Elements of Machine Learning & Data Science

Winter semester 2023/24

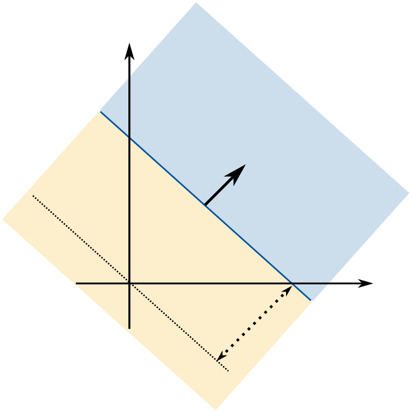
Lecture 15 – Linear Regression

05.12.2023

Prof. Bastian Leibe

Machine Learning Topics

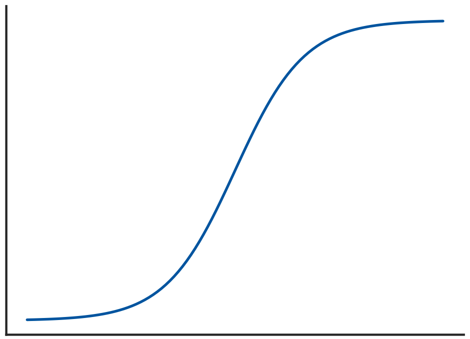
1. Introduction to ML
2. Probability Density Estimation
3. **Linear Discriminants**
4. Linear Regression
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics



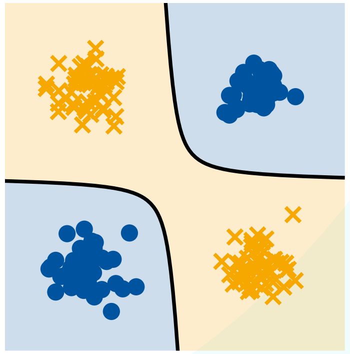
Linear Discriminant Functions

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n} (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

Least-Squares Classification



Activation Functions



Basis Functions

Recap: Generalized Linear Models

- So far: model classification by linear discriminant function

$$y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x} + w_0$$

- Generalize this with an activation function $g(\cdot)$:

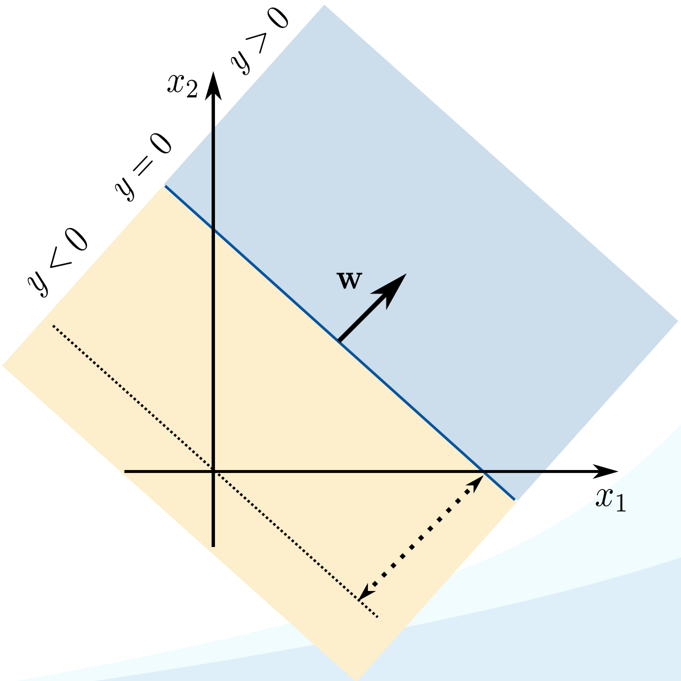
$$y(\mathbf{x}) = g(\mathbf{w}^\mathsf{T}\mathbf{x} + w_0)$$

- Remarks
  - $g(\cdot)$ may be non-linear.
  - Decision surfaces correspond to

$$y(\mathbf{x}) = \text{const} \iff \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = \text{const}$$

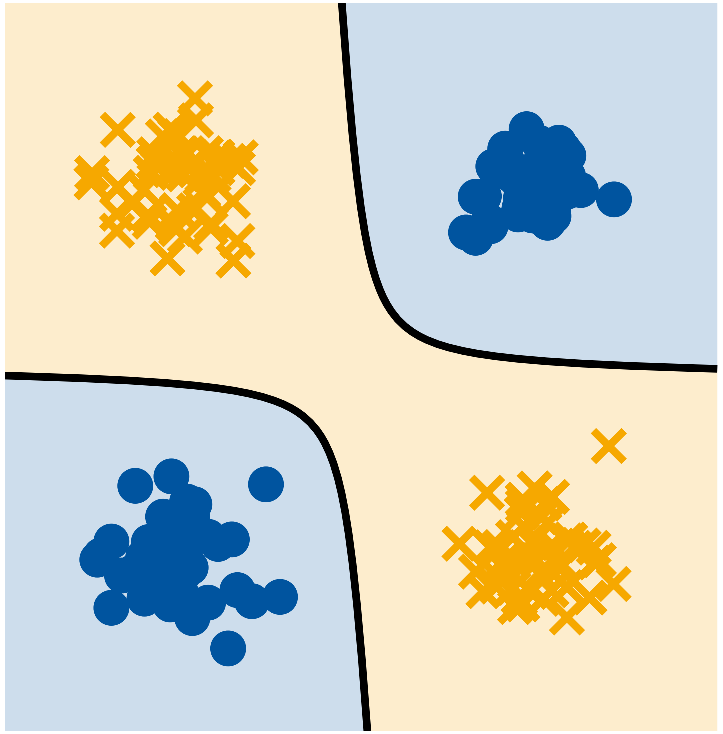
$\Rightarrow$ *If $g(\cdot)$ is monotonous (which is typically the case), the decision boundaries are still linear functions of $\mathbf{x}$.*

Generalized Linear Model



Linear Discriminants

1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. **Basis Functions**
6. Error Function Analysis

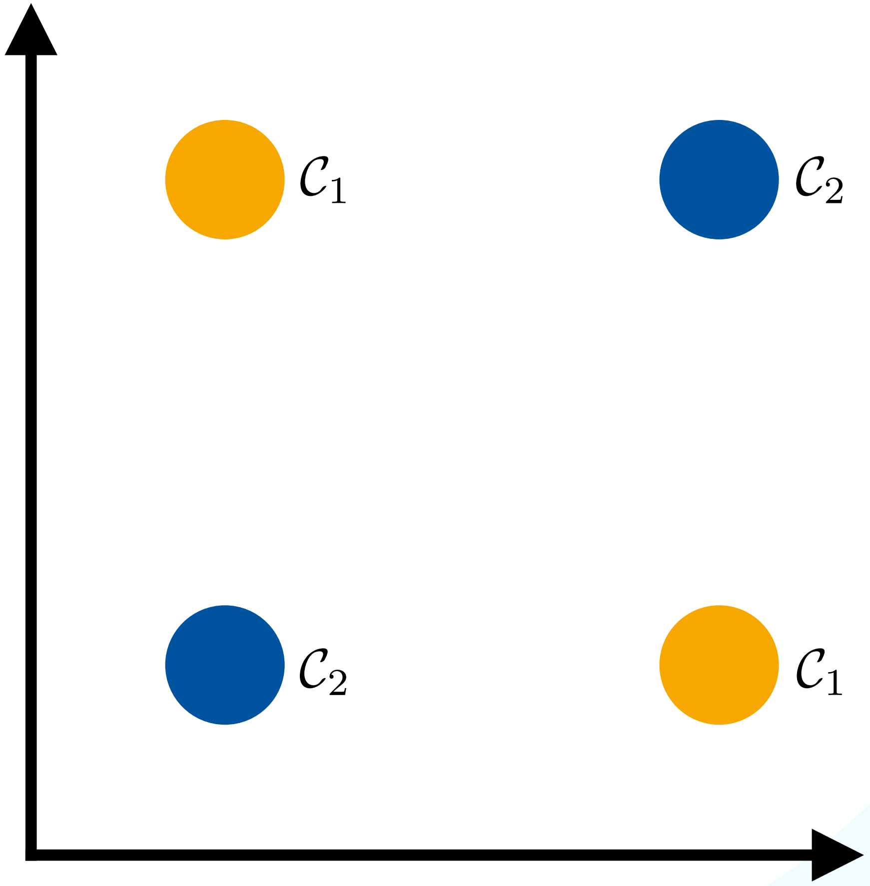


Basis Functions

- So far: assumed linear separability
  - Very restrictive assumption, classical counterexample: XOR
  - We need non-linear decision boundaries...
- Solution: use non-linear basis functions $\phi_j(\mathbf{x})$:

$$y(\mathbf{x}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) + w_0$$

- By choosing the right ϕ, every continuous function can (in principle) be approximated with arbitrary accuracy



Intuition

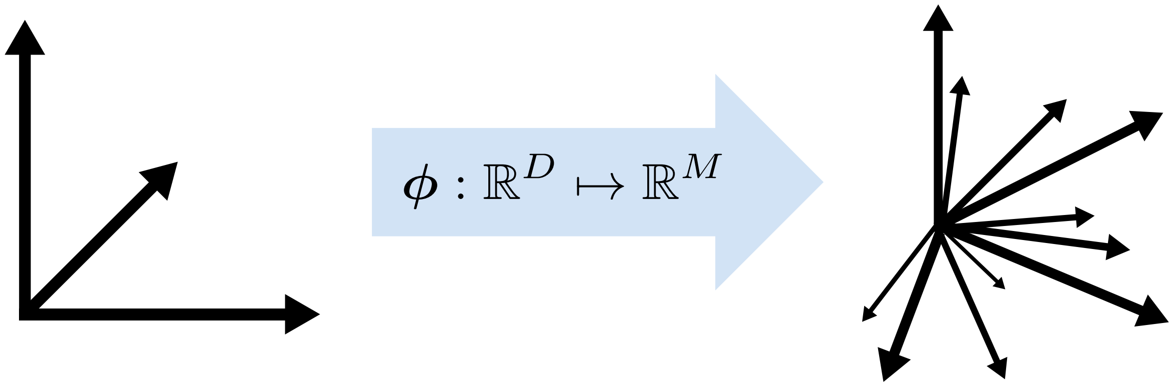
$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) = \mathbf{w}_k^{\mathsf{T}}\phi(\mathbf{x})$$

This is still a linear problem in $\boldsymbol{\phi}(\mathbf{x})$.

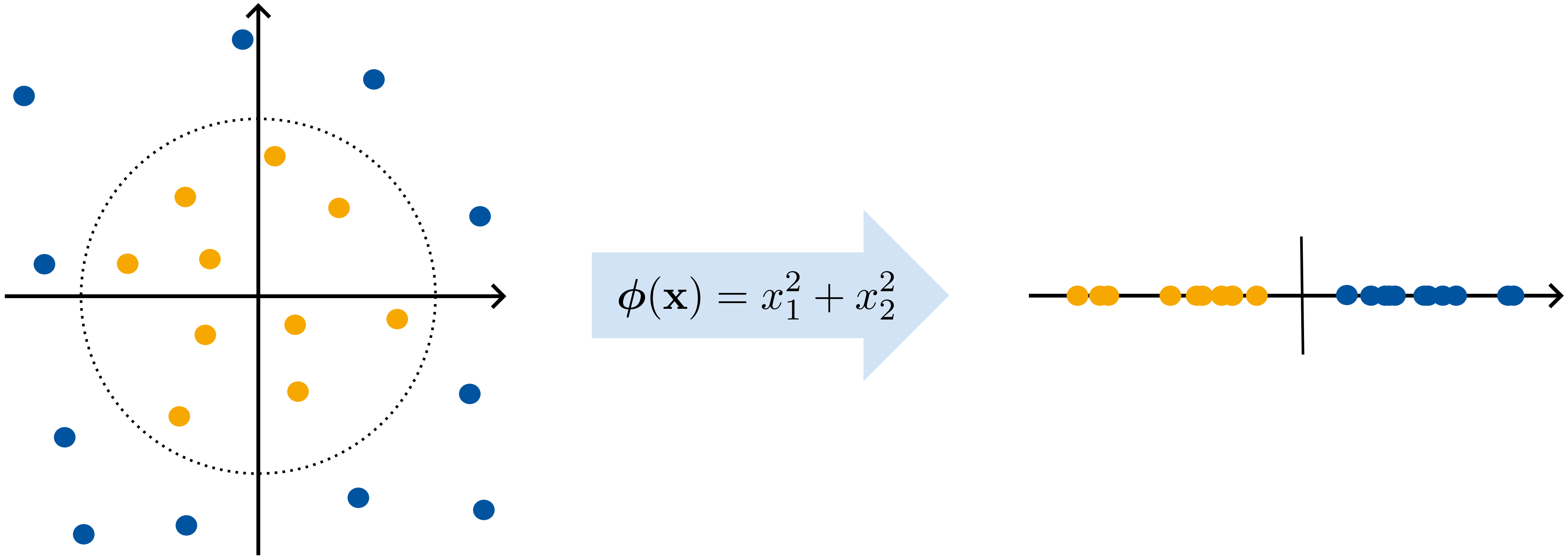
$\phi_j(\mathbf{x})$ are called basis functions.

But, depending on $\phi(\cdot)$, it may now be a nonlinear problem in $\mathbf{x}$.

Typically, $\phi_0(\mathbf{x}) = 1$ so that w_0 acts as a bias.



Usually, ϕ maps into a higher-dimensional space.



Not linearly separable

Linearly separable

Example: Polynomial Basis Functions

- Polynomial basis functions map x to powers of x:

$$\phi(x) = (x^m, x^{m-1}, \ldots, x, 1)^\mathsf{T}$$

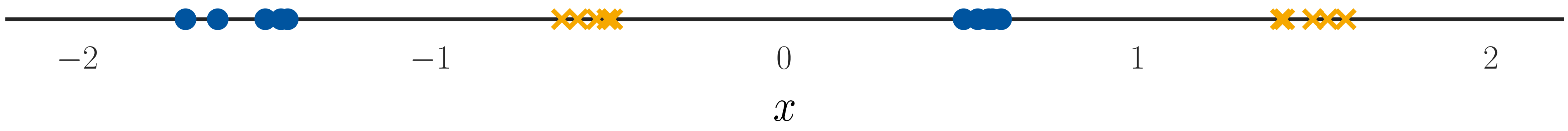
- When we optimize $\mathbf{w}^\mathsf{T}\phi(x)$ with polynomial basis functions, we implicitly optimize the coefficients of a polynomial in x:

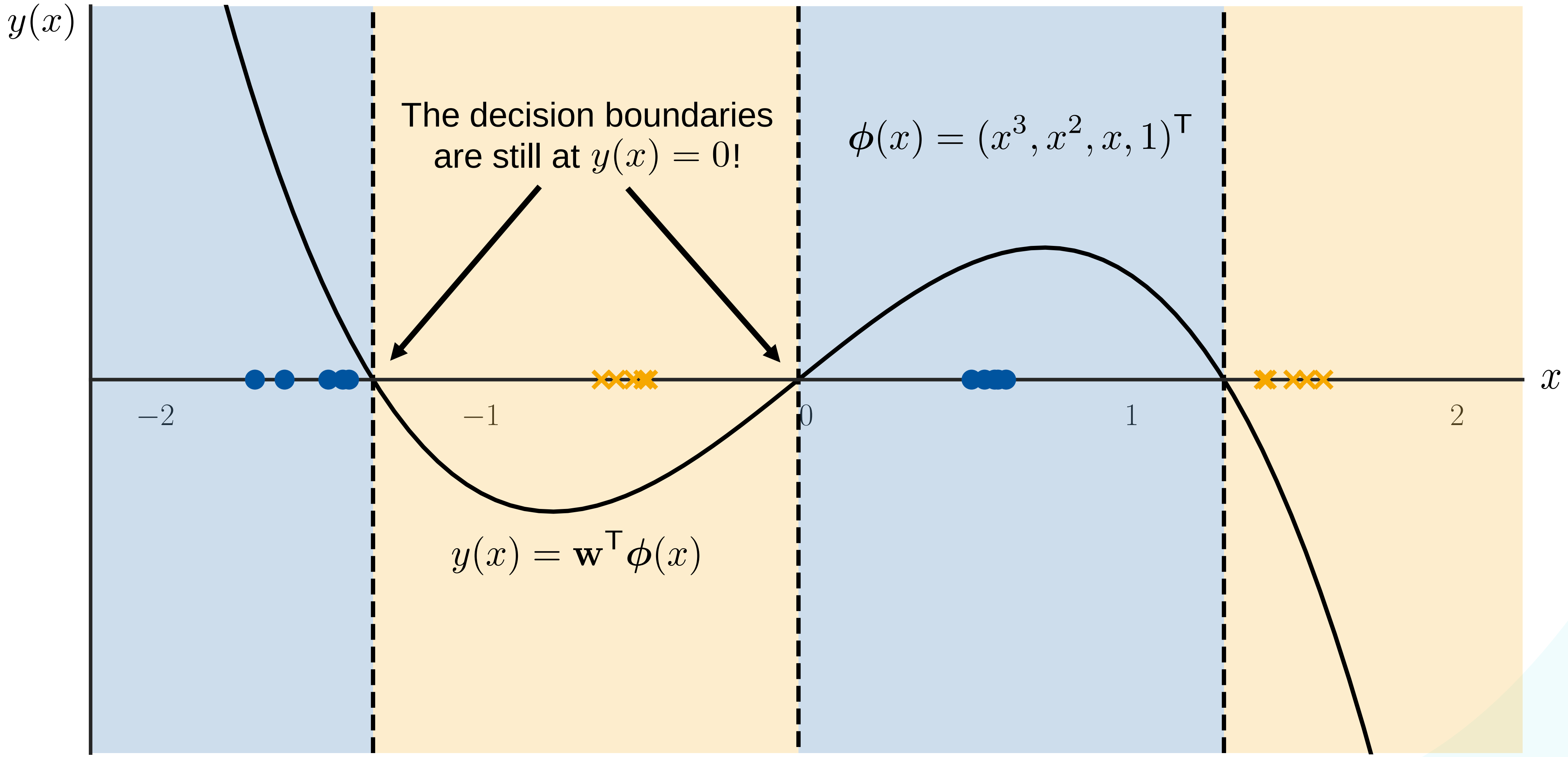
$$y(x) = \mathbf{w}^\mathsf{T}\phi(x)$$

$$= w_m x^m + w_{m-1}x^{m-1} + \ldots + w_1 x + w_0$$

- As before, we decide for $\mathcal{C}_1$ if $y(x) > 0$.

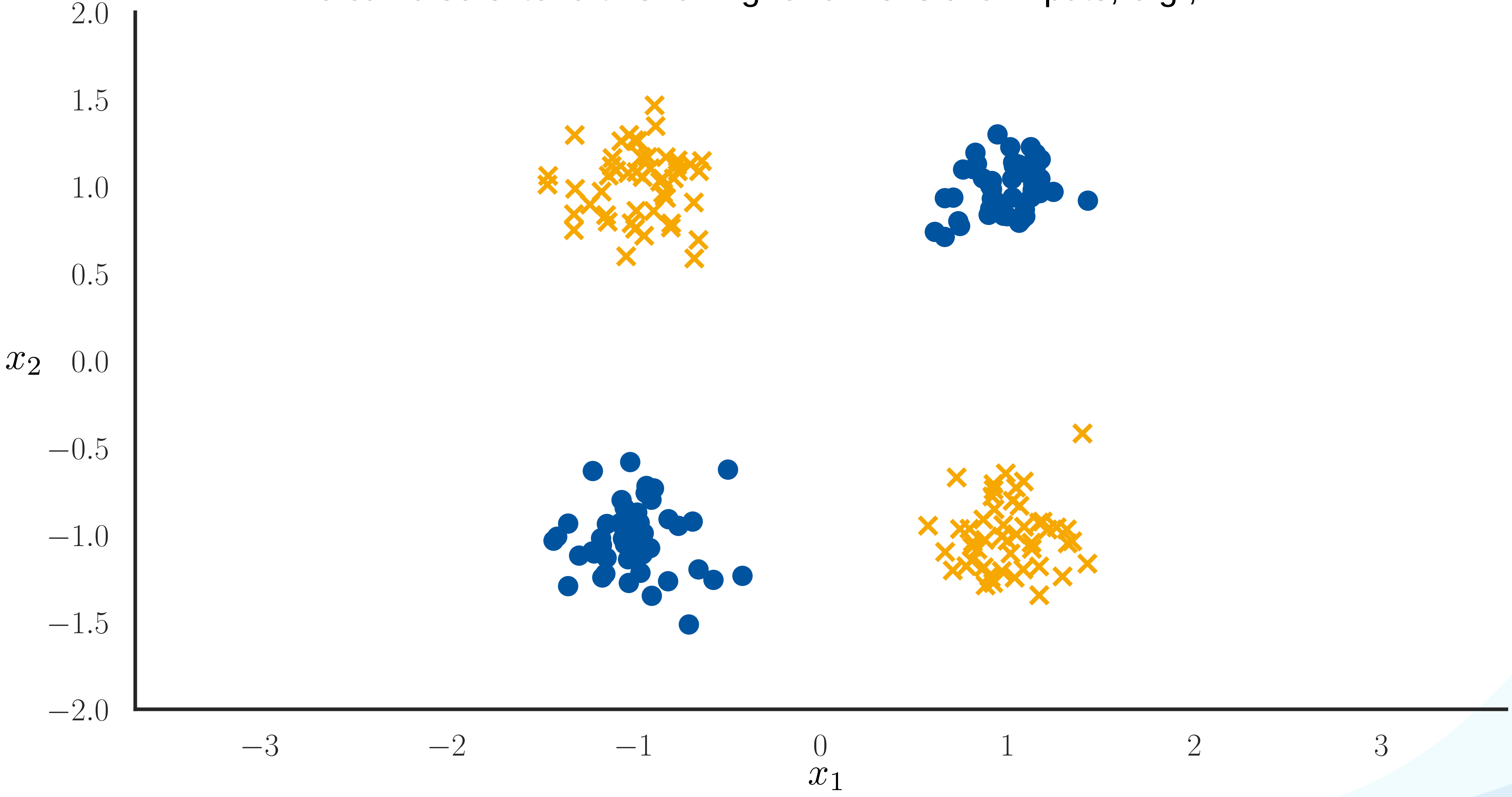
Let's use a third-degree polynomial: $\phi(x) = (x^3, x^2, x, 1)^\mathsf{T}$



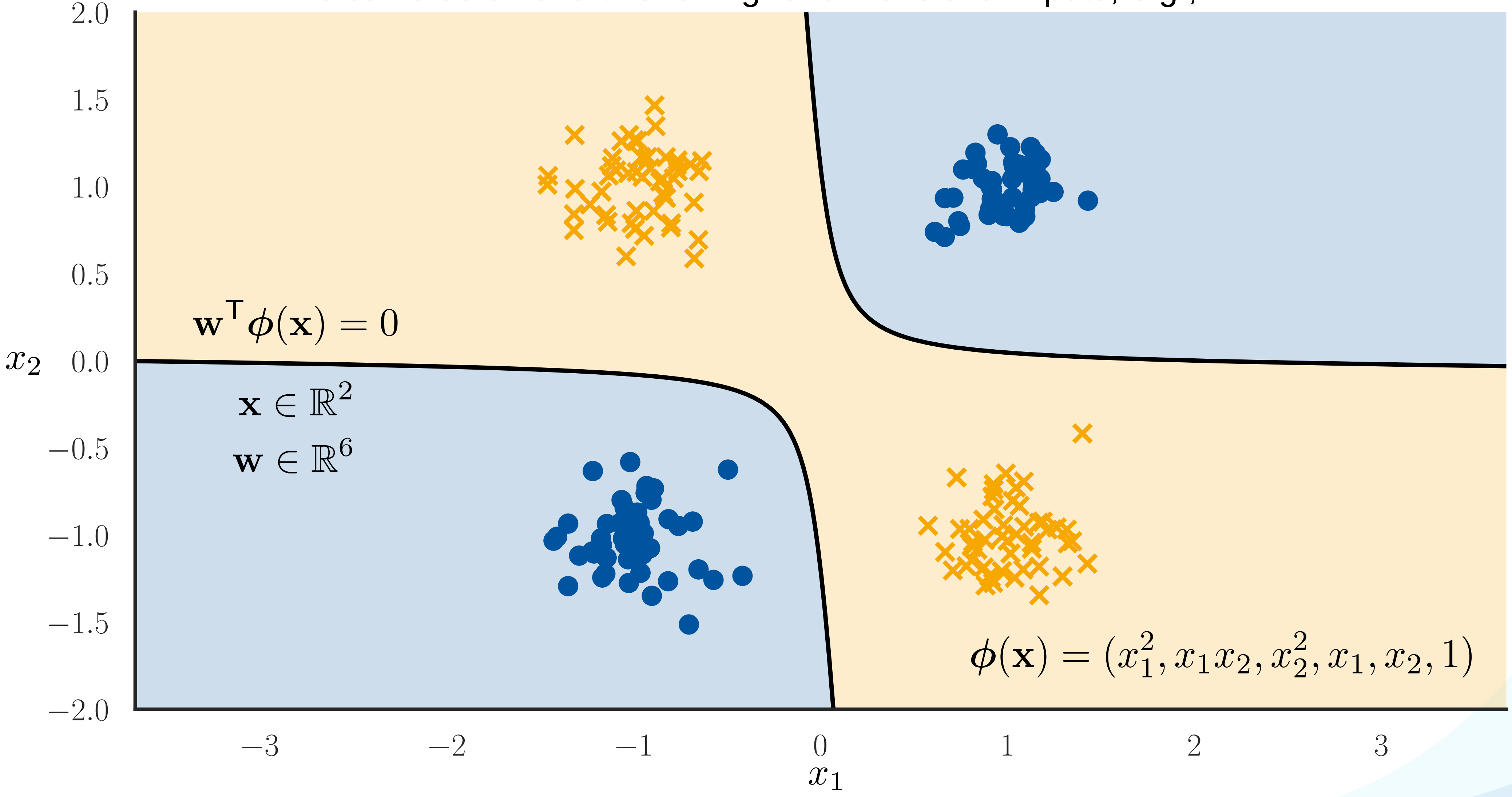


$y(x)$
The decision boundaries are still at $y(x) = 0$!
$\phi(x) = (x^3, x^2, x, 1)^\mathsf{T}$
x
-2
-1
0
1
2
$y(x) = \mathbf{w}^\mathsf{T}\phi(x)$

We can also extend this for higher dimensional inputs, e.g., 2D:



We can also extend this for higher dimensional inputs, e.g., 2D:



Discussion: Basis Functions

Advantages

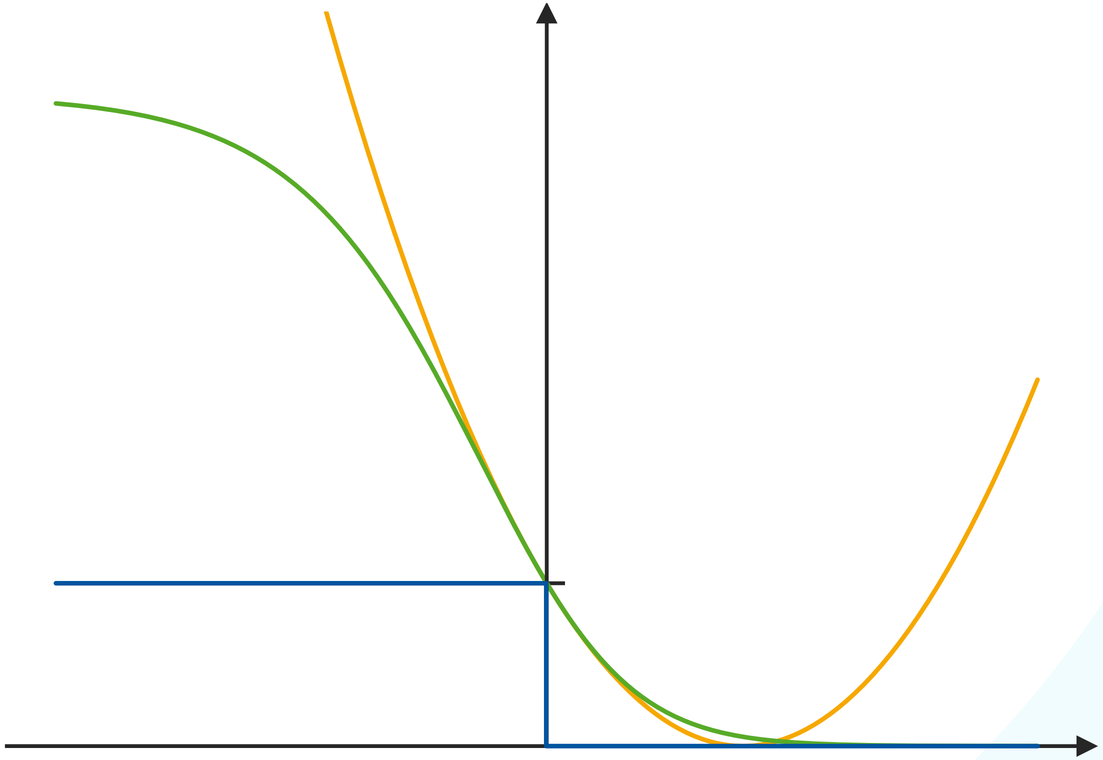
- Basis functions allow us to address linearly non-separable problems
- The problem is still linear in $\phi(\mathbf{x})$ (but may be nonlinear in $\mathbf{x}$).
- We can think of $\phi(\mathbf{x})$ as transforming the data into a feature space in which the problem is easier to solve.
- In general, it is easier to find a separating hyperplane in higher-dimensional spaces.

Limitations

- The right choice of $\phi(\mathbf{x})$ depends on the problem and is another hyperparameter to optimize.
- Flexibility is limited by the curse of dimensionality. Evaluating $\mathbf{w}^\mathsf{T}\phi(\mathbf{x})$ can be expensive in high-dimensional spaces.
- Choosing a higher-dimensional feature space $\phi(\mathbf{x})$ increases the capacity of the classifier and may lead to overfitting.

Linear Discriminants

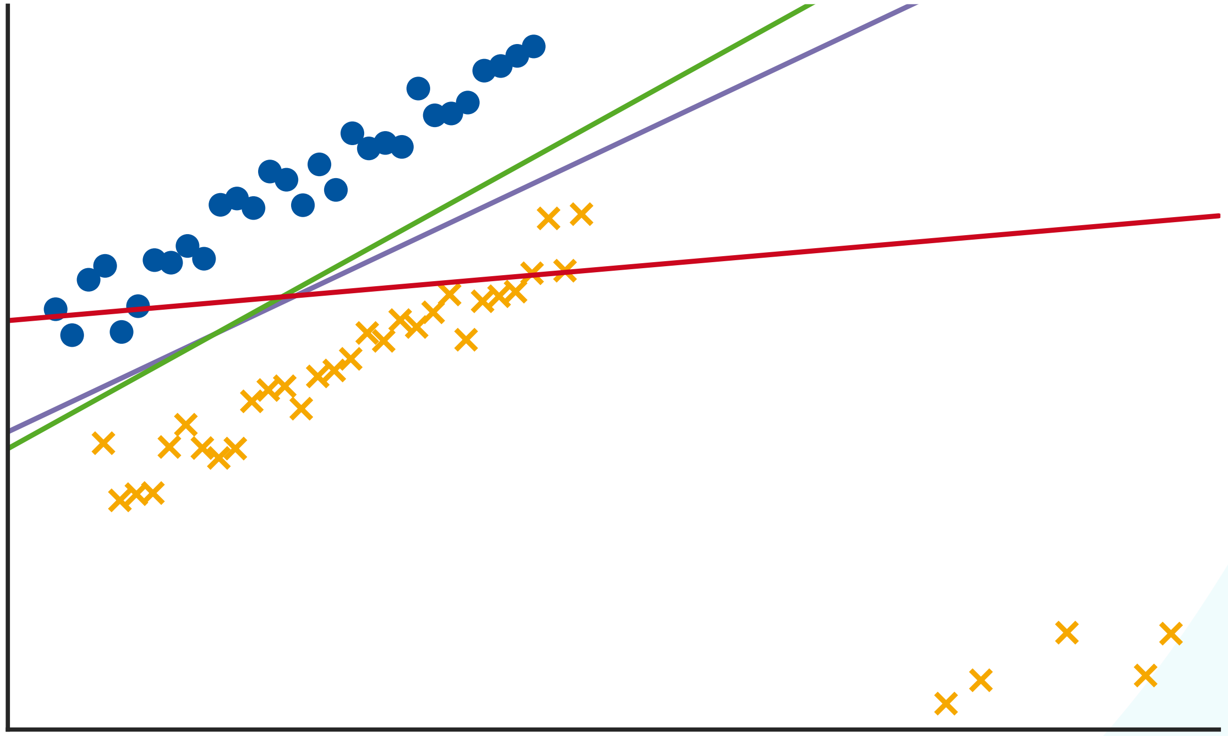
1. Motivation: Discriminant Functions
2. Linear Discriminant Functions
3. Least-Squares Classification
4. Generalized Linear Discriminants
5. Basis Functions
6. **Error Function Analysis**



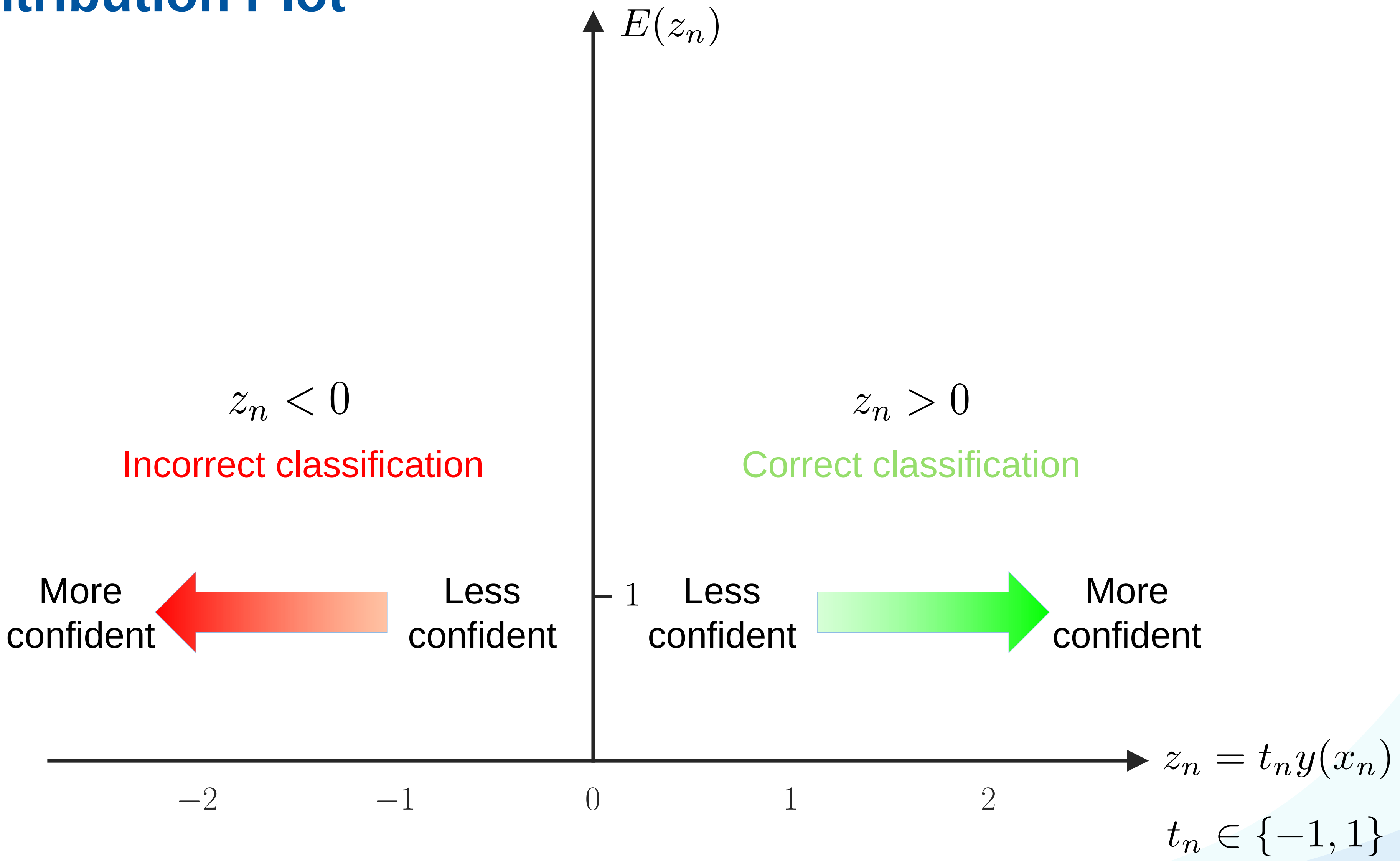
Error Function Analysis

- We have seen how to learn generalized linear discriminant models by optimizing an error function.
  - We observed problems with Least-Squares Classification based on the squared error function.
  - In particular, sensitivity to outliers
  - Can we predict when such problems will occur?

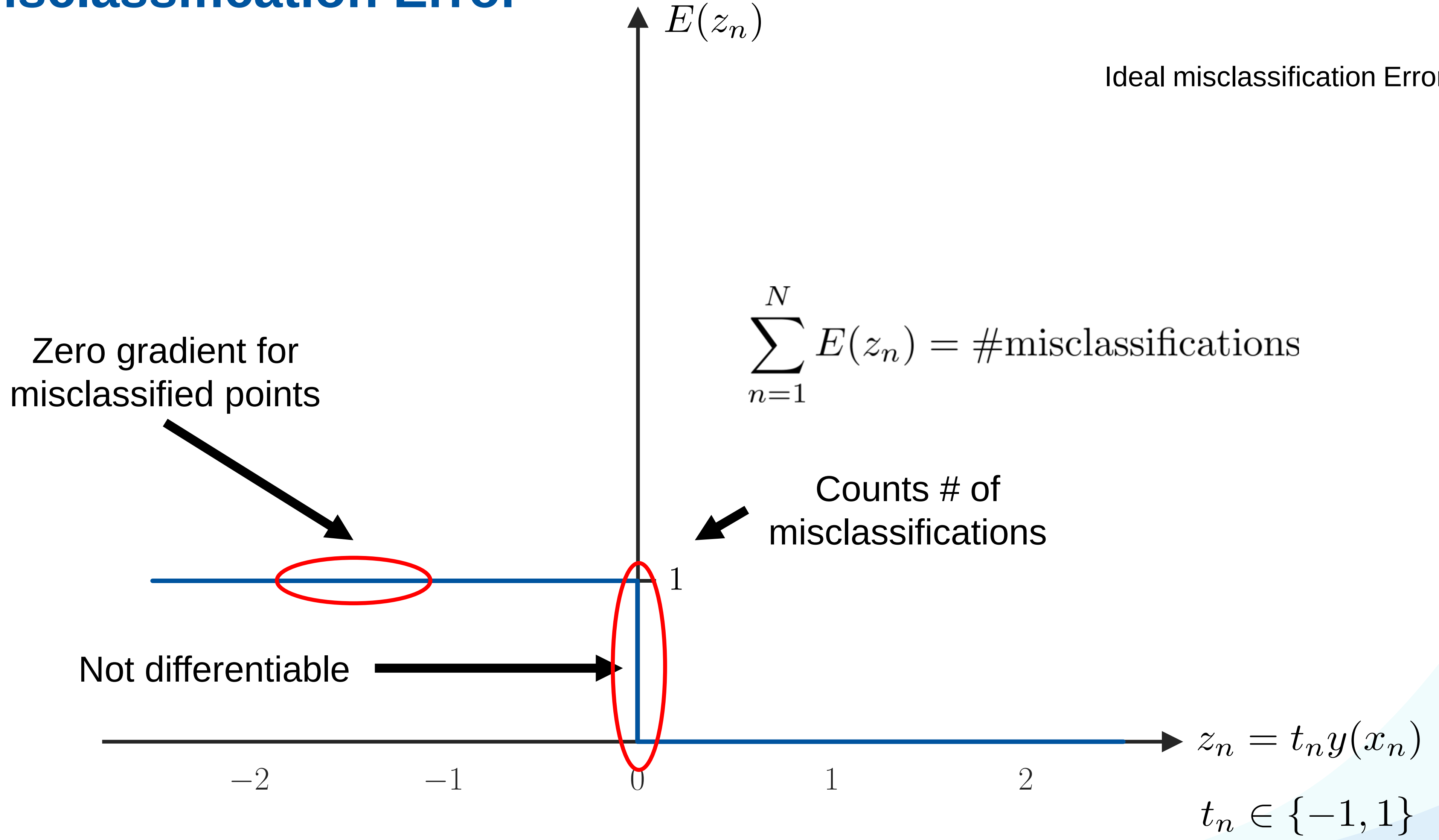
- *Let's analyze the behavior of error functions in more detail…*



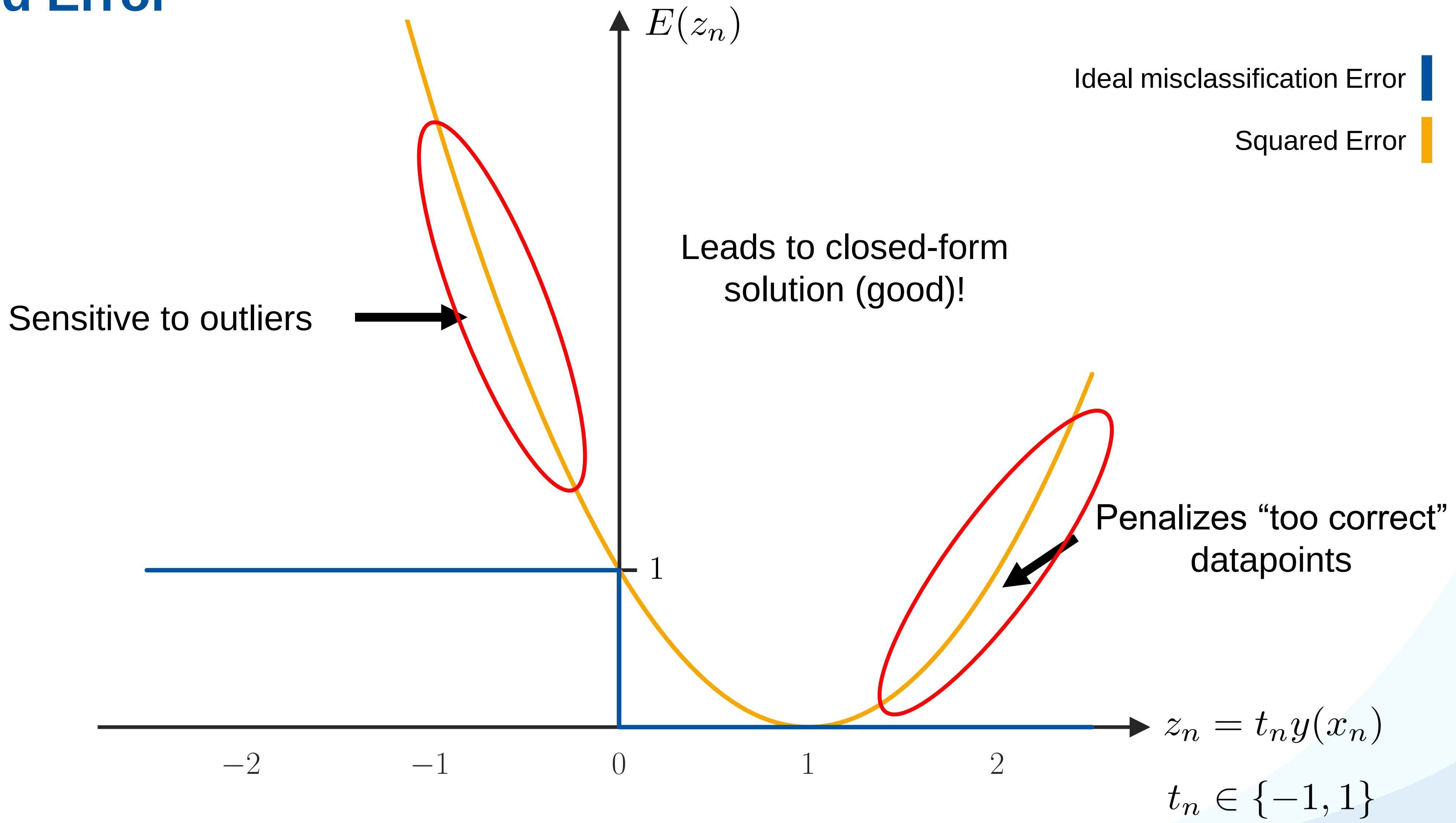
Error Contribution Plot



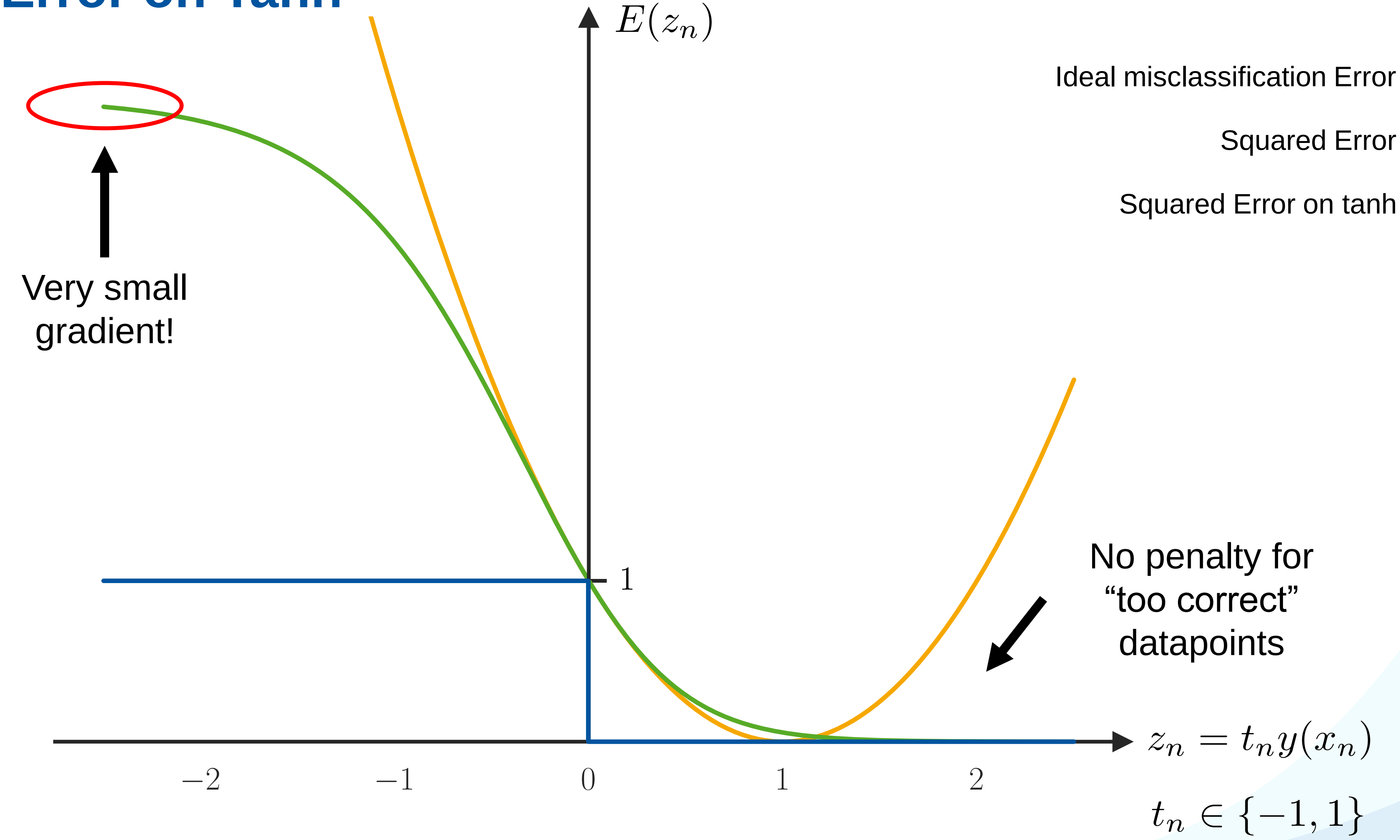
Ideal Misclassification Error



Squared Error

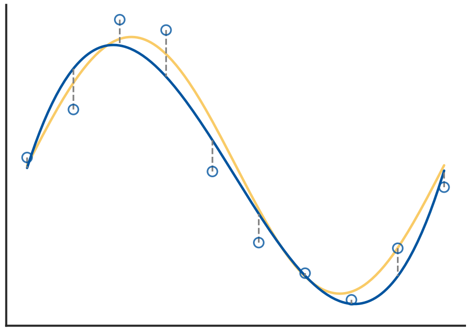


Squared Error on Tanh



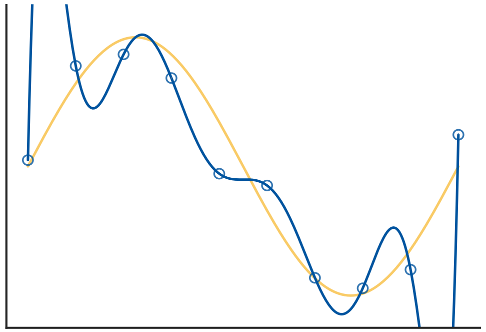
Machine Learning Topics

1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. **Linear Regression**
5. Logistic Regression
6. Support Vector Machines
7. AdaBoost
8. Neural Network Basics

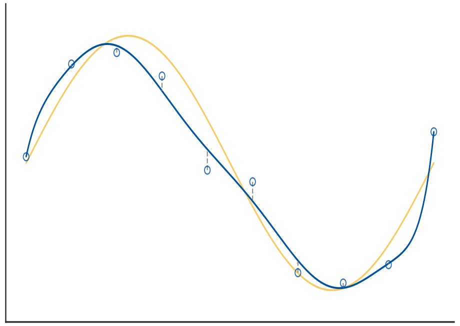


$$y(\mathbf{x}) = \mathbf{w}^{\top}\phi(\mathbf{x}) + w_0$$

Linear Regression Functions

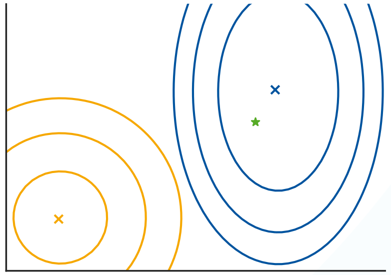


Overfitting



$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

Regularization

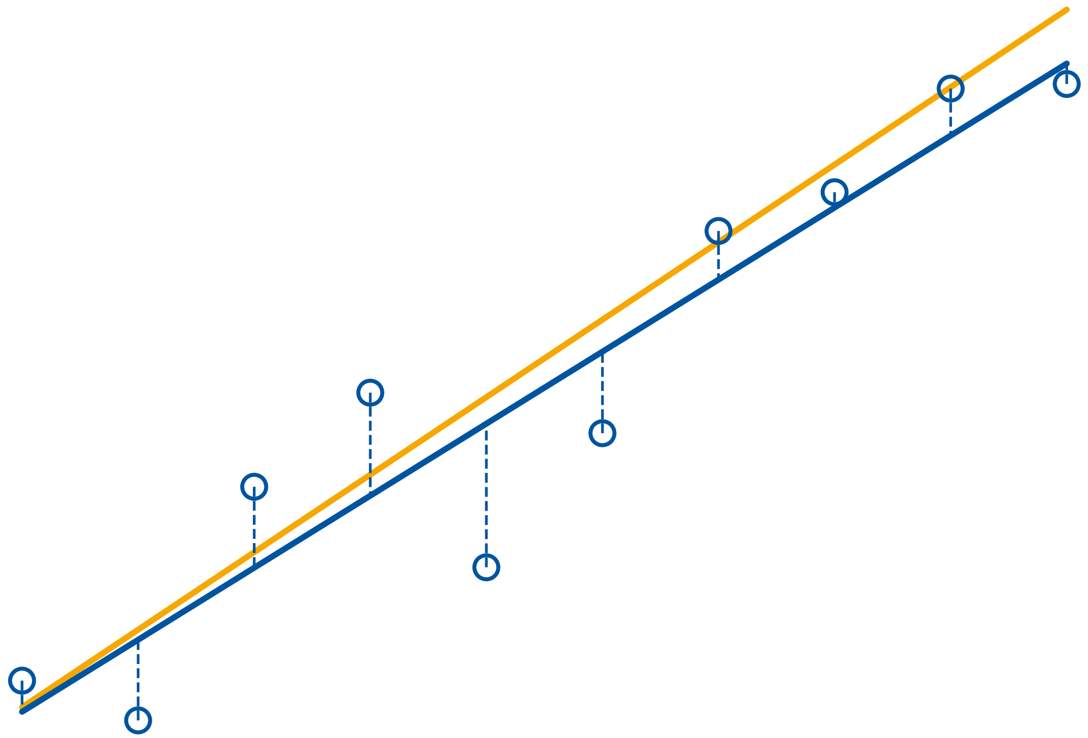


$$\mathbf{w} = (\mathbf{\Phi}^{\top}\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^{\top}\mathbf{t}$$

Ridge Regression

Linear Regression

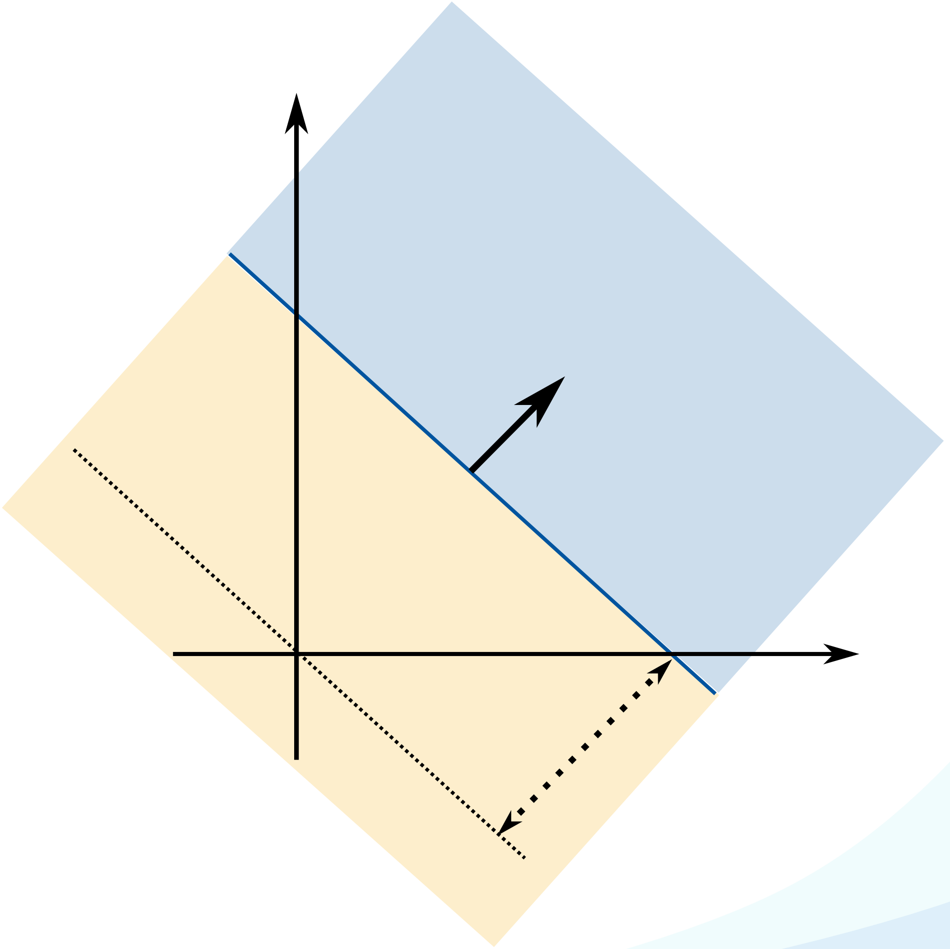
1. **Motivation**
2. Least-Squares Regression
3. Regularization
4. Ridge Regression
5. The Bias-Variance Tradeoff



Motivation: Linear Regression

- We have seen how to build classifiers with linear functions:

$$y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) + w_0$$



Motivation: Linear Regression

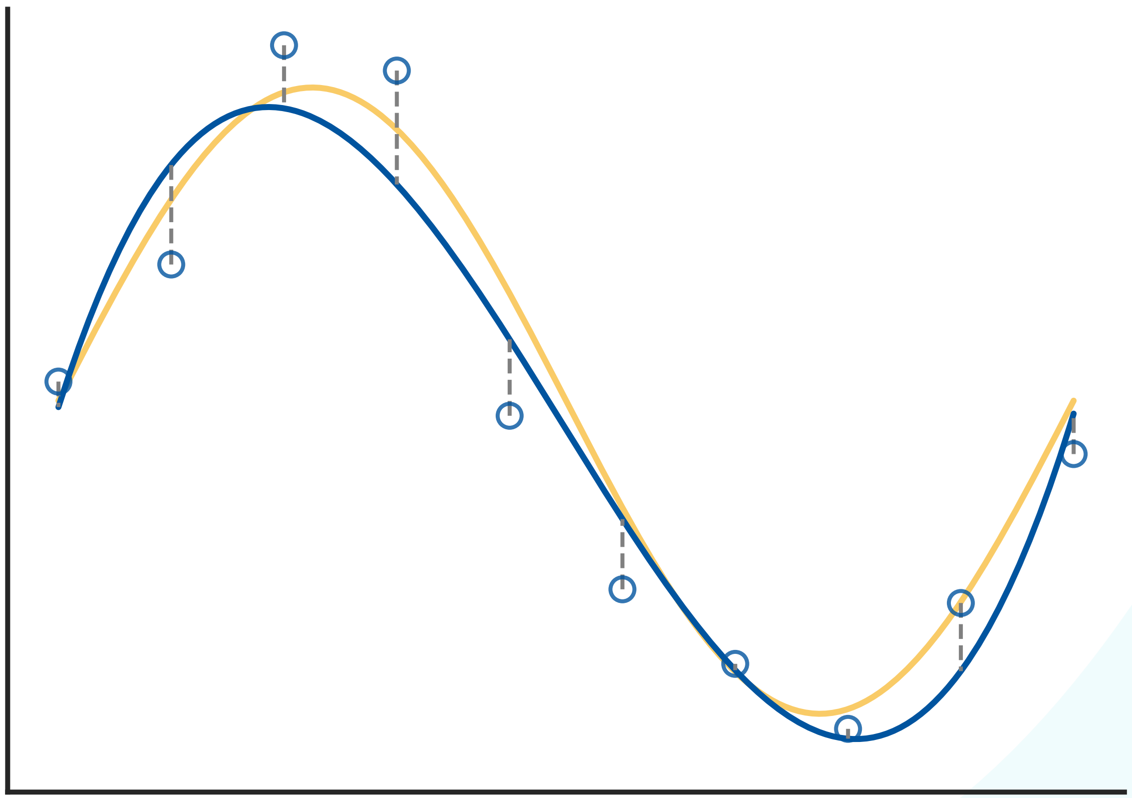
- We have seen how to build classifiers with linear functions:

$$y(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) + w_0$$

- Now, we will use this model to estimate arbitrary functions using real-valued labels $t_n \in \mathbb{R}$.
- Key assumption: data is generated by some function $h(\mathbf{x})$ with Gaussian noise:

$$t_n = h(\mathbf{x}) + \epsilon$$

- This model is called linear regression.



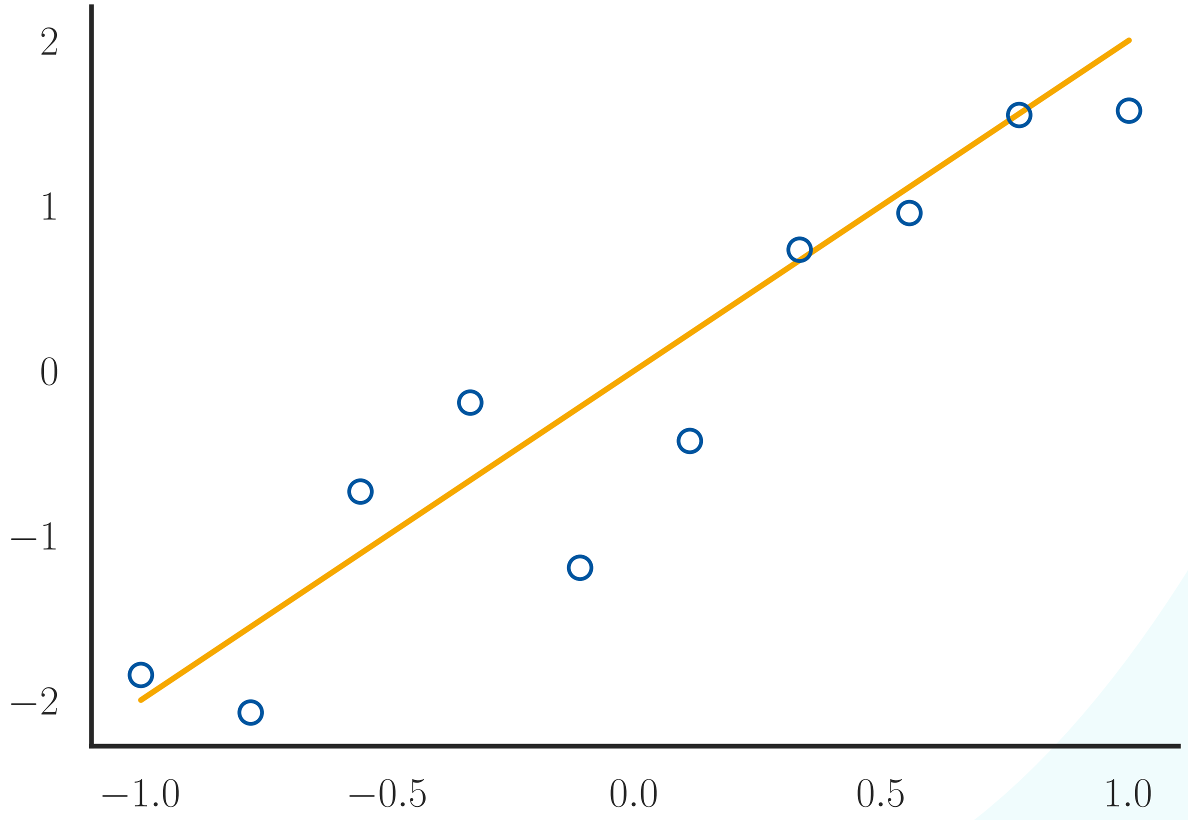
Example: Linear target functions

$$y(x) = w_1 x + w_0$$

- We assume ground truth is a linear function:

$$f(x) = mx + b$$

- We try to find a line that minimizes the distance to the samples.



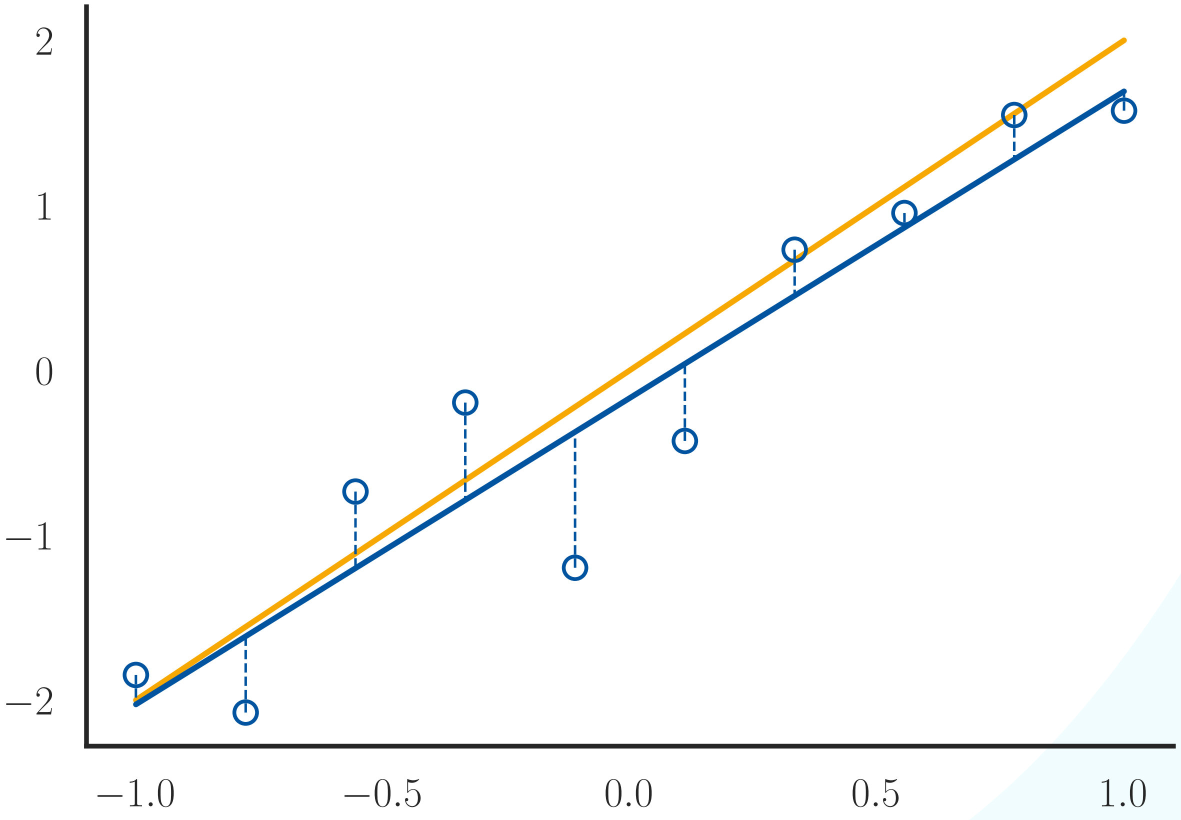
Example: Linear target functions

$$y(x) = w_1 x + w_0$$

- We assume ground truth is a linear function:

$$f(x) = mx + b$$

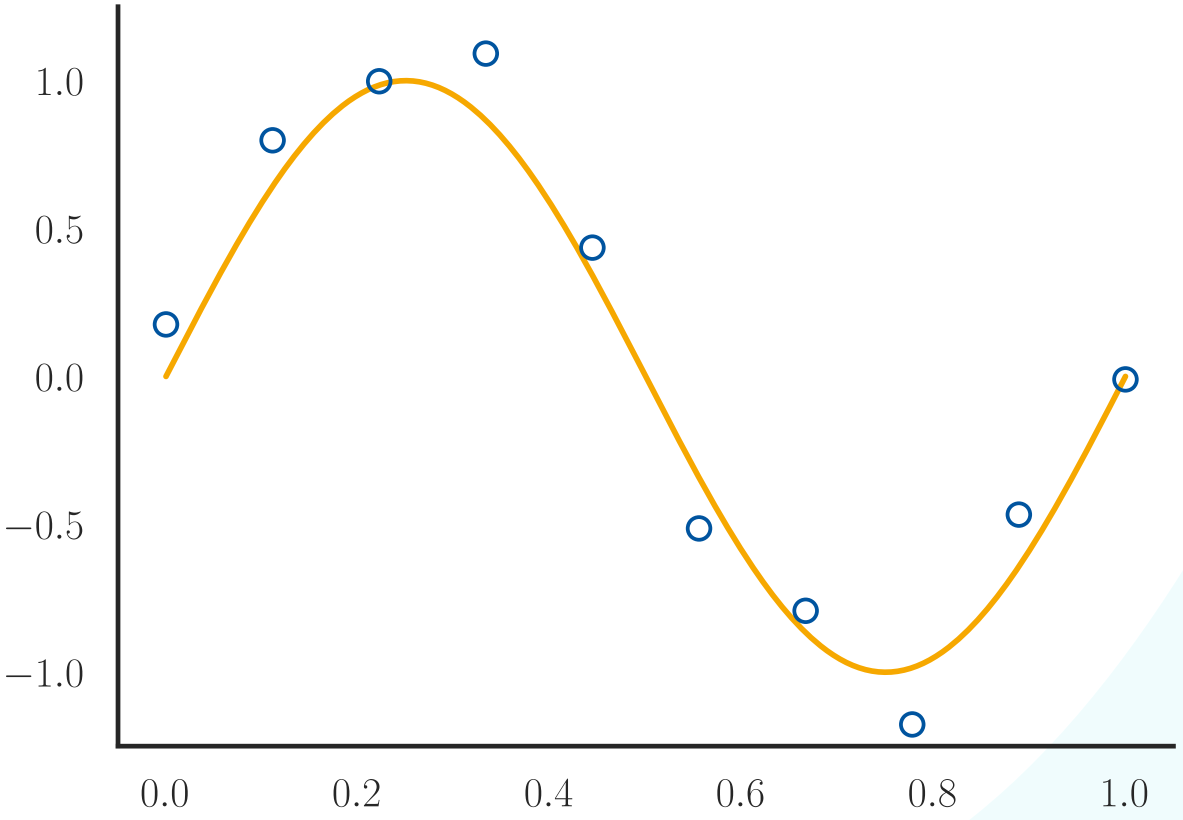
- We try to find a line that minimizes the distance to the samples.



Example: Non-linear target functions

$$y(x) = \mathbf{w}^{\mathsf{T}} \phi(x) + w_0$$

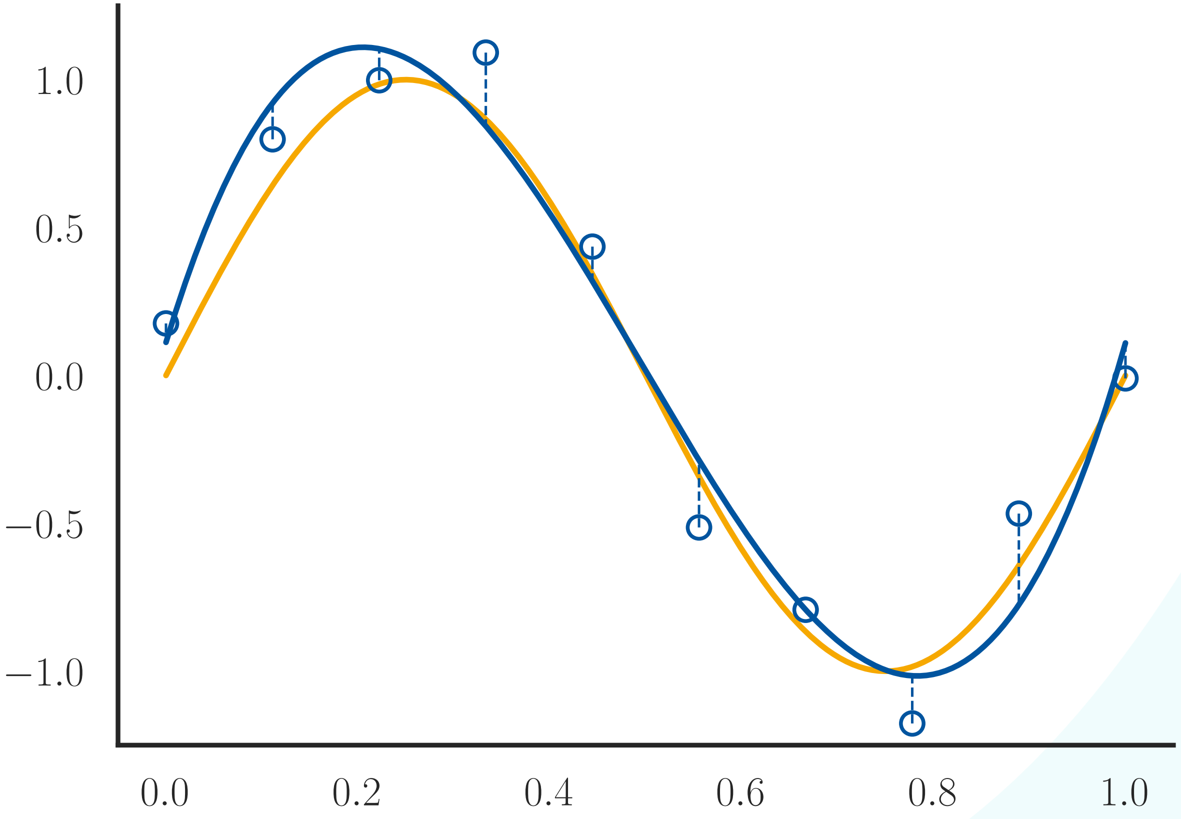
- We can use basis functions to fit arbitrary functions $f(\mathbf{x})$.
- For example, polynomial basis functions.



Example: Non-linear target functions

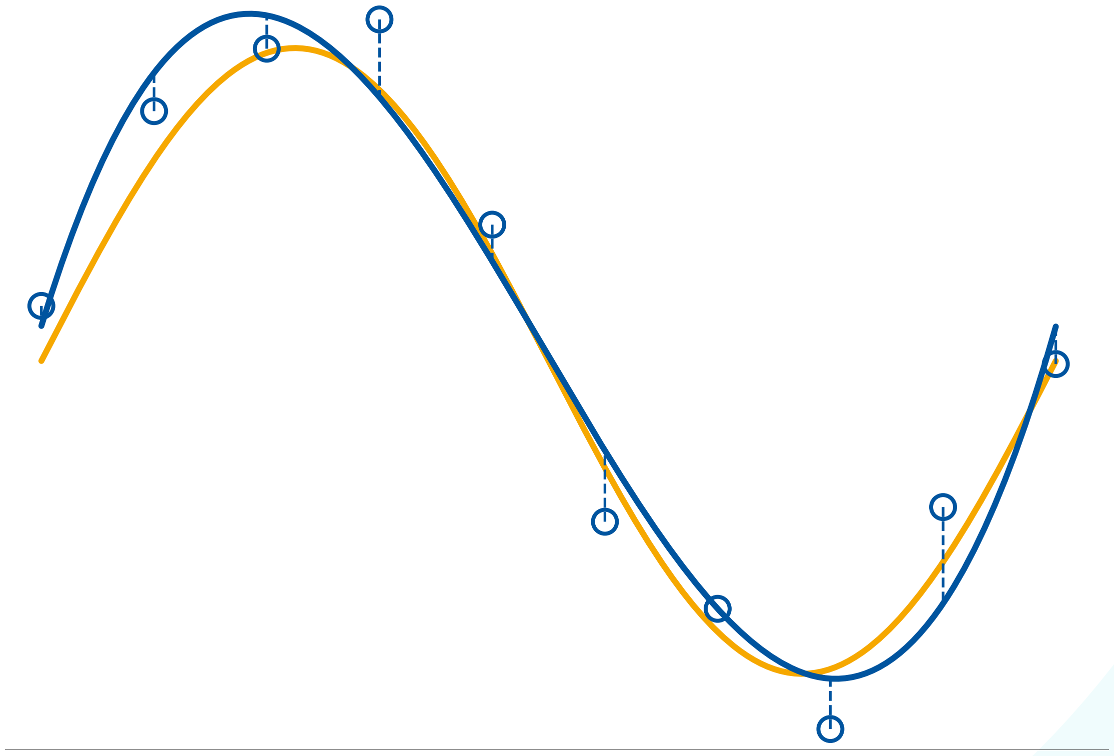
$$y(x) = \mathbf{w}^{\mathsf{T}}\phi(x) + w_0$$

- We can use basis functions to fit arbitrary functions $f(\mathbf{x})$.
- For example, polynomial basis functions.
- Finding a good set of basis functions usually requires some insight into the data...



Linear Regression

1. Motivation
2. **Least-Squares Regression**
3. Regularization
4. Ridge Regression
5. The Bias-Variance Tradeoff



Least-Squares Regression

- We want to optimize the difference between our predictor $y(\mathbf{x}_n; \mathbf{w})$ and the targets t_n.
- The only difference is that our targets t_n are now continuous values.
- Again, use the familiar squared error objective:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

- This has the same solution as for classification (normal equations).

$$y(\mathbf{x}_n; \mathbf{w}) = \mathbf{w}^\mathsf{T} \phi(\mathbf{x}_n)$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^{N} (\mathbf{w}^\mathsf{T} \phi_n - t_n) \phi_n$$

$$= \mathbf{\Phi}^\mathsf{T} (\mathbf{\Phi} \mathbf{w} - \mathbf{t}) \stackrel{!}{=} 0$$

$$\Rightarrow \mathbf{w} = (\mathbf{\Phi}^\mathsf{T} \mathbf{\Phi})^{-1} \mathbf{\Phi}^\mathsf{T} \mathbf{t}$$

Example: Fitting a polynomial

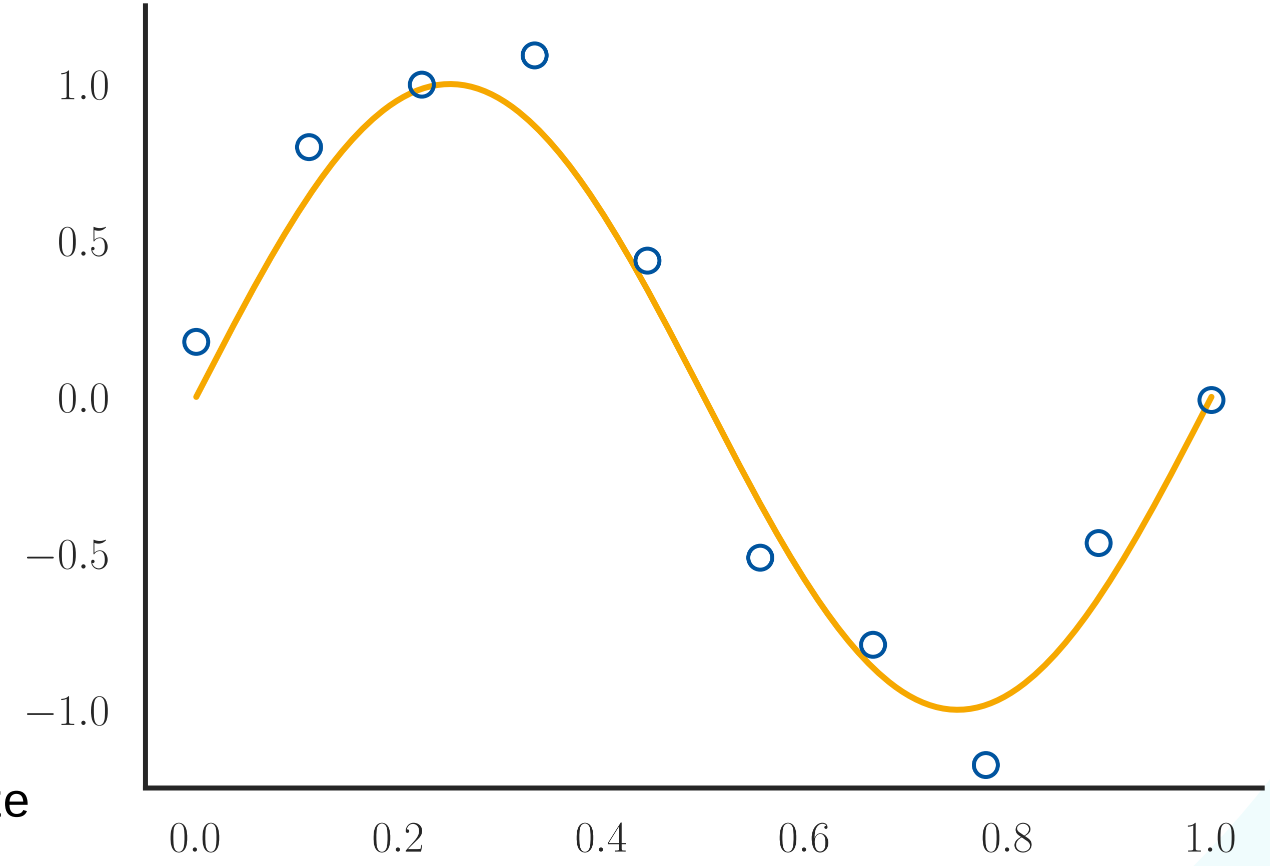
- This is clearly not a linear function.
- Let's use polynomial basis functions:

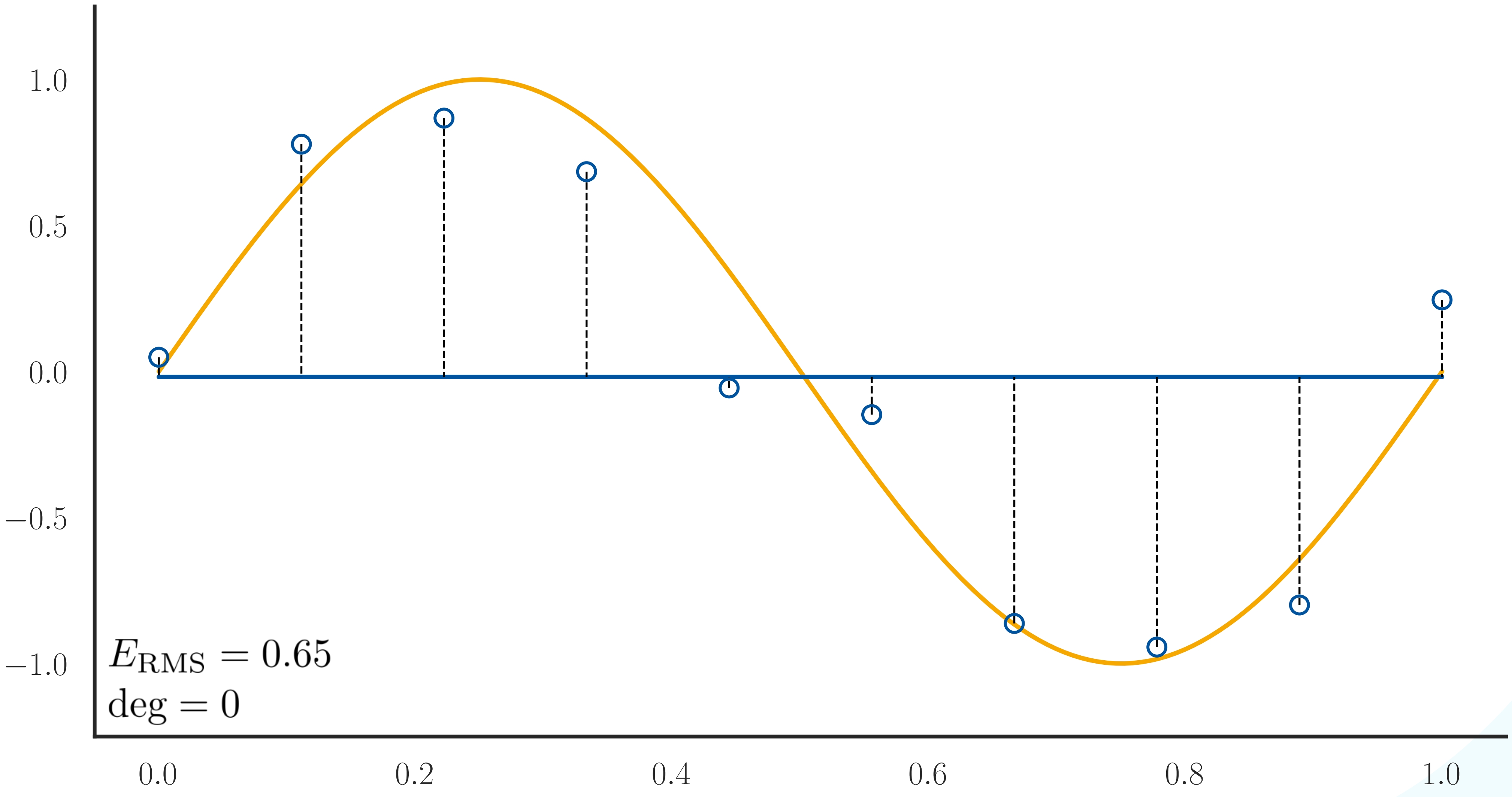
$$\phi_j(x) = (x^j)$$

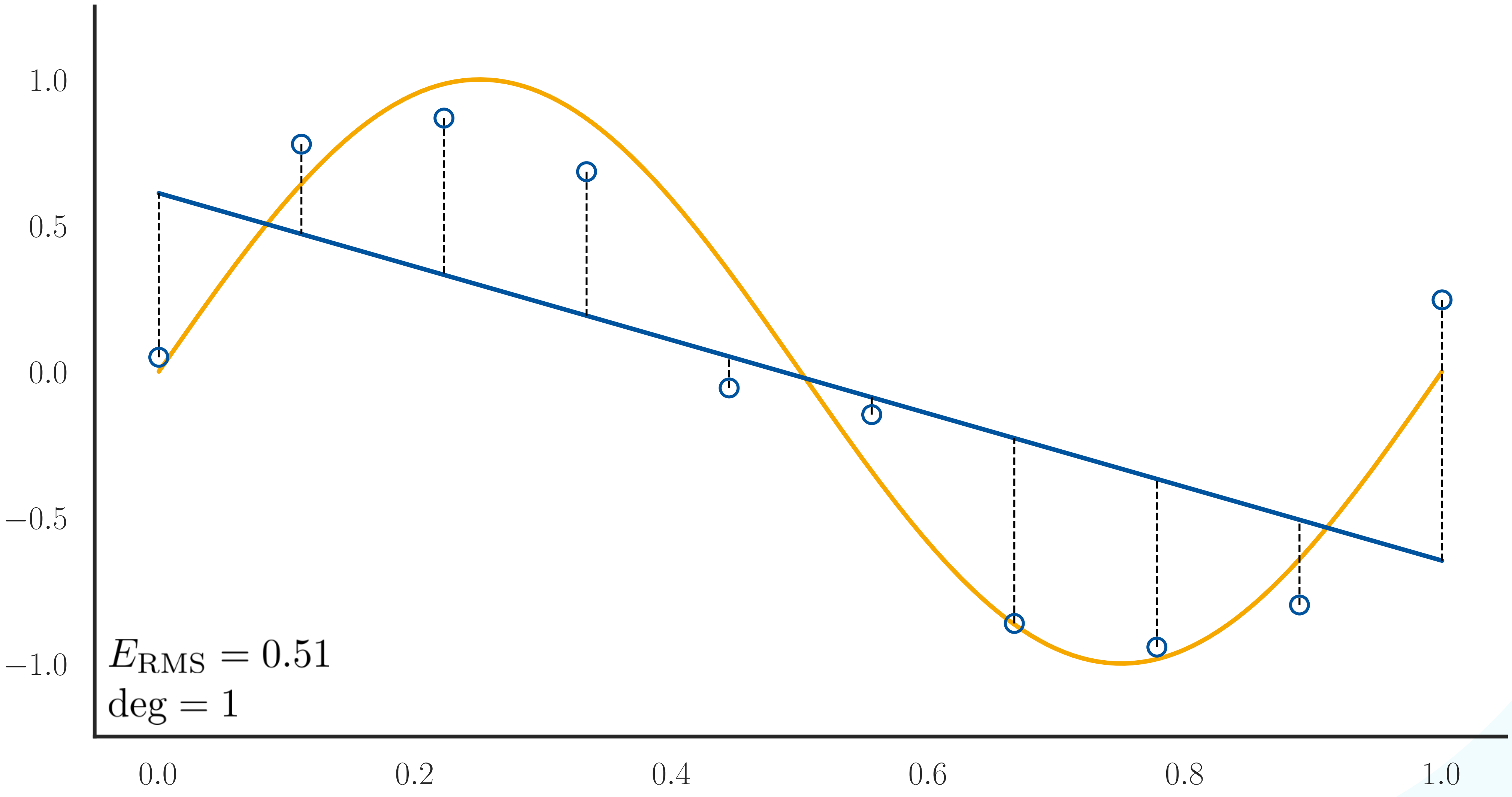
- Which degree should we use?
- Compare different models by their Root Mean Square Error:

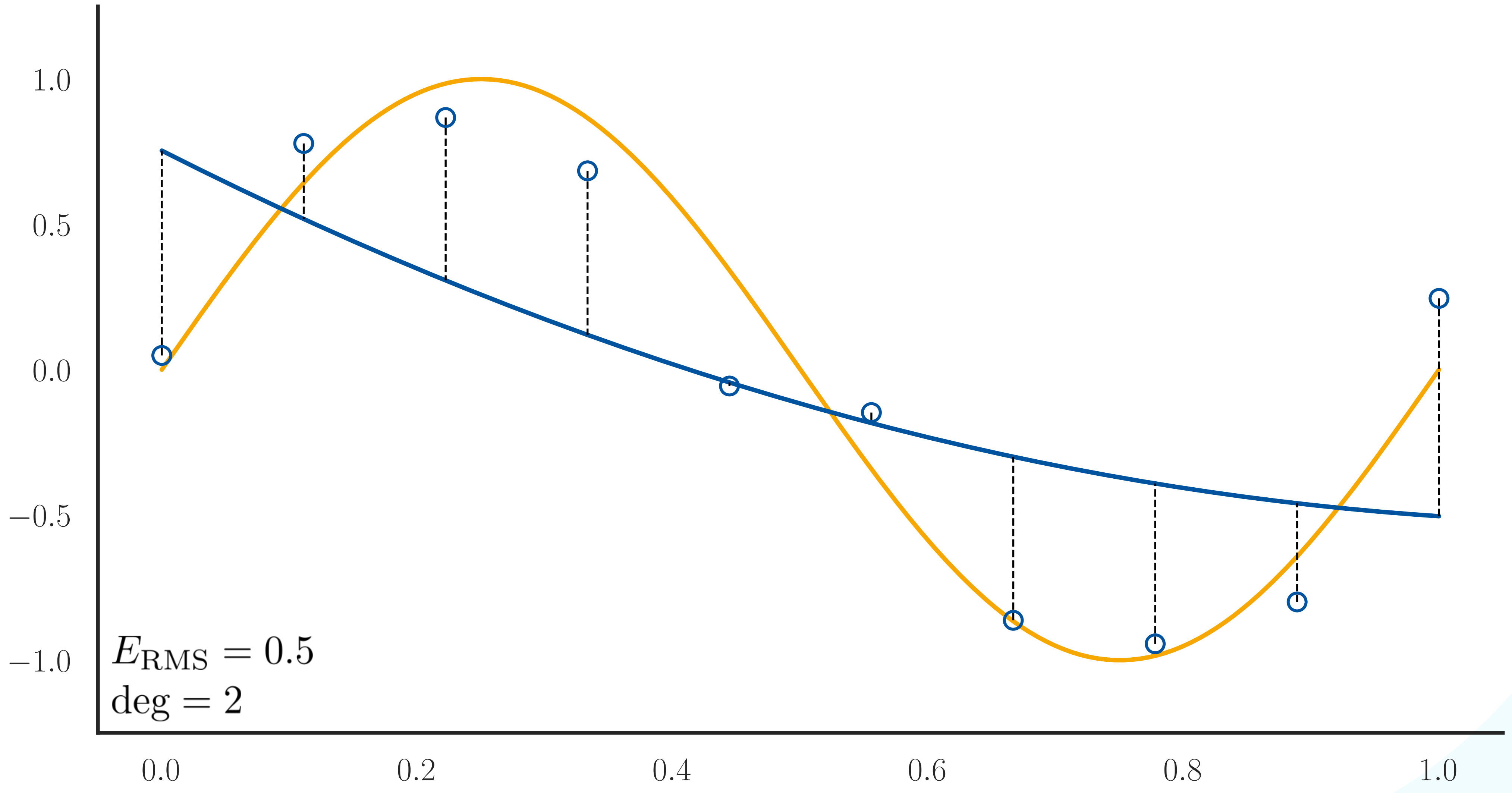
$$E_{\text{RMS}} = \sqrt{\frac{2E(\mathbf{w}^*)}{N}}$$

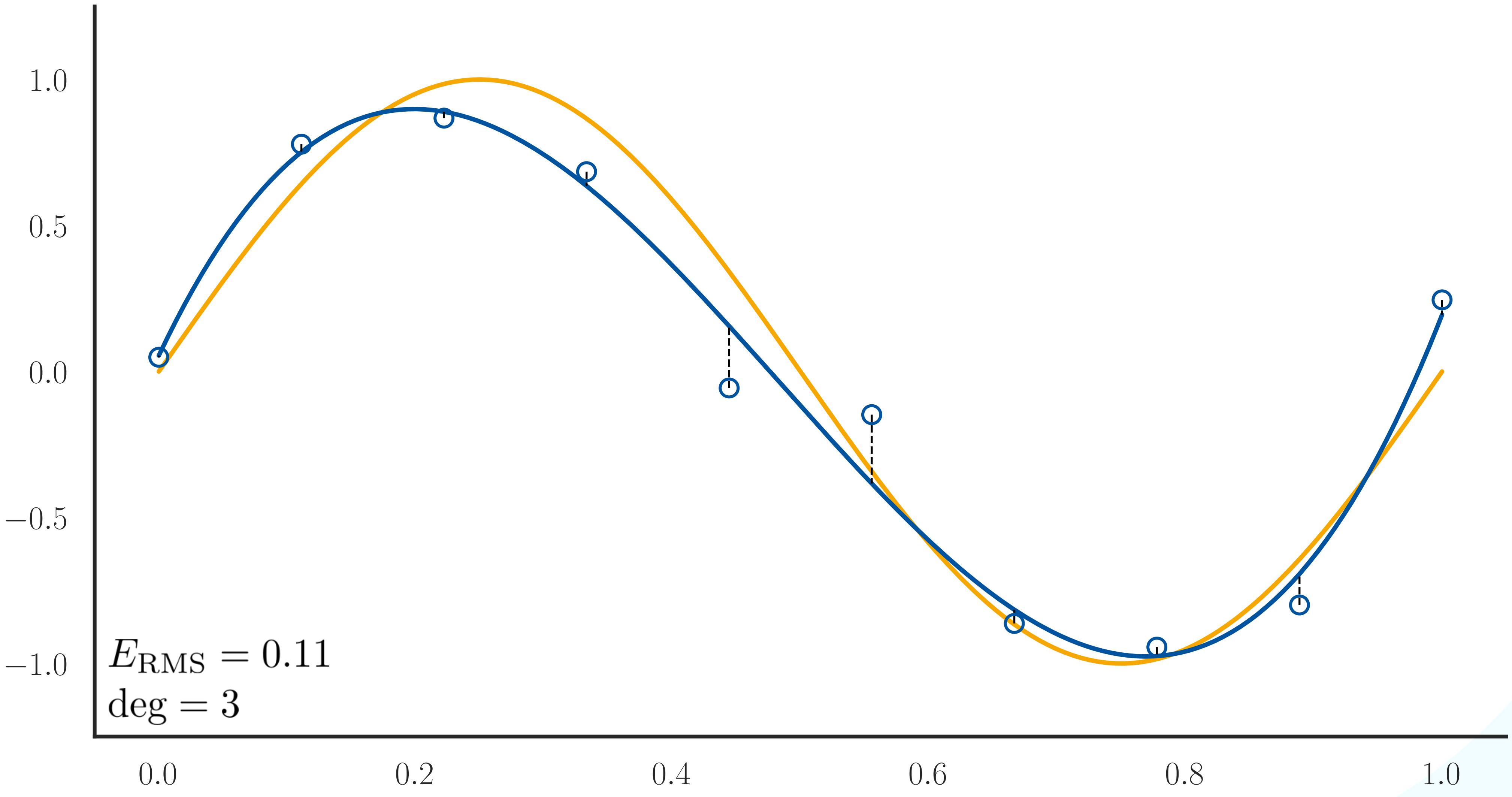
- RMS is independent of training set size and has the same scale as the data.

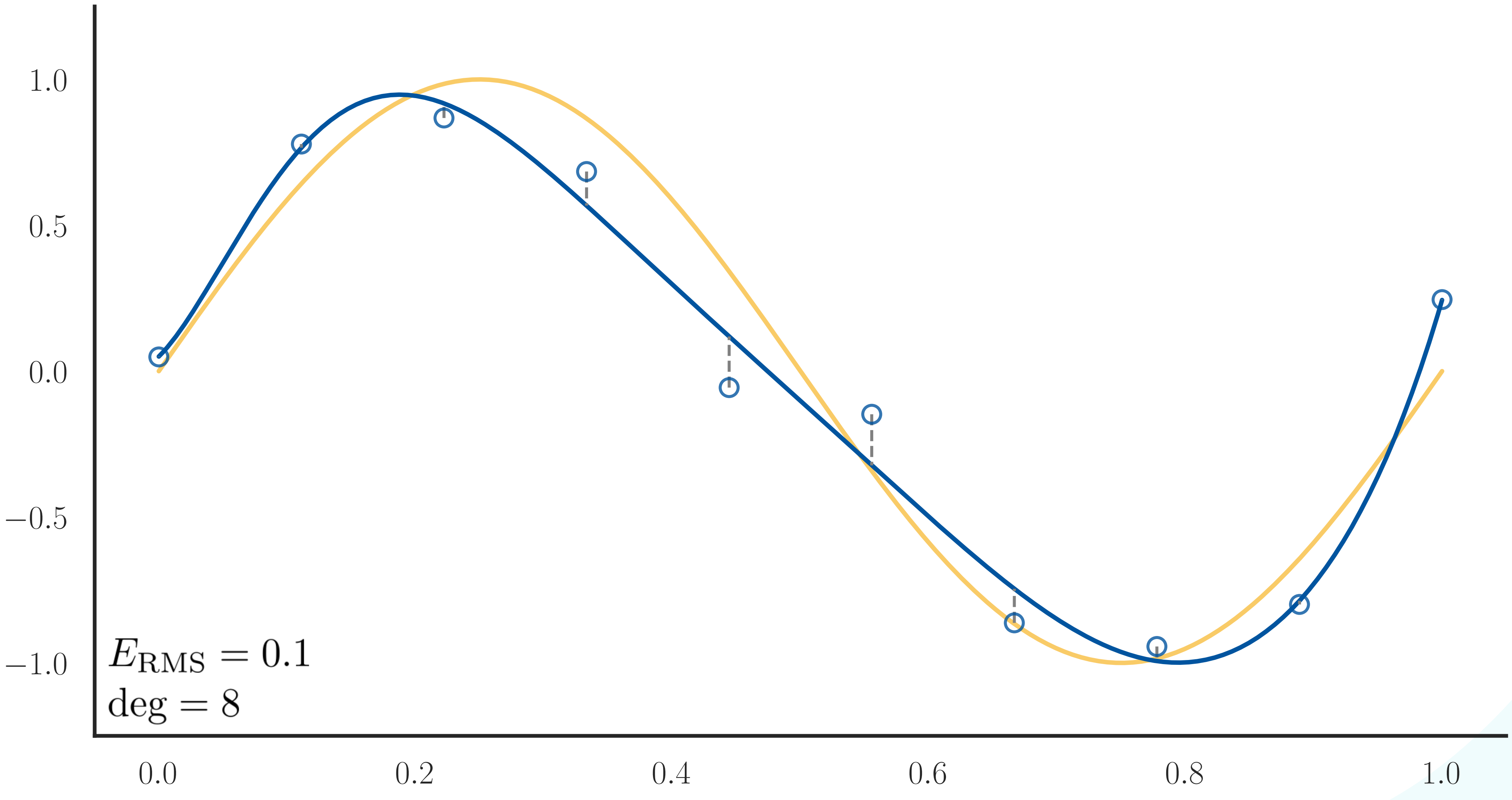


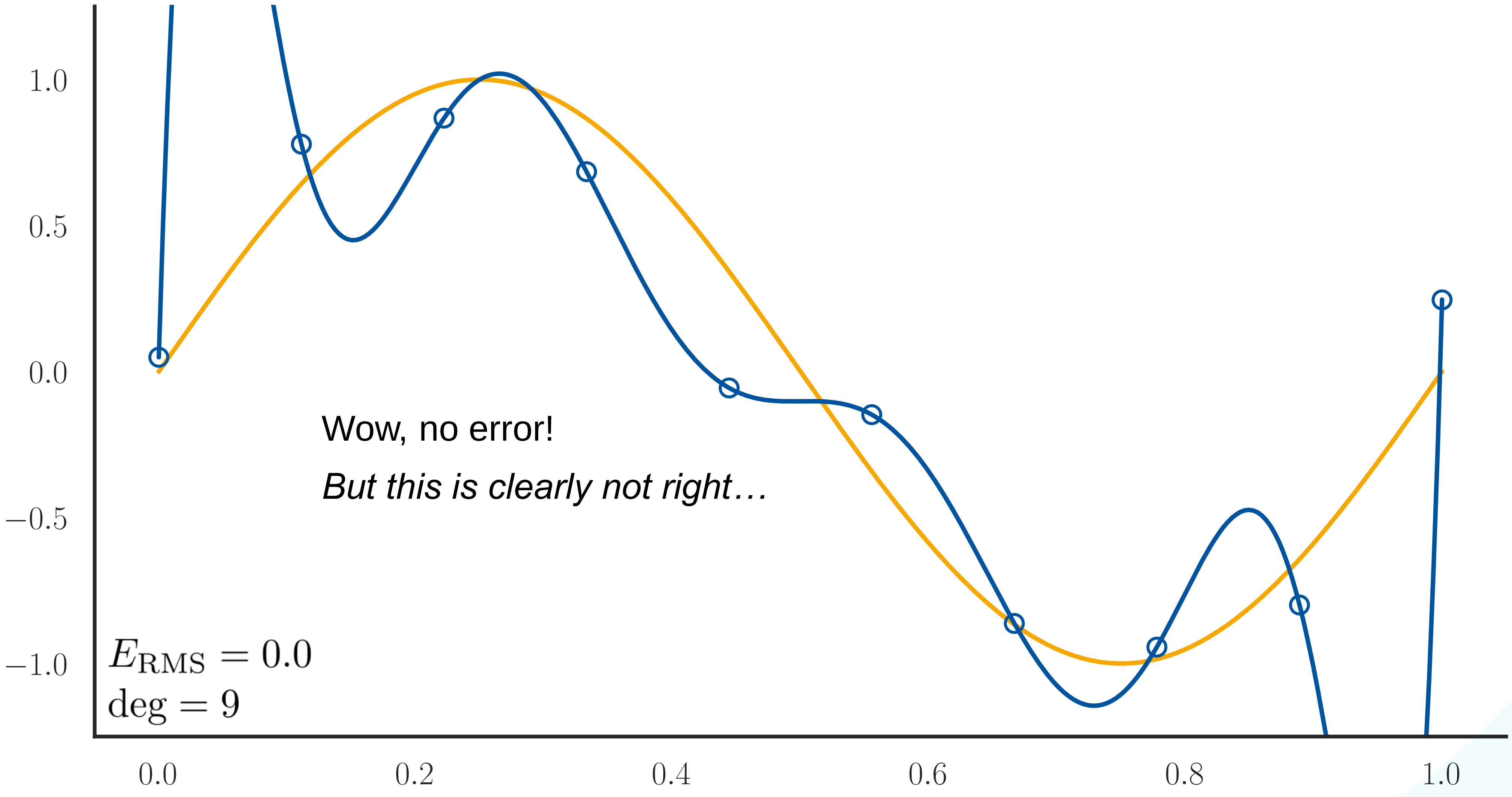

1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
$E_{\mathrm{RMS}} = 0.65$
$\mathrm{deg} = 0$


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
$E_{\text{RMS}} = 0.51$
deg = 1


1.0
0.5
0.0
−0.5
−1.0
$E_{\mathrm{RMS}} = 0.5$
deg = 2
0.0
0.2
0.4
0.6
0.8
1.0

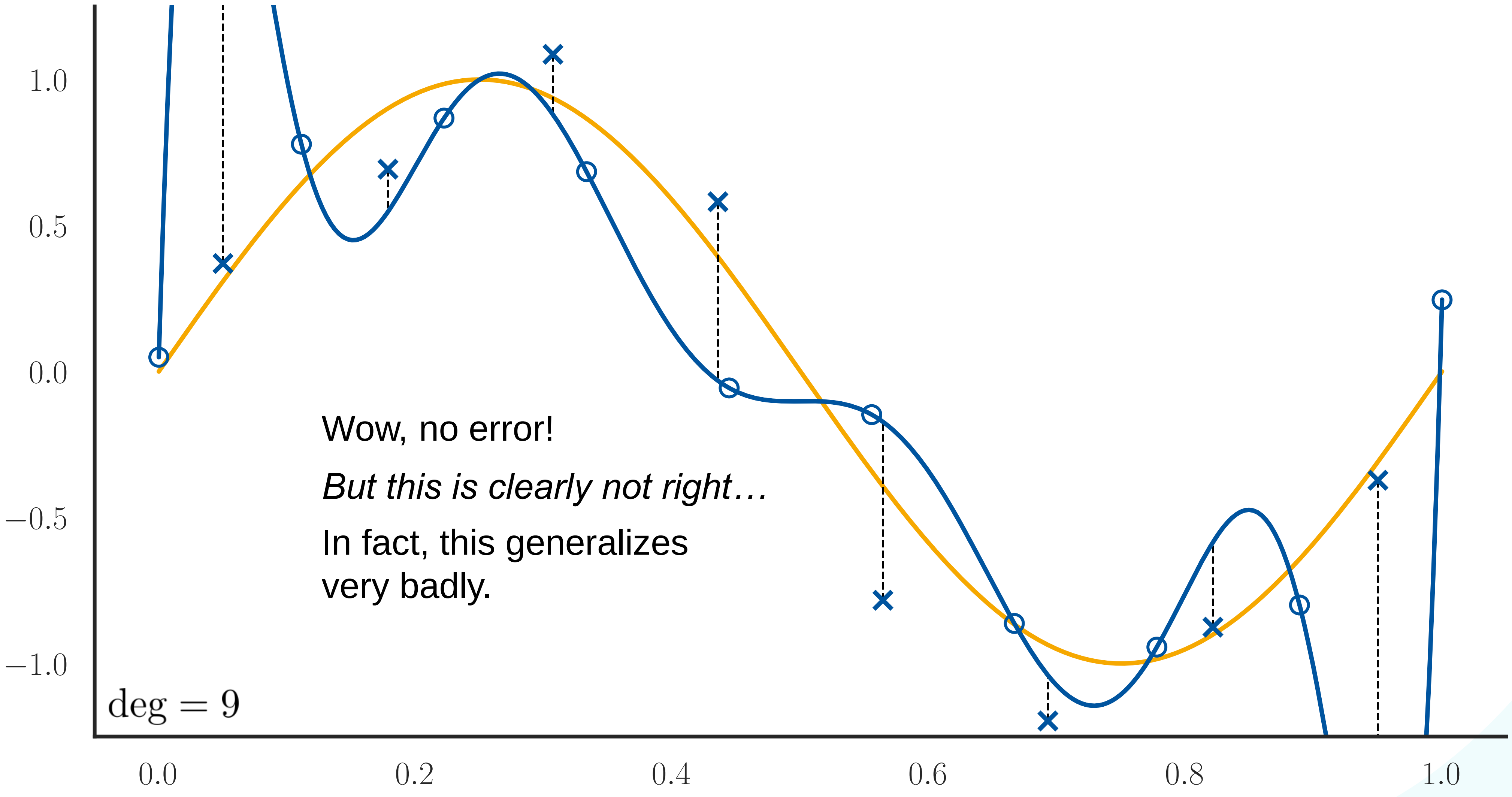

1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
$E_{\mathrm{RMS}} = 0.11$
deg = 3


1.0
0.5
0.0
−0.5
−1.0
$E_{\mathrm{RMS}} = 0.1$
deg = 8
0.0
0.2
0.4
0.6
0.8
1.0



Wow, no error!

But this is clearly not right…



Wow, no error!

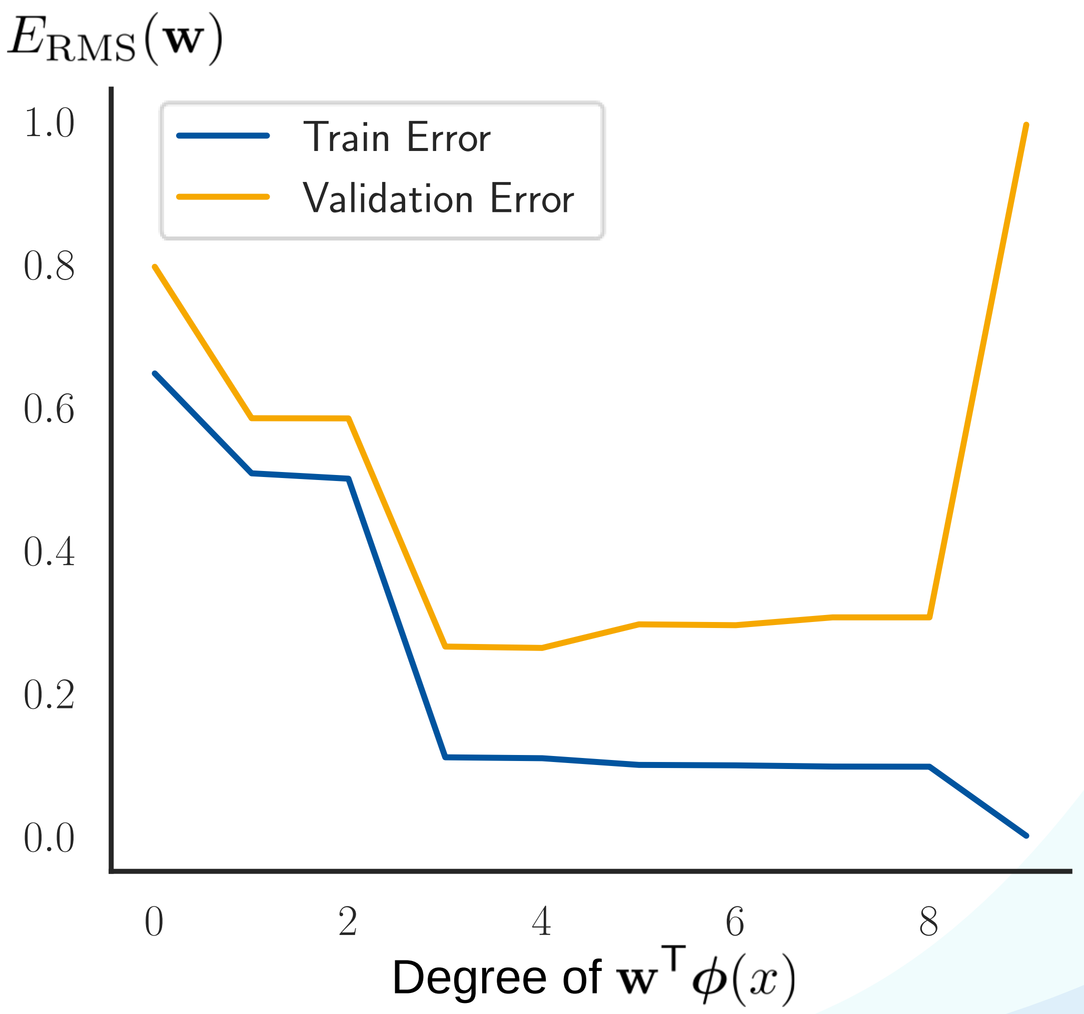
But this is clearly not right…

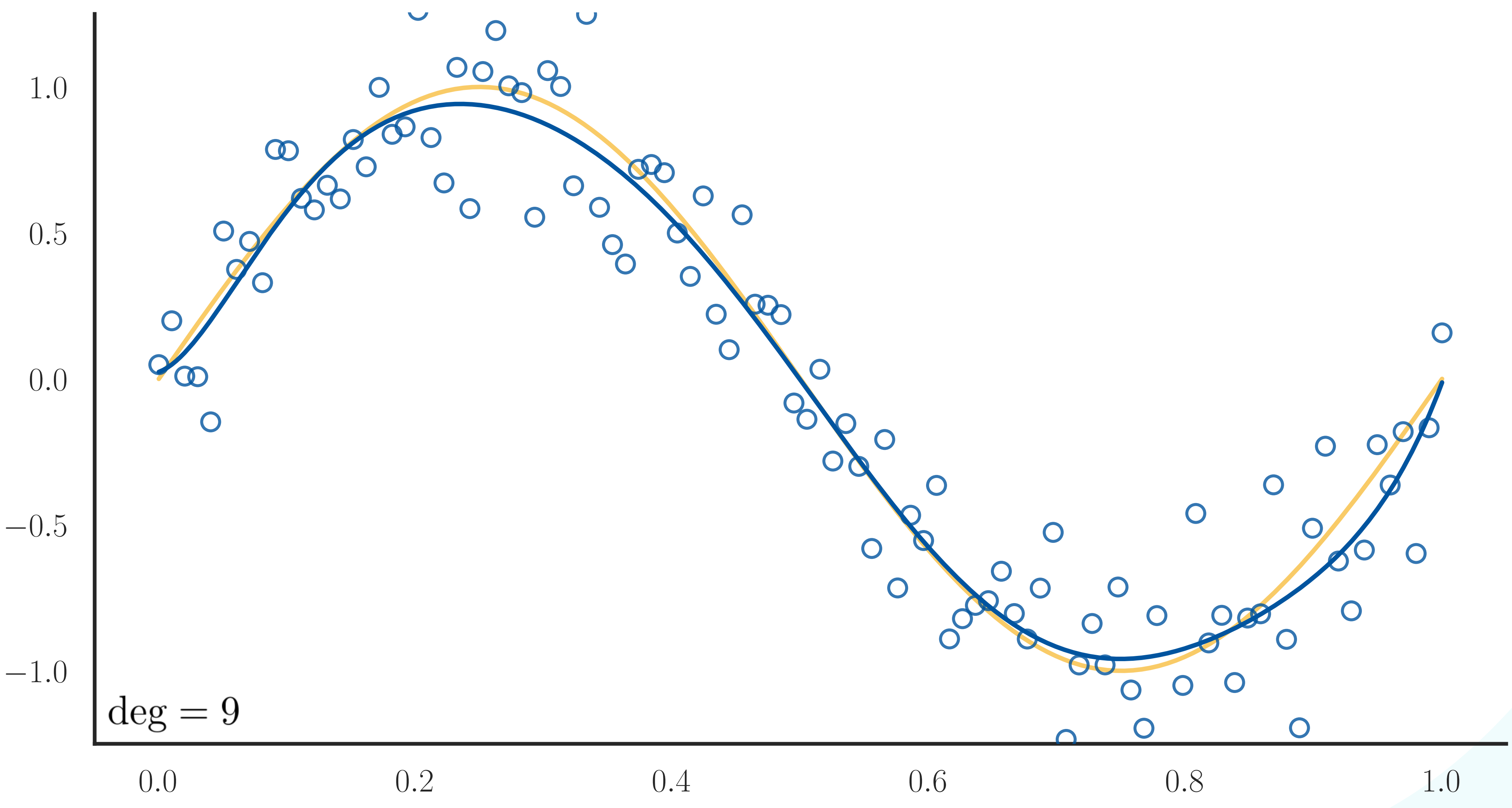
In fact, this generalizes very badly.

Overfitting

- We fit the dataset perfectly, but the resulting function is clearly not what we want.
- This phenomenon is called overfitting.
- Remember: we assume $t_n = h(\mathbf{x}_n) + \epsilon$.
- Our model is “too” powerful and models the noise instead of the underlying function!

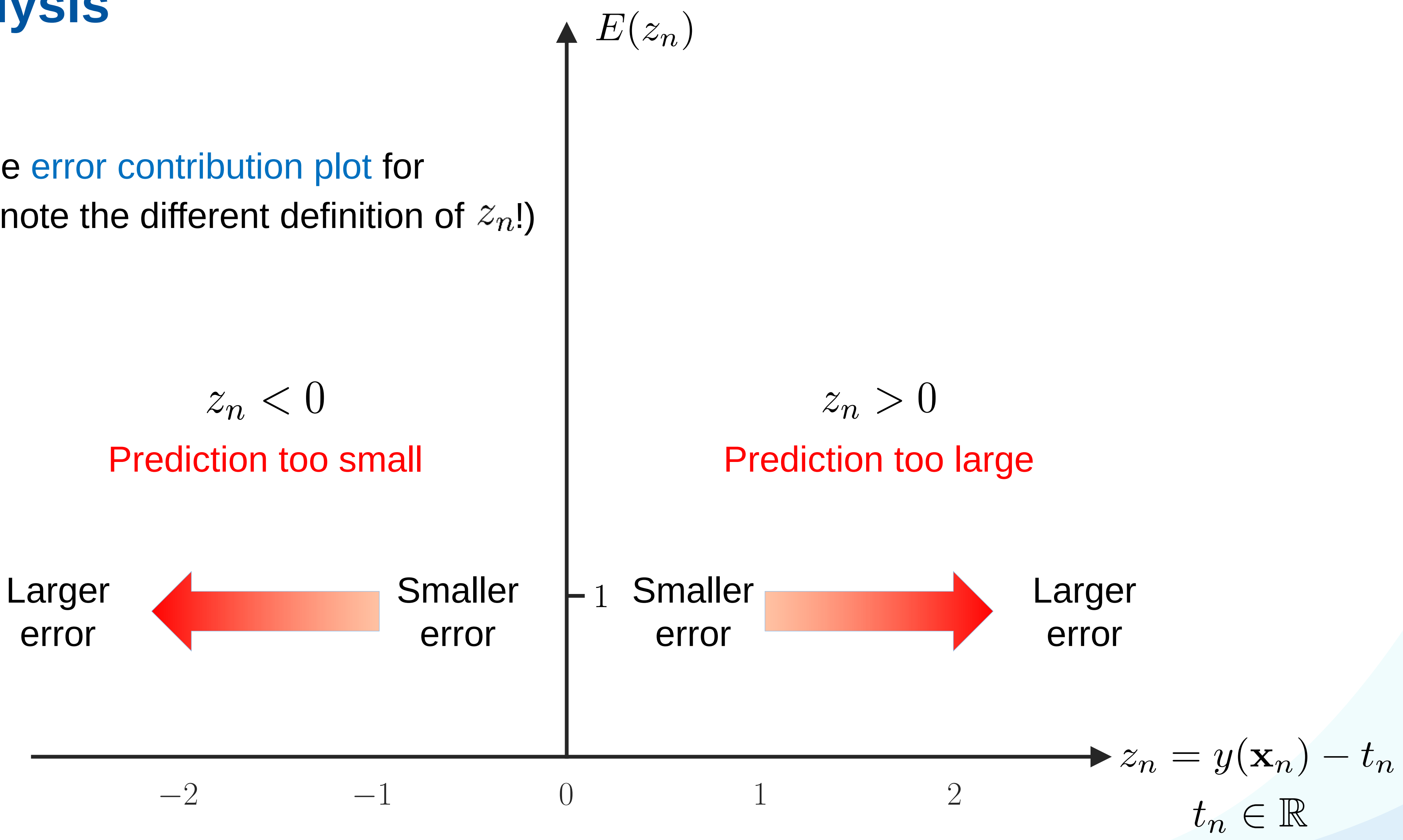
- *What can we do to avoid overfitting?*
  - One solution: More data!



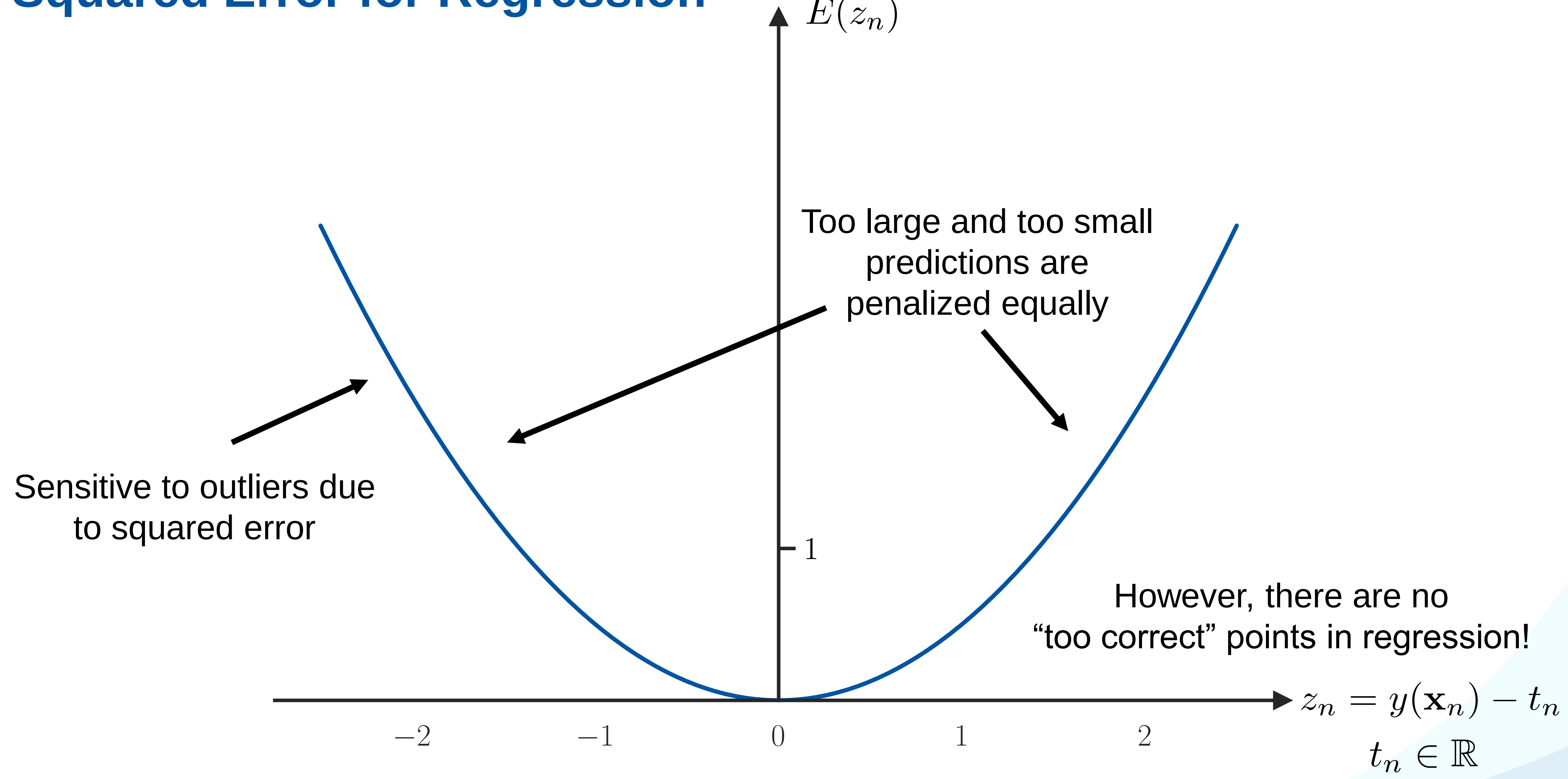

1.0
0.5
0.0
−0.5
−1.0
deg = 9
0.0
0.2
0.4
0.6
0.8
1.0

Error Analysis

- Variant of the error contribution plot for regression (note the different definition of z_n!)



Squared Error for Regression



Discussion: Least-Squares Regression

Advantages

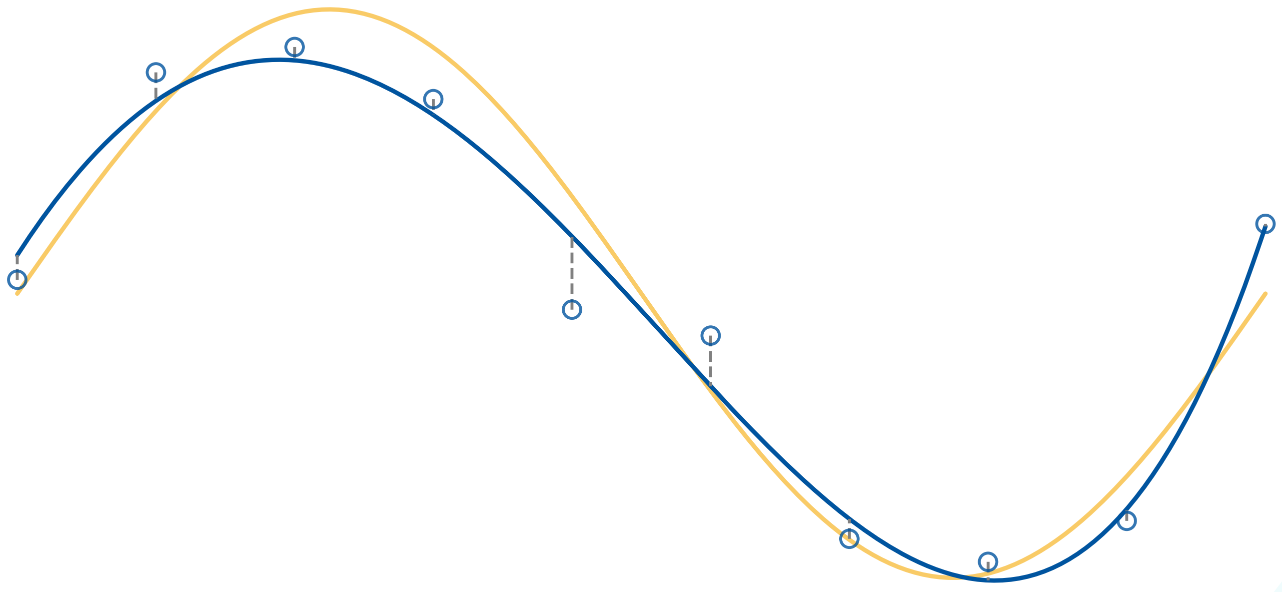
- Squared error leads to closed-form solution of the regression problem.
- We can use basis functions to fit non-linear functions while staying within the framework of linear regression.
- Polynomial basis functions with different degrees of the polynomial result in regression functions with different capacities to approximate the target function.
- We can compare their results using the RMS error.

Limitations

- The squared error for regression is not robust to outliers, but it does not exhibit the systematic problems of least-squares classification.
- Overfitting when the degree of the polynomial becomes too large.
- Overfitting is a function of the available amount of data (and is more likely to occur with small training sets)

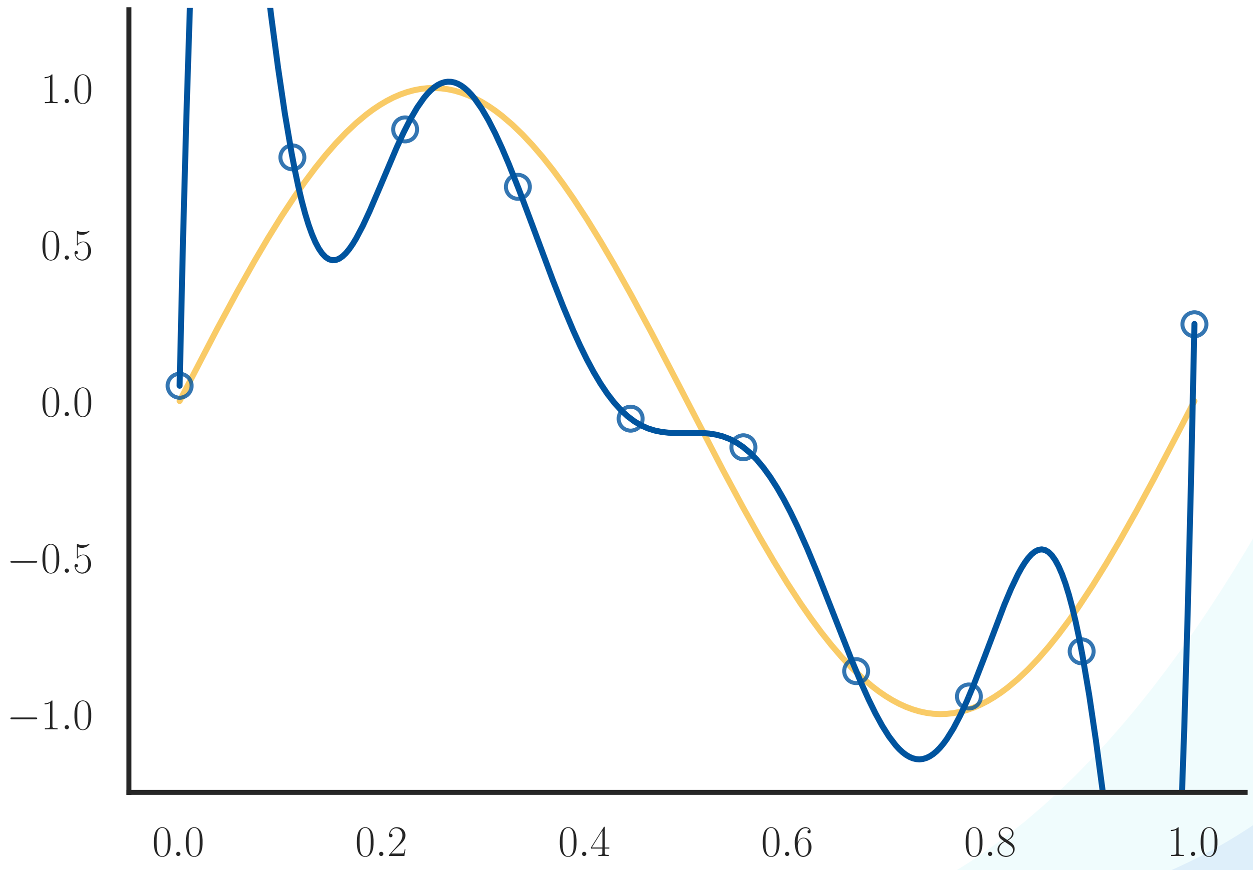
Linear Regression

1. Linear Regression
2. Least-Squares Regression
3. **Regularization**
4. Ridge Regression
5. The Bias-Variance Tradeoff



Regularization

- With enough parameters, our model will overfit to the training set.



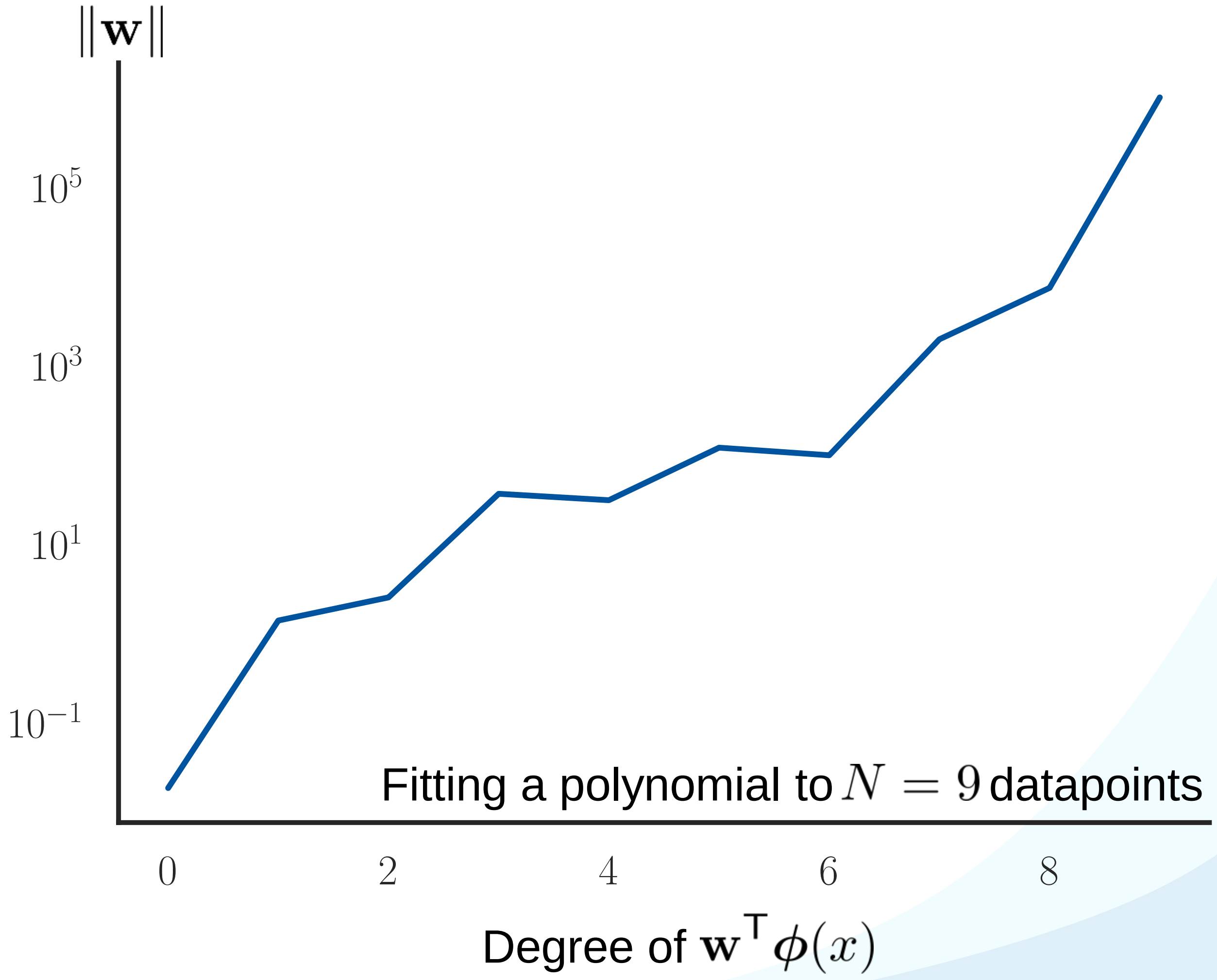
Regularization

- With enough parameters, our model will overfit to the training set.
- This leads to very large coefficient values w_i and thus to a large $\|\mathbf{w}\|$.
- Solution: penalize large parameters.

$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

- $L(\mathbf{w})$ is called the loss term. Here, we can use the familiar squared loss.
- $\Omega(\mathbf{w})$ is called the regularizer. Here, we use a squared regularizer.



Note: Excluding the Bias

- The bias w_0 is usually not regularized, since it does not change the functions' complexity.
- Therefore, we do not include it in $\Omega(\mathbf{w})$ here.
- We can fit the model without a bias by estimating $\mathbf{w}$ on centered data:

$$t_n^c = t_n - \bar{t} \qquad \mathbf{x}_n^c = \mathbf{x}_n - \bar{\mathbf{x}}$$

$$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n \qquad \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$$

- And computing w_0 afterwards:

$$w_0 = \bar{t} - \mathbf{w}^\mathsf{T}\bar{\mathbf{x}}$$

$$y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) + w_0$$

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}((\mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n) + w_0) - t_n)^2$$

$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\sum_{j=1}^{M} w_j^2$$

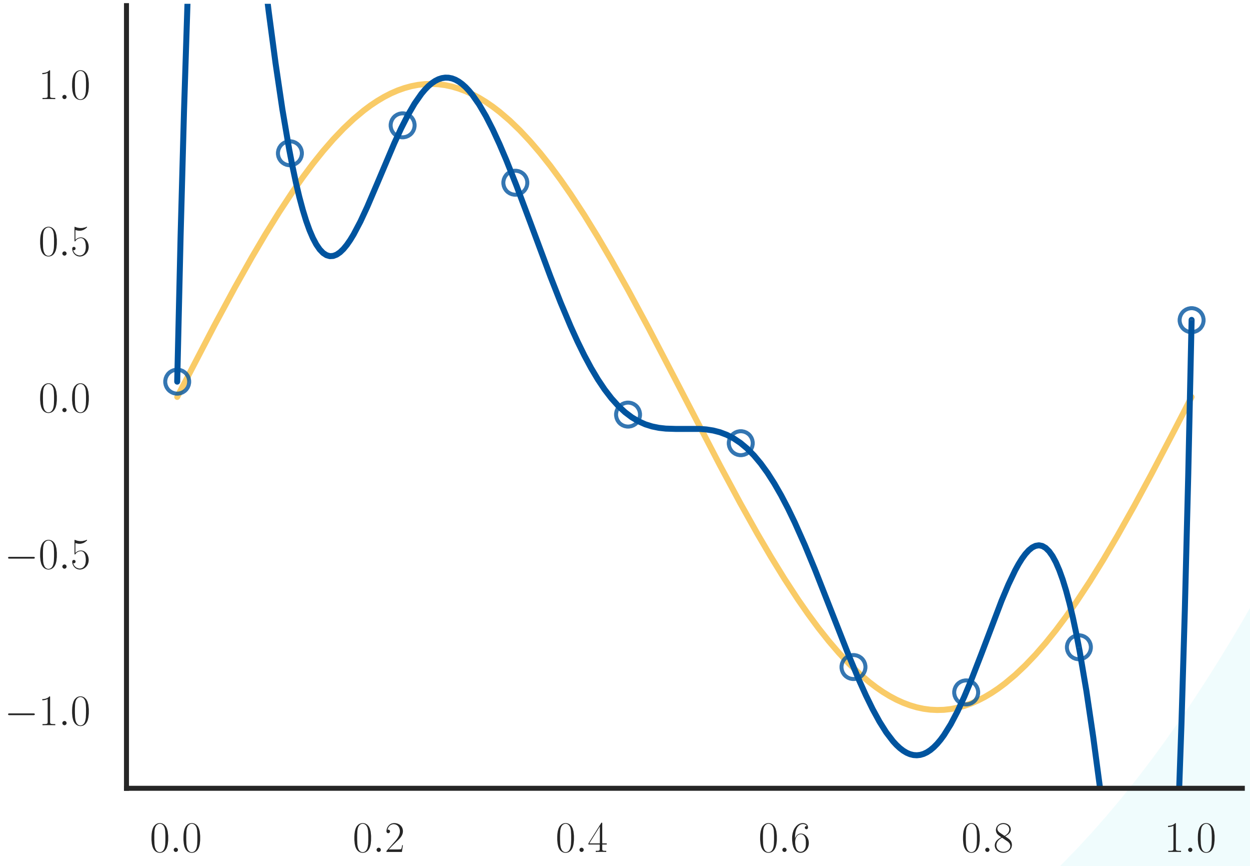
Example: Regularizing a Polynomial

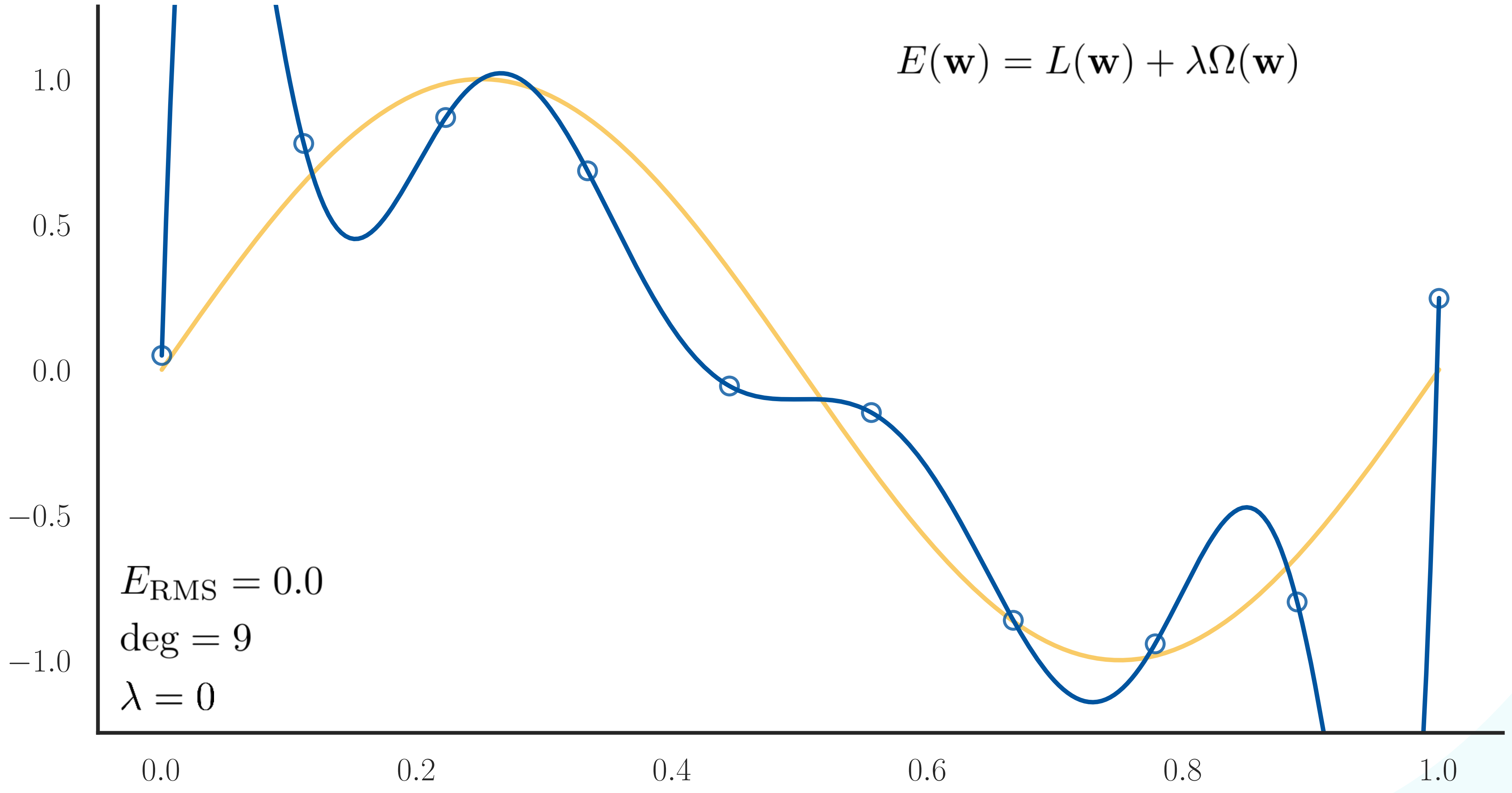
- Again, use polynomial basis functions:

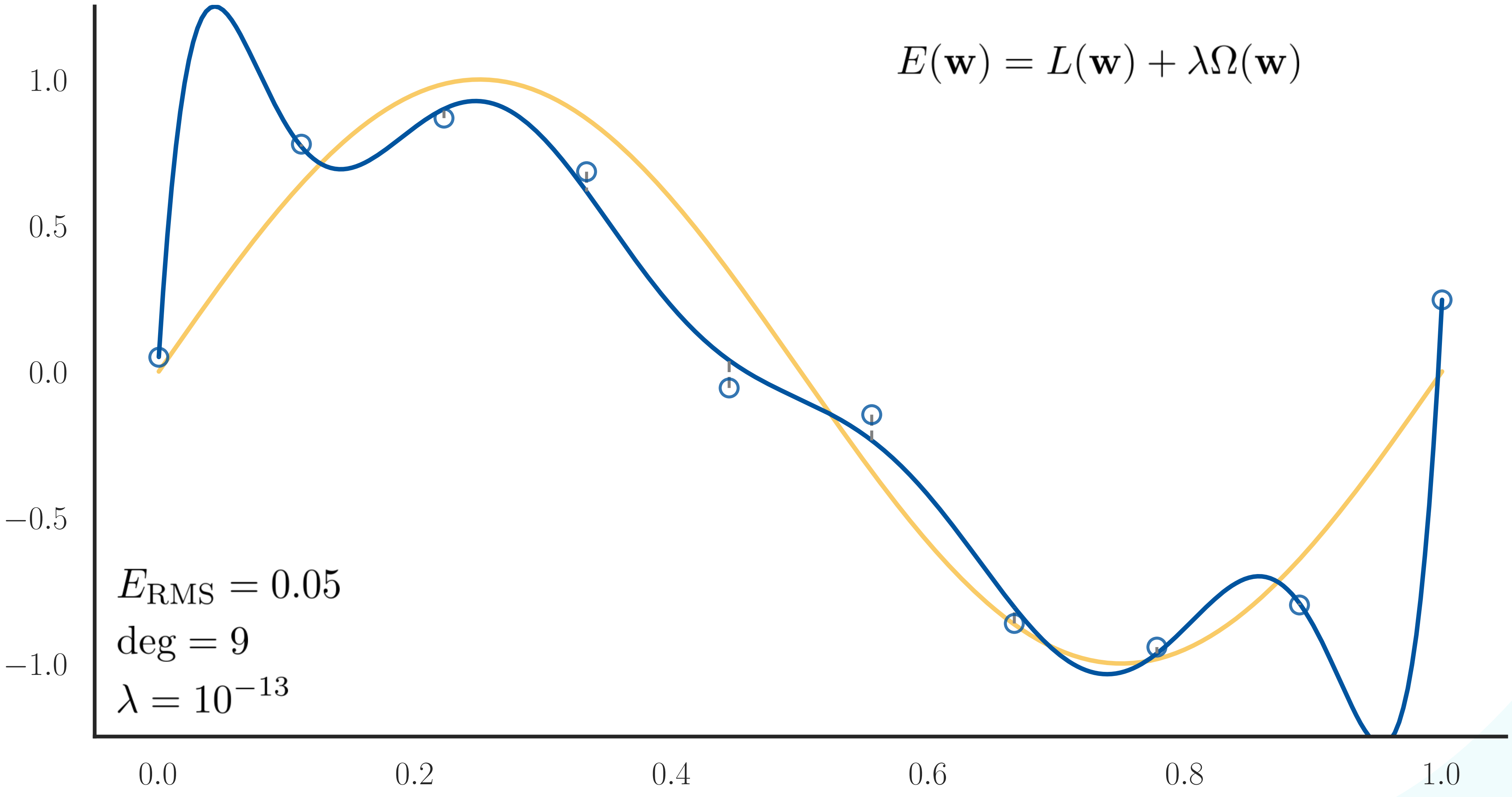
$$\phi_j(x) = (x^j)$$

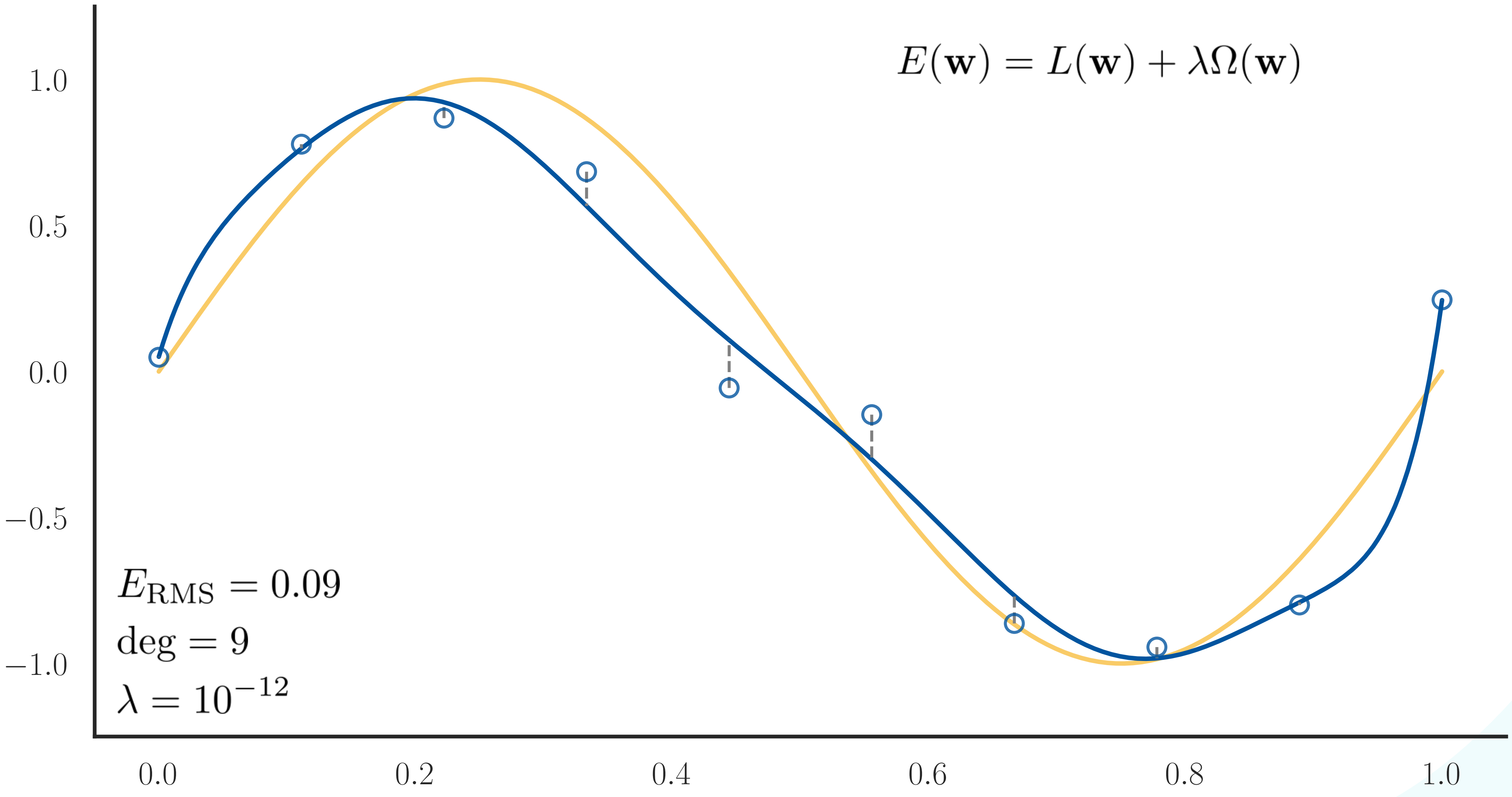
- Start off with an overfitting model.

- *How much should we regularize?*

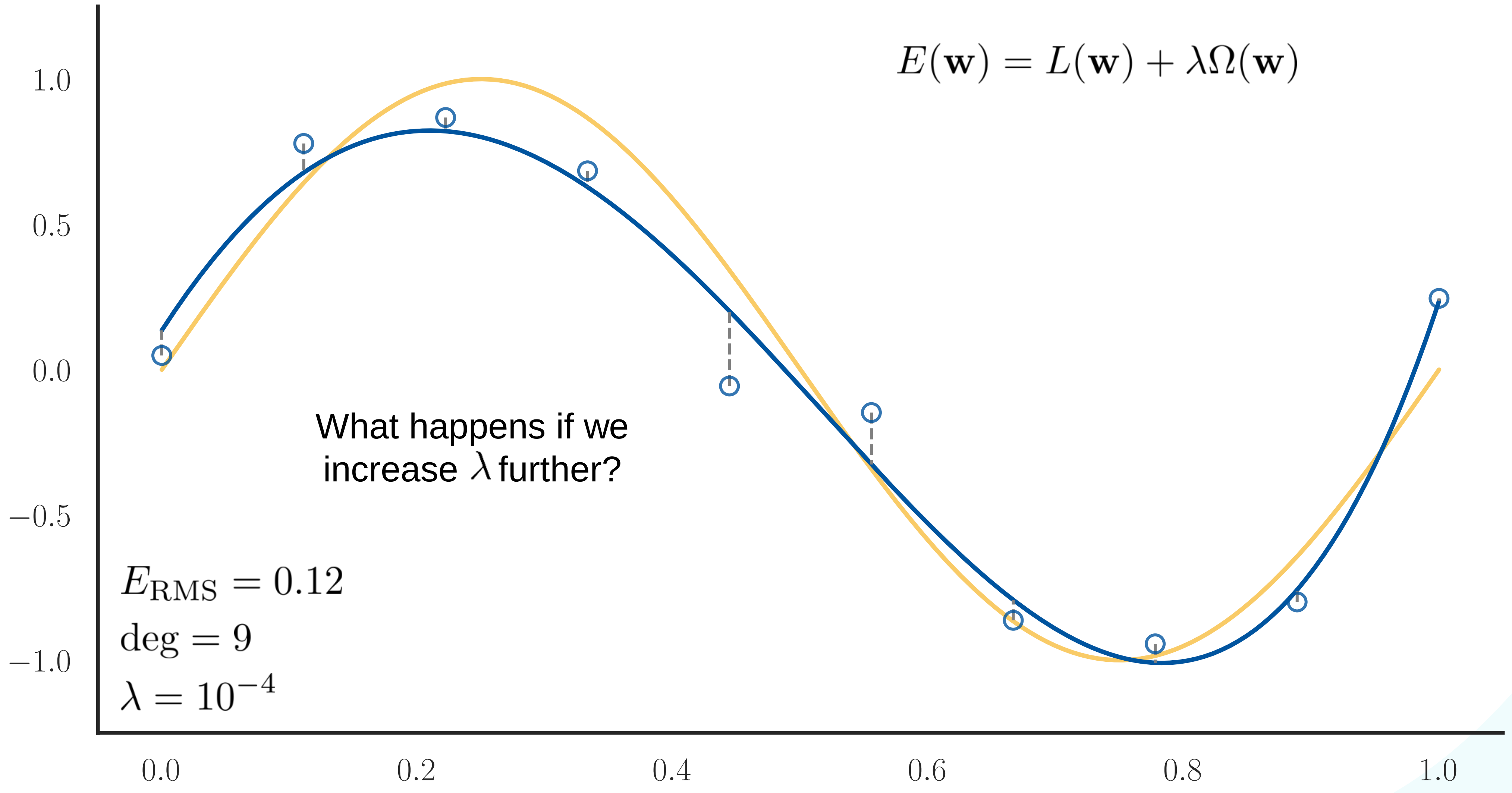


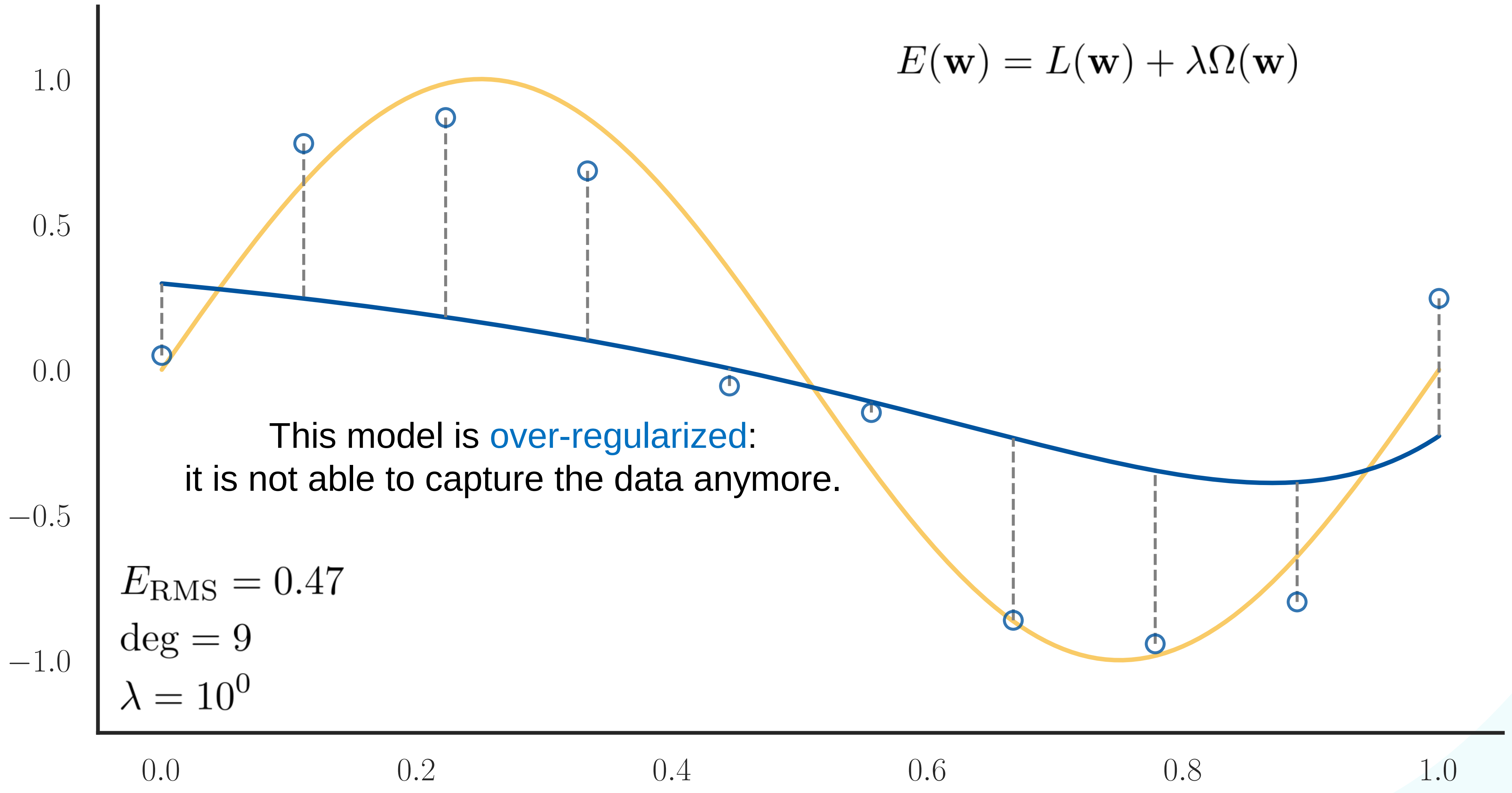

$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$
$E_{\mathrm{RMS}} = 0.0$
$\mathrm{deg} = 9$
$\lambda = 0$
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$
$E_{\text{RMS}} = 0.05$
$\text{deg} = 9$
$\lambda = 10^{-13}$
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0



$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$
1.0
0.5
0.0
−0.5
−1.0
$E_{\text{RMS}} = 0.09$
$\text{deg} = 9$
$\lambda = 10^{-12}$
0.0
0.2
0.4
0.6
0.8
1.0

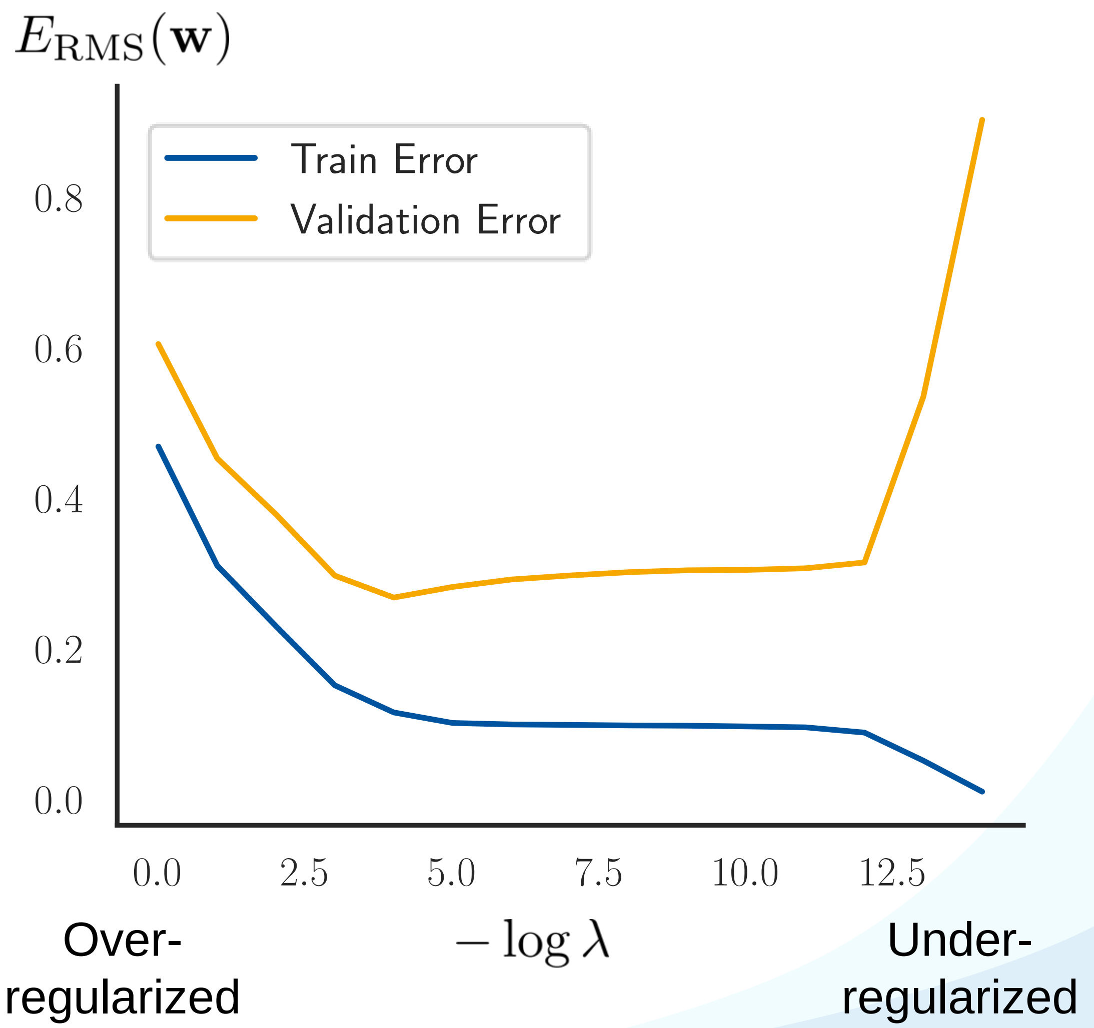

$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
What happens if we increase λ further?
$E_{\text{RMS}} = 0.12$
$\text{deg} = 9$
$\lambda = 10^{-4}$



$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$
1.0
0.5
0.0
−0.5
−1.0
This model is over-regularized:
it is not able to capture the data anymore.
$E_{\text{RMS}} = 0.47$
$\deg = 9$
$\lambda = 10^0$
0.0
0.2
0.4
0.6
0.8
1.0

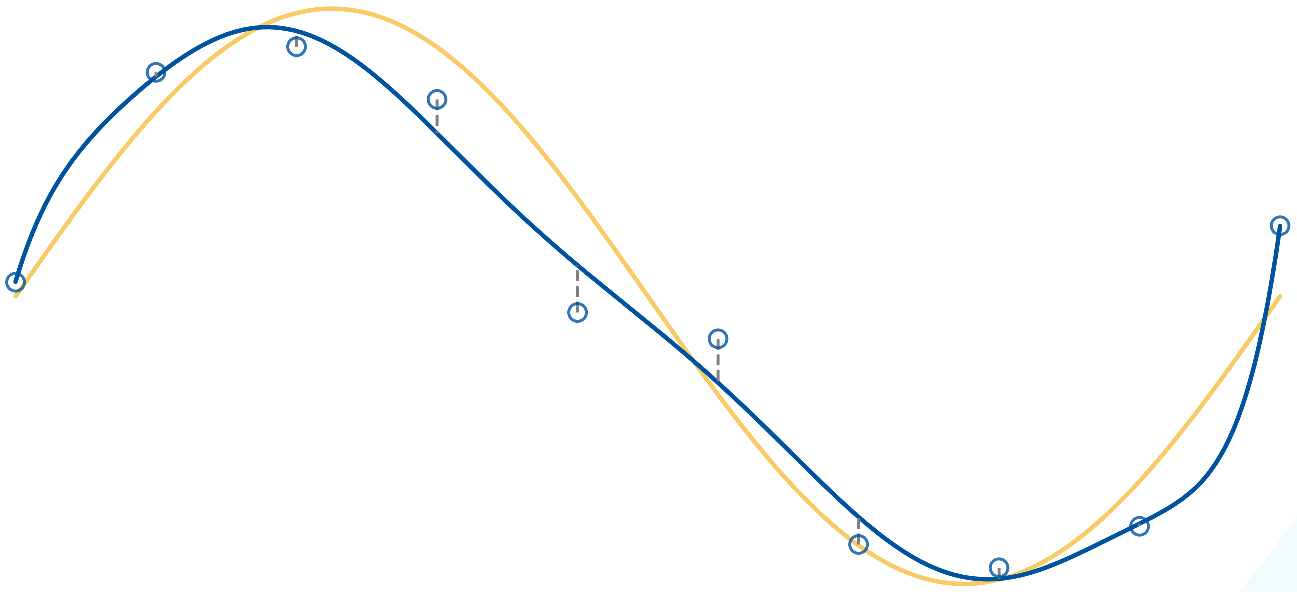
Choosing the right Regularization

- Regularization allows us to apply complex model on small datasets.
- However, we shifted the problem from selecting a suitable model to selecting a suitable regularization.
- The regularization factor λ becomes a hyperparameter.



Linear Regression

1. Motivation
2. Least-Squares Regression
3. Regularization
4. **Ridge Regression**
5. The Bias-Variance Tradeoff



Ridge Regression

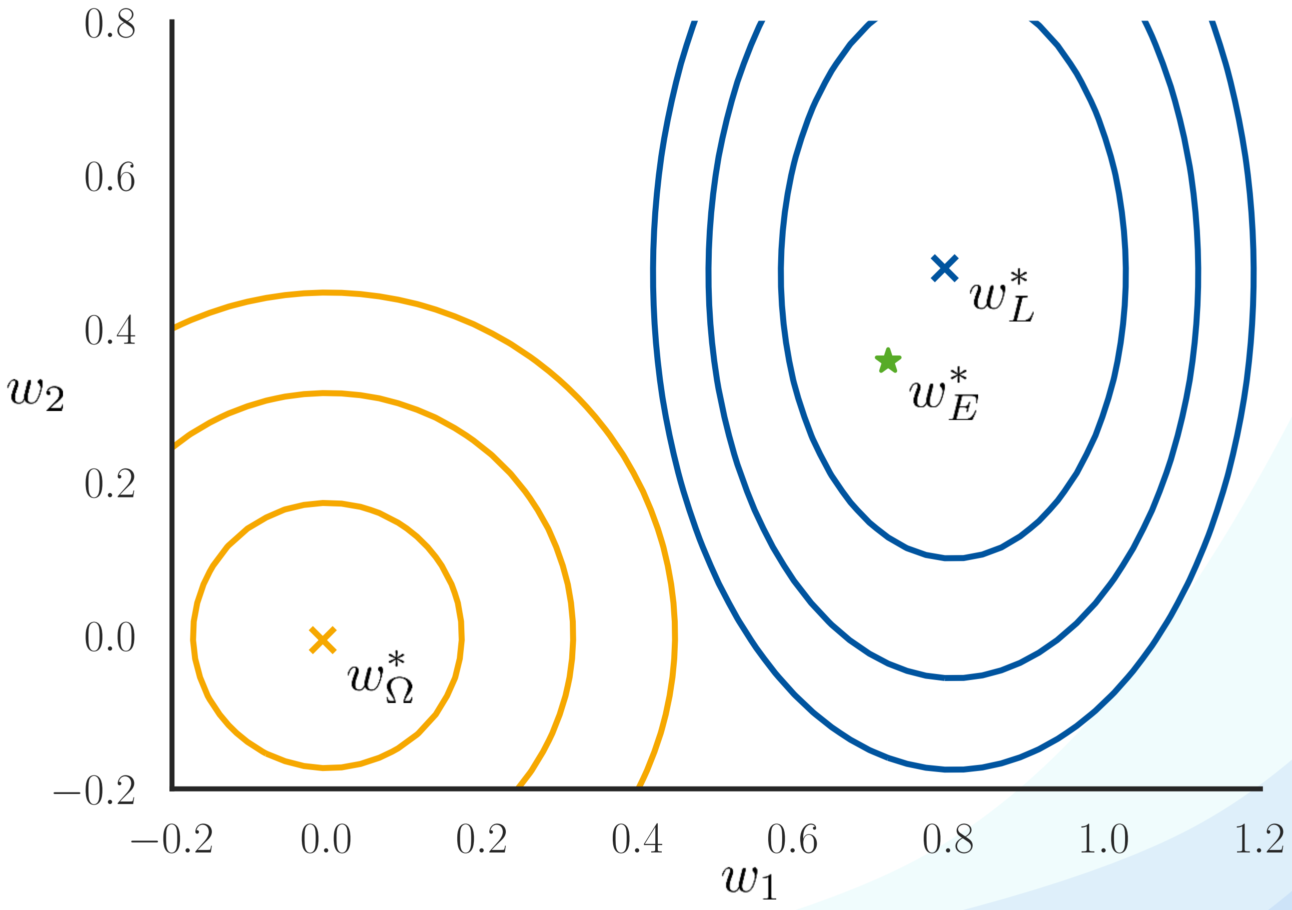
- We want to jointly minimize the squared error and the regularization term:

$$E(\mathbf{w}) = L(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}_n;\mathbf{w}) - t_n)^2$$

$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

- This model is called ridge regression.



Derivation

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}_n;\mathbf{w}) - t_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^\top\phi(\mathbf{x}_n) - \mathbf{t}_n)\phi(\mathbf{x}_n) + \lambda\mathbf{w} \stackrel{!}{=} 0$$

$$\mathbf{\Phi}^\top(\mathbf{\Phi}\mathbf{w} - \mathbf{t}) + \lambda\mathbf{w} = 0$$

$$(\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})\mathbf{w} = \mathbf{\Phi}^\top\mathbf{t}$$

$$\mathbf{w} = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{t}$$

Effect of regularization: keeps the inverse well-conditioned.

References and Further Reading

- More information about Linear Discriminants is available in Chapter 4.1 of Bishop's book. For more information about Linear Regression, read Chapter 3.1.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006