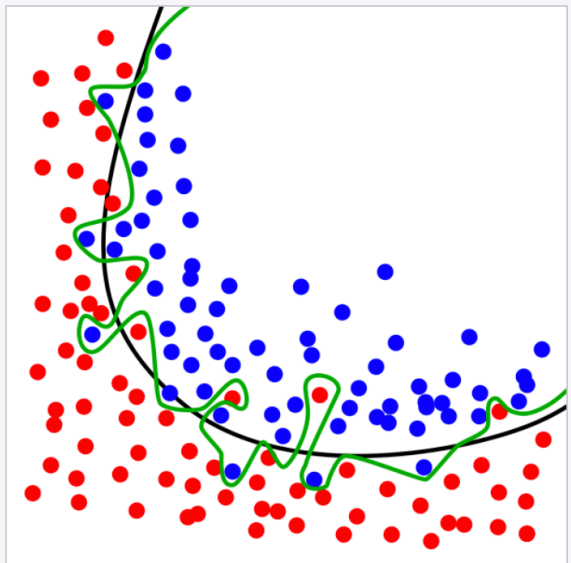
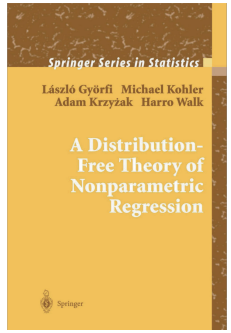


Figure 1. The green line represents an overfitted model and the black line represents a regularized model. While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data, compared to the black line.

# Conventional wisdom in stat/ml




22 2. How to Construct Nonparametric Regression Estimates?

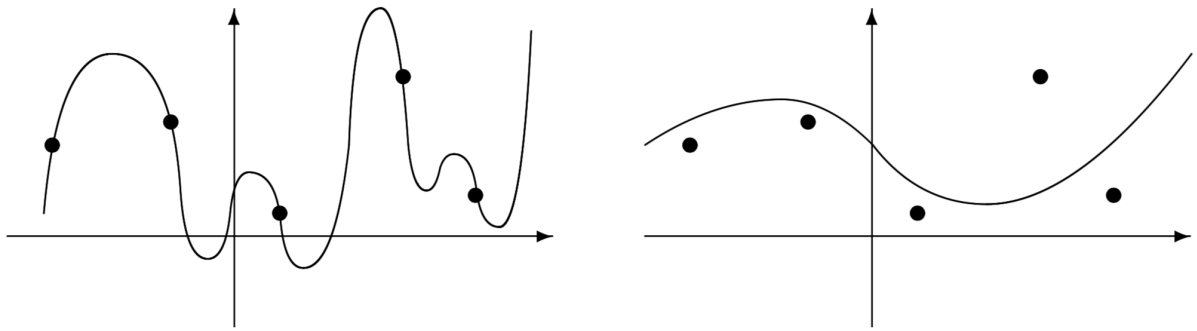
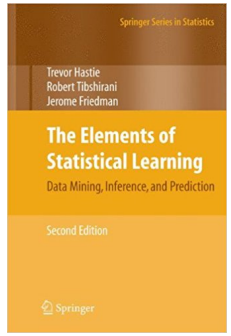


Figure 2.3. The estimate on the right seems to be more reasonable than the estimate on the left, which interpolates the data.

over $\mathcal{F}_n$. Least squares estimates are defined by minimizing the empirical $L_2$ risk over a general set of functions $\mathcal{F}_n$ (instead of (2.7)). Observe that it doesn't make sense to minimize (2.9) over all (measurable) functions $f$, because this may lead to a function which interpolates the data and hence is not a reasonable estimate. Thus one has to restrict the set of functions over

# Conventional wisdom in stat/ml



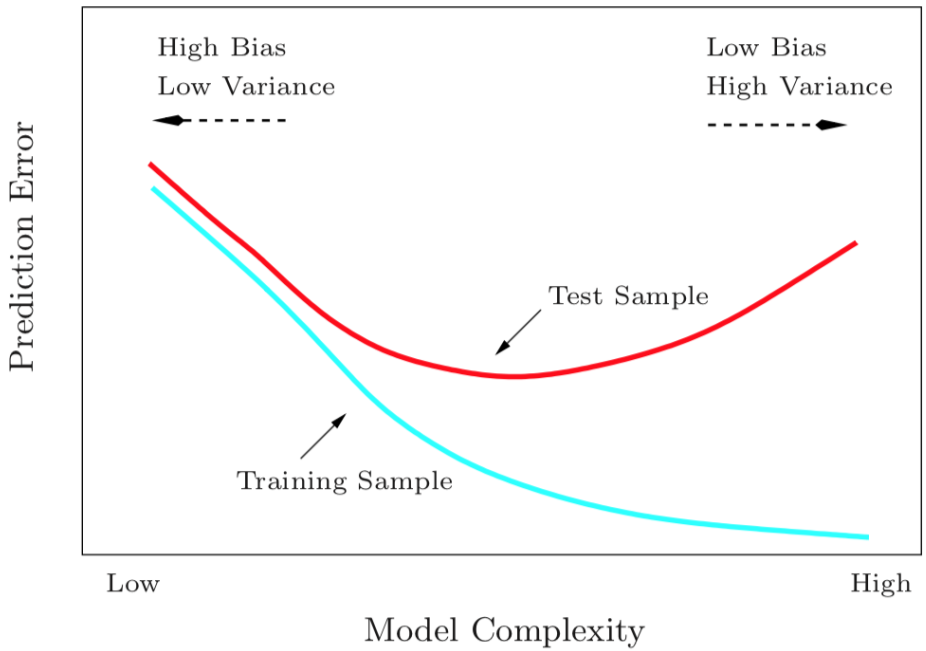
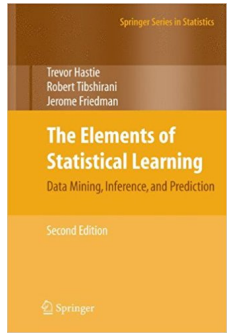




**FIGURE 2.11.** *Test and training error as a function of model complexity.*

Figure 2.11 shows the typical behavior of the test and training error, as model complexity is varied. The training error tends to decrease whenever we increase the model complexity, that is, whenever we fit the data harder. However with too much fitting, the model adapts itself too closely to the training data, and will not generalize well (i.e., have large test error). In

# Conventional wisdom in stat/ml




as seen in Figure 7.1. Training error consistently decreases with model complexity, typically dropping to zero if we increase the model complexity enough. However, a model with zero training error is overfit to the training data and will typically generalize poorly.

The story is similar for a qualitative or categorical response $G$ taking

# Conventional wisdom in stat/ml

Wyner et al. (2017):

> situations. Indeed, common lore in statistical learning suggests that perfectly fitting the training data must inevitably lead to "overfitting." This aversion is built into the DNA of a statistician who has been trained to believe, axiomatically, that data can always be decomposed into signal and noise. Traditionally, the "signal" is always modeled smoothly. The resulting residuals represent the "noise" or the random component in the data. The statistician's art is to walk the balance between the signal and the noise, extracting as much signal as possible without extending the fit to the noise. In this light, it is counterintuitive that any classifier can ever be successful if every training example is "interpolated" by the algorithm and thus fit without error.
>
> The computer scientist, on the other hand, does not automatically decompose problems into signal and noise. In many classical problems, like image detection, there is no noise in the classical sense. Instead there are only complex signals. There are still residuals, but they do not represent irreducible random errors. If the task is to classify images into those with cats and without, the problem is hard not because it is noisy. There are no cats wearing dog disguises. Consequently, the computer scientist has no dogmatic aversion to interpolating training data. This was the breakthrough.

# Outline

Observe: 1-Nearest Neighbor is interpolating. However, one cannot guarantee $\mathbb{E}\mathbf{L}(\widehat{f}_n) - \mathbf{L}(f^*)$ small.

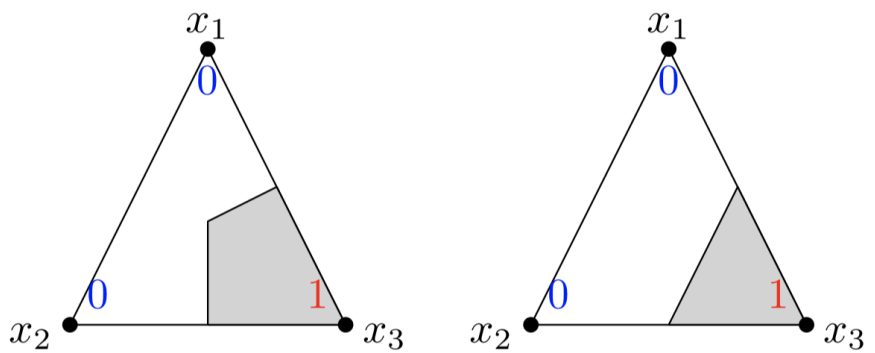
Cover-Hart '67:

$$\lim_{n\to\infty} \mathbb{E}\mathbf{L}(\widehat{f}_n) \leq 2\mathbf{L}(f^*)$$

# Simplicial interpolation

(Belkin, Hsu & Mitra 2018) showed under regularity conditions, simplicial interpolation $\widehat{f}_n$

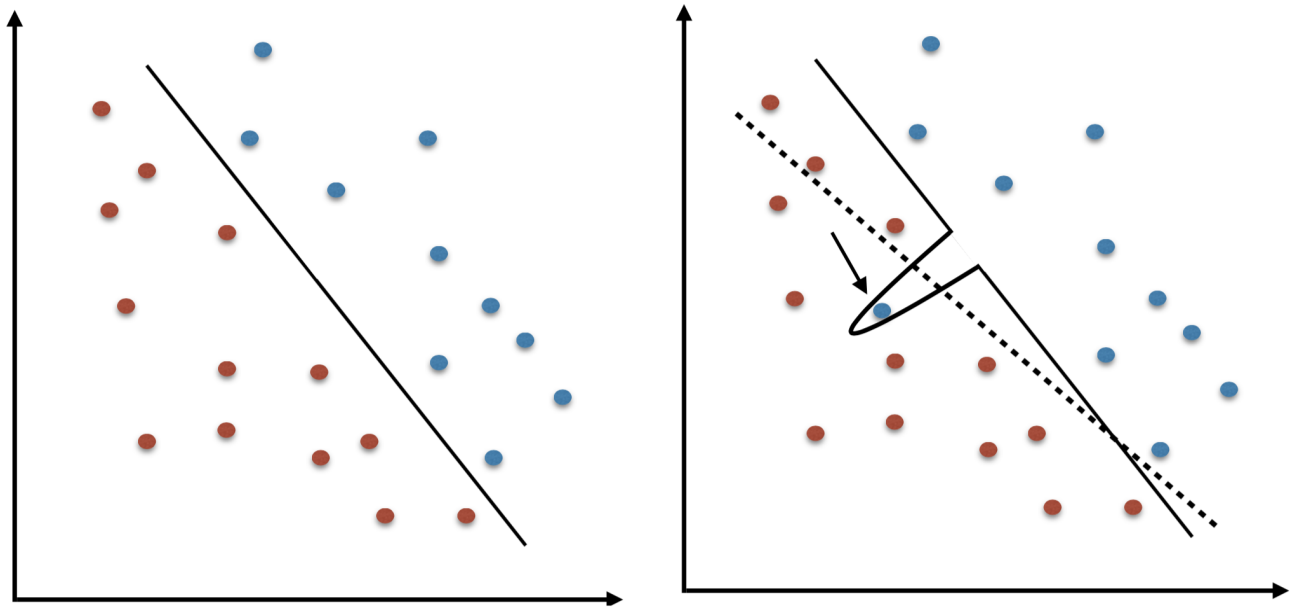
$$\limsup_{n\to\infty} \mathbb{E}\left\|\widehat{f}_n - f_*\right\|^2 \leq \frac{2}{d+2}\mathbb{E}(f^*(X) - Y)^2$$




Nearest neighbor Simplicial interpolation

**Blessing of high dimensionality!**

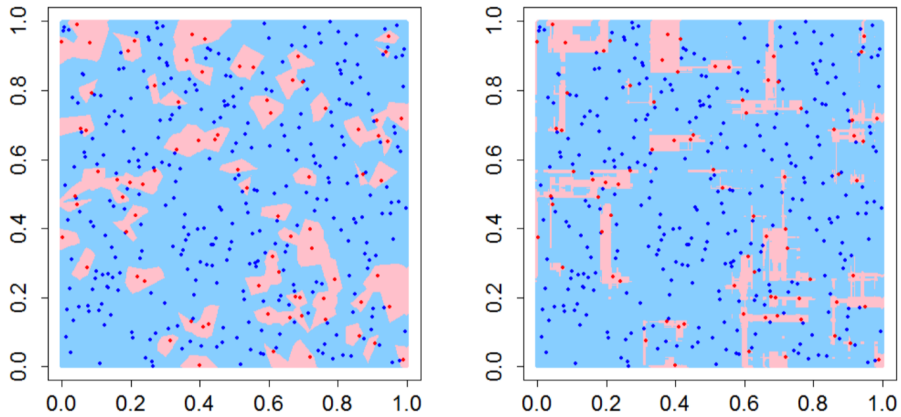
# Interpolation for Classification – Wyner et al




- Spiked-smooth functions

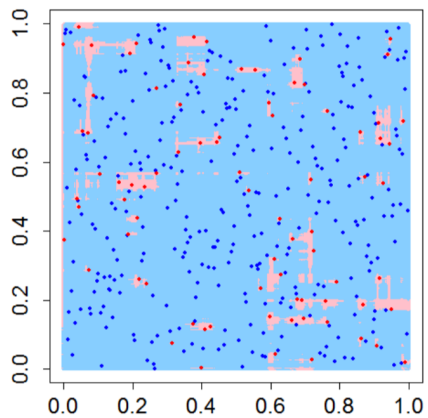
Wyner et al, "Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers," 2017

# Interpolation for Classification – Wyner et al




(a) one-NN

(b) AdaBoost




(c) Random Forests

Figure 5: Performance of one-NN, AdaBoost, and random forests on a pure noise response surface with $\mathbb{P}(y = 1|\mathbf{x}) = .8$ and $n = 400$ training points.

Wyner et al, "Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers," 2017

Belkin, M., Hsu, D. & Mitra, P. (2018), 'Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate', *arXiv preprint arXiv:1806.05161* .

Belkin, M., Ma, S. & Mandal, S. (2018), 'To understand deep learning we need to understand kernel learning', *arXiv preprint arXiv:1802.01396* .

Belkin, M., Rakhlin, A. & Tsybakov, A. B. (2018), 'Does data interpolation contradict statistical optimality?', *arXiv preprint arXiv:1806.09471* .

Caponnetto, A. & De Vito, E. (2007), 'Optimal rates for the regularized least-squares algorithm', *Foundations of Computational Mathematics* **7**(3), 331–368.

Devroye, L., Györfi, L. & Krzyżak, A. (1998), 'The hilbert kernel regression estimate', *Journal of Multivariate Analysis* **65**(2), 209–227.

El Karoui, N. (2010), 'The spectrum of kernel random matrices', *The Annals of Statistics* **38**(1), 1–50.

Johnstone, I. M. (2001), 'On the distribution of the largest eigenvalue in principal components analysis', *Annals of statistics* pp. 295–327.

Liang, T. & Rakhlin, A. (2018), 'Just interpolate: Kernel" ridgeless" regression can generalize', *arXiv preprint arXiv:1808.00387* .

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1998), 'Boosting the margin: A new explanation for the effectiveness of voting methods', *The annals of statistics* **26**(5), 1651–1686.

Shepard, D. (1968), A two-dimensional interpolation function for irregularly-spaced data, *in* 'Proceedings of the 1968 23rd ACM national conference', ACM, pp. 517–524.

Wyner, A. J., Olson, M., Bleich, J. & Mease, D. (2017), 'Explaining the success of adaboost and random forests as interpolating classifiers', *The Journal of Machine Learning Research* **18**(1), 1558–1590.

Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. (2016), 'Understanding deep learning requires rethinking generalization', *arXiv preprint arXiv:1611.03530* .