10-315: Introduction to Machine Learning Lecture 7 – Regularization

Henry Chai

2/9/22

Linear Regression

- Given *design matrix*

$$\boldsymbol{A} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} = \begin{bmatrix} 1 & X_1^{(1)} & \cdots & X_1^{(p)} \\ 1 & X_2^{(1)} & \cdots & X_2^{(p)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_n^{(1)} & \cdots & X_n^{(p)} \end{bmatrix} \in \mathbb{R}^{n \times p+1}$$

- and *target vector*

$$\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \in \mathbb{R}^n$$

- the goal of linear regression is to find

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\, J(\beta) = \underset{\beta}{\operatorname{argmin}}\, \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$$

Poll Review: Is $J(\beta)$ convex in β?

$$J(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$$

A) Convex, quadratic in β

B) Non-convex, $\boldsymbol{A}$ may not be positive semi-definite

C) Depends on conditioning (ratio of max:min eigenvalues) of $\boldsymbol{A}^T\boldsymbol{A}$

D) Convex, $\boldsymbol{A}^T\boldsymbol{A}$ is positive semi-definite

Minimizing the Mean Squared Error

$$J(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$$

$$= \frac{1}{n}(\beta^T\boldsymbol{A}^T\boldsymbol{A}\beta - 2\beta^T\boldsymbol{A}^T\boldsymbol{Y} - \boldsymbol{Y}^T\boldsymbol{Y})$$

$$\nabla_\beta J(\beta) = \frac{1}{n}(2\boldsymbol{A}^T\boldsymbol{A}\beta - 2\boldsymbol{A}^T\boldsymbol{Y})$$

$$\nabla_\beta J(\hat{\beta}) = \frac{1}{n}\left(2\boldsymbol{A}^T\boldsymbol{A}\hat{\beta} - 2\boldsymbol{A}^T\boldsymbol{Y}\right) = 0$$

$$\boldsymbol{A}^T\boldsymbol{A}\hat{\beta} = \boldsymbol{A}^T\boldsymbol{Y}$$

$$\hat{\beta} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Y}$$

Minimizing the Mean Squared Error

$$\hat{\beta} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Y}$$

1. Is $\boldsymbol{A}^T\boldsymbol{A}$ invertible?

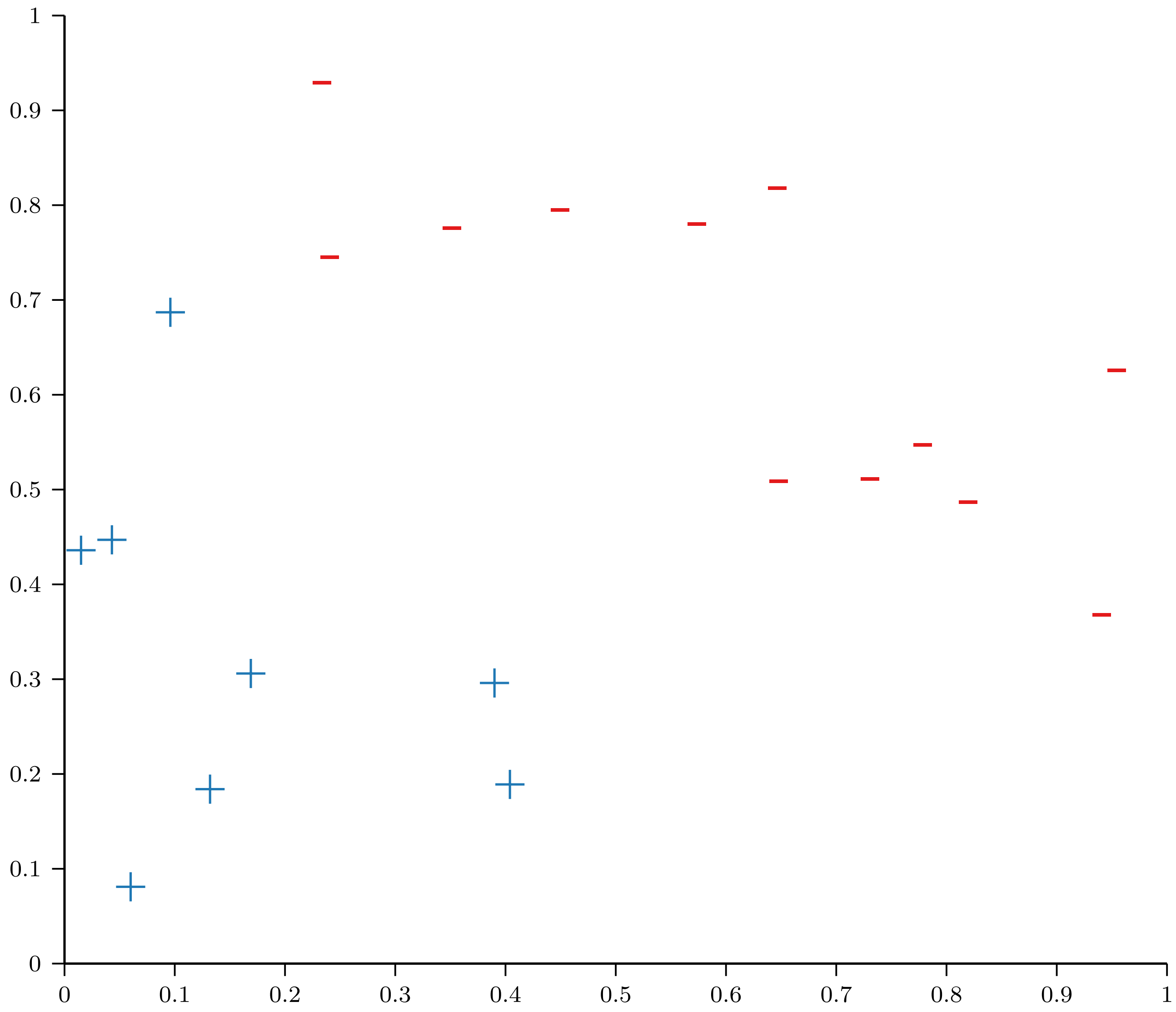
   When $n \gg p + 1$, $\boldsymbol{A}^T\boldsymbol{A}$ is (almost always) full rank and therefore, invertible

2. If so, how computationally expensive is inverting $\boldsymbol{A}^T\boldsymbol{A}$?

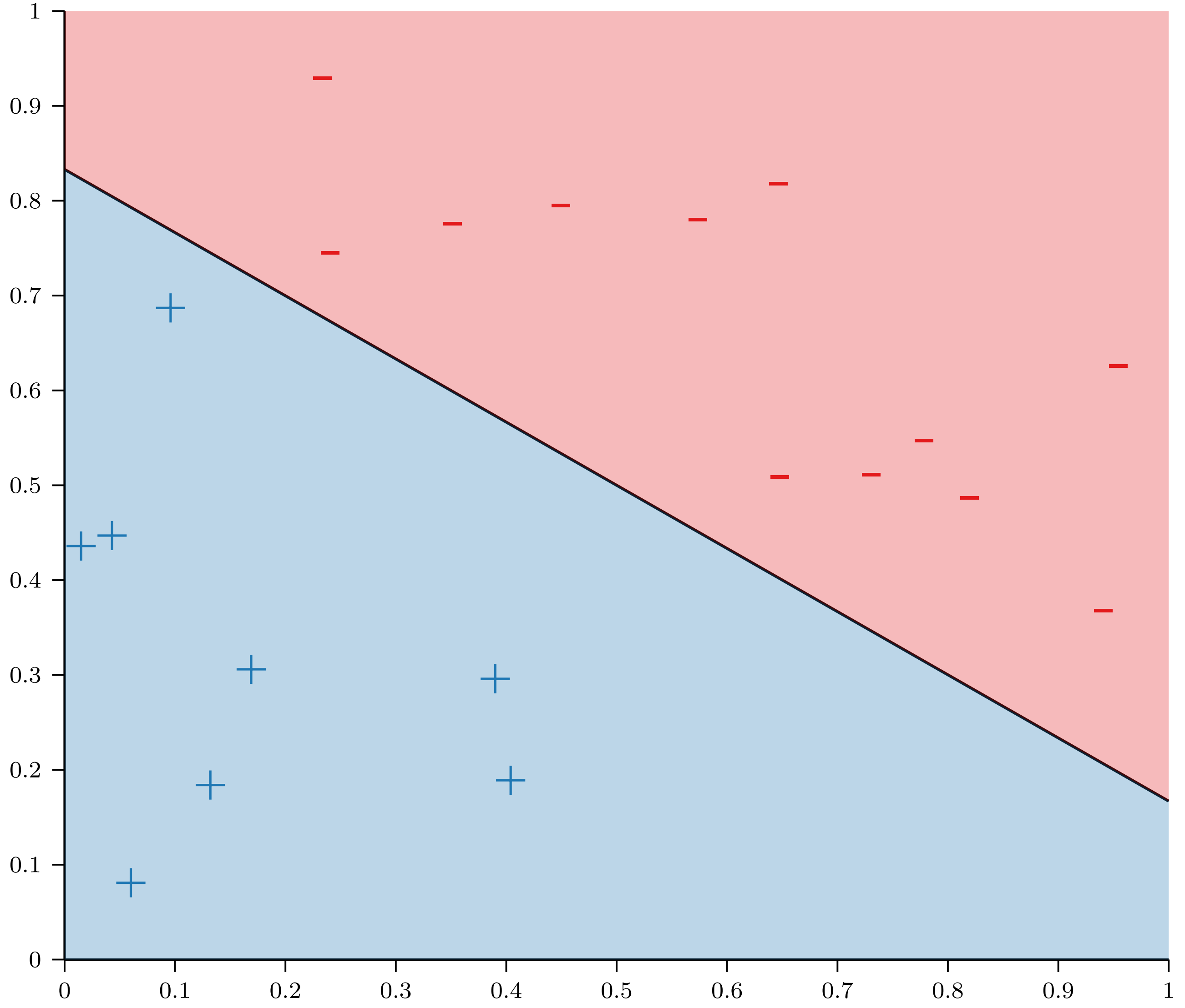
   $\boldsymbol{A}^T\boldsymbol{A} \in R^{p+1\times p+1}$ so inverting $\boldsymbol{A}^T\boldsymbol{A}$ takes $O(p^3)$ time!
   Can use gradient *descent* to speed things up:

$$\beta^{(t+1)} = \beta^{(t)} - \eta\nabla_\beta J(\beta) = \beta^{(t)} - \frac{2\eta}{n}(\boldsymbol{A}^T\boldsymbol{A}\beta - \boldsymbol{A}^T\boldsymbol{Y})$$

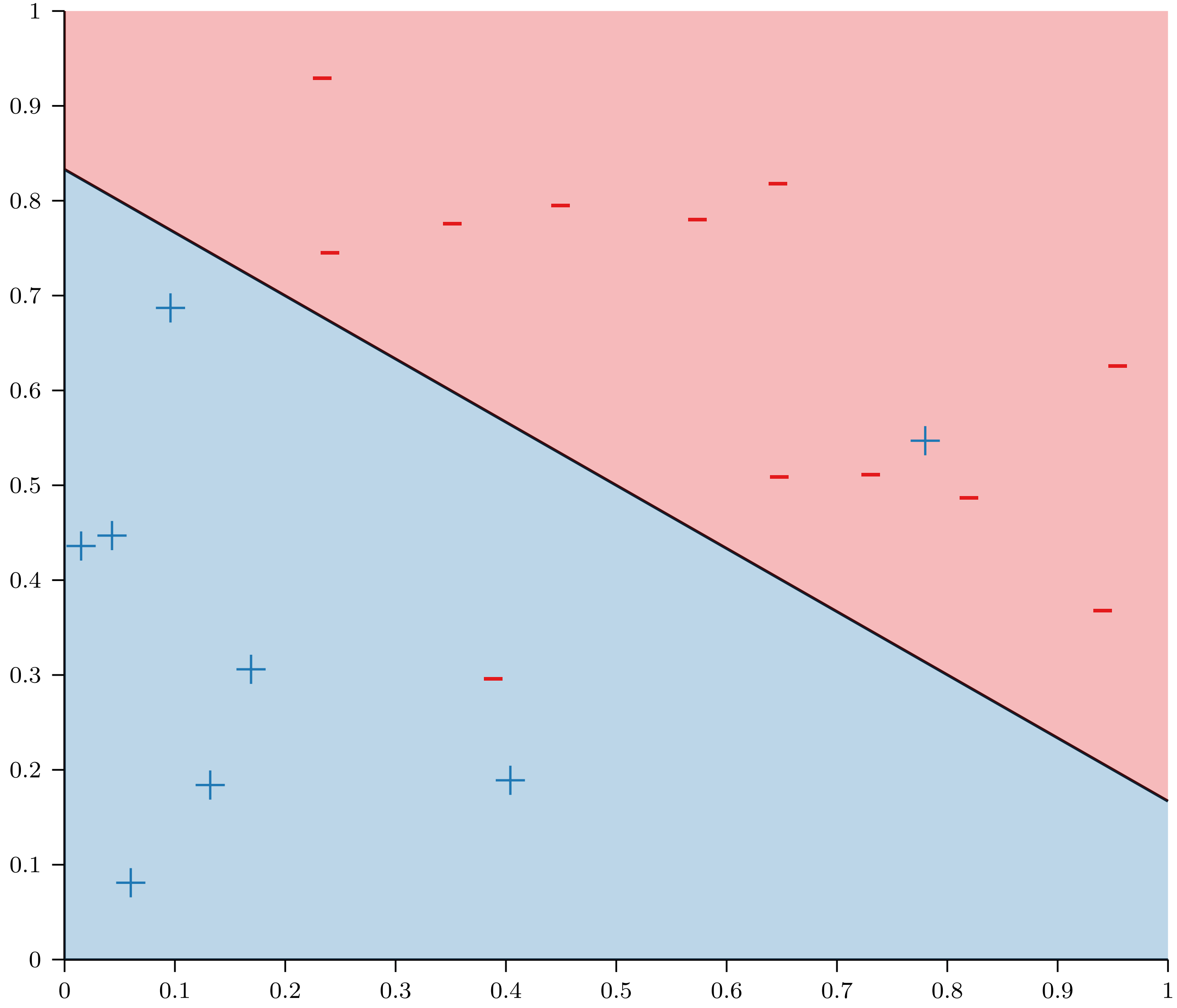
Linear Models



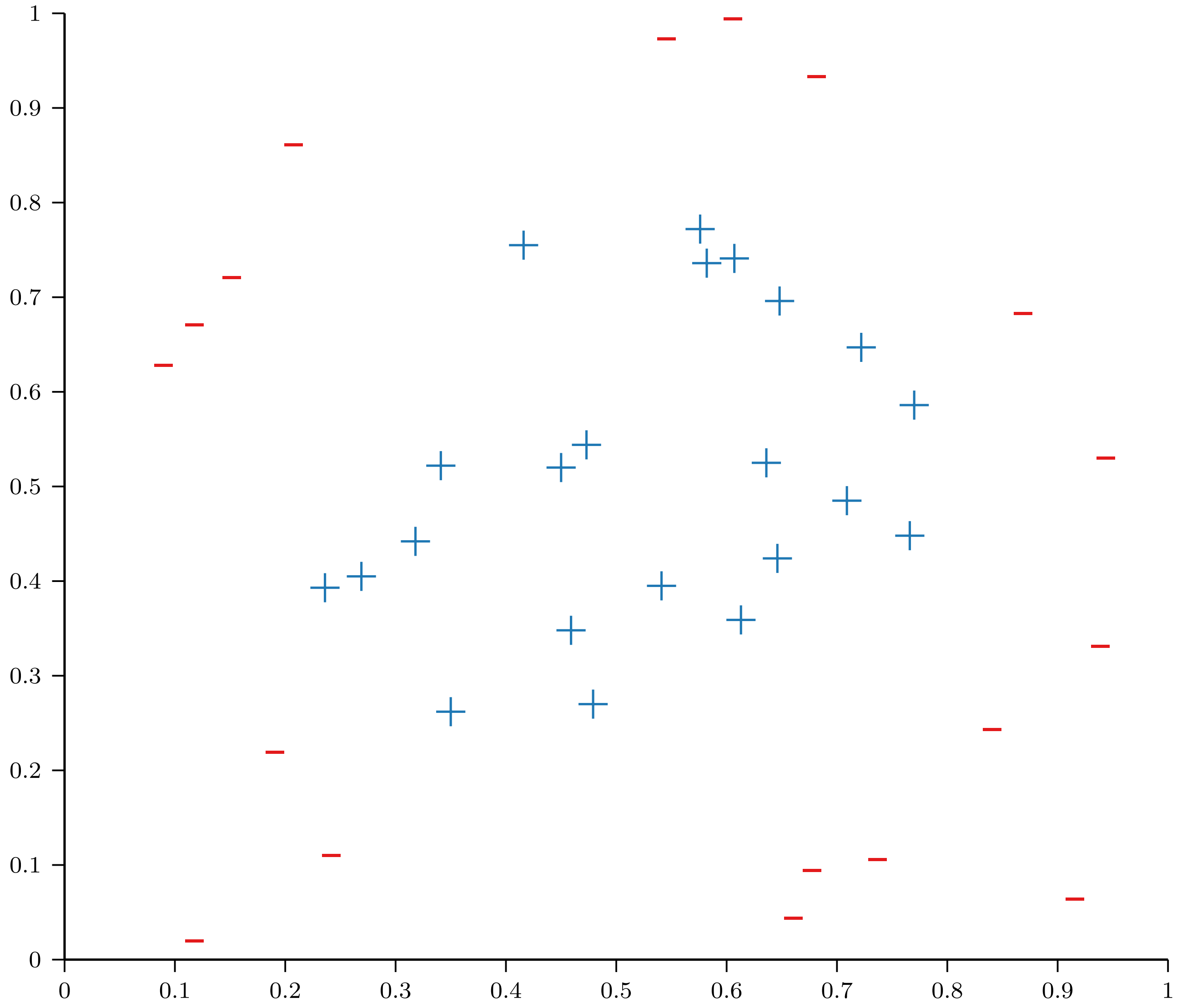
Linear Models



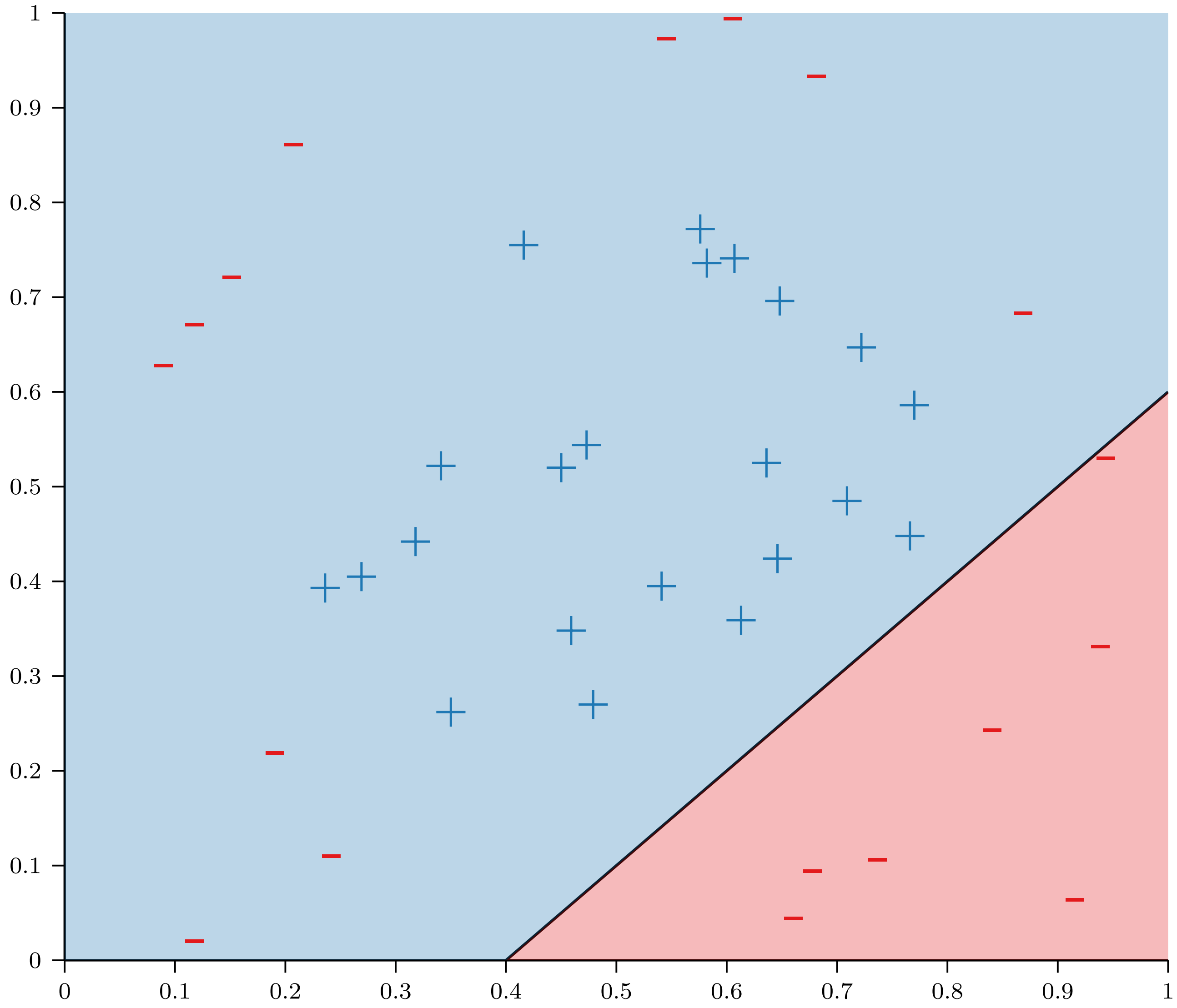
Linear Models



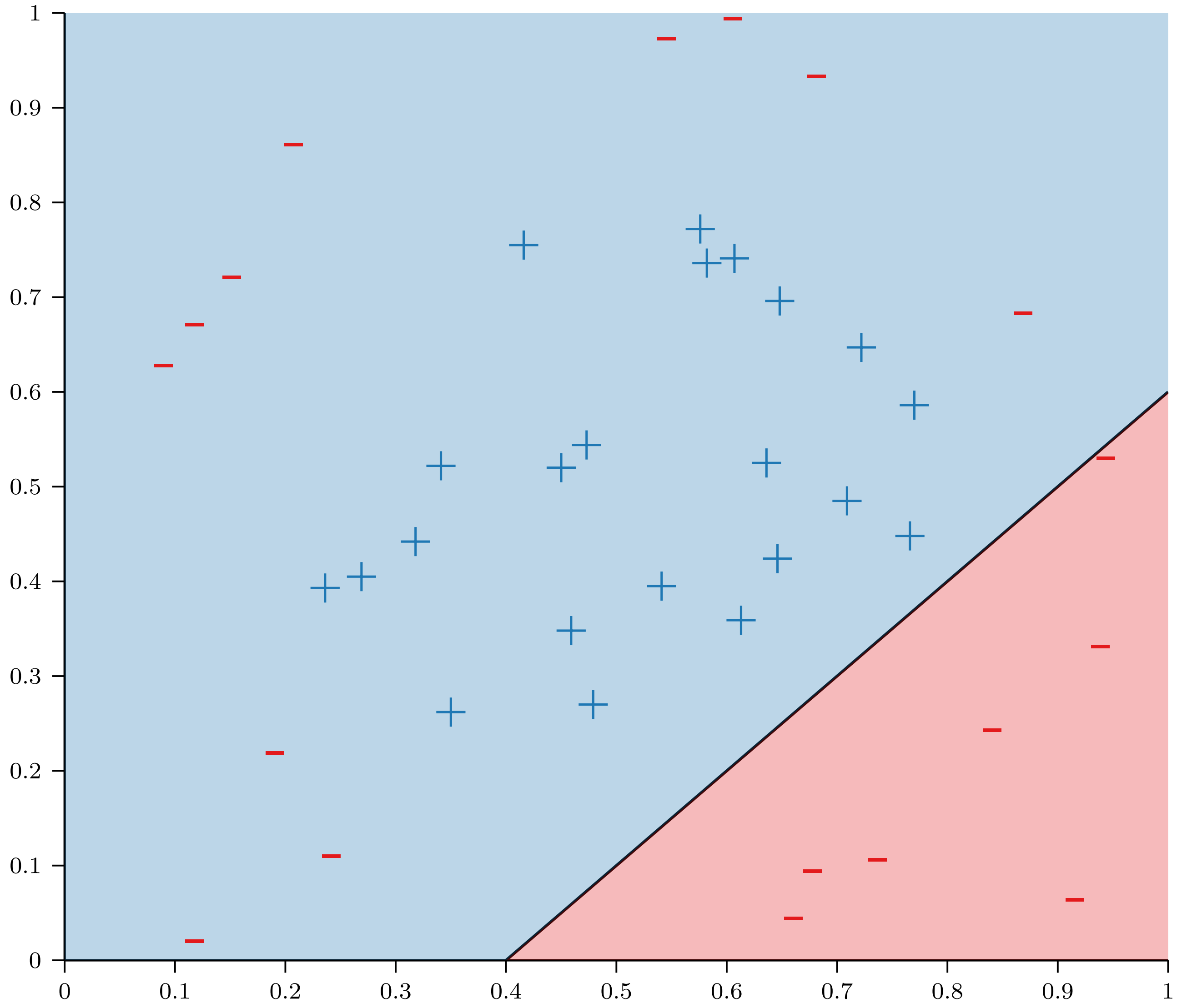
Linear Models?



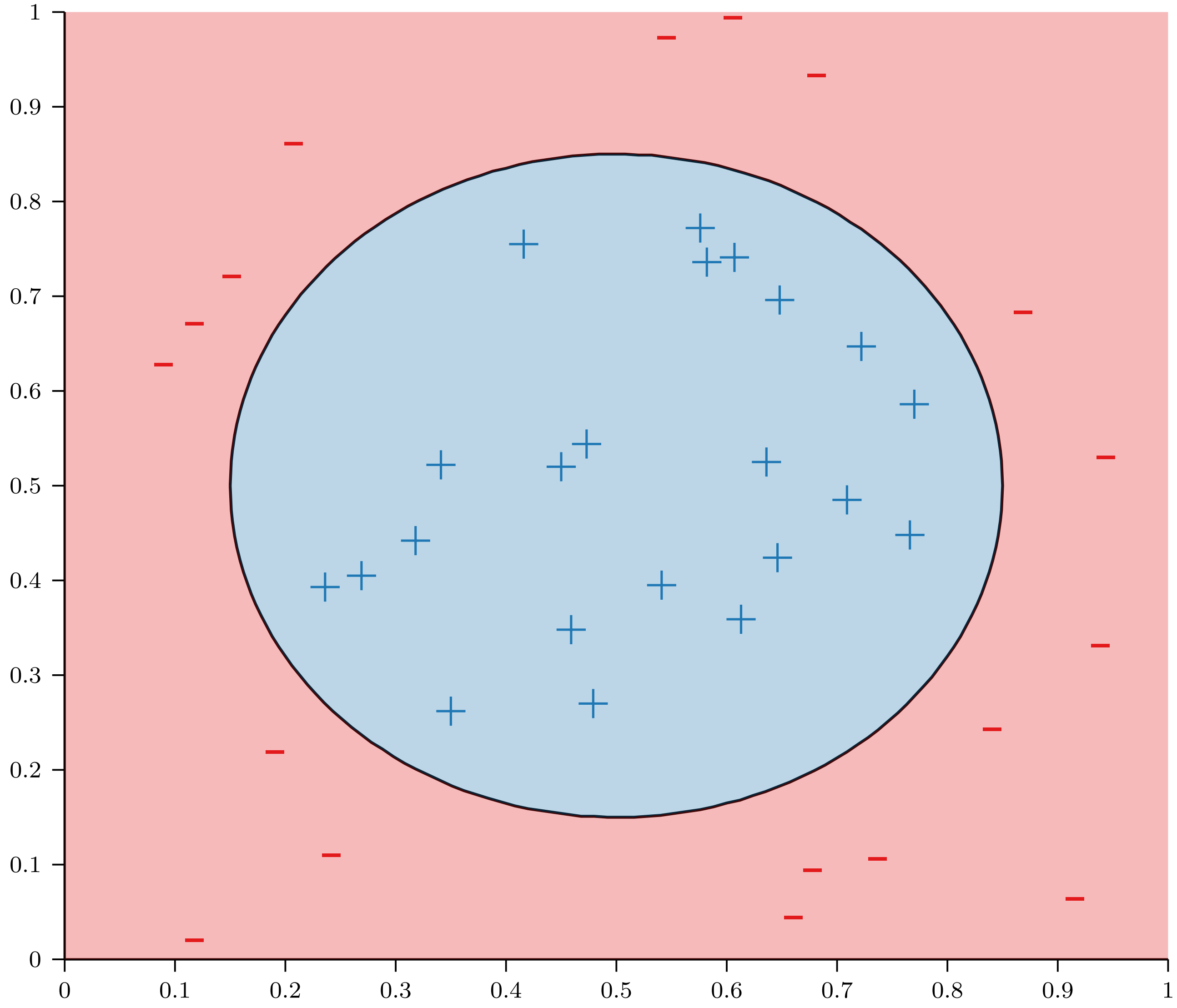
Linear Models?



Linear Models?



Nonlinear Models

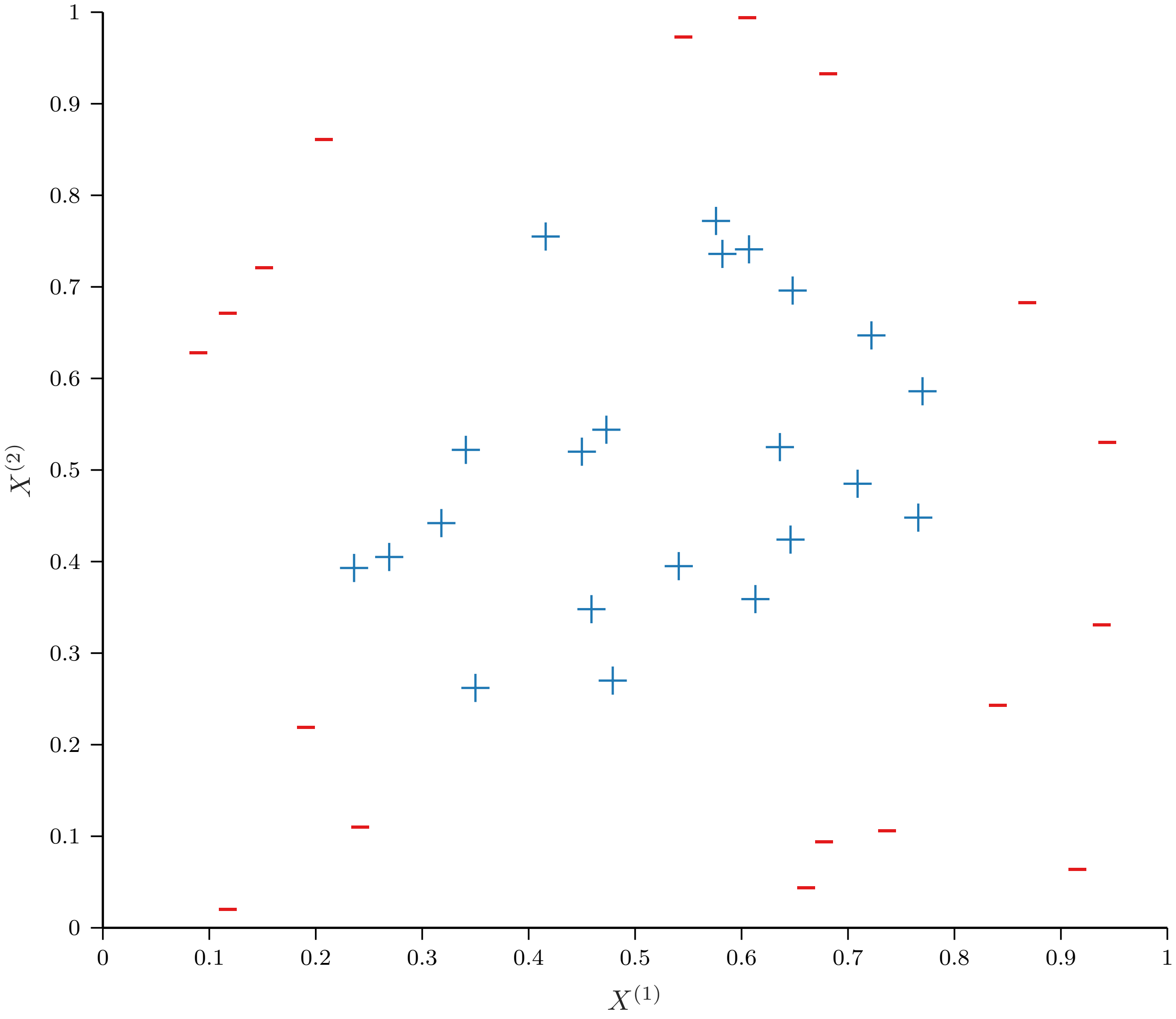


Feature Transforms

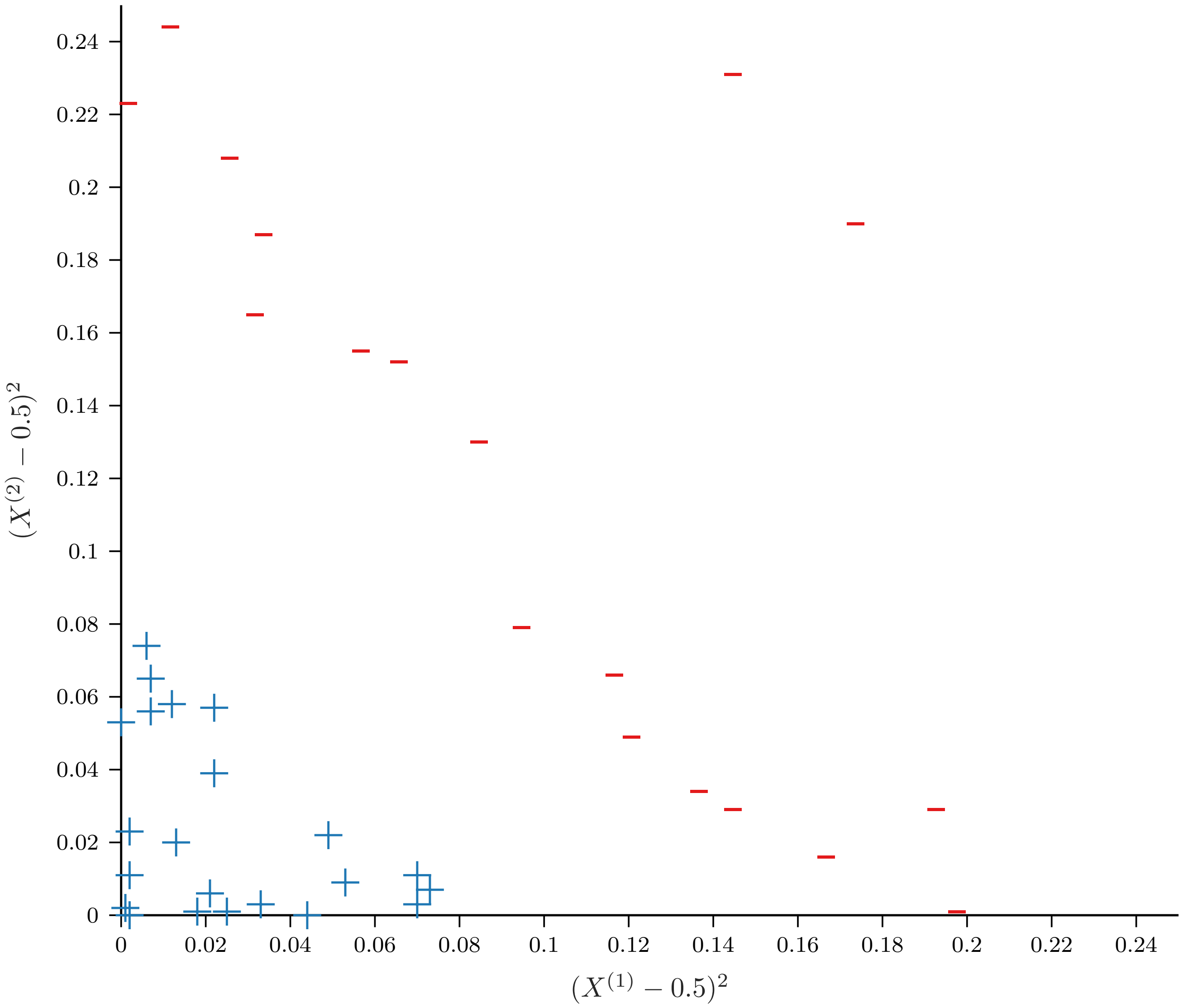
- Given p-dimensional inputs $X = \left[X^{(1)}, \dots, X^{(p)}\right]$, first compute some transformation of our input, e.g.,

$$\phi\left(\left[X^{(1)}, X^{(2)}\right]\right) = \left[\left(X^{(1)} - 0.5\right)^2, \left(X^{(2)} - 0.5\right)^2\right]$$

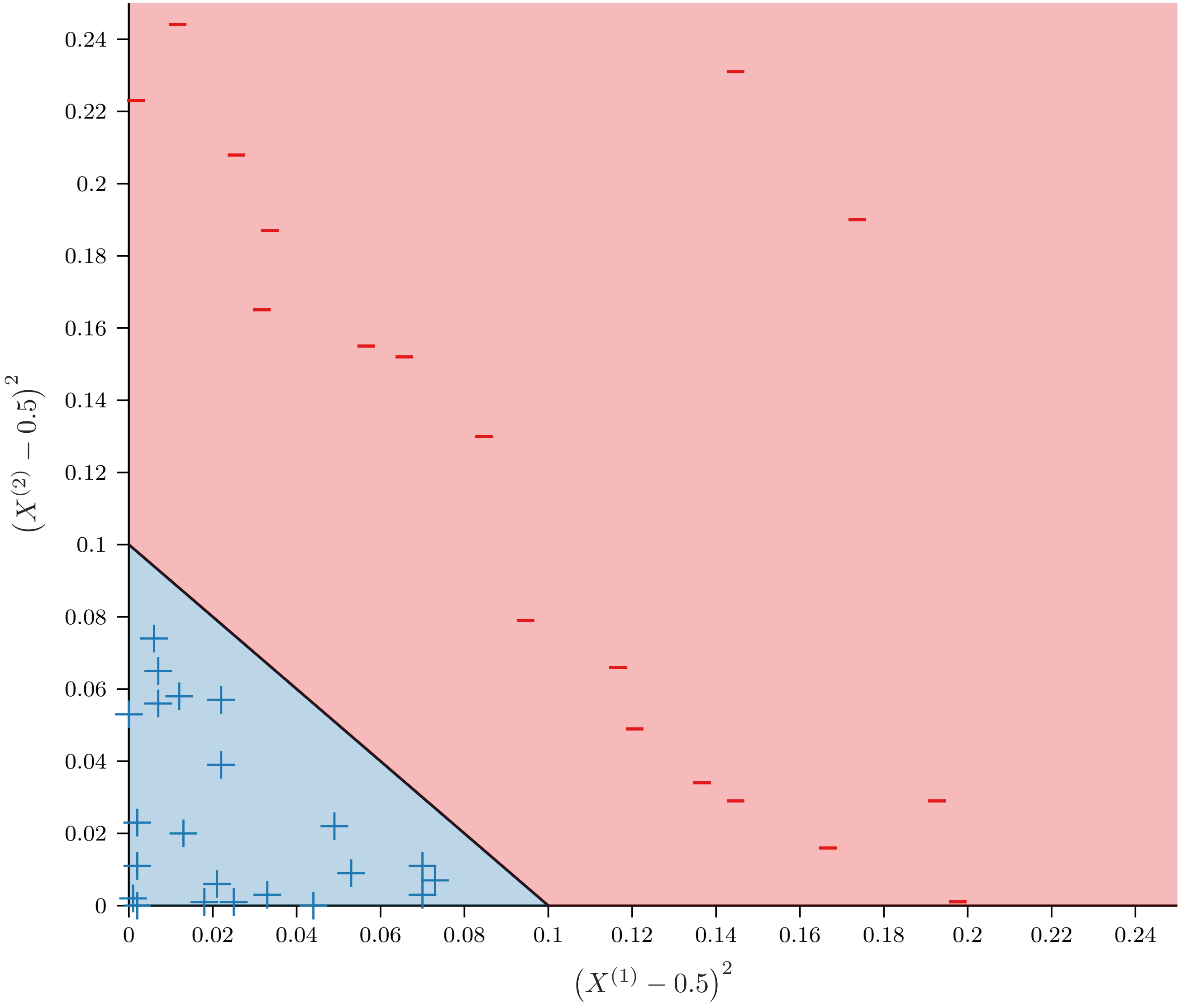
Nonlinear Models



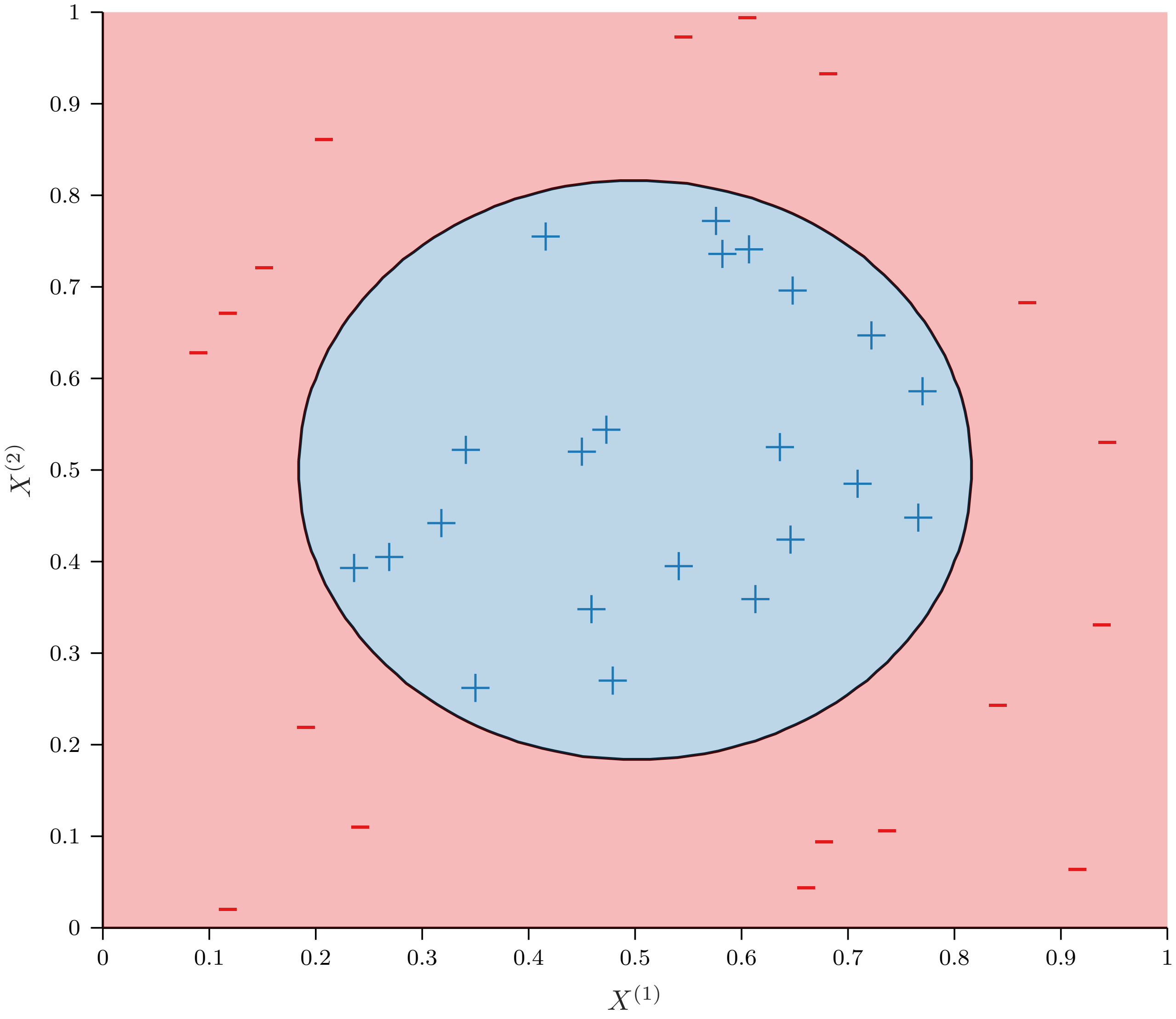
Nonlinear Models



Nonlinear Models



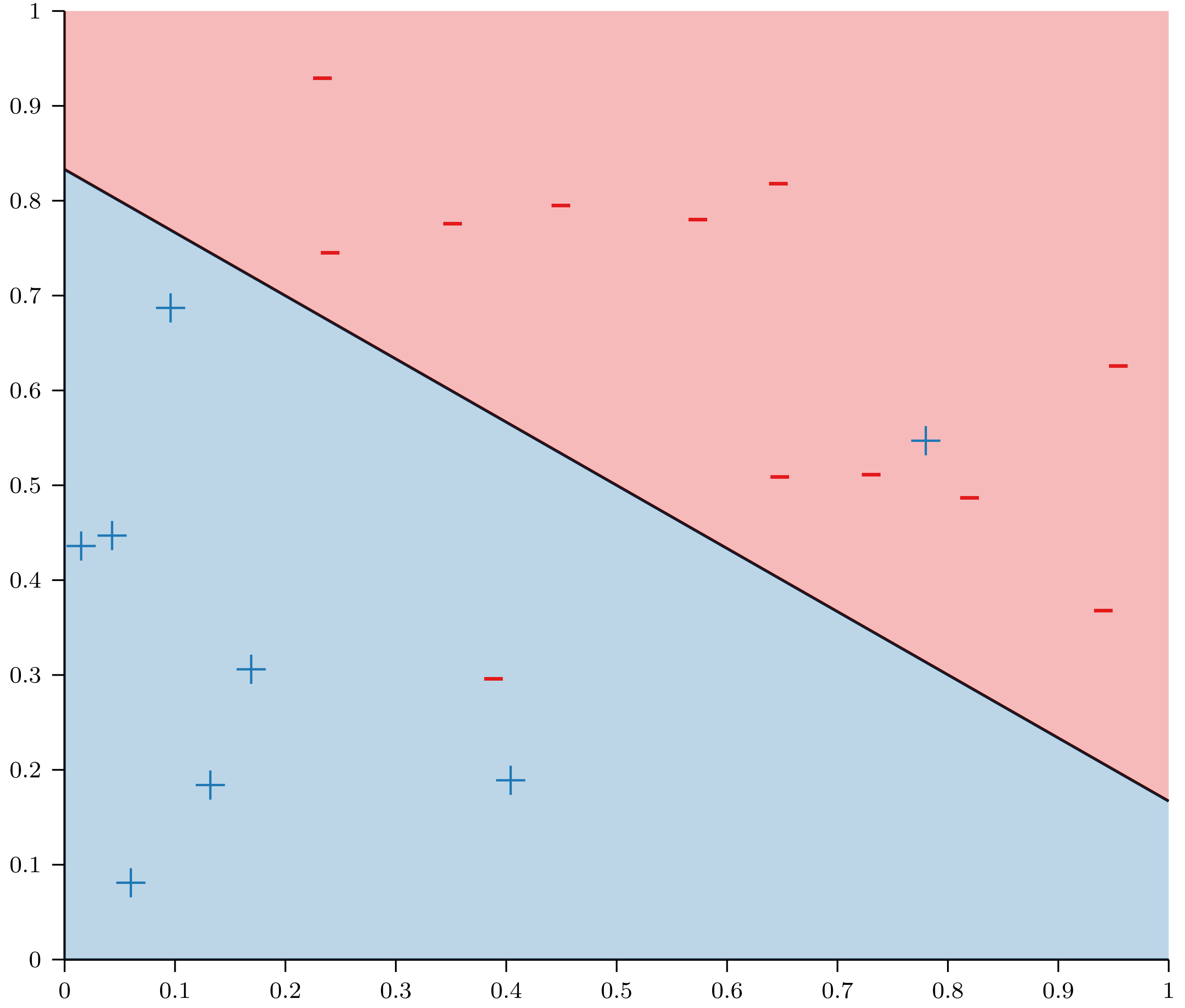
Nonlinear Models



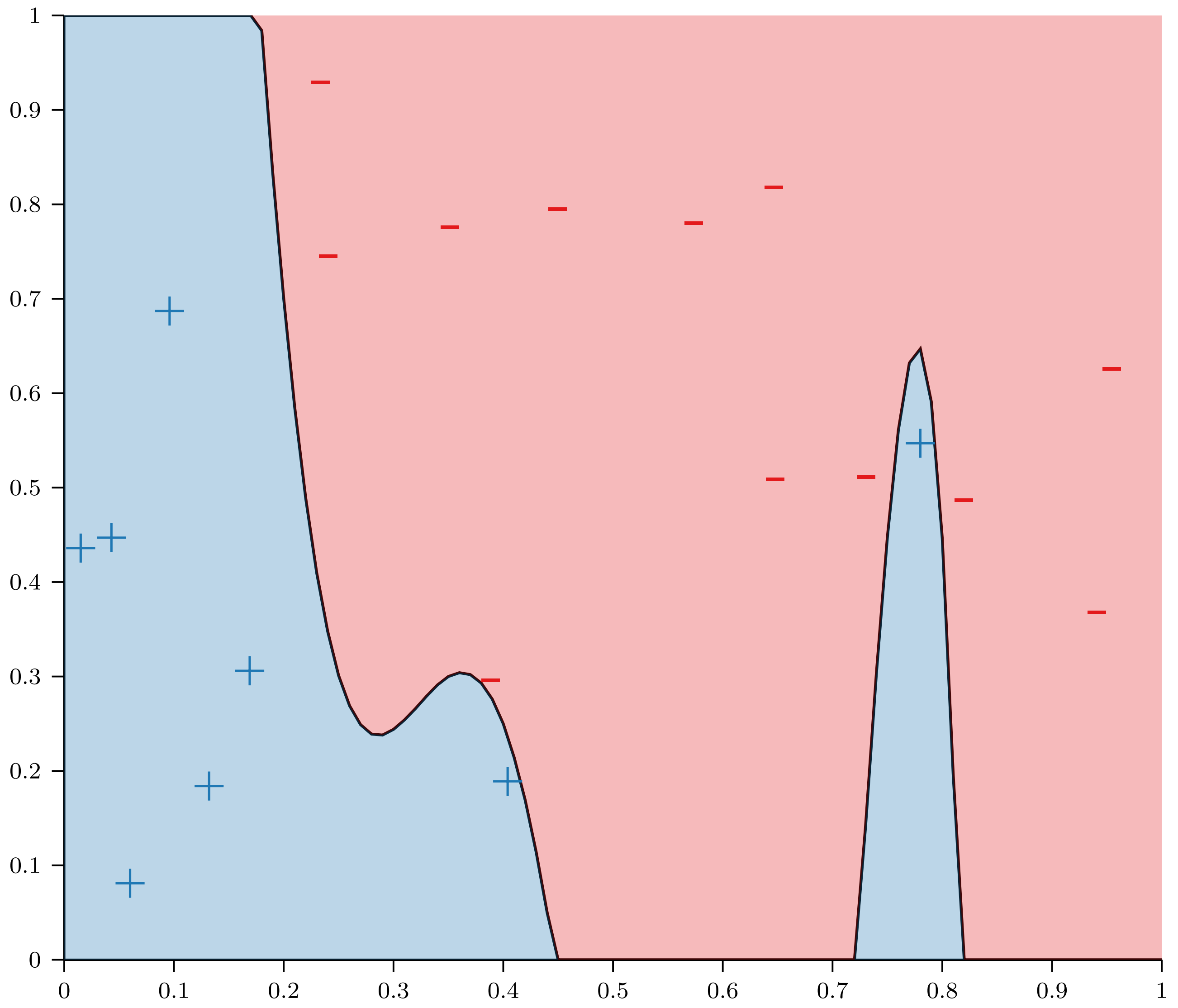
General k^{th}-order Transforms

- $\phi_{2,2}([X^{(1)}, X^{(2)}]) = \left[X^{(1)}, X^{(2)}, {X^{(1)}}^2, X^{(1)}X^{(2)}, {X^{(2)}}^2\right]$
- $\phi_{2,3}([X^{(1)}, X^{(2)}]) =$
$$\left[\phi_{2,2}([X^{(1)}, X^{(2)}]), {X^{(1)}}^3, {X^{(1)}}^2 X^{(2)}, X^{(1)}{X^{(2)}}^2, {X^{(2)}}^3\right]$$
- $\phi_{2,4}([X^{(1)}, X^{(2)}]) =$
$$[\phi_{2,3}([X^{(1)}, X^{(2)}]), {X^{(1)}}^4, {X^{(1)}}^3 X^{(2)}, {X^{(1)}}^2 {X^{(2)}}^2, X^{(1)}{X^{(2)}}^3, {X^{(2)}}^4]$$
- $\phi_{2,Q}$ maps a 2-dimensional input to a $\frac{Q(Q+3)}{2}$-dimensional output
- Scales even worse for higher-dimensional inputs…

Linear Models



Nonlinear Models?



Feature Transforms: Tradeoffs

	Low-Dimensional Input Space	High-Dimensional Input Space
Training Error	High	Low
Generalization	Good	Bad

Overfitting

Feature Transforms: Experiment

- $X \in \mathbb{R}$, $Y \in \mathbb{R}$ and $n = 20$
- Targets are generated by a 10^{th}-order polynomial in X with additive Gaussian noise:

$$Y = \sum_{d=0}^{10} a_d X^d + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2)$$

- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomials
  - $\phi_{1,2}(X) = [X, X^2]$
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$

Poll 1: Which model do you think will have lower *training* error?

A. $\mathcal{F}_2$
B. $\mathcal{F}_{10}$

- $X \in \mathbb{R}$, $Y \in \mathbb{R}$ and $n = 20$
- Targets are generated by a 10^{th}-order polynomial in X with additive Gaussian noise:

$$Y = \sum_{d=0}^{10} a_d X^d + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2)$$

- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomials
  - $\phi_{1,2}(X) = [X, X^2]$
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$

Poll 2: Which model do you think will have lower *true* error?

A. $\mathcal{F}_2$
B. $\mathcal{F}_{10}$

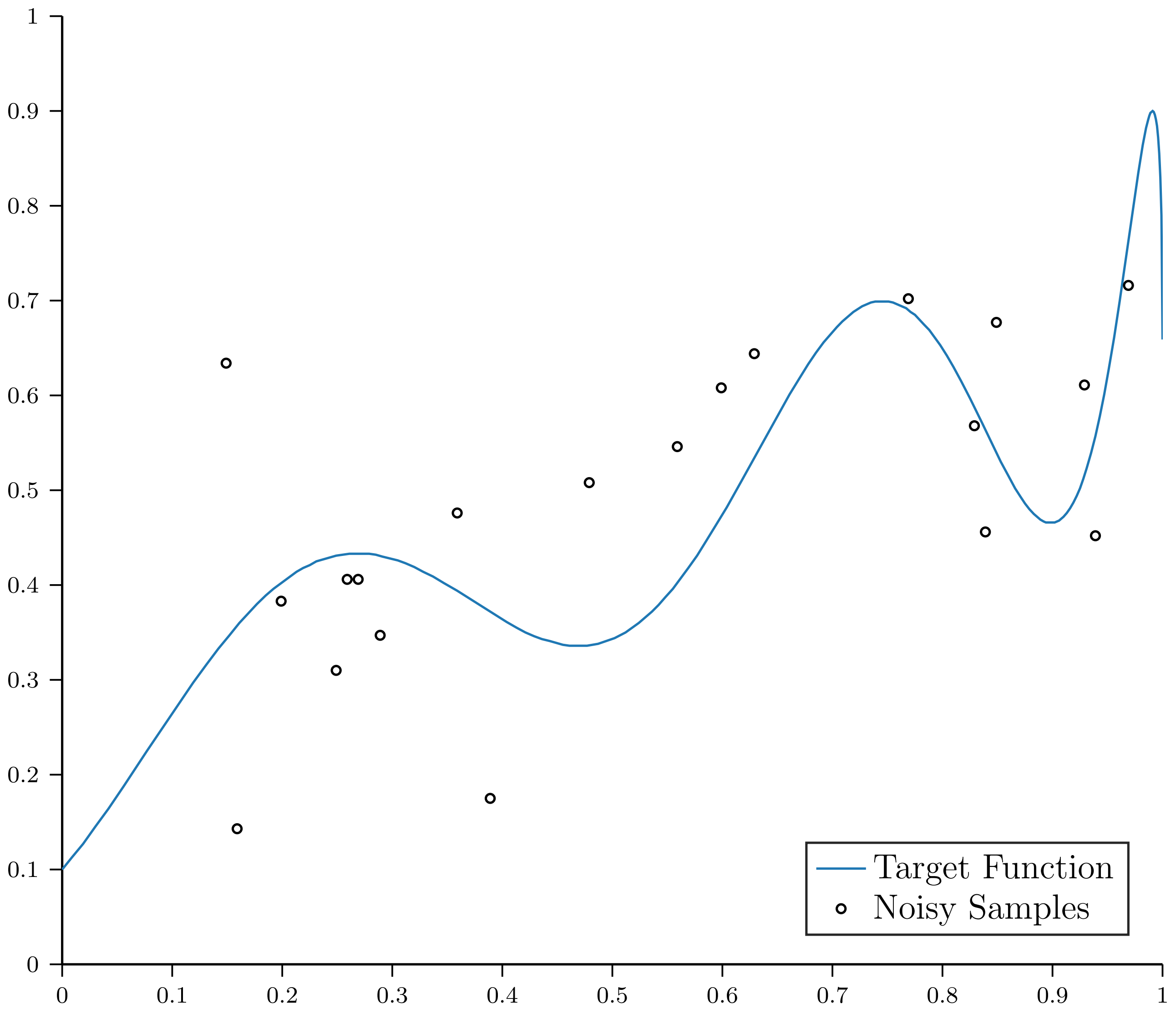
- $X \in \mathbb{R}$, $Y \in \mathbb{R}$ and $n = 20$
- Targets are generated by a 10^{th}-order polynomial in X with additive Gaussian noise:

$$Y = \sum_{d=0}^{10} a_d X^d + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2)$$

- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomials
  - $\phi_{1,2}(X) = [X, X^2]$
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$

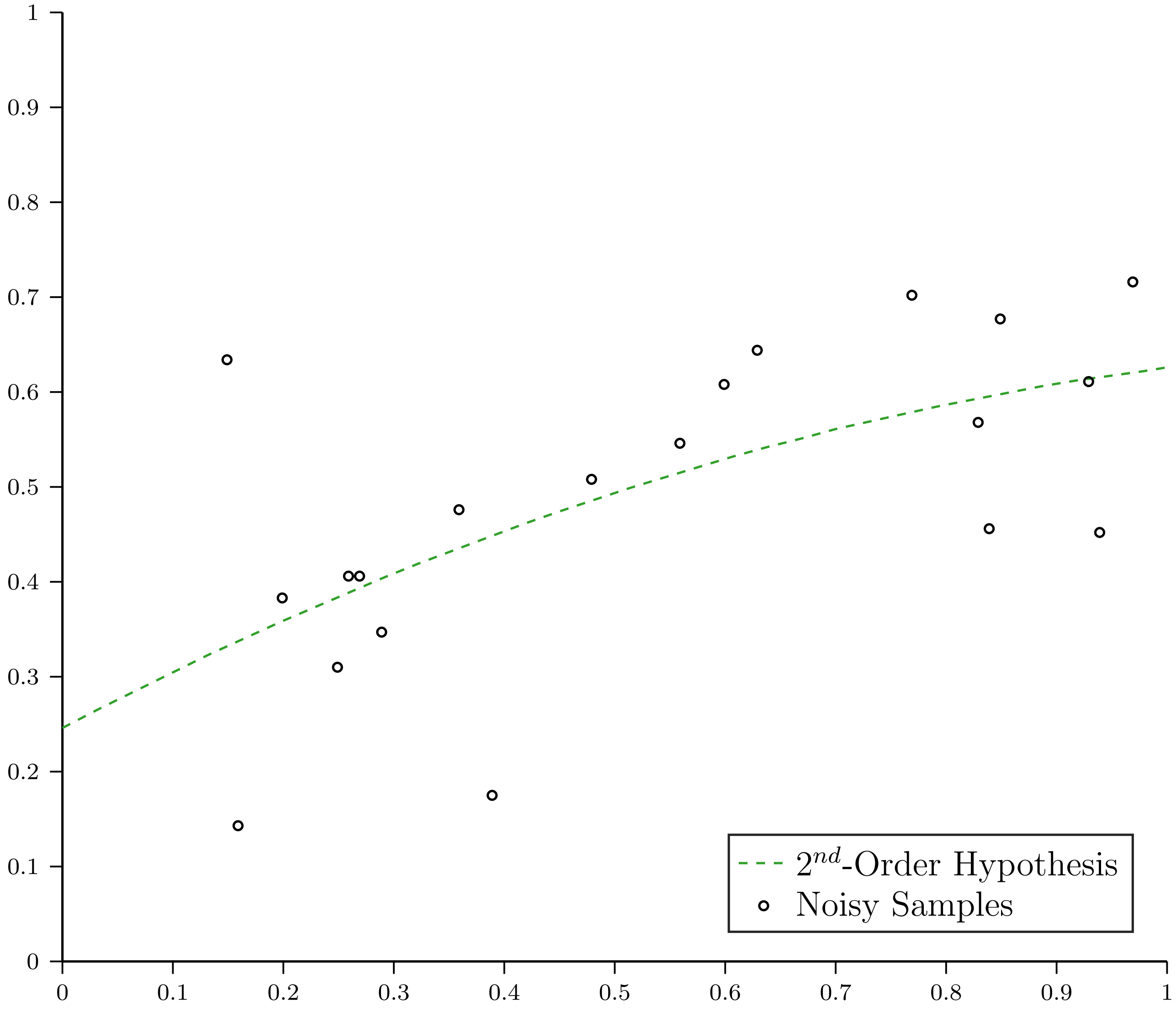
Noisy Targets

- 10-dimensional target function with additive Gaussian noise
- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomial
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomial



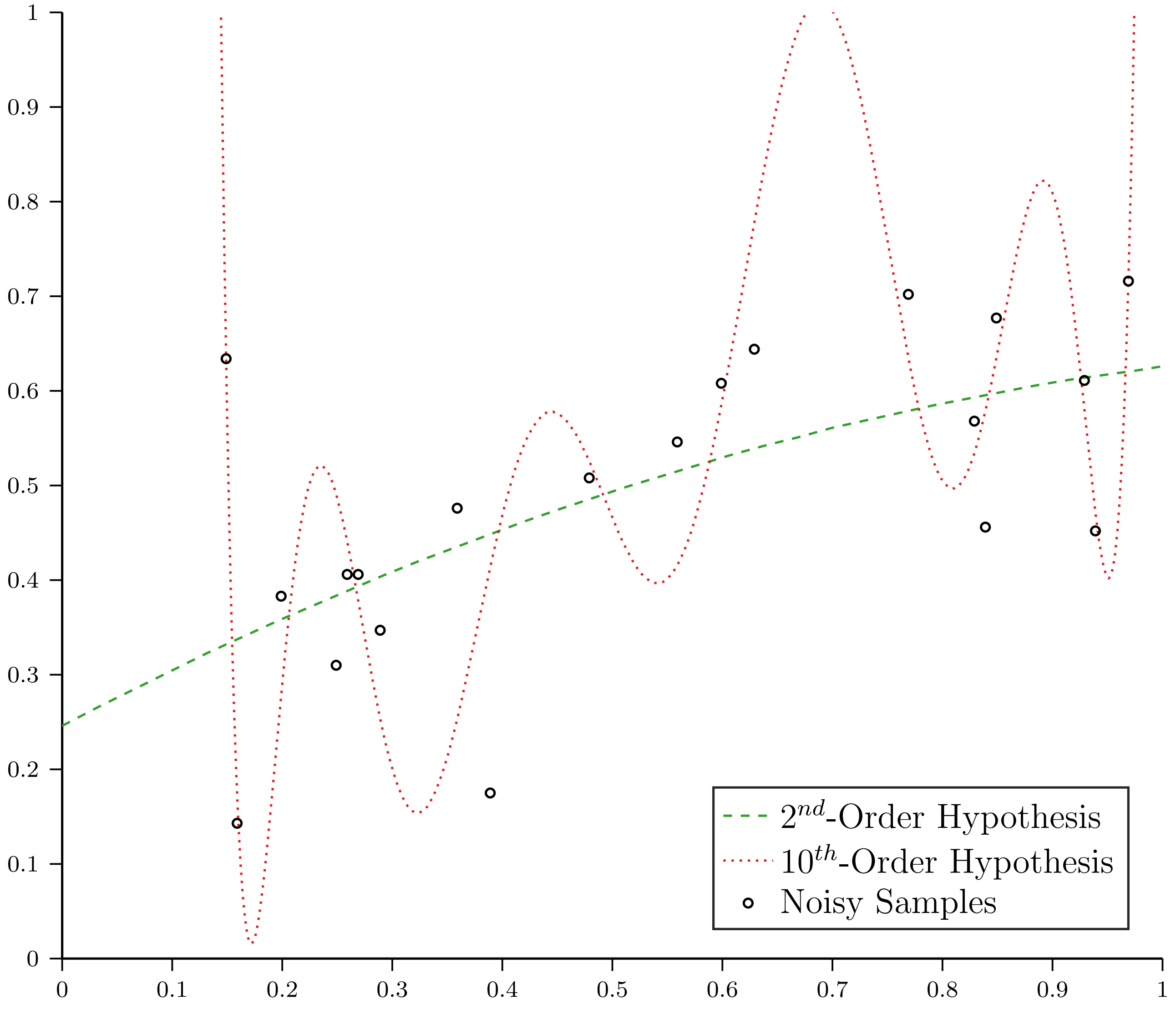
Noisy Targets

- 10-dimensional target function with additive Gaussian noise
- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomial
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomial



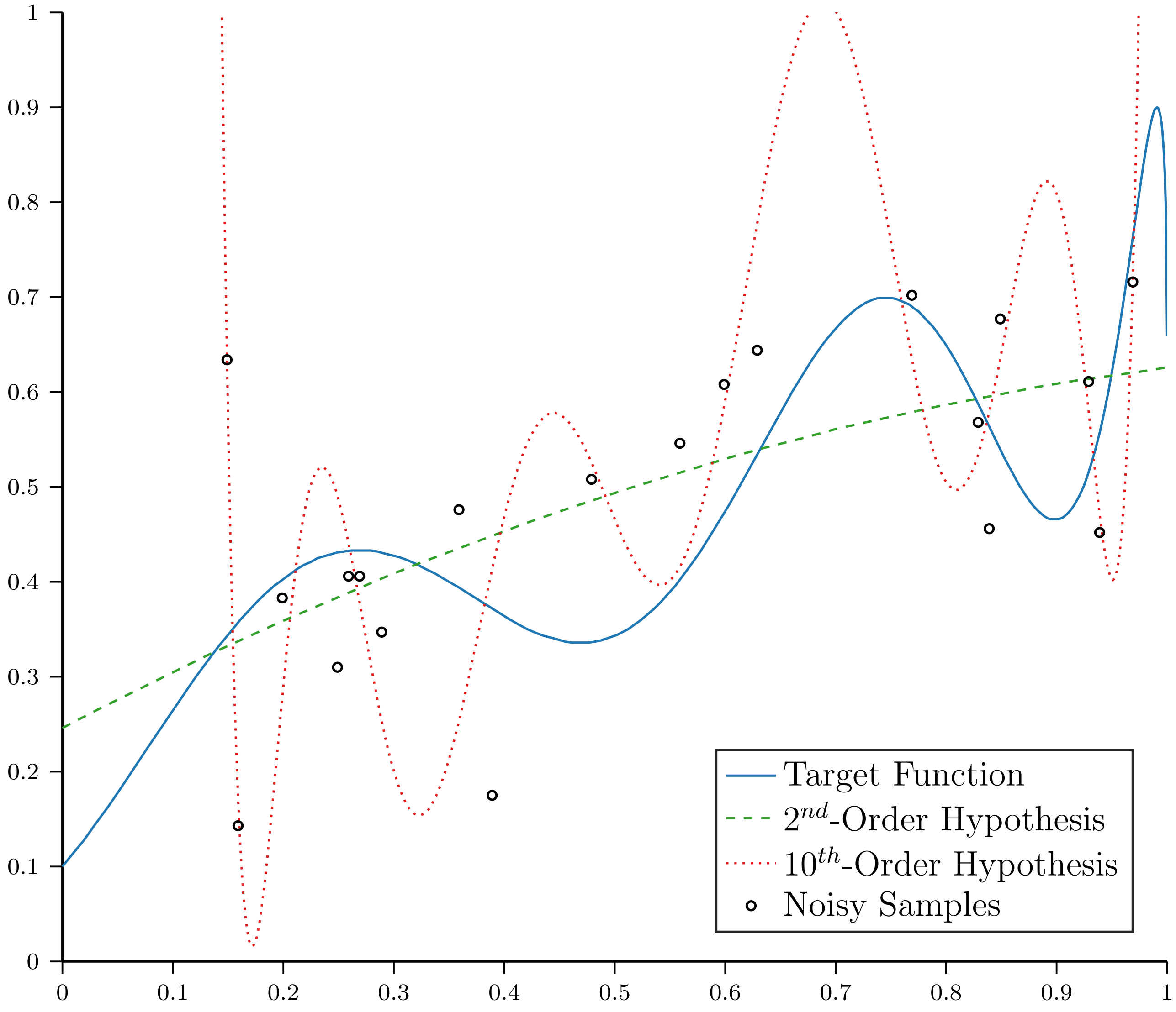
Noisy Targets

- 10-dimensional target function with additive Gaussian noise
- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomial
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomial



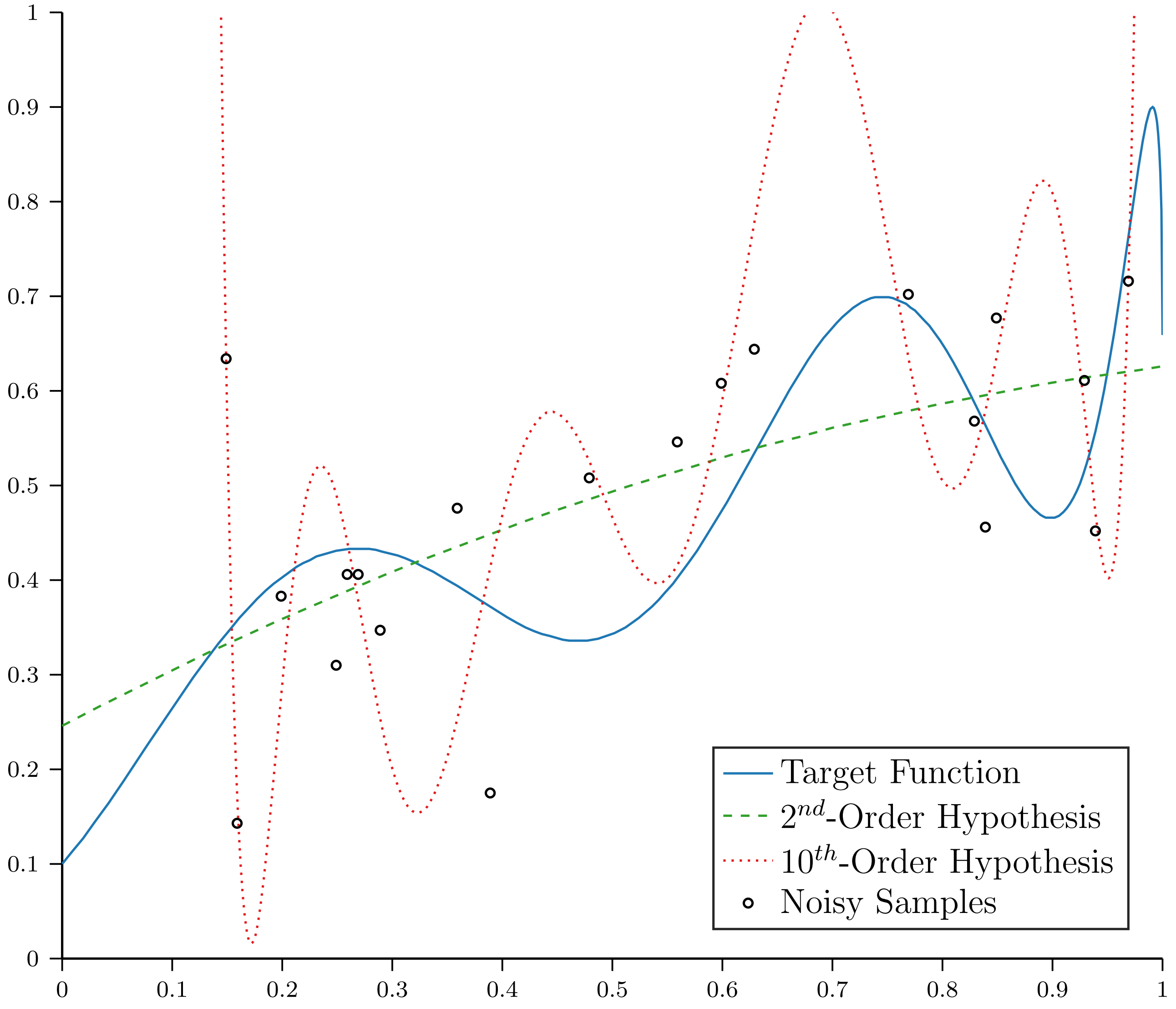
Noisy Targets

- 10-dimensional target function with additive Gaussian noise
- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomial
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomial



Noisy Targets

	$\mathcal{F}_2$	$\mathcal{F}_{10}$
Training Error	0.016	0.011
True Error	0.009	3797



Feature Transforms: Experiment

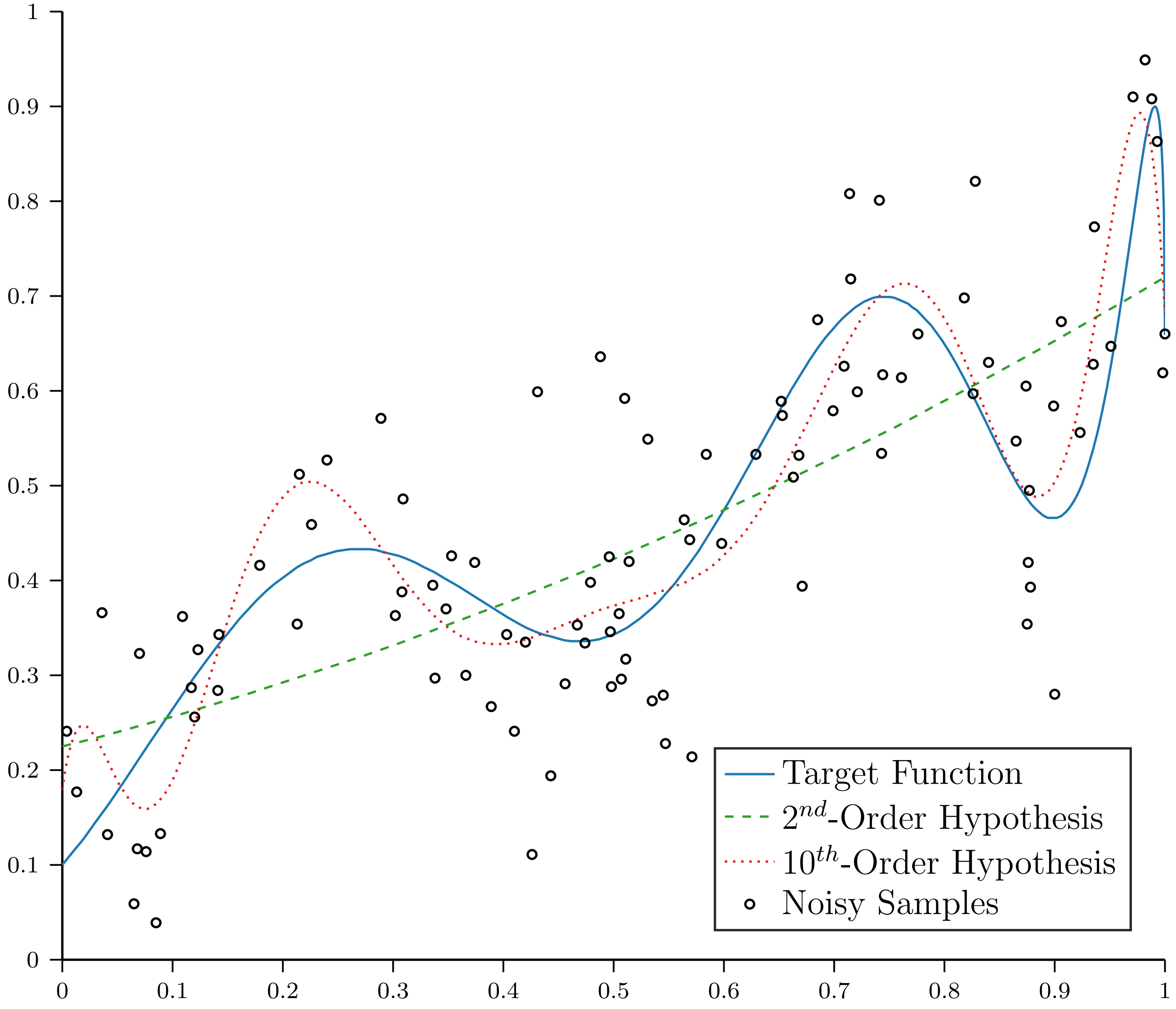
- $X \in \mathbb{R}$, $Y \in \mathbb{R}$ and $n = 100$
- Targets are generated by a 10^{th}-order polynomial in X with additive Gaussian noise:

$$Y = \sum_{d=0}^{10} a_d X^d + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2)$$

- $\mathcal{F}_2 = 2^{\text{nd}}$-order polynomials
  - $\phi_{1,2}(X) = [X, X^2]$
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$

Noisy Targets

	$\mathcal{F}_2$	$\mathcal{F}_{10}$
Training Error	0.018	0.010
True Error	0.009	0.003



Regularization

- Constrain models to prevent them from overfitting
- Learning algorithms are optimization problems and regularization imposes constraints on the optimization

Hard Constraints

- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$
- Given $\boldsymbol{A} = \begin{bmatrix} 1 & \phi_{1,2}(X_1) \\ 1 & \phi_{1,2}(X_2) \\ \vdots & \vdots \\ 1 & \phi_{1,2}(X_n) \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$ find

  $\beta = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}]$ that minimizes

$$\frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$$

- Subject to

$$\beta_3 = \beta_4 = \beta_5 = \beta_6 = \beta_7 = \beta_8 = \beta_9 = \beta_{10} = 0$$

Hard Constraints

- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$
- Given $\boldsymbol{A} = \begin{bmatrix} 1 & \phi_{1,2}(X_1) \\ 1 & \phi_{1,2}(X_2) \\ \vdots & \vdots \\ 1 & \phi_{1,2}(X_n) \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$ find

  $\beta = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}]$ that minimizes

$$\frac{1}{n}\sum_{i=1}^{n}\left(\left(\sum_{d=0}^{10} X_i^d \beta_d\right) - Y_i\right)^2$$

- Subject to

$$\beta_3 = \beta_4 = \beta_5 = \beta_6 = \beta_7 = \beta_8 = \beta_9 = \beta_{10} = 0$$

Hard Constraints

- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomials
  - $\phi_{1,10}(X) = [X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^{10}]$
- Given $\boldsymbol{A} = \begin{bmatrix} 1 & \phi_{1,2}(X_1) \\ 1 & \phi_{1,2}(X_2) \\ \vdots & \vdots \\ 1 & \phi_{1,2}(X_n) \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$ find $\beta = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}]$ that minimizes

$$\frac{1}{n}\sum_{i=1}^{n}\left(\left(\sum_{d=0}^{2} X_i^d \beta_d\right) - Y_i\right)^2$$

- Subject to nothing!

Hard Constraints

- $\mathcal{F}_2$ = 2^{nd}-order polynomials
  - $\phi_{1,2}(X) = [X, X^2]$
- Given $\boldsymbol{A} = \begin{bmatrix} 1 & \phi_{1,2}(X_1) \\ 1 & \phi_{1,2}(X_2) \\ \vdots & \vdots \\ 1 & \phi_{1,2}(X_n) \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$ find $\beta = [\beta_0, \beta_1, \beta_2]$ that minimizes

$$\frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$$

- Subject to nothing!

Soft Constraints

- More generally, ϕ can be any nonlinear transformation, e.g., exp, log, sin, sqrt, etc...
- Given $\boldsymbol{A} = \begin{bmatrix} 1 & \phi_1(X_1) & \cdots & \phi_m(X_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(X_n) & \cdots & \phi_m(X_1) \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$, find β that minimizes

$$\frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$$

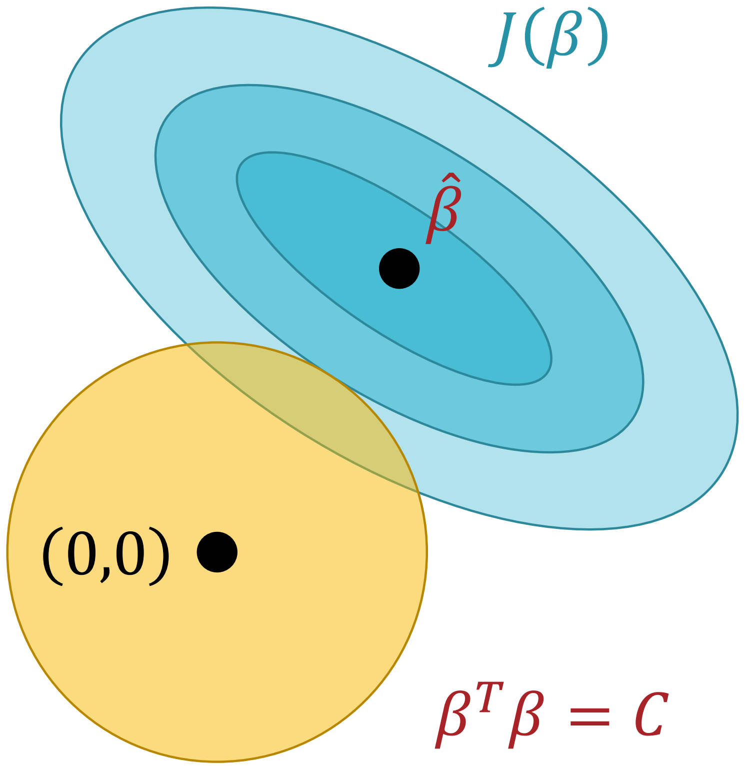
- Subject to:

$$\|\beta\|_2^2 = \beta^T\beta = \sum_{d=0}^{m} \beta_d^2 \leq C$$

Soft Constraints

minimize $J(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$

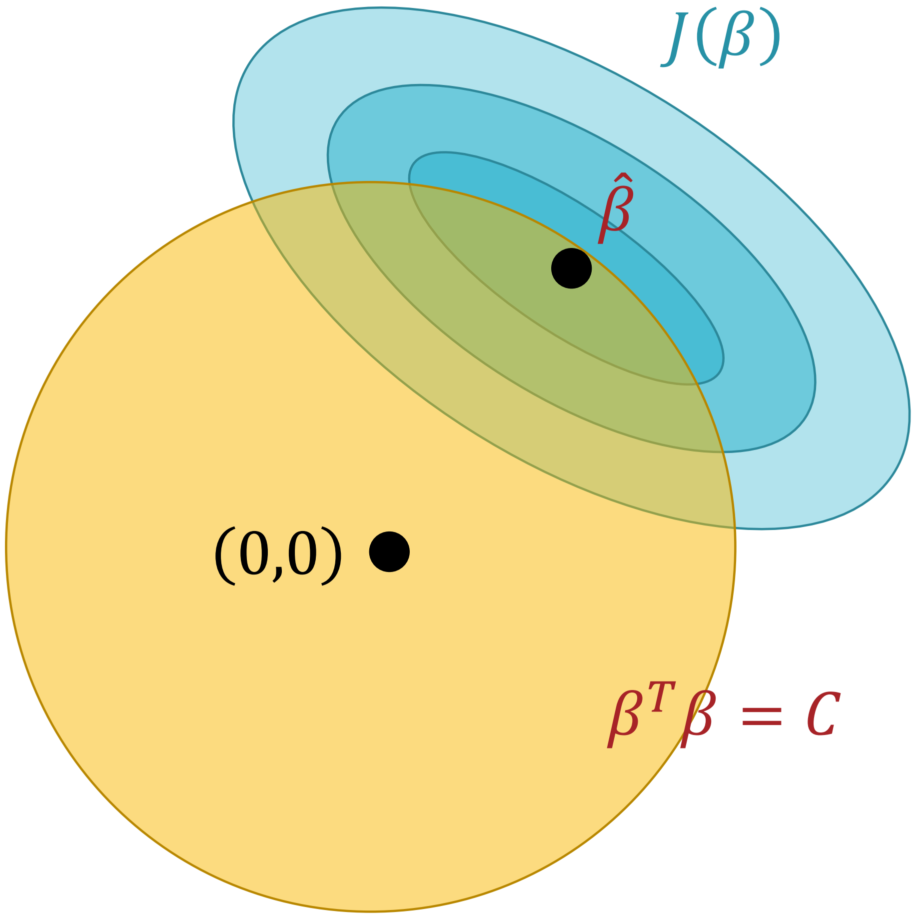
subject to $\beta^T\beta \leq C$



Soft Constraints

minimize $J(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$

subject to $\beta^T\beta \leq C$



Soft Constraints

minimize $J(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$

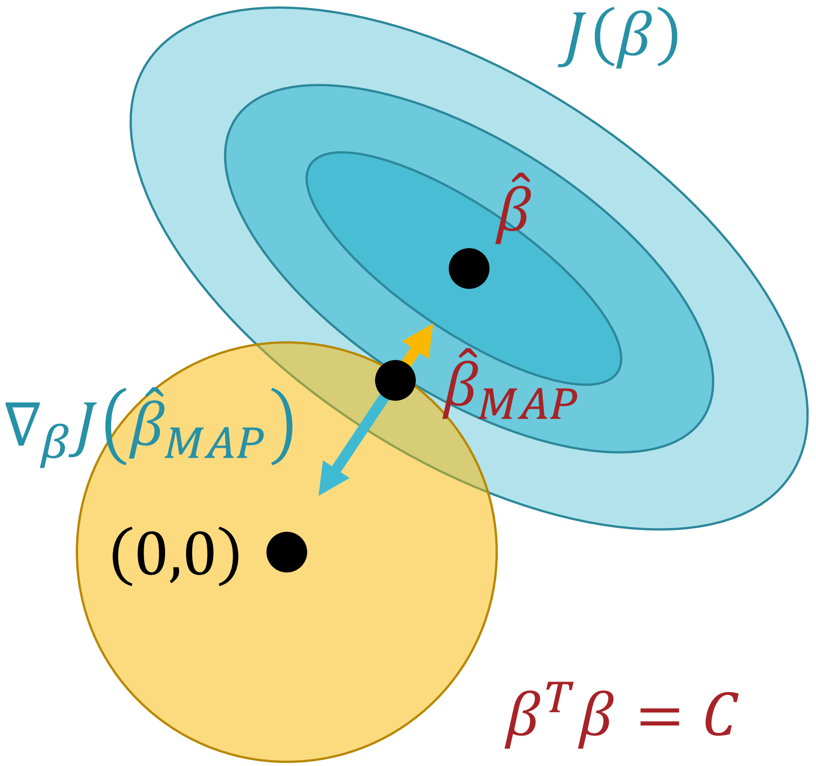
subject to $\beta^T\beta \leq C$

$$\nabla_\beta J(\hat{\beta}_{MAP}) \propto -\frac{2}{n}\hat{\beta}_{MAP}$$

$$\nabla_\beta J(\hat{\beta}_{MAP}) = -\frac{2}{n}\lambda_C\hat{\beta}_{MAP}$$

$$\nabla_\beta J(\hat{\beta}_{MAP}) + \frac{2}{n}\lambda_C\hat{\beta}_{MAP} = 0$$

$$\nabla_\beta \left(J(\hat{\beta}_{MAP}) + \frac{\lambda_C}{n}(\hat{\beta}_{MAP})^T\hat{\beta}_{MAP}\right) = 0$$



Soft Constraints: Solving for $\hat{\beta}_{MAP}$

minimize $J(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})$

subject to $\beta^T\beta \leq C$

$$\Updownarrow$$

minimize $J_{AUG}(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y}) + \frac{\lambda_C}{n}\beta^T\beta$

Ridge Regression

$$\text{minimize } J_{AUG}(\beta) = \frac{1}{n}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y}) + \frac{\lambda_C}{n}\beta^T\beta$$

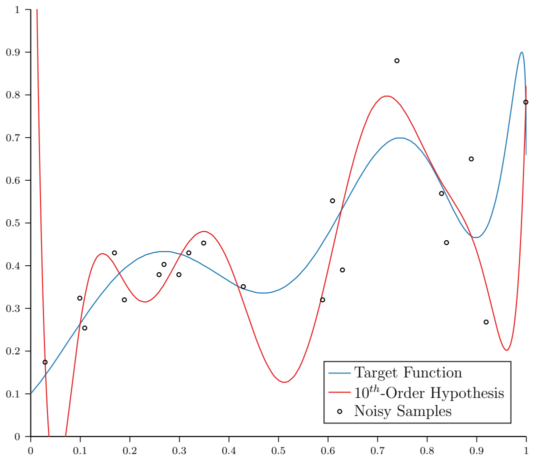
$$\nabla_\beta J_{AUG}(\beta) = \frac{2}{n}(\boldsymbol{A}^T\boldsymbol{A}\beta - \boldsymbol{A}^T\boldsymbol{Y} + \lambda_C\beta)$$

$$\frac{2}{n}\left(\boldsymbol{A}^T\boldsymbol{A}\hat{\beta}_{MAP} - \boldsymbol{A}^T\boldsymbol{Y} + \lambda_C\hat{\beta}_{MAP}\right) = 0$$

$$(\boldsymbol{A}^T\boldsymbol{A} + \lambda_C I_{m+1})\hat{\beta}_{MAP} = \boldsymbol{A}^T\boldsymbol{Y}$$

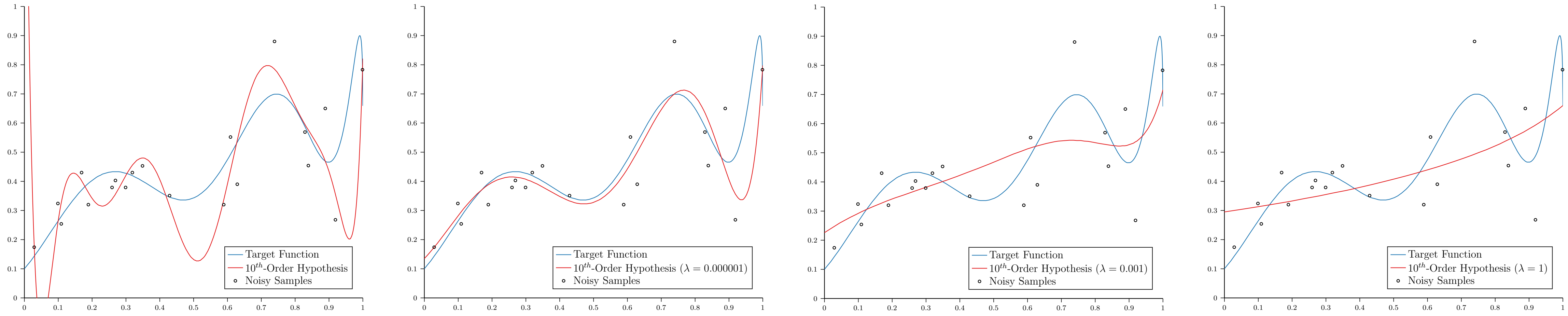
$$\hat{\beta}_{MAP} = (\boldsymbol{A}^T\boldsymbol{A} + \underbrace{\lambda_C I_{m+1}})^{-1}\boldsymbol{A}^T\boldsymbol{Y}$$

Adding this positive ($\lambda_C \geq 0$) diagonal matrix can help if $\boldsymbol{A}^T\boldsymbol{A}$ is under-determined!



Ridge Regression

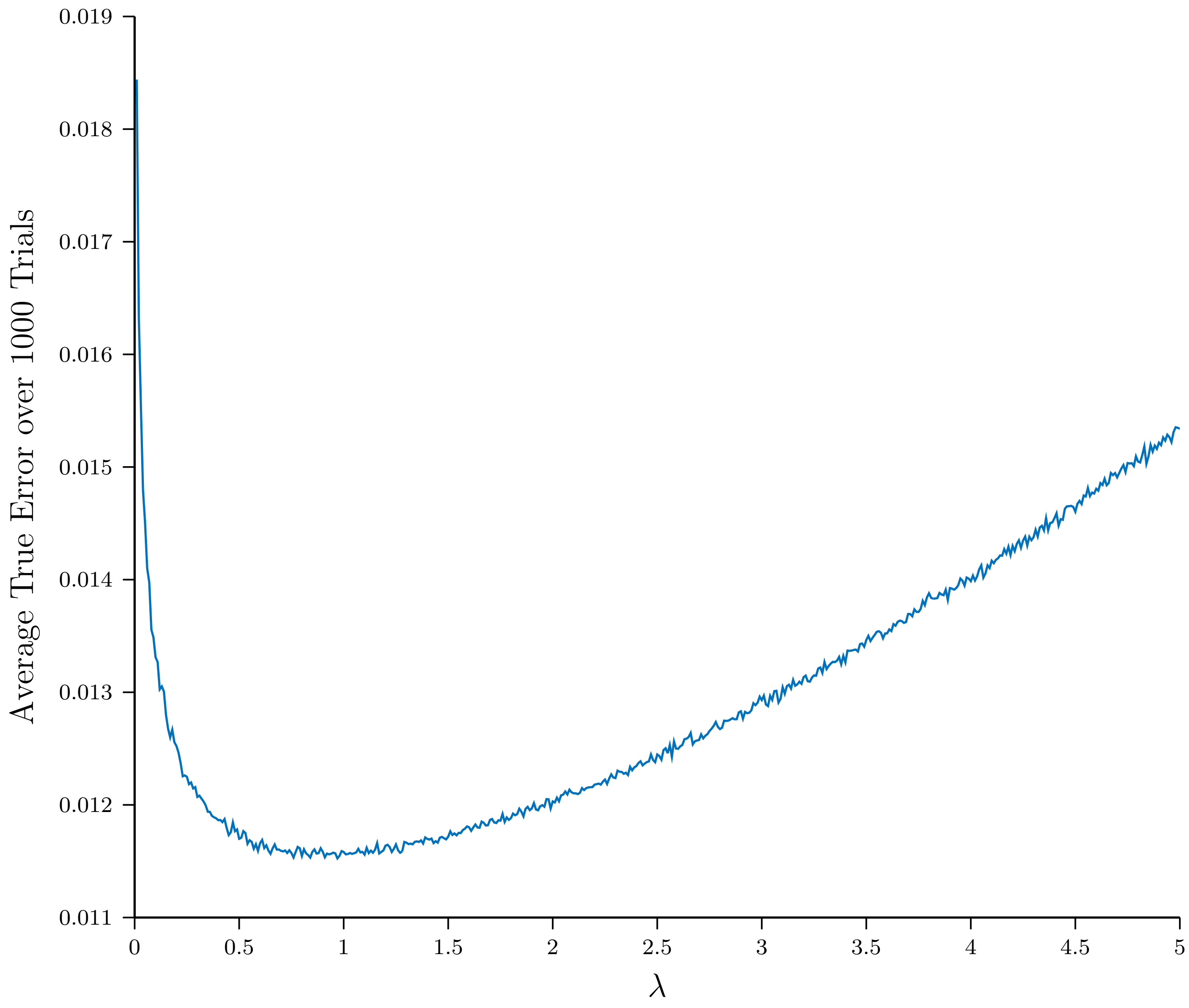
- 10-dimensional target function with additive Gaussian noise
- $\mathcal{F}_{10} = 10^{\text{th}}$-order polynomial



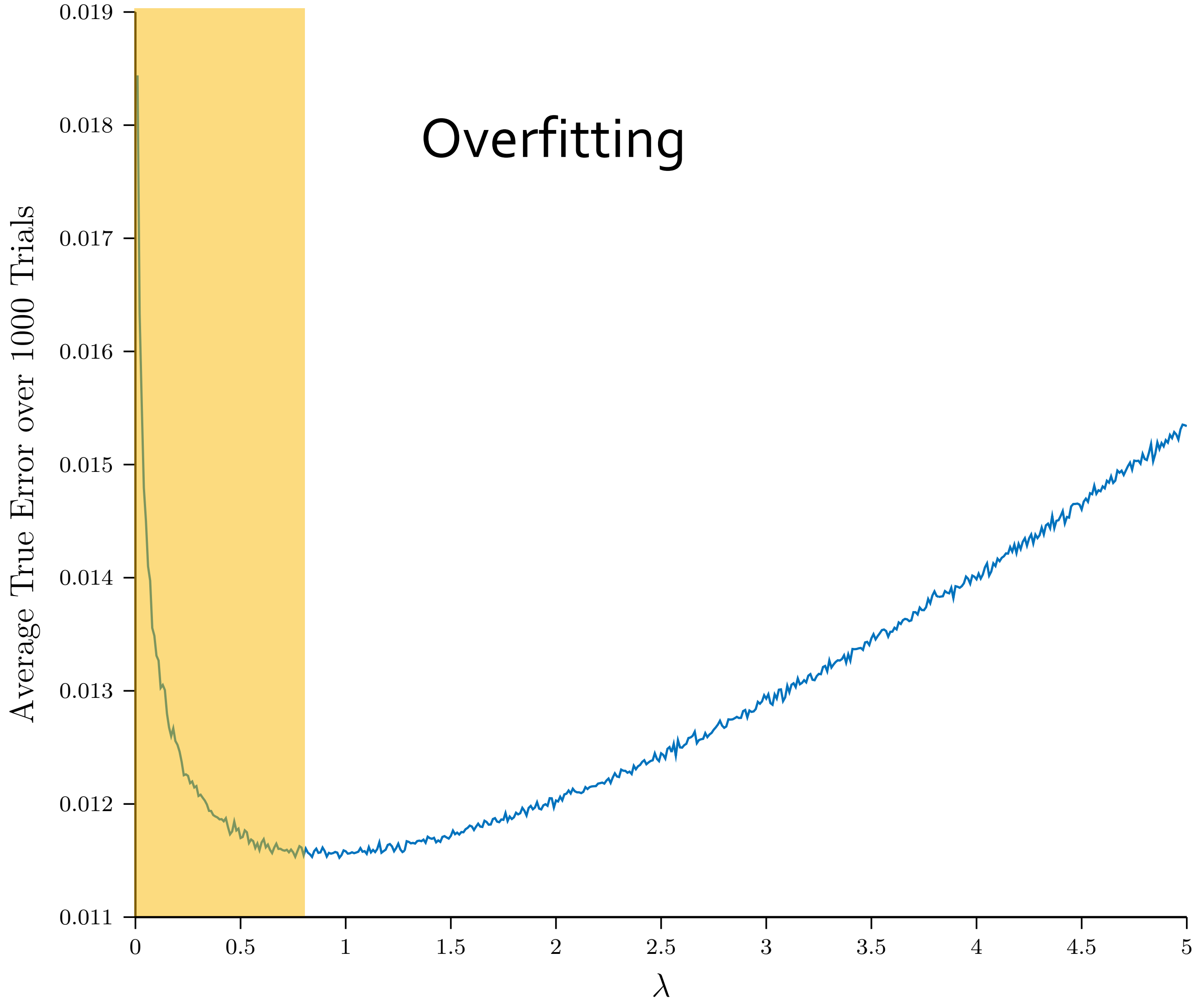
Ridge Regression

	$\lambda_C = 0$
True Error	0.059
	Overfit

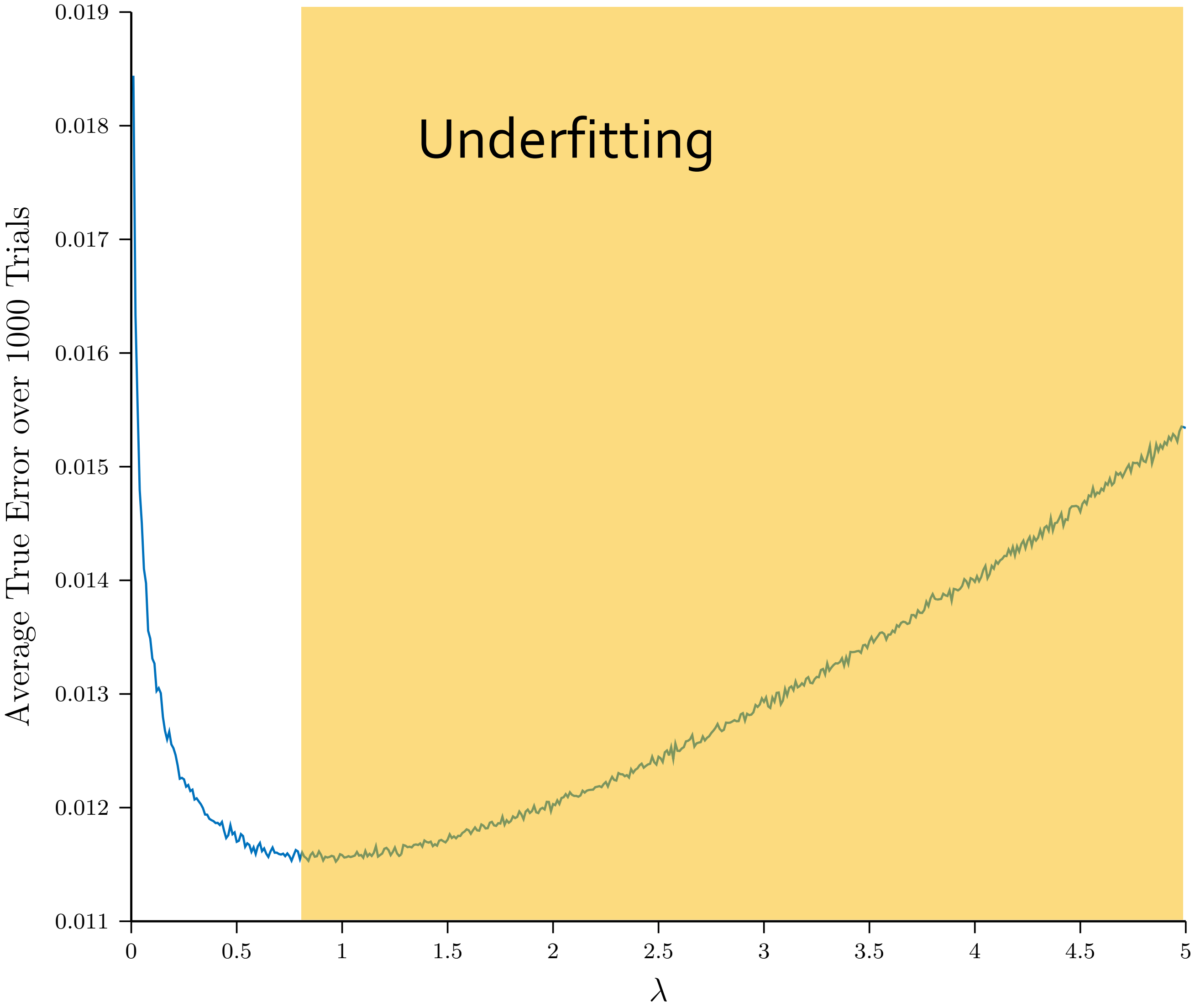
Setting λ



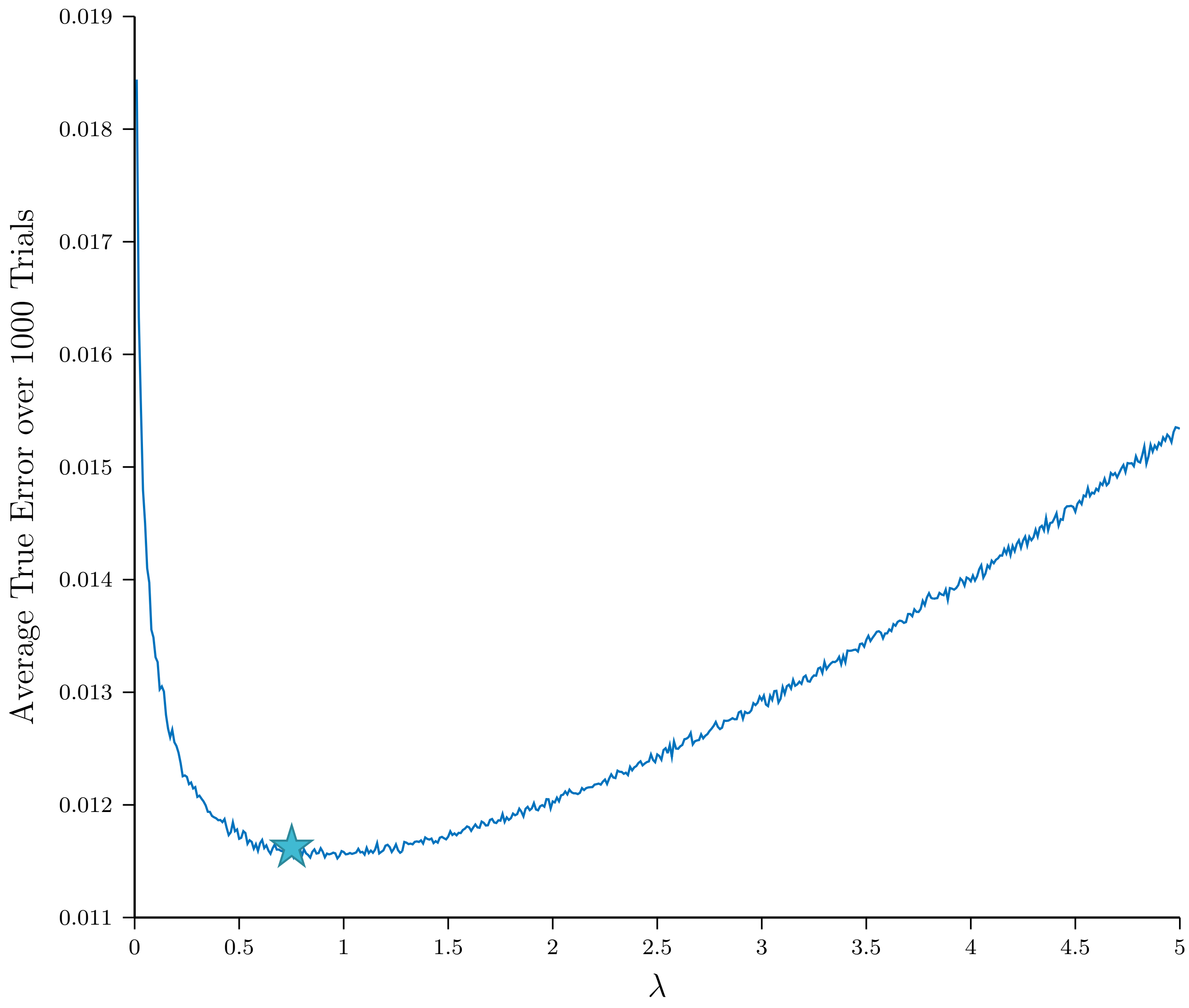
Setting λ



Setting λ

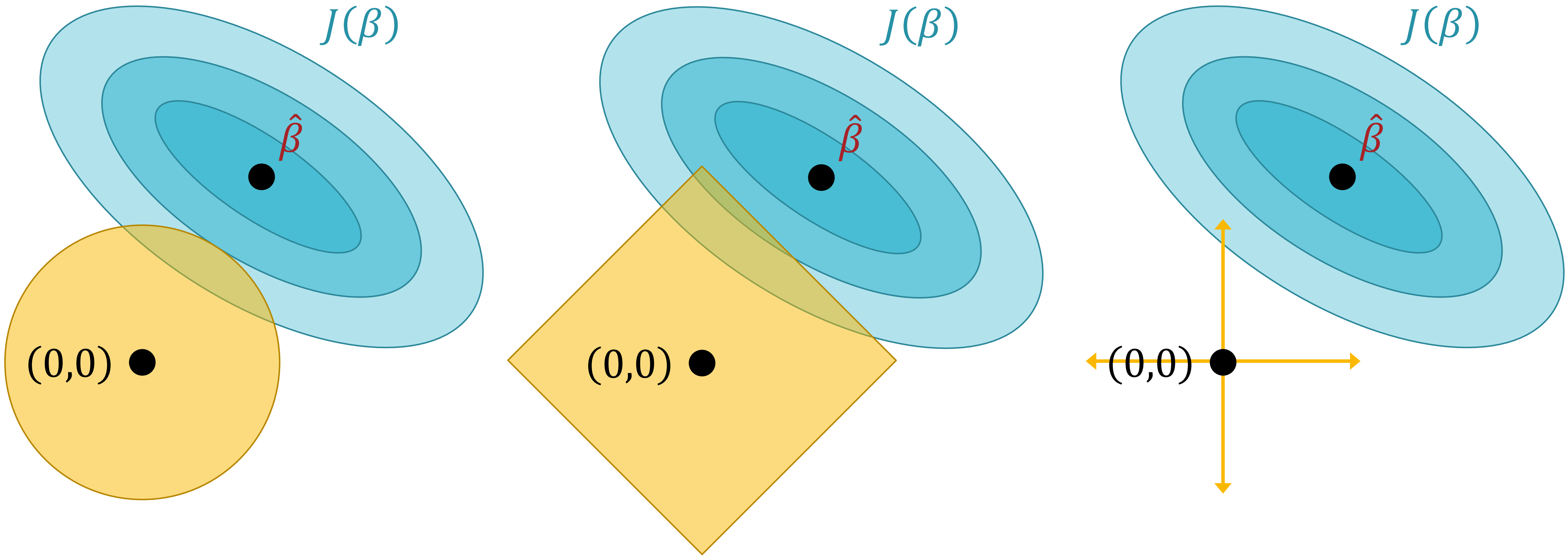


Setting λ



Other Regularizers

$J(\beta) + \lambda\, pen(\beta)$		
Ridge or $L2$	$pen(\beta) = \|\beta\|_2^2 = \sum_{d=0}^{p} \beta_d^2$	Encourages small weights
Lasso or $L1$	$pen(\beta) = \|\beta\|_1 = \sum_{d=0}^{p} \lvert\beta_d\rvert$	Encourages sparsity
$L0$	$pen(\beta) = \|\beta\|_0 = \sum_{d=0}^{p} \mathbb{1}(\beta_d \neq 0)$	Encourages sparsity (intractable)



Ridge or $L2$

Lasso or $L1$

$L0$

Other Regularizers

M(C)LE for Linear Regression

- If we assume a linear model with additive Gaussian noise

$$Y = X\beta + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2) \rightarrow Y \sim N(X\beta, \sigma^2)$$

- Then given $\boldsymbol{A} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$ the MLE of β is

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \log P(Y|A, \beta)$$

$$= \underset{\beta}{\operatorname{argmax}} \log \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y})\right)$$

$$= \underset{\beta}{\operatorname{argmin}} (\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y}) = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Y}$$

MAP for Linear Regression

- If we assume a linear model with additive Gaussian noise

$$Y = X\beta + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2) \rightarrow Y \sim N(X\beta, \sigma^2)$$

and a Gaussian prior on the weights…

$$\beta \sim N\left(0, \frac{\sigma^2}{\lambda}\right) \rightarrow p(\beta) \propto \exp\left(-\frac{1}{2\sigma^2}(\lambda\beta^T\beta)\right)$$

- … then, the MAP of β is the ridge regression solution!

$$\hat{\beta}_{MAP} = \underset{\beta}{\operatorname{argmin}}\, (\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y}) + \lambda\beta^T\beta$$

$$= \left(\boldsymbol{A}^T\boldsymbol{A} + \lambda I_{p+1}\right)^{-1}\boldsymbol{A}^T\boldsymbol{Y}$$

MAP for Linear Regression

- If we assume a linear model with additive Gaussian noise

$$Y = X\beta + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2) \rightarrow Y \sim N(X\beta, \sigma^2)$$

and a Laplace prior on the weights…

$$\beta \sim \text{Laplace}\left(0, \frac{2\sigma^2}{\lambda}\right) \rightarrow p(\beta) \propto \exp\left(-\frac{1}{2\sigma^2}(\lambda \|\beta\|_1)\right)$$

- … then, the MAP of β is the lasso regression solution!

$$\hat{\beta}_{MAP} = \underset{\beta}{\operatorname{argmin}}\, (\boldsymbol{A}\beta - \boldsymbol{Y})^T(\boldsymbol{A}\beta - \boldsymbol{Y}) + \lambda\|\beta\|_1$$

- No closed form solution but can solve via sub-gradient descent