Introduction to Machine Learning (IML)

LECTURE #6: MODEL SELECTION

236756 – 2024 SPRING – TECHNION

LECTURER: NIR ROSENFELD

Today

- **Part II**: *the different aspects of learning*
    1. Statistics: generalization and PAC theory (finish up)
    2. Modeling: (today)
        - Model selection
        - Regularization
        - Evaluation and validation
    3. Optimization: convexity, gradient descent
    4. Practical aspects and potential pitfalls

PAC and VC – wrap up

Recall

- **Remember**: everything in learning is a random variable (sample set, learned model, performance, …)
- **Definition**:

  H is PAC-learnable if $\exists A, \exists\, m_H(\epsilon, \delta) \in \text{poly}\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$

  such that $\forall D$ (realizable) and $\forall \epsilon, \delta \in [0,1]$, if $m \geq m_H(\epsilon, \delta)$, then:

  $$P_{S\sim D^m}(L_D(h_S) \geq \epsilon) \leq \delta$$

- For **finite model classes**, following bound holds:

  $$P_{S\sim D^m}(L_D(h_S) \geq \epsilon) \leq |H|e^{-m\epsilon} \leq \delta$$

- **Interpretations:**

  1. $m \geq \frac{\log|H| + \log\frac{1}{\delta}}{\epsilon}$ 2. $\epsilon \geq \frac{\log|H| + \log\frac{1}{\delta}}{m} \approx \frac{1}{m}$

- **What about non-realizable?**

Recall

- **Remember**: everything in learning is a random variable (sample set, learned model, performance, …)
- **Definition**:

  H is Agnostic-PAC-learnable if $\exists A, \exists\, m_H(\epsilon,\delta) \in \text{poly}\left(\frac{1}{\epsilon},\frac{1}{\delta}\right)$ such that $\forall D$ and $\forall \epsilon,\delta \in [0,1]$, if $m \geq m_H(\epsilon,\delta)$, then:

  $$P_{S\sim D^m}\left(L_D(h_S) - L_D(h^*) \geq \epsilon\right) \leq \delta$$

- For **finite model classes**, got following bound:

  $$P_{S\sim D^m}\left(L_D(h_S) - L_D(h^*) \geq \epsilon\right) \leq 2|H|e^{-2m\epsilon^2} \leq \delta$$

- **Interpretations:**

  1. $m \geq \frac{\log 2|H| + \log\frac{1}{\delta}}{2\epsilon^2}$ 2. $\epsilon \geq \sqrt{\frac{\log 2|H| + \log\frac{1}{\delta}}{2m}} \approx \frac{1}{\sqrt{m}}$ (vs. $\frac{1}{m}$ in realizable case)

- **What about infinite classes?**

Beyond finite classes

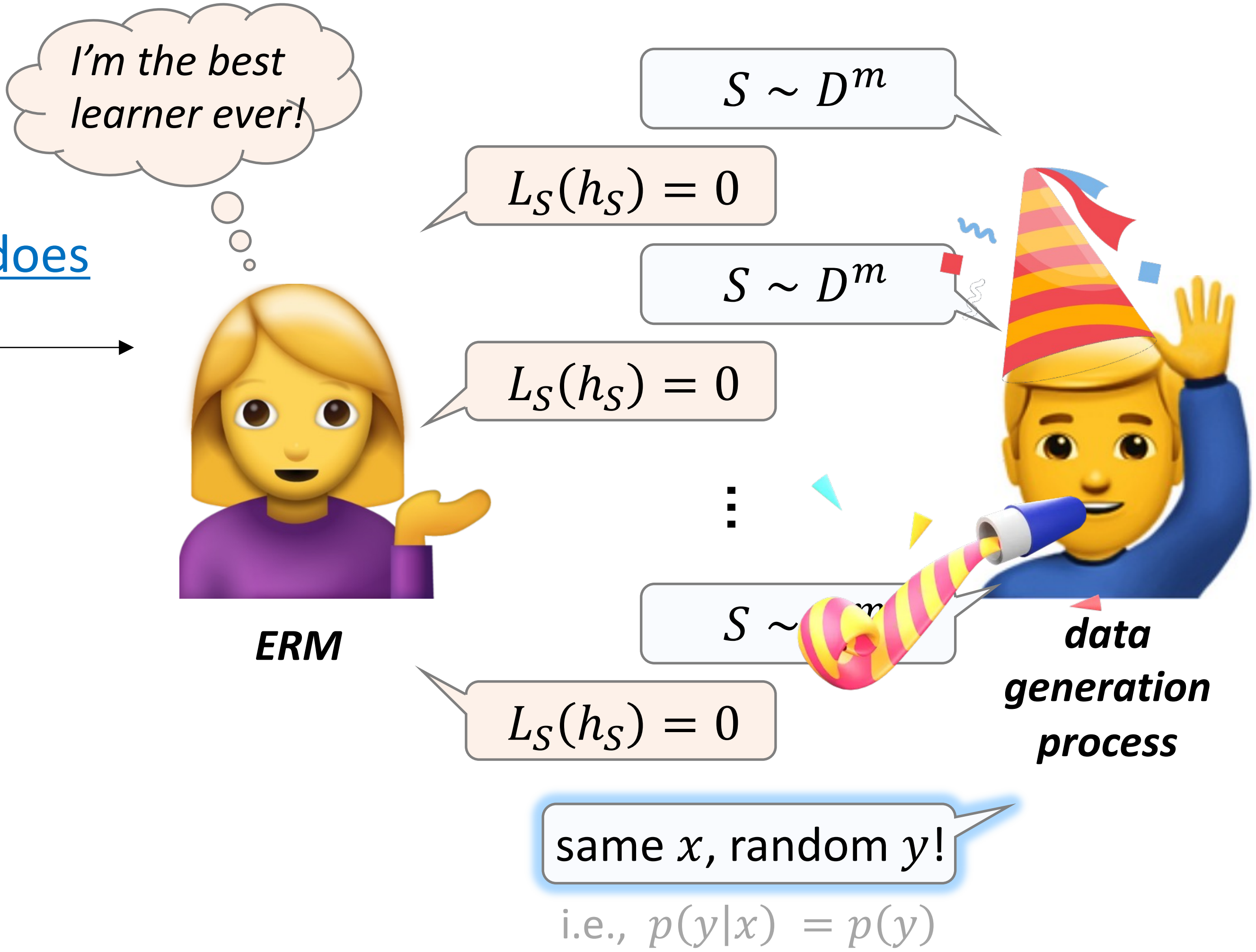
- The previous bound showed finite H are learnable (with dependence on $\log|H|$)
- Is this bound useful for…
  - decision trees? (think!)
  - linear halfspaces? (think!)
  - RBF kernels? (think!)
  - 1D thresholds? (think!)
- **Q**: does $|H| = \infty$ mean we can't learn?
- **A**: Not necessarily! (we proved finite H => learnable, not the negation)
- **Recall**: for 1D thresholds (infinite class!), we showed $\epsilon \approx O\left(\frac{1}{m}\right)$ (under realizability)
- **Conclusion**: $|H|$ is probably not the "correct" measure
- **Note**: there is no single "correct" measure, only *useful* measures; we will see one next

VC dimension

- **Idea:**
  consider not what each h is, but what it does

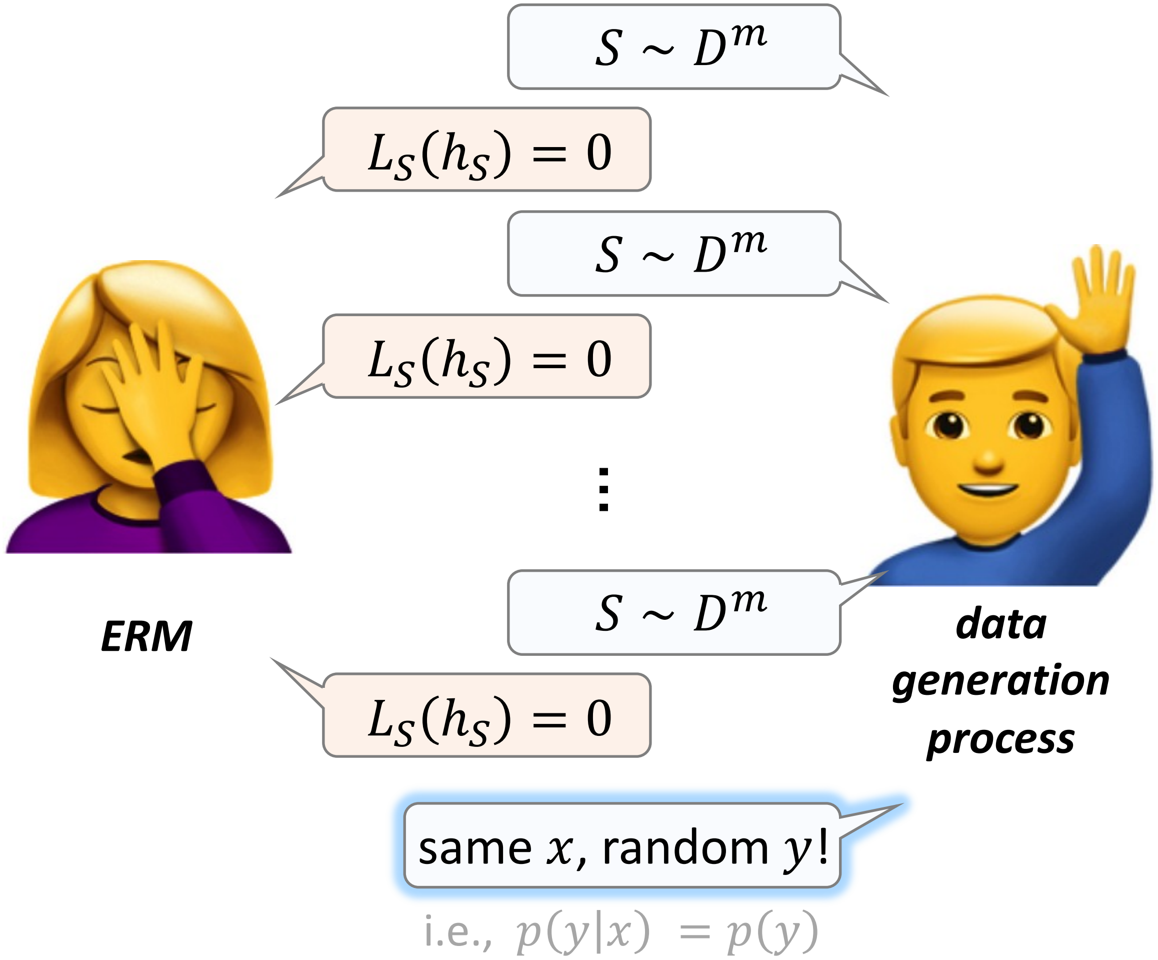
VC dimension

- **Idea:**
  consider not what each h is, but what it does
- **Intuition – when learning fails**



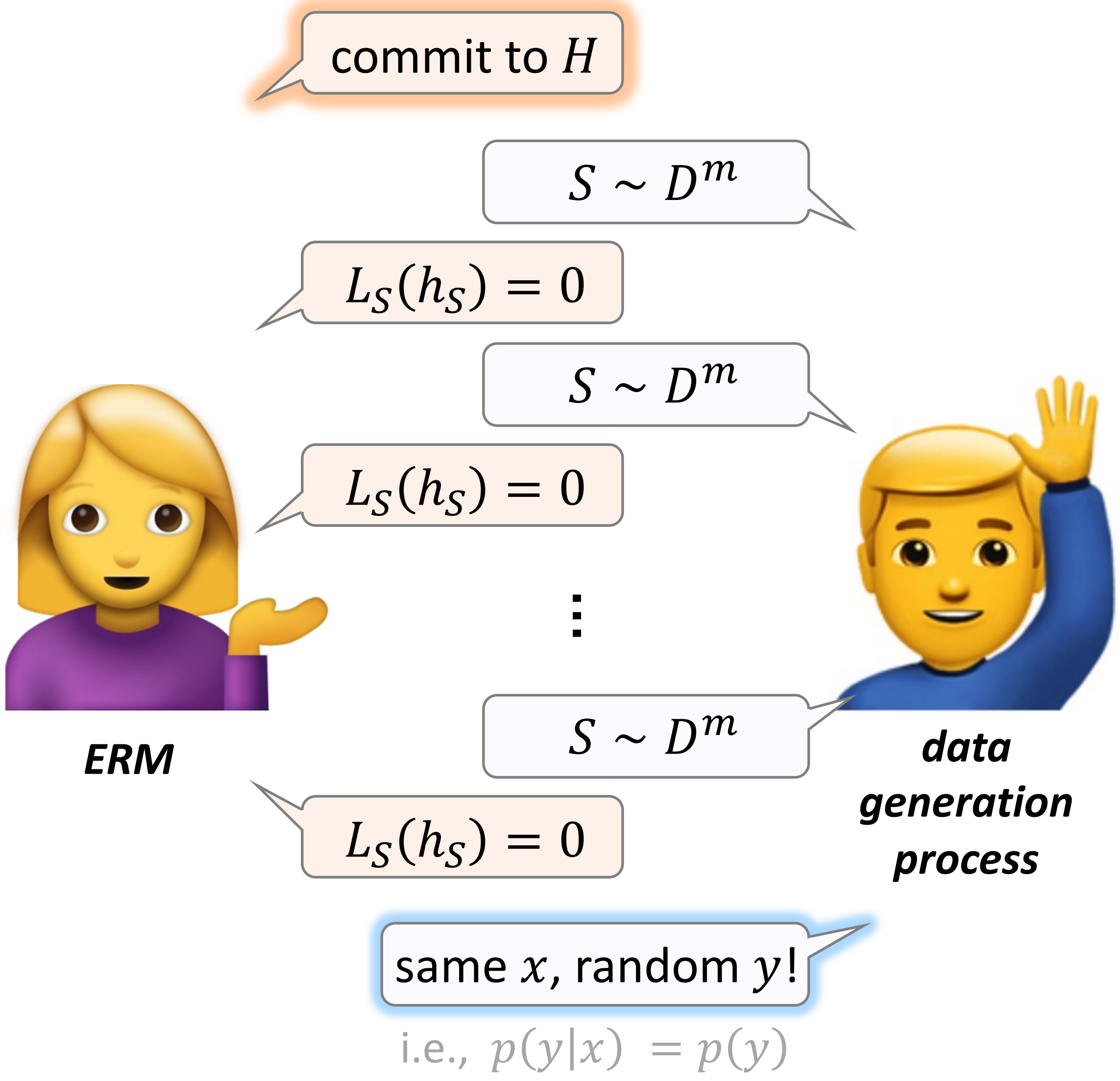
VC dimension

- **Idea:**
  consider not what each h <u>is</u>, but what it <u>does</u>
- **Intuition – when learning fails**
- **Take away:**
  "explaining everything ≡ explaining nothing"
- **VC theory** quantifies this idea (Vapnik–Chervonenkis)



VC dimension

- **Idea:**
  consider not what each h <u>is</u>, but what it <u>does</u>
- **Intuition – when learning fails**
- **Take away:**
  "explaining everything ≡ explaining nothing"
- **VC theory** quantifies this idea (Vapnik–Chervonenkis)
- **Remember**: learnability is property of H!
- The **VC dimension** of H is the largest set on which $L_S = 0$ is possible for <u>any</u> labeling
- **Main result**: learning breaks once H can perfectly fit arbitrary label assignments
  (= noise! remember overfitting?)



VC dimension

- The notion of "explaining everything" is defined using the idea of *shattering* a set of examples:

- **Definition:** Let $C = \{x_i\} \in \mathcal{X}^m$, then *$H$ shatters C* if:

  $$\forall \{y_i\} \in \{\pm 1\}^m \quad \exists\, h \in H \quad \text{s.t.} \quad h(x_i) = y_i \ \forall i \in [m]$$

i.e., for any labeling of C, applying ERM to $S(C) = \{(x_i, y_i)\}_{i=1}^m$ gives $L_{S(C)}\big(h_{S(C)}\big) = 0$.

- **Definition**: The *VC-dimension of H* is the size of the largest set that H shatters, denoted $VC(h)$

i.e., $VC(H) = m$ if exists C of size m that H shatters, but H does not shatter all larger sets

Finding VC

- Given H, how do we prove $VC(H)$?
- **Rules of the game**:
  1. guess some m
  2. show <u>exists</u> C of size m that H shatters $\Longrightarrow VC \le m$
  3. show H does not shatter <u>all</u> sets C of size $m+1 \Longrightarrow VC \ge m$

- **Examples:**
  1. 1D thresholds [on board]
  2. 1D intervals [on board]
  3. Linear halfspaces? (tirgul!)
  4. RBF kernel? (think!)

1. m: $\exists\{x\}\ \forall y\ \exists$ perfect h (shatter)
2. $m+1$: $\forall\{x\}\ \exists y\ \forall h$ errs (<u>can't</u> shatter)

VC dimension

- The VC dimension of H tells us how many samples are needed for learning
- This is called the *sample complexity* of H, denoted $m_H(\epsilon, \delta)$ (look familiar?)
- **Fundamental theorem of learning:** (partial; won't prove)
  If $VC(H) \leq \infty$, then H is:
  1. **PAC-learnable** with *vs.* $\log|H|$
     $$m_H(\epsilon, \delta) = \Theta\left(\frac{VC(H)\log 1/\epsilon + \log 1/\delta}{\epsilon}\right)$$
  2. **Agnostic PAC-learnable** with
     $$m_H(\epsilon, \delta) = \Theta\left(\frac{VC(H) + \log 1/\delta}{\epsilon^2}\right)$$

what about $|H| = \infty$*?*

exact characterization

(almost) same ϵ, δ *rates as in finite* H

- **Think**: what is the VC of a finite H?
- **Bonus**: If $VC(H) = \infty$, then H is essentially not learnable (in the PAC sense)
- But there are other notions of learning! (e.g., SVM-RBF has $VC = \infty$, but is still great for learning)

Uses and limitations

- Say you want to learn with SVM (and assume you know the VC of halfspaces*)
- **Theory is your friend:**
  - Theory asks: tell me your desired ϵ and δ (this is unavoidable!)
  - Theory says: you need (order of) m examples!

- **Great, but need to remember:**

1. VC and PAC are worst-case (are you really doomed if you only have $< m$ samples?)
2. Even agnostic PAC relies on distributional assumptions (the elephant in the room: i.i.d.)
3. ERM is (computationally) hard! SVM minimizes *hinge loss*, not 0/1 loss (UC assumes exact ERM)
4. Guarantees are probabilistic but (in most cases) **you only see one sample set** → **up next!**

* Just wait till the tirgul!

Model Selection

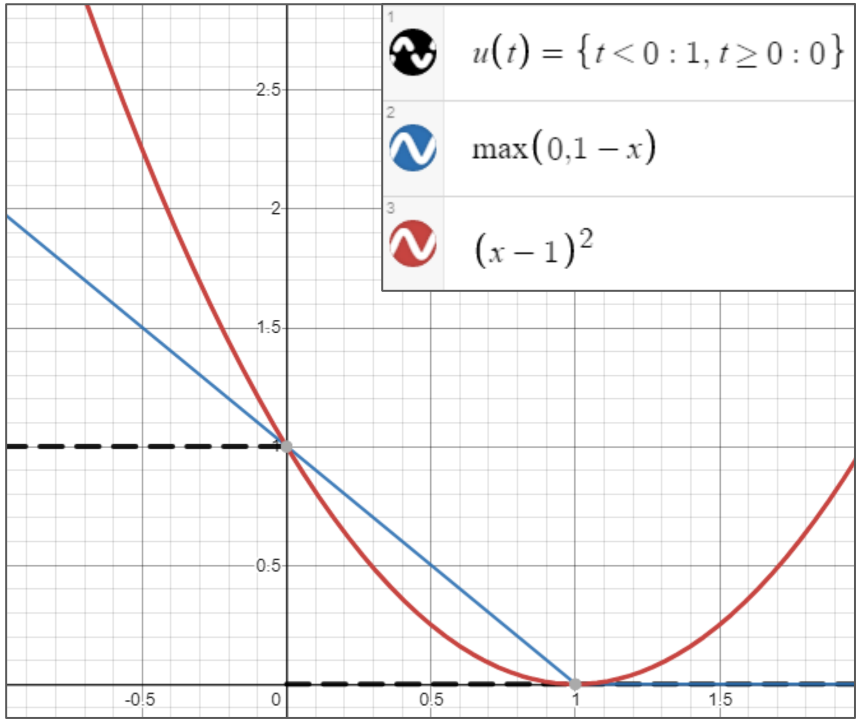
A tale of bias and variance

- **Recall**: PAC asks: $P_{S\sim D^m}(L_D(h_S) - L_D(h^*) \geq \epsilon) \leq \delta$
- **Alternative**: expected error $\mathbb{E}_{S\sim D^m}[L_D(h_S)]$
- **Think**: why are we not considering $L_D(h^*)$ here?
- To simplify the math, we'll analyze **squared loss:**

$$L_D^{sqr}(h) = \mathbb{E}_{(x,y)\sim D}[(h(x) - y)^2]$$

(instead of 0/1 or hinge; will revisit when we talk about regression)

- **Let's analyze!** [on board]
- **Definitions:**
  - Expected label: $\bar{y}(x) = \mathbb{E}_{y\sim D_{Y|X=x}}[y] \in [0,1]$
    (for squared error, this is the optimal classifier; won't prove)
  - Expected loss (given h): $\mathbb{E}_{(x,y)\sim D}[(h(x) - y)^2]$
  - Expected "classifier": $\bar{h} = \mathbb{E}_{S\sim D^m}[h_S]$



Error decomposition

$$\mathbb{E}_{S\sim D^m}\left[L_D^{sqr}(h_S)\right] = \mathbb{E}_{S\sim D^m}\mathbb{E}_{(x,y)\sim D}[(h_S(x) - y\,)^2]$$

…

$$= \mathbb{E}_{x,y}\,[(\bar{y}(x) - y)^2] + \mathbb{E}_x\left[\left(\bar{h}(x) - \bar{y}(x)\right)^2\right] + \mathbb{E}_{S,x}\left[\left(h_S(x) - \bar{h}(x)\right)^2\right]$$

Full derivation here:
https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

Bias-Variance decomposition

given that we chose hypothesis hs

- Got elegant decomposition – three interpretable sources of error:

$$\underbrace{\mathbb{E}_{S\sim D^m}\left[L_D^{sqr}(h_S)\right]}_{\textit{expected error}} = \underbrace{\mathbb{E}_{x,y}\left[(\bar{y}(x) - y)^2\right]}_{\textit{noise}} + \underbrace{\mathbb{E}_x\left[\left(\bar{h}(x) - \bar{y}(x)\right)^2\right]}_{\textit{bias}^2} + \underbrace{\mathbb{E}_{S,x}\left[\left(h_S(x) - \bar{h}(x)\right)^2\right]}_{\textit{variance}}$$

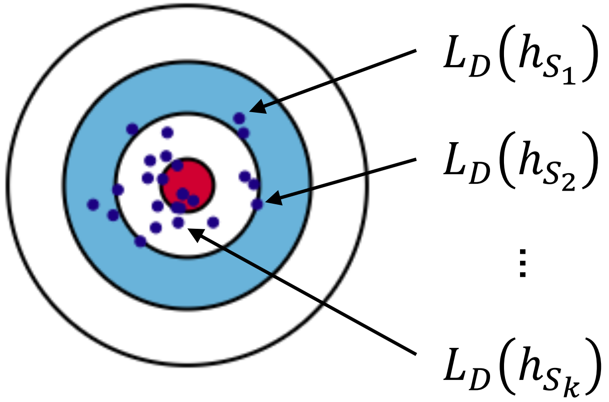
- **Noise**:
    - For squared loss, $\bar{y}$ is an optimal predictor (a.k.a. Bayes-optimal; won't show)
    - Property of data distribution (i.e., the statistical relation between x and y)
    - Does not depend on choice of model (hence called "irreducible" error)

Bias-Variance decomposition

- Got elegant decomposition – three interpretable sources of error:

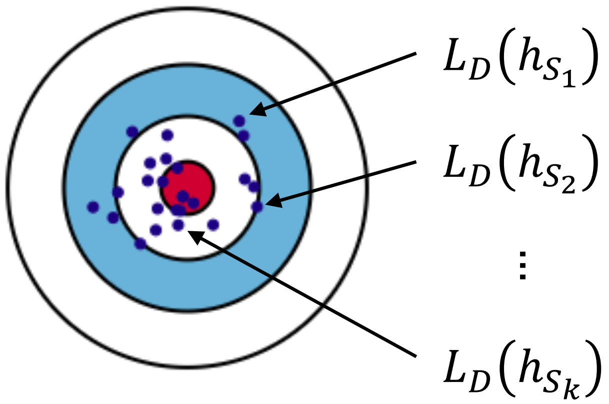
$$\underbrace{\mathbb{E}_{S\sim D^m}\left[L_D^{sqr}(h_S)\right]}_{\textbf{\textit{expected error}}} = \underbrace{\mathbb{E}_{x,y}\left[(\bar{y}(x)-y)^2\right]}_{\textbf{\textit{noise}}} + \underbrace{\mathbb{E}_x\left[\left(\bar{h}(x)-\bar{y}(x)\right)^2\right]}_{\textbf{\textit{bias}}^2} + \underbrace{\mathbb{E}_{S,x}\left[\left(h_S(x)-\bar{h}(x)\right)^2\right]}_{\textbf{\textit{variance}}}$$

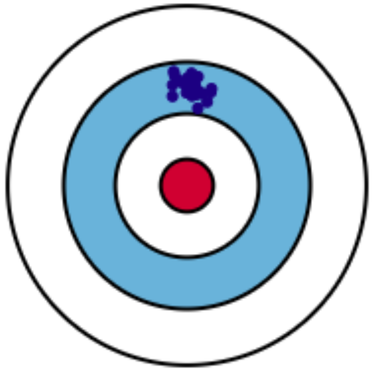
- **Bias**:
  - Quantifies how well our chosen <u>model class</u> fits the data (on average)
  - Does not depend on the data sampled (but does depend on data size) nor on learned model
- **Variance**: (of algorithm; w.r.t. S)
  - Measures how learned models h_S vary
    (how "sensitive" the learning algorithm is to changes in its input S)
  - Average model $\bar{h}$ as reference point
    (asks: relative to $\bar{h}$, how "specialized" is h_S to S?)
  - Does not consider predictive error directly (no dependence on y)



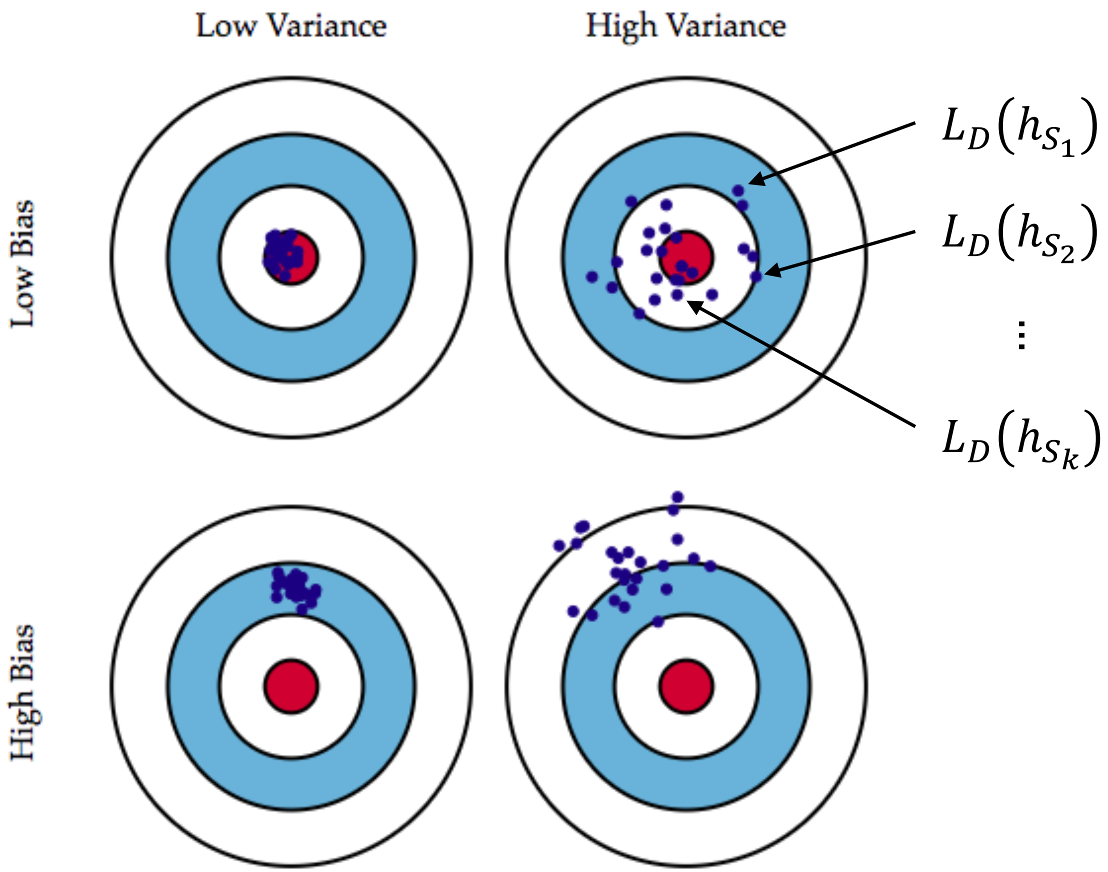
http://scott.fortmann-roe.com/docs/BiasVariance.html

High Variance





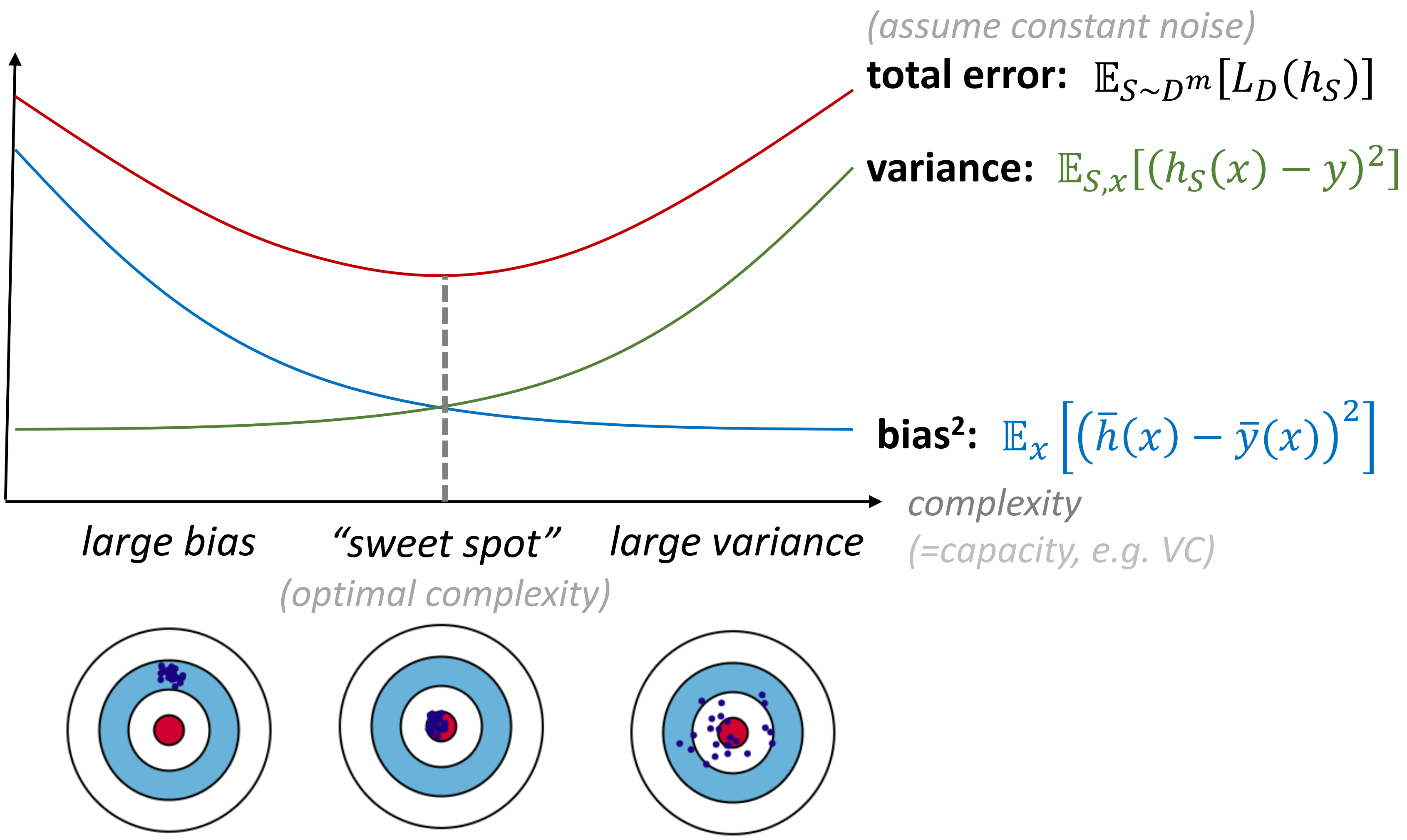
http://scott.fortmann-roe.com/docs/BiasVariance.html



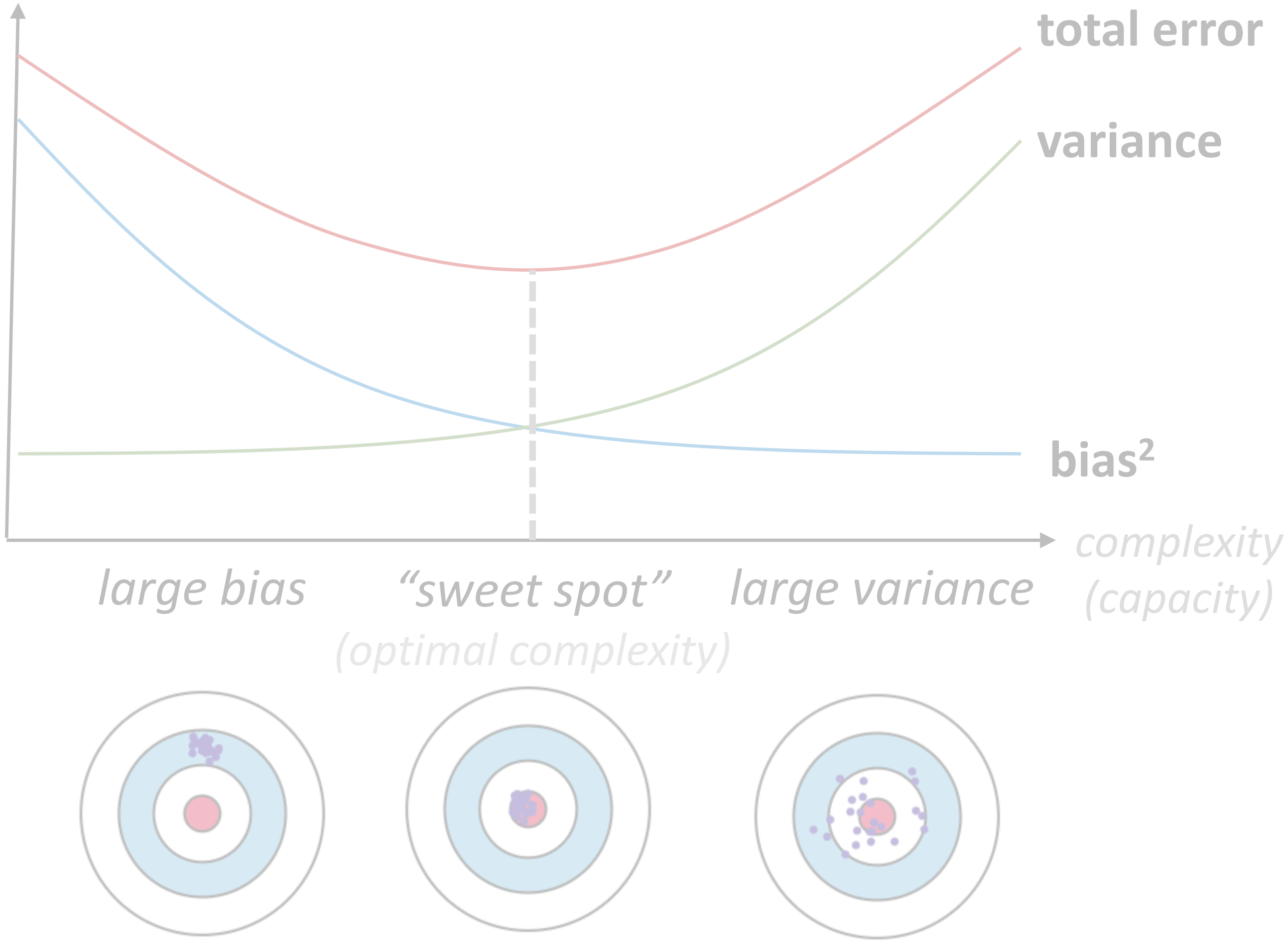
Q: Can we choose what regime we're in?

A: Yes, to some extent – by controlling model complexity.

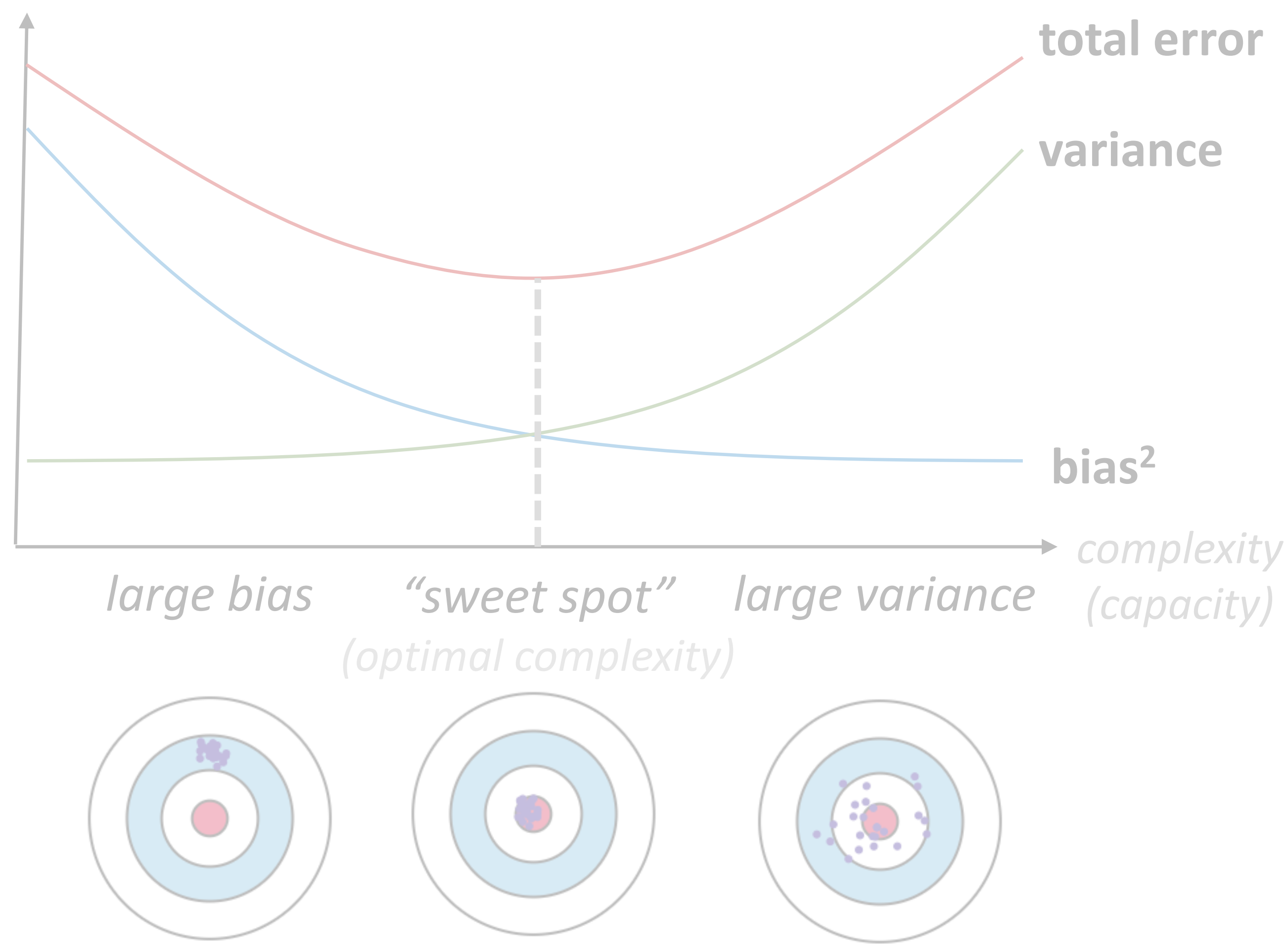
http://scott.fortmann-roe.com/docs/BiasVariance.html

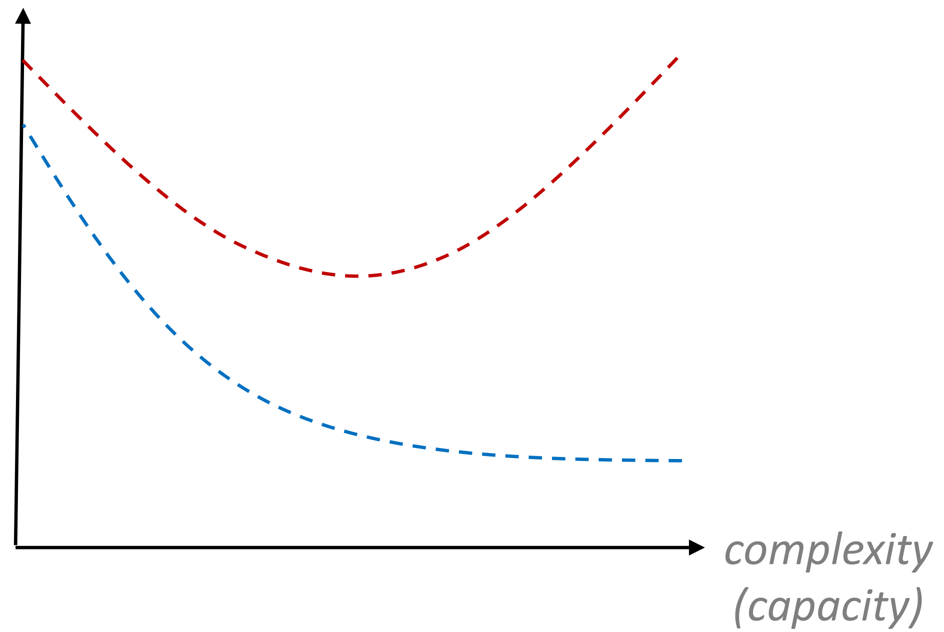


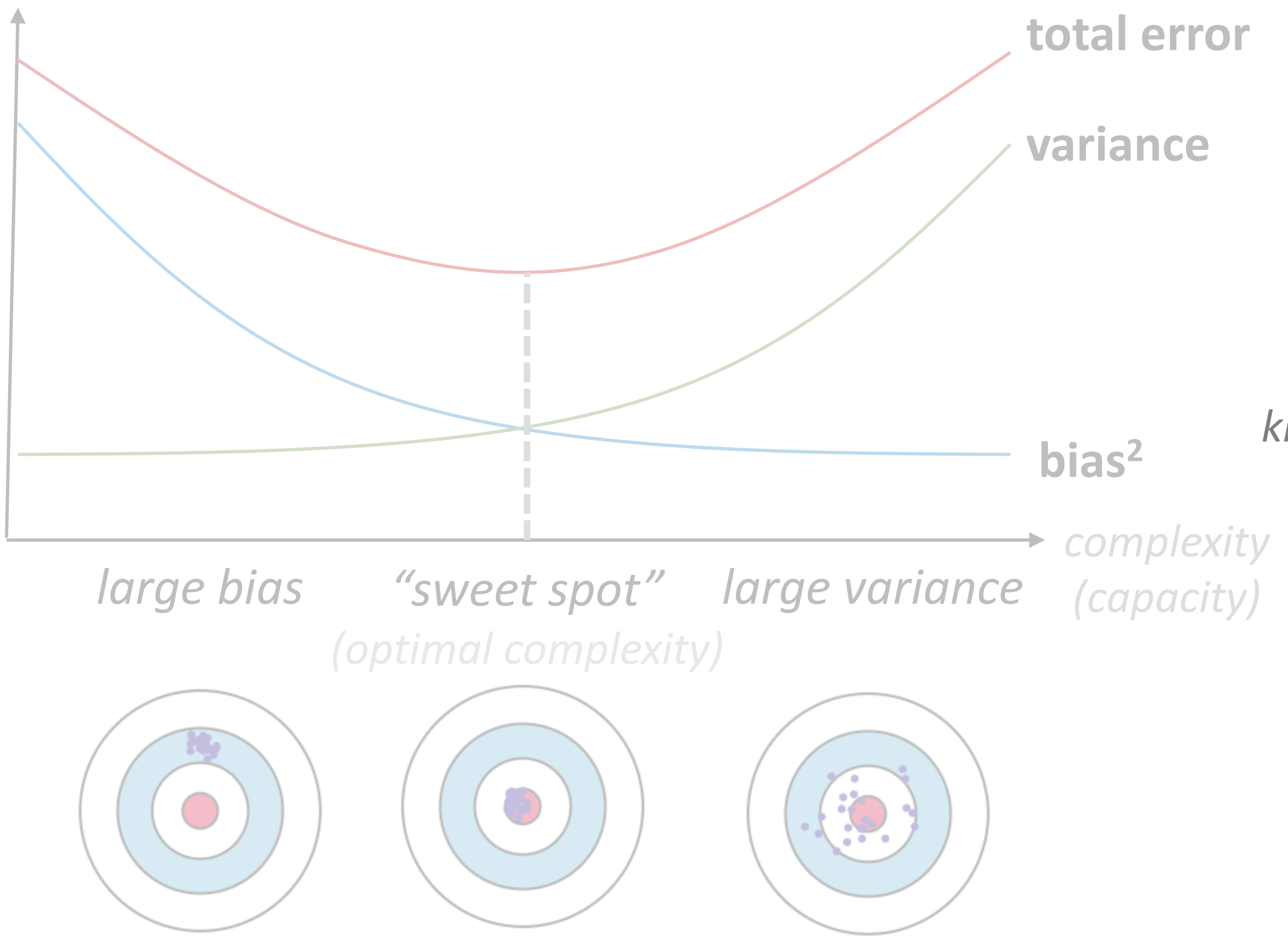
(assume constant noise)
total error: $\mathbb{E}_{S\sim D^m}[L_D(h_S)]$
variance: $\mathbb{E}_{S,x}[(h_S(x)-y)^2]$
bias²: $\mathbb{E}_x\left[\left(\bar{h}(x)-\bar{y}(x)\right)^2\right]$
complexity
(=capacity, e.g. VC)
large bias
"sweet spot"
(optimal complexity)
large variance



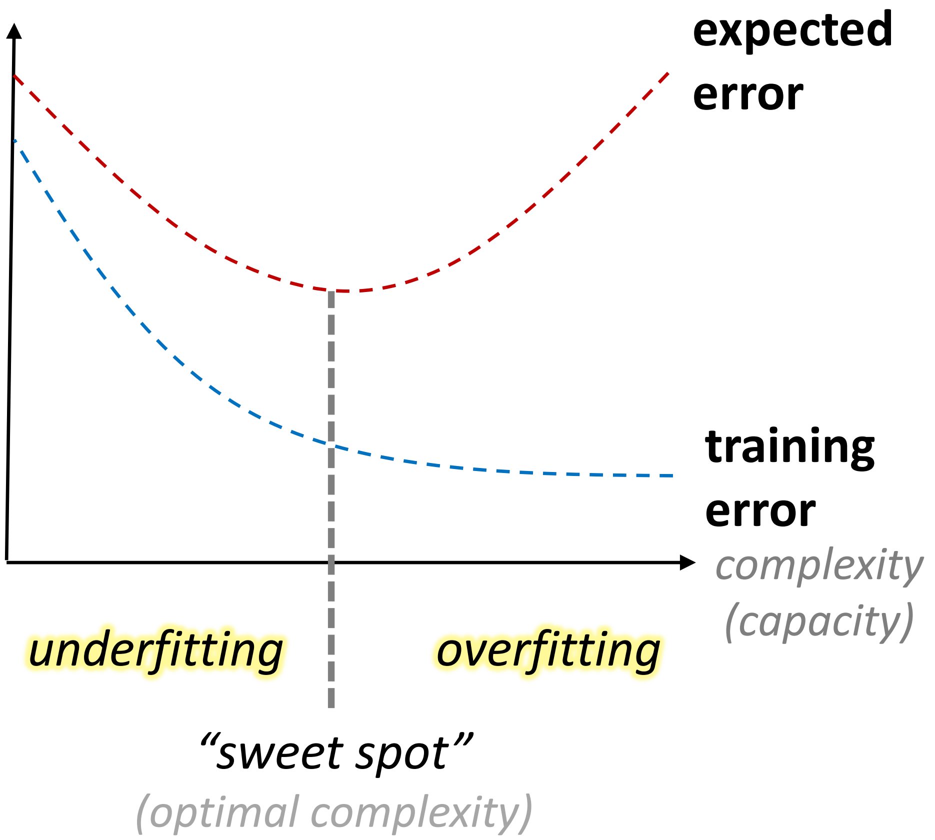
we've seen this sort of plot before…


total error
variance
bias^2
complexity
(capacity)
large bias
“sweet spot”
(optimal complexity)
large variance


complexity
(capacity)



now we know why!



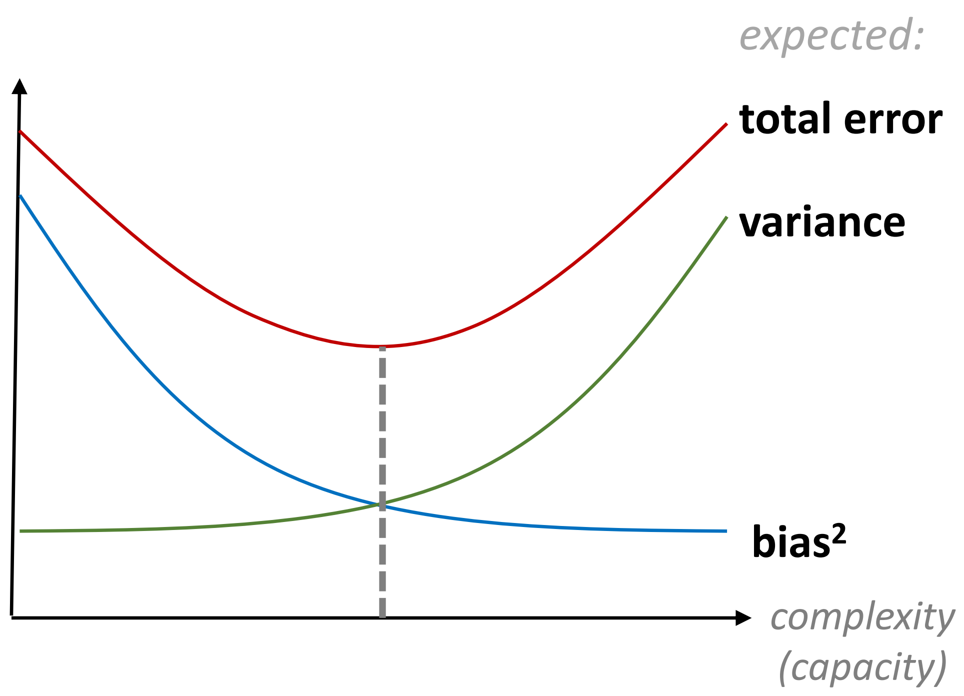
Q: What is same? What is different?

From theory to practice

- **Recall**: goal in learning is to reduce expected error
- **Decomposition says:**

  error = noise + bias^2 + variance

- **Take-away I**:
  Inherent tradeoff between bias and variance
- And we can control operating point (to some extent)
  - ➢ Large bias? Increase complexity!
  - ➢ Large variance? decrease complexity!
- Many learning problems have U-shaped errors
- We'd like to find the "sweet spot" (will see how soon)

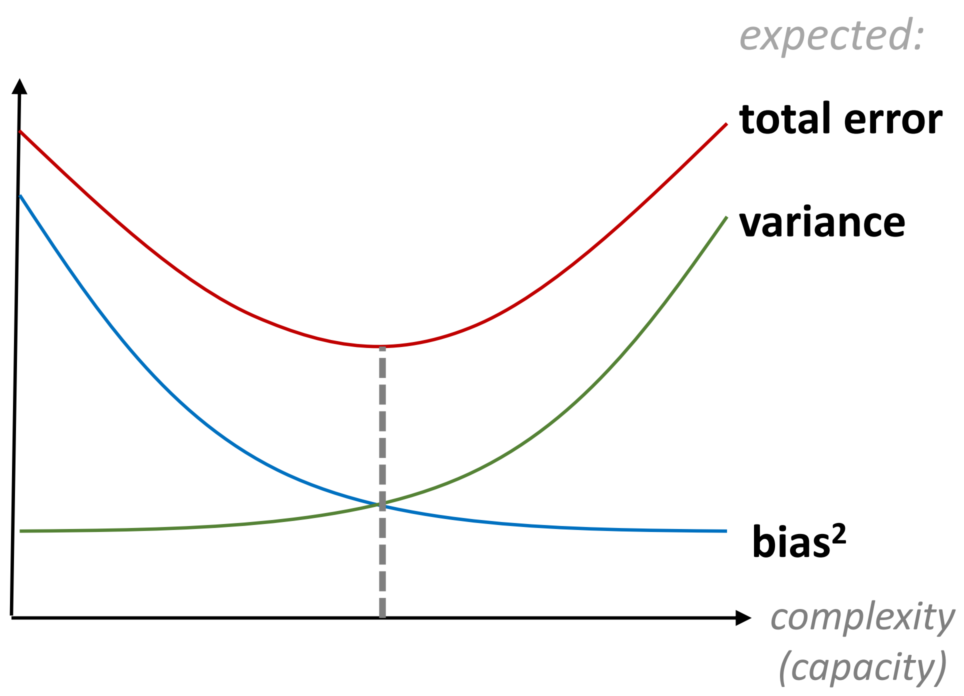


From theory to practice

- **Recall**: goal in learning is to reduce expected error
- **Decomposition says:**

  error = noise + bias^2 + variance

- **Take-away II:**
  Effective modeling = "choose your battles"
- Easier to individually target each source of error
- **One solution**: hard-code into learning objective
- (Hint: we've actually already seen this in action!)

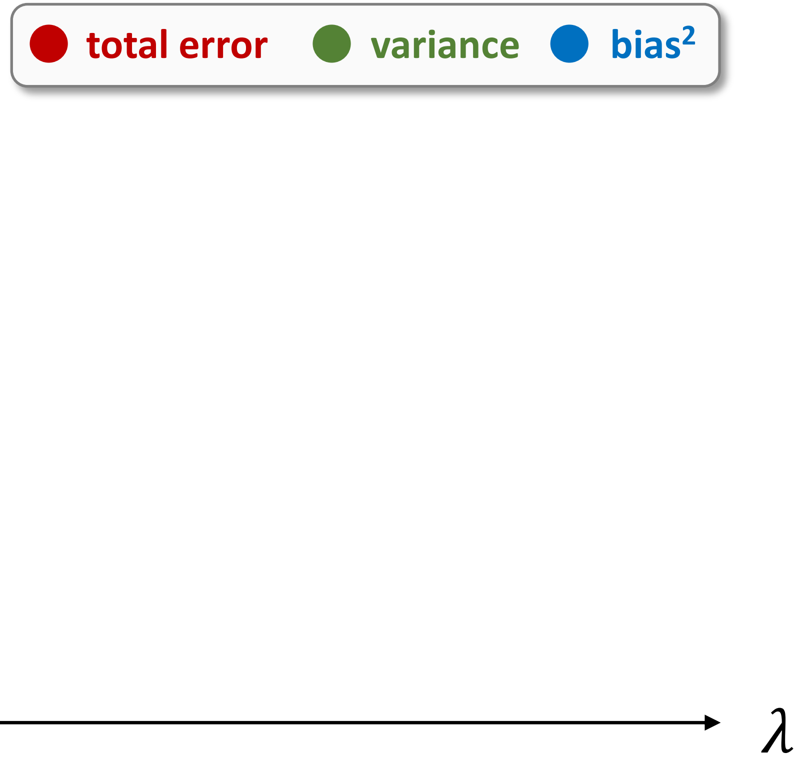


Regularization

SVM revisited

- **Recall**:
  - **Want**: low expected error $L_D(h)$
  - **Have**: low empirical error $L_S(h)$
  - **Care**: generalization $L_D(h_S)$ for $h_S = A(S)$
- **Soft SVM objective**: $\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$
- **Q:** We know SVM is not ERM… so what is it?
- **Loss**: replaced 0/1 with hinge proxy for optimization reasons
- **Norm**: pushes solution $\hat{w}$ away from 0/1-optimal w^*
- Justification for adding $\lambda\|w\|$:
  1. **Modeling**: max margin
  2. **Optimization**: strong convexity
  3. **Stats**: bias-variance! (up next)
- Approach called **Regularized Loss Minimization** (RLM)

SVM as RLM



- **Soft SVM objective:**

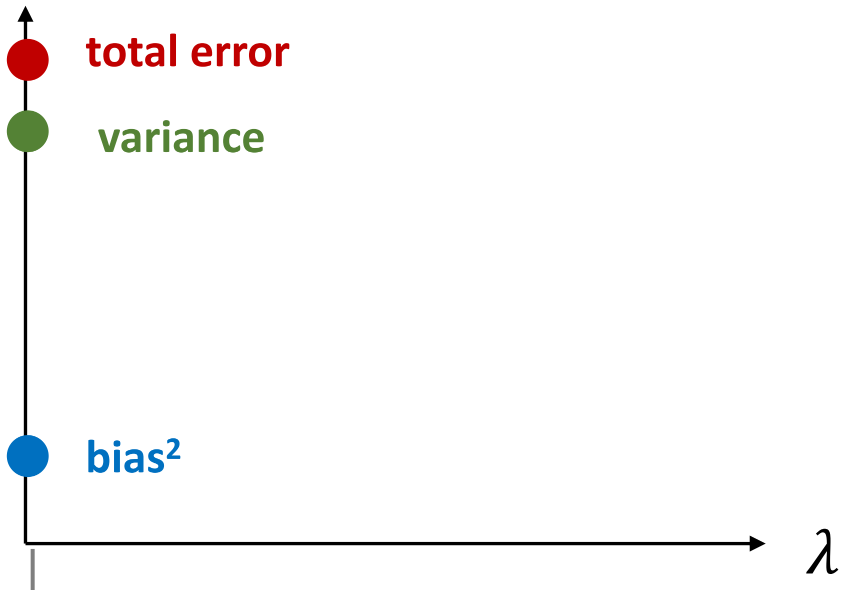
$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$$

SVM as RLM

- **Soft SVM objective:**

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$$

total error variance bias²

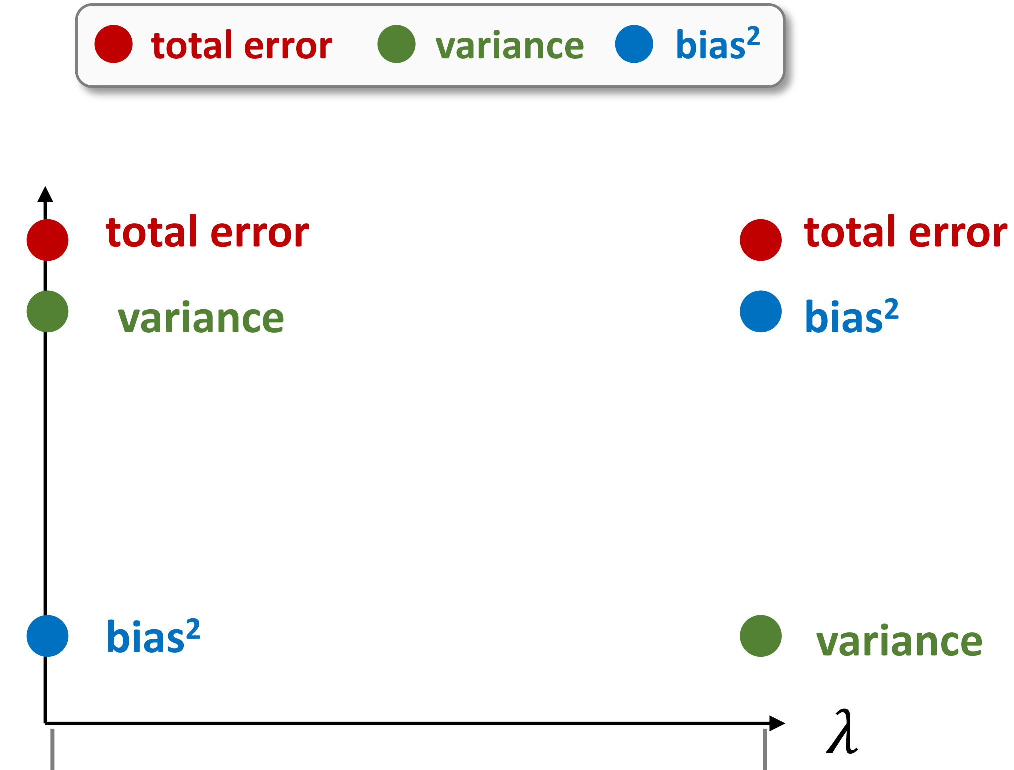


$\lambda = 0$:
- no regularization
- each h_S tailored specifically to S

SVM as RLM

- **Soft SVM objective:**

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$$



$\lambda = 0$:
- no regularization
- each h_S tailored specifically to S

$\lambda \to \infty$:
- only regularization
- always $h_S = 0$
- zero variance
- $\equiv$ random guessing

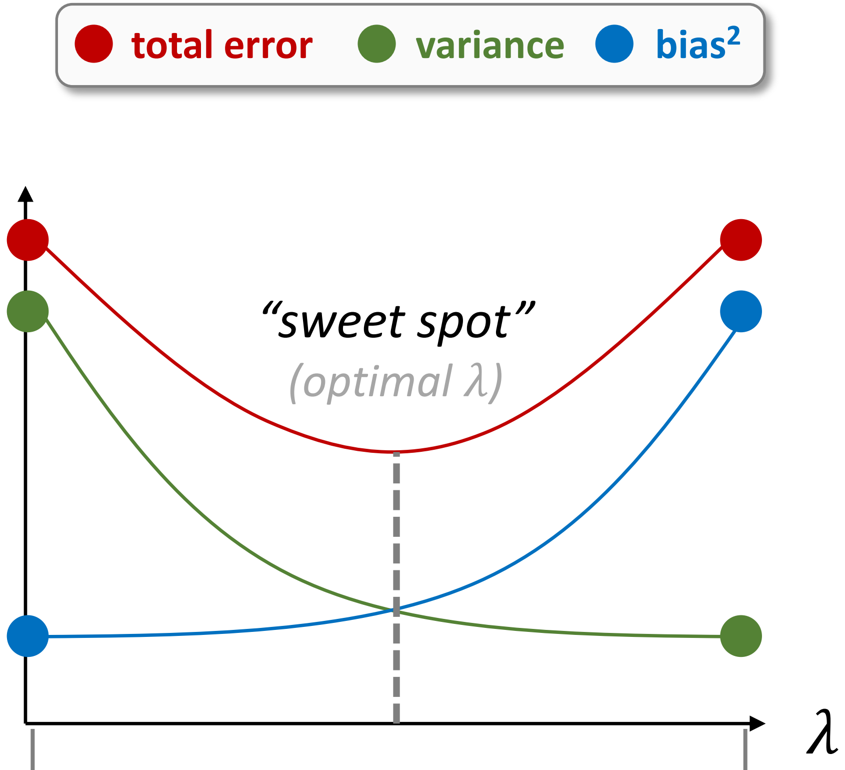
SVM as RLM

- **Soft SVM objective:**

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \max\{0,1-y_i\cdot w^\top x_i\} + \lambda\|w\|_2^2$$

- **This looks like a bias-variance plot!**
  (even though it's hinge and not squared loss)
  (and not by coincidence)

- **Interpretation – decoupled learning objective:**
  - **loss** – directly reduces bias
    (and ensures tractable optimization!)
  - **regularization** – indirectly reduces variance
    (while knowingly incurring of some bias)



$\lambda = 0$:
- no regularization
- each h_S tailored specifically to S

$\lambda \to \infty$:
- only regularization
- always $h_S = 0$
- zero variance
- $\equiv$ random guessing

- In effect, regularization controls ("regulates") complexity. **Let's see how**.

Generalization of SVM

(use $w \in H$ and $h \in H$ interchangeably)

- **Define**: bounded-norm linear models $H_B = \{w \in \mathbb{R}^d : \|w\|_2 \leq B\}$
- **Theorem**: (won't prove) for any m, Soft SVM with $\lambda = \sqrt{2/(B^2 m)}$ satisfies:

$$\mathbb{E}_{S\sim D^m}\left[L_D^{\text{hinge}}(w_S)\right] \leq \min_{w \in H_B} L_D^{\text{hinge}}(w) + B\sqrt{\frac{8}{m}}$$

- **Conclusion**: low-norm models $w \in H_B$ generalize better!

- Can use above to get PAC-style sample complexity: $m_H(\epsilon, \delta) \geq O\left(\frac{B^2}{\epsilon^2 \delta}\right)$
- Compare to VC bound: $m_H(\epsilon, \delta) \geq \tilde{O}\left(\frac{VC + \log 1/\delta}{\epsilon^2}\right)$ (for 0/1 loss)
- **Conclusion**: restricting norm as means to control complexity

A broader modeling perspective

- **Soft SVM**: $\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$

- **General template**: $\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \underbrace{\boxed{\frac{1}{m} \sum_{i=1}^{m} \ell(y_i, w^\top x_i)}}_{\textbf{loss}} + \underbrace{\boxed{\lambda R(w)}}_{\textbf{regularization}}$

- Regularized Loss Minimization (RLM):
  - **Interpretation #1**: knowingly add bias to reduce variance
  - **Interpretation #2**: "of all models with equally low loss, choose the one that has [...]“
  - **Interpretation #3:** means to structurally (and softly) plug in prior knowledge

A broader modeling perspective

$$\underbrace{\boxed{\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, w^\top x_i)}}_{\text{loss}} + \underbrace{\boxed{\lambda R(w)}}_{\text{regularization}}$$

- **Q**: What type of prior knowledge does "small L2-norm" express?
  (don't confuse small scale with small norm! small norm implies "small" in very certain way)
- **A**: for Soft SVM – large margin (but also useful in general!)

- **There are many, many ways to regularize.** For example, other norms.
- [DESMOS]
- **Q**: What type of prior knowledge does the L1-norm express?
- **A**: **Sparsity** – encourages learned w_S to have only few non-zero entries (see tirgul)

https://www.desmos.com/calculator/lgzkjvvac3

Model selection

Model selection

- Assume you decide to learn using Soft SVM:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$$

- **Q**: How should you choose λ?
- **Attempt #1**: optimize loss over train set:

$$\underset{w \in \mathbb{R}^d, \lambda \in \mathbb{R}_+}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \lambda \|w\|_2^2$$

- **Wrong**! Will always give $\lambda = 0$ (think – why?)
- **Attempt #2**: use data to (empirically) estimate $L_D(h_S; \lambda)$

Error estimation

- Assume you have access to an additional sample set $V \sim D^{m_v}$ of size m_v
- Define $L_V(h)$ as the empirical error over V
- **Idea**: use V to estimate expected error of classifier
- **Theorem**: Let $h \in H$. Then for any $\delta \in [0,1]$, with probability $\geq 1 - \delta$, it holds that

$$|L_V(h) - L_D(h)| \leq \sqrt{\frac{\log 2/\delta}{2m_v}}$$

- **Proof**: Apply Hoeffding bound
- **Interpretation**: $L_V(h)$ is very good estimator of $L_D(h)$
- **Notice**: bound is independent of the choice of $H \ni h$
  (**read**: estimation works equally well for any regardless the complexity of h)

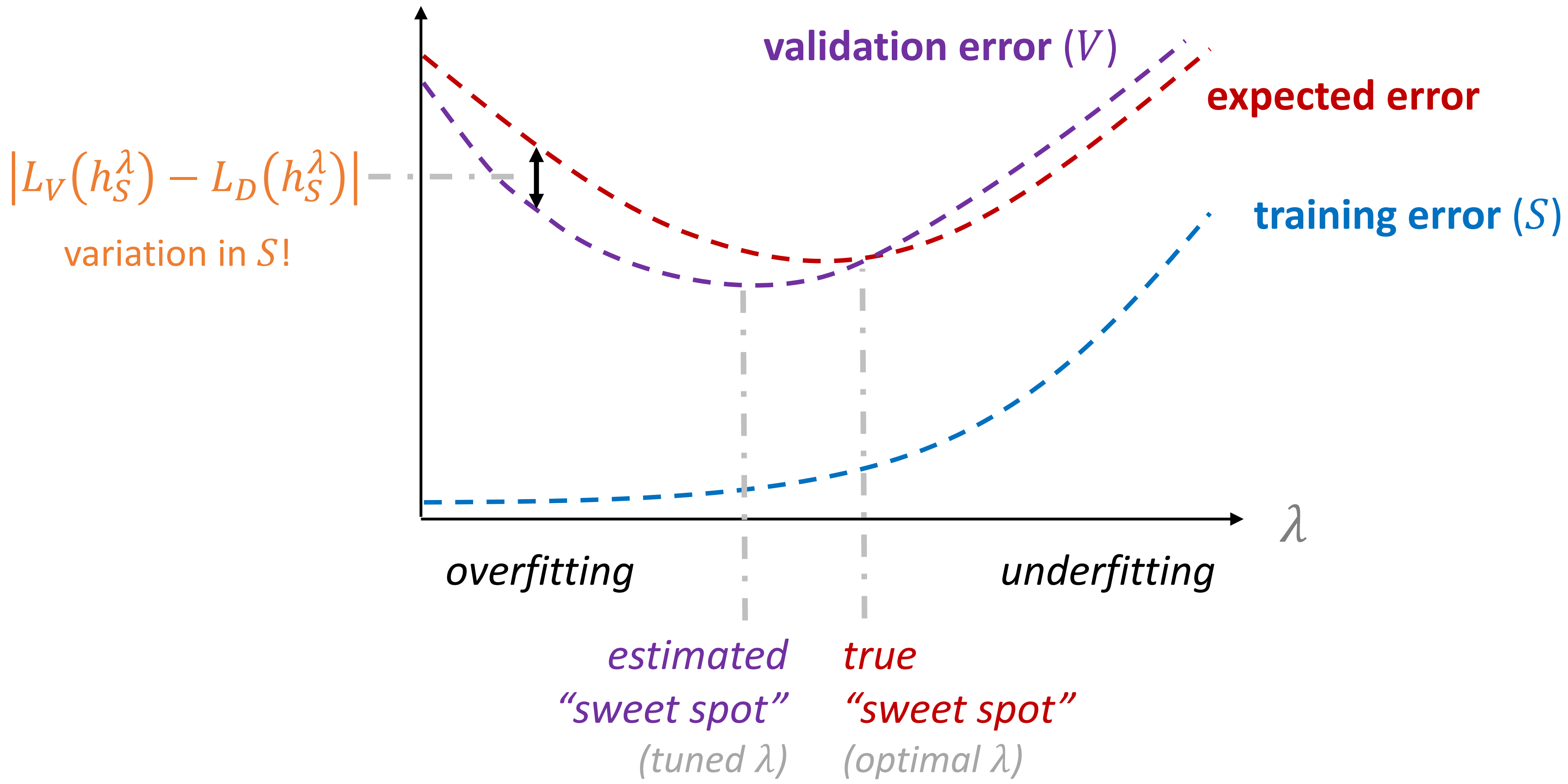
Error estimation

- **Q:** Can L_V help us <u>choose</u> λ when <u>learning</u>? (rather than for a fixed h and given λ)
- **"Tuning"**:
  1. Set range of λ, e.g., $\lambda \in \Lambda = \{2^{-8}, 2^{-7}, \dots, 2^{8}\}$
  2. For each λ, use S to learn best h (using that λ)
  3. Choose best λ using L_V (as estimate of L_D)
- V is called the *validation set* or the *held-out set*
- **Theorem:** Denote h_S^λ the model learned using λ. Let $H_\Lambda = \{h_S^\lambda : \lambda \in \Lambda\} \subset H$. Then for any $\delta \in [0,1]$, with probability $\geq 1 - \delta$, it holds that

$$\forall h \in H_\Lambda, \qquad |L_V(h) - L_D(h)| \leq \sqrt{\frac{\log 2|\Lambda| + \log 1/\delta}{2m_v}}$$

- **A:** Think of H_Λ as a finite class, which we know is learnable – and apply PAC bound
- Works as long as V is sampled <u>independently</u> (of S, H, etc.)

Overfitting, revisited



Cross validation (CV)

- In reality, we usually don’t have an “additional” sample set V
- In practice, must make use of S for both training (h) and tuning (λ)
- **Q**: How can we ensure training and tuning are independent?
- **Common solution**: k-fold cross-validation

Cross validation (CV)

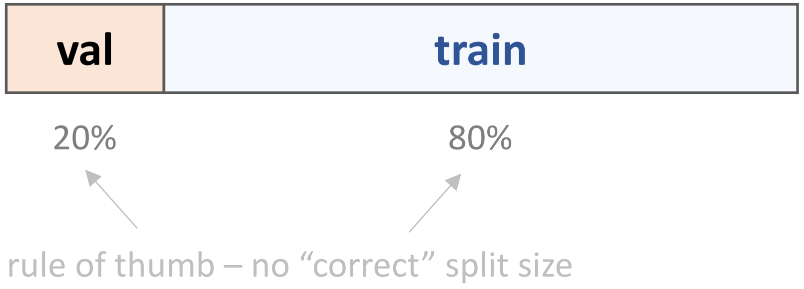
- In reality, we usually don't have an "additional" sample set V
- In practice, must make use of S for both training (h) and tuning (λ)
- **Q**: How can we ensure training and tuning are independent?
- **Common solution**: k-fold cross-validation

sampled data: **train**

Cross validation (CV)

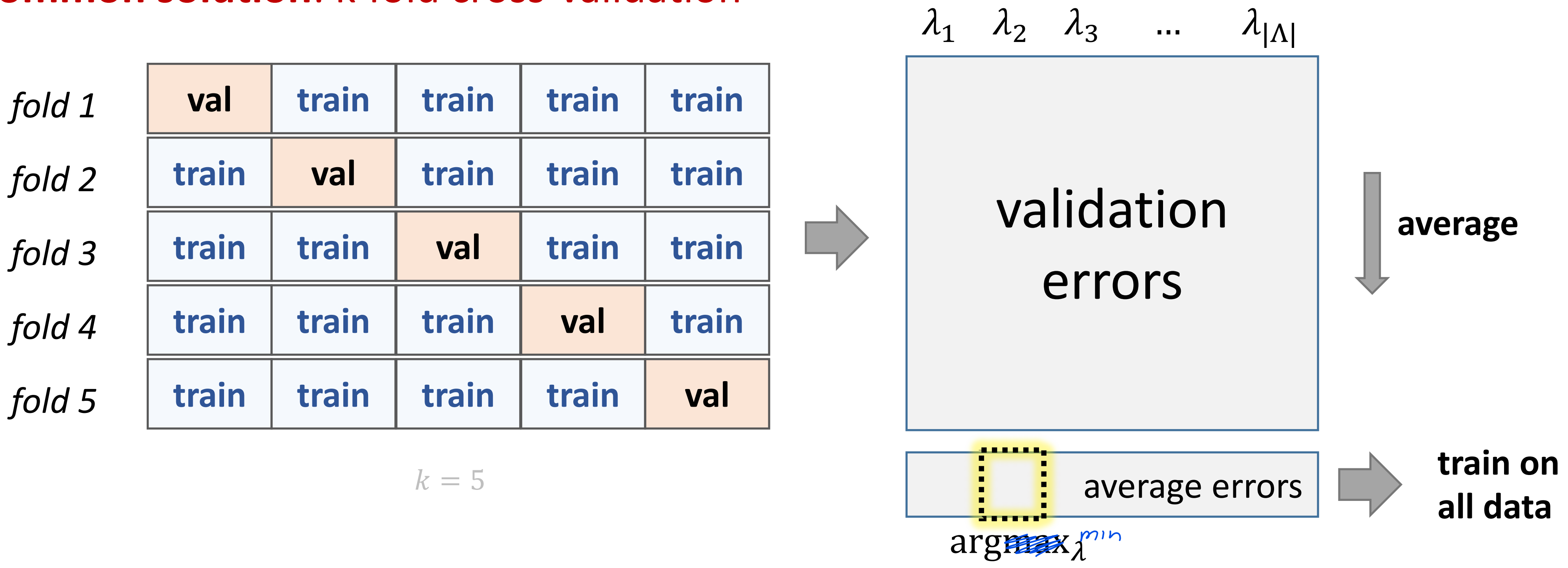
- In reality, we usually don’t have an “additional” sample set V
- In practice, must make use of S for both training (h) <u>and</u> tuning (λ)
- **Q**: How can we ensure training and tuning are independent?
- **Common solution**: k-fold cross-validation

random split:



Cross validation (CV)

- In reality, we usually don't have an "additional" sample set V
- In practice, must make use of S for both training (h) and tuning (λ)
- **Q**: How can we ensure training and tuning are independent?
- **Common solution**: k-fold cross-validation



Discussion

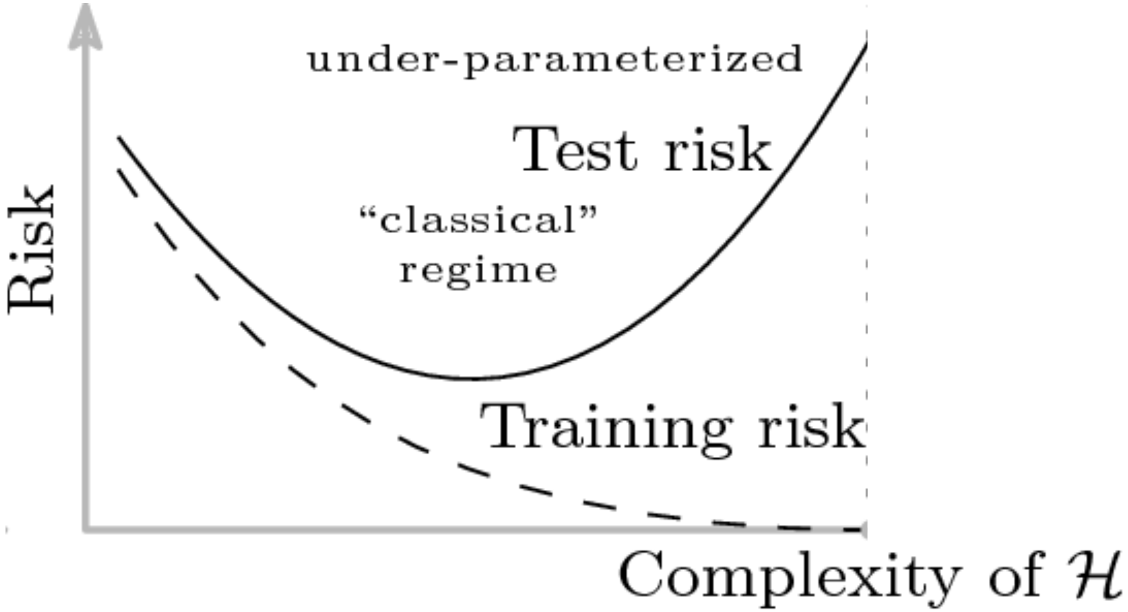
- Three sources of error: **bias**, **variance**, and **noise**
- Smart modeling = carefully control each source
- **Different means to control**:
  - Feature collection/selection
  - Model class complexity (e.g., VC)
  - Hard-code into objective (e.g., RLM)
- **Thought experiment**: what is the difference between varying d and varying $||w||$?
- These inevitably introduce another layer of modeling, requiring either:
  - Prior knowledge ("decision trees should work well for this problem")
  - Tuning by (cross-)validation (use data to determine optimal complexity)

Discussion

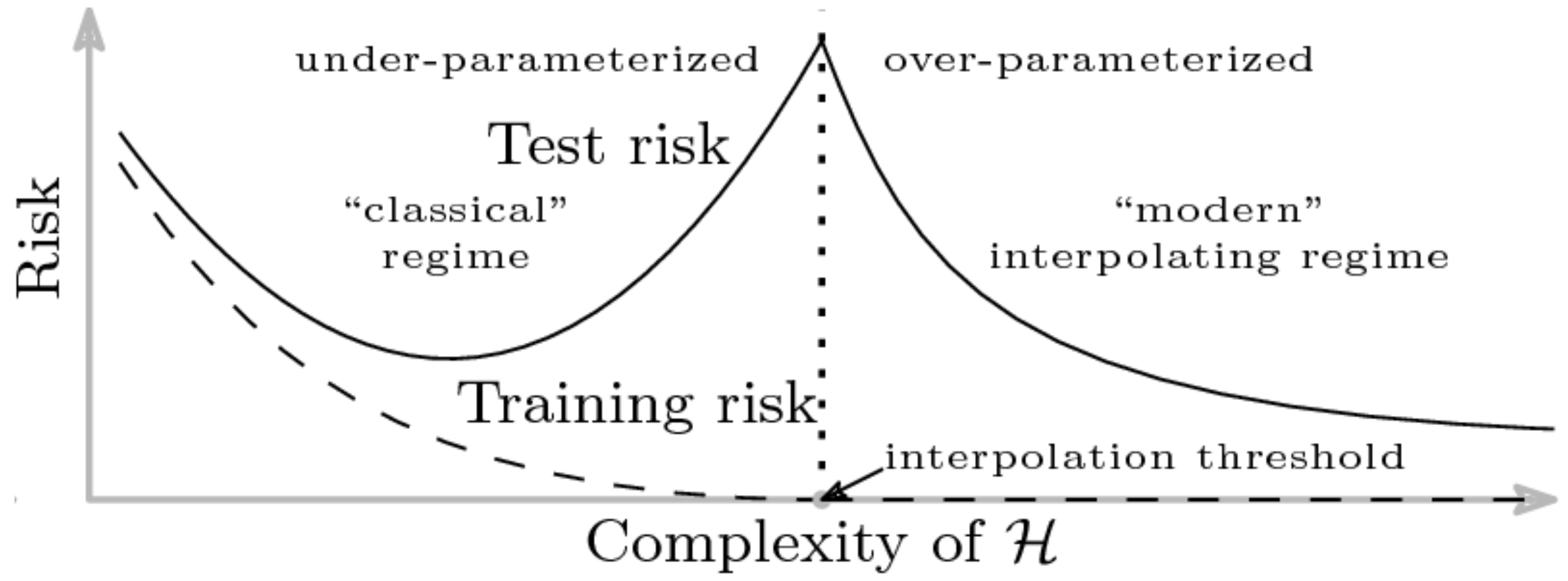
- Many ways to tune hyper-params (e.g., λ): binary search, grid, random
- **Remember**: tuning is costly (in samples, runtime, etc.)
- **Important**: tuning is part of "learning"! (just not by ERM – still uses sample set)
- **Bonus**: neural nets have tons of hyper-params, making tuning critical

- **In practice**: train-val-test split
- Beware leakage! (this is the source of most "innocent" errors)
- Beware "global" overfitting (e.g., repeated usage of same public dataset)

The limits of theory



The limits of theory



Next week(s)

- **Part II**: *the different aspects of learning*
  1. ~~Statistics: generalization and PAC theory~~
  2. ~~Modeling: model selection and evaluation~~
  3. Optimization: convexity, gradient descent
  4. Practical aspects and potential pitfalls