10-701: Introduction to Machine Learning Lecture 5 – Regularization

Henry Chai & Zack Lipton

9/13/23

Front Matter

- Announcements:
  - HW1 released 9/6, due 9/20 at 11:59 PM
  - **Important scheduling note:** Recitation on 9/15 (Friday) has been replaced with instructor OH
- Recommended Readings:
  - Murphy, Section 7.5
  - Murphy, Section 14.4

Recall: Fisher Iris Dataset

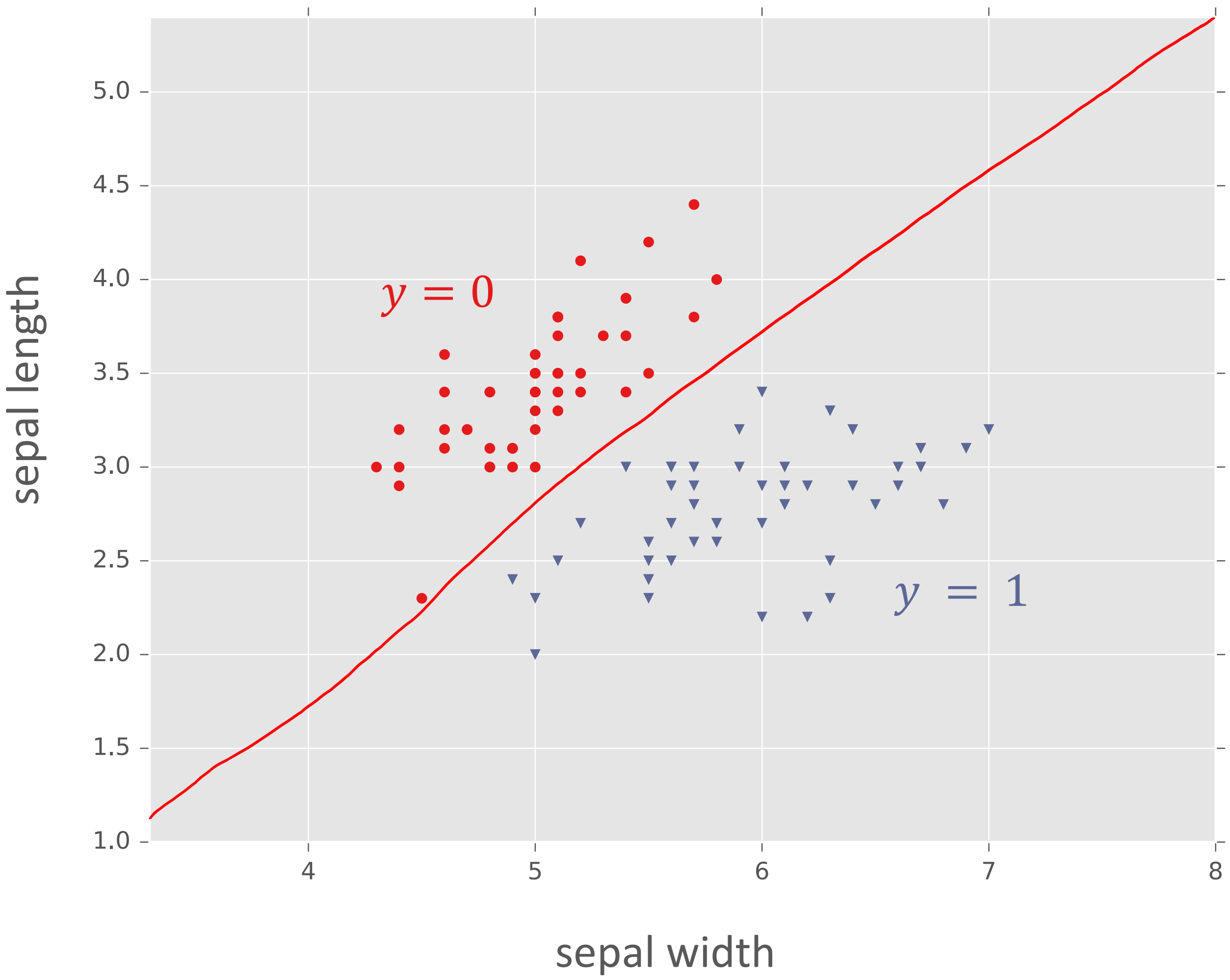
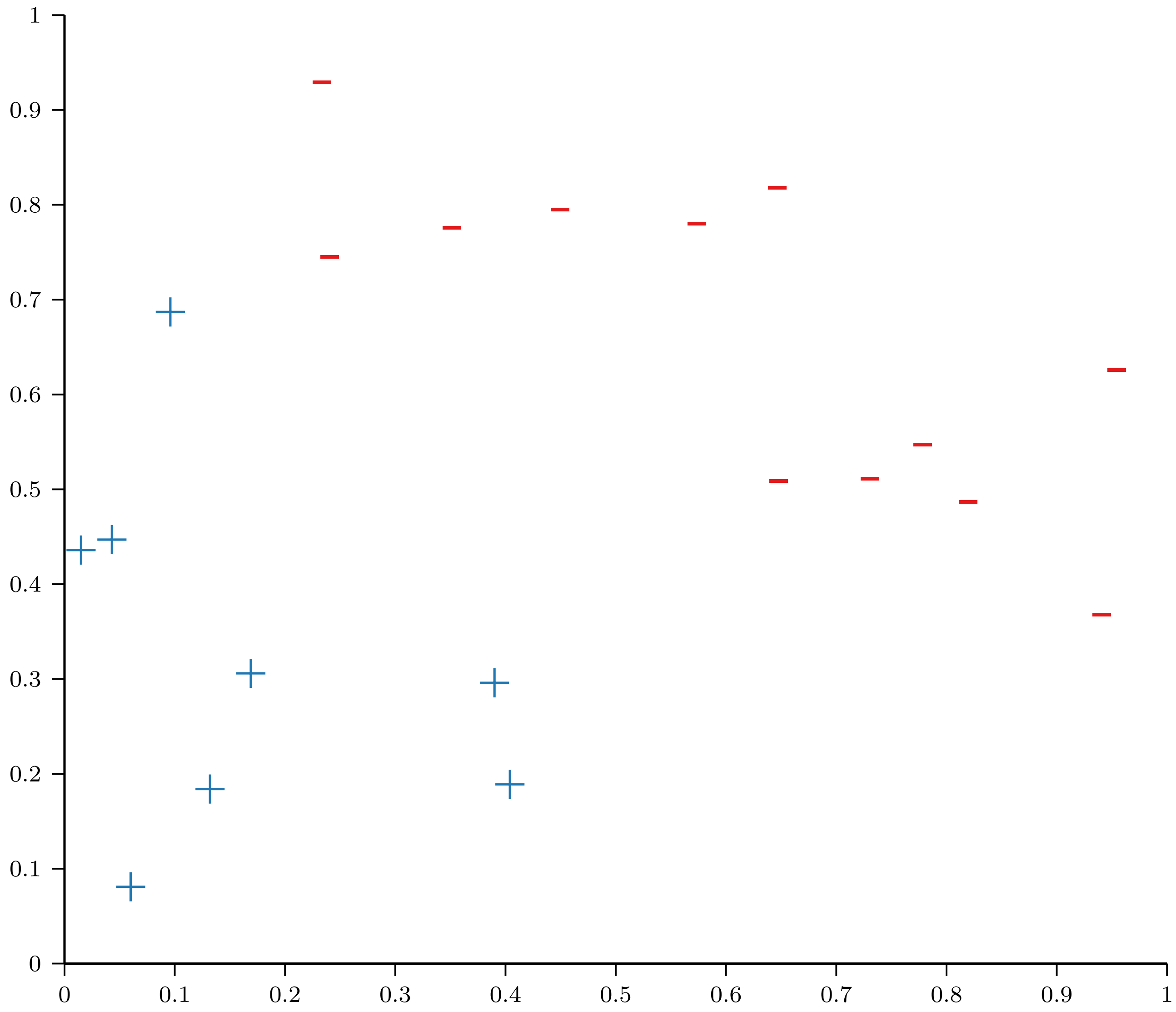
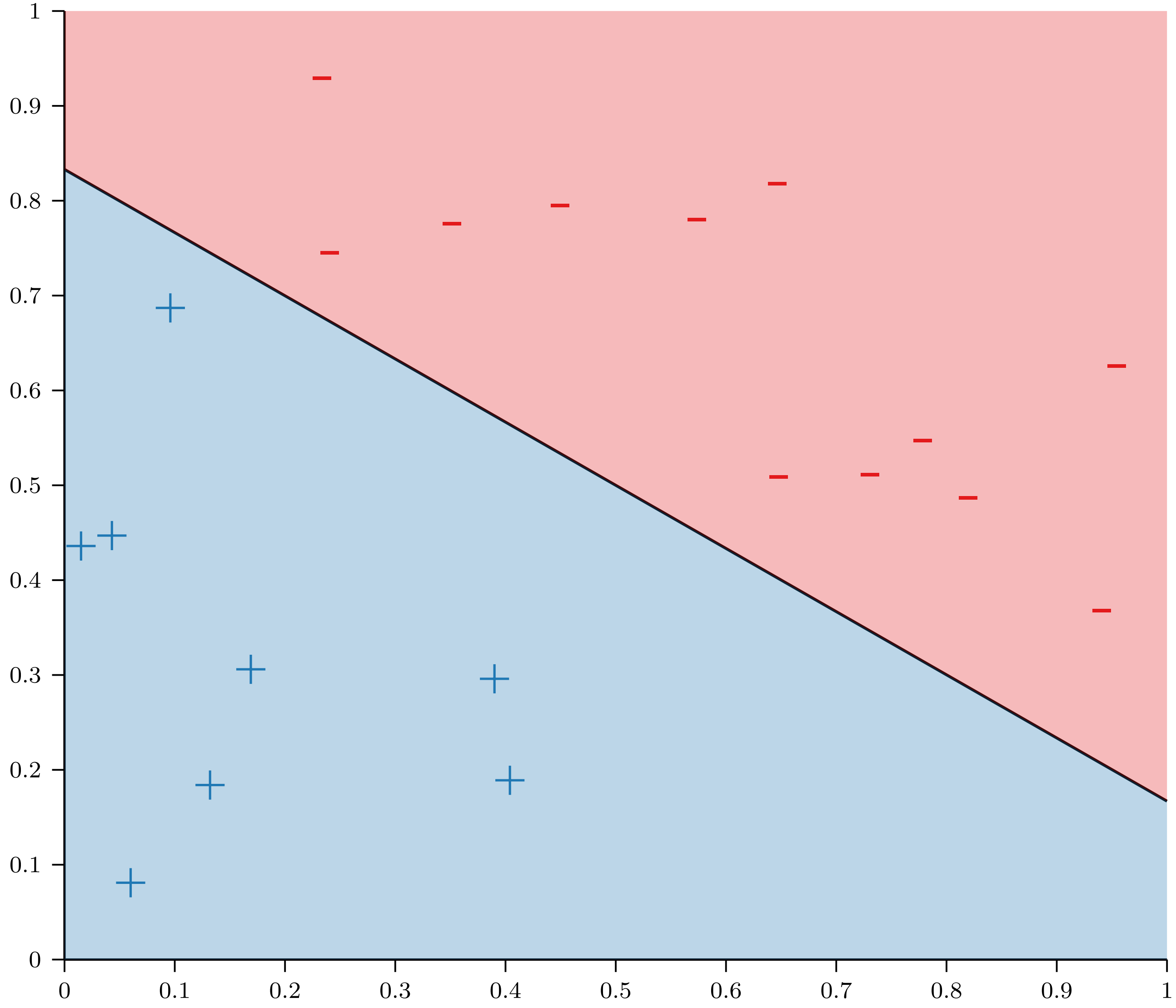


Figure courtesy of Matt Gormley

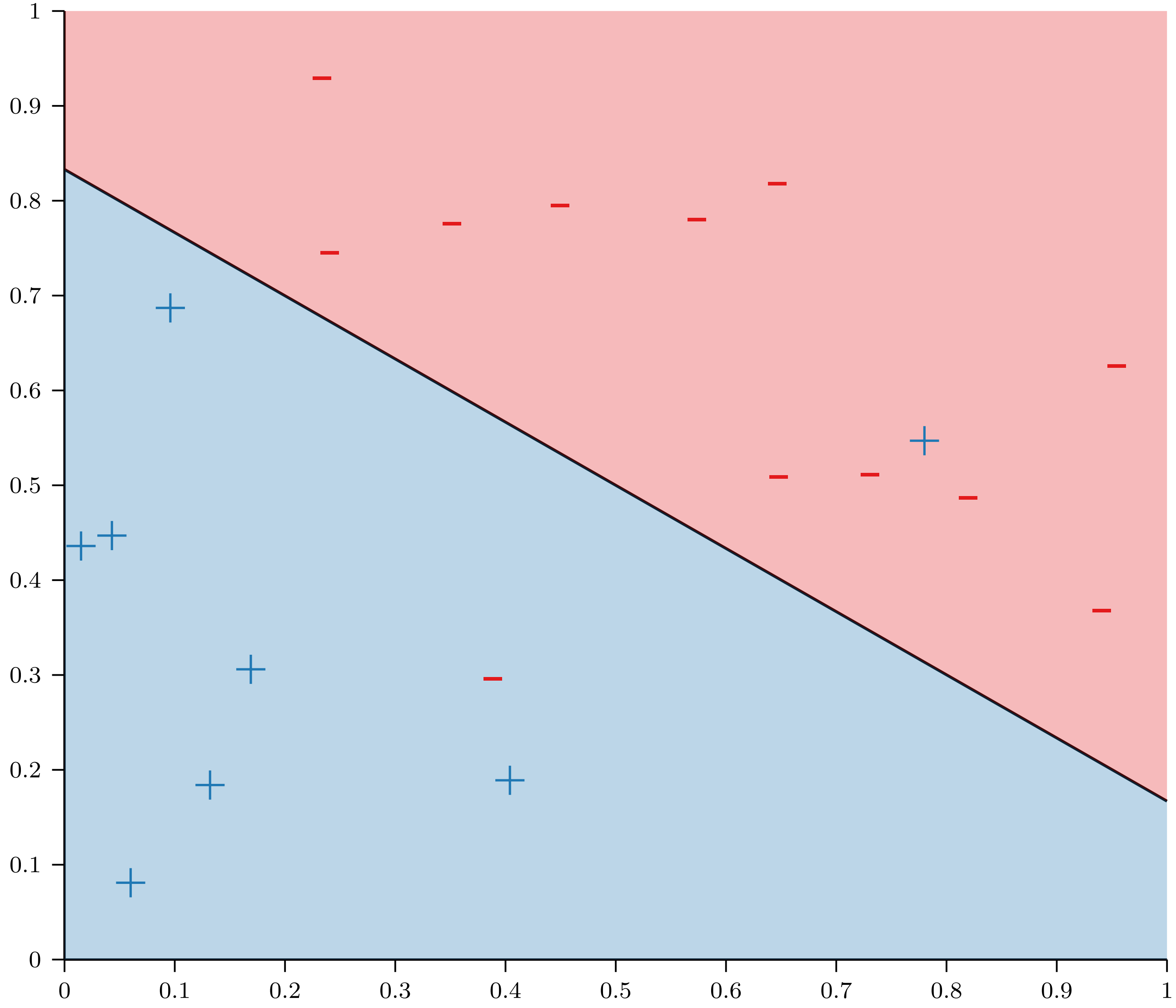
Linear Models



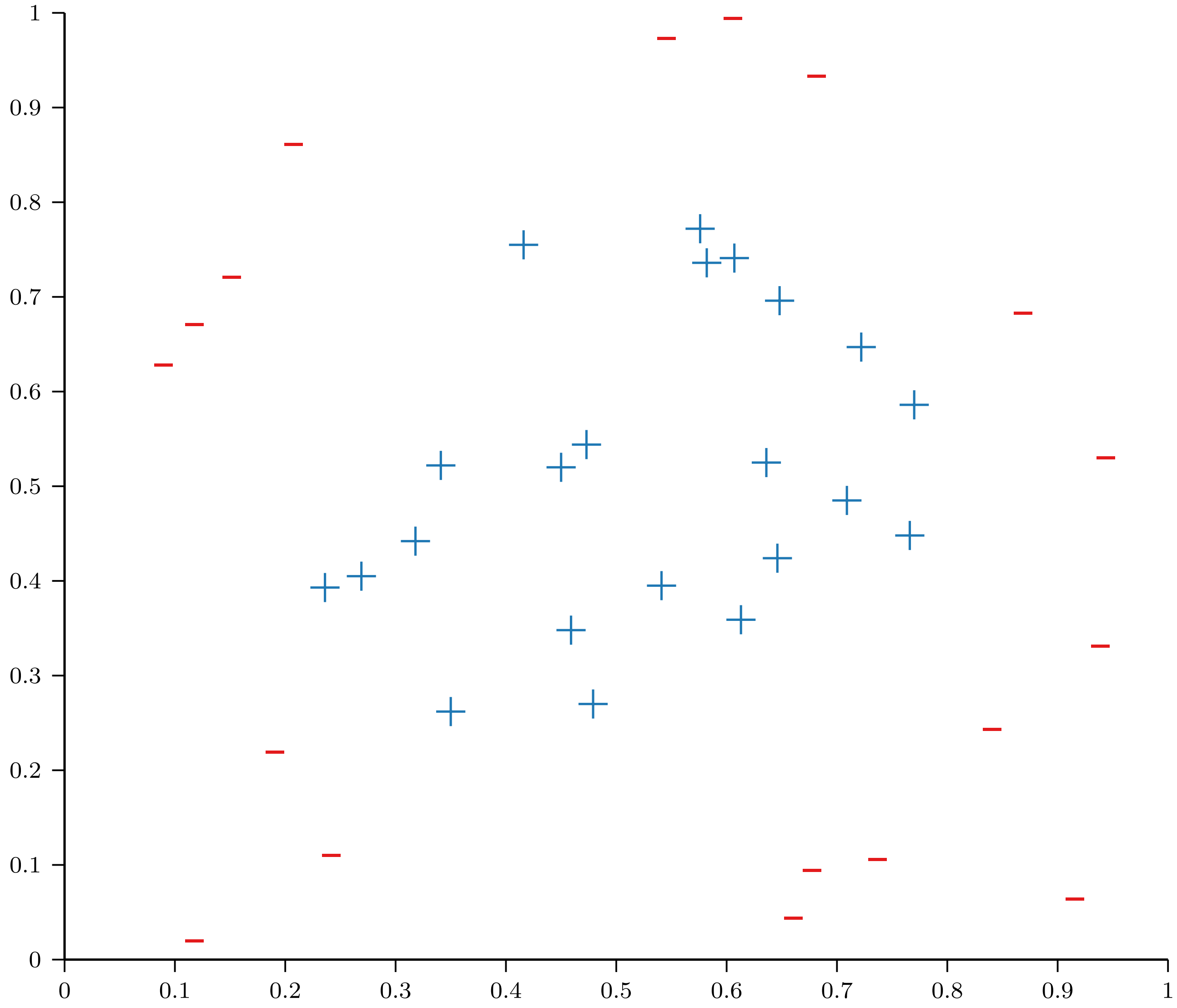
Linear Models



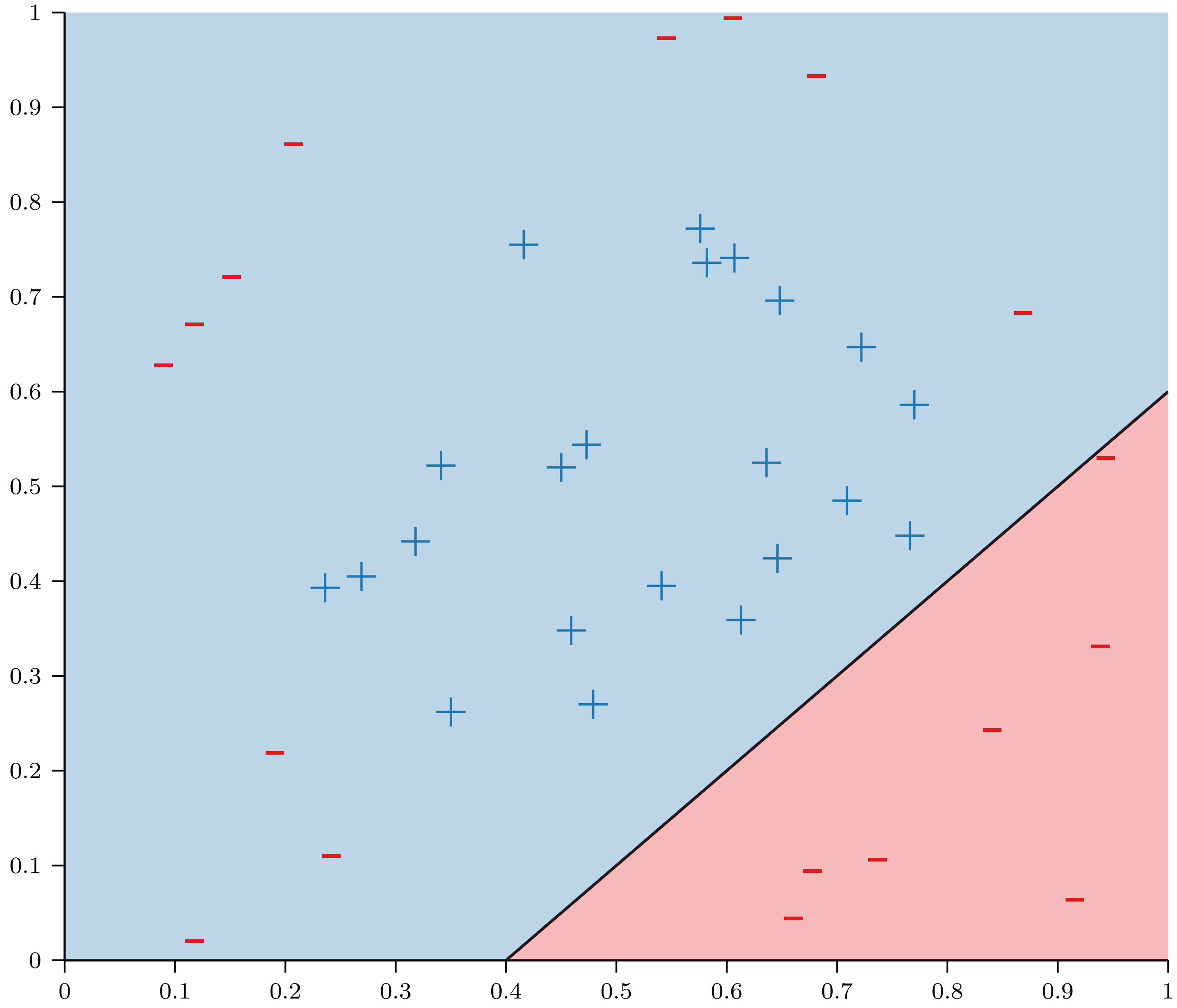
Linear Models



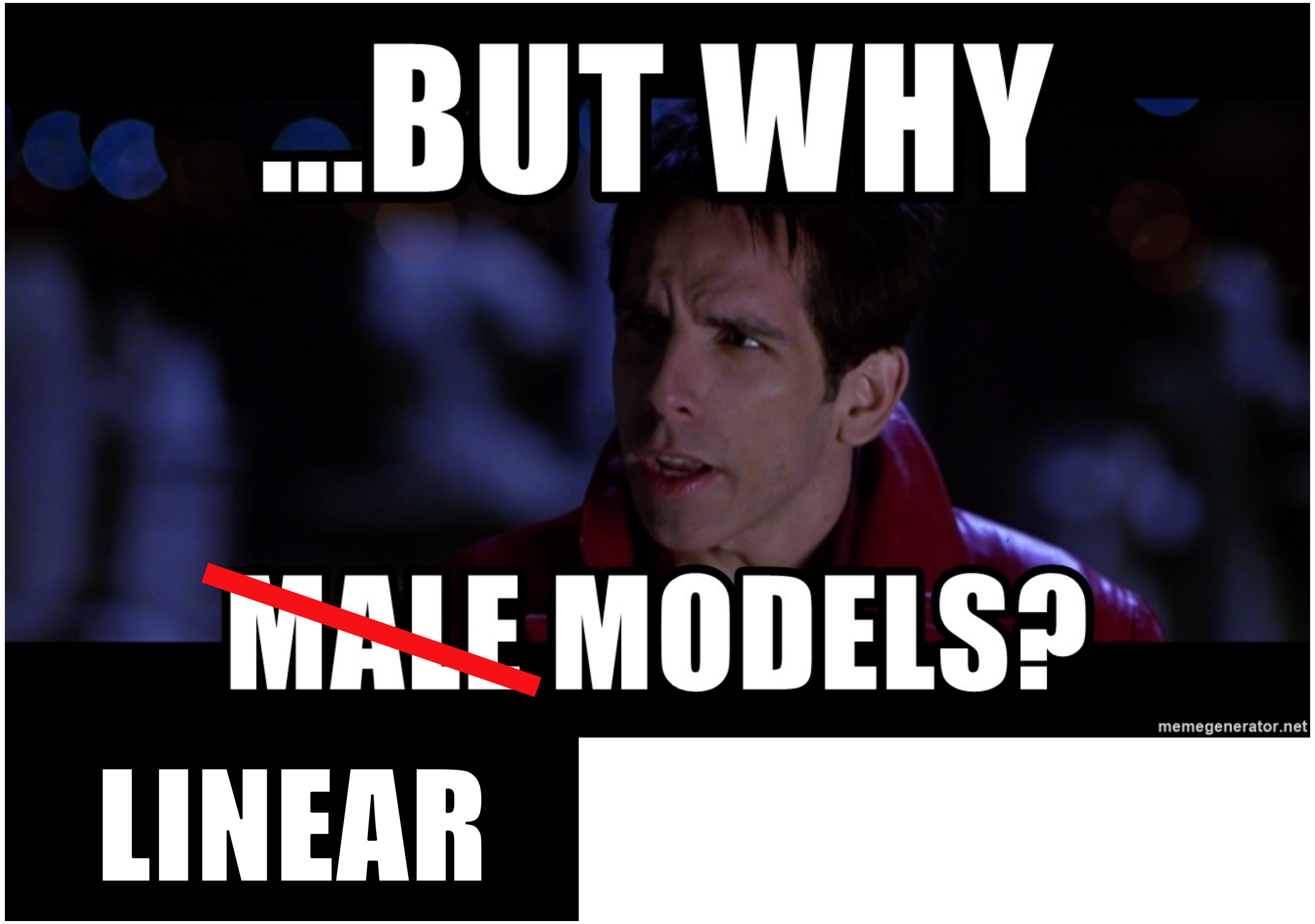
Linear Models?



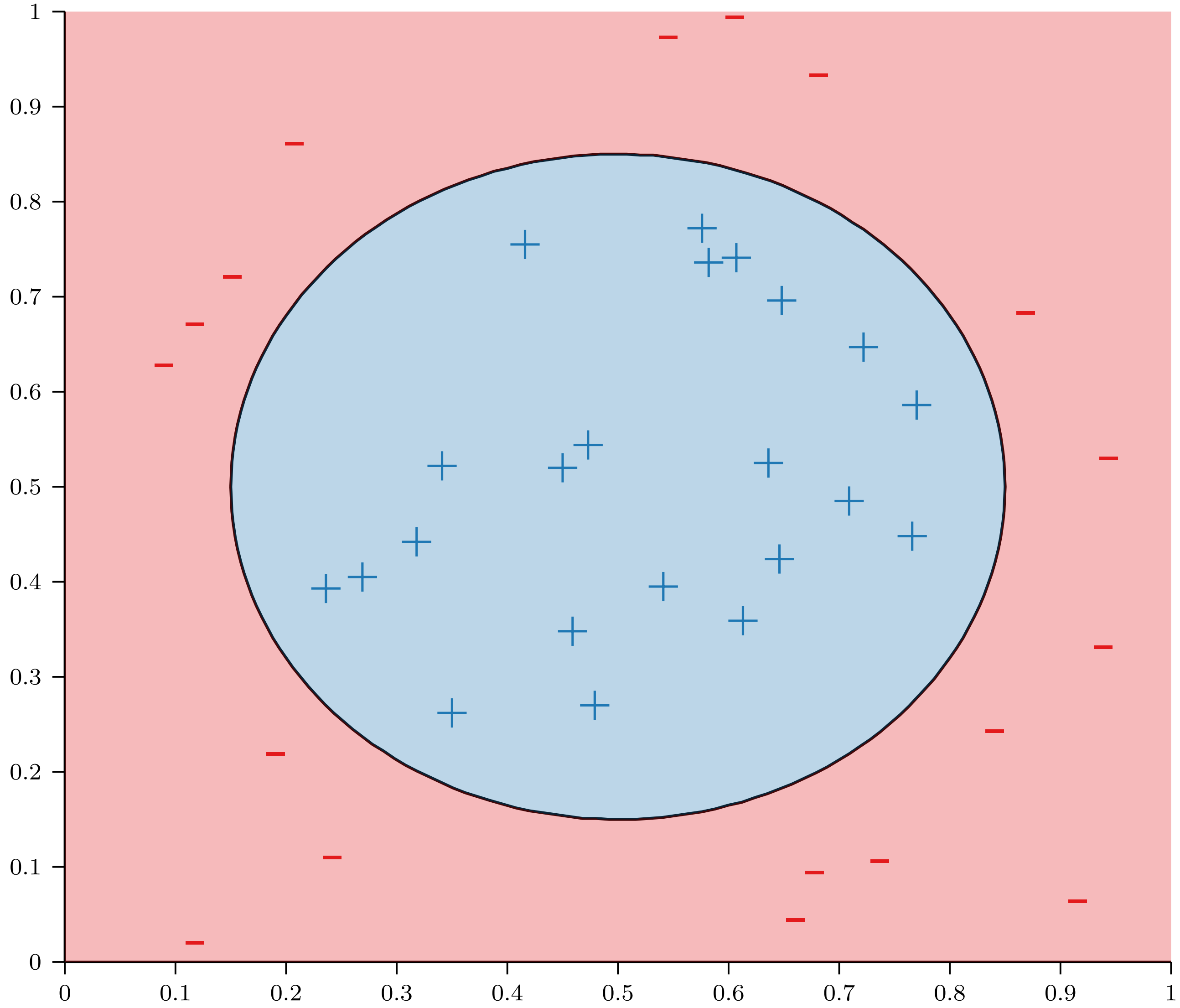
Linear Models?



Linear Models?



Nonlinear Models

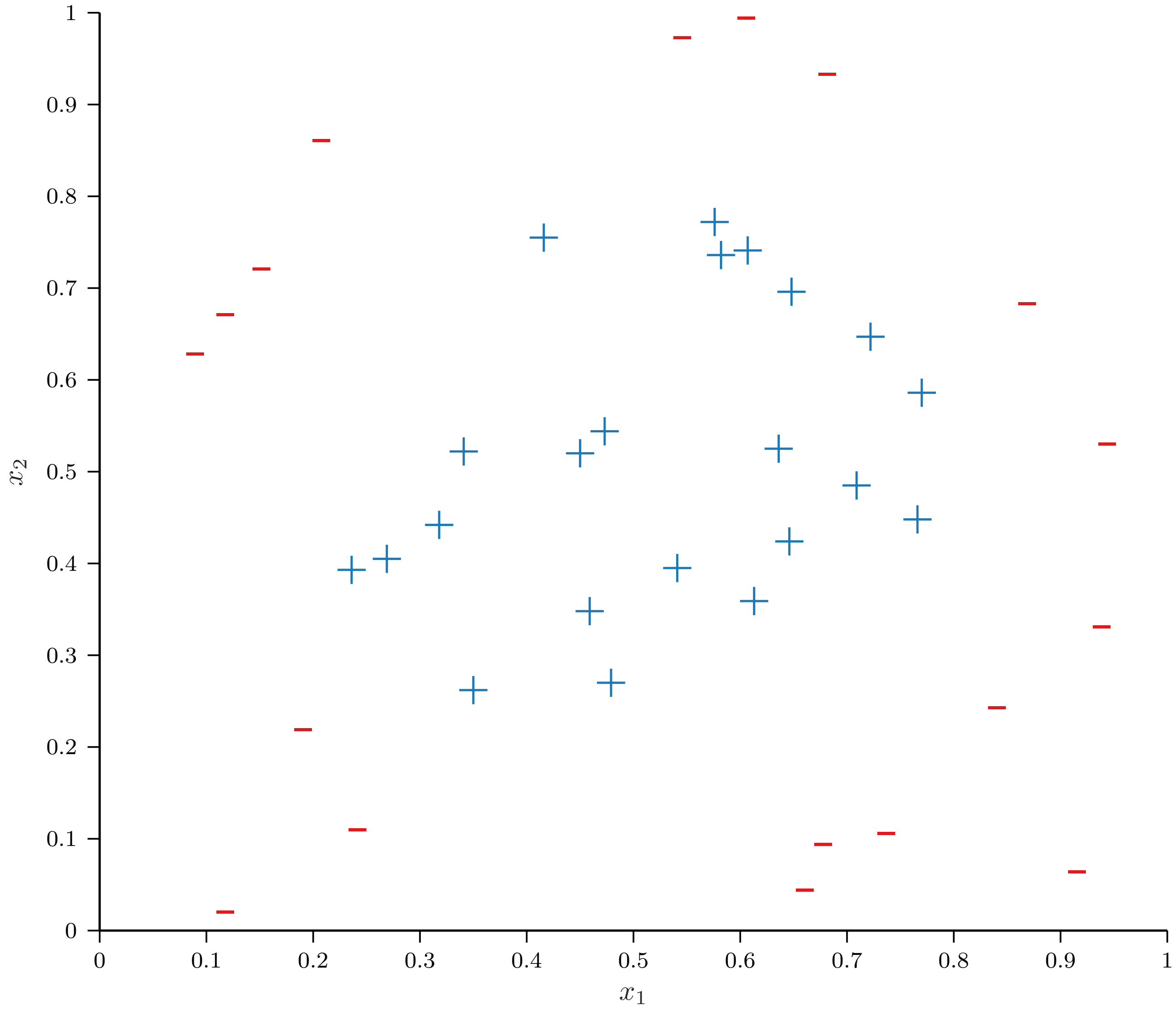


Feature Transforms

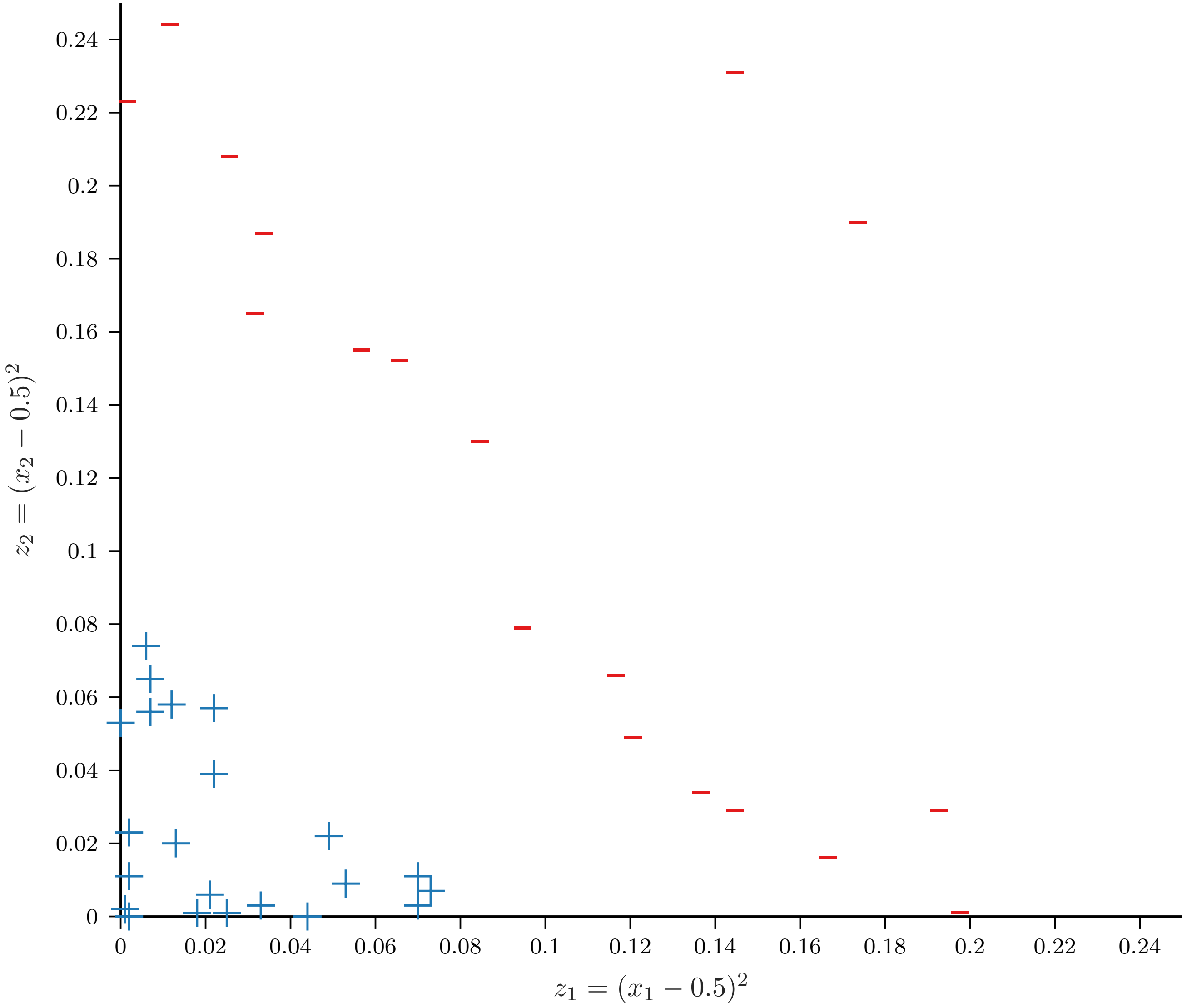
- Given D-dimensional inputs $\boldsymbol{x} = [x_1, \ldots, x_D]$, first compute some transformation of our input, e.g.,

$$\phi([x_1, x_2]) = [z_1 = (x_1 - 0.5)^2, z_2 = (x_2 - 0.5)^2]$$

Nonlinear Models

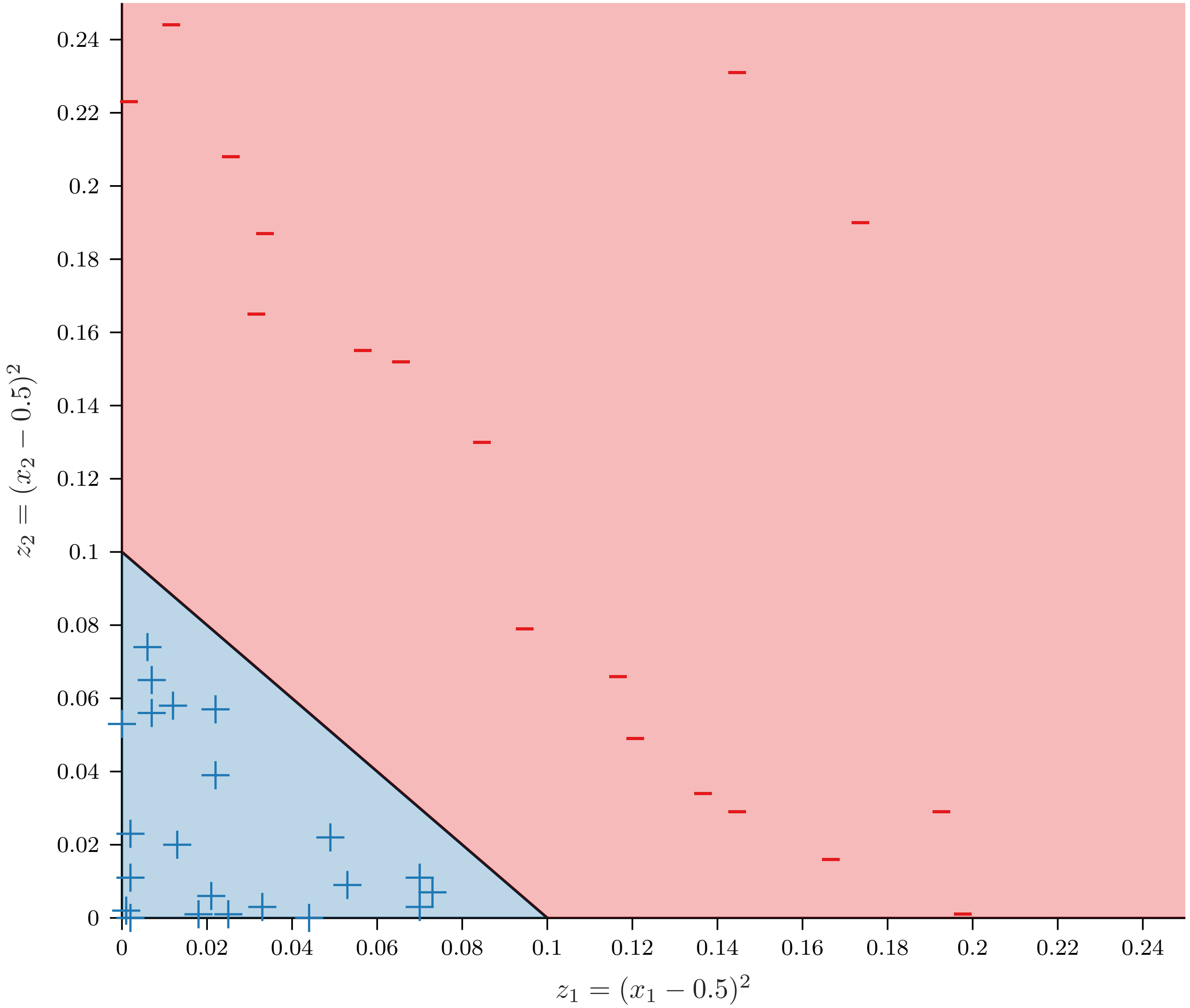


Nonlinear Models

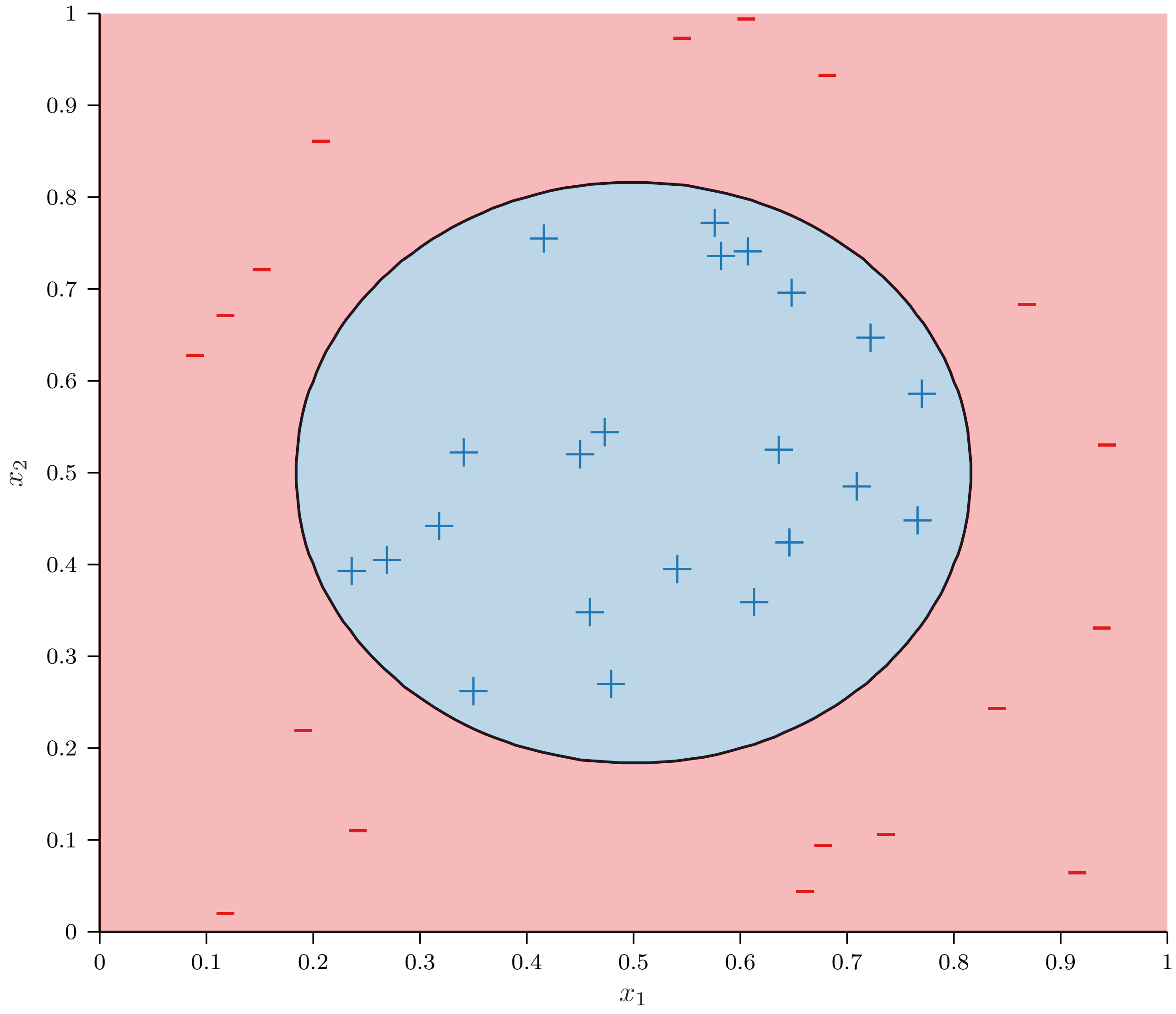


Nonlinear Models

$z_2 = -z_1 + 0.1$



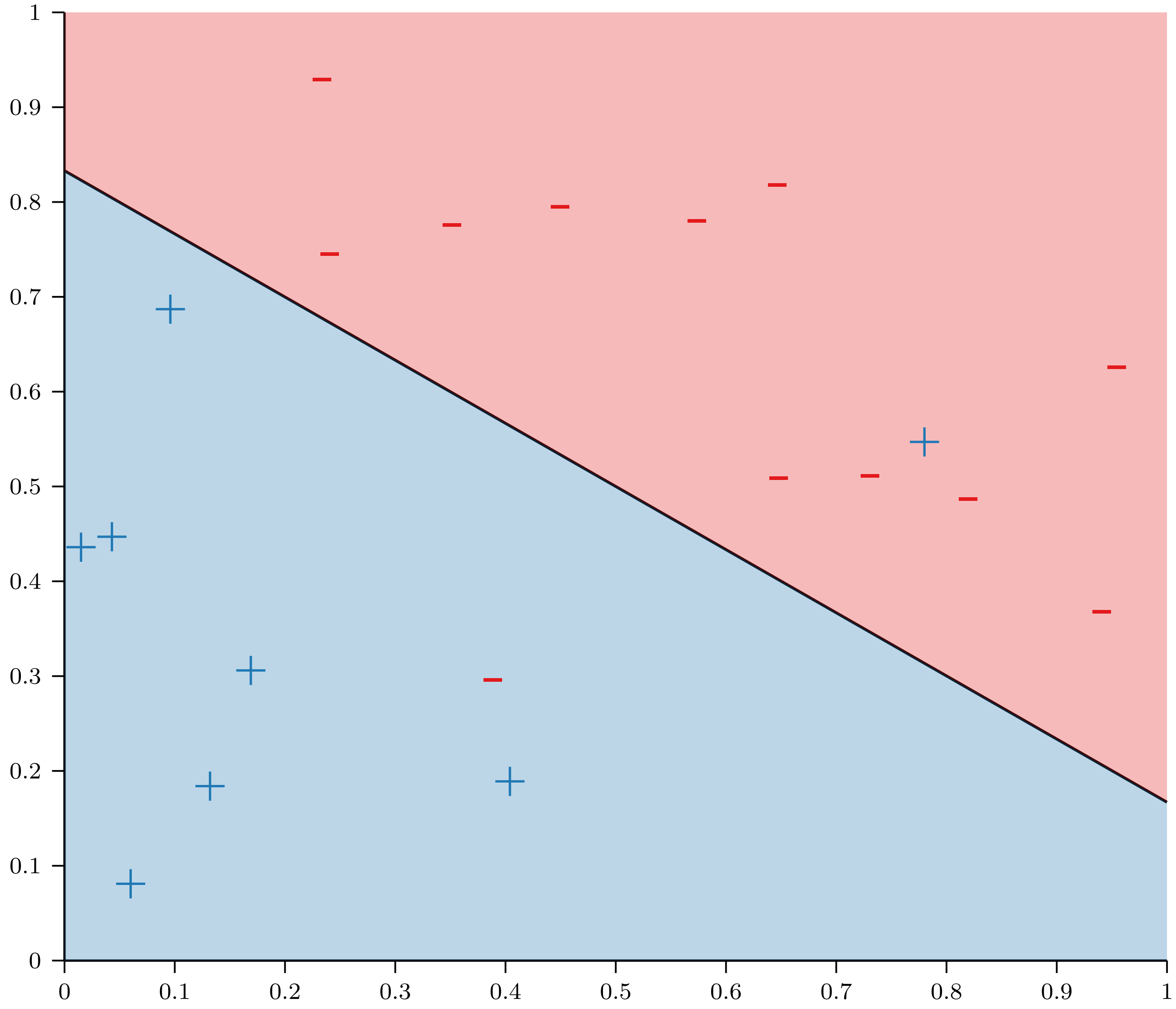
Nonlinear Models



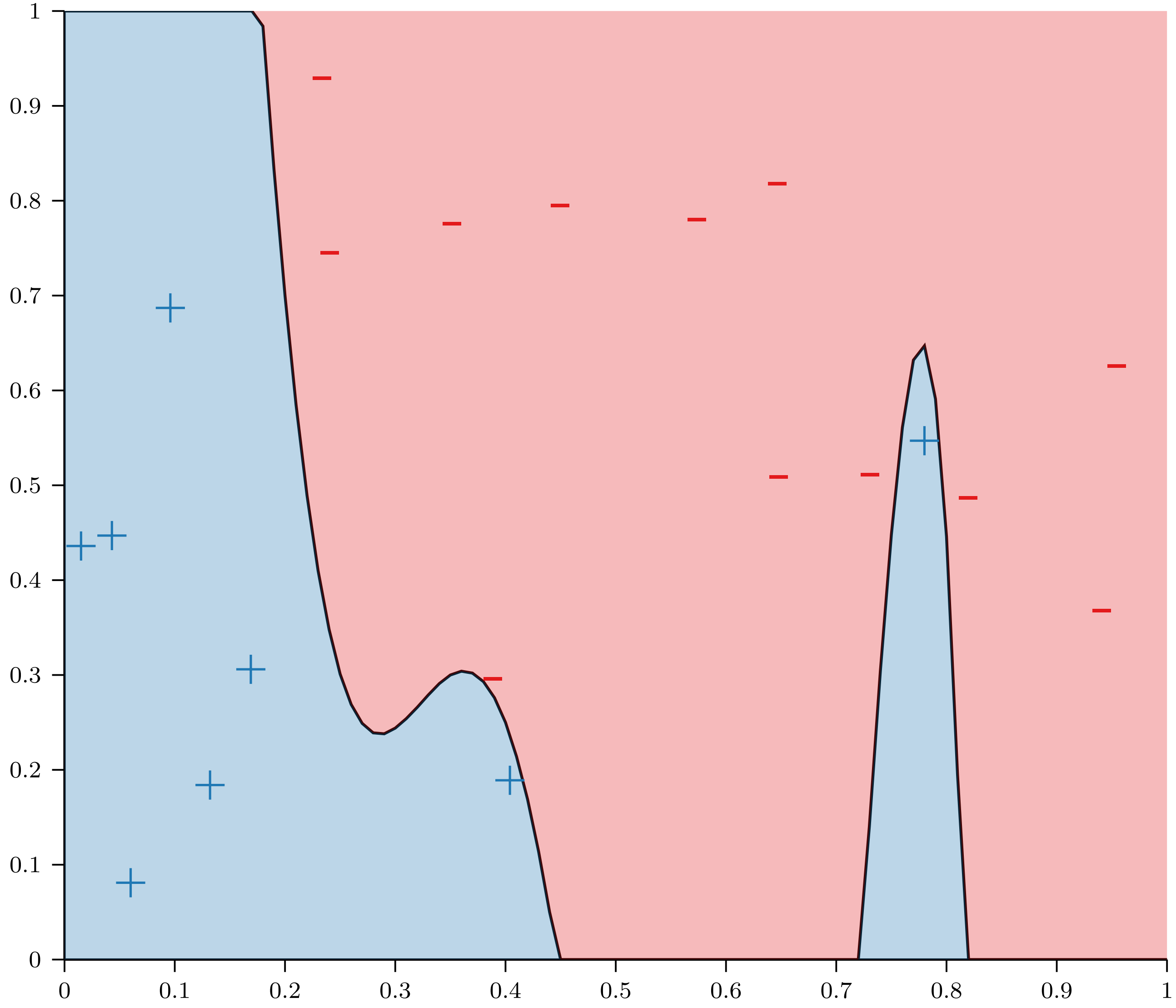
General Q^{th}-order Transforms

- $\phi_{2,2}([x_1, x_2]) = [x_1, x_2, x_1^2, x_1 x_2, x_2^2]$
- $\phi_{2,3}([x_1, x_2]) = [x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3]$
- $\phi_{2,4}([x_1, x_2]) =$ $[x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3, x_1^4, x_1^3 x_2, x_1^2 x_2^2, x_1 x_2^3, x_2^4]$
- $\phi_{2,Q}$ maps a 2-D input to a $O(Q^2)$-D 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

Feature Transforms: Experiment

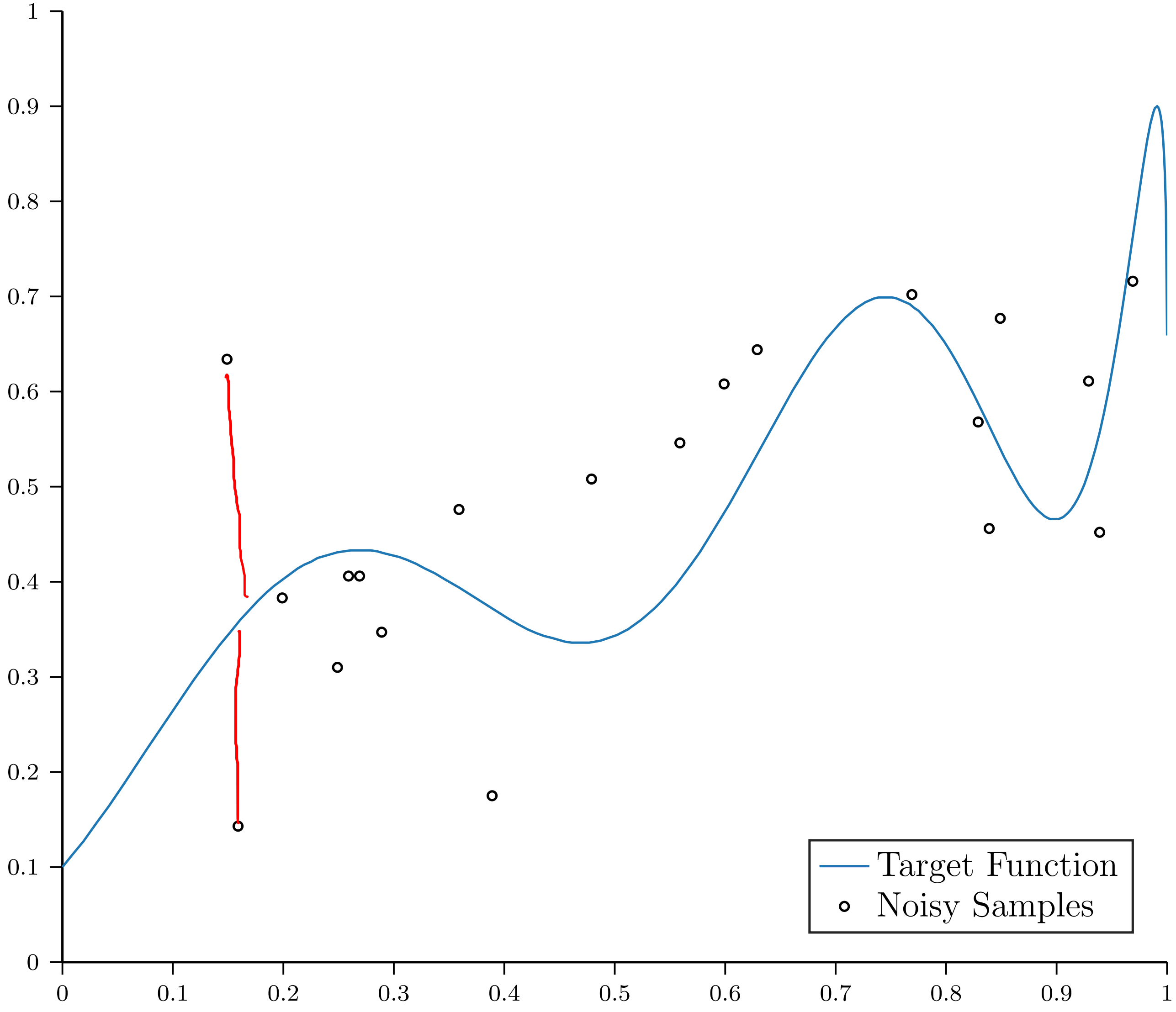
- $x \in \mathbb{R}$, $y \in \mathbb{R}$ and $N = 20$
- Targets are generated by a 10^{th}-order polynomial in x with additive Gaussian noise:

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

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

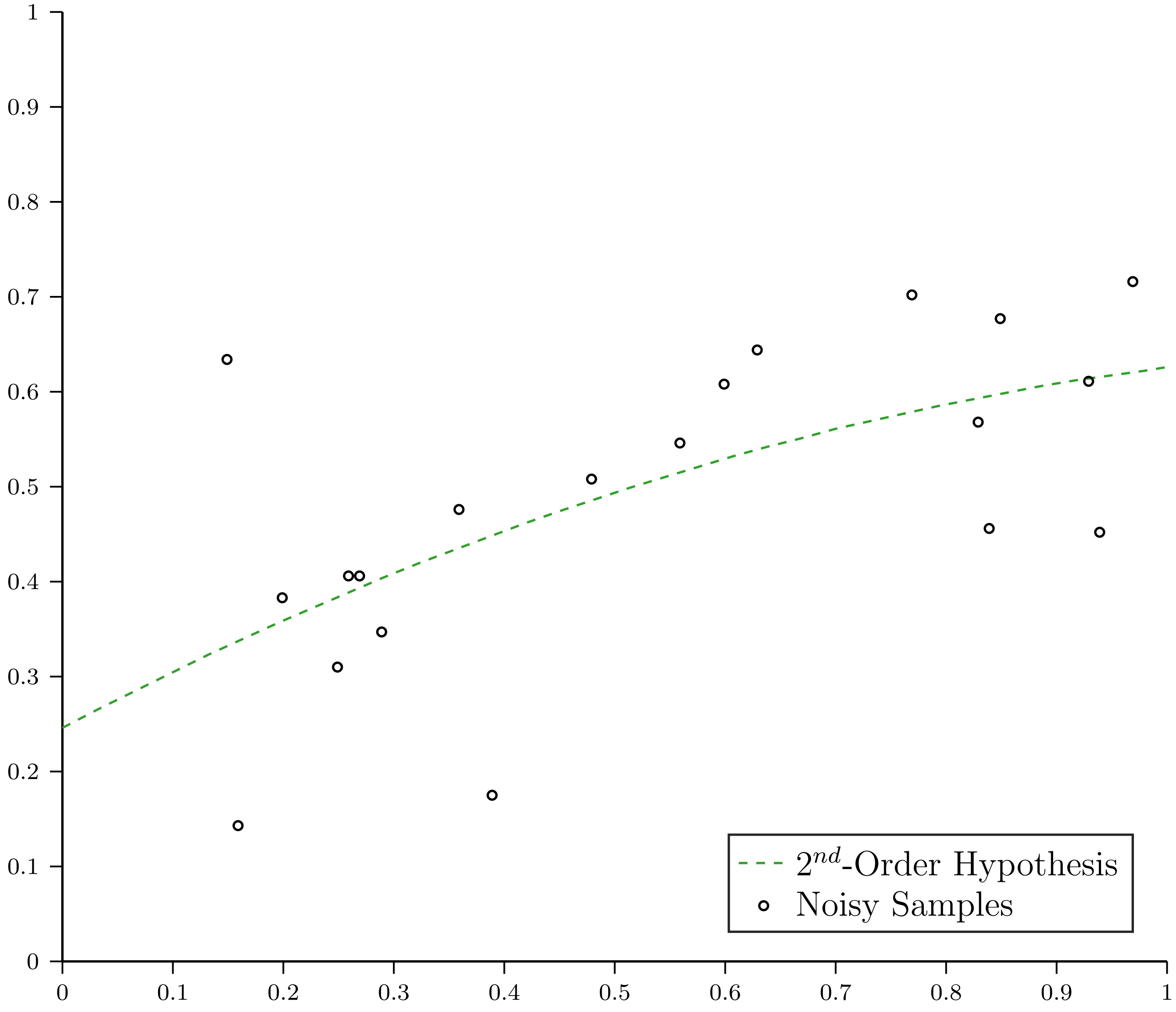
Noisy Targets

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



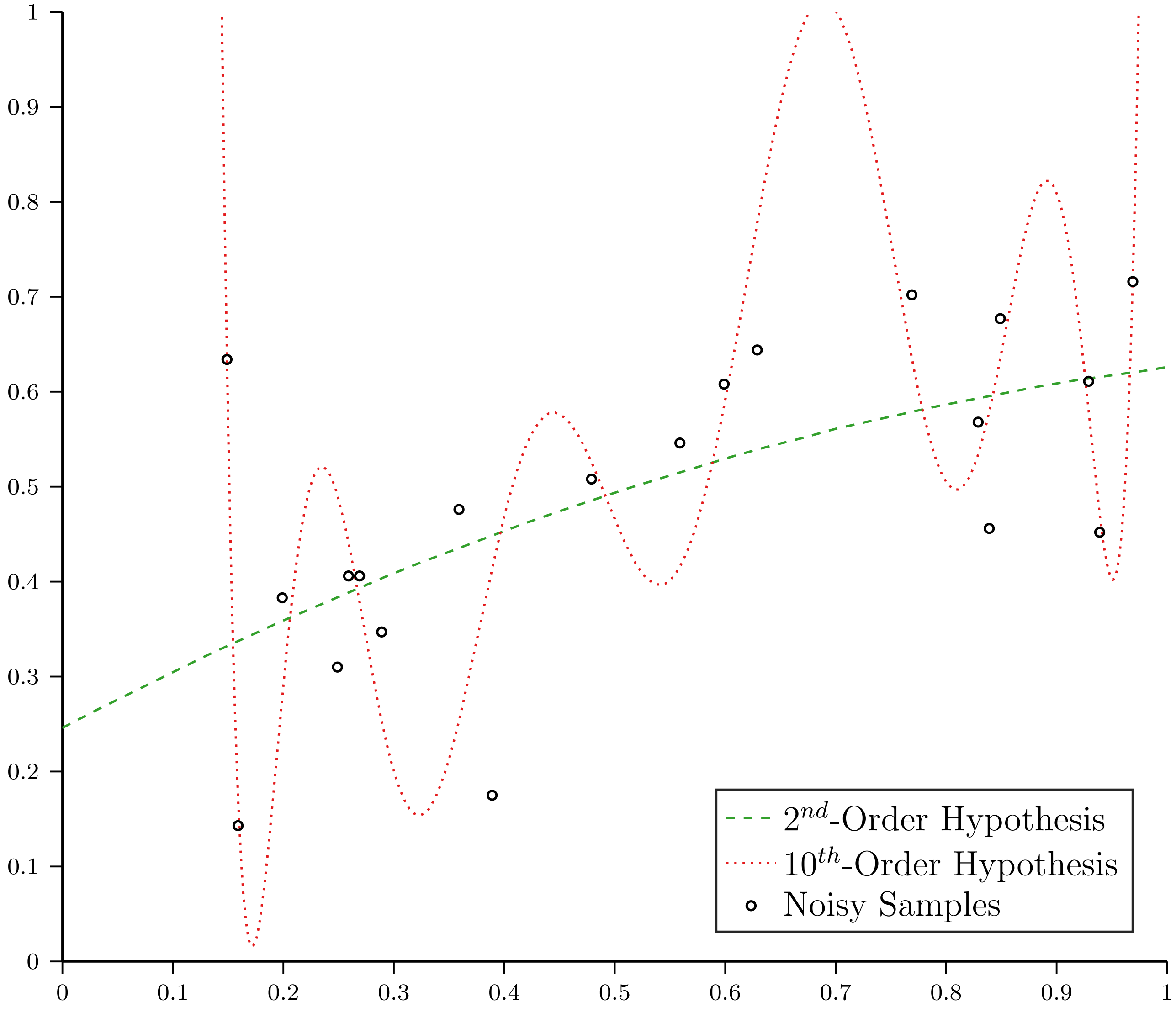
Noisy Targets

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



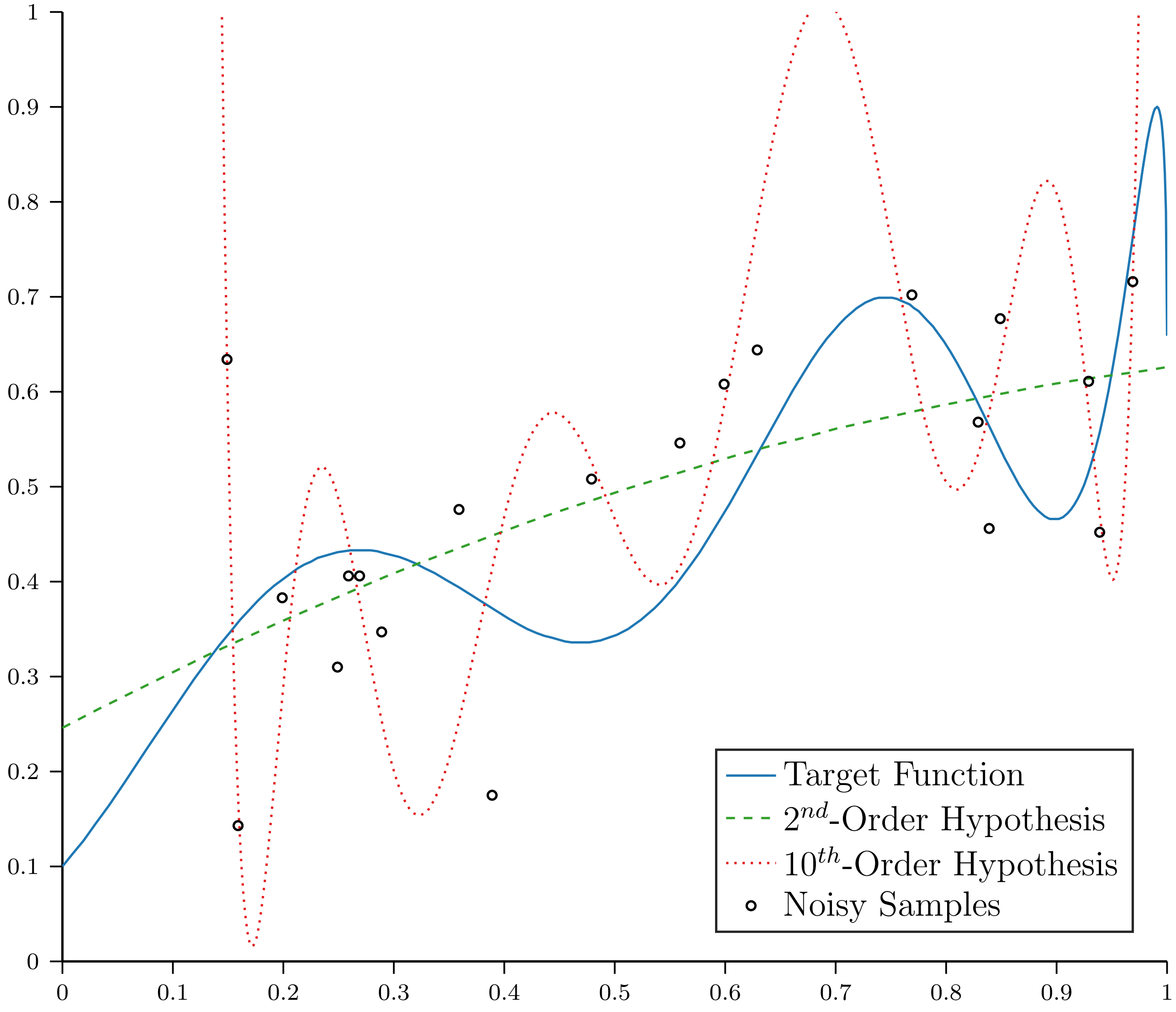
Noisy Targets

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



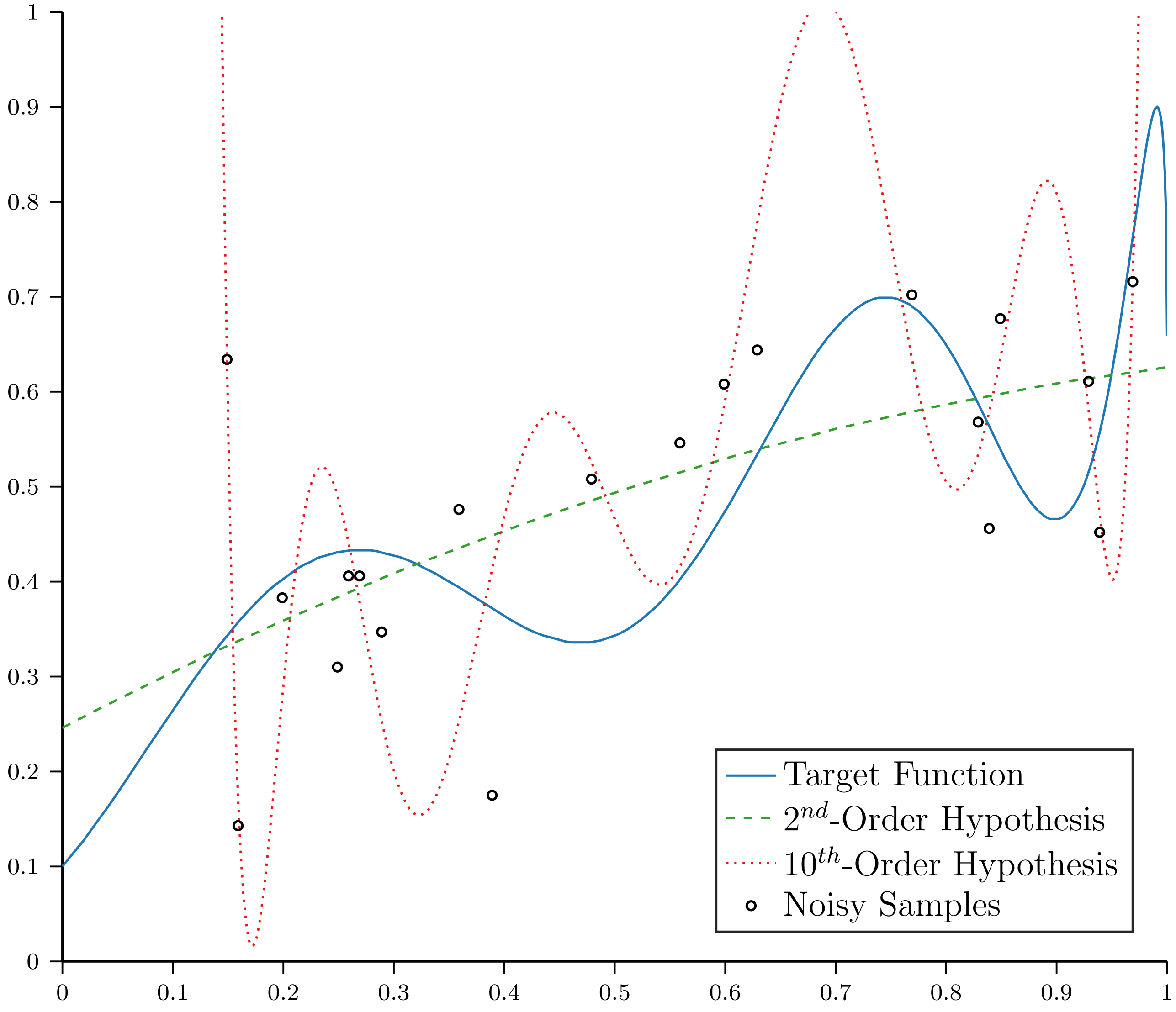
Noisy Targets

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



Noisy Targets

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



Feature Transforms: Experiment

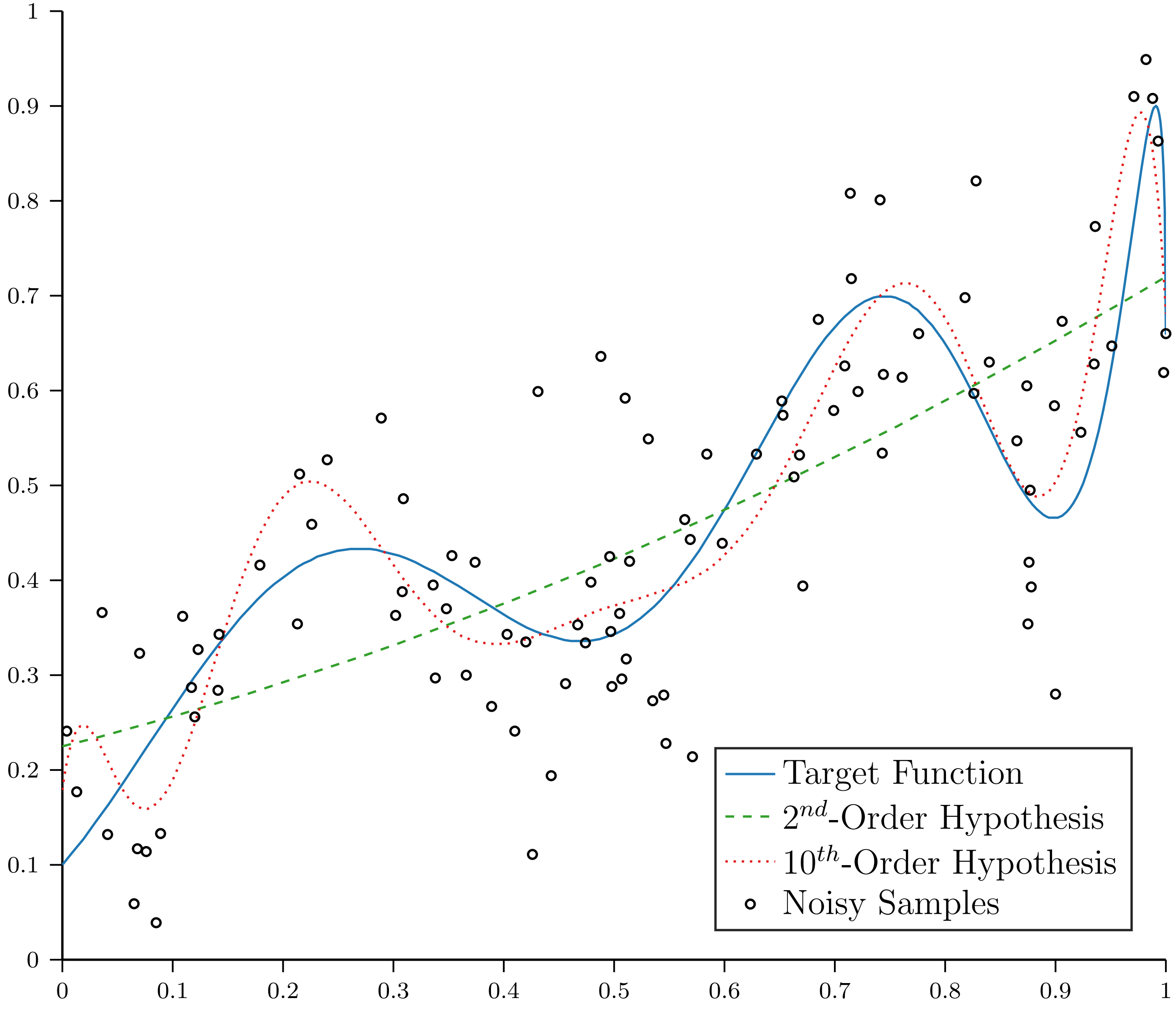
- $x \in \mathbb{R}$, $y \in \mathbb{R}$ and $N = 100$
- Targets are generated by a 10^{th}-order polynomial in x with additive Gaussian noise:

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

- $\mathcal{H}_2 = 2^{\text{nd}}$-order polynomials
  - $\phi_{1,2}(x) = [x, x^2]$
- $\mathcal{H}_{10} = 10^{\text{th}}$-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

	$\mathcal{H}_2$	$\mathcal{H}_{10}$
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

- $\mathcal{H}_{10} = 10^{\text{th}}$-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 $\mathrm{X} = \begin{bmatrix} 1 & \phi_{1,10}\left(x^{(1)}\right) \\ 1 & \phi_{1,10}\left(x^{(2)}\right) \\ \vdots & \vdots \\ 1 & \phi_{1,10}\left(x^{(N)}\right) \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$ find

  $\boldsymbol{\omega} = [\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}]$ that minimizes

$$\frac{1}{N}(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})^T(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})$$

- Subject to

$$\omega_3 = \omega_4 = \omega_5 = \omega_6 = \omega_7 = \omega_8 = \omega_9 = \omega_{10} = 0$$

Hard Constraints

- $\mathcal{H}_{10} = 10^{\text{th}}$-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 $\mathrm{X} = \begin{bmatrix} 1 & \phi_{1,10}\left(x^{(1)}\right) \\ 1 & \phi_{1,10}\left(x^{(2)}\right) \\ \vdots & \vdots \\ 1 & \phi_{1,10}\left(x^{(N)}\right) \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$ find

  $\boldsymbol{\omega} = [\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}]$ that minimizes

$$\frac{1}{N}\sum_{n=1}^{N}\left(\left(\sum_{d=0}^{10} x_d^{(n)}\omega_d\right) - y^{(n)}\right)^2$$

- Subject to

$$\omega_3 = \omega_4 = \omega_5 = \omega_6 = \omega_7 = \omega_8 = \omega_9 = \omega_{10} = 0$$

Hard Constraints

- $\mathcal{H}_{10} = 10^{\text{th}}$-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 $\mathrm{X} = \begin{bmatrix} 1 & \phi_{1,10}\left(x^{(1)}\right) \\ 1 & \phi_{1,10}\left(x^{(2)}\right) \\ \vdots & \vdots \\ 1 & \phi_{1,10}\left(x^{(N)}\right) \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$ find $\boldsymbol{\omega} = [\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}]$ that minimizes

$$\frac{1}{N}\sum_{n=1}^{N}\left(\left(\sum_{d=0}^{2} x_d^{(n)}\omega_d\right) - y^{(n)}\right)^2$$

- Subject to nothing!

Hard Constraints

- $\mathcal{H}_2 = 2^{\text{nd}}$-order polynomials
  - $\phi_{1,2}(x) = [x, x^2]$
- Given $\mathrm{X} = \begin{bmatrix} 1 & \phi_{1,2}\left(x^{(1)}\right) \\ 1 & \phi_{1,2}\left(x^{(2)}\right) \\ \vdots & \vdots \\ 1 & \phi_{1,2}\left(x^{(N)}\right) \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$ find

  $\boldsymbol{\omega} = [\omega_0, \omega_1, \omega_2]$
  that minimizes

$$\frac{1}{N}(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})^T(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})$$

- Subject to nothing!

Soft Constraints

- More generally, ϕ can be any nonlinear transformation, e.g., exp, log, sin, sqrt, etc...
- Given $\mathrm{X} = \begin{bmatrix} 1 & \phi_1\left(\boldsymbol{x}^{(1)}\right) & \cdots & \phi_m\left(\boldsymbol{x}^{(1)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1\left(\boldsymbol{x}^{(N)}\right) & \cdots & \phi_m\left(\boldsymbol{x}^{(N)}\right) \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$, find $\boldsymbol{\omega}$ that minimizes

$$\frac{1}{N}(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})^T(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})$$

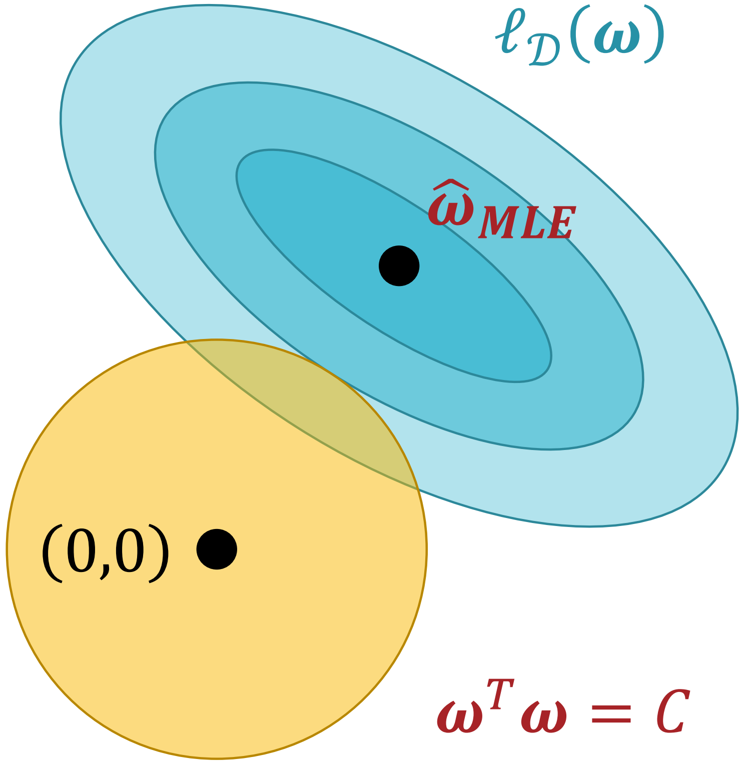
- Subject to:

$$\|\boldsymbol{\omega}\|_2^2 = \boldsymbol{\omega}^T\boldsymbol{\omega} = \sum_{d=0}^{D} \omega_d^2 \leq C$$

Soft Constraints

minimize $\ell_{\mathcal{D}}(\boldsymbol{\omega}) = \overset{\frac{1}{N}}{\swarrow}(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})^T(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})$

subject to $\boldsymbol{\omega}^T\boldsymbol{\omega} \leq C$

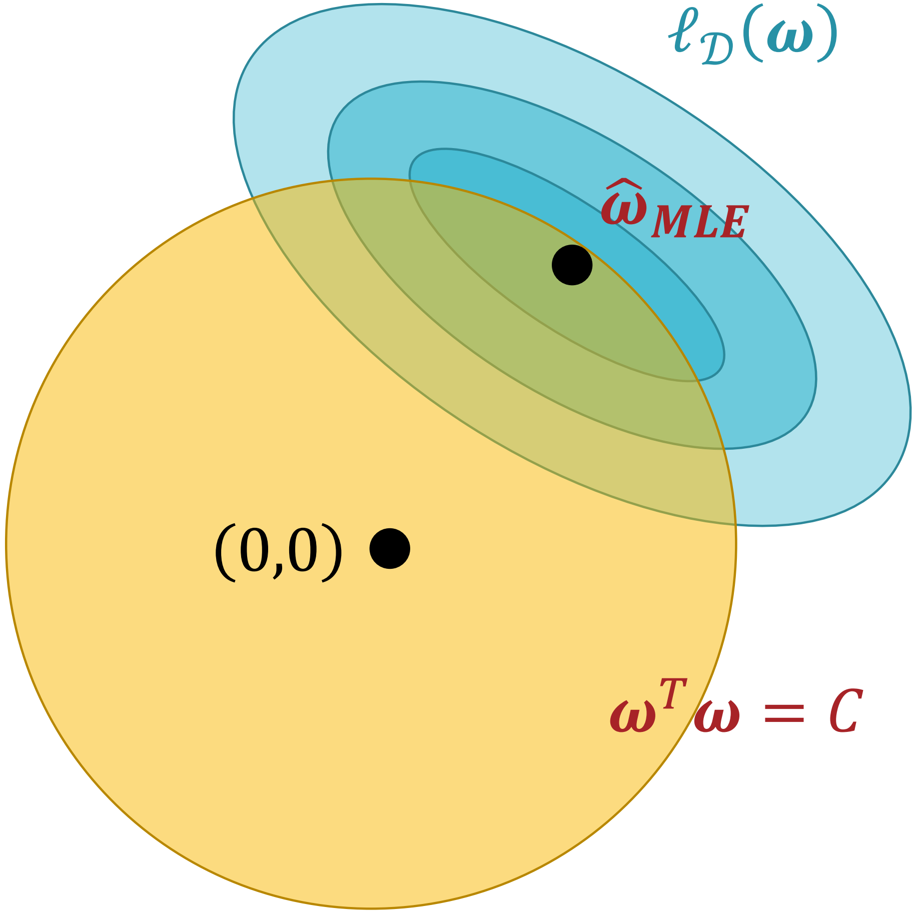

$\ell_{\mathcal{D}}(\boldsymbol{\omega})$
$\hat{\boldsymbol{\omega}}_{\boldsymbol{MLE}}$
(0,0)
$\boldsymbol{\omega}^T\boldsymbol{\omega} = C$

$\omega_1^2 + \omega_2^2 \leq C$

Soft Constraints

minimize $\ell_{\mathcal{D}}(\boldsymbol{\omega}) = (\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})^T(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})$

subject to $\boldsymbol{\omega}^T\boldsymbol{\omega} \leq C$


$\ell_{\mathcal{D}}(\boldsymbol{\omega})$
$\widehat{\boldsymbol{\omega}}_{MLE}$
(0,0)
$\boldsymbol{\omega}^T\boldsymbol{\omega} = C$

Soft Constraints

minimize $\ell_{\mathcal{D}}(\boldsymbol{\omega}) = (X\boldsymbol{\omega} - \boldsymbol{y})^T(X\boldsymbol{\omega} - \boldsymbol{y})$

$r(\omega) \le C \swarrow$

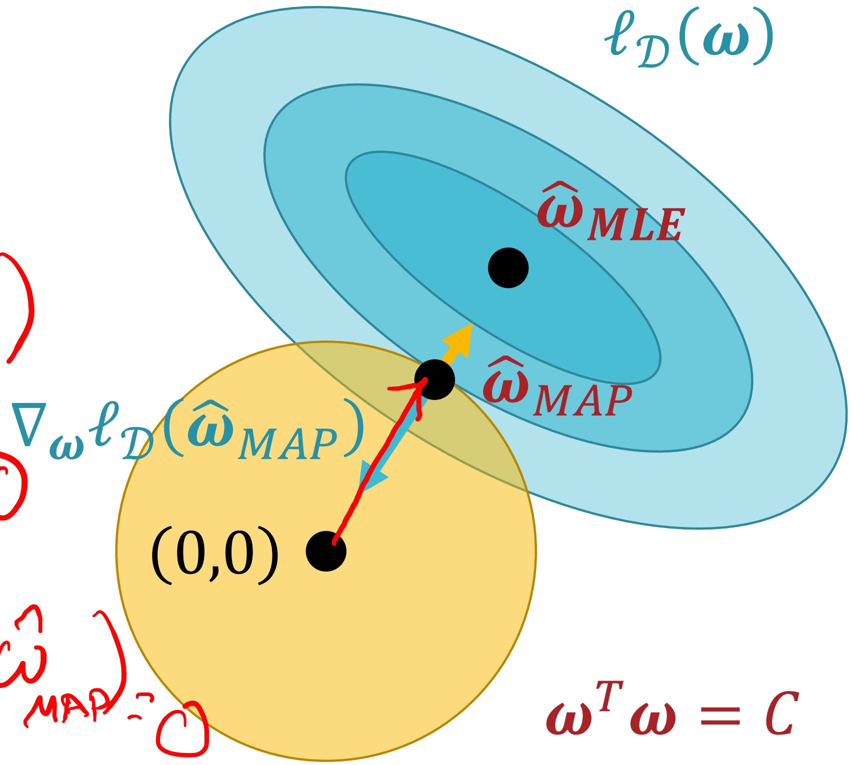
subject to $\boldsymbol{\omega}^T\boldsymbol{\omega} \le C$

$\hat{\omega}_{MAP}$ we know

$\hat{\omega}_{MAP} \propto \nabla_w \ell_D(\hat{\omega}_{MAP})$

$\nabla_w \ell_D(\hat{\omega}_{MAP}) - \lambda_c \hat{\omega}_{MAP} = 0$

$\nabla_w\left(\ell_D(\hat{\omega}_{MAP}) - \frac{\lambda_c}{2} \hat{\omega}_{MAP}^T \hat{\omega}_{MAP}\right) = 0$


$\ell_{\mathcal{D}}(\boldsymbol{\omega})$
$\widehat{\boldsymbol{\omega}}_{MLE}$
$\widehat{\boldsymbol{\omega}}_{MAP}$
$\nabla_{\omega}\ell_{\mathcal{D}}(\widehat{\boldsymbol{\omega}}_{MAP})$
(0,0)
$\boldsymbol{\omega}^T\boldsymbol{\omega} = C$

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

minimize $\ell_{\mathcal{D}}(\boldsymbol{\omega}) = (\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})^T(\mathrm{X}\boldsymbol{\omega} - \boldsymbol{y})$

subject to $\boldsymbol{\omega}^T\boldsymbol{\omega} \leq C$

$$\Updownarrow$$

minimize $\ell_{\mathcal{D}}^{AUG}(\boldsymbol{\omega}) = \ell_{\mathcal{D}}(\boldsymbol{\omega}) + \lambda_C\boldsymbol{\omega}^T\boldsymbol{\omega}$

Ridge Regression

minimize $\ell_{\mathcal{D}}^{AUG}(\boldsymbol{\omega}) = \ell_{\mathcal{D}}(\boldsymbol{\omega}) + \lambda_C \boldsymbol{\omega}^T \boldsymbol{\omega}$

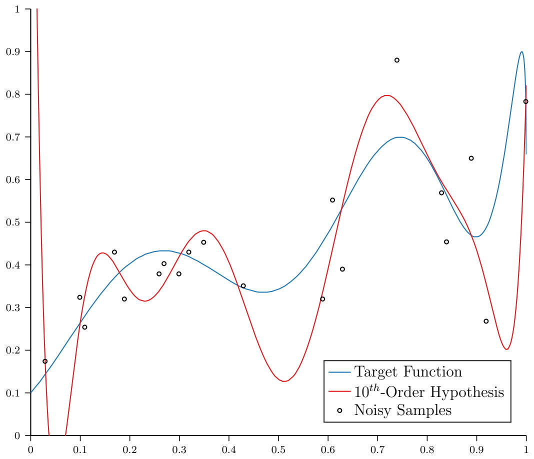
$$\nabla_{\omega} \ell_{D}^{AUG}(\omega) = \nabla \ell_{D}(\omega) + 2\lambda_c \omega$$

$$= (2X^TX\omega - 2X^Ty + 2\lambda_c \omega)$$

$$\Rightarrow 0 = (2X^TX\hat{\omega}_{MAP} - 2X^Ty + 2\lambda_c \hat{\omega}_{MAP})$$

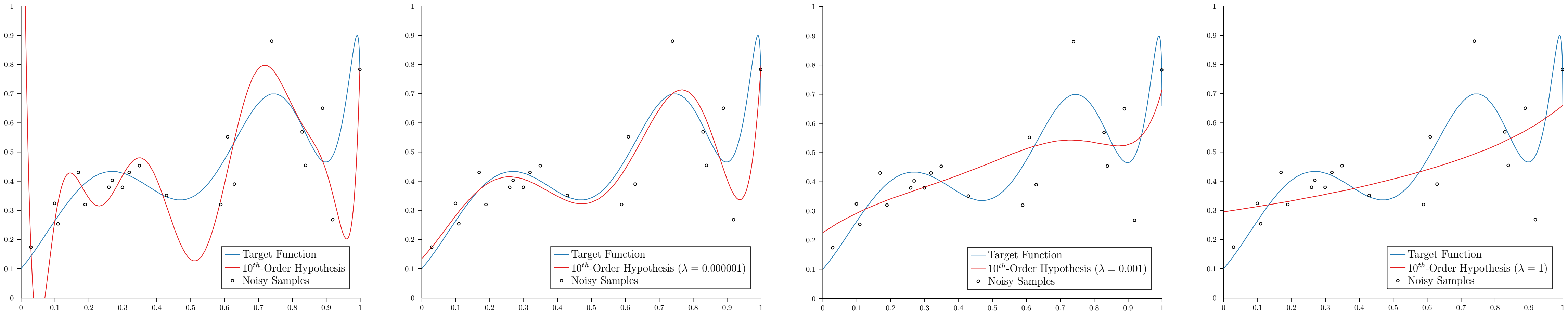
$$\Rightarrow \cancel{2}X^Ty = (\cancel{2}X^TX + \cancel{2}\lambda_c I_{D+1})\hat{\omega}_{MAP}$$

$$\Rightarrow \hat{\omega}_{MAP} = (X^TX + \lambda_c I_{D+1})^{-1} X^Ty$$



Ridge Regression

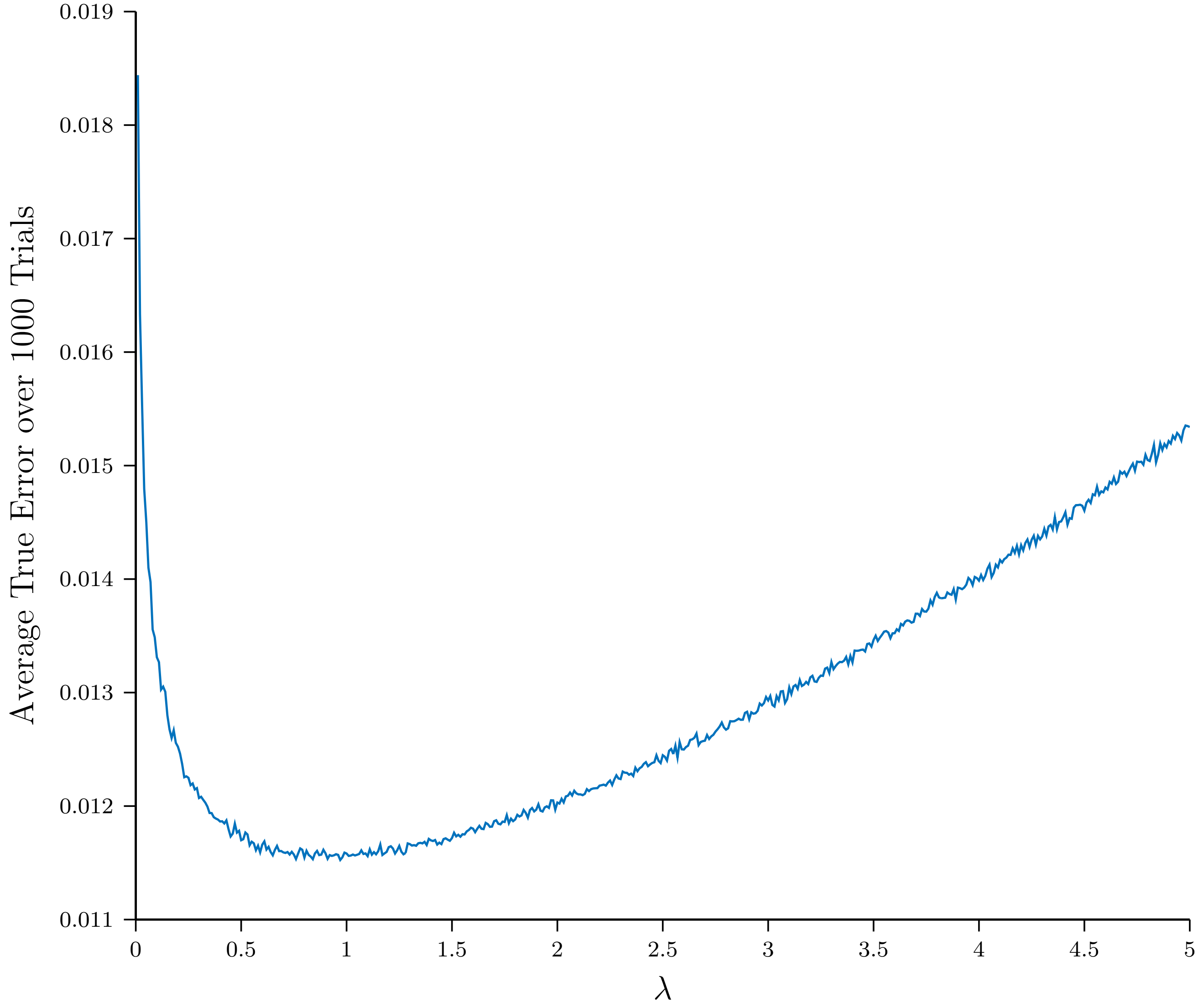
- 10-dimensional target function with additive Gaussian noise
- $\mathcal{H}_{10} = 10^{\text{th}}$-order polynomial



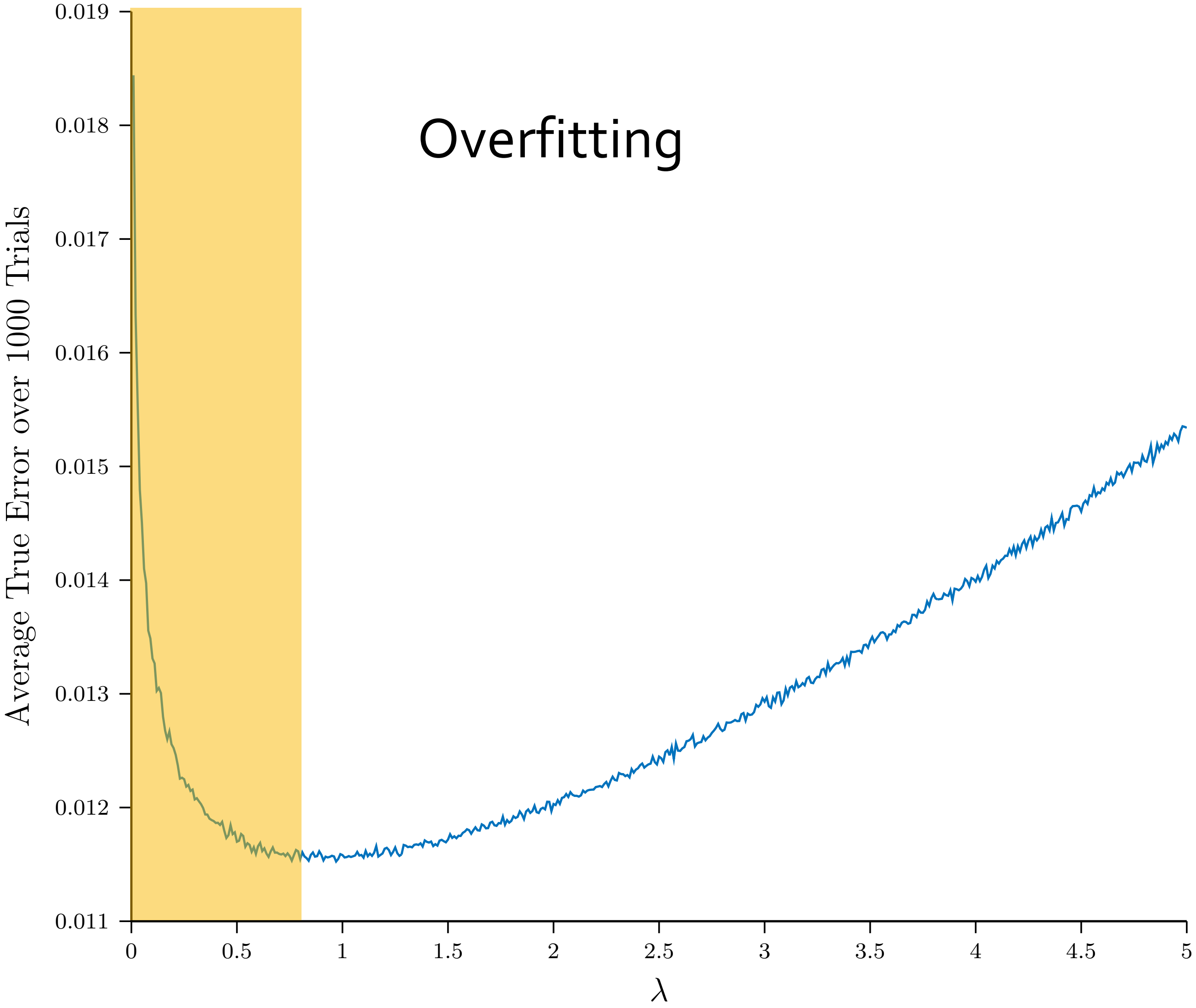
Ridge Regression

	$\lambda_C = 0$
True Error	0.059
	Overfit

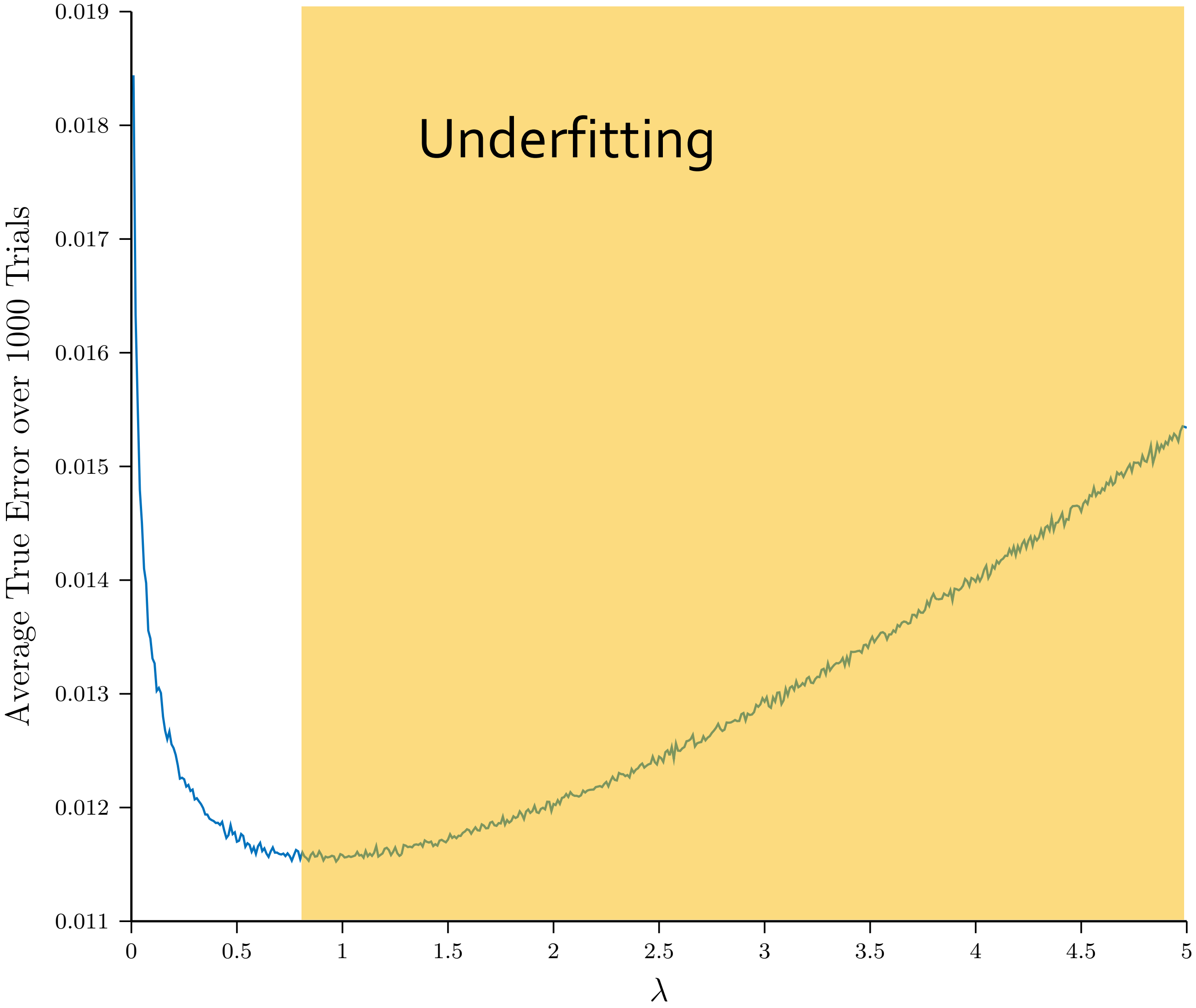
Setting λ



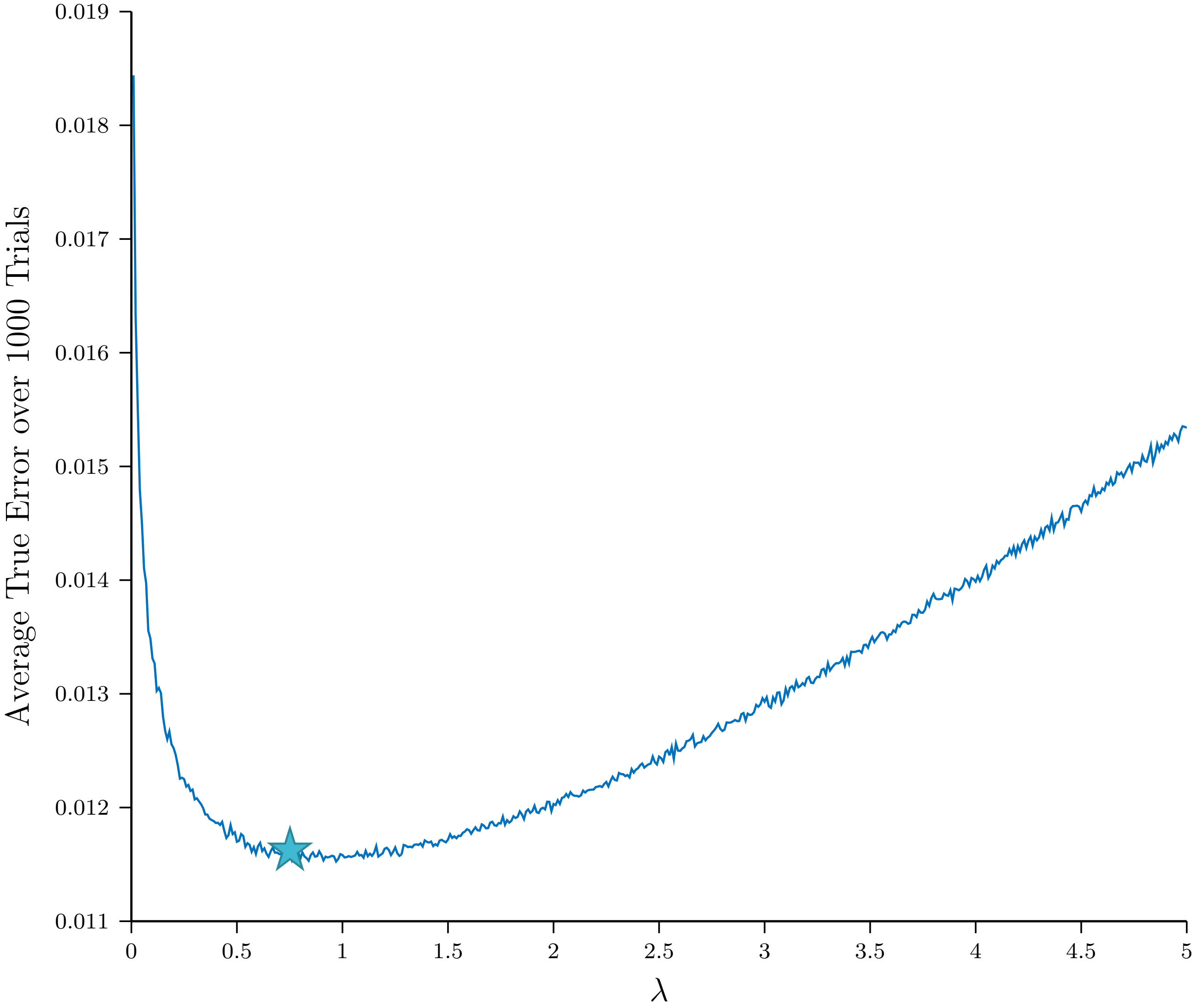
Setting λ



Setting λ

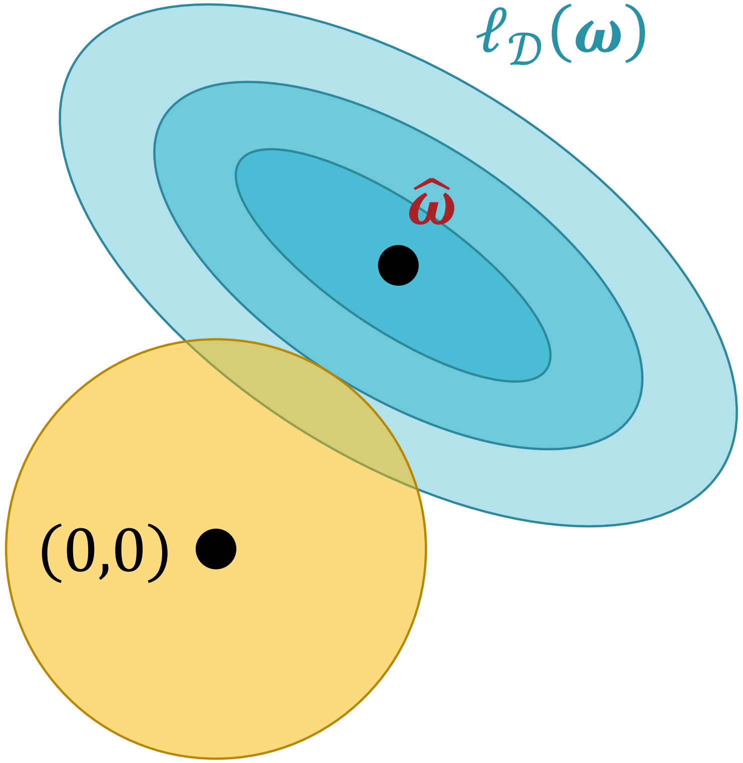


Setting λ

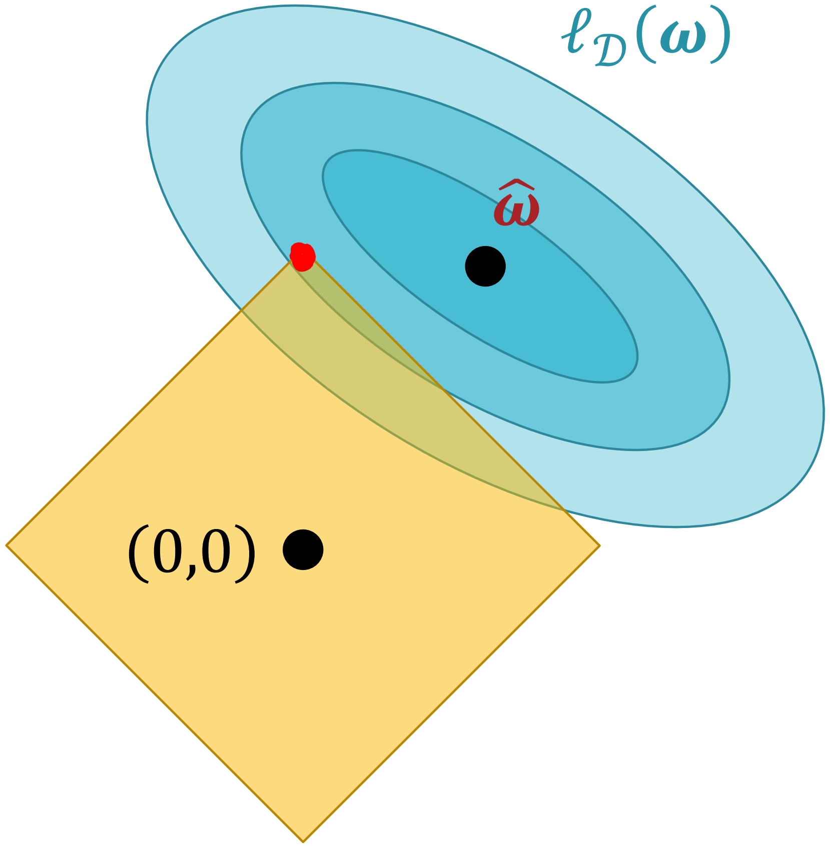


Other Regularizers

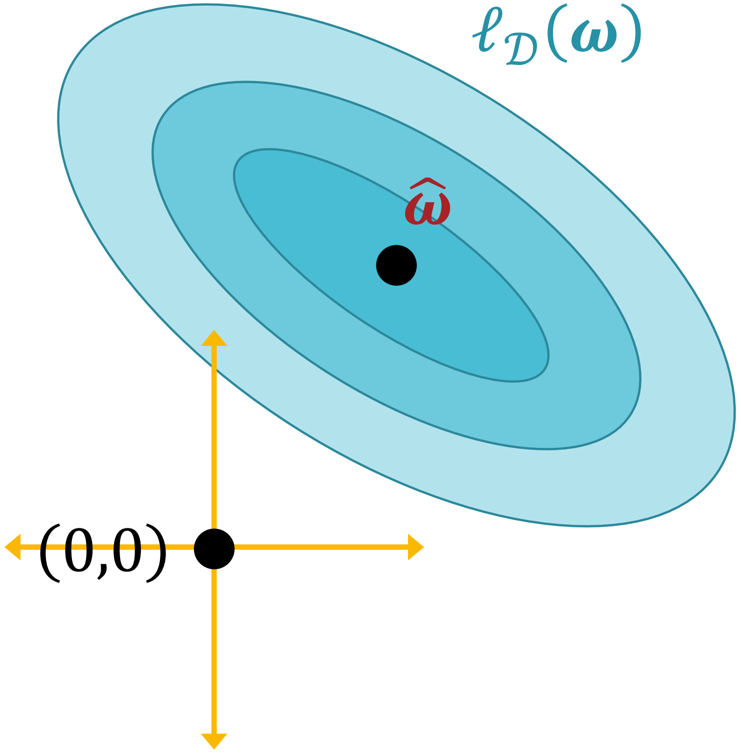
$\ell_{\mathcal{D}}(\boldsymbol{\omega}) + \lambda r(\boldsymbol{\omega})$		
Ridge or $L2$	$r(\boldsymbol{\omega}) = \lVert\boldsymbol{\omega}\rVert_2^2 = \sum_{d=0}^{D} \omega_d^2$	Encourages small weights
Lasso or $L1$	$r(\boldsymbol{\omega}) = \lVert\boldsymbol{\omega}\rVert_1 = \sum_{d=0}^{D} \lvert\omega_d\rvert$	Encourages sparsity
$L0$	$r(\boldsymbol{\omega}) = \lVert\boldsymbol{\omega}\rVert_0 = \sum_{d=0}^{D} \mathbb{1}(\omega_d \neq 0)$	Encourages sparsity (intractable)


$\ell_{\mathcal{D}}(\boldsymbol{\omega})$
$\hat{\boldsymbol{\omega}}$
(0,0)

Ridge or $L2$


$\ell_{\mathcal{D}}(\boldsymbol{\omega})$
$\hat{\boldsymbol{\omega}}$
(0,0)

Lasso or $L1$


$\ell_{\mathcal{D}}(\boldsymbol{\omega})$
$\hat{\boldsymbol{\omega}}$
(0,0)

$L0$

Other Regularizers

Nonlinear Transforms

- Decide on some transformation $\Phi: \mathcal{X} \rightarrow \mathcal{Z}$
- Given $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, learn a hypothesis, $\tilde{h}(\boldsymbol{z})$, using $\widetilde{\mathcal{D}} = \left\{\left(\boldsymbol{z}^{(i)} = \Phi\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)\right\}_{i=1}^{N}$
- Return the corresponding predictor in the original space: $h(\boldsymbol{x}) = \tilde{h}\left(\Phi(\boldsymbol{x})\right)$

Efficiency

- Depending on the transformation Φ and the dimensionality of the original input space $\mathcal{X}$, computing $\Phi(\boldsymbol{x})$ can be prohibitively computationally expensive
  - Computing $\Phi_2(\boldsymbol{x}) = [x_1, x_2, \dots, x_D, x_1^2, x_1 x_2, \dots, x_D^2]$ for $x \in \mathbb{R}^D$ requires $D + \binom{D}{2} + D = \frac{D^2+3D}{2} = O(D^2)$ time
  - Computing $\Phi_{\boxed{10}}(\boldsymbol{x})$ requires $O(D^{10})$ time
- Tradeoff:
  - High-dimensional transformations can result in good hypotheses (as long as they don't overfit) but…
  - High-dimensional transformations are expensive

Inner Product Methods

- Insight: the predictions of many machine learning models can be expressed as a function of inner products between feature vectors i.e., given $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$

$$h_{\mathcal{D}}(\boldsymbol{x}) = g\left(\left\{\boldsymbol{x}^T\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}, \left\{\boldsymbol{x}^{(j)^T}\boldsymbol{x}^{(i)}\right\}_{i,j=1}^{N}\right)$$

- Crucially, feature vectors ***only*** appear in inner products with other feature vectors

- Applying a feature transformation Φ gives

$$h_{\mathcal{D}}(\Phi(\boldsymbol{x})) = \underset{\uparrow}{g}\left(\left\{\underbrace{\Phi(\boldsymbol{x})^T\Phi\left(\boldsymbol{x}^{(i)}\right)}\right\}_{i=1}^{N}, \left\{\underbrace{\Phi\left(\boldsymbol{x}^{(j)}\right)^T\Phi\left(\boldsymbol{x}^{(i)}\right)}\right\}_{i,j=1}^{N}\right)$$

The Kernel Trick

- Idea: for inner product methods, instead of computing $\Phi(\boldsymbol{x})$, use some function K_Φ s.t. $K_\Phi(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}')\ \forall\ \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$
  - $K_\Phi(\boldsymbol{x}, \boldsymbol{x}')$ should be cheaper to compute than $\Phi(\boldsymbol{x})$

Ex: $\Phi_2' = [x_1, \dots, x_D, x_1^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \dots, \sqrt{2}x_{D-1}x_D, x_D^2]$

$$\Phi_2(x)^T\Phi_2(x') = \underbrace{\sum_{i=1}^{D} x_i x_i'}_{} + \sum_{i=1}^{D} x_i^2 x_i'^2 + \sum_{i=1}^{D}\sum_{j\neq i} 2x_i x_j x_i' x_j'$$

$$K(x, x') = x^T x' + (x^T x')^2$$

Computing $K(x, x')$ takes $O(D)$ operations

vs. Computing $\Phi_2(x)^T\Phi_2(x')$ takes $O(D^2)$

Common Kernels

- $K_{\Phi_2'}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}' + (\boldsymbol{x}^T \boldsymbol{x}')^2$
  - Implied feature transformation:

    $$\Phi_2'(\boldsymbol{x}) = \left[x_1, \dots, x_D, x_1^2, \sqrt{2}x_1 x_2, \dots, \sqrt{2}x_{D-1}x_D, x_D^2\right]$$

  - Implied dimensionality: $\frac{D^2+3D}{2}$

- $K_{\Phi_2^{(\gamma)}}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \gamma \boldsymbol{x}^T \boldsymbol{x}')^2 - 1$
  - Implied feature transformation:

    $$\Phi_2^{(\gamma)}(\boldsymbol{x}) = \left[\sqrt{2\gamma}x_1, \dots, \sqrt{2\gamma}x_D, \gamma x_1^2, \gamma x_1 x_2, \dots, \gamma x_D^2\right]$$

    - γ affects the geometry of the transform

  - Implied dimensionality: $\frac{D^2+3D}{2}$

Common Kernels

- Polynomial Kernel: $K_{\Phi_Q^{(\gamma)}}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \gamma \underbrace{\boldsymbol{x}^T \boldsymbol{x}'}_{})^{\boxed{Q}} - 1$
  - Implied dimensionality: $O(D^Q)$
  - γ affects the geometry of the transform
- Gaussian-RBF Kernel: $K_{\Phi_r}(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2r}}$
  - Implied feature transformation: $\Phi_r(\boldsymbol{x}) = \left[e^{-\frac{x_1^2}{2r}}, \dots, e^{-\frac{x_D^2}{2r}},\right.$

$$\left. e^{-\frac{x_1^2}{2r}}\sqrt{\frac{(x_1)^2}{1!r^1}}, \dots, e^{-\frac{x_D^2}{2r}}\sqrt{\frac{(x_D)^2}{1!r^1}}, e^{-\frac{x_1^2}{2r}}\sqrt{\frac{(x_1^2)^2}{2!r^2}}, \dots, e^{-\frac{x_D^2}{2r}}\sqrt{\frac{(x_D^2)^2}{2!r^2}}, \dots\right]$$

Common Kernels

- Polynomial Kernel: $K_{\Phi_Q^{(\gamma)}}(\boldsymbol{x},\boldsymbol{x}') = (1+\gamma\boldsymbol{x}^T\boldsymbol{x}')^Q - 1$
  - Implied dimensionality: $O(D^Q)$
  - γ affects the geometry of the transform
- Gaussian-RBF Kernel: $K_{\Phi_r}(\boldsymbol{x},\boldsymbol{x}') = e^{-\frac{\|x-x'\|^2}{2r}}$
  - Implied feature transformation: $\Phi_r(x) =$

$$\left[\left[e^{-\frac{x_1^2}{2r}}\sqrt{\frac{\left(x_1^d\right)^2}{d!r^d}}, \dots, e^{-\frac{x_D^2}{2r}}\sqrt{\frac{\left(x_1^d\right)^2}{d!r^d}}\right] : d \in \mathbb{N}\right]$$

  - Implied dimensionality: ∞!

Kernels Everywhere!

- Any method that only depends on the Euclidean distance between data points is an inner product method:

$$\|x - x'\|_2^2 = (x - x')^T (x - x')$$

$$= x^T x - 2x^T x' + x'^T x'$$

E.g. you can "kernelize" KNN

$$\|\Phi(x) - \Phi(x')\|_2^2 = K(x,x) - 2K(x,x') + K(x',x')$$

Kernels Everywhere!

- We can also kernelize linear/ridge regression!

$$\hat{\omega}_{MAP} = (X^TX + \lambda_c I_{D+1}) X^T y$$

$$X = \begin{bmatrix} 1 & x^{(1)T} \\ 1 & x^{(2)T} \\ & \vdots \\ 1 & x^{(N)T} \end{bmatrix}$$

$$XX^T = \begin{bmatrix} 1 & x^{(1)T} \\ 1 & x^{(2)T} \\ \vdots & \vdots \\ 1 & x^{(N)T} \end{bmatrix} \begin{bmatrix} 1 & 1 & & 1 \\ x^{(1)} & x^{(2)} & \cdots & x^{(N)} \end{bmatrix}$$

$$\hat{\omega}_{MAP} = \underbrace{X^T}(\underbrace{XX^T} + \lambda_c I_N)^{-1} y$$

$$h_D(x') = \hat{\omega}_{MAP}^T x' = y^T((\underbrace{XX^T} + \lambda_c I_N)^T)^{-1} \underbrace{Xx'}$$

Valid Kernels

- Any function K is a valid kernel if and only if:
  - $\exists$ a transformation Φ s.t.

$$K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}') \ \forall \ \boldsymbol{x}, \boldsymbol{x}'$$

$$\Updownarrow$$

  - the Gram matrix

$$\mathrm{K} = \begin{bmatrix} K\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}\right) & K\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) & \cdots & K\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(N)}\right) \\ K\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right) & K\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)}\right) & \cdots & K\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(N)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ K\left(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}\right) & K\left(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(2)}\right) & \cdots & K\left(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}\right) \end{bmatrix}$$

    is symmetric and positive semi-definite $\forall$ sets $\left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\right\}$

Building New Kernels

- For any valid kernels K_1, K_2 with implied feature transformations Φ_1, Φ_2 and non-negative coefficients c_1, c_2, the following are all valid kernels:
  - $K(\boldsymbol{x}, \boldsymbol{x}') = c_1 K_1(\boldsymbol{x}, \boldsymbol{x}') + c_2 K_2(\boldsymbol{x}, \boldsymbol{x}')$
  - $K(\boldsymbol{x}, \boldsymbol{x}') = c_1 K_1(\boldsymbol{x}, \boldsymbol{x}') K_2(\boldsymbol{x}, \boldsymbol{x}')$
  - $K(\boldsymbol{x}, \boldsymbol{x}') = e^{K_1(\boldsymbol{x}, \boldsymbol{x}')}$

Key Takeaways

- Polynomial/non-linear feature transformations allow for learning non-linear functions/decision boundaries
    - Can lead to overfitting…
        - Address with regularization!
        - Analogous to constrained optimization, solve via method of Lagrange multipliers
        - Regularization level is a hyperparameter
    - Can be computationally expensive…
        - Address with kernels!
        - Alternative to explicitly computing feature transformations for inner product methods