Lecture 15 — Learning Theory (Part 2 of 2)

Alex Schwing and Matus Telgarsky

March 6, 2018

Schedule for today.

- Midterm announcements.
- Overfitting/generalization: reminders.
- Overfitting/generalization: intuition and basics.
- Overfitting/generalization: some asides.

Learning theory reading:
see my learning theory course's resources (click on `http://mjt.cs.illinois.edu/courses/mlt-f17/` or google "matus uiuc mlt-f17").

Midterm announcements.

- **Location (all ECEB):** your netid determines your room:
  - **1002** (this room)**:** aa18 - ryang28.
  - **1013:** sabag2 - xunlin2.
  - **1015:** xyu69 - zzhou51.
- **Time:** start time 6pm, duration 90 minutes.
- **Notes:** can bring one standard size (“US letter”) sheet of notes, front and back, **handwritten**.
- **Review lecture and materials:** wait until Thursday.

Learning Theory (part 2 of 2) – Overfitting/Generalization.

- **Overfitting:** better performance on past data than future data.
- **Generalizing:** similar performance on past and future data.

Learning Theory (part 2 of 2) – Overfitting/Generalization.

- **Overfitting:** better performance on past data than future data.
- **Generalizing:** similar performance on past and future data.

Models

Reasoning about this requires a **model** linking past and future.

- **Statistical learning theory:** past and future examples drawn IID from a common distribution.
- **Online learning:** adversary constructs new examples.

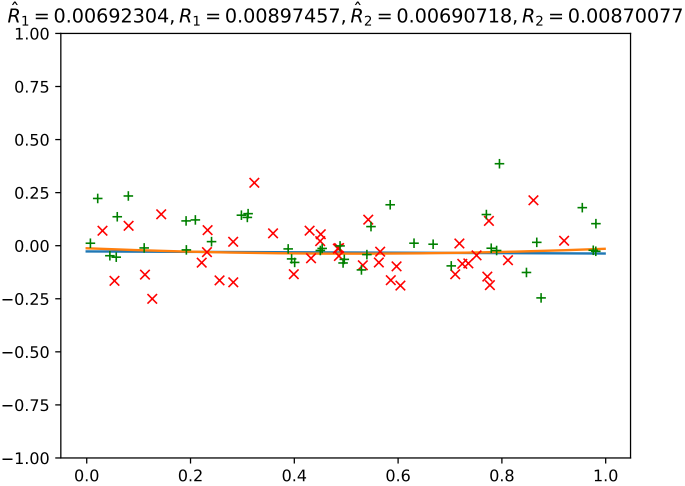
Overfitting by example.

Linear or polynomial least squares?

Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.

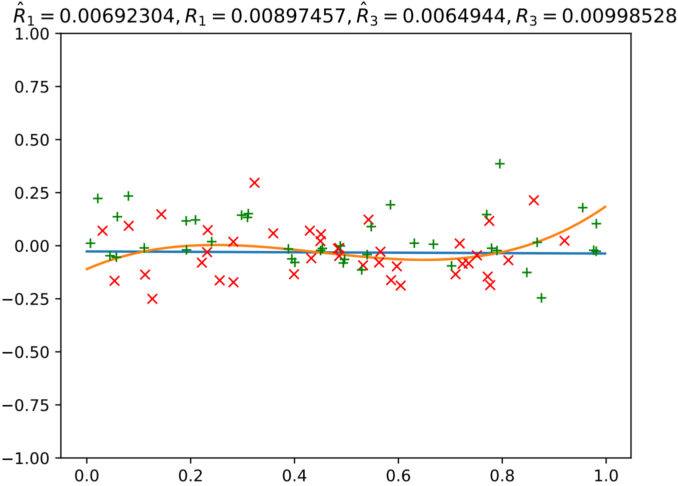
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.

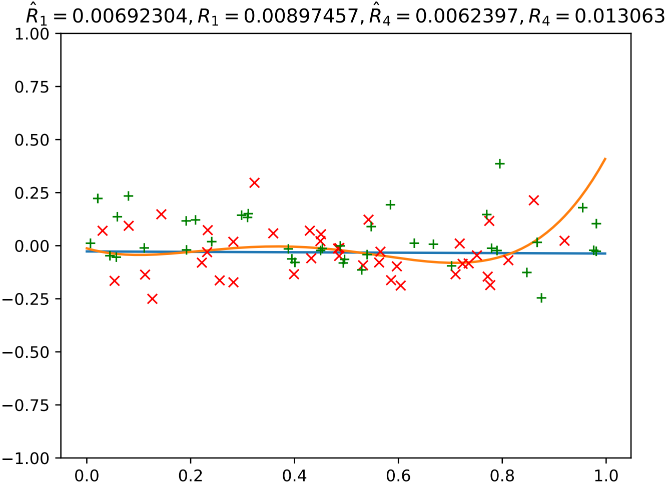


Overfitting by example.

Linear or polynomial least squares?

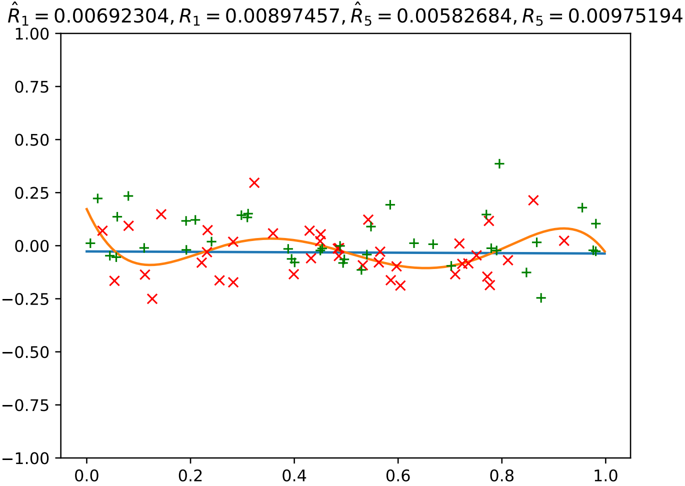
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.

Red: seen data. Green: unseen data.



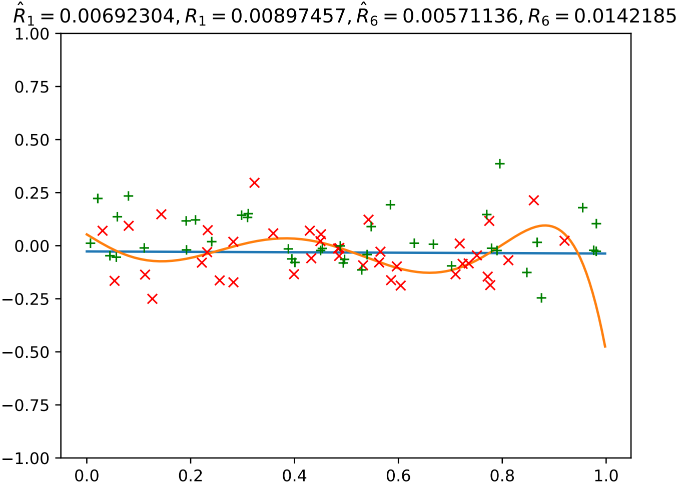
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



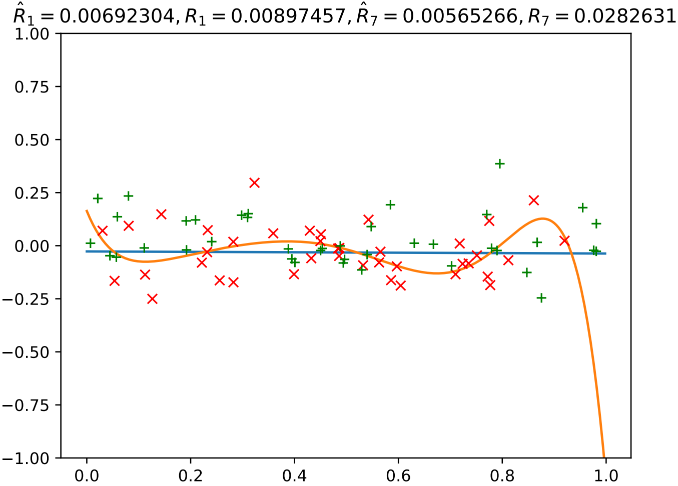
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



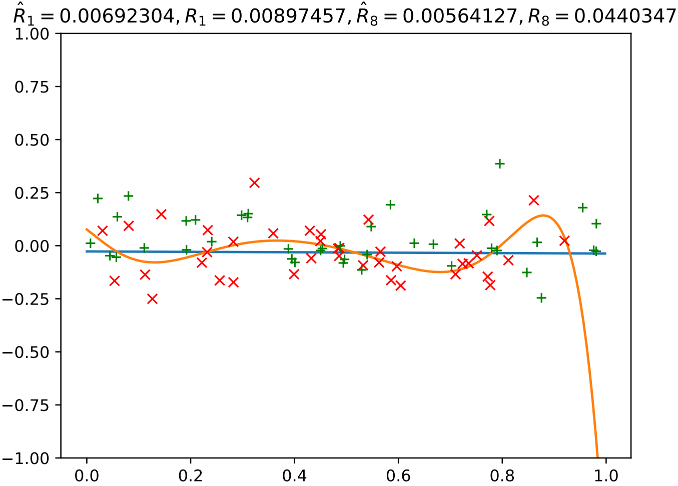
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



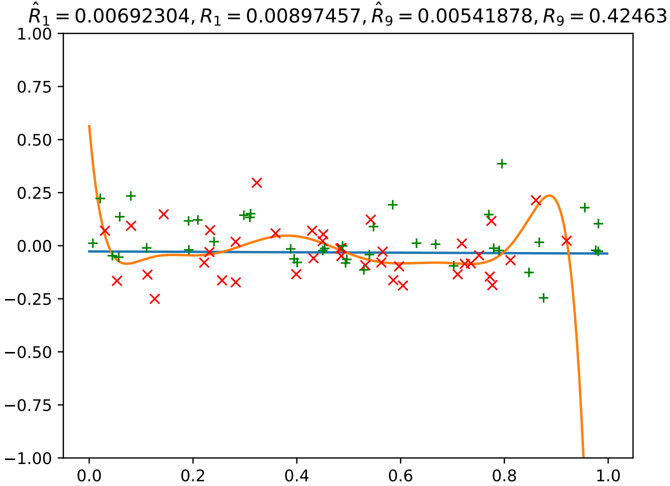
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



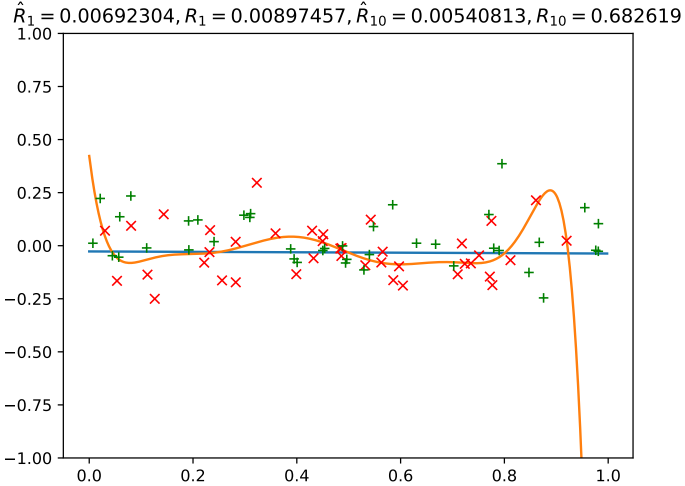
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



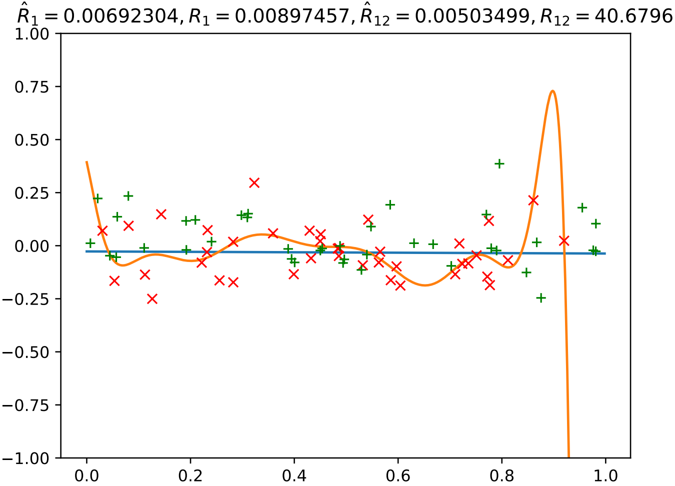
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



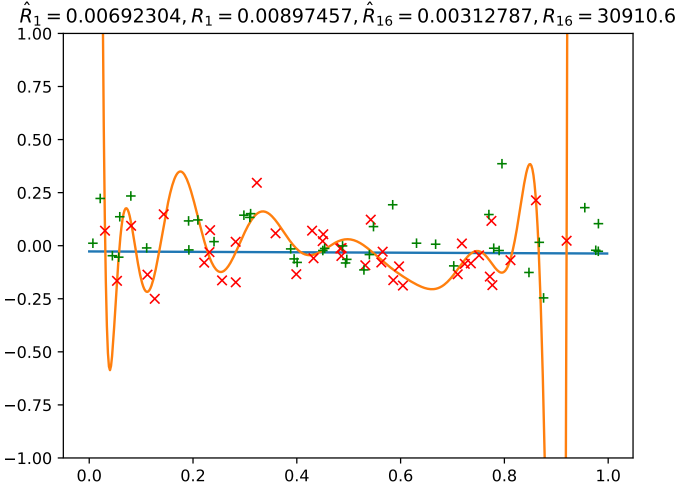
Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.



Overfitting by example.

Linear or polynomial least squares?
Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.
Red: seen data. Green: unseen data.

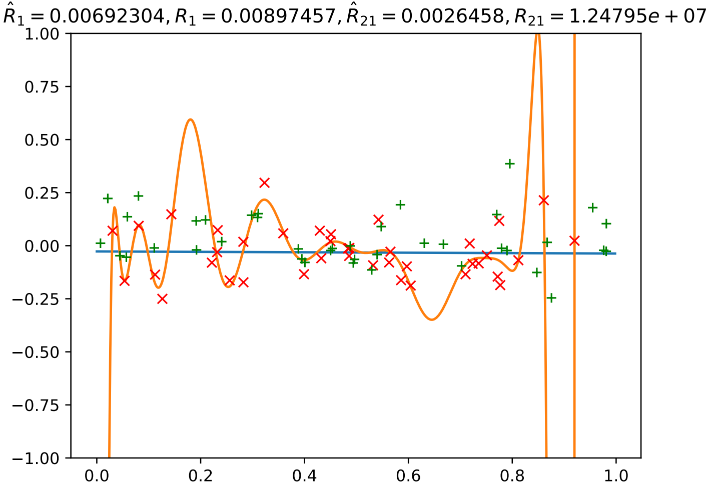


Overfitting by example.

Linear or polynomial least squares?

Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.

Red: seen data. Green: unseen data.

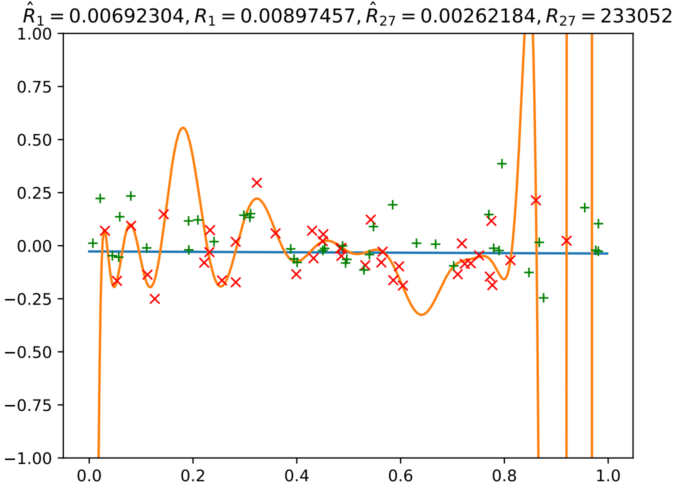


Overfitting by example.

Linear or polynomial least squares?

Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.

Red: seen data. Green: unseen data.

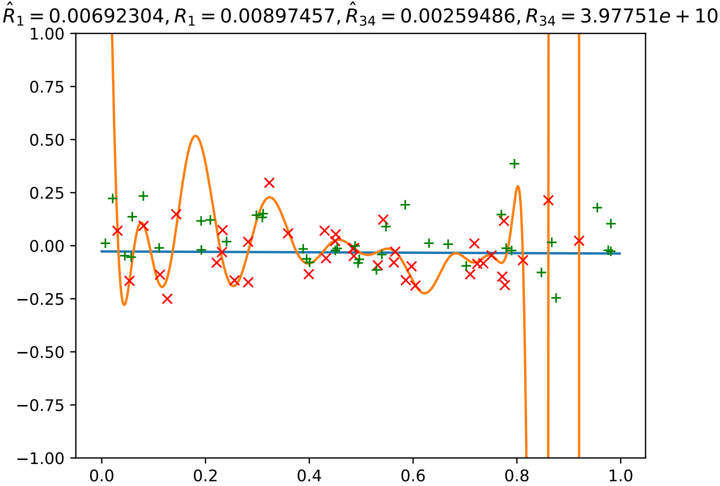


Overfitting by example.

Linear or polynomial least squares?

Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.

Red: seen data. Green: unseen data.

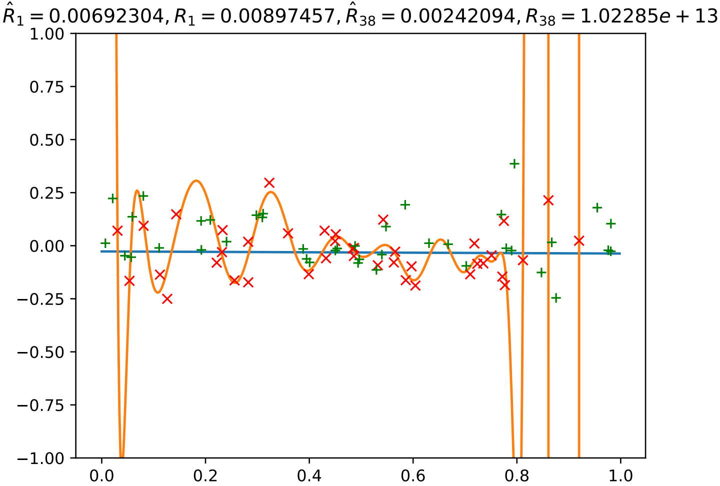


Overfitting by example.

Linear or polynomial least squares?

Truth: $y = 0 \cdot x + \xi, \quad \xi \sim$ Gaussian.

Red: seen data. Green: unseen data.



Overfitting by example.

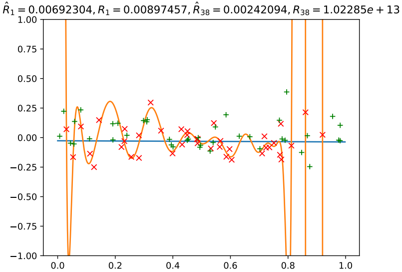


Figure 1: Fitting a degree 38 polynomial.

Intuition: More parameters $\Longrightarrow$ more overfitting.
Can reduce overfitting with model choice and regularization (similar...).
What learning theory gives us: concrete relationships.

Aside/review: least squares code for this plot!

```
# [ . . . ]

for s in [ 'tr', 'te', 'grid' ]:
    X[s][:, 0] = 1.0
    #X[s][:, 1] is random according to some distribution
    for j in range(2, n):
        X[s][:, j] = X[s][:, 1] * X[s][:, j - 1]

# [ . . . ]

for j in range(2, n):
    #better to use the black box N.linalg.lstsq
    w[j] = N.linalg.pinv(X['tr'][:, :j]) @ Y['tr']
    R[j] = dict( (s, N.linalg.norm(X[s][:, :j] @ w[j]
                          - Y[s])**2 / 2 / n)
                 for s in [ 'tr', 'te' ] )
```

Binomials, random walks, coin tosses, classification.

Binomials, random walks, coin tosses, classification.

Last lecture we saw *Hoeffding's Inequality*.

Theorem (Hoeffding's inequality)**.** Suppose each draw from the distribution lies in the interval $[a, b]$. With probability at least $1 - \delta$ over an iid draw of $(z_i)_{i=1}^n$,

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

Binomials, random walks, coin tosses, classification.

Last lecture we saw *Hoeffding's Inequality*.

Theorem (Hoeffding's inequality)**.** Suppose each draw from the distribution lies in the interval $[a, b]$. With probability at least $1 - \delta$ over an iid draw of $(z_i)_{i=1}^n$,

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

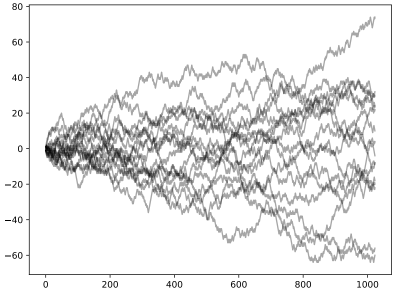
What does this mean?

Interpreting Hoeffding's inequality.

```
for i in range(k):
    path = N.cumsum( N.random.randint(0, 2, n) * 2 - 1 )
    plt.plot(path, color = 'black', alpha = 0.35)
```

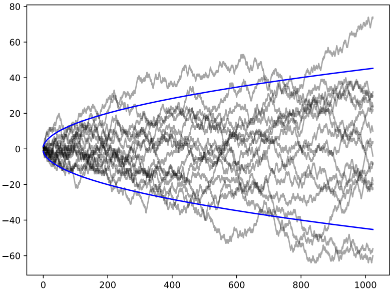
Interpreting Hoeffding's inequality.

```
for i in range(k):
    path = N.cumsum( N.random.randint(0, 2, n) * 2 - 1 )
    plt.plot(path, color = 'black', alpha = 0.35)
```



Interpreting Hoeffding's inequality.

```
for i in range(k):
    path = N.cumsum( N.random.randint(0, 2, n) * 2 - 1 )
    plt.plot(path, color = 'black', alpha = 0.35)
```

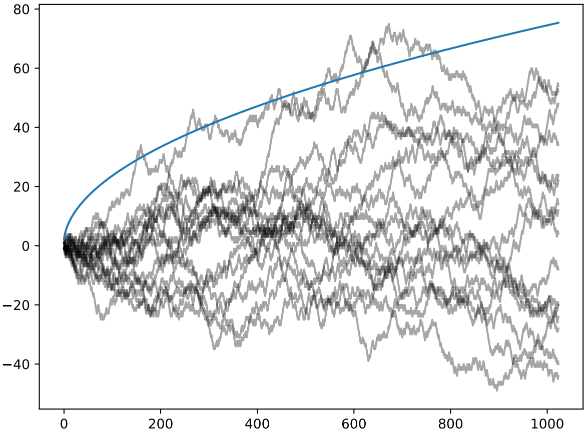


Hoeffding says: for any **fixed** n, with probability $\geq 1 - {}^{1}/_{\sqrt{e}}$,

position $\leq \sqrt{n}$.

Hoeffding's inequality with more walks.

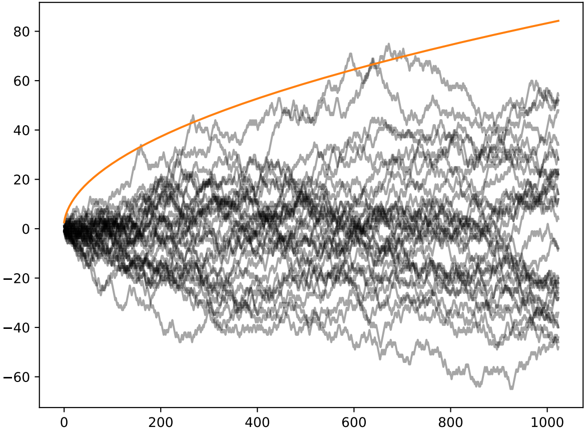
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

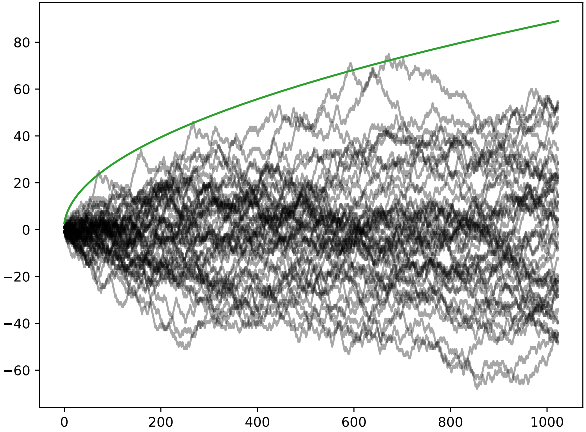
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

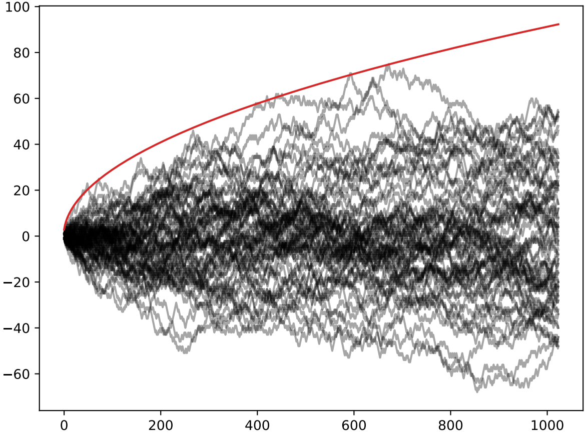
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

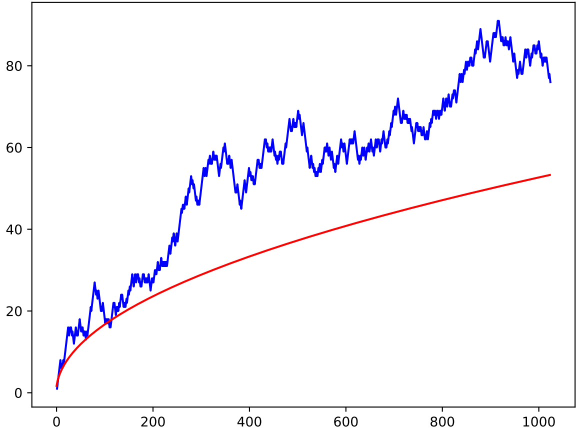
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

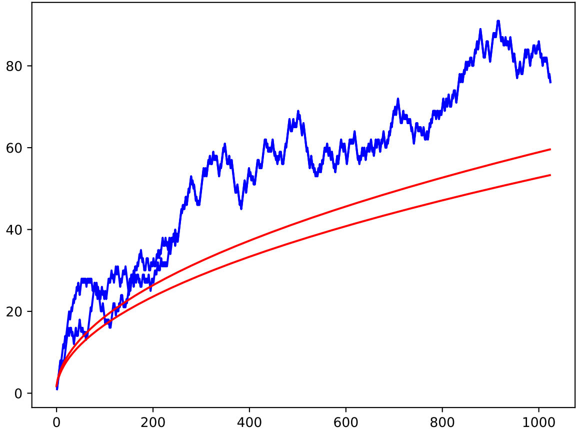
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

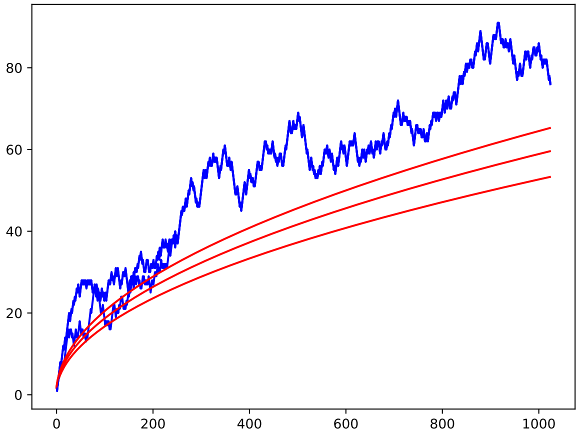
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

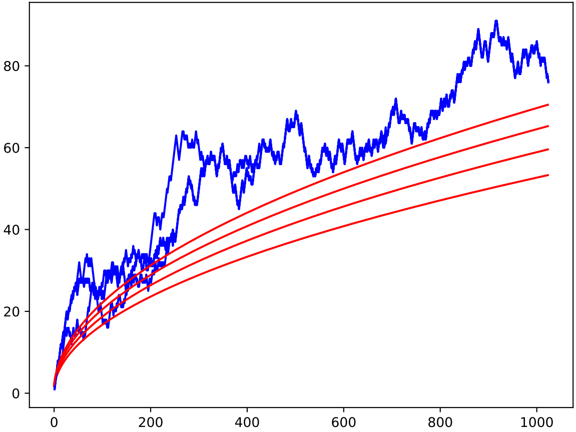
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

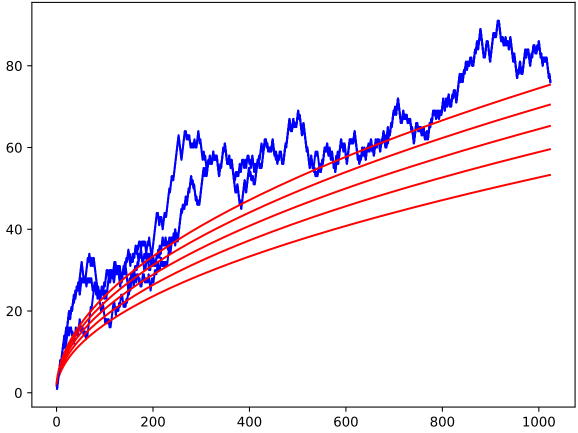
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

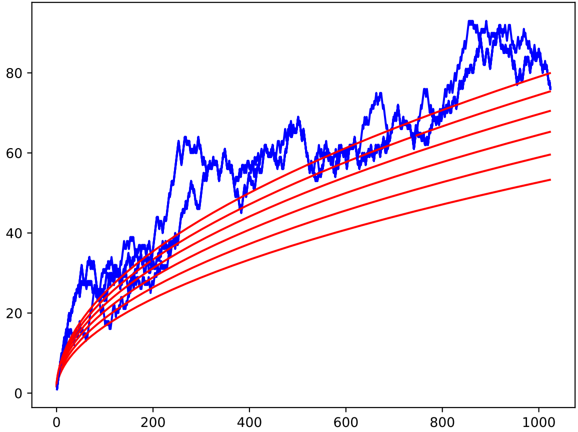
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

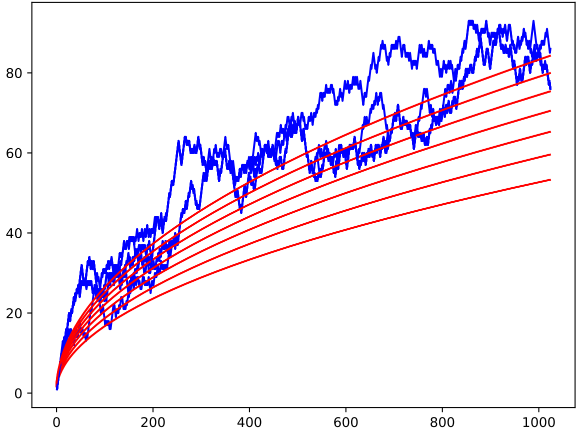
As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Hoeffding's inequality with more walks.

As paths are added, the upper bound **must** increase.



Question: how is the curve being rescaled?

Helpful tool: the union bound.

Given events $(E_1, \ldots, E_k)$, the probability that *some* E_i occurs is

$$\Pr\left(E_1 \vee \cdots \vee E_k\right) \leq \sum_i \Pr(E_i).$$

(**Intuition:** Venn diagram.)

Helpful tool: the union bound.

Given events $(E_1, \ldots, E_k)$, the probability that *some* E_i occurs is

$$\Pr\left(E_1 \vee \cdots \vee E_k\right) \leq \sum_i \Pr(E_i).$$

(**Intuition:** Venn diagram.)

A useful consequence.

Probability that *no* E_i occurs satisfies

$$\Pr\left(\neg E_1 \wedge \cdots \wedge \neg E_k\right) \geq 1 - \sum_i \Pr(E_i).$$

Bounding *many* paths using Hoeffding.

Theorem (Hoeffding's inequality)**.** Consider random variables $(W_1, \ldots, W_k)$ where $W_j := \frac{1}{n}\sum_{i=1}^{n} Z_{j,i}$ with $Z_{j,i} \in [a, b]$, and $Z_{j,i}$ are independent for fixed j, but may be *dependent* for fixed i. With probability at least $1 - \delta$,

$$\max_{j \in [k]} \left(\mathbb{E}(W_j) - W_j\right) \leq (b - a)\sqrt{\frac{\ln(k) + \ln(1/\delta)}{2n}}.$$

Bounding *many* paths using Hoeffding.

Theorem (Hoeffding's inequality)**.** Consider random variables $(W_1, \ldots, W_k)$ where $W_j := \frac{1}{n}\sum_{i=1}^{n} Z_{j,i}$ with $Z_{j,i} \in [a,b]$, and $Z_{j,i}$ are independent for fixed j, but may be *dependent* for fixed i. With probability at least $1-\delta$,

$$\max_{j\in[k]} \left(\mathbb{E}(W_j) - W_j\right) \le (b-a)\sqrt{\frac{\ln(k)+\ln(1/\delta)}{2n}}.$$

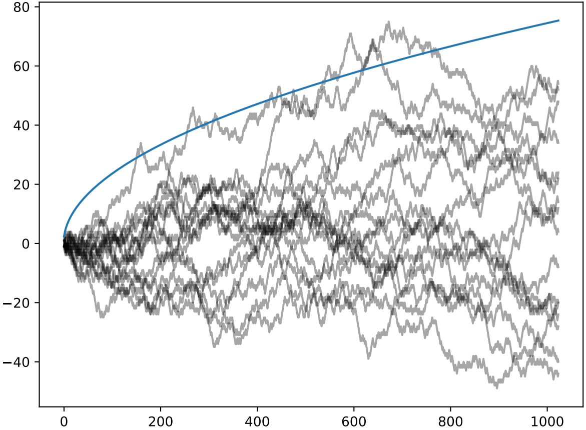
- **Proof.** Define $\epsilon := \sqrt{\ln(k/\delta)/2n}$ and events

$$E_j := \left[\mathbb{E}(W_j) > W_j + \epsilon\right].$$

By Hoeffding, $\Pr(E_j) \le \delta/k$, and by union bound

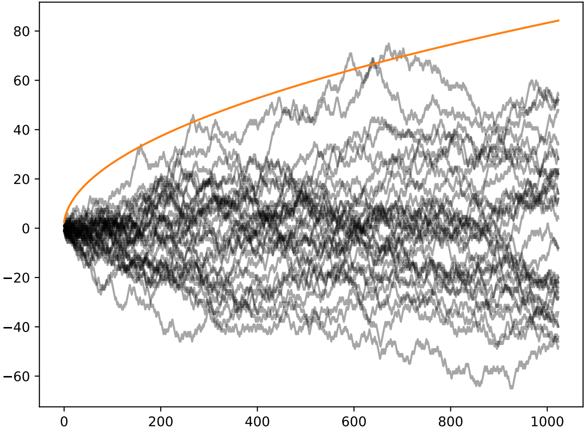
$$\Pr(\cap_j \neg E_j) \ge 1 - \sum_j \Pr(E_j) \ge 1 - \delta.$$

Back to bounding random walks.



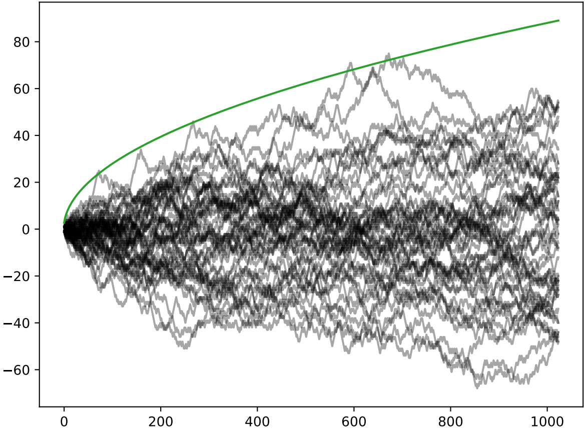
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



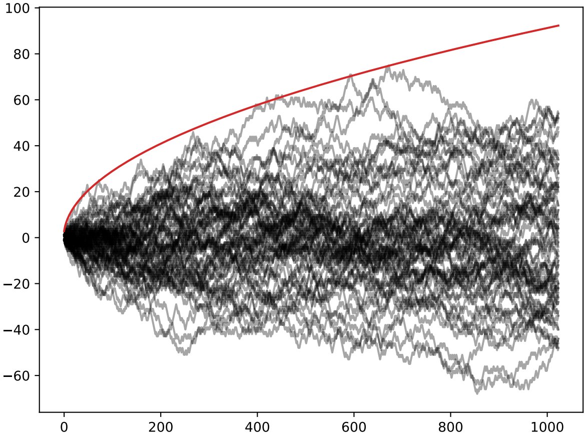
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



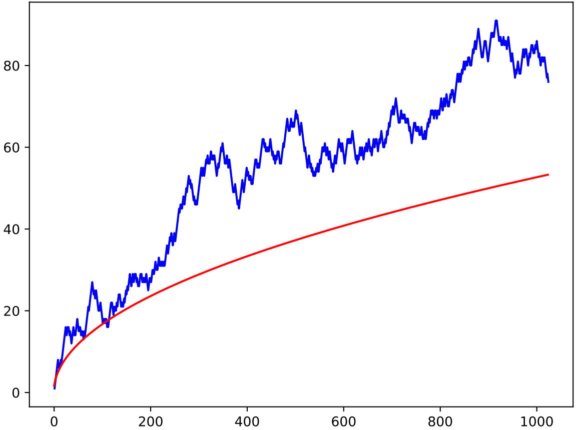
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



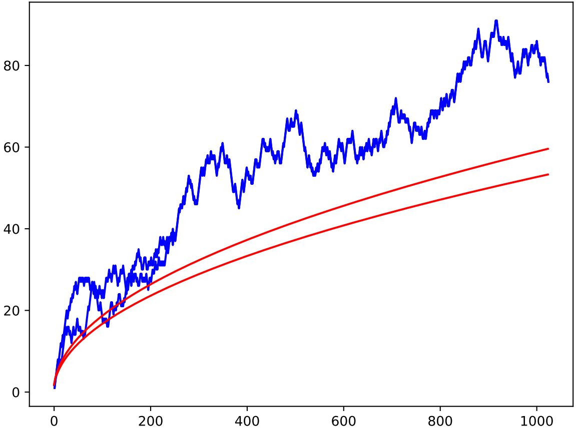
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



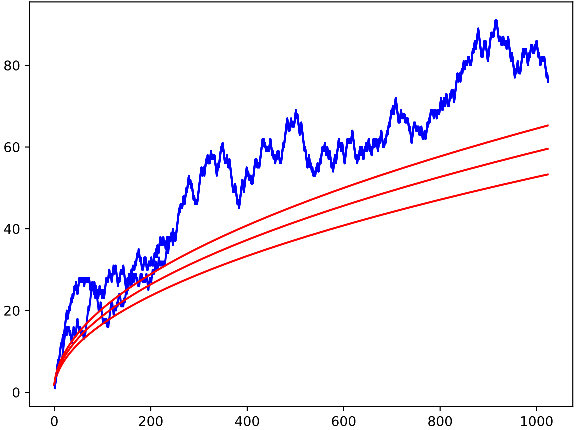
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



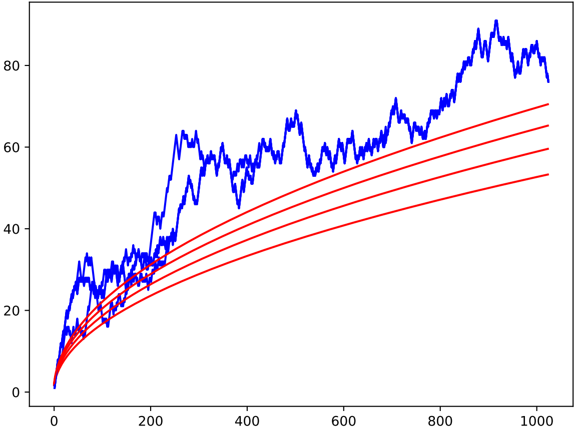
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



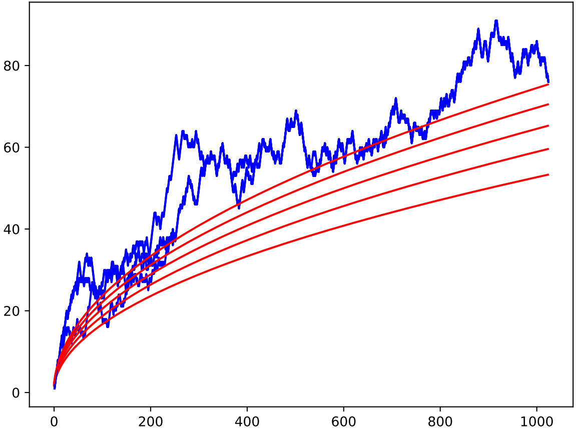
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



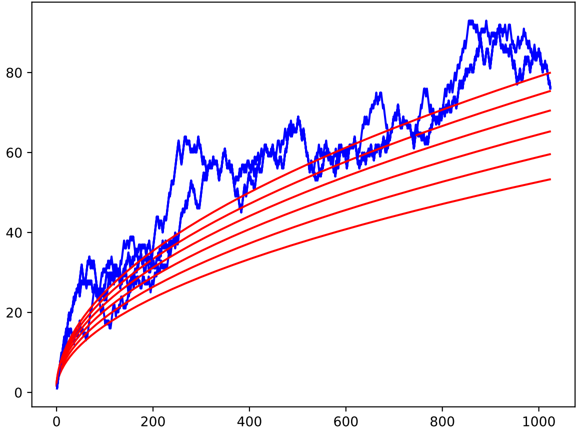
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



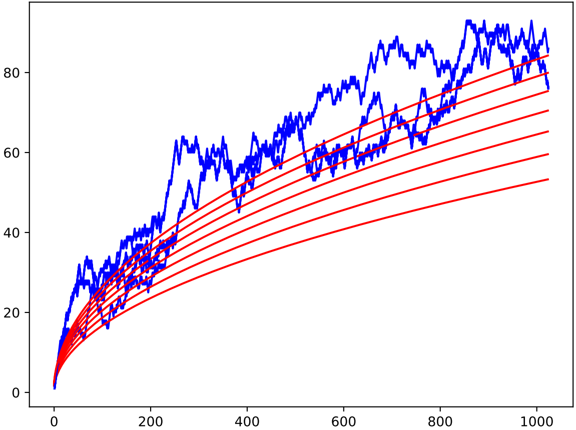
Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Back to bounding random walks.



Question: how is the curve being rescaled?
Answer: $\sqrt{n \cdot \ln(\#\text{paths})}$.

Hoeffding and *one* classifier.

Let f be a *fixed* classifier. Define random variable

$$Z_i := \mathbb{1}\left[f(x_i) \neq y_i\right].$$

Hoeffding gives: with probability at least $1 - \delta$,

$$\begin{aligned}
\Pr[f(X) \neq Y] &= \mathbb{E}(Z_1) \\
&\leq \frac{1}{n}\sum_{i=1}^{n} Z_i + \sqrt{\frac{\ln(1/\delta)}{2n}}. \\
&\leq \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(x_i) \neq y_i] + \sqrt{\frac{\ln(1/\delta)}{2n}}.
\end{aligned}$$

Hoeffding and *many* classifiers.

This is **exactly** the "many paths" bound.

Hoeffding and *many* classifiers.

This is **exactly** the "many paths" bound.

Given classifiers $\mathcal{F}$, similarly: with probability $1-\delta$, **every** $f \in \mathcal{F}$ satisfies

$$\Pr[f(X) \neq Y] \leq \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(x_i) \neq y_i] + \sqrt{\frac{\ln|\mathcal{F}| + \ln(/\delta)}{2n}}.$$

Complexity measures.

Given classifiers $\mathcal{F}$, with probability $1-\delta$, **every** $f \in \mathcal{F}$ satisfies

$$\Pr[f(X) \neq Y] \leq \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[f(x_i) \neq y_i] + \sqrt{\frac{\ln|\mathcal{F}| + \ln(/\delta)}{2n}}.$$

Complexity measures.

Given classifiers $\mathcal{F}$, with probability $1-\delta$, **every** $f \in \mathcal{F}$ satisfies

$$\Pr[f(X) \neq Y] \leq \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(x_i) \neq y_i] + \sqrt{\frac{\ln|\mathcal{F}| + \ln(/\delta)}{2n}}.$$

Generalization.

Given predictors $\mathcal{F}$, with probability at least $1-\delta$, each $f \in \mathcal{F}$ satisfies

$$\mathrm{Risk}(f) \leq \widehat{\mathrm{Risk}}(f) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{\mathrm{Complexity}(\mathcal{F}) + \ln(1/\delta)}{n}}\right).$$

where

$$\mathrm{Risk}(f) := \mathbb{E}\ell(f, X, Y) \qquad \text{and} \qquad \widehat{\mathrm{Risk}}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f, x_i, y_i).$$

Complexity measures.

Given classifiers $\mathcal{F}$, with probability $1-\delta$, **every** $f \in \mathcal{F}$ satisfies

$$\Pr[f(X) \neq Y] \leq \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(x_i) \neq y_i] + \sqrt{\frac{\ln|\mathcal{F}| + \ln(/\delta)}{2n}}.$$

Generalization.

Given predictors $\mathcal{F}$, with probability at least $1-\delta$, each $f \in \mathcal{F}$ satisfies

$$\mathrm{Risk}(f) \leq \widehat{\mathrm{Risk}}(f) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{\mathrm{Complexity}(\mathcal{F}) + \ln(1/\delta)}{n}}\right).$$

where

$$\mathrm{Risk}(f) := \mathbb{E}\ell(f, X, Y) \qquad \text{and} \qquad \widehat{\mathrm{Risk}}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f, x_i, y_i).$$

Remark: holds for all of $\mathcal{F}$, in particular for *selected* $f \in \mathcal{F}$.

Complexity measures.

Given classifiers $\mathcal{F}$, with probability $1 - \delta$, **every** $f \in \mathcal{F}$ (including choice of an algorithm) satisfies

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

where

$$\text{Risk}(f) := \mathbb{E}\ell(f, X, Y) \qquad \text{and} \qquad \widehat{\text{Risk}}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f, x_i, y_i).$$

Complexity measures.

Given classifiers $\mathcal{F}$, with probability $1 - \delta$, **every** $f \in \mathcal{F}$ (including choice of an algorithm) satisfies

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

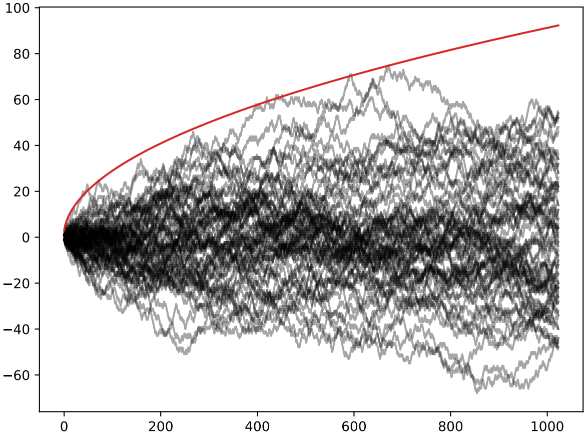
where

$$\text{Risk}(f) := \mathbb{E}\ell(f, X, Y) \qquad \text{and} \qquad \widehat{\text{Risk}}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f, x_i, y_i).$$

Example complexity measures.

- When $|\mathcal{F}| < \infty$, can use $\ln |\mathcal{F}|$.
- For classification, can use **VC dimension**.
- More generally, can use **Rademacher complexity**.

Binomials, random walks, coin tosses, classification.



Questions so far?

A few overfitting/generalization asides.

Aside: scientific experiments.

- **Standard scientific setup:** collect some data, try to fit various hypotheses to it.

Aside: scientific experiments.

- **Standard scientific setup:** collect some data, try to fit various hypotheses to it.
- **This is like checking multiple random walks!** The confidence intervals *must* grow with further hypotheses!

Aside: scientific experiments.

- **Standard scientific setup:** collect some data, try to fit various hypotheses to it.
- **This is like checking multiple random walks!** The confidence intervals *must* grow with further hypotheses!
- This observation is at the core of various "crises" in the application of statistics to science, and give the field of **adaptive data analysis**.

Aside: *covering number* complexities.

We have $\text{Complexity}(\mathcal{F}) \leq \ln |\mathcal{F}|$;
why not **discretize** $\mathcal{F}$ and apply?

Aside: *covering number* complexities.

We have Complexity$(\mathcal{F}) \leq \ln|\mathcal{F}|$;
why not **discretize** $\mathcal{F}$ and apply?

If $\|w - v\|$ small, then **predictions are similar**, meaning

$$|w^\top x - v^\top x| = |(w - v)^\top x| \leq \|w - v\|\|x\|.$$

Aside: *covering number* complexities.

We have $\text{Complexity}(\mathcal{F}) \leq \ln|\mathcal{F}|$;
why not **discretize** $\mathcal{F}$ and apply?

If $\|w - v\|$ small, then **predictions are similar**, meaning

$$|w^\top x - v^\top x| = |(w - v)^\top x| \leq \|w - v\|\|x\|.$$

Consequence: discretizing

$$\mathcal{F} := \left\{x \mapsto w^\top x : \|w\| \leq 1\right\}$$

is same as discretizing $\{w \in \mathbb{R}^d : \|w\| \leq 1\}$,

Aside: *covering number* complexities.

We have $\text{Complexity}(\mathcal{F}) \leq \ln |\mathcal{F}|$;
why not **discretize** $\mathcal{F}$ and apply?

If $\|w - v\|$ small, then **predictions are similar**, meaning

$$|w^\top x - v^\top x| = |(w - v)^\top x| \leq \|w - v\| \|x\|.$$

Consequence: discretizing

$$\mathcal{F} := \left\{ x \mapsto w^\top x : \|w\| \leq 1 \right\}$$

is same as discretizing $\{w \in \mathbb{R}^d : \|w\| \leq 1\}$,
which needs size $\mathcal{O}((1/\epsilon)^d)$, thus

$$\text{Complexity}(\mathcal{F}) \leq \mathcal{O}\left(d \ln(1/\epsilon)\right).$$

Aside: VC dimension.

VC dimension gives a complexity measure for classifiers (binary output).

- **Definition:** the largest data set size (or ∞) which this function class (model) can label in all possible ways.

In earlier bounds, can plug in $\text{Complexity}(\mathcal{F}) \leq \mathcal{O}(\text{VC}(\mathcal{F}) \ln(n))$.

Aside: VC dimension.

VC dimension gives a complexity measure for classifiers (binary output).

- **Definition:** the largest data set size (or ∞) which this function class (model) can label in all possible ways.

In earlier bounds, can plug in $\text{Complexity}(\mathcal{F}) \leq \mathcal{O}(\text{VC}(\mathcal{F}) \ln(n))$.

Examples.

- **Linear separators:** d.
- **ReLU networks:** $\widetilde{\mathcal{O}}(\#\text{parameters} \cdot \#\text{layers})$.

Aside: neural networks.

- Our generalization bound scales with $\frac{\text{Complexity}(\mathcal{F})}{n}$.

Aside: neural networks.

- Our generalization bound scales with $\frac{\text{Complexity}(\mathcal{F})}{n}$.
- $\text{Complexity}(\mathcal{F}) \leq \mathcal{O}(\text{VC}(\mathcal{F}) \ln(n))$, and ReLU networks have $\widetilde{\mathcal{O}}(\#\text{parameters} \cdot \#\text{layers})$.

Aside: neural networks.

- Our generalization bound scales with $\frac{\text{Complexity}(\mathcal{F})}{n}$.
- $\text{Complexity}(\mathcal{F}) \leq \mathcal{O}(\text{VC}(\mathcal{F}) \ln(n))$, and ReLU networks have $\widetilde{\mathcal{O}}(\#\text{parameters} \cdot \#\text{layers})$.
- VC view says: ReLU networks need $\#\text{parameters} \cdot \#\text{layers} < n$.

Aside: neural networks.

- Our generalization bound scales with $\frac{\text{Complexity}(\mathcal{F})}{n}$.
- $\text{Complexity}(\mathcal{F}) \leq \mathcal{O}(\text{VC}(\mathcal{F}) \ln(n))$, and ReLU networks have $\widetilde{\mathcal{O}}(\#\text{parameters} \cdot \#\text{layers})$.
- VC view says: ReLU networks need $\#\text{parameters} \cdot \#\text{layers} < n$.
- **False in practice** (for further discussion, google "deep learning rethinking generalization").

Aside: neural networks.

- Our generalization bound scales with $\frac{\text{Complexity}(\mathcal{F})}{n}$.
- $\text{Complexity}(\mathcal{F}) \leq \mathcal{O}(\text{VC}(\mathcal{F}) \ln(n))$, and ReLU networks have $\widetilde{\mathcal{O}}(\#\text{parameters} \cdot \#\text{layers})$.
- VC view says: ReLU networks need $\#\text{parameters} \cdot \#\text{layers} < n$.
- **False in practice** (for further discussion, google "deep learning rethinking generalization").
- Can use other bounds: "small norm" property(?) of SGD and norm-based generalization.

Aside: Rademacher complexity.

VC is combinatorial/discrete;
Rademacher is a generalization that allows real-valued predictors.

- **Definition.** Let $(\epsilon_1, \ldots, \epsilon)$ be **random sign** ("Rademacher") random variables, meaning $\Pr[\epsilon_i = +1] = \Pr[\epsilon_i = -1] = 1/2$, and all are independent. Then

$$\mathrm{Rad}(\mathcal{F}) := \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i).$$

- **Intuition:** how well $\mathcal{F}$ fits **random signs**.
- **Remark:** VC was **worst case** sign patterns.
- **Remark**: can plug in $\mathrm{Complexity}(\mathcal{F}) \leq n \cdot \mathrm{Rad}(\mathcal{F})^2$.

Aside: Rademacher complexity.

VC is combinatorial/discrete;
Rademacher is a generalization that allows real-valued predictors.

- **Definition.** Let $(\epsilon_1, \ldots, \epsilon)$ be **random sign** ("Rademacher") random variables, meaning $\Pr[\epsilon_i = +1] = \Pr[\epsilon_i = -1] = 1/2$, and all are independent. Then

$$\mathrm{Rad}(\mathcal{F}) := \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i).$$

- **Intuition:** how well $\mathcal{F}$ fits **random signs**.
- **Remark:** VC was **worst case** sign patterns.
- **Remark**: can plug in $\mathrm{Complexity}(\mathcal{F}) \leq n \cdot \mathrm{Rad}(\mathcal{F})^2$.

Linear predictors with norm $\leq R$: then $\mathrm{Rad}(\mathcal{F}) \leq R/\sqrt{n}$.
Neural networks with weight matrices $(W_1, \ldots, W_L)$:

$$\mathrm{Rad}(\mathcal{F}) = \widetilde{\mathcal{O}}\left((\text{gross stuff}) \cdot \prod_{i=1}^{L} \sigma_{\max}(W_i) \right).$$

Aside: ridge regression.

Aside: ridge regression.

Recall the *Ridge Regression Estimator* in matrix/vector form:

$$\hat{w}_\lambda := \arg\min_{w \in \mathbb{R}^d} \frac{1}{2n}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2.$$

Role of λ in standard learning theory questions:

Aside: ridge regression.

Recall the *Ridge Regression Estimator* in matrix/vector form:

$$\hat{w}_\lambda := \arg\min_{w \in \mathbb{R}^d} \frac{1}{2n}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2.$$

Role of λ in standard learning theory questions:

- **Representation:** not just a linear predictor, moreover $\mathcal{F}_\lambda := \left\{x \mapsto w^\top x : \|w\| \leq \sqrt{1/\lambda}\right\}$.

Aside: ridge regression.

Recall the *Ridge Regression Estimator* in matrix/vector form:

$$\hat{w}_\lambda := \arg\min_{w \in \mathbb{R}^d} \frac{1}{2n} \|Xw - y\|_2^2 + \frac{\lambda}{2} \|w\|_2^2.$$

Role of λ in standard learning theory questions:

- **Representation:** not just a linear predictor, moreover $\mathcal{F}_\lambda := \left\{ x \mapsto w^\top x : \|w\| \leq \sqrt{1/\lambda} \right\}$.
- **Optimization:** as discussed in class, to achieve accuracy ϵ, gradient descent needs $\frac{\sigma_{\max}(X)+\lambda}{\sigma_{\min}(X)+\lambda} \ln(1/\epsilon)$ iterations.

Aside: ridge regression.

Recall the *Ridge Regression Estimator* in matrix/vector form:

$$\hat{w}_\lambda := \arg\min_{w \in \mathbb{R}^d} \frac{1}{2n} \|Xw - y\|_2^2 + \frac{\lambda}{2} \|w\|_2^2.$$

Role of λ in standard learning theory questions:

- **Representation:** not just a linear predictor, moreover $\mathcal{F}_\lambda := \left\{ x \mapsto w^\top x : \|w\| \leq \sqrt{1/\lambda} \right\}$.
- **Optimization:** as discussed in class, to achieve accuracy ϵ, gradient descent needs $\frac{\sigma_{\max}(X)+\lambda}{\sigma_{\min}(X)+\lambda} \ln(1/\epsilon)$ iterations.
- **Generalization:** via **Rademacher complexity** and above representation bound, get $\text{Complexity}(\mathcal{F}_\lambda) \leq 1/\lambda$.

Aside: online learning and the perceptron algorithm.

(Details in lecture.)

Summary (of overfitting/generalization).

Summary (of overfitting/generalization).

- **Intuition:** predictors/model too flexible $\implies$ overfit (unless tons of data).
- **Rigorous form:** we have **generalization bounds**, namely bounds between training and test errors of the form

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

and we gave a few definitions of Complexity($\mathcal{F}$).