CSE 417T
Introduction to Machine Learning

Lecture 10

Instructor: Chien-Ju (CJ) Ho

Logistics

- Homework 2: due on **Oct 7** (Friday)

- Exam 1: **October 27 (Thursday)**
  - Topics: LFD Chapters 1 to 5
  - Timed exam (75 min) during lecture time (location TBD)
  - Closed-book exam with 2 letter-size cheat sheets allowed (4 pages in total)
    - No format limitations (it can be typed, written, or a combination)

- Homework 3 will be posted some time next week

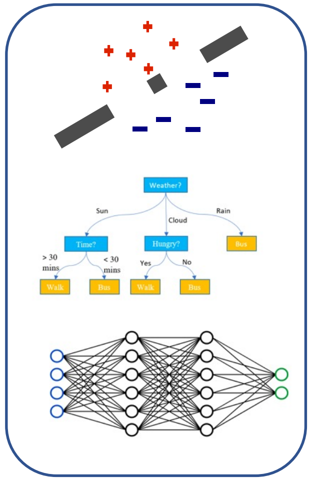
Recap

What We Have Taught So Far

- The explanation of ”machine learning” from the first lecture

Hypothesis / Model

Data (Historical Customer Data)



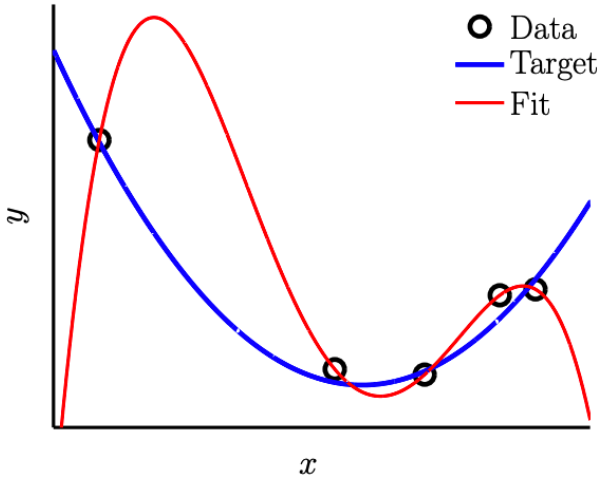
Find a hypothesis that “fits” the data
(The process requires a lot of computation)

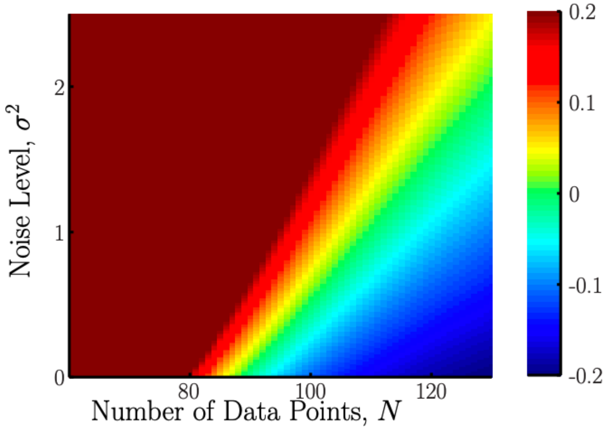
Our progress so far

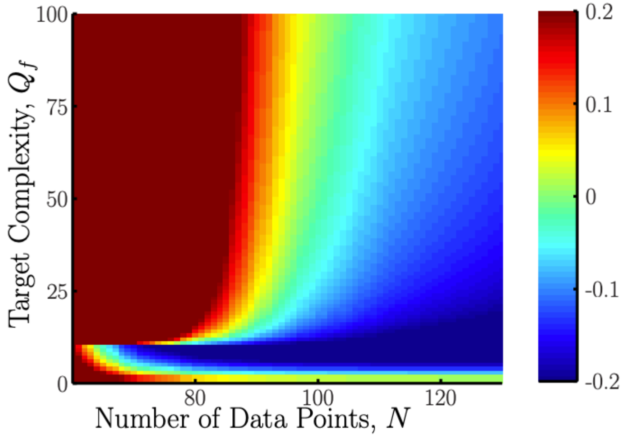
- Generalization of learning
  - What to say about $E_{out}(g)$ from $E_{in}(g)$
- How to find g
  - Using linear models as examples
  - Focus on $g = argmin_{h \in H} E_{in}(g)$

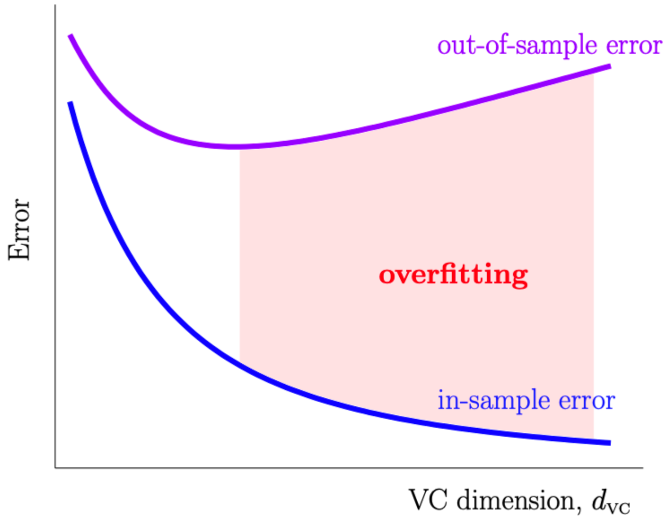
Seems to make sense, but…

Overfitting









Number of data points ↑	Overfitting ↓
Noise ↑	Overfitting ↑
Target complexity ↑	Overfitting ↑

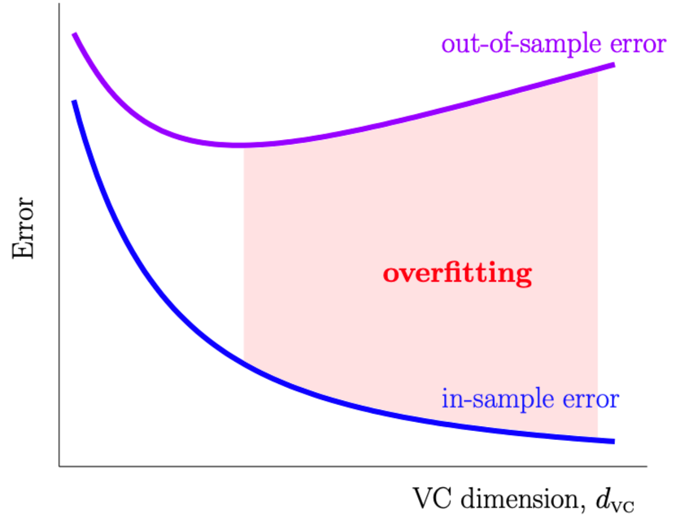
Today’s Lecture

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.

Let me know if you spot errors.

Overfitting and Its Cures

- Overfitting
  - Fitting the data more than is warranted
  - Fitting the noise instead of the pattern of the data
  - Decreasing E_{in} but getting larger E_{out}
  - When H is too strong, but N is not large enough

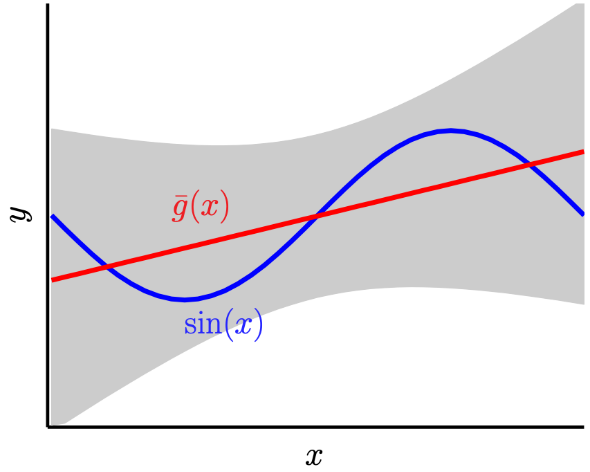


- Regularization
  - Intuition: Constrain H to make overfitting less likely to happen
  - (Topic of this lecture)

- Validation
  - Intuition: Reserve data to estimate E_{out}
  - (Focus of next lecture)

Regularization (Constrain H)

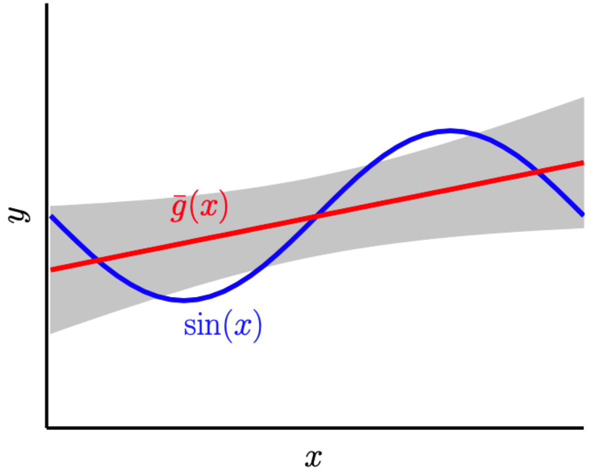
- Informal example:
  - Regression; $f = \sin(\pi x)$; $H = \{h(x) = ax + b\}$; $N = 2$



no regularization

bias $= 0.21$
var $= 1.69$

Regularization:
Constrain the hypothesis set to avoid large a and b



regularization

bias $= 0.23$
var $= 0.33$

How to do this in a principled way?

Hard Constraints

- We have seen hard constraints already

$$H_2 = \{h(x) = w_0 + w_1 x + w_2 x^2\}$$

$$H_{10} = \{h(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_{10} x^{10}\}$$

- H_2 can be written as constrained H_{10}

$$H_2 = \{h \in H_{10} \text{ and } \underbrace{w_3 = w_4 = \cdots = w_{10} = 0}_{\text{Constraints}}\}$$

Soft-Order Constraints

Hard constraints

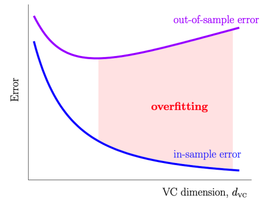
$H_2 = \{h \in H_{10} \text{ and } w_3 = w_4 = \cdots = w_{10} = 0\}$

- Instead of setting the weights to 0

$$H(C) = \left\{h \in H_Q \text{ and } \sum_{q=0}^{Q} w_q^2 \leq C\right\}$$

$$= \left\{h \in H_Q \text{ and } \vec{w}^T\vec{w} \leq C\right\}$$

- Observations
  - When $C \to \infty$, $H(C) = H_Q$
  - When $C_1 \leq C_2$, $H(C_1) \subseteq H(C_2)$ and therefore $d_{vc}\big(H(C_1)\big) \leq d_{vc}(H(C_2))$
  - A smoother way to tune the complexity of hypothesis set



Soft-Order Constraints

$$H(C) = \{h \in H_Q \text{ and } \vec{w}^T \vec{w} \leq C\}$$

- Two main questions
  - How do we choose C
    - Model selection: The same question as selecting H
    - The focus of the next lecture
  - How do we perform learning, i.e., find a $g \in H(C)$ such that $g \approx f$
    - Solve the following constrained optimization problem

minimize $E_{in}(\vec{w})$

subject to $\vec{w}^T \vec{w} \leq C$

Constrained to Unconstrained Optimization

- Constrained optimization ⇔ Unconstrained optimization

minimize $E_{in}(\vec{w})$

subject to $\vec{w}^T\vec{w} \le C$

equivalent

minimize $E_{in}(\vec{w}) + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$

- Why the above is true?
  - Will talk about how to utilize Lagrangian relaxation to get this in the 2nd half of the semester
  - For now, let's think about it graphically

minimize $E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

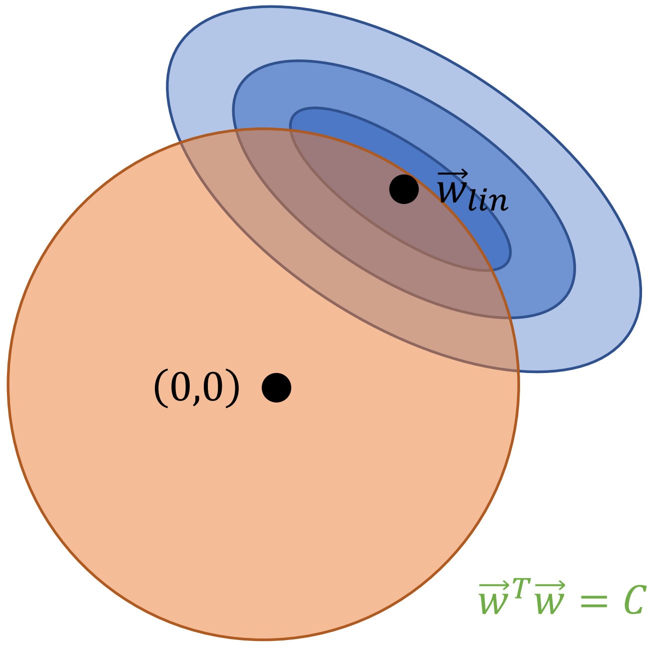
- Notations
  - $\vec{w}_{lin}$: the solution for $\min E_{in}(\vec{w})$
  - $\vec{w}_{reg}$:the solution for $\min E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

minimize $E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

- Notations
  - $\vec{w}_{lin}$: the solution for $\min E_{in}(\vec{w})$
  - $\vec{w}_{reg}$:the solution for $\min E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

- When C is large enough, i.e., $\vec{w}_{lin}^T\vec{w}_{lin} \leq C$
  - $\vec{w}_{reg} = \vec{w}_{lin}$

- When C is not large enough, i.e., $\vec{w}_{lin}^T\vec{w}_{lin} > C$
  - $\vec{w}_{reg}^T\vec{w}_{reg} = C$



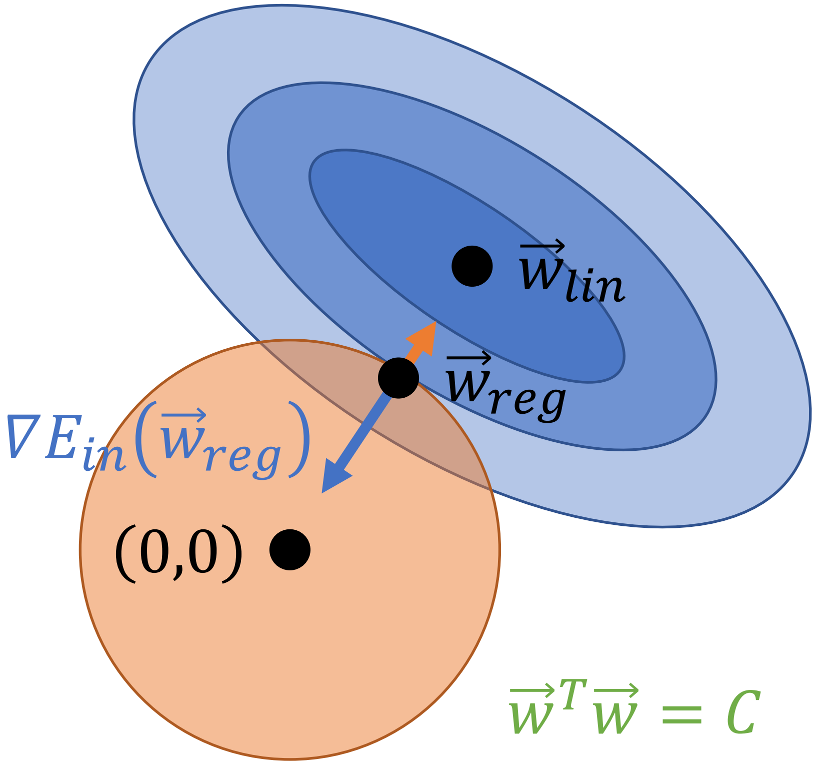
minimize $E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

- When C is not large enough

- $\vec{w}_{lin}$: the solution for $\min E_{in}(\vec{w})$
- $\vec{w}_{reg}$:the solution for $\min E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

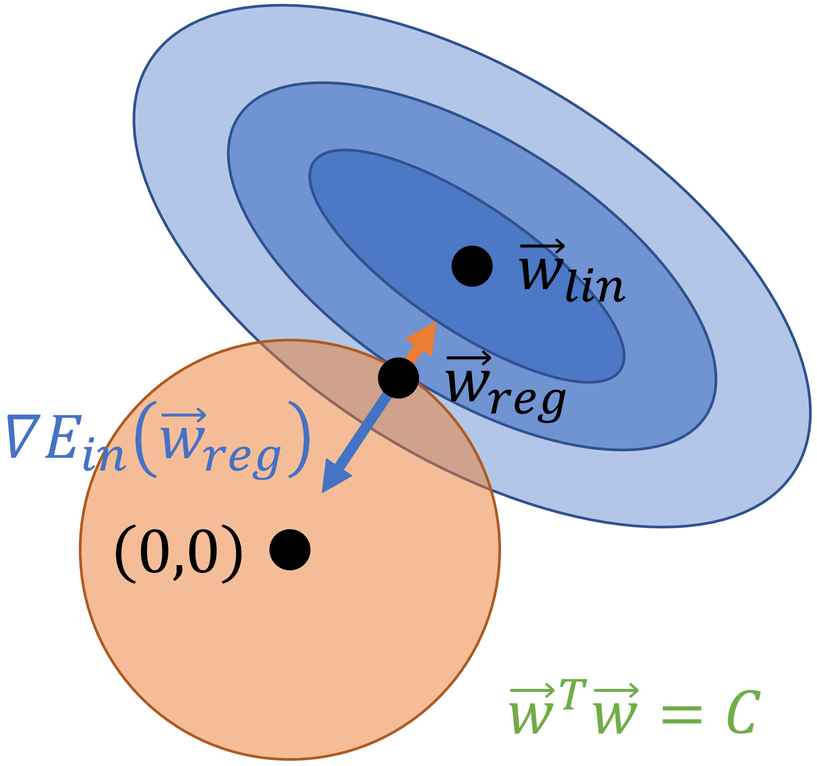
minimize $E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

- When C is not large enough
  - Using graphical arguments
    - $\vec{w}_{reg} \propto -\nabla_{\vec{w}} E_{in}(\vec{w}_{reg})$



minimize $E_{in}(\vec{w})$ subject to $\vec{w}^T\vec{w} \leq C$

- When C is not large enough
  - Using graphical arguments
    - $\vec{w}_{reg} \propto -\nabla_{\vec{w}} E_{in}(\vec{w}_{reg})$
  - That is, we can find some constant $\lambda_C \geq 0$ such that
    - $\nabla_{\vec{w}} E_{in}(\vec{w}_{reg}) = -\frac{2\lambda_C}{N}\vec{w}_{reg}$
  - Therefore,
    - $\nabla_{\vec{w}}\left(E_{in}(\vec{w}_{reg}) + \frac{\lambda_C}{N}\vec{w}_{reg}^T\vec{w}_{reg}\right) = 0$
  - This implies, $\vec{w}_{reg}$ is the solution for
    - minimize $E_{in}(\vec{w}) + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$



Constrained to Unconstrained Optimization

- Constrained optimization ⇔ Unconstrained optimization

minimize $E_{in}(\vec{w})$

subject to $\vec{w}^T\vec{w} \leq C$

equivalent

$\longleftrightarrow$

minimize $E_{in}(\vec{w}) + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$

augmented error

- Interpretations of regularization
  - Constraining H (by adding constraints)
  - Adding penalty to complex hypothesis in augmented errors

Augmented Error

$\vec{w}^T\vec{w}$: weight decay

minimize $E_{in}(\vec{w})$

subject to $\vec{w}^T\vec{w} \leq C$

minimize $E_{in}(\vec{w}) + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$

- Define augmented error
  - $E_{aug}(\vec{w}) = E_{in}(\vec{w}) + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$
  - Algorithm: Find $\vec{w}^* = argmin\ E_{aug}(\vec{w})$

- A bit more discussion
  - When $C \to \infty$, $\lambda_C = 0$
  - Smaller C (stronger constraints)
    - => larger λ_C
    - => smaller H
    - => stronger regularization
  - Use λ_C to tune the level of regularization

Side note:
You will see people/us interchangeably use λ_C and $\frac{\lambda_C}{N}$ to be the constant, depending on whether the dependency on N is emphasized.

General Form of Regularization

$$E_{aug}(h, \lambda, \Omega) = E_{in}(\vec{w}) + \frac{\lambda}{N}\Omega(h)$$

- Key parameters
  - Ω: Regularizer
  - λ: Amount of regularization

- Does the form look familiar: VC Theory
  - $E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc}\frac{\ln N}{N}}\right)$

- If we pick the right Ω, E_{aug} can be a better proxy for E_{out}

How to Pick the Right Ω

- No definite answer, but generally
  - We like to pick Ω that leads to “smoother” hypothesis
    - Overfitting is due to noise
    - Informally, noise is usually ”high frequency”
  - We prefer Ω that makes the optimization easier (e.g., convex/differentiable)
    - Similar to picking the error measure
  - We might have some other objective in mind
    - Ex: L-1 regularizer leads to weight vectors with more 0s
      - $E_{aug}(\vec{w}) = E_{in}(\vec{w}) + \lambda\|\vec{w}\|_1 = E_{in}(\vec{w}) + \lambda \sum_i |w_i|$
- What if we pick the wrong Ω (Think about weight growth)
  - We might still fix it by picking the right λ – using validation in the next lecture

More Discussion on Regularization

Why $\vec{w}^T\vec{w}$ is Called Weight Decay

- Run gradient descent on $E_{aug}(\vec{w}) = E_{in}(\vec{w}) + \lambda_C \vec{w}^T\vec{w}$

- The update rule would be

$$\vec{w}(t+1) \leftarrow \vec{w}(t) - \eta \nabla_{\vec{w}} E_{aug}(\vec{w}(t))$$

$$\Rightarrow \vec{w}(t+1) \leftarrow (1 - 2\eta\lambda_C)\vec{w}(t) - \eta \nabla_{\vec{w}} E_{in}(\vec{w}(t))$$

Proving this is in your HW3.

We are decaying the weights first, then do the update

Linear Regression with Weight Decay

- $E_{aug}(\vec{w}) = E_{in}(w) + \frac{\lambda_C}{N}\vec{w}^T\vec{w} = \frac{1}{N}\|X\vec{w} - \vec{y}\|^2 + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$

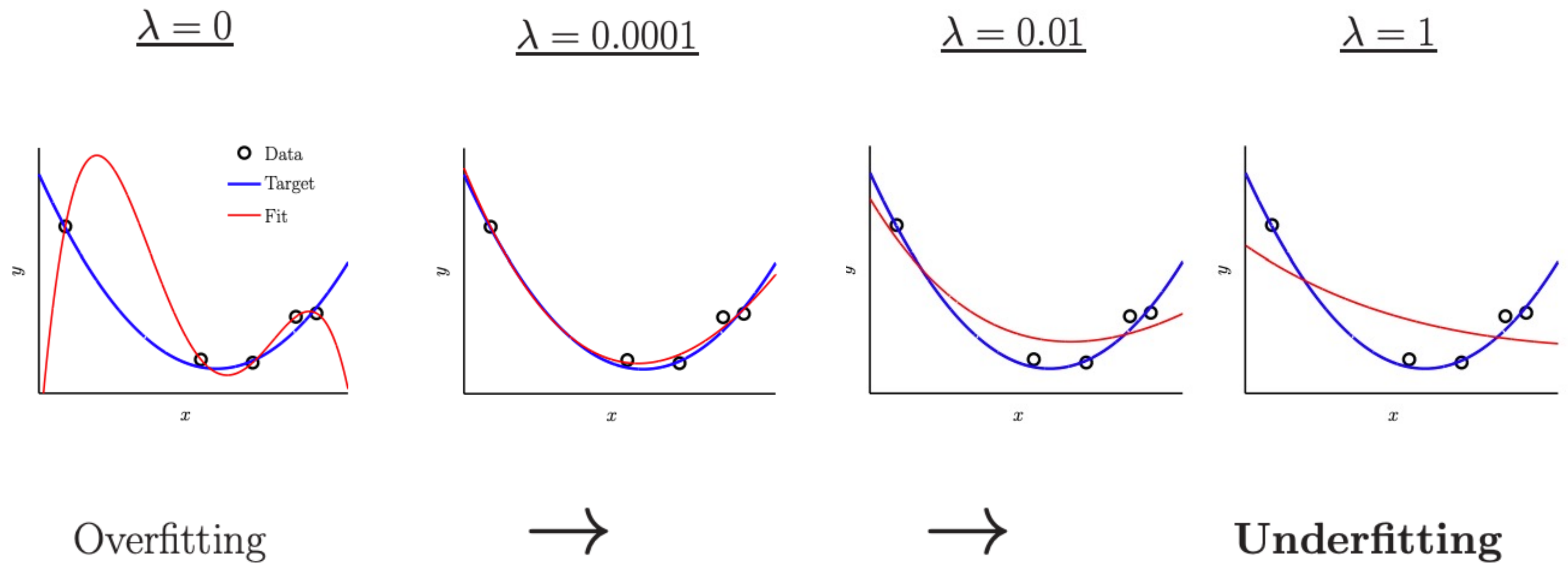
- Solve $\nabla_{\vec{w}} E_{aug}(\vec{w})|_{\vec{w}=\vec{w}_{reg}} = 0$, we get
  - $\frac{2}{N}\left(X^T X\vec{w}_{reg} - X^T\vec{y} + \lambda_C\vec{w}_{reg}\right) = 0$
  - $(X^T X + \lambda_C I)\vec{w}_{reg} = X^T\vec{y}$
  - $\vec{w}_{reg} = (X^T X + \lambda_C I)^{-1} X^T\vec{y}$

Notation: I is an identity matrix: only the elements in the diagonals are 1, and all others are 0.

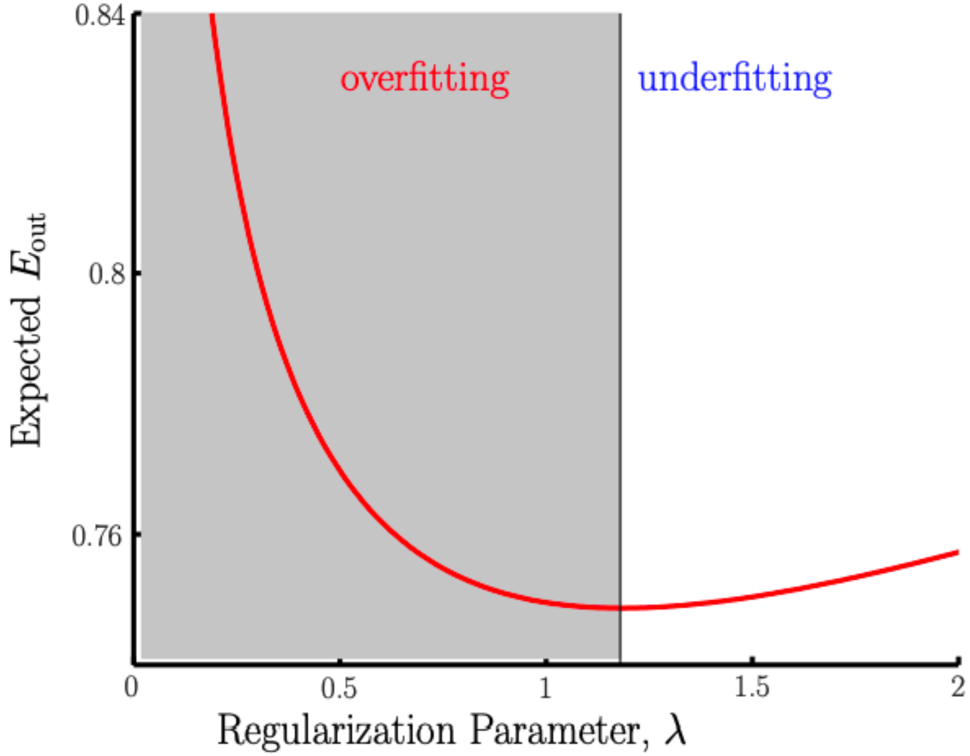
This is called “Ridge Regression” in statistics.

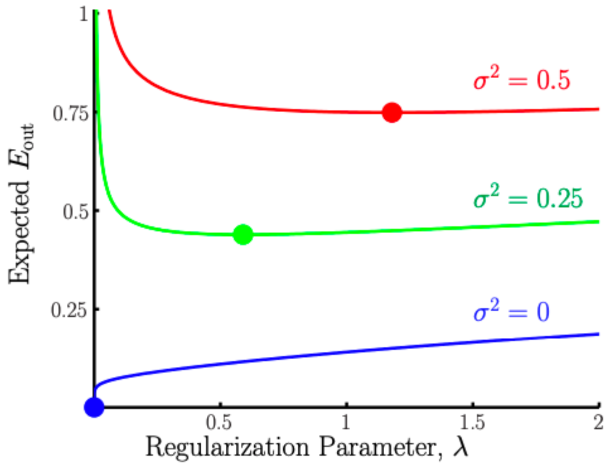
Effect of Regularization (Different λ)

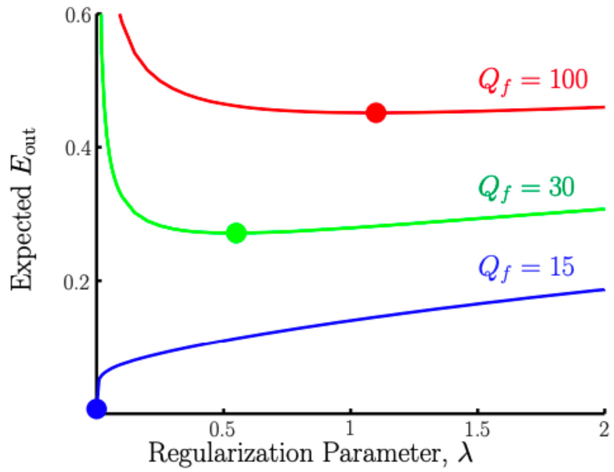
- Minimizing $E_{in}(\vec{w}) + \frac{\lambda}{N}\vec{w}^T\vec{w}$ with different λ



Overfitting and Underfitting



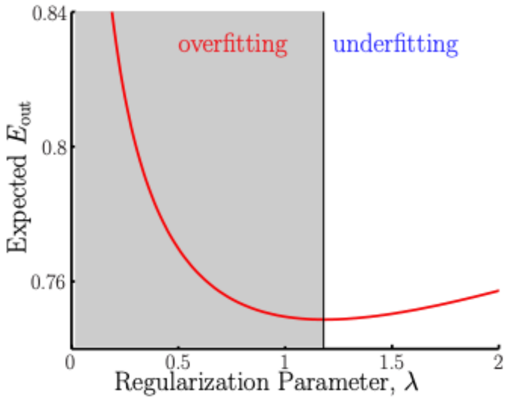




Need to pick the right λ:
Using validation: Focus of next lecture

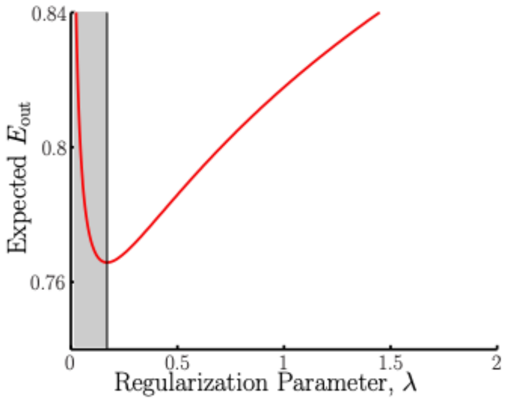
Variations on Weight Decay (Different Ω)

Uniform Weight Decay



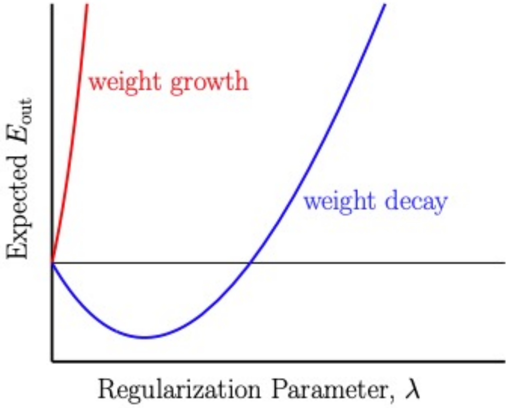
$$\sum_{q=0}^{Q} w_q^2$$

Low Order Fit



$$\sum_{q=0}^{Q} q w_q^2$$

Weight Growth!



$$\sum_{q=0}^{Q} \frac{1}{w_q^2}$$

How to Pick the Right Ω

- As discussed earlier
  - Intuition: pick Ω that leads to “smoother” hypothesis
    - Overfitting is due to noise
    - Informally, noise is generally ”high frequency”
  - Computation: prefer Ω that makes the optimization easier (e.g., convex/differentiable)
    - Similar to picking the error measure
  - We might have some other objective in mind
    - Ex: L-1 regularizer leads to weight vectors with more 0s

- What if we pick the wrong Ω (weight growth)
  - We might still fix it by picking the right λ – using validation

Summarizing Regularization

- Regularization is everywhere in machine learning

- Two main ways of thinking about regularization
  - Constrain H to make overfitting less likely to happen
    - Will discuss more regularization methods in the 2nd half of the semester
    - Pruning for decision trees, early stopping / dropout for neural networks, etc

  - Define augmented error E_{aug} to better approximate E_{out}
    - $E_{aug}(h, \lambda, \Omega) = E_{in}(\vec{w}) + \frac{\lambda}{N}\Omega(h)$

- We show the equivalence of the two for weight decay
  - The conceptual equivalence is general with Lagrangian relaxation (will cover later in the semester)

Validation

Prevent Overfitting

$$E_{out}(g) = E_{in}(g) + \text{overfit penalty}$$

- Regularization
  - Choose a regularizer Ω to approximate the penalty
- Validation
  - Directly estimate E_{out} (The real goal of learning is to minimize E_{out})

Review: Test Set

- Out-of-sample error $E_{out}(g) = \mathbb{E}_{\vec{x}}[e(g(\vec{x}), y)]$
  - Key: $\vec{x}$ is out of sample

- Test set $D_{test} = \{(\vec{x}_1, y_1), \dots, (\vec{x}_K, y_K)\}$
  - Reserve K data points used to estimate E_{out}
  - None of the data points in test set can be involved in training

- Using the data in test set to estimate E_{out}
  - Since all data points in D_{test} are out of sample

Test Set

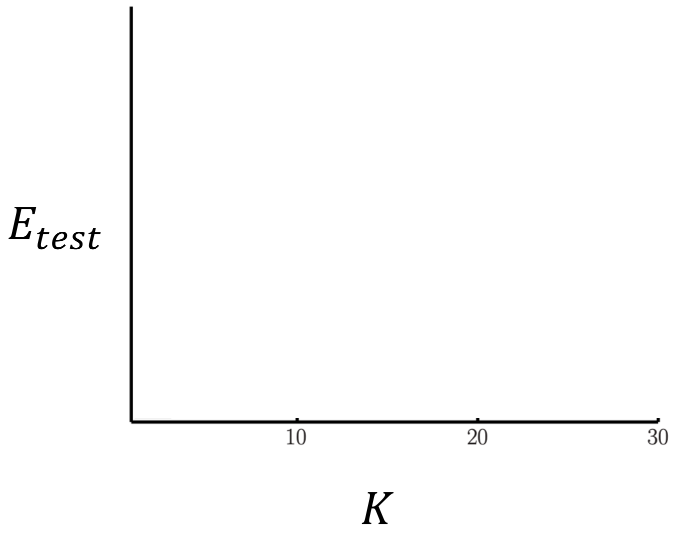
- Test set $D_{test} = \{(\vec{x}_1, y_1), \dots, (\vec{x}_K, y_K)\}$
- For a g learned using only training set

- Let $E_{test}(g) = \frac{1}{K}\sum_{k=1}^{K} e(g(\vec{x}_k), y_k)$
  - $E_{test}(g)$ is an unbiased estimate of $E_{out}(g)$
    - $\mathbb{E}[E_{test}(g)] = \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}[e(g(\vec{x}_k), y_k)] = E_{out}(g)$
  - Single hypothesis Hoeffding bound applies
    - $E_{out}(g) \leq E_{test}(g) + O\left(\sqrt{\frac{1}{K}}\right)$

Where are Test Set From?

- Given a data set D of N points
  - $D = D_{train} \cup D_{test}$
  - Reserving K points for test set means we only have $N - K$ points for training

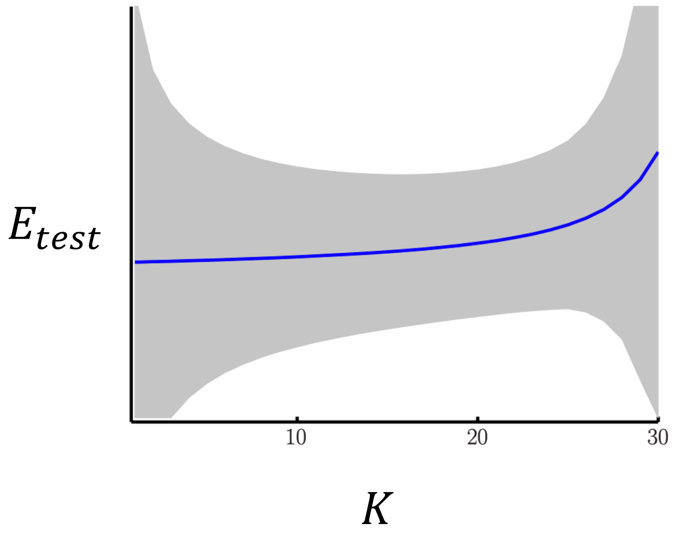
- Effect of the choice of K



Where are Test Set From?

- Given a data set D of N points
  - $D = D_{train} \cup D_{test}$
  - Reserving K points for test set means we only have $N - K$ points for training

- Effect of the choice of K



Rule of Thumb: $K^* = \frac{N}{5}$

Utilizing the Whole D

- Process:
  - $D = D_{train} \cup D_{test}$ where $|D_{test}| = K, |D_{train}| = N - K$
  - Learn some hypothesis g^- using only D_{train}
  - Estimate $E_{out}(g^-)$ using D_{test}
  - Let g be the hypothesis that would be learned using D
- Generally (informally, not theoretically proven)
  - Training on more data leads to better learned hypothesis
  - $E_{out}(g) \leq E_{out}(g^-)$

Validation: Beyond Test Set

- What if we want to estimate E_{out} multiple times?

- Model selection:
  - Should I use linear models or decision trees?
  - Should I set the regularization parameter λ to 0.1, 0.01, or 0.001?
    - A model with different λ can be considered as different model

- Validation set
  - $D = D_{train} \cup D_{val}$

  - Key difference: We need to account for the multiple usages of D_{val}

VC Dimension of d-dim Perceptron

Recall the Definitions

- Shatter
  - H **shatters** $(\vec{x}_1, \ldots, \vec{x}_N)$ if $|H(\vec{x}_1, \ldots, \vec{x}_N)| = 2^N$
  - H can induce all label combinations for $(\vec{x}_1, \ldots, \vec{x}_N)$
- Break point
  - k is a **break point** for H if no data set of size k can be shattered by H
  - k is a break point for $H \leftrightarrow m_H(k) < 2^k$
- VC Dimension: $d_{vc}(H)$ or d_{vc}
  - The VC dimension of H is the largest N such that $m_H(N) = 2^N$
  - Equivalently, if k^* is the smallest break point for H, $d_{vc}(H) = k^* - 1$

VC Dimension of d-dimension Perceptron

- Claim:
  - The VC Dimension of d-dim perceptron is $d + 1$

- How to prove it?
  1. Show that the VC dimension of d-dim perceptron $\geq d + 1$
  2. Show that the VC dimension of d-dim perceptron $\leq d + 1$

- To prove $d_{vc}(H) \geq d+1$, what do we need to prove?

    A. There is a set of $d+1$ points that can be shattered by H
    B. There is a set of $d+1$ points that cannot be shattered by H
    C. Every set of $d+1$ points can be shattered by H
    D. Every set of $d+1$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
  A. There is a set of $d + 1$ points that can be shattered by H
  B. There is a set of $d + 1$ points that cannot be shattered by H
  C. Every set of $d + 1$ points can be shattered by H
  D. Every set of $d + 1$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?

  A. There is a set of $d + 1$ points that can be shattered by H
  B. There is a set of $d + 1$ points that cannot be shattered by H
  C. Every set of $d + 1$ points can be shattered by H
  D. Every set of $d + 1$ points cannot be shattered by H

- To prove $d_{vc}(H) \leq d + 1$, what do we need to prove?

  A. There is a set of $d + 1$ points that can be shattered by H
  B. There is a set of $d + 2$ points that cannot be shattered by H
  C. Every set of $d + 2$ points can be shattered by H
  D. Every set of $d + 1$ points cannot be shattered by H
  E. Every set of $d + 2$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d+1$, what do we need to prove?

  A. There is a set of $d+1$ points that can be shattered by H
  B. There is a set of $d+1$ points that cannot be shattered by H
  C. Every set of $d+1$ points can be shattered by H
  D. Every set of $d+1$ points cannot be shattered by H

- To prove $d_{vc}(H) \leq d+1$, what do we need to prove?

  A. There is a set of $d+1$ points that can be shattered by H
  B. There is a set of $d+2$ points that cannot be shattered by H
  C. Every set of $d+2$ points can be shattered by H
  D. Every set of $d+1$ points cannot be shattered by H
  E. Every set of $d+2$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
  There is a set of $d + 1$ points that can be shattered by H

- To prove $d_{vc}(H) \leq d + 1$, what do we need to prove?
  Every set of $d + 2$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d+1$, what do we need to prove?
  There is a set of $d+1$ points that can be shattered by H

Proof Sketch:

1. Let's construct a dataset of $d+1$ points: $X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_{d+1}^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 1 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 0 & \dots & 0 & 0 \end{bmatrix}$; It's easy to check that X^{-1} exist

2. For any possible dichotomy $\vec{y}$, there exists a $\vec{w}$ such that $X\vec{w} = \vec{y}$, i.e., $\vec{w} = X^{-1}\vec{y}$
3. Therefore, d-dim perceptron can shatter X

- To prove $d_{vc}(H) \leq d+1$, what do we need to prove?
  Every set of $d+2$ points cannot be shattered by H

Proof Sketch:

1. For every set of $d+2$ points (in d+1 dimensions), there exists a point that can be written as linear combinations of the others.
2. Denote the point $\vec{x}_{d+2}$, we have $\vec{x}_{d+2} = \sum_{i=1}^{d+1} a_i \vec{x}_i$
3. Consider the dichotomy $(y_1, \dots, y_{d+2}) = (\mathrm{sign}(a_1), \dots, \mathrm{sign}(a_{d+1}), -1)$, we can show that no linear separator can generate this dichotomy (think about why).
4. Therefore, for every set of $d+2$ points, there exist at least one dichotomy that H cannot induce.