Machine Learning Theory

CS 446 / ECE 449

2022-03-01 18:37:08 -0600 (1566808)

"`pytorch` **meta-algorithm**".

1. Clean/augment data.
2. Pick model/architecture.
3. Pick a loss function measuring model fit to data.
4. Run a gradient descent variant to fit model to data.
5. Tweak 1-4 until training error is small.
6. Tweak 1-5, possibly reducing model complexity, until testing error is small.

What does math/theory give us?

- Establish why certain pieces work.
- Break other pieces.
- Help us build new pieces.

Plan for today

- Approximation error.
- Optimization error.
- Generalization error.

Approximation error

Approximation error

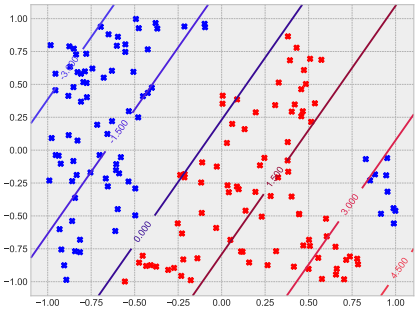
"`pytorch` **meta-algorithm**".

⋮

5. Tweak 1-4 until training error is small.
6. Tweak 1-5, possibly reducing model complexity, until testing error is small.

If our model can not approximate a diverse set of predictors,
we can't hope for much in steps 5 and 6.

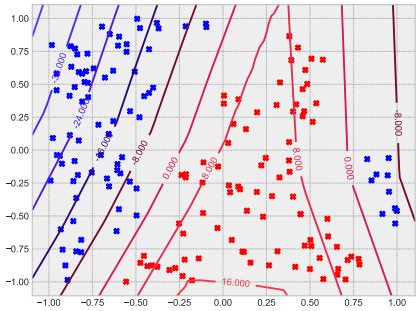
Sometimes, linear just isn't enough



Linear predictor:

$$\boldsymbol{x} \quad \mapsto \quad \boldsymbol{w}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}.$$

Some points misclassified.



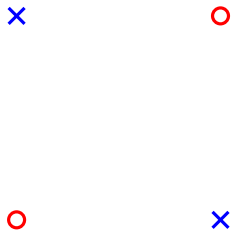
ReLU network:

$$\boldsymbol{x} \mapsto \boldsymbol{W}_2 \sigma_{\mathsf{r}}(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2.$$

0 misclassifications!

Classical example: XOR

Classical "XOR problem" (Minsky-Papert-'69).
(Check wikipedia for "AI Winter".)

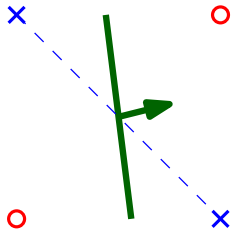


Theorem. On this data, any linear classifier (with affine expansion) makes at least one mistake.

Picture proof. Recall: linear classifiers correspond to separating hyperplanes.

Classical example: XOR

Classical "XOR problem" (Minsky-Papert-'69).
(Check wikipedia for "AI Winter".)



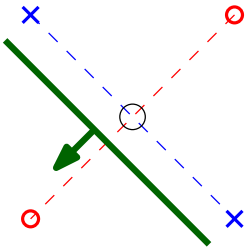
Theorem. On this data, any linear classifier (with affine expansion) makes at least one mistake.

Picture proof. Recall: linear classifiers correspond to separating hyperplanes.

- If it splits the blue points, it's incorrect on one of them.

Classical example: XOR

Classical "XOR problem" (Minsky-Papert-'69).
(Check wikipedia for "AI Winter".)



Theorem. On this data, any linear classifier (with affine expansion) makes at least one mistake.

Picture proof. Recall: linear classifiers correspond to separating hyperplanes.

- If it splits the blue points, it's incorrect on one of them.
- If it doesn't split the blue points,
  then one halfspace contains the common midpoint,
  and therefore wrong on at least one red point.

One layer was not enough. How about two?

Theorem (aka "universal approximation"; Barron '89, Cybenko '89, Hornik-Stinchcombe-White '89, Funahashi '89, Leshno et al '92, ...). Given any continuous function $f : \mathbb{R}^d \to \mathbb{R}$, any accuracy $\epsilon > 0$, and any continuous activation $\sigma : \mathbb{R} \to \mathbb{R}$ which is not a polynomial, there exists a width m and parameters $(\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2) \in \mathbb{R}^{m \times d} \times \mathbb{R}^m \times \mathbb{R}^{1 \times m}$ so that

$$\sup_{\boldsymbol{x} \in [0,1]^d} \left| f(\boldsymbol{x}) - \boldsymbol{W}_2 \sigma \left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1 \right) \right| \leq \epsilon.$$

One layer was not enough. How about two?

Theorem (aka "universal approximation"; Barron '89, Cybenko '89, Hornik-Stinchcombe-White '89, Funahashi '89, Leshno et al '92, ...). Given any continuous function $f : \mathbb{R}^d \to \mathbb{R}$, any accuracy $\epsilon > 0$, and any continuous activation $\sigma : \mathbb{R} \to \mathbb{R}$ which is not a polynomial, there exists a width m and parameters $(\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2) \in \mathbb{R}^{m \times d} \times \mathbb{R}^m \times \mathbb{R}^{1 \times m}$ so that

$$\sup_{\boldsymbol{x} \in [0,1]^d} \left| f(\boldsymbol{x}) - \boldsymbol{W}_2 \sigma \left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1 \right) \right| \leq \epsilon.$$

Remarks.

- Together with XOR example, justifies using nonlinearities.
- Does not justify (very) deep networks, or other aspects of modern architectures.
- Similar results exist for other powerful models we've discussed, e.g., RBF SVM.
- Only says these models exist, not that we can find them algorithmically.
- Required width may be prohibitively large.

Optimization error

Optimization error

"`pytorch` **meta-algorithm**".

⋮

4. Run a gradient descent variant to fit model to data.
5. Tweak 1-4 until training error is small.

⋮

- Our convex optimization lectures analyzed step 4;
  that analysis **does not go through for deep networks**.
- There is a tremendous amount of research here, but still it is not satisfactory.
  - Most of it is on the "Neural Tangent Kernel", which only holds near initialization and reduces to convex optimization.
- Refined questions like ADAM vs SGD are currently out of reach.
- Won't say more in these lectures ☹.

Generalization error

Generalization error

"`pytorch` **meta-algorithm**".

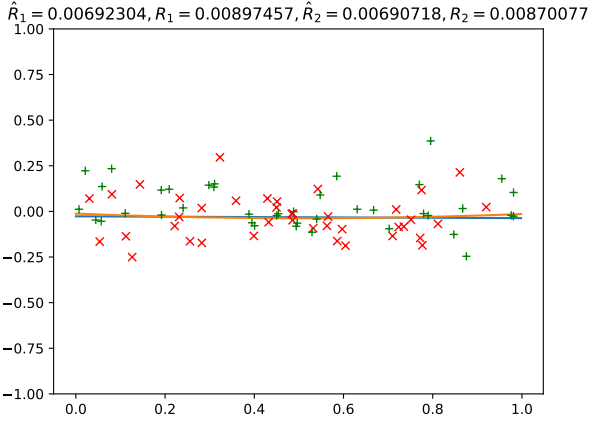
1. Clean/augment data.
2. Pick model/architecture.

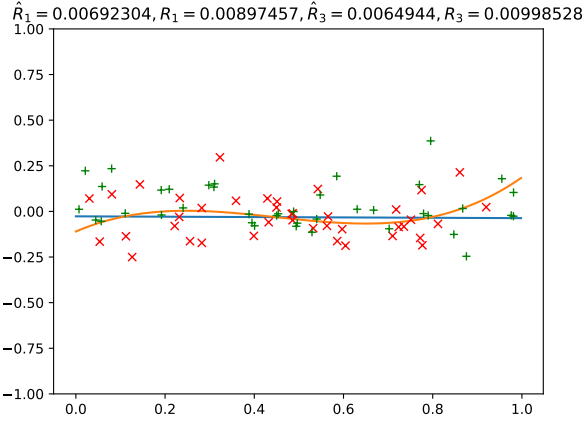
⋮

6. Tweak 1-5, possibly reducing model complexity, until testing error is small.

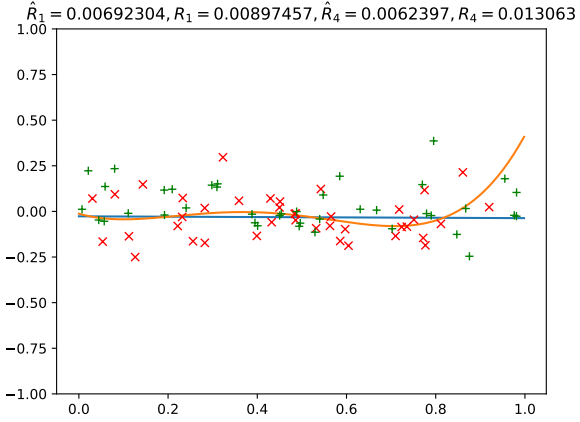
Questions:

- What is "model complexity"?
- Why should reducing it decrease test error?

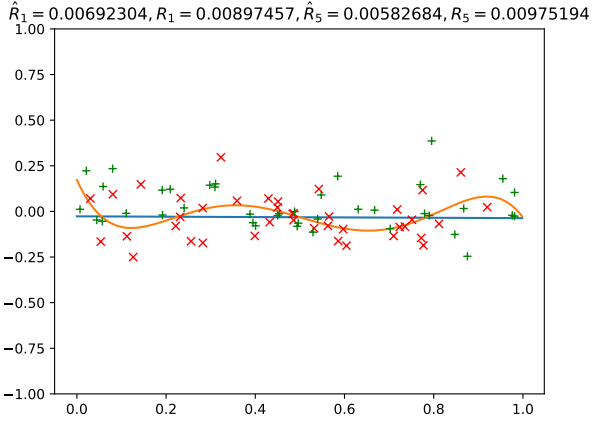

$\hat{R}_1 = 0.00692304, R_1 = 0.00897457, \hat{R}_2 = 0.00690718, R_2 = 0.00870077$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0



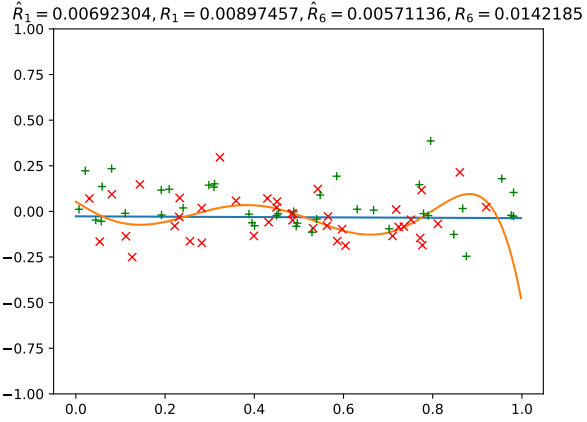
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



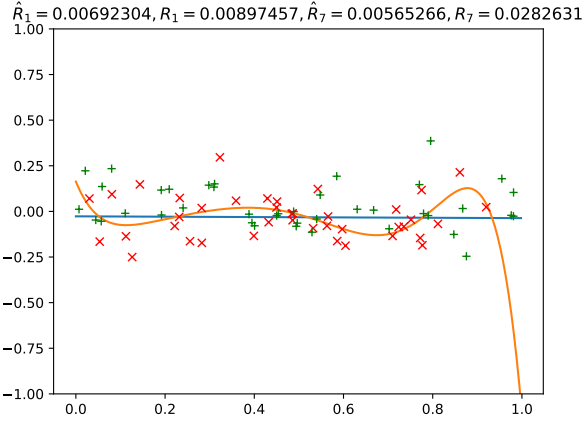
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



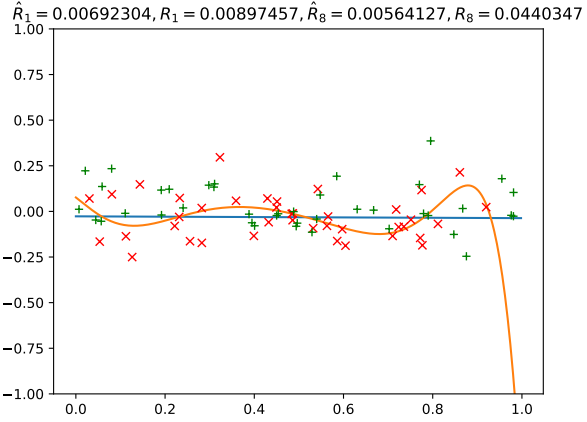
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



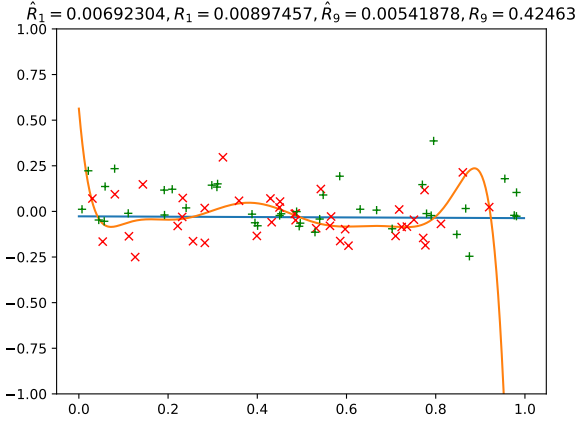
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



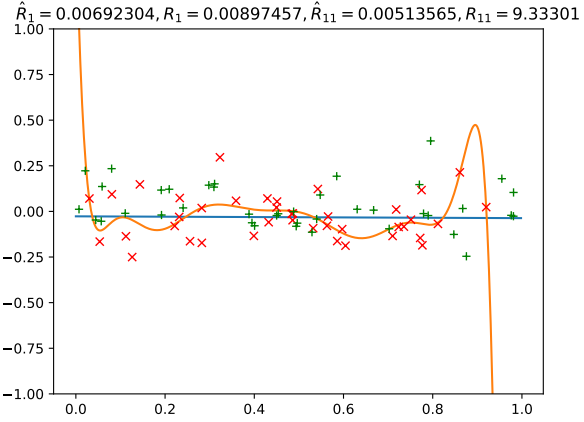
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



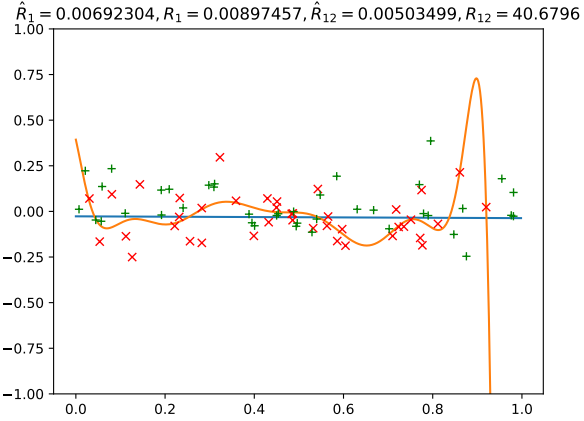
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



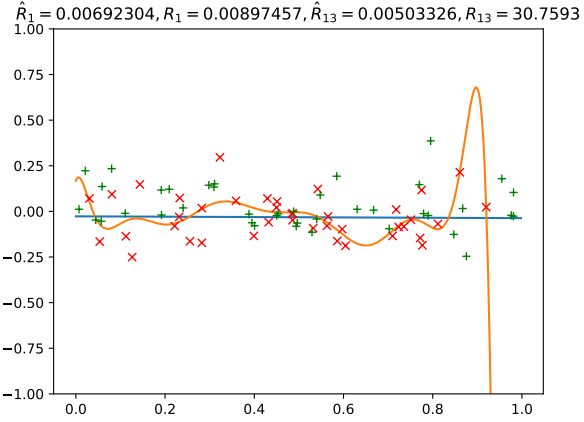
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



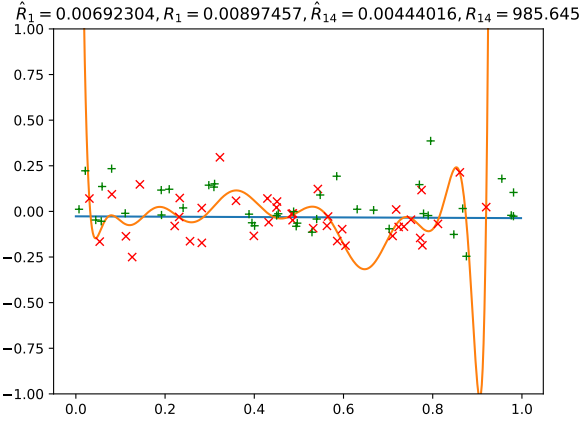
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



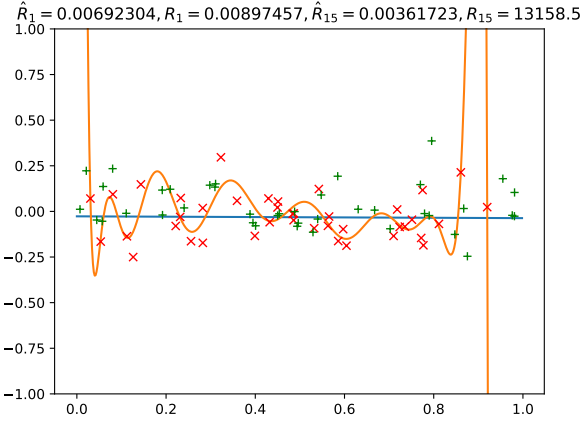
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



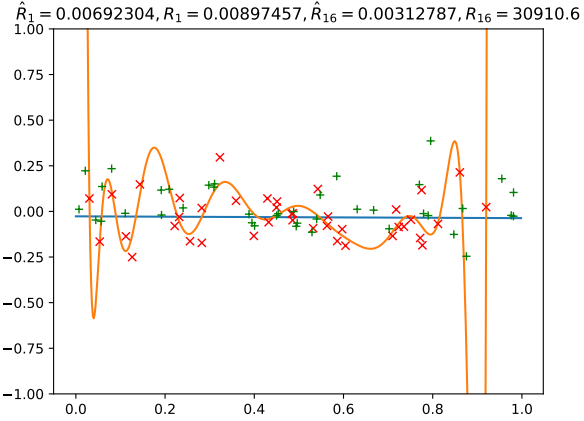
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



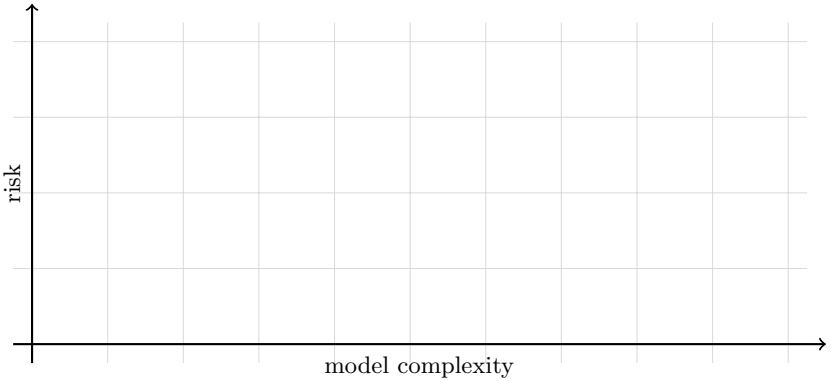
Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



Recall this polynomial-fitting example (ideal function: $\boldsymbol{x} \mapsto 0$);
training data is red, testing data is green.



- Polynomials: degree (number parameters!) increases to right.
- **Classical view:** training error $\to 0$, test error parabolic.
- Regularization: $\lambda \to 0$ when moving to right.
- deep networks: more parameters increases or decreases complexity?
- k-nn: k decreases when moving to the right.

Most complexity notions make the model "less smooth" as we go to the right.

Regularization?

Consider SVM with no kernel. What can we say about the set of predictors

$$\mathcal{F} := \left\{ \boldsymbol{x} \mapsto \boldsymbol{w}^\mathsf{T} \boldsymbol{x} : \boldsymbol{w} \in \mathbb{R}^d \right\}?$$

Regularization?

Consider SVM with no kernel. What can we say about the set of predictors

$$\mathcal{F} := \left\{ \boldsymbol{x} \mapsto \boldsymbol{w}^\mathsf{T} \boldsymbol{x} : \boldsymbol{w} \in \mathbb{R}^d \right\}?$$

For any regularized ERM solution $\hat{\boldsymbol{w}} := \arg\min_{\boldsymbol{w}} \widehat{\mathcal{R}}(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$
and since the hinge loss is nonnegative (whereby $\widehat{\mathcal{R}}(\boldsymbol{w}) \geq 0$ for all $\boldsymbol{w} \in \mathbb{R}^d$),

$$\frac{\lambda}{2}\|\hat{\boldsymbol{w}}\|^2 \leq \widehat{\mathcal{R}}(\hat{\boldsymbol{w}}) + \frac{\lambda}{2}\|\hat{\boldsymbol{w}}\|^2 \leq \widehat{\mathcal{R}}(\boldsymbol{0}) + \frac{\lambda}{2}\|\boldsymbol{0}\|^2 = \frac{1}{n}\sum_{i=1}^n \max\left\{0,\ 1 - \boldsymbol{0}^\mathsf{T}\boldsymbol{x}_i y_i\right\} = 1,$$

and so SVM is working with the finer set

$$\mathcal{F}_\lambda := \left\{ \boldsymbol{w} \mapsto \boldsymbol{w}^\mathsf{T}\boldsymbol{x} : \|\boldsymbol{w}\|^2 \leq \frac{2}{\lambda} \right\}.$$

Regularization?

Consider SVM with no kernel. What can we say about the set of predictors

$$\mathcal{F} := \left\{ \boldsymbol{x} \mapsto \boldsymbol{w}^\mathsf{T} \boldsymbol{x} : \boldsymbol{w} \in \mathbb{R}^d \right\}?$$

For any regularized ERM solution $\hat{\boldsymbol{w}} := \arg\min_{\boldsymbol{w}} \widehat{\mathcal{R}}(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$
and since the hinge loss is nonnegative (whereby $\widehat{\mathcal{R}}(\boldsymbol{w}) \geq 0$ for all $\boldsymbol{w} \in \mathbb{R}^d$),

$$\frac{\lambda}{2}\|\hat{\boldsymbol{w}}\|^2 \leq \widehat{\mathcal{R}}(\hat{\boldsymbol{w}}) + \frac{\lambda}{2}\|\hat{\boldsymbol{w}}\|^2 \leq \widehat{\mathcal{R}}(\boldsymbol{0}) + \frac{\lambda}{2}\|\boldsymbol{0}\|^2 = \frac{1}{n}\sum_{i=1}^n \max\left\{0,\ 1 - \boldsymbol{0}^\mathsf{T}\boldsymbol{x}_i y_i\right\} = 1,$$

and so SVM is working with the finer set

$$\mathcal{F}_\lambda := \left\{ \boldsymbol{w} \mapsto \boldsymbol{w}^\mathsf{T}\boldsymbol{x} : \|\boldsymbol{w}\|^2 \leq \frac{2}{\lambda} \right\}.$$

(A similar derivation holds more generally, e.g., with other nonnegative losses.)

Computer Science > Learning

Understanding deep learning requires rethinking generalization

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

(Submitted on 10 Nov 2016 (v1), last revised 26 Feb 2017 (this version, v2))

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference betv
wisdom attributes small generalization error either to properties of the model family, or to the regularization techniqu
Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neu
Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained w
labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization, and occurs even
unstructured random noise. We corroborate these experimental findings with a theoretical construction showing tha
perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usu
We interpret our experimental findings by comparison with traditional models.

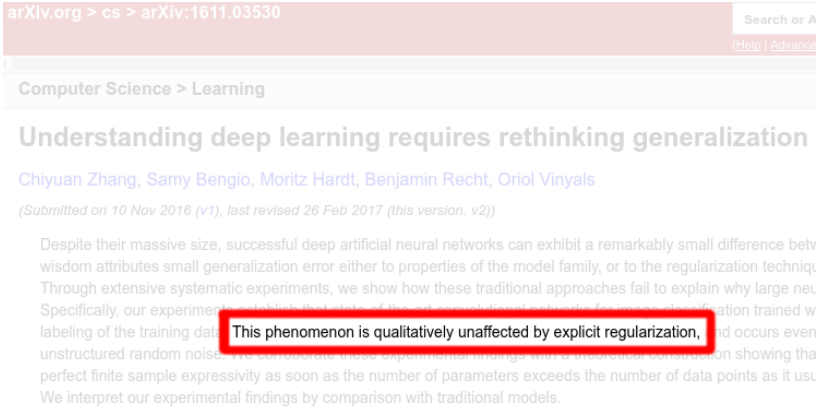
Understanding deep learning requires rethinking generalization

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

(Submitted on 10 Nov 2016 (v1), last revised 26 Feb 2017 (this version, v2))

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference betv
wisdom attributes small generalization error either to properties of the model family, or to the regularization techniqu
Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neu
Specifically, our experimen[illegible]ation trained w
labeling of the training dat[illegible]nd occurs even
unstructured random noise[illegible]ation showing tha
perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usu
We interpret our experimental findings by comparison with traditional models.

This phenomenon is qualitatively unaffected by explicit regularization,



For deep networks, the role of explicit ℓ_2 regularization is unclear.

Many other factors contribute to generalization, e.g., data augmentation, architecture choices (width, normalization layers, dropout, and even overall structure), optimizer choice. . . ; overall the exact generalization consequences are not understood.

Why should training error and test error be close?

Consider a fixed predictor $f : \mathcal{X} \to \mathbb{R}$.

- Suppose $((\boldsymbol{x}_i, y_i))_{i=1}^n$ are drawn IID from some distribution.
- Define $Z_i := \mathbb{1}[\operatorname{sgn}(f(\boldsymbol{x}_i)) \neq y_i]$;
  thus $(Z_i)_{i=1}^n$ **are also IID random variables**.

Why should training error and test error be close?

Consider a fixed predictor $f : \mathcal{X} \to \mathbb{R}$.

- Suppose $((\boldsymbol{x}_i, y_i))_{i=1}^n$ are drawn IID from some distribution.
- Define $Z_i := \mathbb{1}[\operatorname{sgn}(f(\boldsymbol{x}_i)) \neq y_i)$;
  thus $(Z_i)_{i=1}^n$ **are also IID random variables**.
- Their common mean $\mathbb{E}Z_1$ satisfies

$$\mathbb{E}Z_1 = \mathbb{E}\frac{1}{n}\sum_{i=1}^n \mathbb{1}[\operatorname{sgn}(f(\boldsymbol{x}_i) \neq y_i] = \mathbb{E}\frac{1}{n}\sum_{i=1}^n \ell_{\text{zo}}(y_i, f(\boldsymbol{x}_i)) = \mathcal{R}_{\text{zo}}(f),$$

  where ℓ_{zo} and $\mathcal{R}_{\text{zo}}$ are the zero-one loss and risk **over the distribution/population!**

Why should training error and test error be close?

Consider a fixed predictor $f : \mathcal{X} \to \mathbb{R}$.

- Suppose $((\boldsymbol{x}_i, y_i))_{i=1}^n$ are drawn IID from some distribution.
- Define $Z_i := \mathbb{1}[\operatorname{sgn}(f(\boldsymbol{x}_i)) \neq y_i]$;
  thus $(Z_i)_{i=1}^n$ **are also IID random variables**.
- Their common mean $\mathbb{E}Z_1$ satisfies

$$\mathbb{E}Z_1 = \mathbb{E}\frac{1}{n}\sum_{i=1}^n \mathbb{1}[\operatorname{sgn}(f(\boldsymbol{x}_i) \neq y_i] = \mathbb{E}\frac{1}{n}\sum_{i=1}^n \ell_{\text{zo}}(y_i, f(\boldsymbol{x}_i)) = \mathcal{R}_{\text{zo}}(f),$$

  where ℓ_{zo} and $\mathcal{R}_{\text{zo}}$ are the zero-one loss and risk **over the distribution/population!**
- We want to say $\mathcal{R}_{\text{zo}}(f)$ is small, but only observe

$$\widehat{\mathcal{R}}_{\text{zo}}(f) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}[\operatorname{sgn}(f(\boldsymbol{x}_i) \neq y_i)];$$

what can we say about $\mathcal{R}_{\text{zo}}(f)$, given $\widehat{\mathcal{R}}_{\text{zo}}(f)$?

Hoeffding's inequality revisited

Theorem (Hoeffding's inequality). Given IID $Z_i \in [a, b]$ and any $\epsilon > 0$,

$$\Pr\left[\frac{1}{n}\sum_i Z_i - \mathbb{E}Z_1 \geq \epsilon\right] \leq \exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right).$$

[New rephrasing (cf. lecture 12)**.]** Alternatively, with probability at least $1 - \delta$,

$$\frac{1}{n}\sum_{i=1}^{n} Z_i \leq \mathbb{E}Z_1 + (b-a)\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

Hoeffding's inequality revisited

Theorem (Hoeffding's inequality). Given IID $Z_i \in [a, b]$ and any $\epsilon > 0$,

$$\Pr\left[\frac{1}{n}\sum_i Z_i - \mathbb{E}Z_1 \geq \epsilon\right] \leq \exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right).$$

[New rephrasing (cf. lecture 12)**.]** Alternatively, with probability at least $1 - \delta$,

$$\frac{1}{n}\sum_{i=1}^{n} Z_i \leq \mathbb{E}Z_1 + (b-a)\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

Remarks.

- Using $Z_i := \mathbb{1}[\mathrm{sgn}(f(\boldsymbol{x}_i)) \neq y_i]$, with probability at least $1 - \delta$,

$$\mathcal{R}_{\mathrm{zo}}(f) = \mathbb{E}\frac{1}{n}\sum_{i=1}^{n} Z_i \leq \frac{1}{n}\sum_{i=1}^{n} Z_i + \sqrt{\frac{\ln(1/\delta)}{2n}} = \widehat{\mathcal{R}}_{\mathrm{zo}}(f) + \sqrt{\frac{\ln(1/\delta)}{2n}}.$$

  E.g., if replace $\ln(1/\delta)$ with 10, holds with probability $\geq 1 - e^{-10} \approx 0.99994$.
- Can flip inequality by replacing Z_i with $-Z_i$;
  can control absolute value $|\sum_i Z_i/n - \mathbb{E}Z_1|$ via union bound.
- General version: see **concentration of measure**. (Beyond scope of course.)

Hoeffding and algorithm output

For **fixed** f, with probability $\geq 1 - \delta$, $\mathcal{R}_{\mathsf{zo}}(f) \leq \widehat{\mathcal{R}}_{\mathsf{zo}}(f) + \sqrt{\frac{\ln(1/\delta)}{2n}}$.

Has no "model complexity" term;
must break if applied to output of a learning algorithm.
(E.g., expect different bound for linear predictor than for ResNet.)

"Fixed f"?

IID $((\boldsymbol{x}_i, y_i))_{i=1}^n$ given, and define

$$\hat{f}(\boldsymbol{x}) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{"bear"} & \text{otherwise.} \end{cases}$$

Can easily have $\widehat{\mathcal{R}}_{0/1}(\hat{f}) = 0$ but $\mathcal{R}_{0/1}(\hat{f}) = 1$!

"Fixed f"?

IID $((\boldsymbol{x}_i, y_i))_{i=1}^n$ given, and define

$$\hat{f}(\boldsymbol{x}) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{"bear"} & \text{otherwise.} \end{cases}$$

Can easily have $\widehat{\mathcal{R}}_{0/1}(\hat{f}) = 0$ but $\mathcal{R}_{0/1}(\hat{f}) = 1$!

Remarks.

- **Hoeffding can not be applied:**

"Fixed f"?

IID $((\boldsymbol{x}_i, y_i))_{i=1}^n$ given, and define

$$\hat{f}(\boldsymbol{x}) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{"bear"} & \text{otherwise.} \end{cases}$$

Can easily have $\widehat{\mathcal{R}}_{0/1}(\hat{f}) = 0$ but $\mathcal{R}_{0/1}(\hat{f}) = 1$!

Remarks.

- **Hoeffding can not be applied: $\hat{f}$ depends on $((\boldsymbol{x}_i, y_i))_{i=1}^n$**, therefore $(Z_i)_{i=1}^n$ with $Z_i = \mathbb{1}[\operatorname{sgn}(\hat{f}(\boldsymbol{x}_i)) \neq y_i]$ are not all independent!
- There exist various fixes, here are some:

"Fixed f"?

IID $((\boldsymbol{x}_i, y_i))_{i=1}^n$ given, and define

$$\hat{f}(\boldsymbol{x}) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{"bear"} & \text{otherwise.} \end{cases}$$

Can easily have $\widehat{\mathcal{R}}_{0/1}(\hat{f}) = 0$ but $\mathcal{R}_{0/1}(\hat{f}) = 1$!

Remarks.

- **Hoeffding can not be applied:** **$\hat{f}$ depends on $((\boldsymbol{x}_i, y_i))_{i=1}^n$**, therefore $(Z_i)_{i=1}^n$ with $Z_i = \mathbb{1}[\operatorname{sgn}(\hat{f}(\boldsymbol{x}_i)) \neq y_i]$ are not all independent!
- There exist various fixes, here are some:
  1. Compute **validation error** $\widehat{\mathcal{R}}_{\text{val}}$ using **validation set** disjoint from training set.

"Fixed f"?

IID $((\boldsymbol{x}_i, y_i))_{i=1}^n$ given, and define

$$\hat{f}(\boldsymbol{x}) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{"bear"} & \text{otherwise.} \end{cases}$$

Can easily have $\widehat{\mathcal{R}}_{0/1}(\hat{f}) = 0$ but $\mathcal{R}_{0/1}(\hat{f}) = 1$!

Remarks.

- **Hoeffding can not be applied:** **$\hat{f}$ depends on $((\boldsymbol{x}_i, y_i))_{i=1}^n$**, therefore $(Z_i)_{i=1}^n$ with $Z_i = \mathbb{1}[\operatorname{sgn}(\hat{f}(\boldsymbol{x}_i)) \neq y_i]$ are not all independent!
- There exist various fixes, here are some:
  1. Compute **validation error** $\widehat{\mathcal{R}}_{\text{val}}$ using **validation set** disjoint from training set.
  2. Pay a **model complexity penalty term**: with probability $\geq 1 - \delta$, **for all** $f \in \mathcal{F}$ **simultaneously**,

$$\mathcal{R}(f) \leq \widehat{\mathcal{R}}(f) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{\text{complexity}(\mathcal{F}) + \ln(1/\delta)}{n}}\right),$$

where valid definitions of "complexity($\mathcal{F}$)" are beyond the scope of this course. This analysis, however, is extremely loose ☹.

"Fixed f"?

IID $((\boldsymbol{x}_i, y_i))_{i=1}^n$ given, and define

$$\hat{f}(\boldsymbol{x}) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{"bear"} & \text{otherwise.} \end{cases}$$

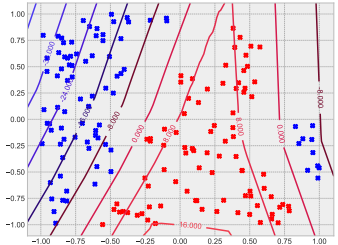
Can easily have $\widehat{\mathcal{R}}_{0/1}(\hat{f}) = 0$ but $\mathcal{R}_{0/1}(\hat{f}) = 1$!

Remarks.

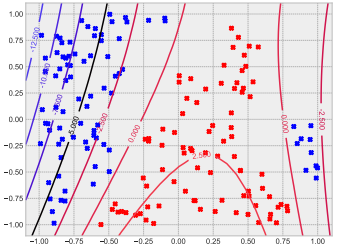
- **Hoeffding can not be applied:** **$\hat{f}$ depends on $((\boldsymbol{x}_i, y_i))_{i=1}^n$**, therefore $(Z_i)_{i=1}^n$ with $Z_i = \mathbb{1}[\operatorname{sgn}(\hat{f}(\boldsymbol{x}_i)) \neq y_i]$ are not all independent!
- There exist various fixes, here are some:
  1. Compute **validation error** $\widehat{\mathcal{R}}_{\text{val}}$ using **validation set** disjoint from training set.
  2. Pay a **model complexity penalty term**: with probability $\geq 1 - \delta$, **for all** $f \in \mathcal{F}$ **simultaneously**,

     $$\mathcal{R}(f) \leq \widehat{\mathcal{R}}(f) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{\text{complexity}(\mathcal{F}) + \ln(1/\delta)}{n}}\right),$$

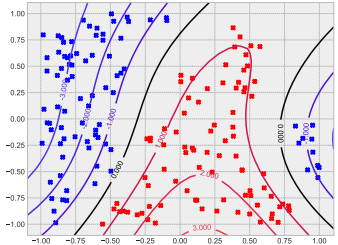
     where valid definitions of "complexity($\mathcal{F}$)" are beyond the scope of this course. This analysis, however, is extremely loose ☹.
  3. Use refined reasoning about how algorithm interacts with data.



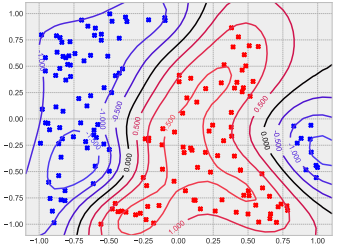
ReLU network.



Quadratic SVM.



RBF SVM ($\sigma = 1$).



RBF SVM ($\sigma = 0.1$).

Different models exhibit different complexity on the same data.

Summary for today

- Approximation error.
- Optimization error.
- Generalization error.

(Appendix.)

If we obtain $f \in \mathcal{F}$ via gradient descent (or other learning algorithm), then f depends on the training set, and it seems that the random variables $Z_i := \mathbb{1}[f(\boldsymbol{x}_i) \neq y_i]$ are not independent. The $\hat{f}$ given in lecture has very particular structure which made the situation more delicate, so here's a more detailed explanation.

To make the randomness explicit, let's define

$$\hat{f}(\boldsymbol{x}; ((\boldsymbol{x}_i, y_i))_{i=1}^n) := \begin{cases} y_i & \text{when } \boldsymbol{x} = \boldsymbol{x}_i \\ \text{“bear”} & \text{otherwise;} \end{cases}$$

we've now made it clear that $\hat{f}$ itself is a random variable that depends on $((\boldsymbol{x}_i, y_i))_{i=1}^n$. (Also, let's technically assume that $\boldsymbol{x}_i = \boldsymbol{x}_j$ only when $i = j$ with probability 1, as in the uniform example.) If we define

$$Z_i := \mathbb{1}[\hat{f}(\boldsymbol{x}_i; ((\boldsymbol{x}_j, y_j))_{j=1}^n) \neq y_i] = \mathbb{1}[\hat{f}(\boldsymbol{x}_i; (\boldsymbol{x}_i, y_i)) \neq y_i] = \mathbb{1}[y_i \neq y_i] = 0,$$

and since Z_i formally depends on the randomness of all n examples,

$$\mathbb{E} Z_i = \mathbb{E}\left[\mathbb{E}[Z_i] \mid Z_i\right] = \mathbb{E}[0] = 0.$$

In this case, though, since the data points never coincide, we could just as well define $Z_i := \mathbb{1}[\hat{f}(x_i; (x_i, y_i)) \neq y_i] = \mathbb{1}[y_i \neq y_i] = 0$, a deterministic quantity, and moreover the $(Z_i)_{i=1}^n$ are now independent. What gives?

(Continues on next slide.)

With everything boiled down like this, the real issue is that $\mathbb{E}Z_i \neq \widehat{\mathcal{R}}(\hat{f})$. Specifically, what we'd like to reason about is

$$\Pr\left[\hat{f}(x; ((x_i, y_i))_{i=1}^n) \neq y \;\middle|\; ((x_i, y_i))_{i=1}^n\right];$$

that is, $\hat{f}$ is a random variable, it depends on a training set, and we'd like to reason about its performance on future data. So for instance, we may want to instead consider n additional examples $((\tilde{x}_i, \tilde{y}_i))_{i=1}^{n+1}$, and define further random variables

$$U_i := \mathbb{1}[\hat{f}(\tilde{x}_i; ((x_i, y_i))_{i=1}^n \neq \tilde{y}_i];$$

moreover, if we take the conditional expectation, we get the earlier desired conditional error:

$$\mathbb{E}_{(\tilde{x}_i, \tilde{y}_i)}\left[U_i | ((x_i, y_i))_{i=1}^n\right].$$

However, these U_i are different from the Z_i we had before.

Supplemental reading: many relevant chapters in "Understanding Machine Learning" by Shai Shalev-Shwartz & Shai Ben-David.