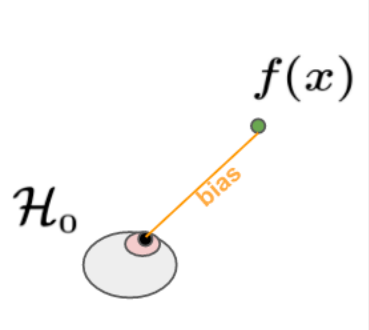
Regularization

Bias Variance Trade off

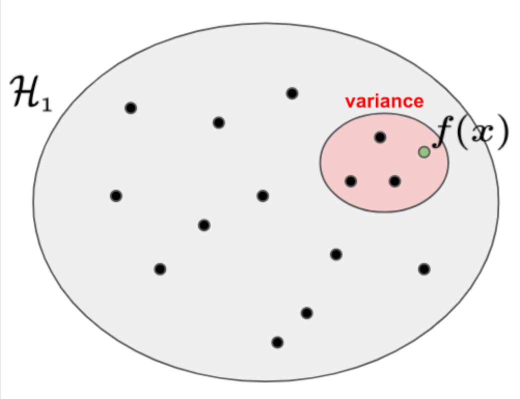
$$E_{X\in D_{test}}(squared\ loss) = E_X\left\{\underbrace{\left[E_D\left(f(X) - \hat{f}(X)\right)\right]^2}_{\text{Bias of } \hat{f}} + \underbrace{Var_D\left[\hat{f}(X)\right]}_{\text{Variance of of } \hat{f}} + \underbrace{Var_{\varepsilon}(\varepsilon)}_{\text{Irreducible error}}\right\}$$



Smaller representational capacity

Bias down, variance up

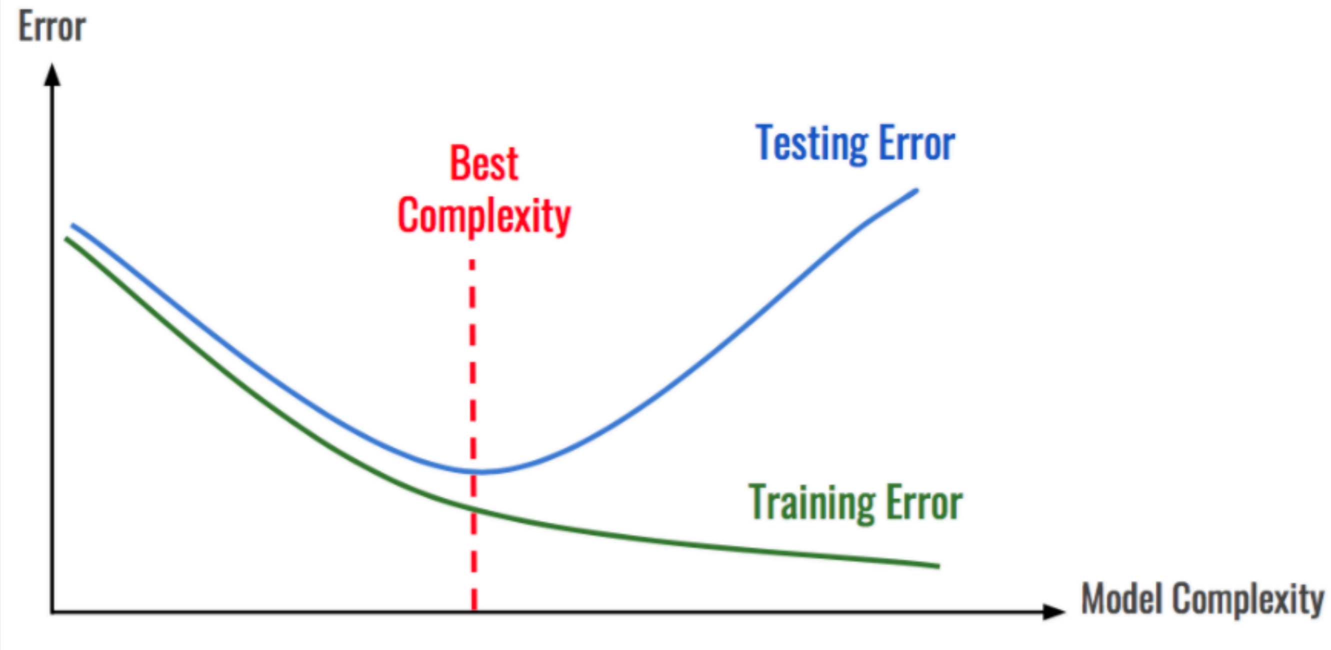
Bias up, variance down



Agenda

- Taming the sine wave using regularization
- Ridge and lasso regularization
- Examples

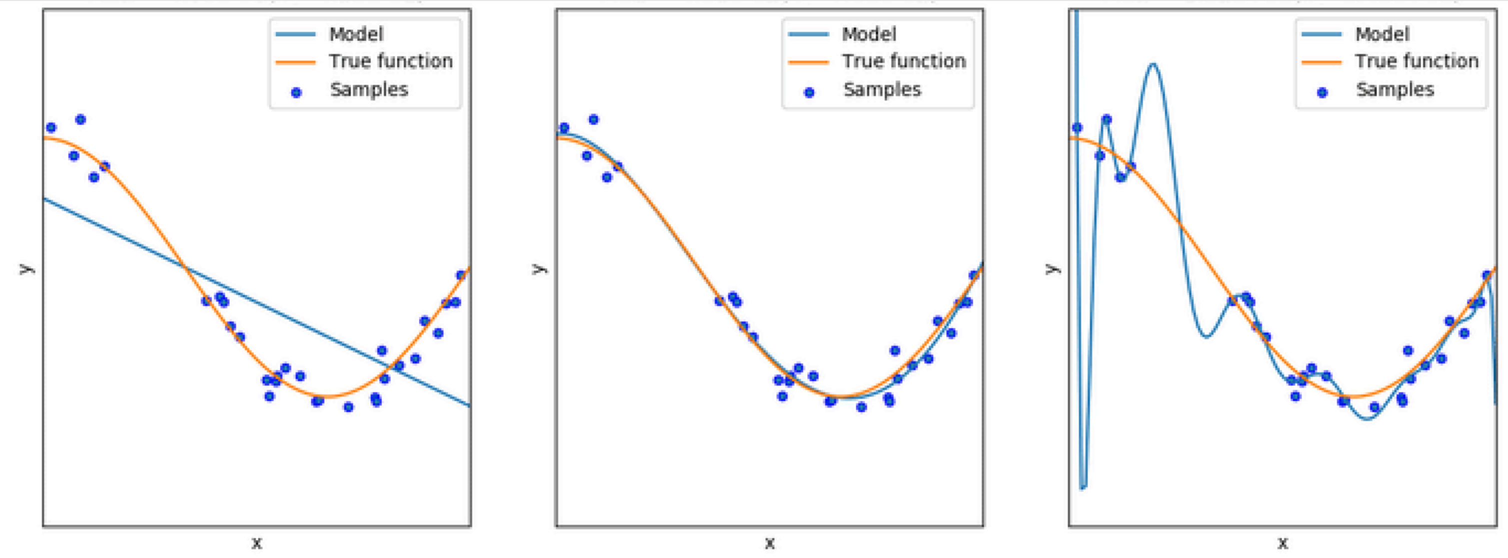
Overfitting



In linear regression model, as we increase the number of predictors, the training error will always decrease

Overfitting

Polynomial regression



Overfit: fitting the data more than is warranted

Polynomial Regression

SEARCH SPACE

$$\mathcal{F}_1 : f(x) = w_0 + w_1 x$$

$$\mathcal{F}_2 : f(x) = w_0 + w_1 x + w_2 x^2$$

$$\mathcal{F}_3 : f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

…

$$\mathcal{F}_K : f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$$

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_K$$

Regularized Regression

SEARCH SPACE

$$\mathcal{F}_1 = \{w \to w \cdot x \mid \|w\|_2 \leq W\}$$

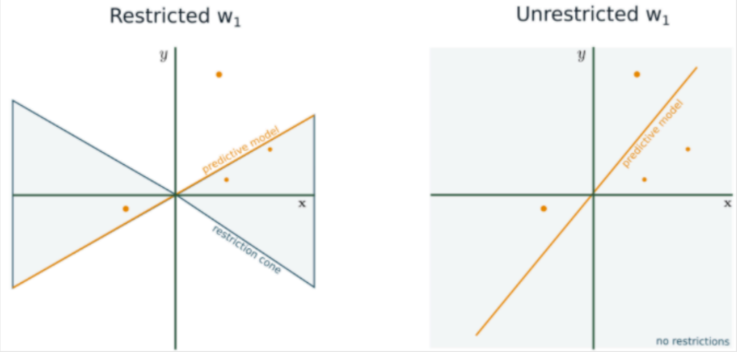
$$\mathcal{F}_2 = \{w \to w \cdot x \mid \|w\|_2 \leq 2W\}$$

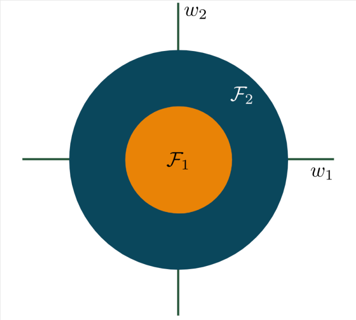
$$\mathcal{F}_3 = \{w \to w \cdot x \mid \|w\|_2 \leq 3W\}$$

…

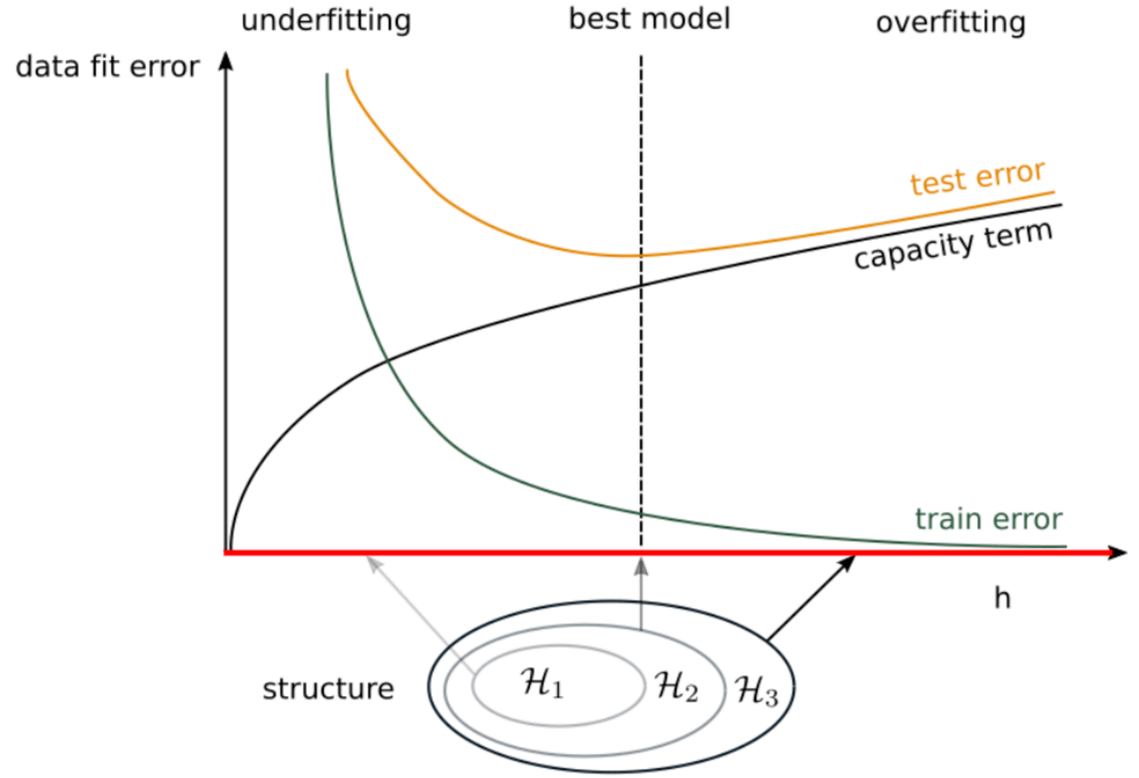
$$\mathcal{F}_\infty = \{w \to w \cdot x \mid \|w\|_2 \leq \infty\}$$

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_K$$



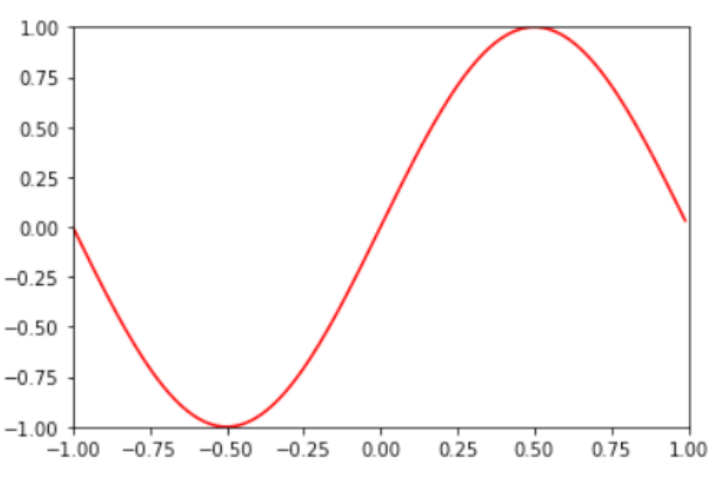


Complexity Search Space



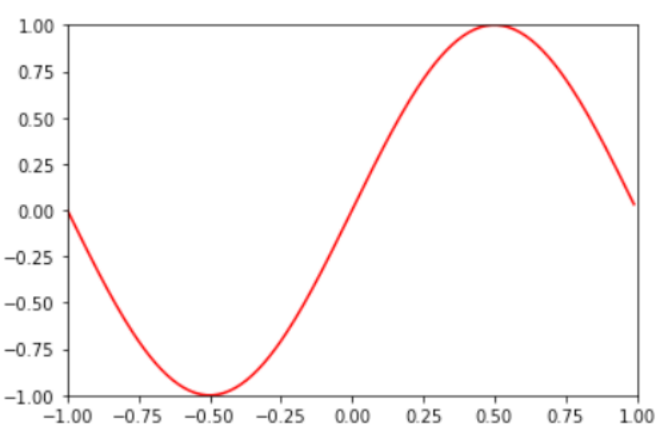
Approximating a sine wave

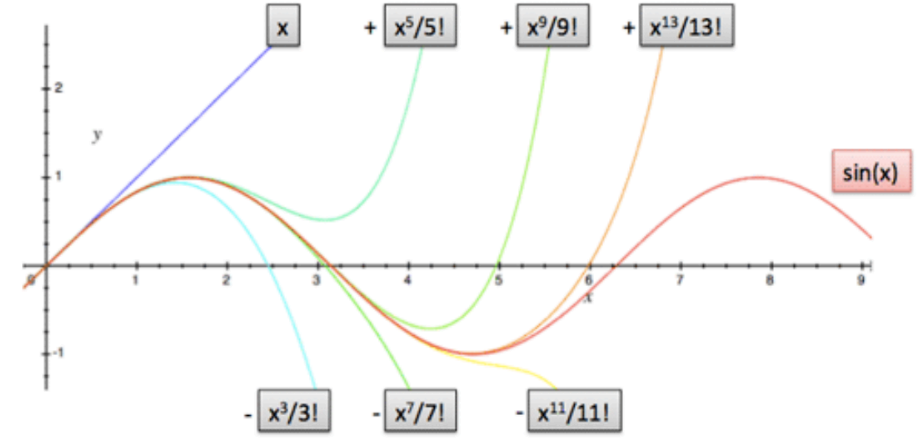
True function $f(x) = \sin(\pi x)$. We have $f: [-1, 1] \to R$



Idea: Use polynomial regression to approximate the sine wave

Let's look at the Taylor expansion





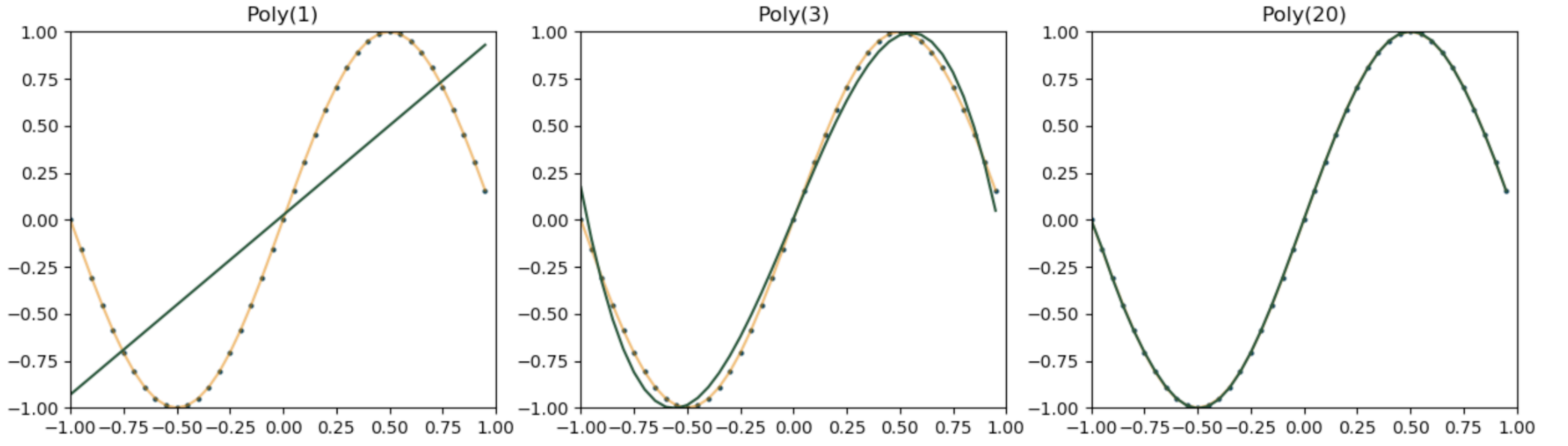
$$f(x) = \sin(\pi x)$$

$$\sin(\pi x) \approx \pi x - \frac{\pi^3}{3!}x^3 + \frac{\pi^5}{5!}x^5 - \frac{\pi^7}{7!}x^7 + \frac{\pi^9}{9!}x^9 - \cdots$$

Higher Degree is **always better.**

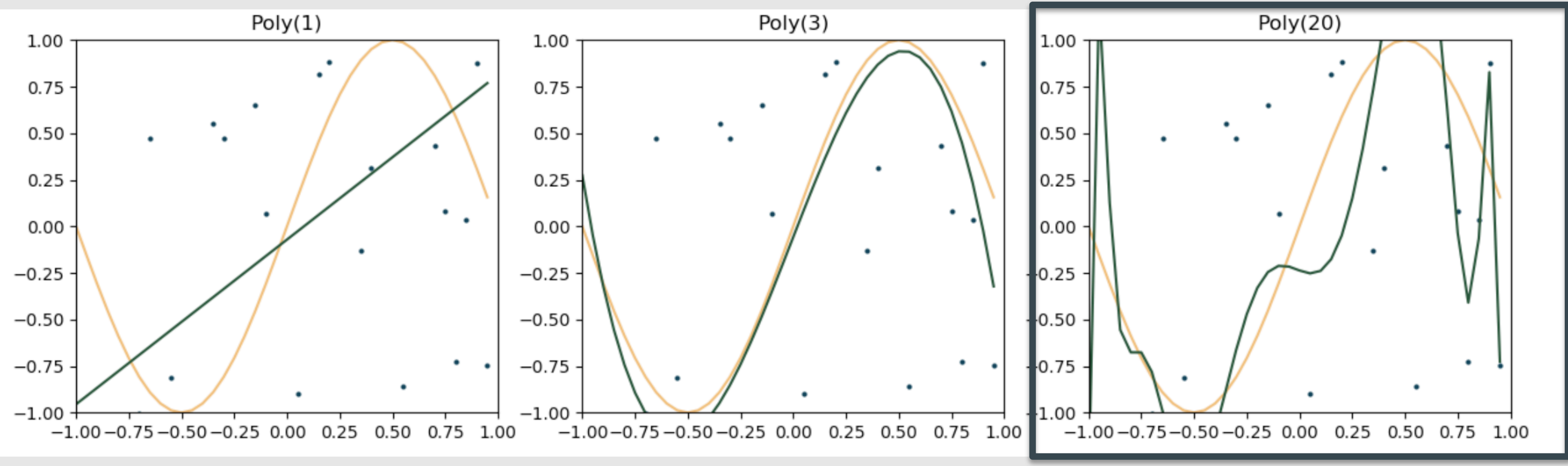
But with **decreasing returns.**

Approximation: Poly 20 > Poly 3



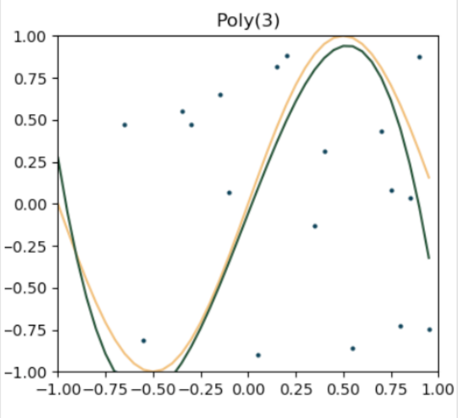
$$\sin(\pi x) \approx \pi x - \frac{\pi^3}{3!}x^3 + \frac{\pi^5}{5!}x^5 - \frac{\pi^7}{7!}x^7 + \frac{\pi^9}{9!}x^9 - \cdots$$

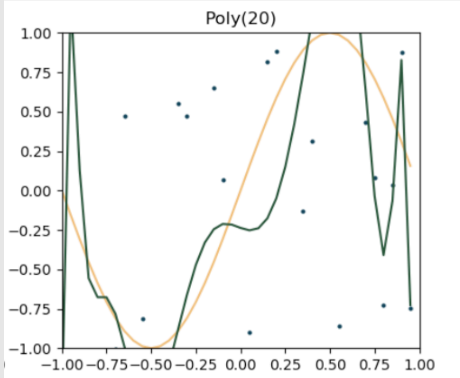
Approximation with noise: Poly 3 > Poly 20



$$\sin(\pi x) \approx \pi x - \frac{\pi^3}{3!}x^3 + \frac{\pi^5}{5!}x^5 - \frac{\pi^7}{7!}x^7 + \frac{\pi^9}{9!}x^9 - \cdots$$

Let’s inspect the Taylor expansion





$$\sin(\pi x) \approx \pi x - \frac{\pi^3}{3!}x^3 + \frac{\pi^5}{5!}x^5 - \frac{\pi^7}{7!}x^7 + \frac{\pi^9}{9!}x^9 - \cdots$$

$\mathbf{x}$	1	x^1	x^2	x^3	x^4	x^5	x^6	x^7	$\ldots$
$\mathbf{w}_{\text{Taylor}}$	0	$\frac{\pi^1}{1!}$	0	$-\frac{\pi^3}{3!}$	0	$\frac{\pi^5}{5!}$	0	$-\frac{\pi^7}{7!}$	$\ldots$
$\mathbf{w}_{\text{poly}(3)}$	-0.23	3.00	1.60	-3.59	0	0	0	0	$\ldots$
Δ	0.23	0.14	1.60	1.58	0	$\frac{\pi^5}{5!}$	0	$\frac{\pi^7}{7!}$	$\ldots$
$\mathbf{w}_{\text{poly}(20)}$	-0.24	-0.44	1.15	31.67	0.48	-75.53	15.10	2.12	$\ldots$
Δ	0.24	3.59	1.15	36.84	0.48	78.08	15.10	2.72	$\ldots$

How could we potentially limit the complexity?

$$\|w_{Taylor}\|_2 \approx 6.6$$
$$\|w_{Poly(20)}\|_2 \approx 370$$

Restricting w

$$\text{Minimize} \frac{1}{N}\sum_{i=1}^{N} l\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

$$\text{S.T } \|w\|_2 \leq R$$

L2 Regularization

In Regression Setting: $Y = w_0 + w^T X$

Minimize $\frac{1}{N}\sum_{i=1}^{N} l\left(y^{(i)}, f\left(x^{(i)}\right)\right)$

S.T $\|w\|_2 \leq R$

Minimize $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - w^{\mathrm{T}} x^{(i)} - w_0)^2$

S.T $\|w\|_2 \leq R$

Ridge Regression

Ridge Regression

In Regression Setting: $Y = w_0 + w^T X$

$$\text{Min} \sum_{i=1}^{N} (y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} w_j - w_0)^2$$

$$\text{S.T} \ \sum_{j=1}^{p} w_j^2 \ \le R$$

Can Not use gradient descent !

Equivalent format

$$\text{Min} \sum_{i=1}^{N} (y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} w_j - w_0)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

- The is one to one correspondence between λ and R
- The intercept w_0 is left out of the penalty term
- Penalization by L2 norm of the parameters in neural networks is also known as weight decay

Ridge Regression (optional)

$$\min_x \{f(x) + \lambda g(x)\} \quad <1> \quad \Longleftrightarrow \quad \begin{cases} \min f(x) \\ \text{s.t. } g(x) \le R \end{cases} \quad <2>$$

Proof: Suppose $x^* = \arg\min_x \{f(x) + \lambda g(x)\}$

Assume: $R = g(x^*)$

Proof by contradiction: Suppose solution for $<2>$ is x'

Then $g(x') \le R = g(x^*)$

We also have $f(x') \le \cancel{g}\, f(x^*)$

which will give us $f(x') + \lambda g(x') < f(x^*) + \lambda g(x^*)$

This contradict with $x^* = \arg\min_x \{f(x) + \lambda g(x)\}$

Ridge Regression

$$Y = w_0 + w^T X$$

$$\text{Min} \sum_{i=1}^{N} (y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} w_j - w_0)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

- The standard least squares coefficient estimate in simple linear regression is scale equivariant: multiply X_j by a constant c simply leads to a scaling the of coefficient estimates
- Ridge regression is NOT equivariant under the scaling of the inputs. We usually standardize the inputs first before applying ridge regression

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sqrt{\frac{1}{n} \sum_{i=1}^{N} \left(x_j^{(i)} - \bar{x}_j \right)^2}}$$

Ridge Regression

$$Y = w_0 + w^T X$$

$$J(w) = \sum_{i=1}^{N} (y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} w_j - w_0)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

Then we can write the penalized residual sum of squares (λ is the regularization parameter)

$$J(w) = (y - Xw)^T (y - Xw) + \lambda w^T w$$

To minimize $J(w)$, consider the first order derivative:

$$\nabla_w J(w) = -2X^T (y - Xw) + 2\lambda w = 0$$

Ridge regression has closed form solution

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

Ridge Regression

Solution for ridge regression: $\hat{w} = (X^T X + \lambda I)^{-1} X^T y$

Matrix $X^T X$ is p.s.d. , consider eigendecomposition of $X^T X$

$$X^T X = U \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p^2 \end{bmatrix} U^T$$

Even when X is not full rank, we still have

$$X^T X + \lambda I = U \begin{bmatrix} \sigma_1^2 + \lambda & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p^2 + \lambda \end{bmatrix} U^T$$

is always symmetric and positive definite (hence has full rank)

Hence $\hat{w} = (X^T X + \lambda I)^{-1} X^T y$ always exists and unique!

Lasso Regression

Model: $Y = w_0 + w^T X$

<u>Lasso Regression</u>

Loss function: $\sum_{i=1}^{N}(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} w_j - w_0)^2$

Regularization: $\sum_{j=1}^{p} |w_j| \ \leq R$

<u>Or Equivalently</u>

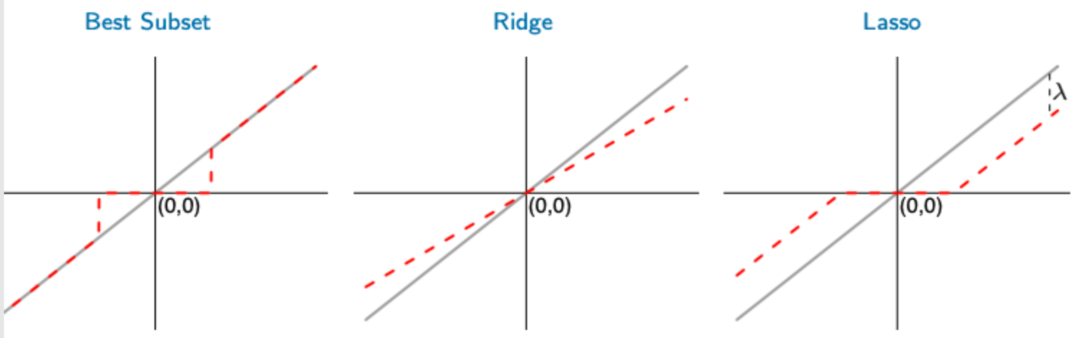
Min $\sum_{i=1}^{N}(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} w_j - w_0)^2 + \lambda \sum_{j=1}^{p} |w_j|$

Ridge and Lasso Regression

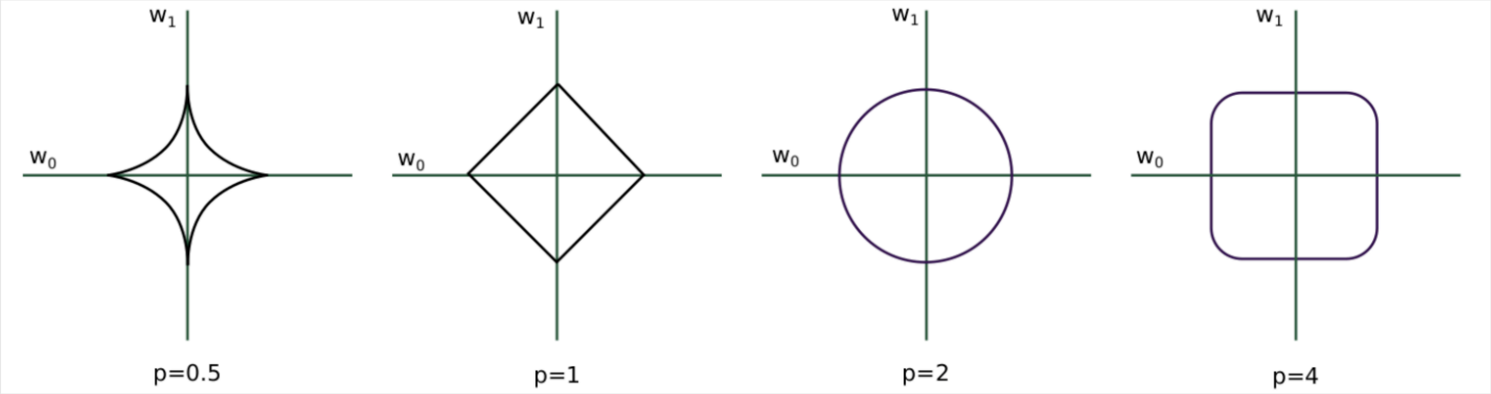
When the input matrix X is orthogonal,

- Ridge regression does a proportional shrinkage.
- Lasso translates each coefficient by a constant factor λ, truncating at zero.
- Best-subset selection drops all variables with coefficients smaller than the M^{th} largest

Estimator	Formula
Feature Selection	$\hat{w}_j \cdot I(\hat{w}_j > \hat{w}_{(M)})$
Ridge	$\hat{w}_j/(1+\lambda)$
Lasso	$Sign(\hat{w}_j)(\lvert\hat{w}_j\rvert - \lambda)_+$

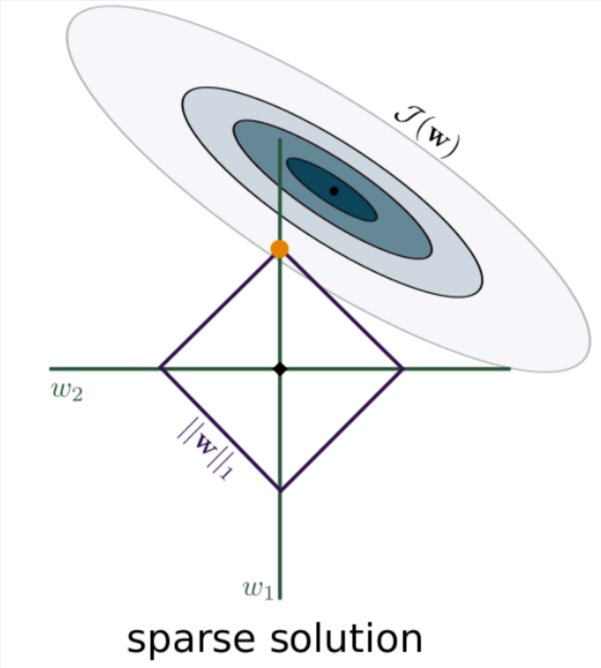


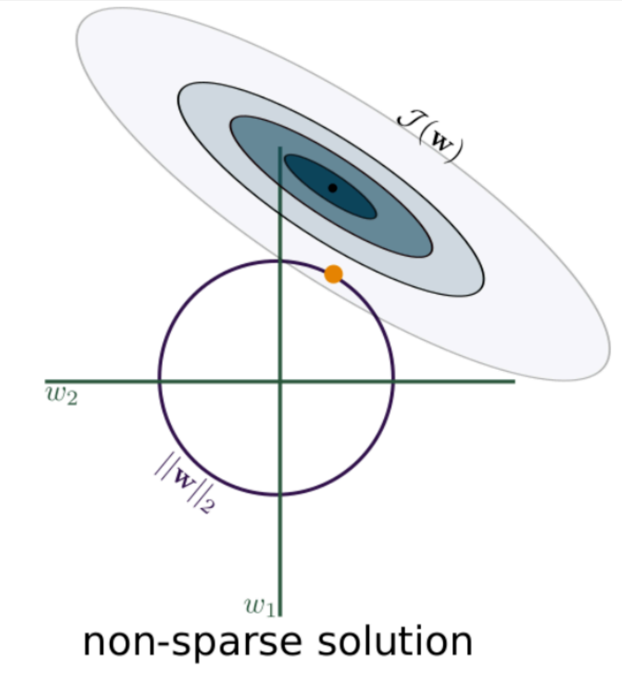
Note on Regularization



$$\|w\|_p = \left(\sum_{j=1}^{N} \left|w_j\right|^p\right)^{1/p}$$

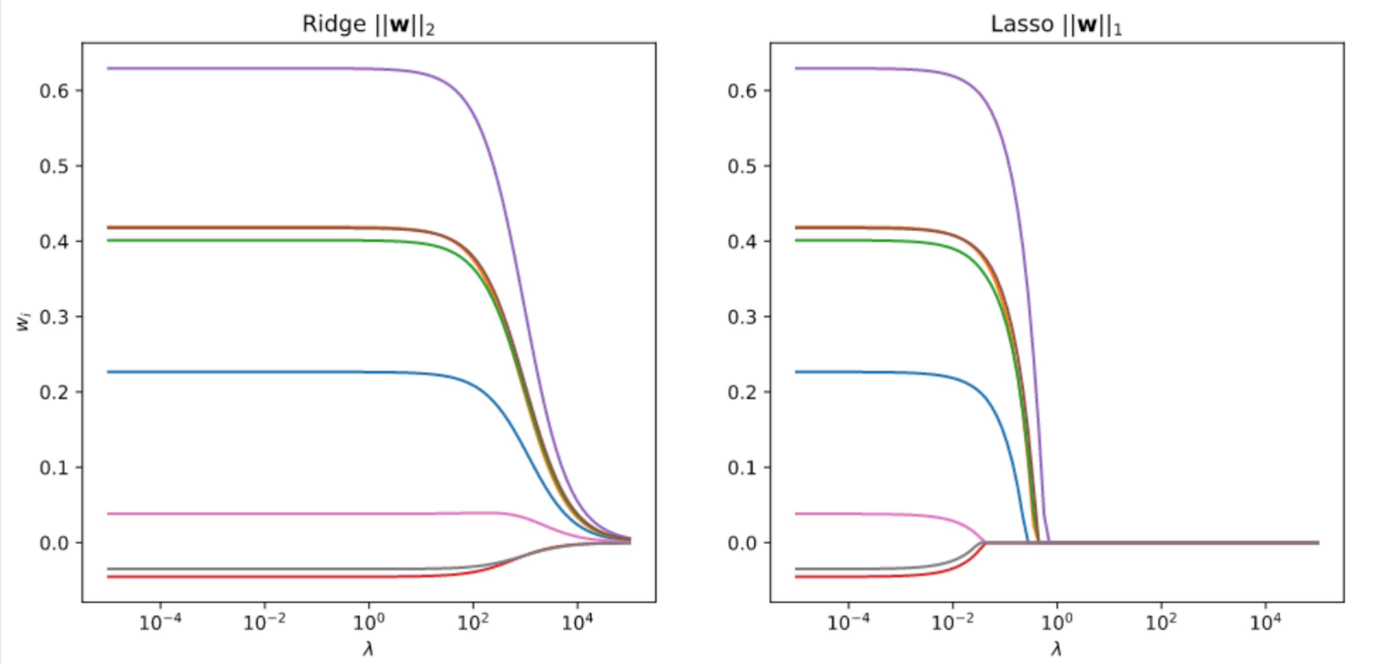
Note on regularization



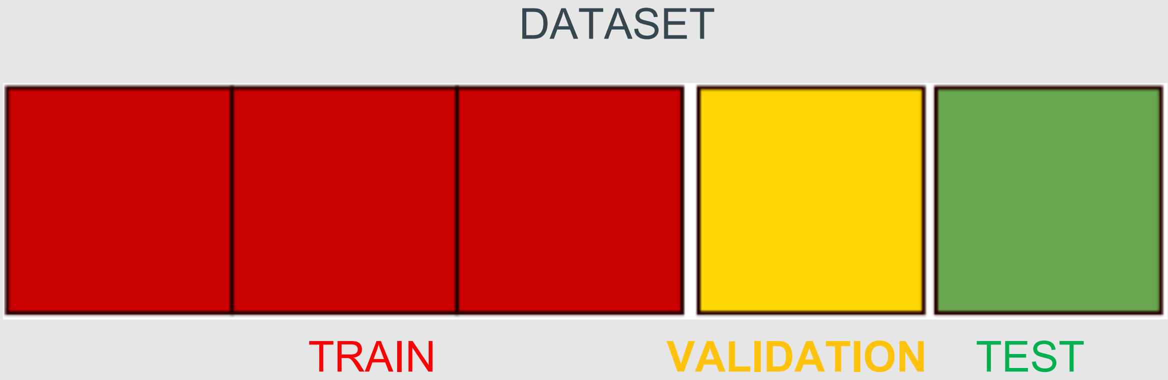


The lasso regularization performs variable selection

Note on regularization



How to choose tuning parameter λ



Exercise

Restricting w in python using optimization

Extend sklearn so that we can do

$$\text{Minimize } \frac{1}{N}\sum_{i=1}^{N} l\left(y^{(i)}, f\left(x^{(i)}\right)\right)$$

$$\text{S.T } \|w\| \leq W$$

Object oriented programming

```python
from sklearn.base import BaseEstimator

class ConstantModel(BaseEstimator):
    def __init__(self):
        print("A constant model was born")
        self.expected_value = None

    def fit(self, X, y):
        self.expected_value = np.mean(y)
        return self

    def predict(self, X):
        return self.expected_value
```

```python
model = ConstantModel()
```

```
A constant model was born
```

```python
model.fit([1,2,3,4,5,6],[2,2,3,3,2,3])
model.predict(1)
```

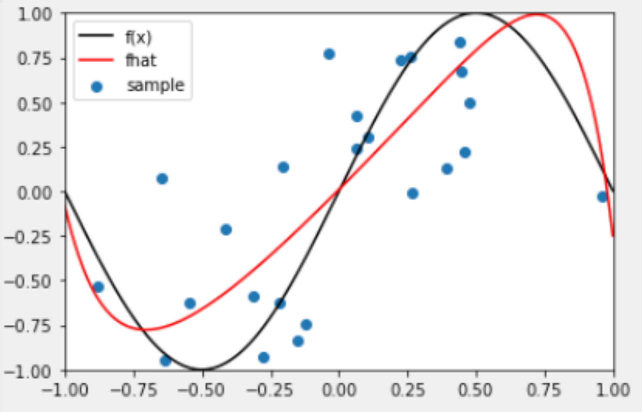
```
2.5
```

Exercise

- Predict what will happen when you set D=11, W=0. Then try it out
- Predict what will happen when you set D=11, W=∞. Then try it out by setting W=10e10
- Try to find the right value of W that gives a result similar to the plot below for a 11-degree polynomial

```python
# train the model with your chosen parameters
model = MyConstrainedRegression(W=...,D=...,verbose=True)
model.fit(X,y)

# plot the result:
_,ax = plt.subplots(1,1)
ax.scatter(X,y,label="sample");
plotf(ax,f, label="f(x)")
plotf(ax, model.predict, c='r',label="fhat")
```



Extra: vary the parameter W