Visual Computing Institute
RWTH AACHEN UNIVERSITY

Elements of Machine Learning & Data Science

Winter semester 2023/24

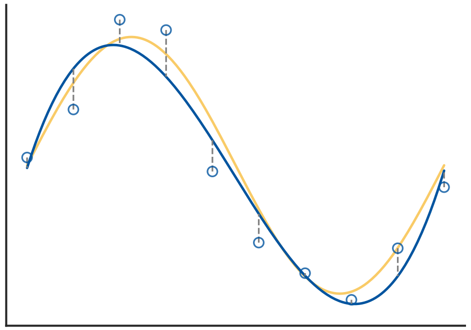
Lecture 16 – Logistic Regression

08.12.2023

Prof. Bastian Leibe

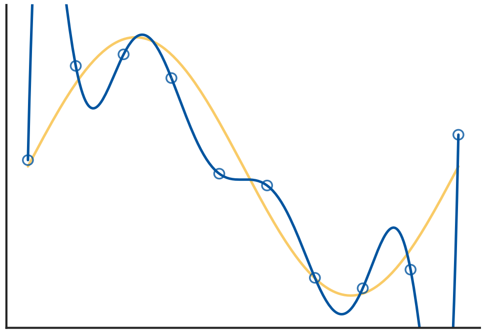
Machine Learning Topics

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

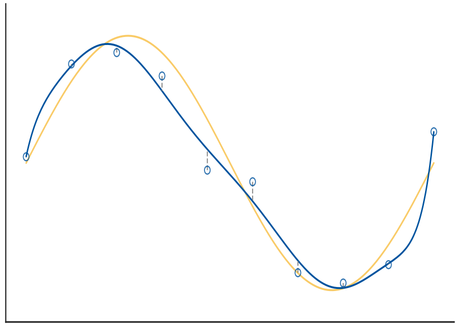


$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + w_0$$

Linear Regression Functions

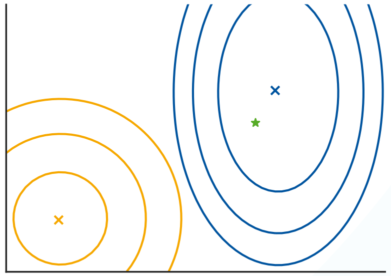


Overfitting



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

Regularization



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

Ridge Regression

Recap: 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).

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

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

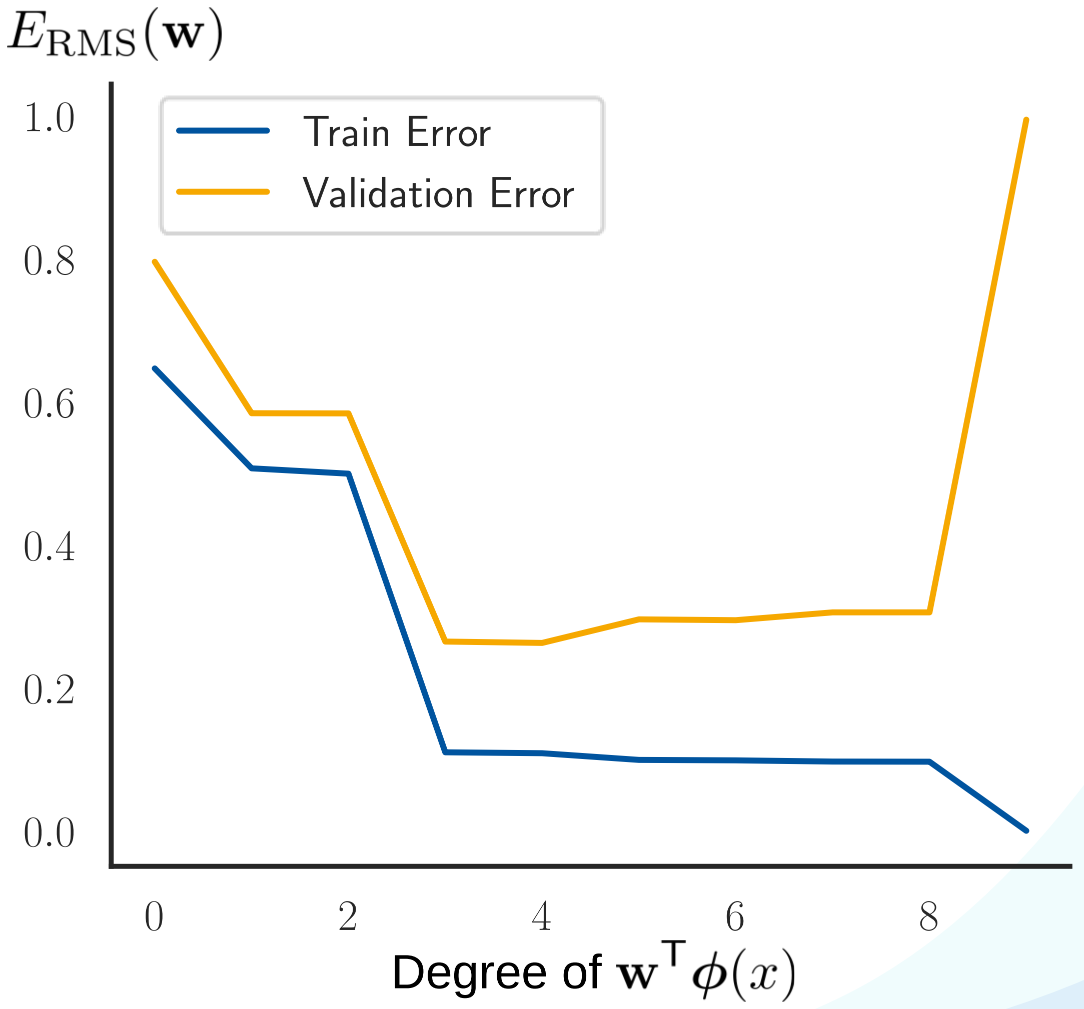
$$= \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{w} - \mathbf{t}) \stackrel{!}{=} 0$$

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

Recap: 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?*



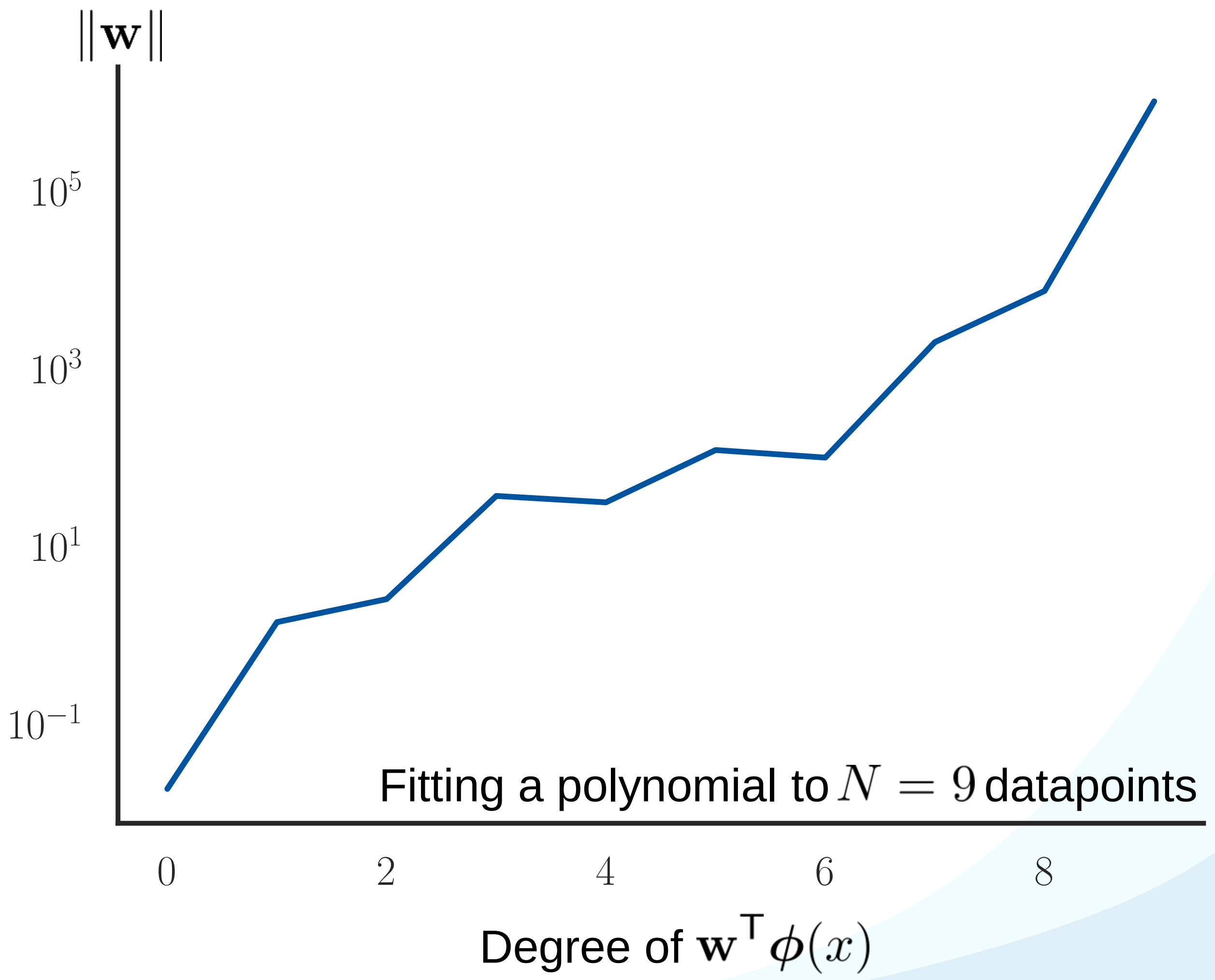
Recap: Regularization

- 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.

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

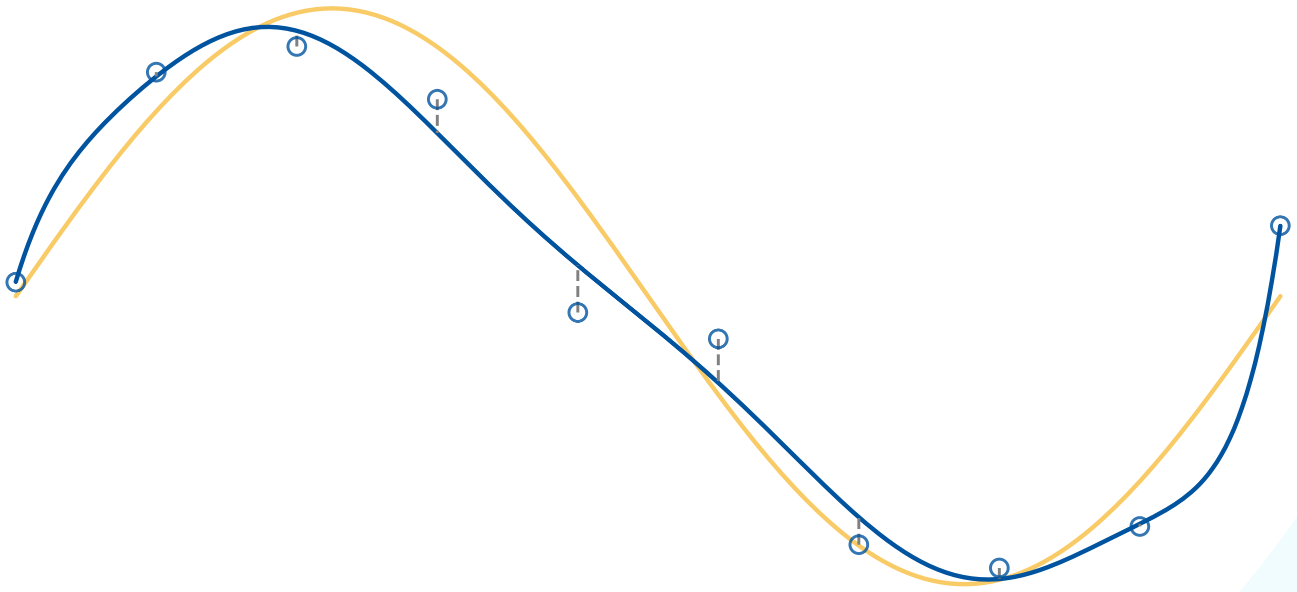
$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

- $L(\mathbf{w})$ 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.



Linear Regression

1. Motivation
2. Least-Squares Regression
3. Regularization
4. **Ridge Regression**



Ridge Regression

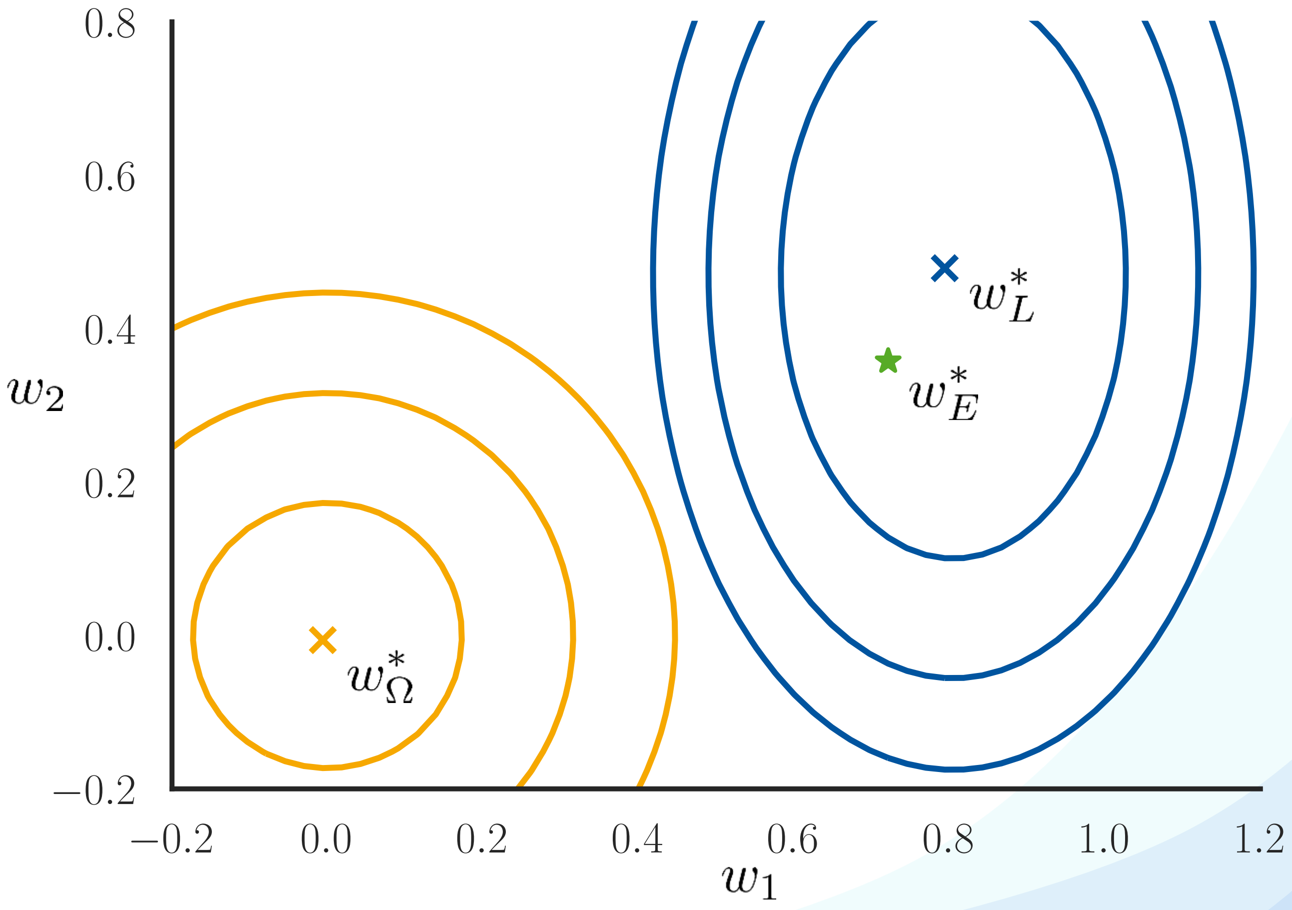
- We want to jointly minimize the squared error and the regularization term:

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

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

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

- This model is called ridge regression.



Derivation

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

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

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

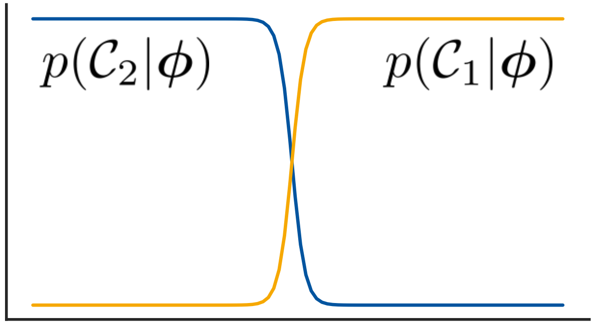
$$(\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})\mathbf{w} = \mathbf{\Phi}^\top\mathbf{t}$$

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

Effect of regularization: keeps the inverse well-conditioned.

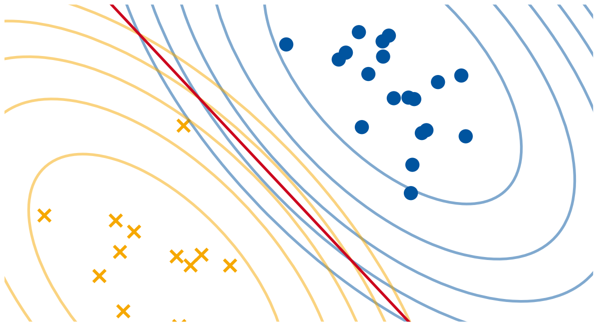
Machine Learning Topics

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

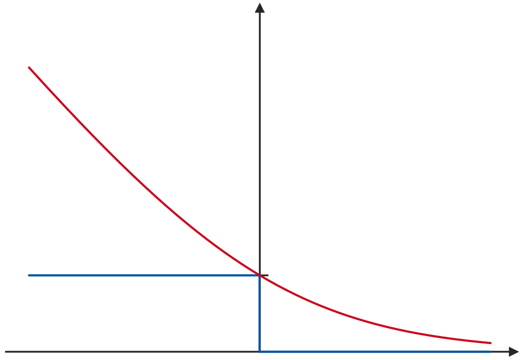


$$y(\boldsymbol{\phi}) = p(\mathcal{C}_1|\boldsymbol{\phi}) = \sigma(\mathbf{w}^{\mathsf{T}}\boldsymbol{\phi} + w_0)$$

Logistic Regression Formulation

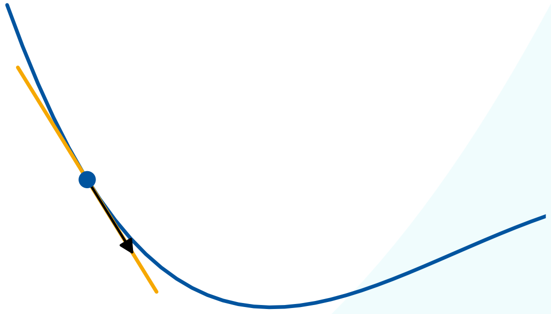


Parameter Efficiency



$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w})$$

Cross-Entropy Error

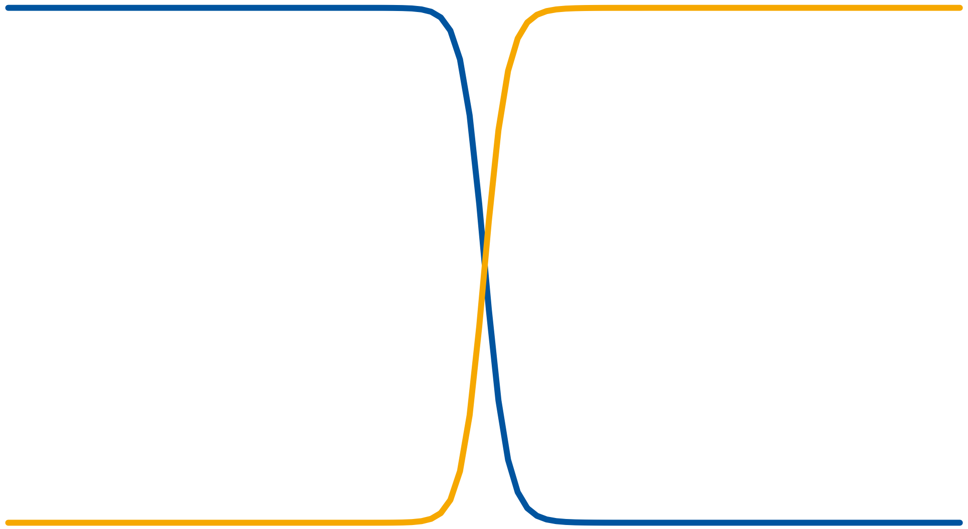


$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

Iterative Optimization

Logistic Regression

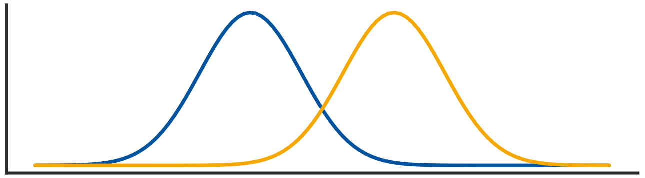
1. **Logistic Regression Formulation**
2. Motivation and Background
3. Iterative Optimization
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis



Motivation

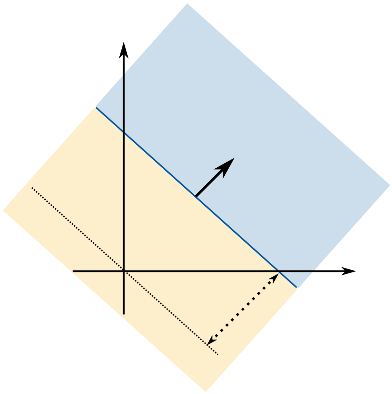
- We have seen how to build probabilistic classifiers using Bayes' Theorem:

$$y_k(\mathbf{x}) = p(\mathcal{C}_k|\mathbf{x}) \propto p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$



- We have directly modeled the decision boundary with linear discriminants:

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

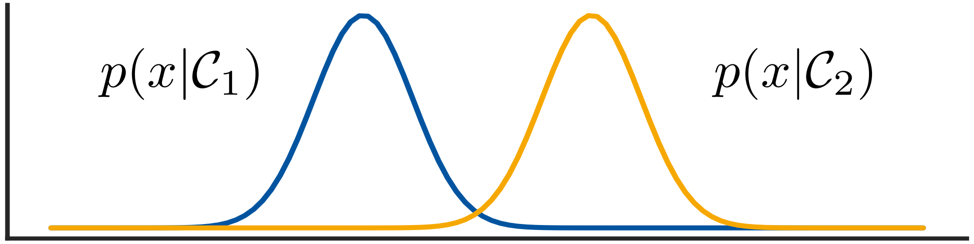


- In the following, we will combine those two ideas
  - We will model the posterior $p(\mathcal{C}_k|\mathbf{x})$
  - But we will do that using a linear discriminant function $y(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0$
- The resulting model will be called logistic regression.

Reminder: Probabilistic Classification

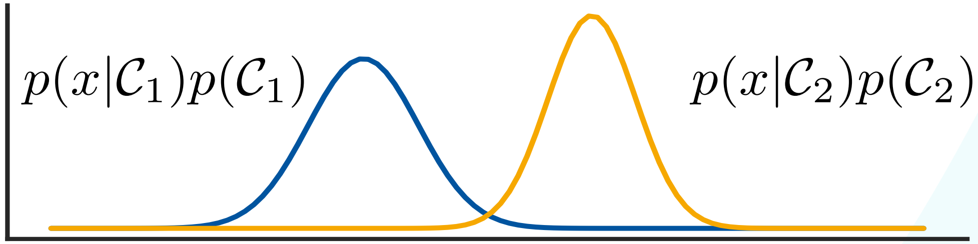
- Remember what we did in probabilistic classification
  - We modeled the likelihood of each class

$$p(\mathbf{x}|\mathcal{C}_k)$$



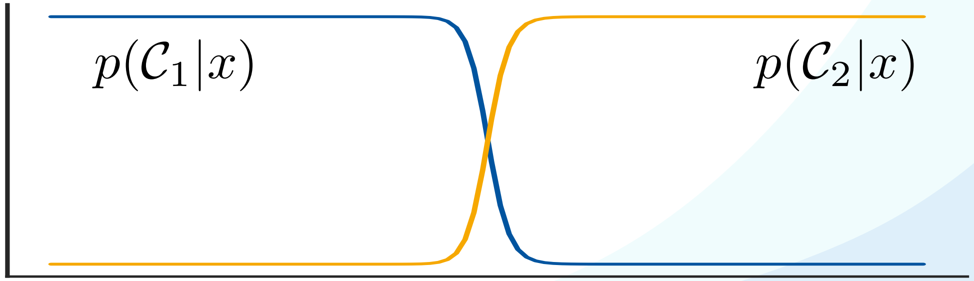
  - We scaled the likelihoods with the priors

$$p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$



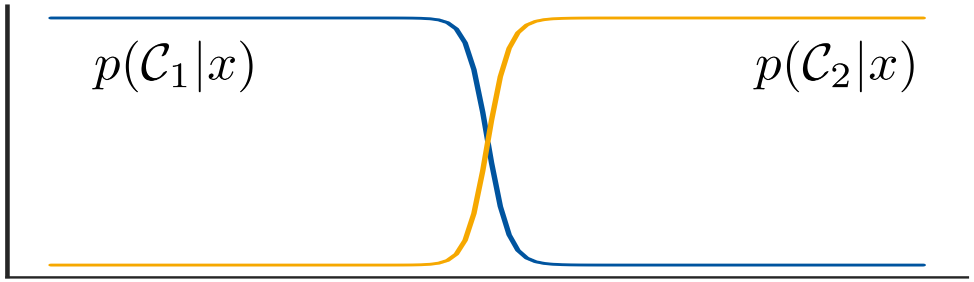
  - We normalized to compute the posterior

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$



- Let's now start with the posterior and rewrite it

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$
$$= \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}}$$
$$= \frac{1}{1 + \exp(-a)} \qquad =: \sigma(a)$$

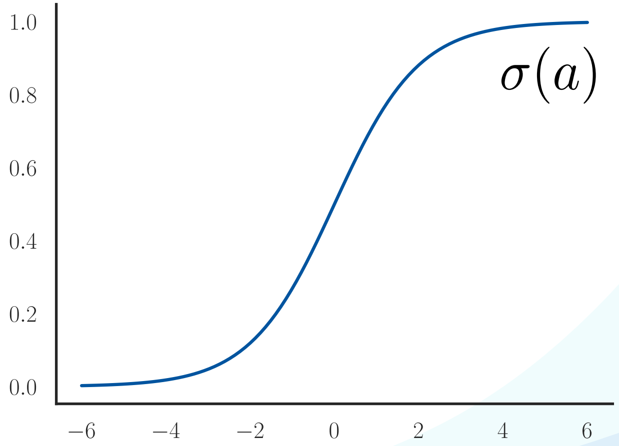


- This is the equation for the logistic sigmoid function $\sigma(a)$

  $\Rightarrow$ If we set

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

  the logistic sigmoid expresses a posterior probability!



Properties of the Logistic Sigmoid

- Definition:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- Inverse (also known as logit function):

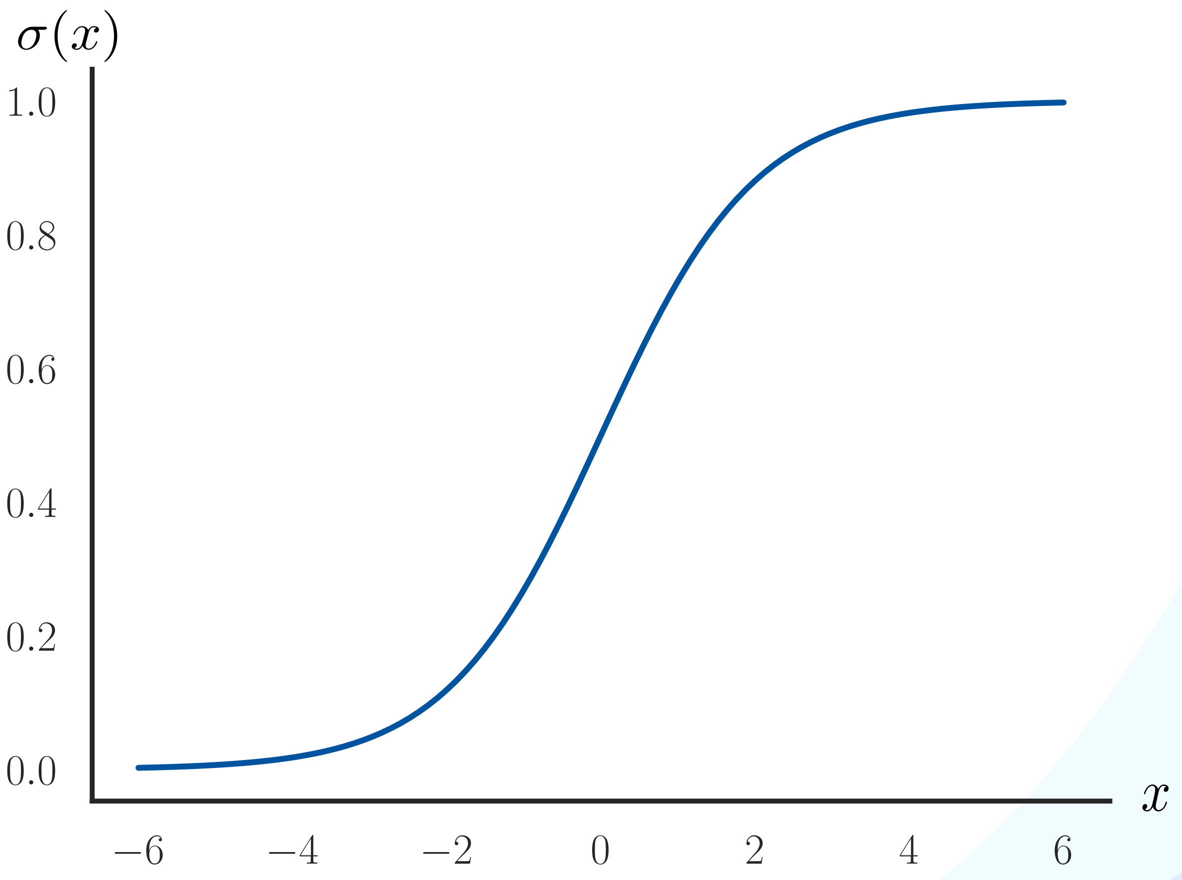
$$a = \ln\left(\frac{\sigma(a)}{1 - \sigma(a)}\right)$$

- Symmetry:

$$\sigma(-a) = 1 - \sigma(a)$$

- Derivative:

$$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$$



Logistic Regression

- We now define the logistic regression model
  - For the start, let us assume two classes $\mathcal{C}_1, \ \mathcal{C}_2$.
  - We model the class posteriors $p(\mathcal{C}_k|\mathbf{x})$ as

$$p(\mathcal{C}_1|\mathbf{x}) = y(\mathbf{x}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x})$$

$$p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$$

Logistic sigmoid activation function:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

  - I.e., we define a linear discriminant model

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

    that is meant to represent the class posterior with the help of a logistic sigmoid activation function $\sigma(a)$.
  - Our target labels are now $t_n \in \{0, 1\}$.

Error Function

- Consider a data set $\mathcal{D} = \{(\phi_1, t_1), \ldots, (\phi_N, t_N)\}$
  - with data points $\mathbf{\Phi} = [\phi_1, \ldots, \phi_N]^\mathsf{T}, \ \phi_n = \phi(\mathbf{x}_n)$
  - And target labels $\mathbf{t} = [t_1, \ldots, t_N]^\mathsf{T}, \ t_n \in \{0, 1\}$

- Maximum likelihood approach
  - With $y_n = p(\mathcal{C}_1|\phi_n) = \sigma(\mathbf{w}^\mathsf{T}\phi_n)$
  - We model the probability of the target labels $\mathbf{t}$ given our model parameters $\mathbf{w}$ as

Trick: use t_n as an indicator variable

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \begin{cases} p(\mathcal{C}_1|\phi_n) & , t_n = 1 \\ p(\mathcal{C}_2|\phi_n) & , t_n = 0 \end{cases} = \prod_{n=1}^{N} \begin{cases} y_n & , t_n = 1 \\ (1 - y_n) & , t_n = 0 \end{cases} = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}$$

- Maximum likelihood approach

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}$$

  - Define the error function as the negative log-likelihood

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} (t_n \ln y_n + (1 - t_n) \ln(1 - y_n))$$

  - This function is known as the binary cross-entropy error.

Softmax Regression

- Multi-class extension of logistic regression
  - Generalization to K classes with target labels in 1-of-K notation $\mathbf{t}_n = [0, 1, \ldots, 0]^\mathsf{T}$
  - Again, we define a linear discriminant function that models the class posteriors

$$\mathbf{y}(\mathbf{x}; \mathbf{w}) = \begin{bmatrix} p(\mathcal{C}_1|\mathbf{x}; \mathbf{w}) \\ p(\mathcal{C}_2|\mathbf{x}; \mathbf{w}) \\ \vdots \\ p(\mathcal{C}_K|\mathbf{x}; \mathbf{w}) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(\mathbf{w}_j^\mathsf{T}\mathbf{x})} \begin{bmatrix} \exp(\mathbf{w}_1^\mathsf{T}\mathbf{x}) \\ \exp(\mathbf{w}_2^\mathsf{T}\mathbf{x}) \\ \vdots \\ \exp(\mathbf{w}_K^\mathsf{T}\mathbf{x}) \end{bmatrix}$$

  - This makes use of the softmax function as a multi-class extension of the logistic sigmoid

$$\text{softmax}(\mathbf{a}) = \frac{\exp(a_k)}{\sum_{j=1}^K \exp(a_j)}$$

- We can write the binary cross-entropy error as

$$E(\mathbf{w}) = -\sum_{n=1}^{N}(t_n \ln y_n + (1-t_n)\ln(1-y_n))$$
$$= -\sum_{n=1}^{N}\sum_{k=0}^{1}(\mathbb{I}(t_n = k)\ln p(\mathcal{C}_k|\mathbf{x}_n;\mathbf{w}))$$

indicator function

$$\mathbb{I}(\varphi) = \begin{cases} 1 & \text{if } \varphi \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

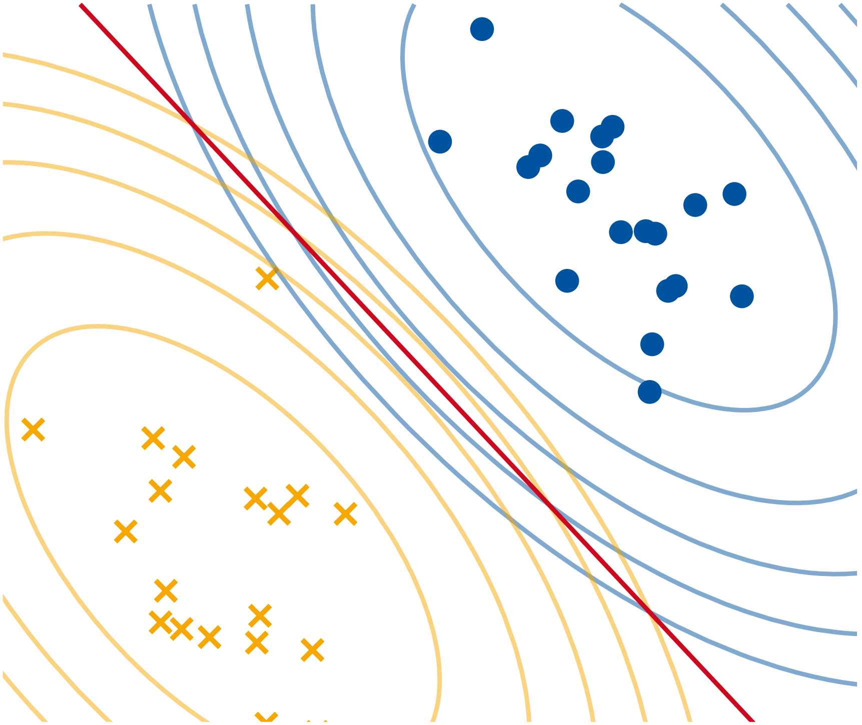
- Using one-hot labels $\mathbf{t}_n$, the generalization to K classes is:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\mathbb{I}(t_{kn} = 1)\ln\frac{\exp(\mathbf{w}_k^\top\mathbf{x})}{\sum_{j=1}^{K}\exp(\mathbf{w}_j^\top\mathbf{x})}\right)$$

- This function is known as the multi-class cross-entropy error or softmax cross-entropy error.

Logistic Regression

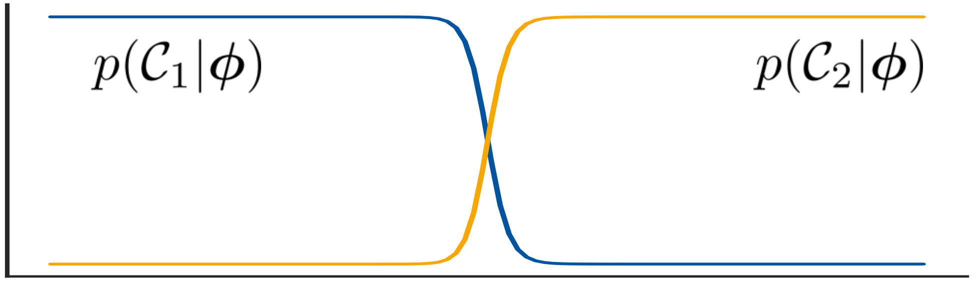
1. Logistic Regression Formulation
2. **Motivation and Background**
3. Iterative Optimization
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis



Motivation: Why Logistic Regression?

- Logistic Regression uses models of the form

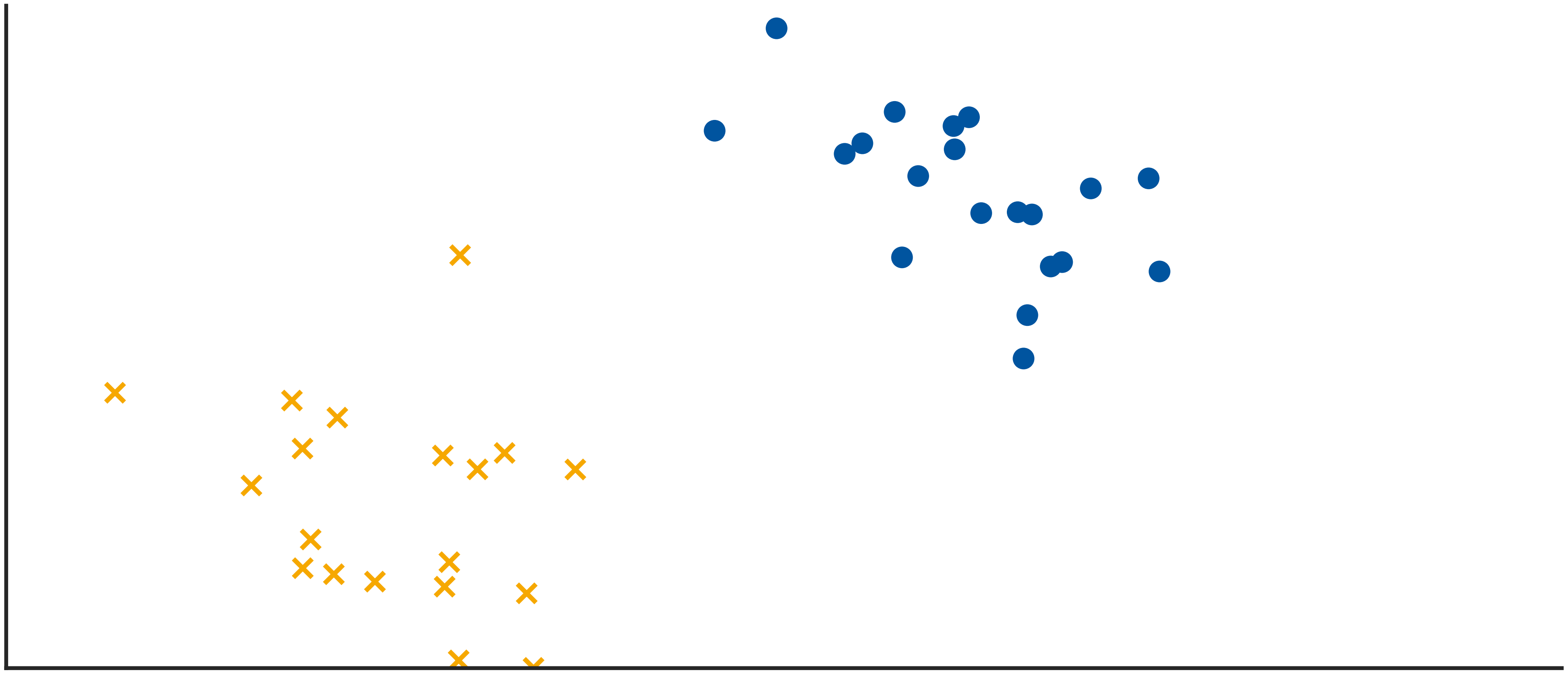
$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^\mathsf{T}\phi)$$
$$p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$



- Interpretation
  - We model the class posteriors $p(\mathcal{C}_k|\phi)$, as required to make Bayes optimal decisions.
  - We have seen previously that we can obtain $p(\mathcal{C}_k|\phi) = p(\phi|\mathcal{C}_k)p(\mathcal{C}_k)$.
  - However, here we model $p(\mathcal{C}_k|\phi)$ as a linear discriminant function $y(\phi) = \sigma(\mathbf{w}^\mathsf{T}\phi)$ instead.

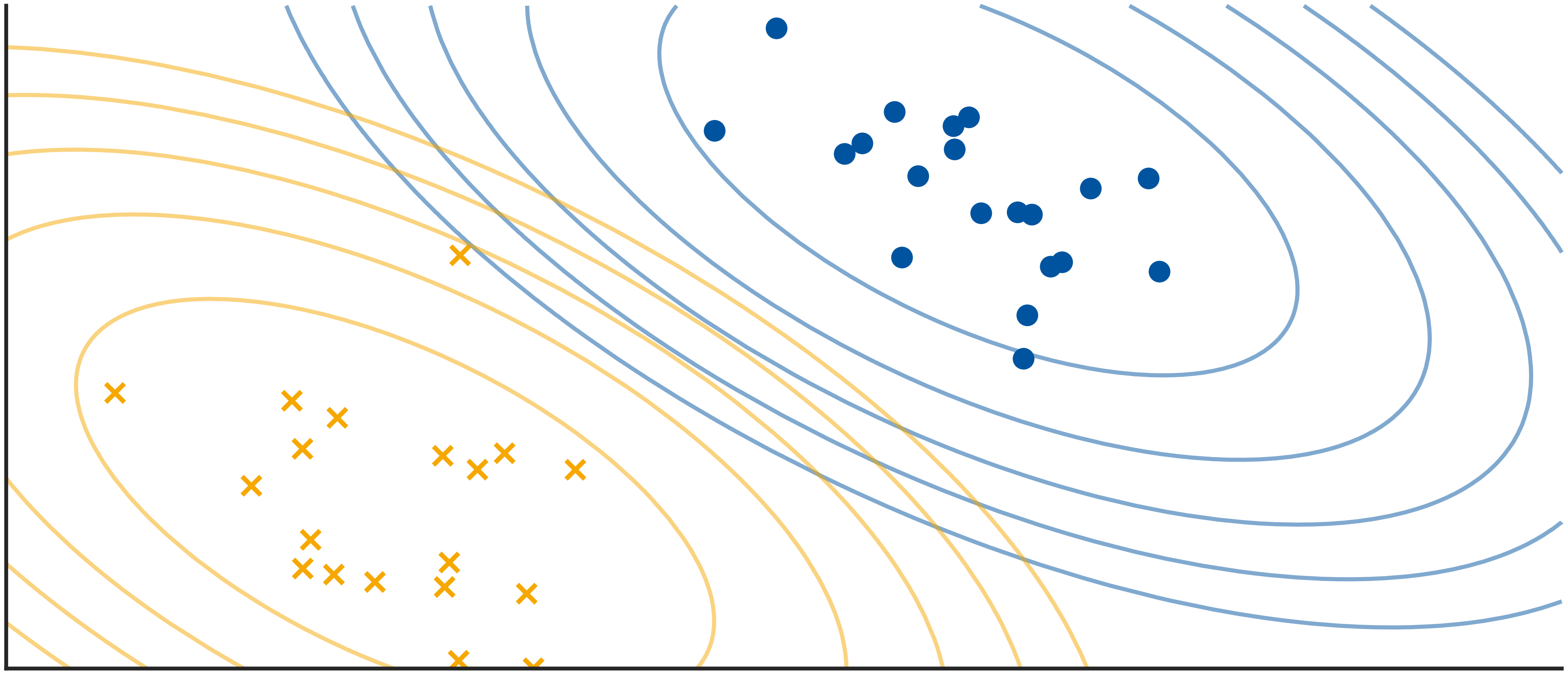
- *Why should we do this?*
  - *What advantage does such a model have compared to direct modeling of the probabilities?*

Example



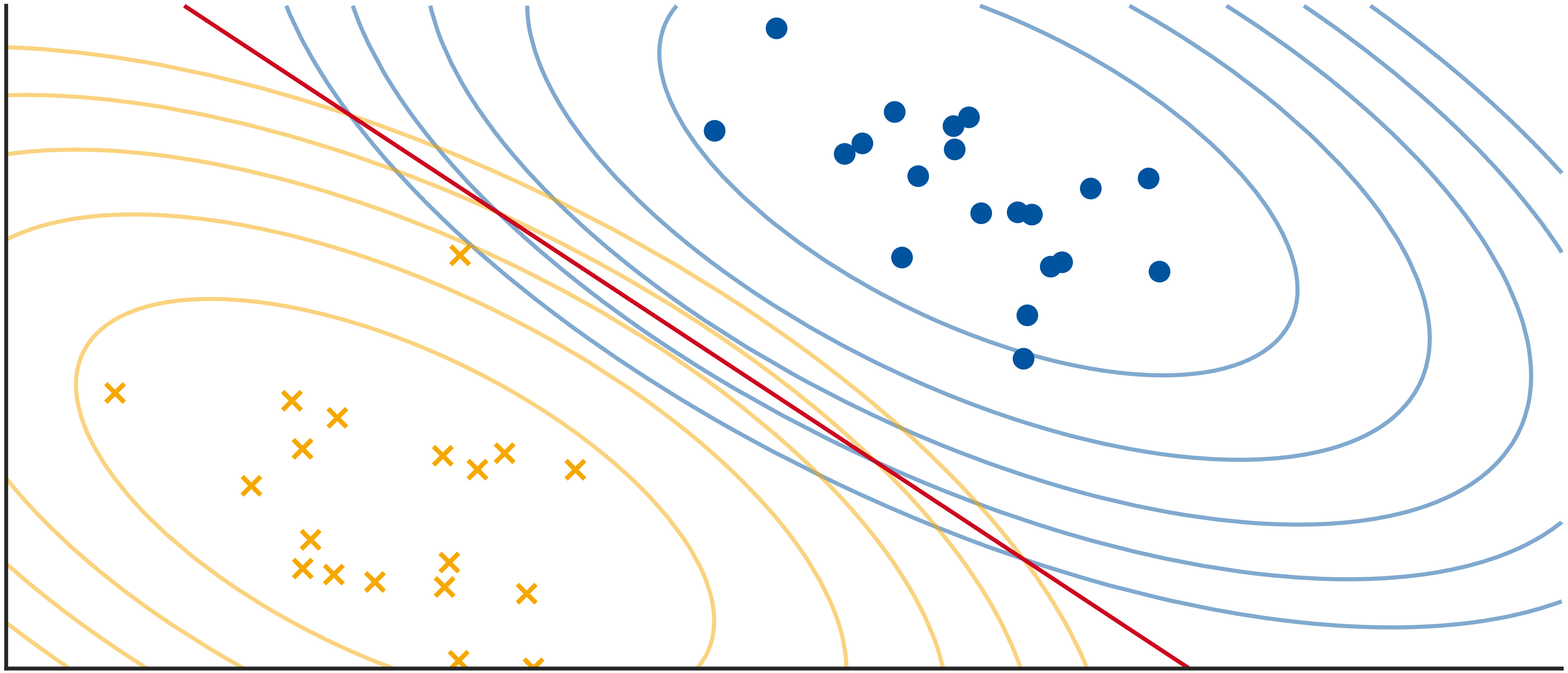
Let's assume the $p(\phi|\mathcal{C}_k)$ are modeled using Gaussians with equal covariances.

Example



Let's assume the $p(\phi|\mathcal{C}_k)$ are modeled using Gaussians with equal covariances.

Example



Let’s assume the $p(\phi|\mathcal{C}_k)$ are modeled using Gaussians with equal covariances.

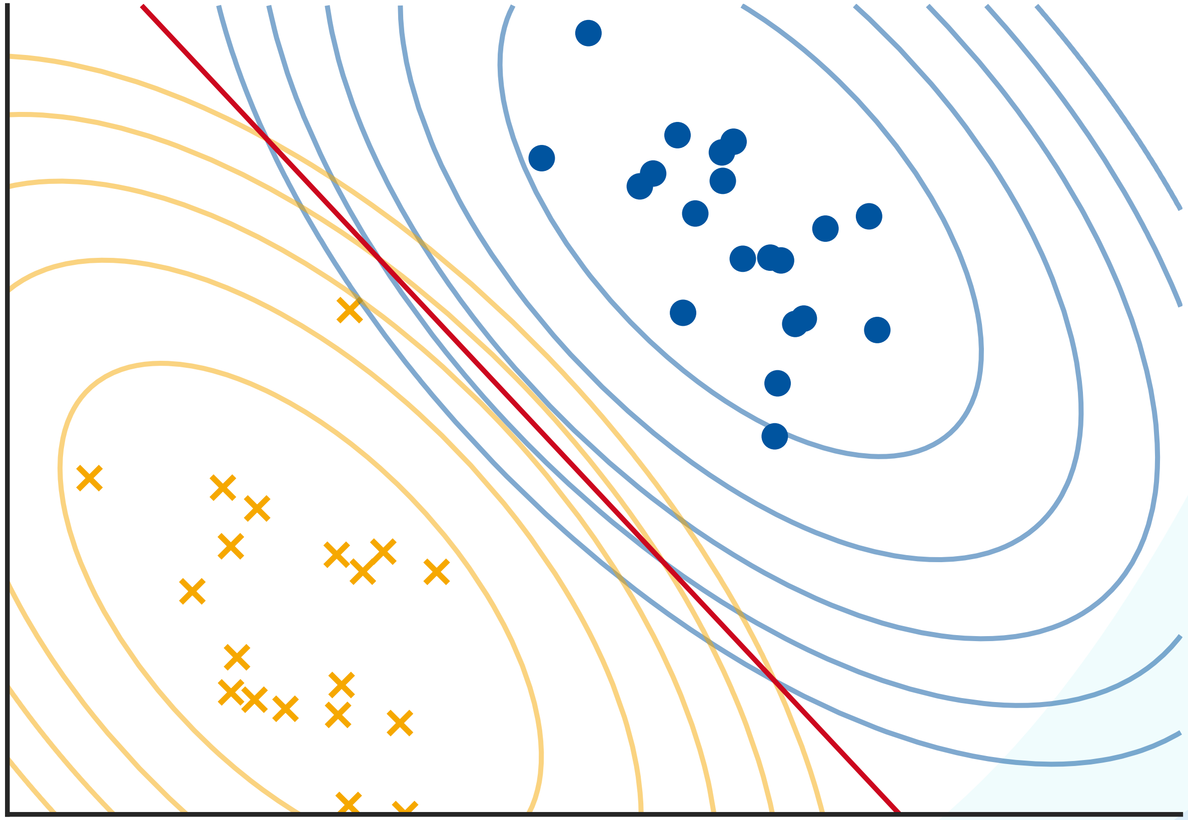
$\Rightarrow$ The decision boundary between them will be linear!

Parameter Efficiency

- #Parameters needed for generative models:
  - Assuming an M-dimensional feature space
  - Prior $p(\mathcal{C}_1)$ 1
  - Means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ $2M$
  - Covariances $\boldsymbol{\Sigma}$ $M(M+1)/2$

  $\Rightarrow$ Total $M(M+5)/2+1$

- #Parameters needed for logistic regression:
  - Weights $\mathbf{w}$ M



$\Rightarrow$ *For large M, logistic regression has clear advantages!*

Discussion: Logistic Regression

Advantages

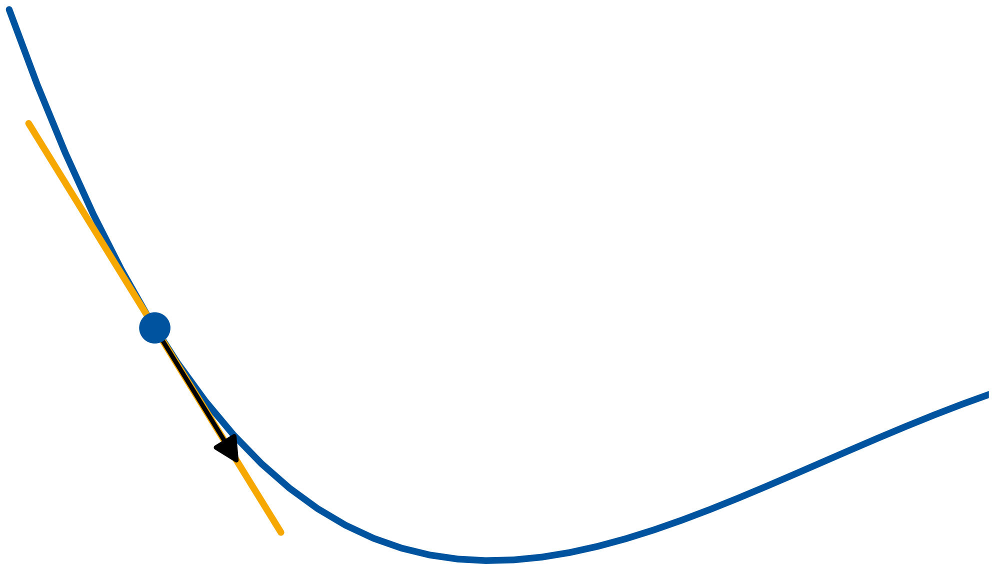
- Nice probabilistic interpretation, directly represents the posterior.
- Requires fewer parameters than modeling the likelihood + prior.
- Cross-Entropy error is convex: unique minimum exists.
- More robust than least-squares.

Limitations

- No closed-form solution, requires iterative optimization approach.

Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. **Iterative Optimization**
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. Error Function Analysis

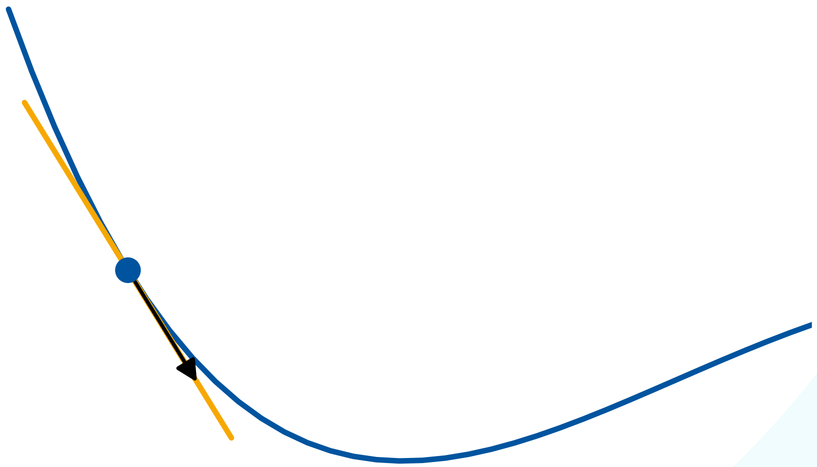


Iterative Optimization

- In general, generalized linear discriminants with nonlinear activation and/or basis functions can no longer be optimized in closed form.

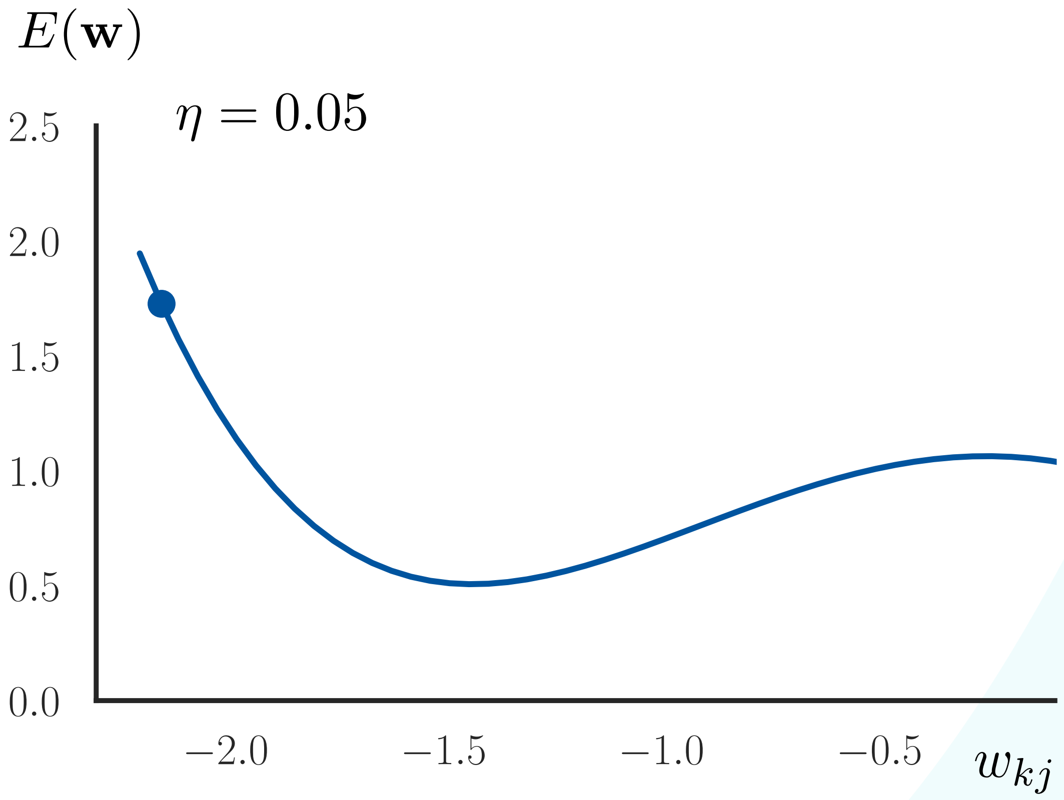
$$y_k(\mathbf{x}) = g\left(\sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x})\right) = g\left(\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x})\right)$$

- Instead, we use iterative optimization schemes.
- Here: Gradient Descent.
  - Start with initial guess for parameter values.
  - Move towards a minimum of the error function by following the direction of steepest descent.
  - Iterate until convergence



Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.

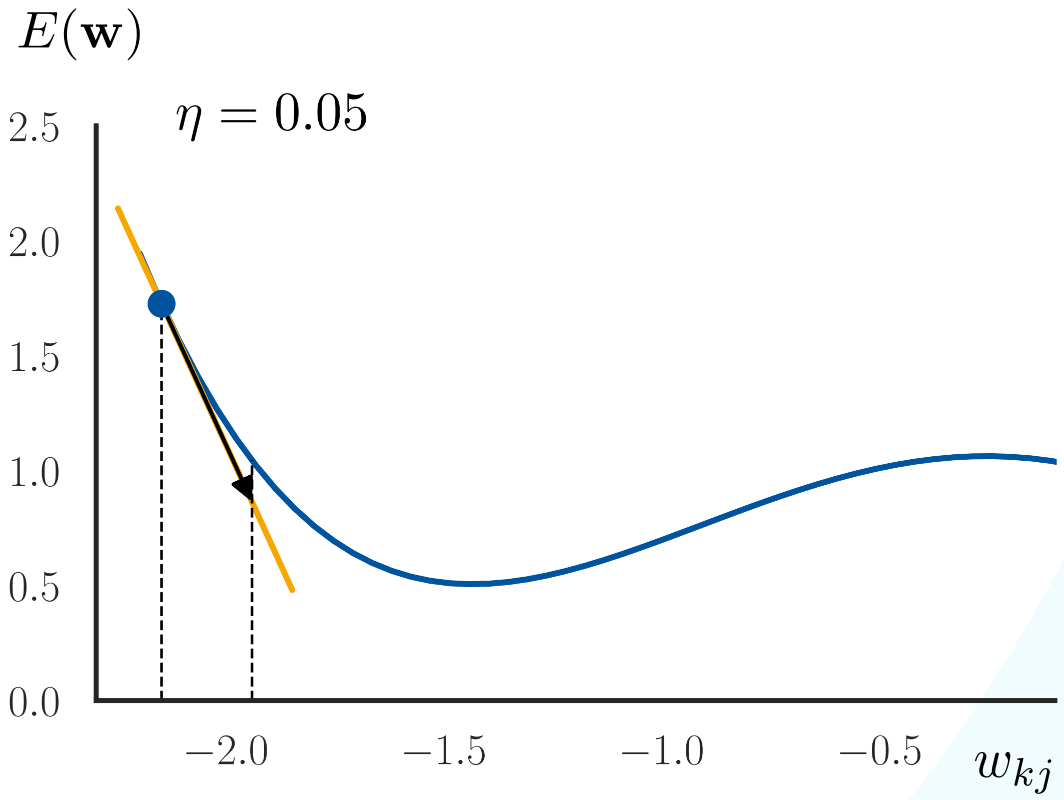


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left.\frac{\partial E(\mathbf{w})}{\partial w_{kj}}\right|_{\mathbf{w}^{(\tau)}}$$

- η is called the learning rate.
- This corresponds to a 1st-order Taylor expansion.
  - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

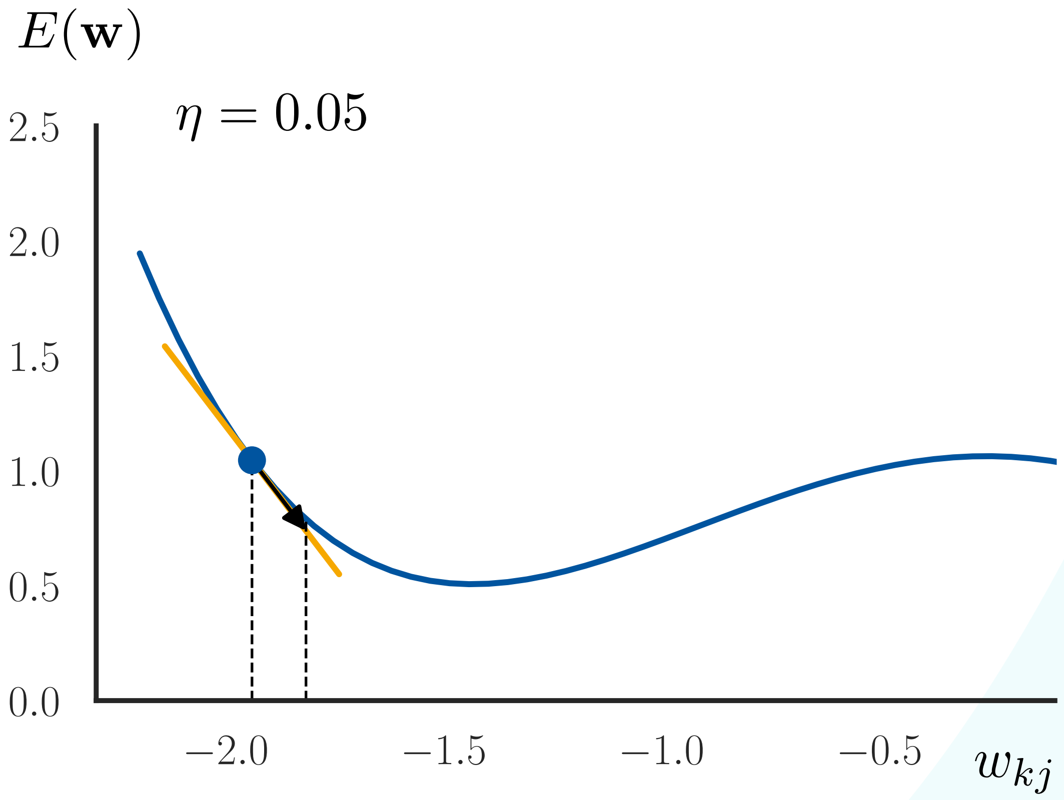


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

- η is called the learning rate.
- This corresponds to a 1st-order Taylor expansion.
    - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

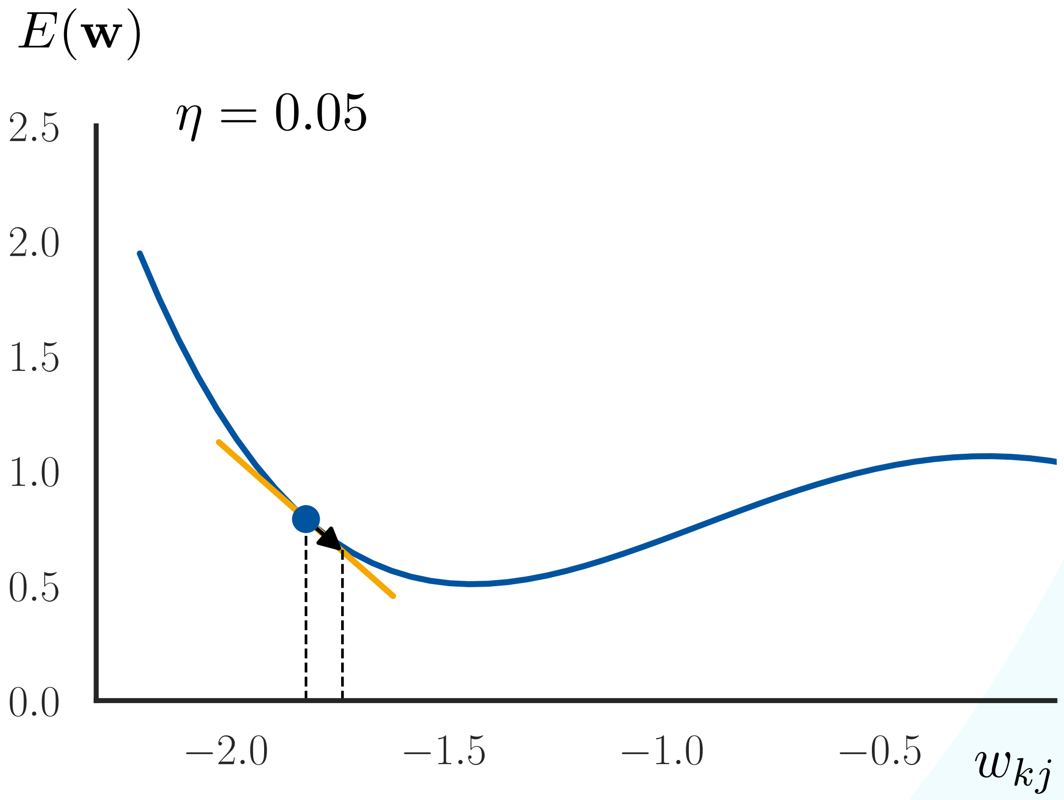


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

- η is called the learning rate.
- This corresponds to a 1st-order Taylor expansion.
  - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.

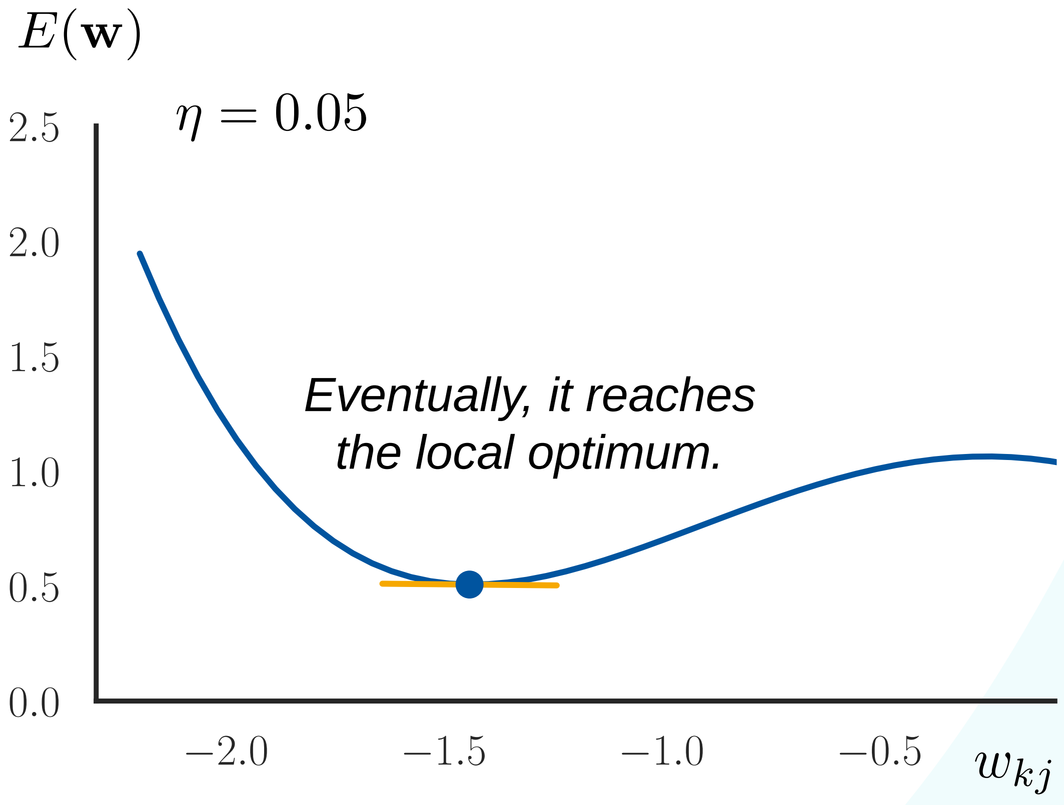


Idea: Gradient Descent

- Start with an initial guess of parameter values $w_{kj}^{(0)}$.
- Follow the gradient to move to a (local) minimum:

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

- η is called the learning rate.
- This corresponds to a 1st-order Taylor expansion.
  - I.e., we approximate the error function by its tangent plane around the current point $\mathbf{w}^{(\tau)}$.
- Repeat this procedure for a number of steps.



Discussion: Gradient Descent

Advantages

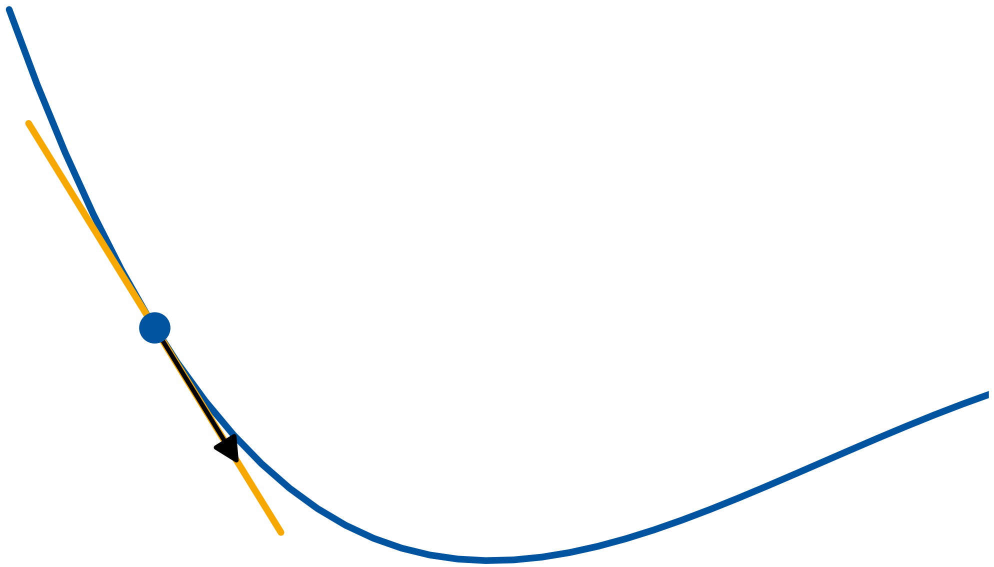
- Simple approach for iterative optimization.
- Approximates the error function by its tangent plane around the current point in order to find the direction of steepest descent.

Limitations

- Local optimization. Unless the error function is convex, will only converge to a local optimum.
- Relatively slow convergence (can be improved by second-order approaches).
- In practice, finding a good step size (learning rate) is important for fast convergence.

Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Optimization
4. **First-Order Gradient Descent**
5. Second-Order Gradient Descent
6. Error Function Analysis



First-order Optimization

- Logistic regression uses the binary cross-entropy error:

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \left(t_n \ln y(\mathbf{x}_n; \mathbf{w}) + (1 - t_n) \ln(1 - y(\mathbf{x}_n; \mathbf{w}))\right)$$

- Properties
  - Convex function, so it has a unique minimum
  - But no closed-form solution

- We need to use iterative methods for optimization
  - Let's try (first-order) gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

Gradient of the Cross-Entropy Error

$$E(\mathbf{w}) = -\sum_{n=1}^{N} (t_n \ln y_n + (1-t_n)\ln(1-y_n))$$

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N} \left(t_n \frac{\frac{\partial}{\partial \mathbf{w}} y_n}{y_n} + (1-t_n) \frac{\frac{\partial}{\partial \mathbf{w}}(1-y_n)}{(1-y_n)} \right)$$

$$= -\sum_{n=1}^{N} \left(t_n \frac{y_n(1-y_n)}{y_n}\phi_n + (1-t_n)\frac{y_n(1-y_n)}{(1-y_n)}\phi_n \right)$$

$$= -\sum_{n=1}^{N} ((t_n - t_n y_n - y_n + t_n y_n)\phi_n)$$

$$= \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

$$y_n = y(\mathbf{x}_n; \mathbf{w})$$

$$\sigma'(a) = \sigma(a)(1-\sigma(a))$$

$$\frac{\partial y_n}{\partial \mathbf{w}} = y_n(1-y_n)\phi_n$$

$$\phi_n = \phi(\mathbf{x}_n)$$

- The gradient for logistic regression is

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n$$

- We can plug this into gradient descent:

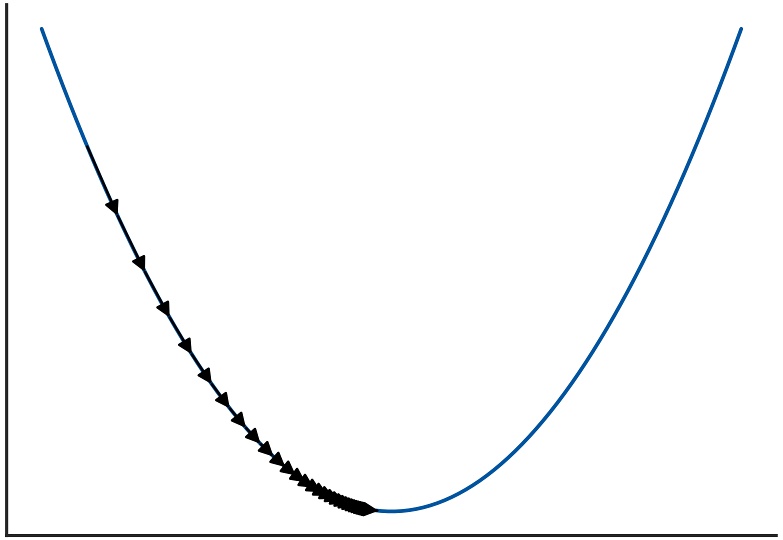
$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}) \\ &= \mathbf{w}^{(\tau)} - \eta \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n\end{aligned}$$

How should we choose the learning rate?

- This update rule is known as the Delta rule (= LMS rule)
  - *Simply feed back the input data points, weighted by the classification error.*

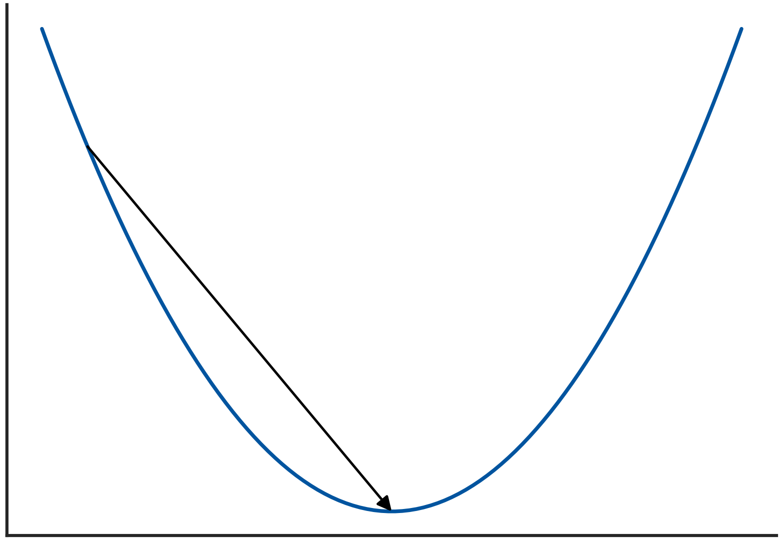
Effects of the learning rate

η too small



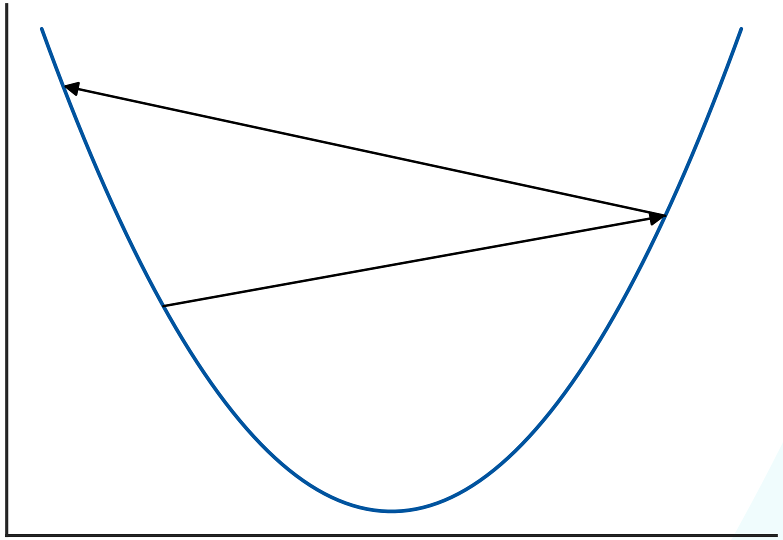
Convergence is slow

η_{opt}



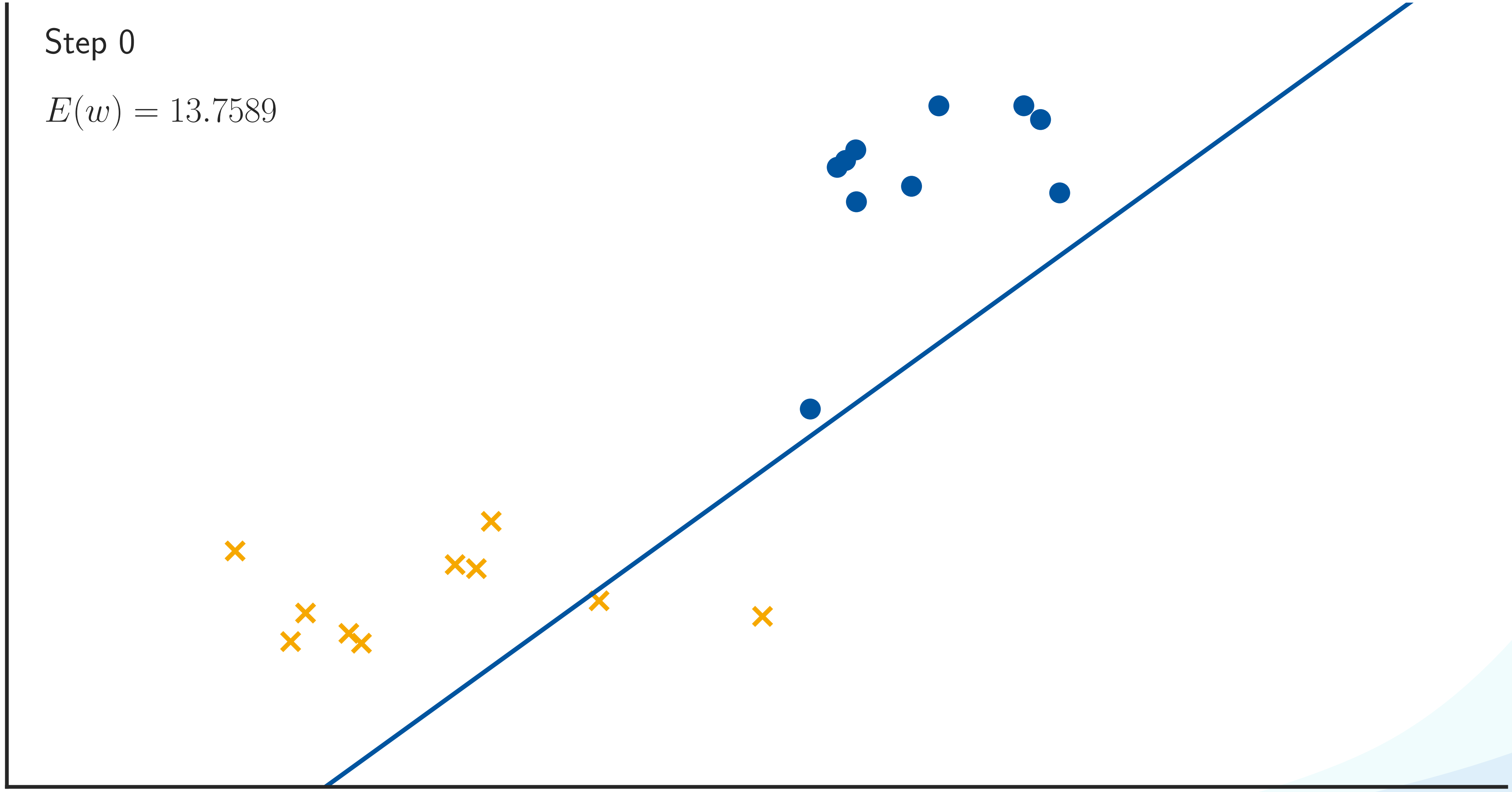
Converges ideally in
a single step

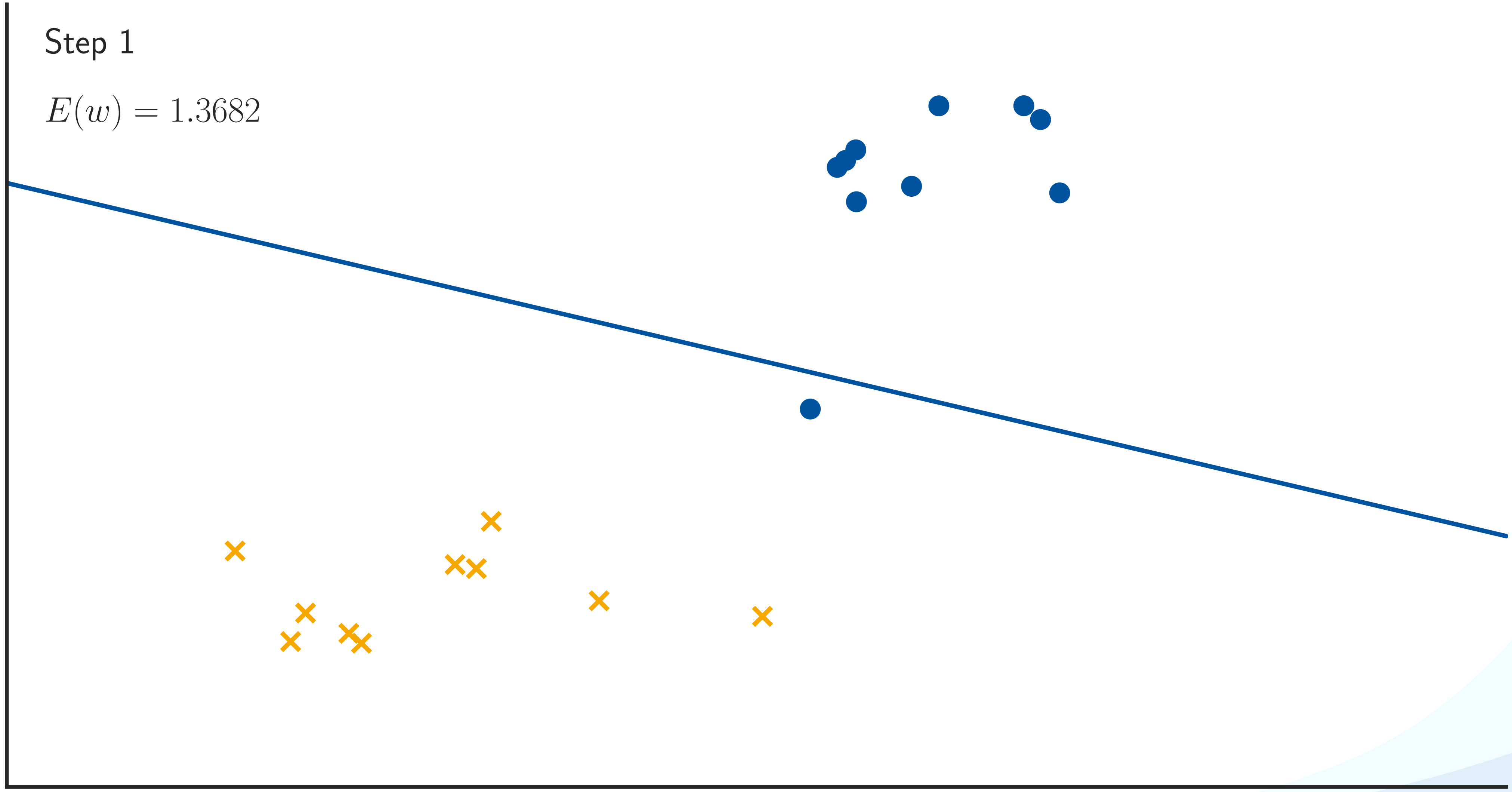
η too large

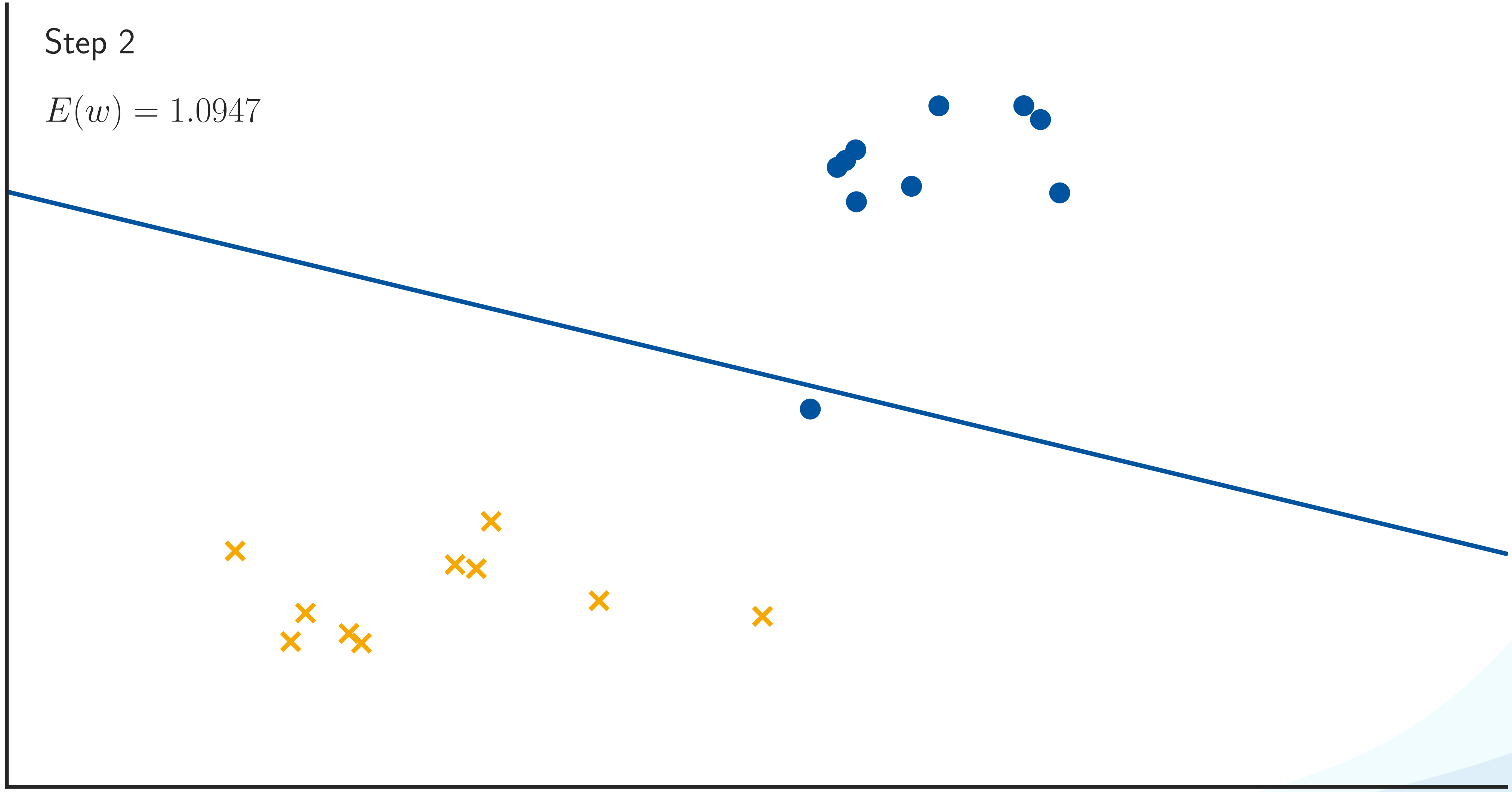


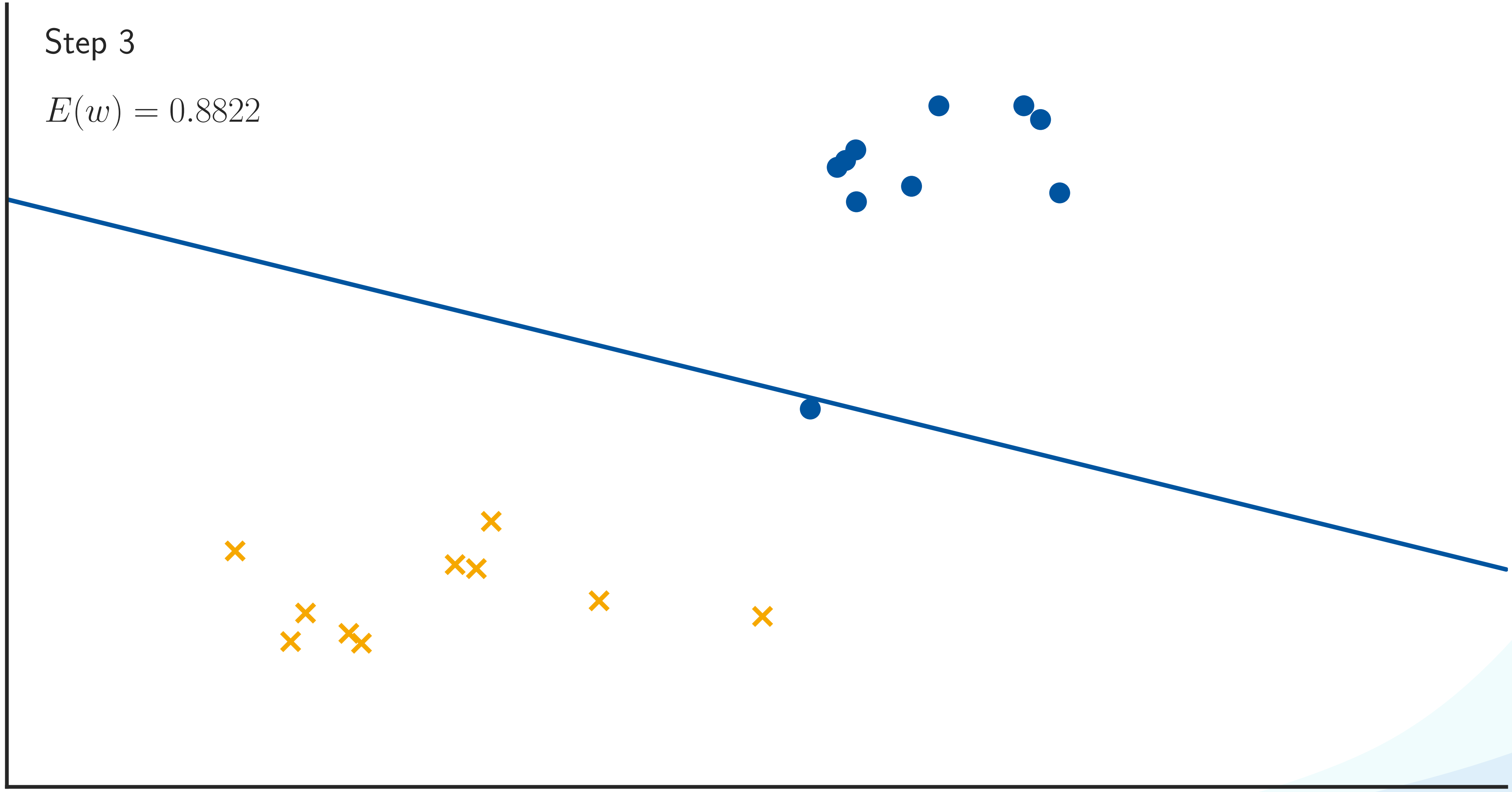
Might not converge

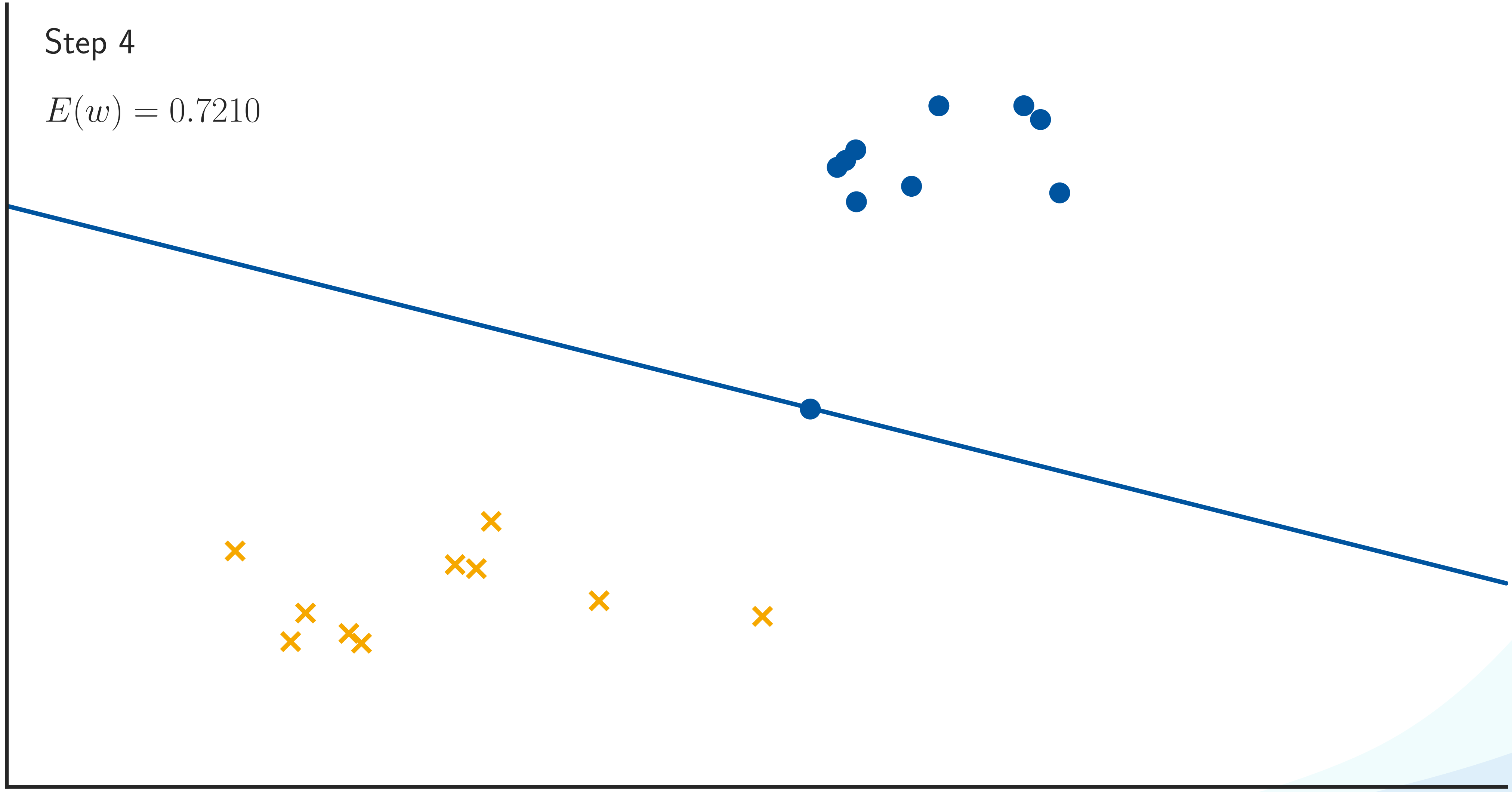
Example: Logistic Regression with Gradient Descent

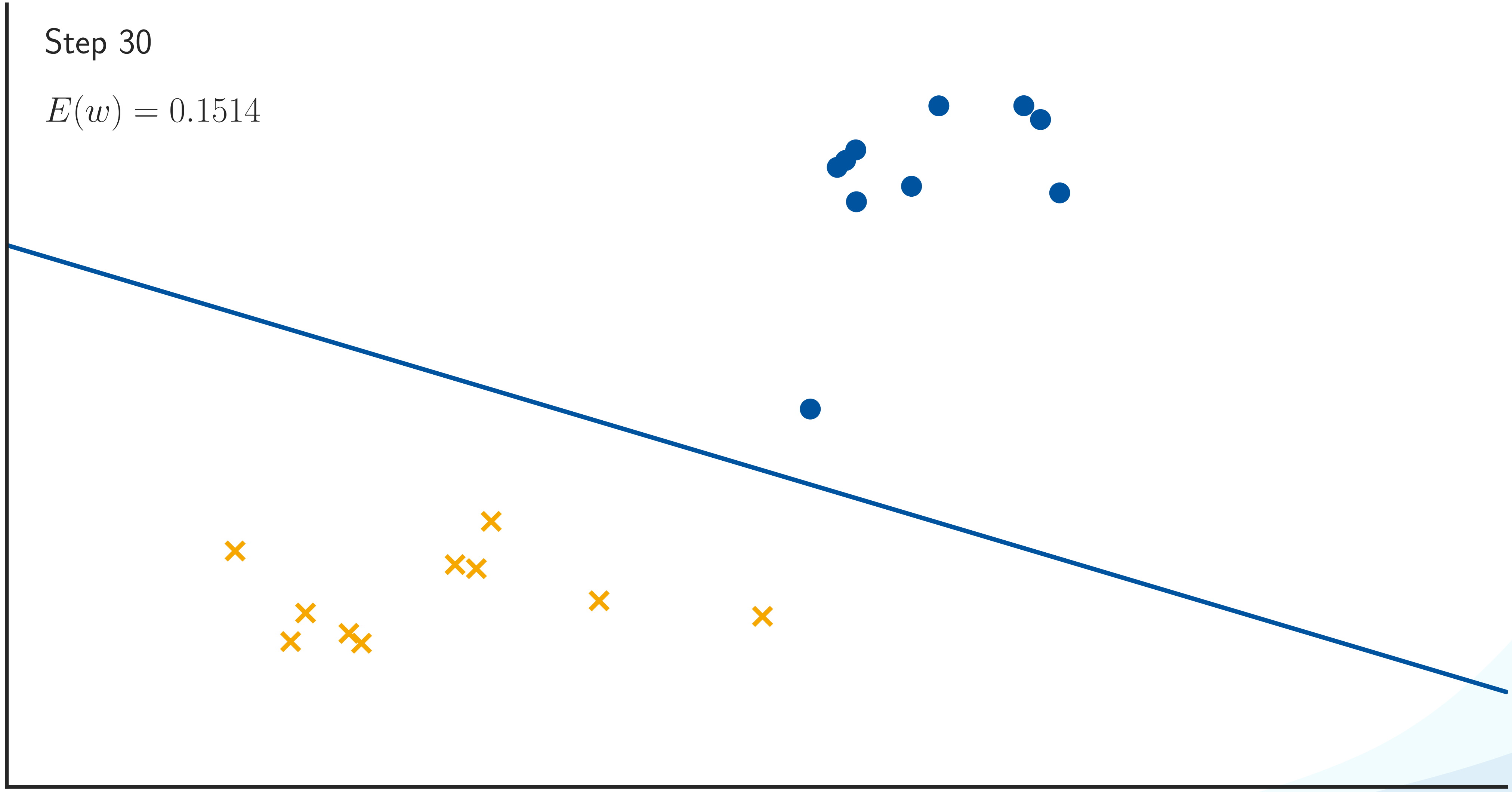


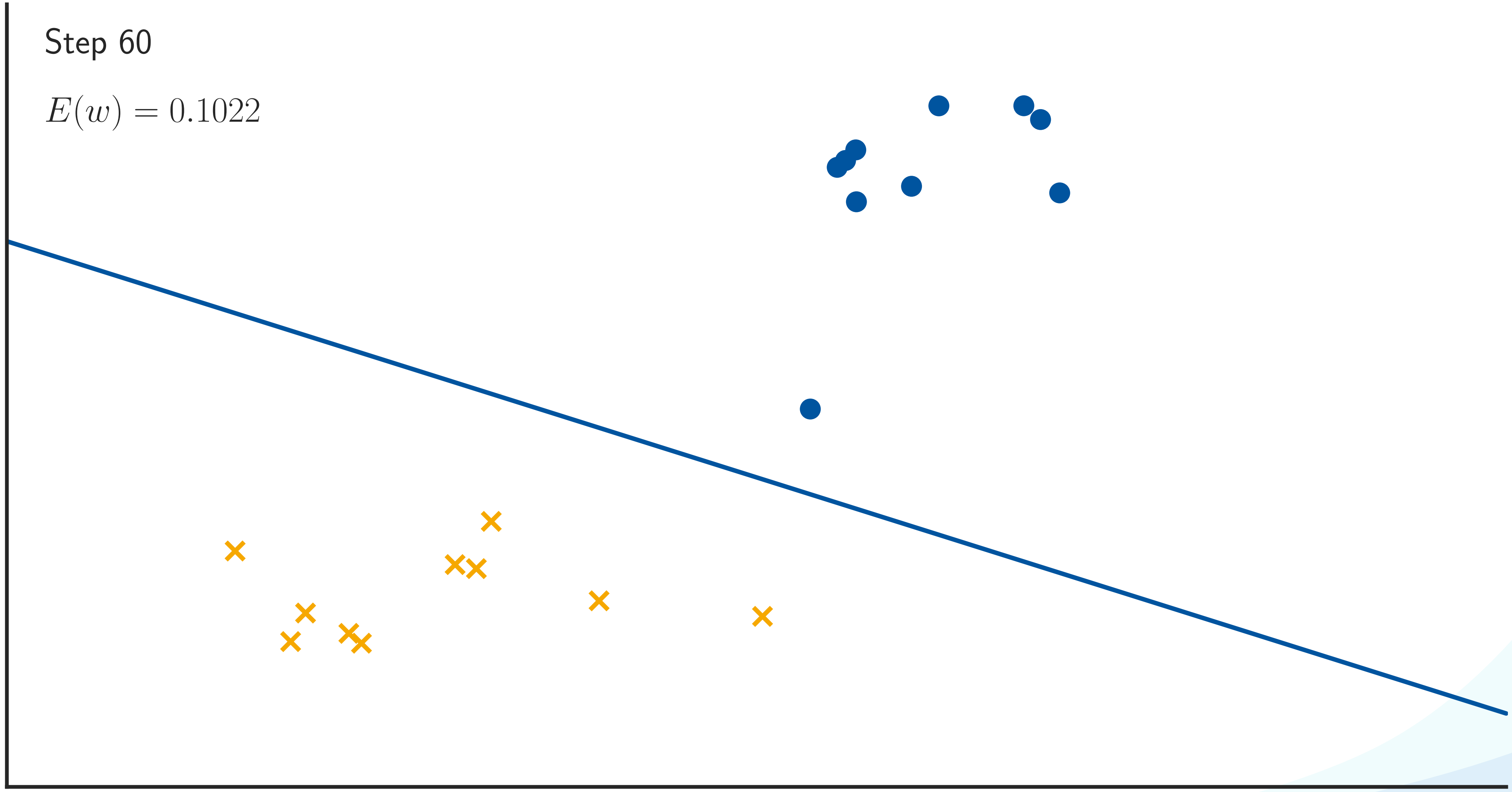

Step 1
$E(w) = 1.3682$

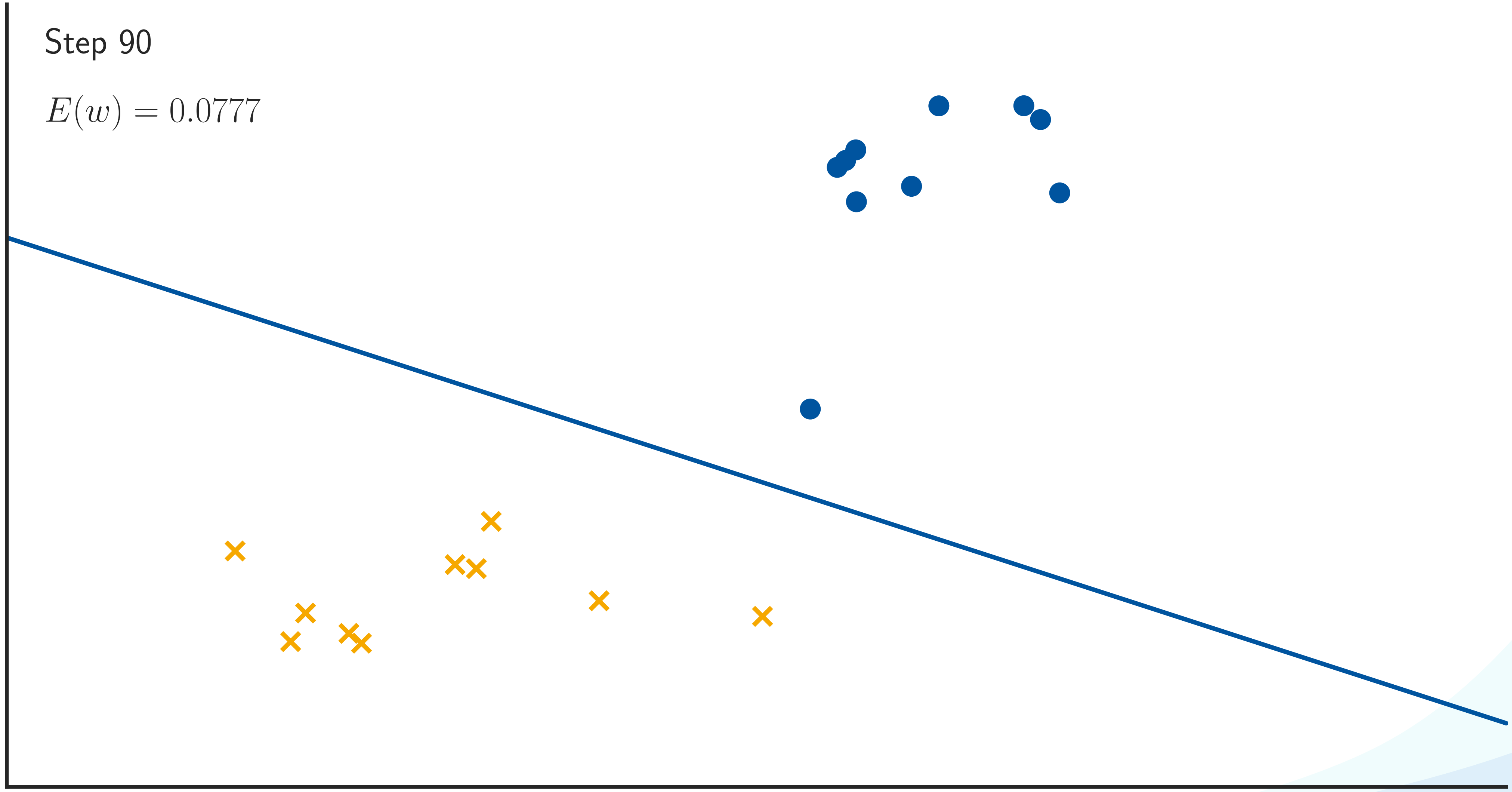

Step 2
$E(w) = 1.0947$


Step 3
$E(w) = 0.8822$


Step 4
$E(w) = 0.7210$


Step 30
$E(w) = 0.1514$


Step 60
$E(w) = 0.1022$


Step 90
$E(w) = 0.0777$

Discussion: Logistic Regression with Gradient Descent

Advantages

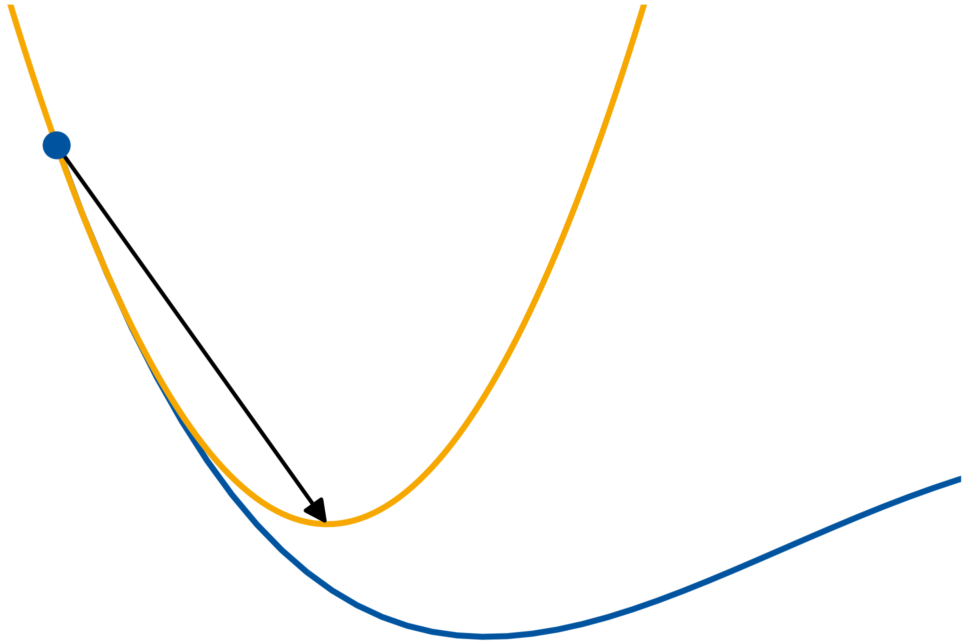
- Simple iterative optimization scheme with a familiar update rule (Delta rule).

Limitations

- Slow convergence
- Need to choose a suitable learning rate.

Logistic Regression

1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Optimization
4. First-Order Gradient Descent
5. **Second-Order Gradient Descent**
6. Error Function Analysis



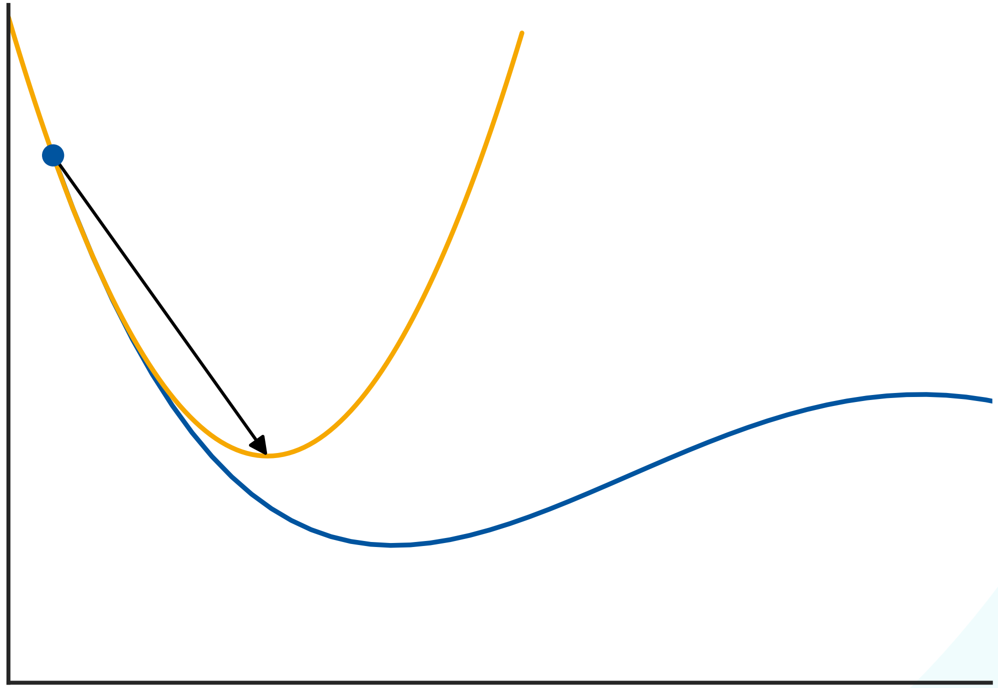
Second-Order Optimization

- So far, we have optimized the cross-entropy error with gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w})$$

- This is a first-order approximation, and it heavily depends on the learning rate η.

- Instead, we can apply a second-order optimization scheme that converges faster and is independent of the learning rate.



Newton-Raphson Gradient Descent

- Second-order Newton-Raphson update scheme:

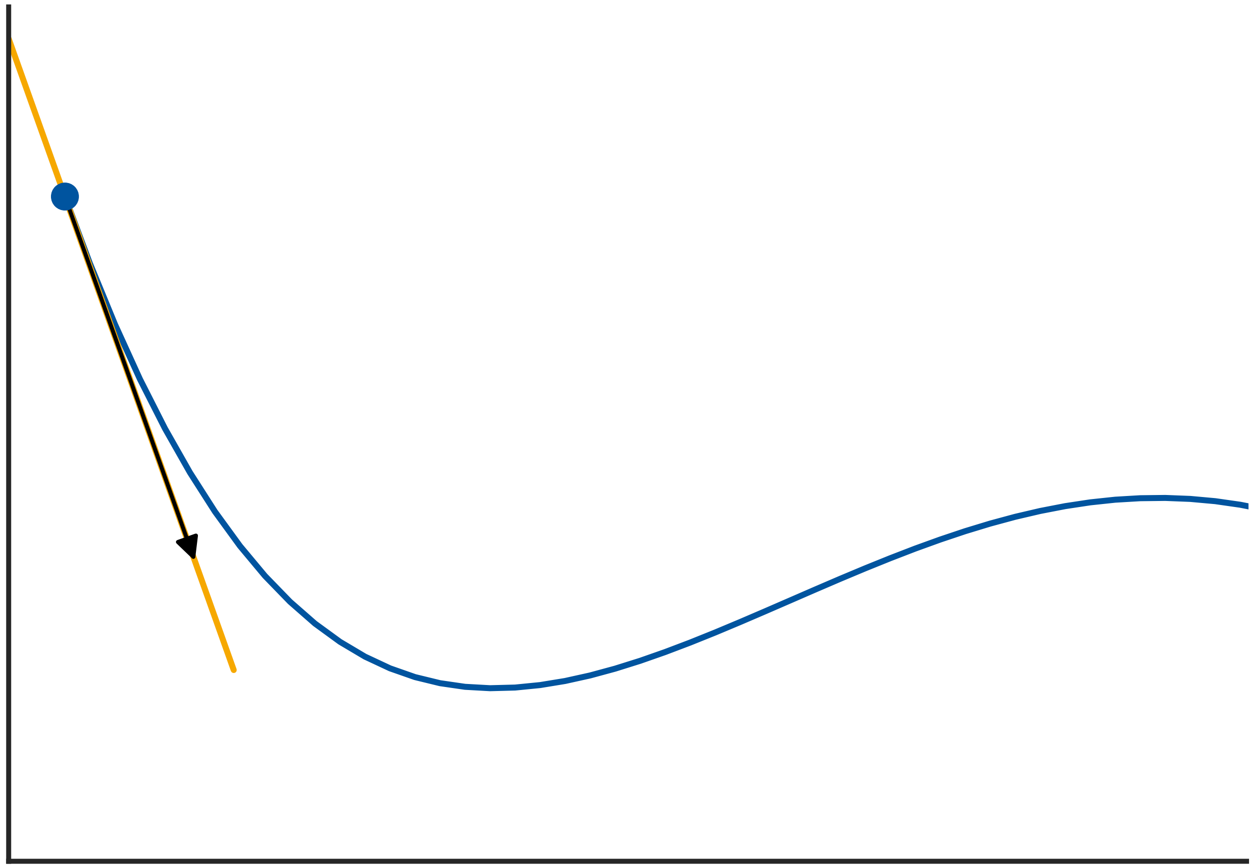
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

- Here, $\mathbf{H} = \nabla\nabla E(\mathbf{w})$ is the Hessian matrix, i.e., the matrix of second derivatives:

$$\mathbf{H}_{ij} = \frac{\partial^2 E(\mathbf{w})}{\partial w_i \partial w_j}$$

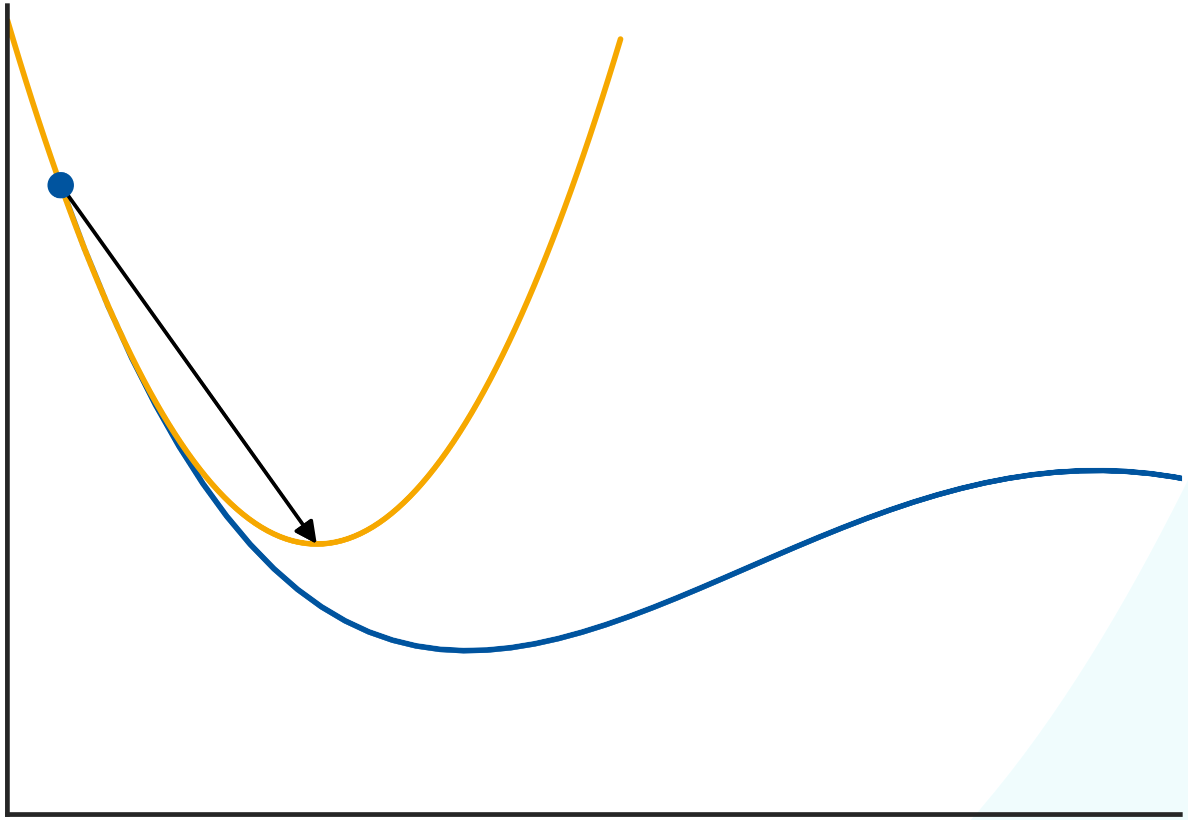
- Properties
  - Local quadratic approximation
  - Much faster convergence by taking into account the curvature of the error function.

Intuition



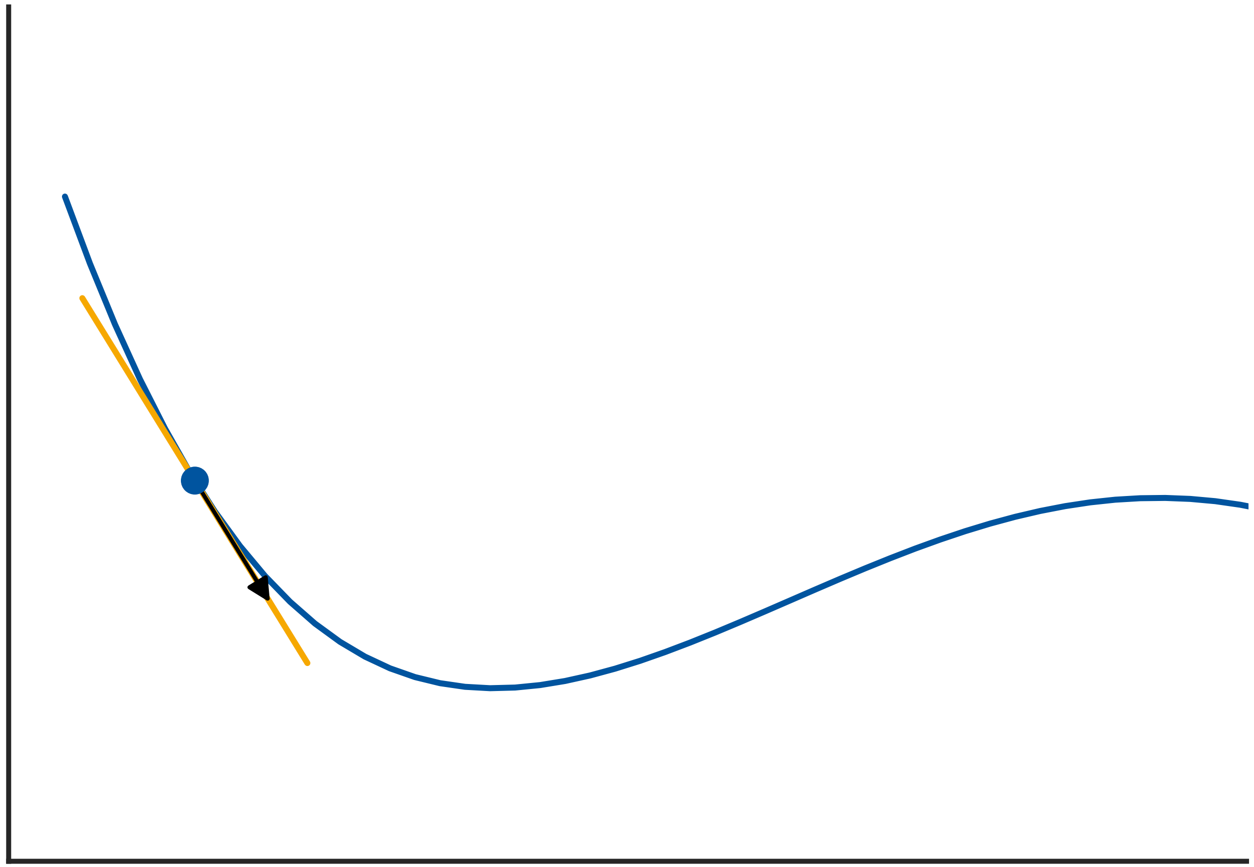
First-Order

$\eta = 0.005$



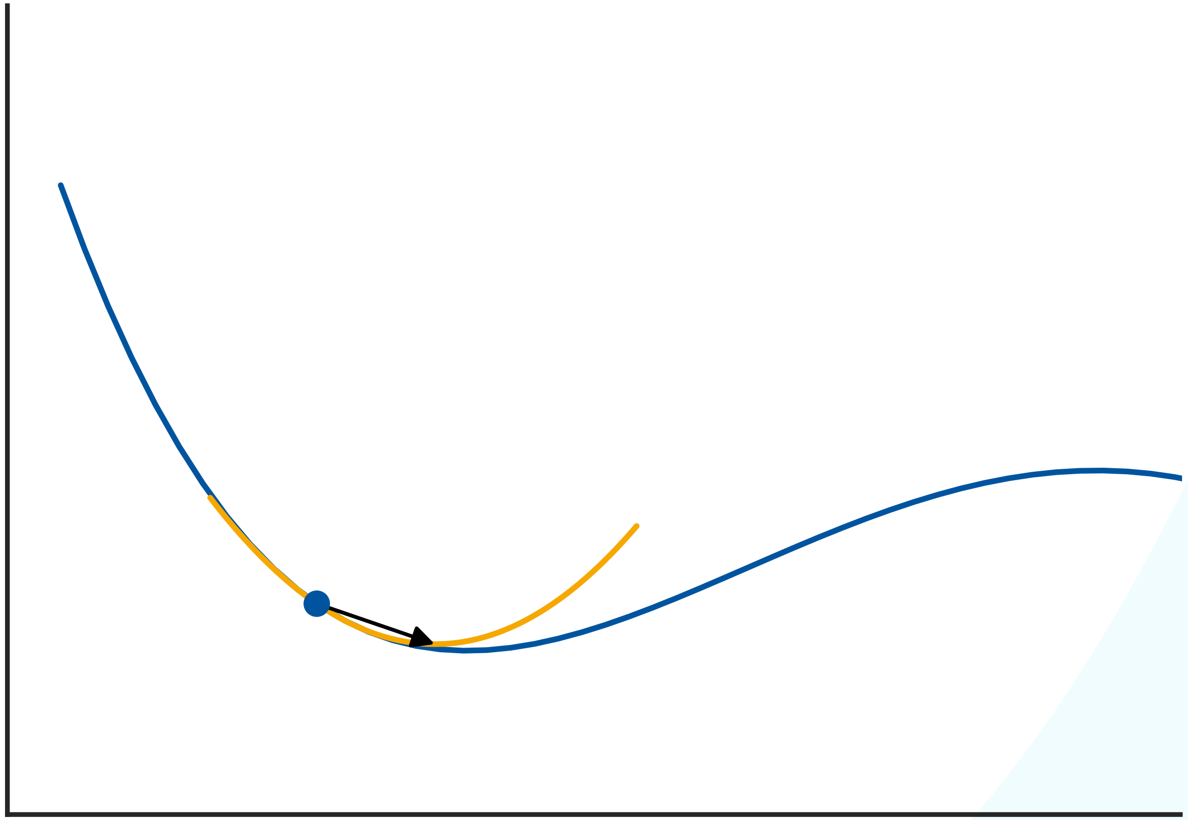
Second-Order

Intuition



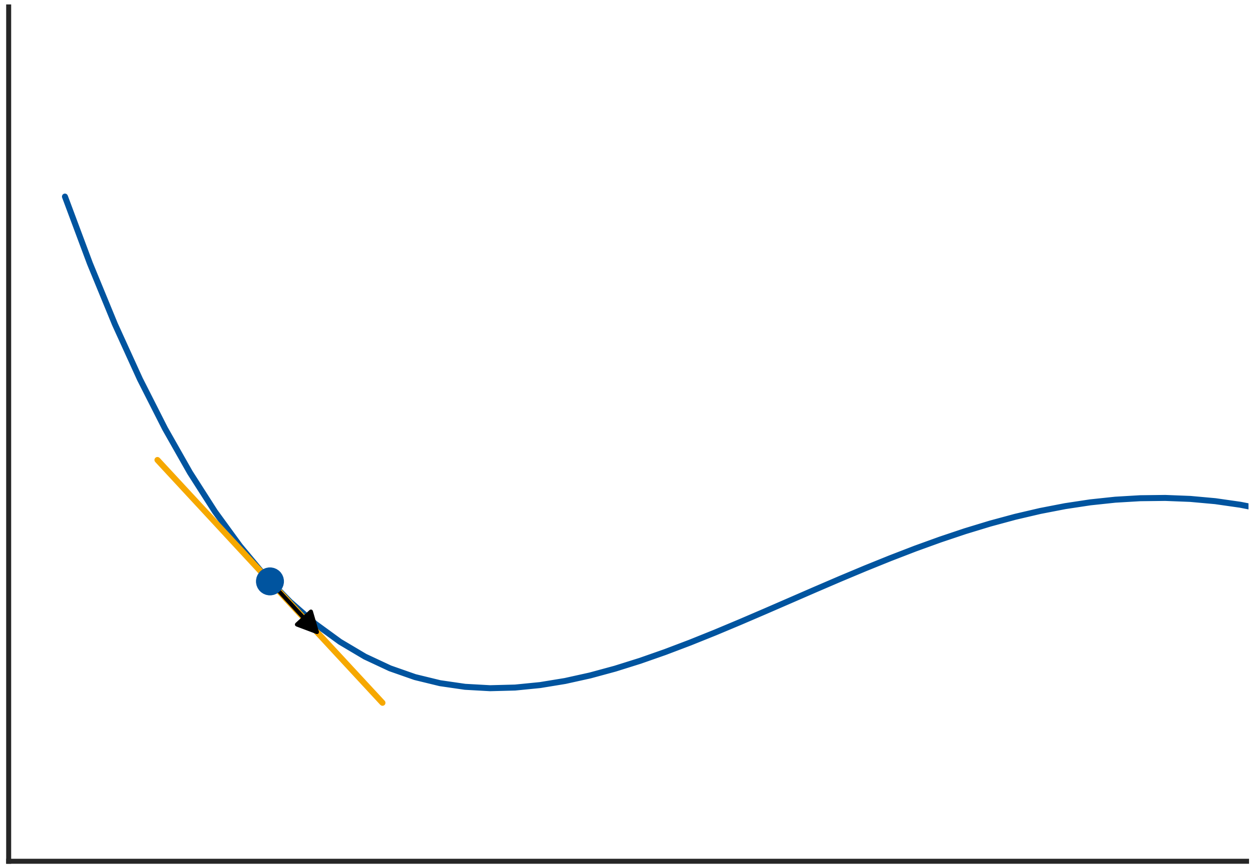
First-Order

$\eta = 0.005$



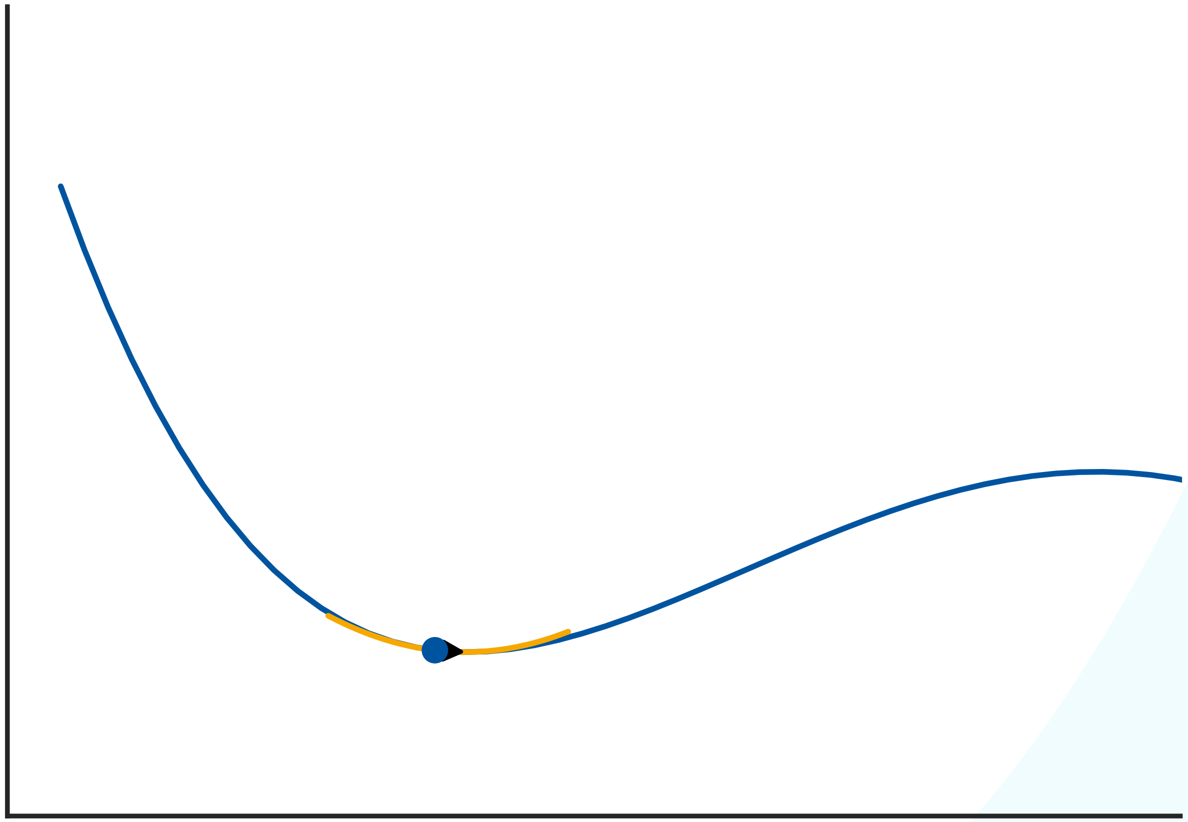
Second-Order

Intuition



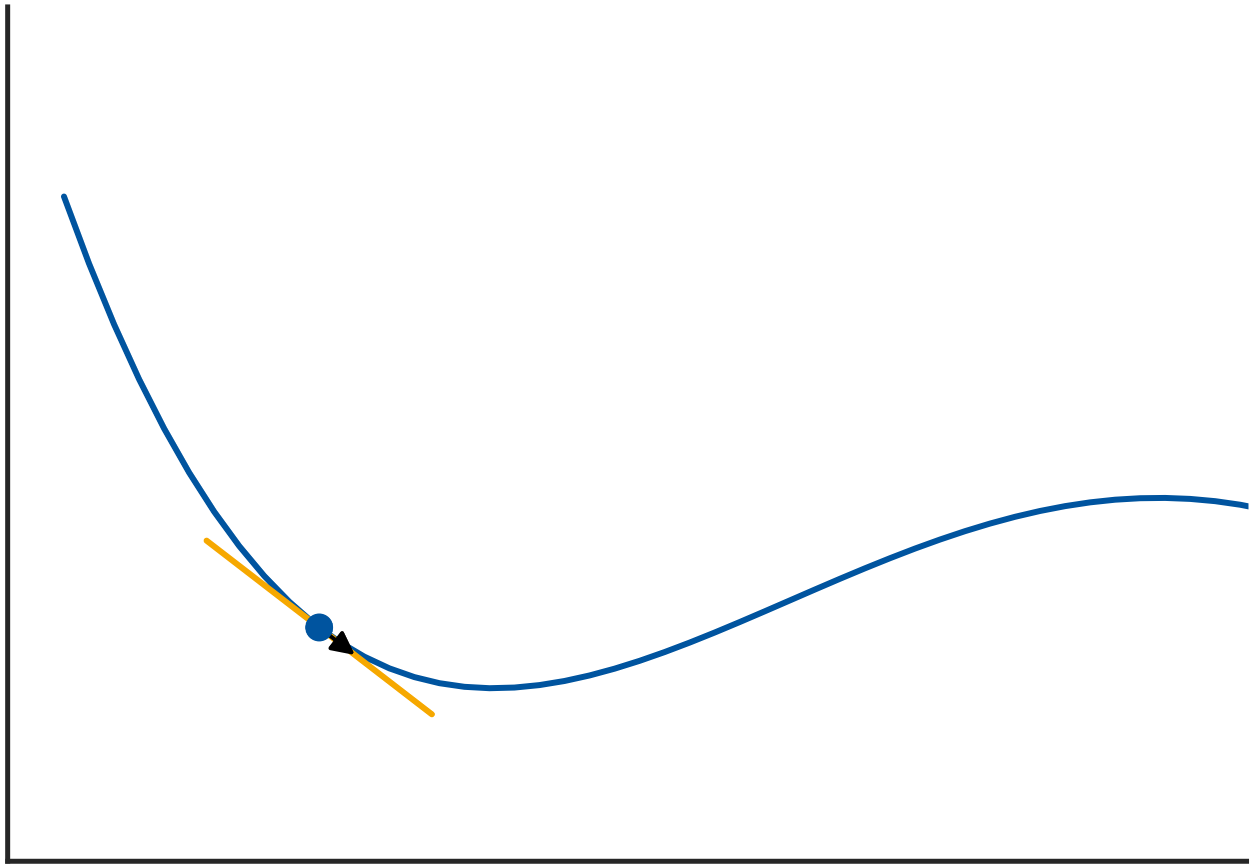
First-Order

$\eta = 0.005$



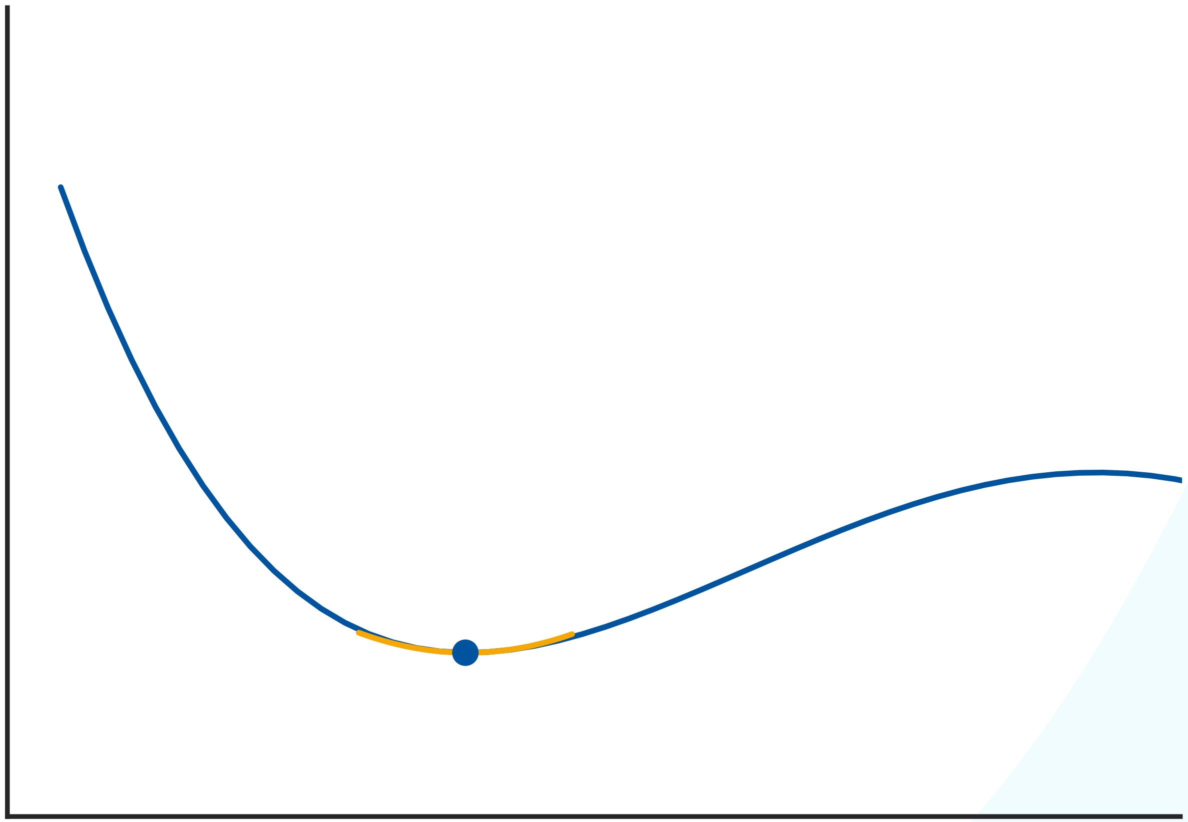
Second-Order

Intuition



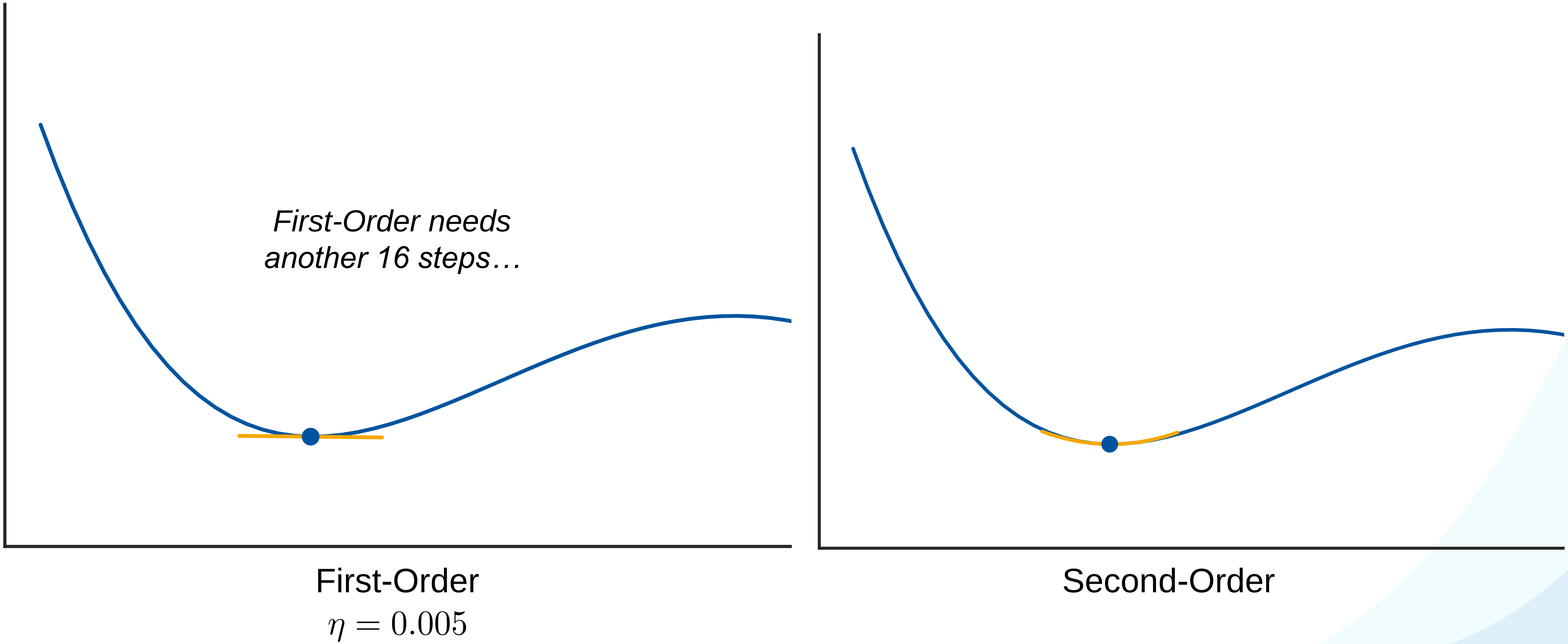
First-Order

$\eta = 0.005$



Second-Order

Intuition


First-Order needs
another 16 steps…
First-Order
$\eta = 0.005$
Second-Order

Newton-Raphson for Least-Squares

- First, we apply it to least-squares:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^\top \phi_n - t_n)^2$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^\top \phi_n - t_n)\phi_n \quad = \mathbf{\Phi}^\top \mathbf{\Phi}\mathbf{w} - \mathbf{\Phi}^\top \mathbf{t}$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \phi_n \phi_n^\top = \mathbf{\Phi}^\top \mathbf{\Phi}$$

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1^\top \\ \vdots \\ \phi_N^\top \end{pmatrix}$$

- Resulting update scheme (normal equations):

$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1}(\mathbf{\Phi}^\top \mathbf{\Phi}\mathbf{w}^{(\tau)} - \mathbf{\Phi}^\top \mathbf{t}) \\ &= (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1}\mathbf{\Phi}^\top \mathbf{t}\end{aligned}$$

This is the closed-form solution of the least-squares objective!

Newton-Raphson for the Cross-Entropy Error

- Now, let's try Newton-Raphson on the cross-entropy error function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n \ln y_n + (1-t_n)\ln(1-y_n))$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^{\mathsf{T}} = \boldsymbol{\Phi}^{\mathsf{T}}\mathbf{R}\boldsymbol{\Phi}$$

$$\sigma'(a) = \sigma(a)(1-\sigma(a))$$

$$\frac{\partial y_n}{\partial \mathbf{w}} = y_n(1-y_n)\boldsymbol{\phi}_n$$

- where $\mathbf{R} \in \mathbb{R}^{N\times N}$ is an $N \times N$ diagonal matrix with $R_{nn} = y_n(1-y_n)$.
- The Hessian now depends on $\mathbf{w}$ through the weighting matrix $\mathbf{R}$.

Iteratively Reweighted Least Squares (IRLS)

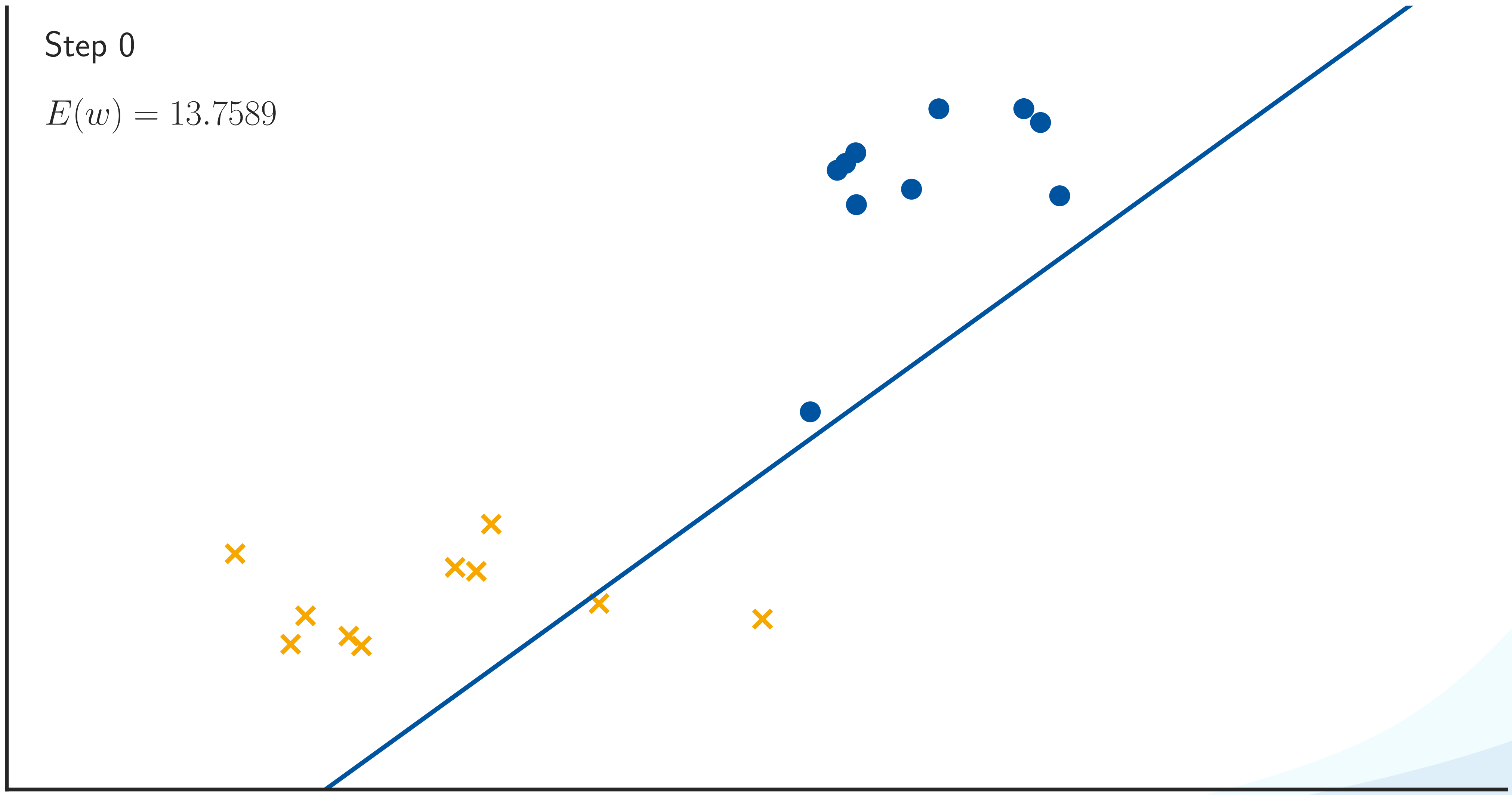
- Update equations:

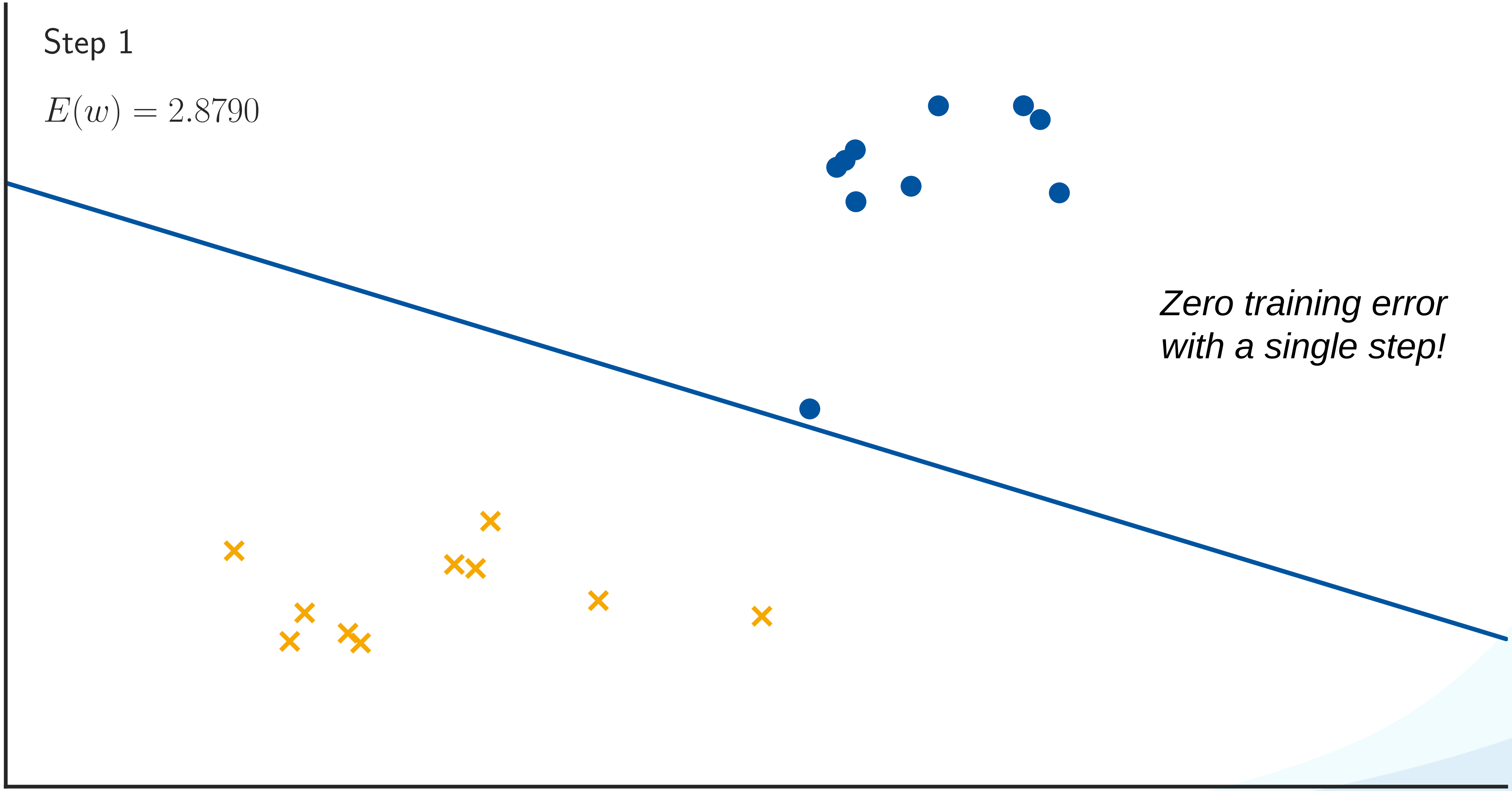
$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\boldsymbol{\Phi}^\top \mathbf{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top (\mathbf{y} - \mathbf{t}) \\ &= (\boldsymbol{\Phi}^\top \mathbf{R} \boldsymbol{\Phi})^{-1} \left(\boldsymbol{\Phi}^\top \mathbf{R} \boldsymbol{\Phi} \mathbf{w}^{(\tau)} - \boldsymbol{\Phi}^\top (\mathbf{y} - \mathbf{t}) \right) \\ &= (\boldsymbol{\Phi}^\top \mathbf{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{R} \mathbf{z} \end{aligned}$$

$$\text{with} \quad \mathbf{z} = \boldsymbol{\Phi} \mathbf{w}^{(\tau)} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})$$

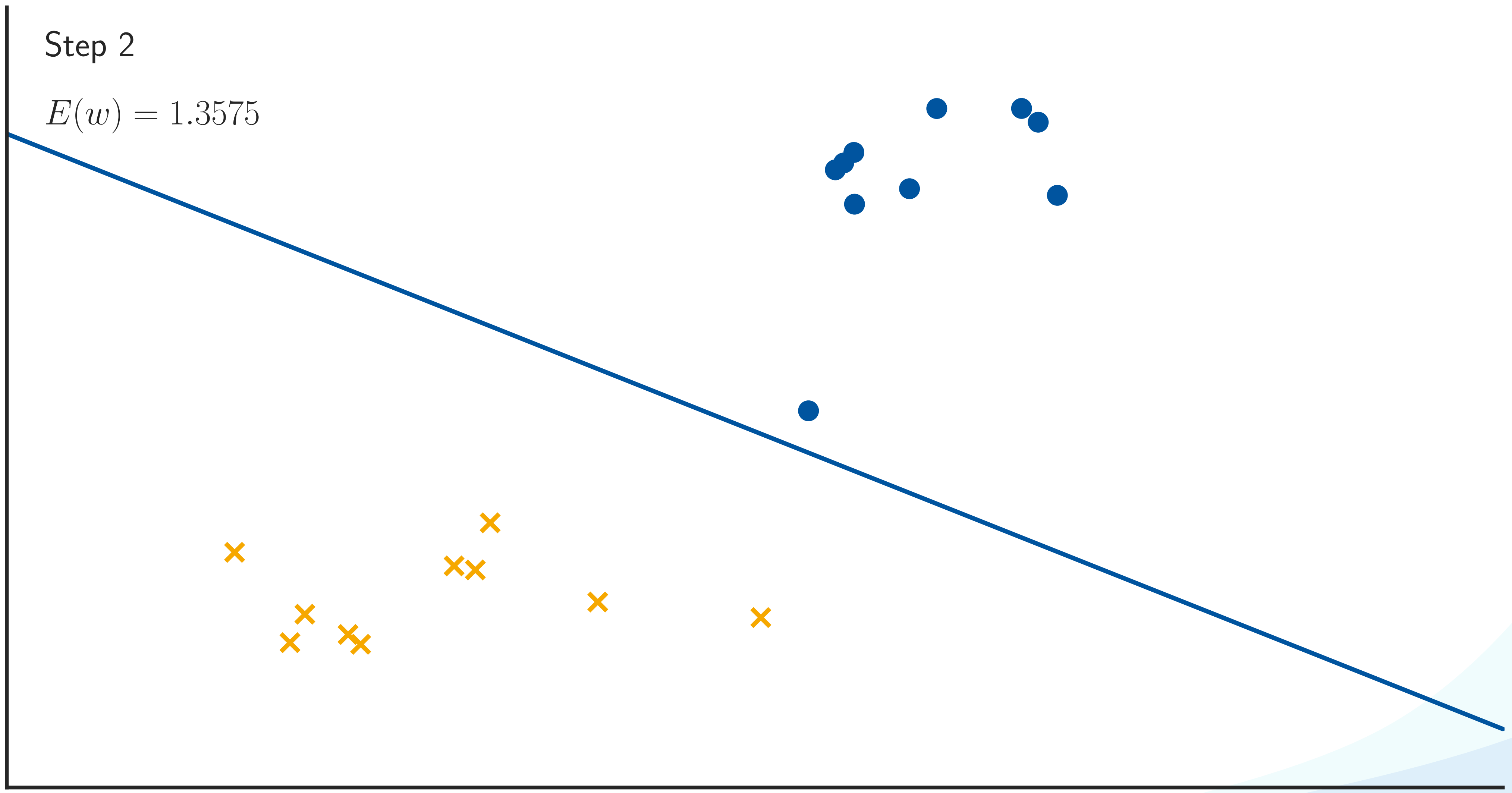
- Very similar form (normal equations).
  - But now with non-constant weighting matrix $\mathbf{R}$ (depends on $\mathbf{w}$).
  - Need to apply normal equations iteratively.
  - This is called Iteratively Reweighted Least-Squares (IRLS).

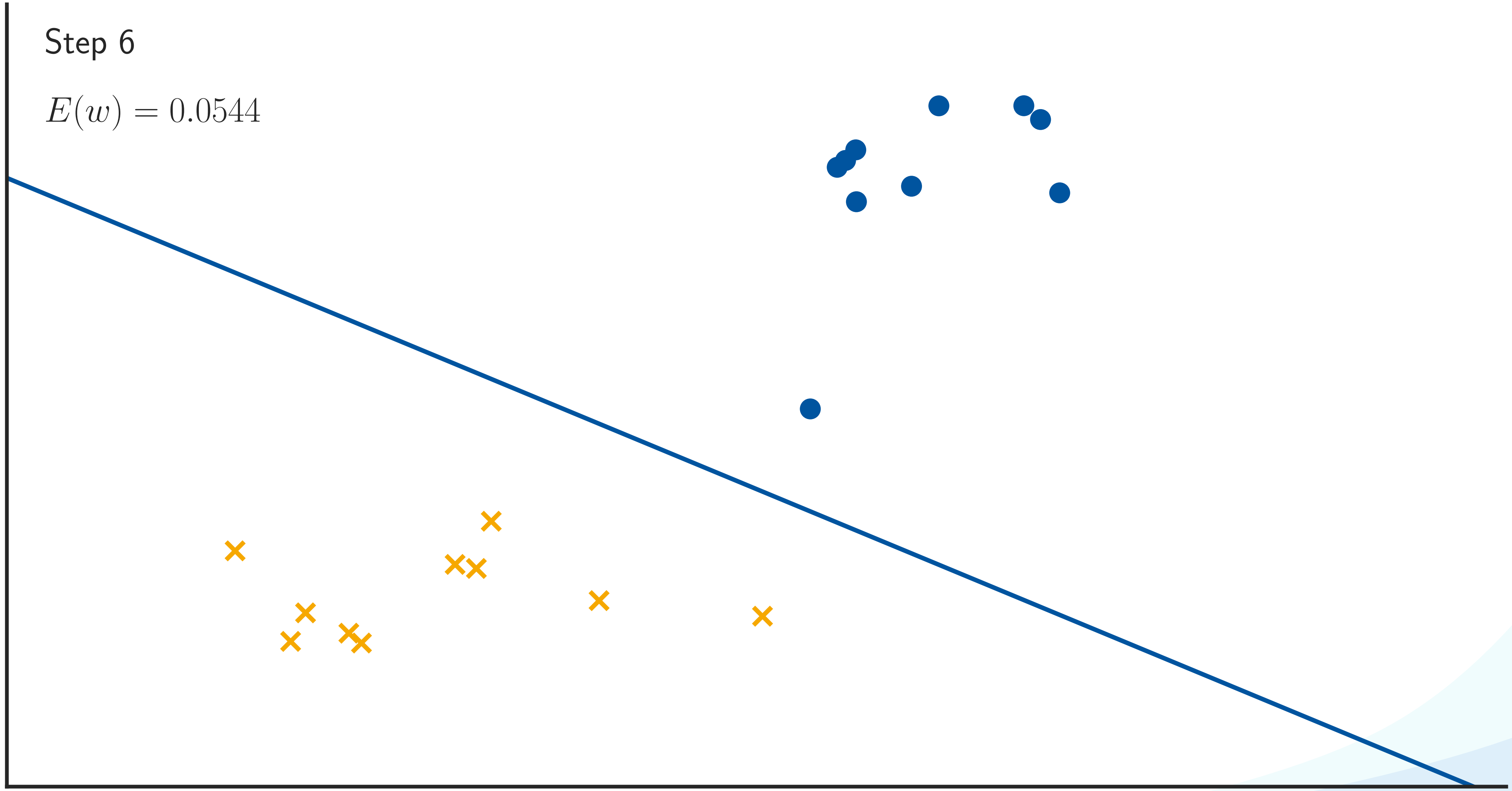
Example: Logistic Regression with IRLS

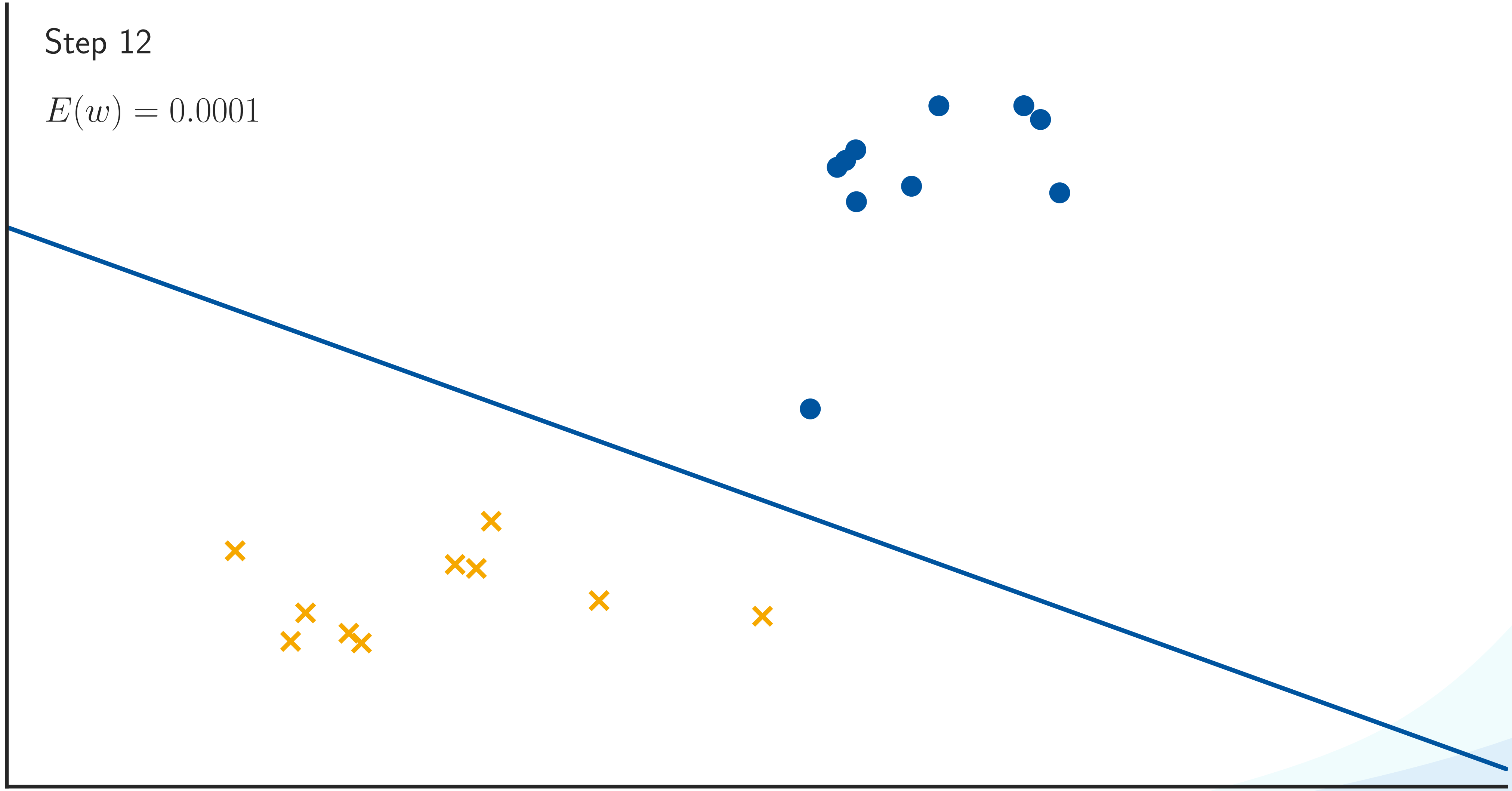




Zero training error with a single step!


Step 2
$E(w) = 1.3575$


Step 6
$E(w) = 0.0544$


Step 12
$E(w) = 0.0001$

Discussion: Second-Order Optimization

Advantages

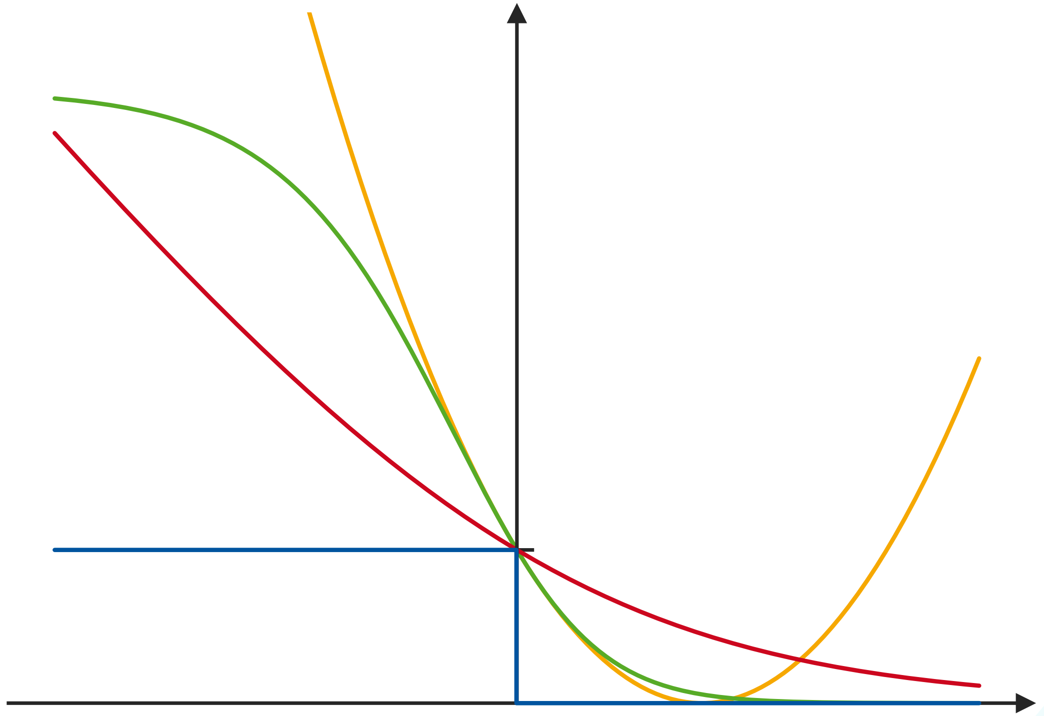
- Faster convergence than first-order methods

Limitations

- Second-order approach, relies on computing second derivatives.
- Computing (and inverting) the Hessian matrix is expensive for problems with many parameters.

Logistic Regression

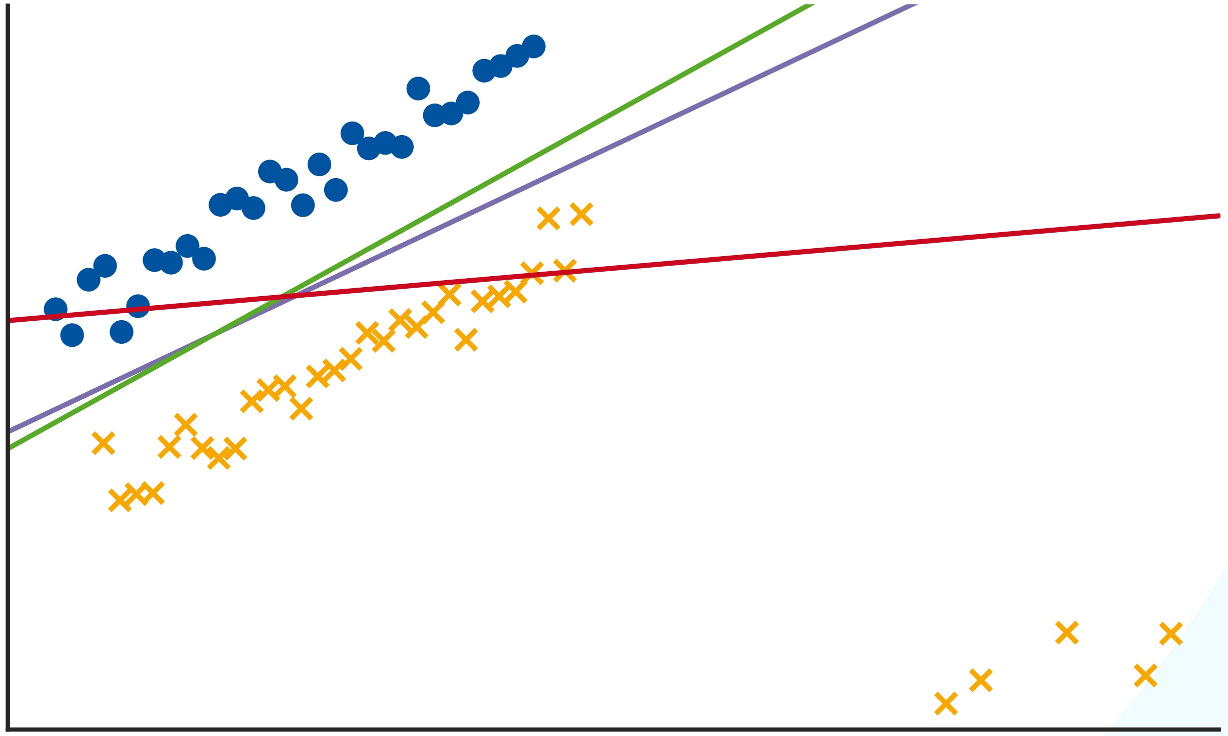
1. Logistic Regression Formulation
2. Motivation and Background
3. Iterative Estimation
4. First-Order Gradient Descent
5. Second-Order Gradient Descent
6. **Error Function Analysis**



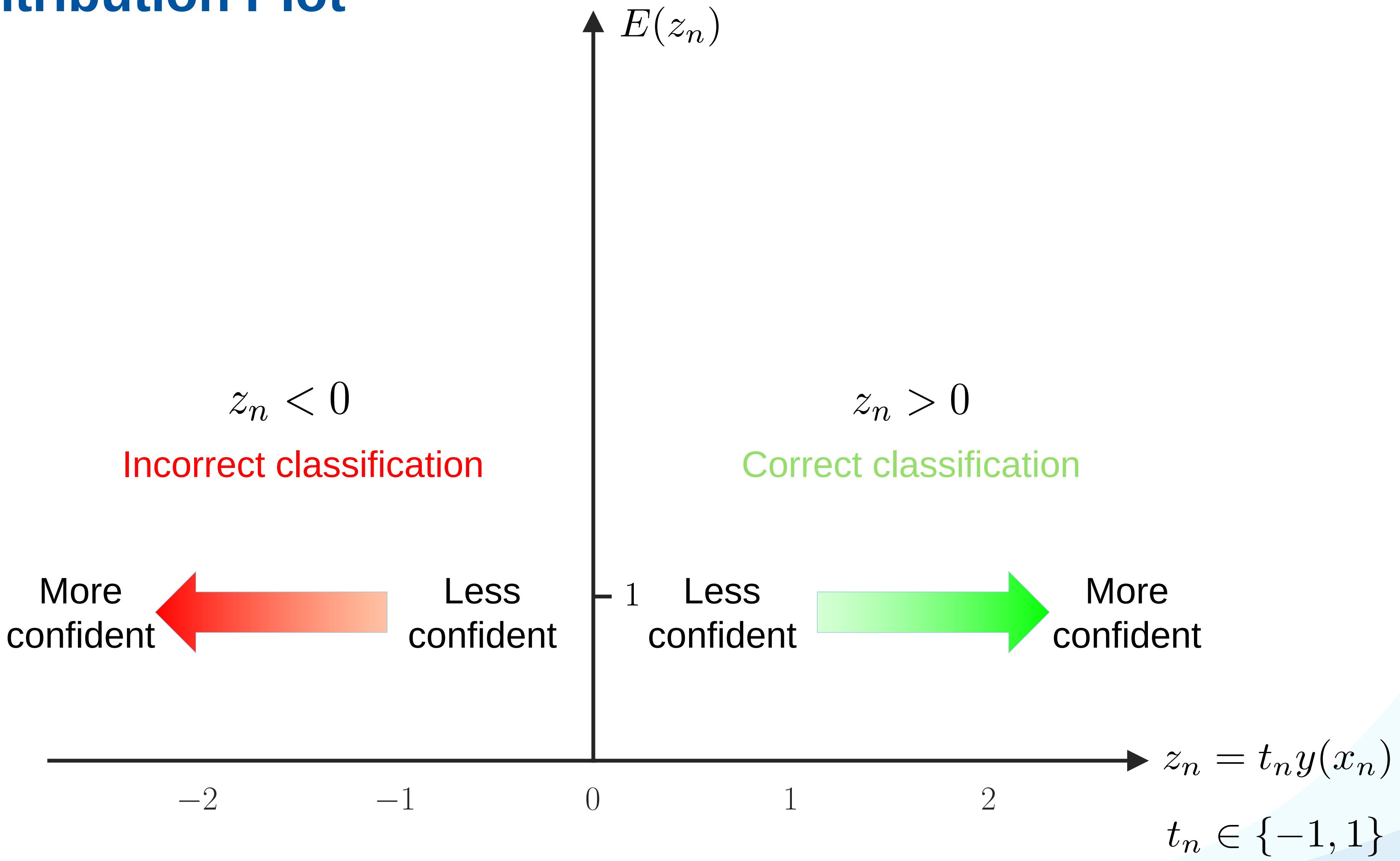
Error Function Analysis

- 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.
  - We have seen that logistic regression behaves more robustly.

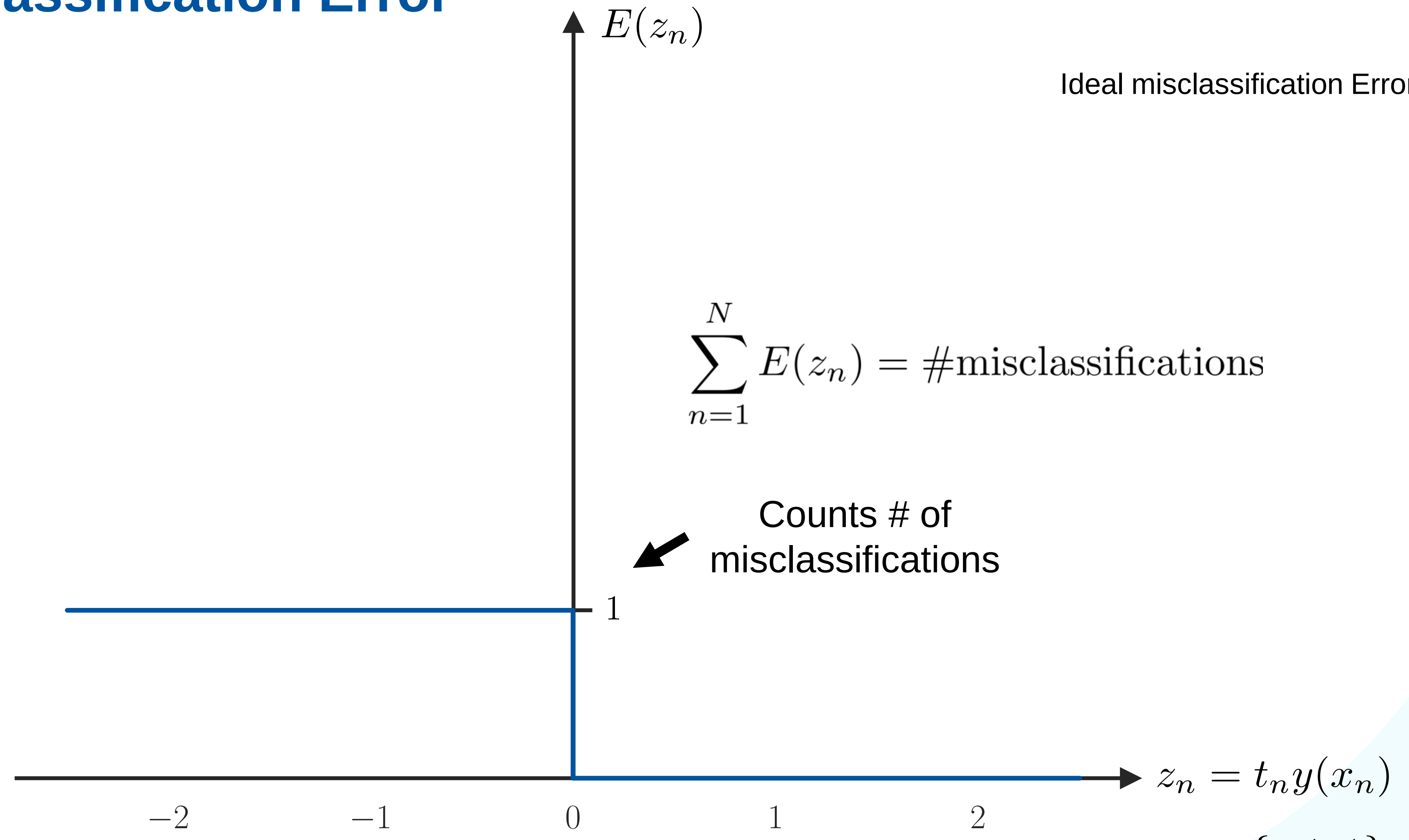
- *Let's analyze the cross-entropy error in more detail…*



Error Contribution Plot



Ideal Misclassification Error



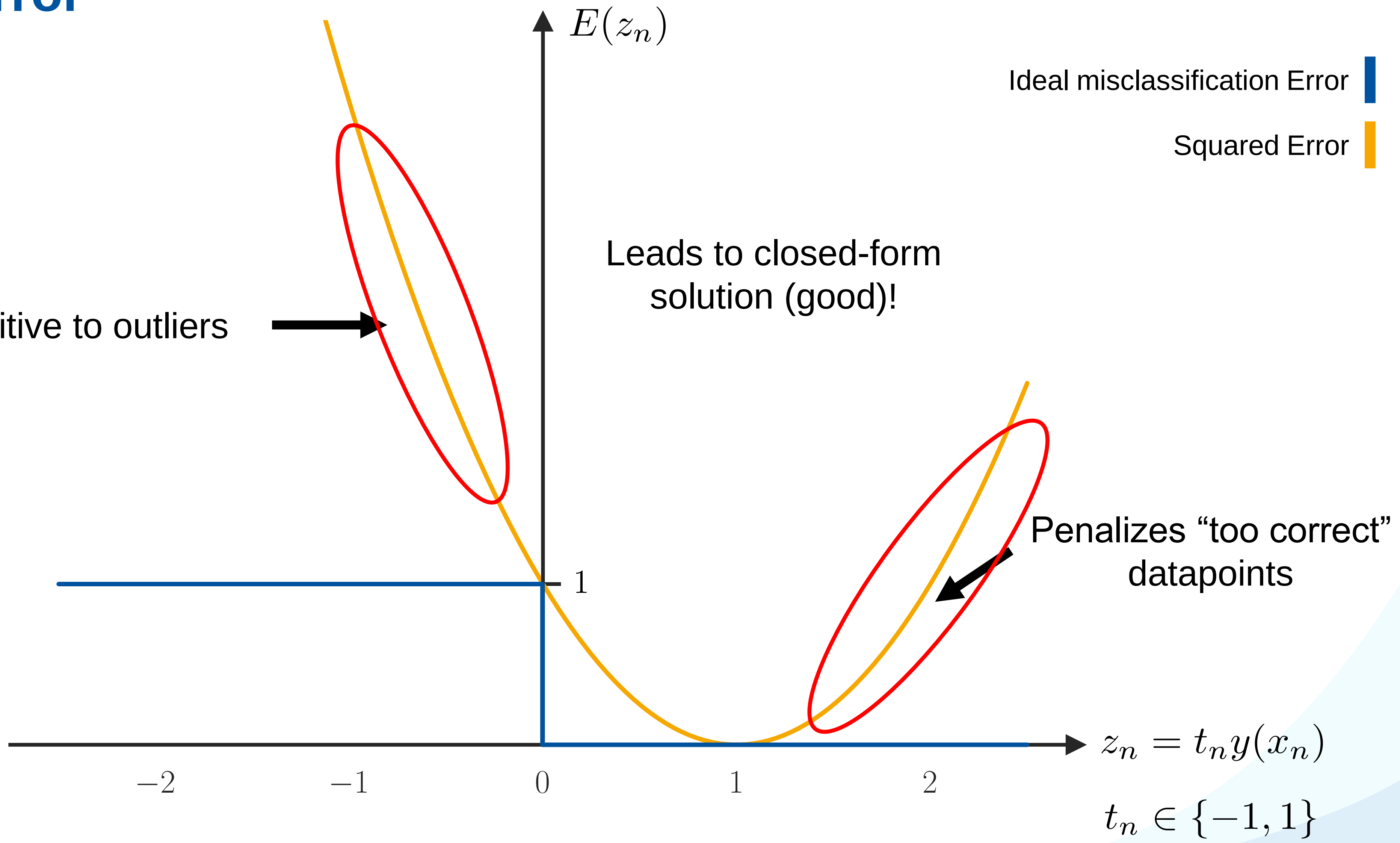
Ideal misclassification Error

$$\sum_{n=1}^{N} E(z_n) = \#\text{misclassifications}$$

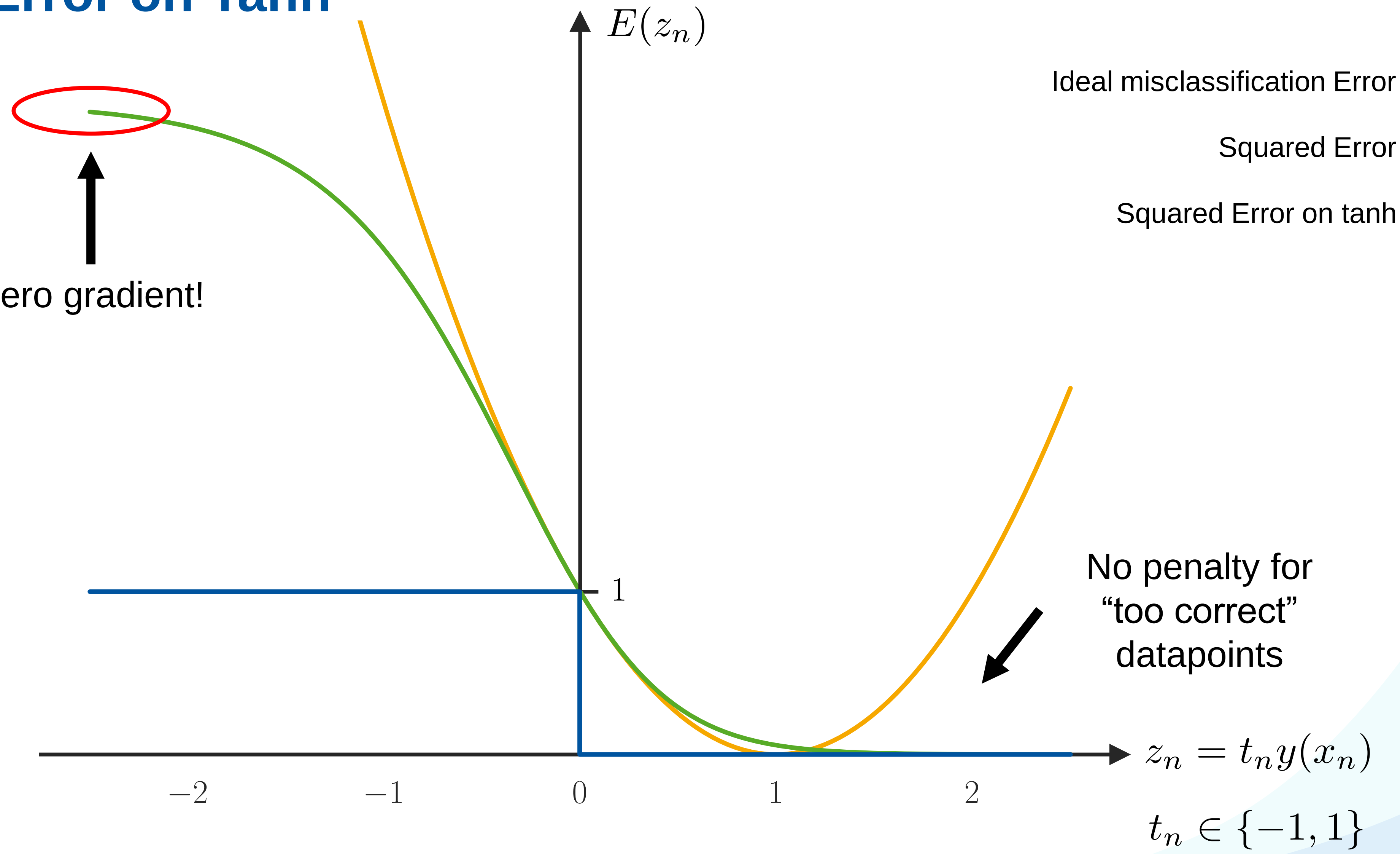
Counts # of misclassifications

$t_n \in \{-1, 1\}$

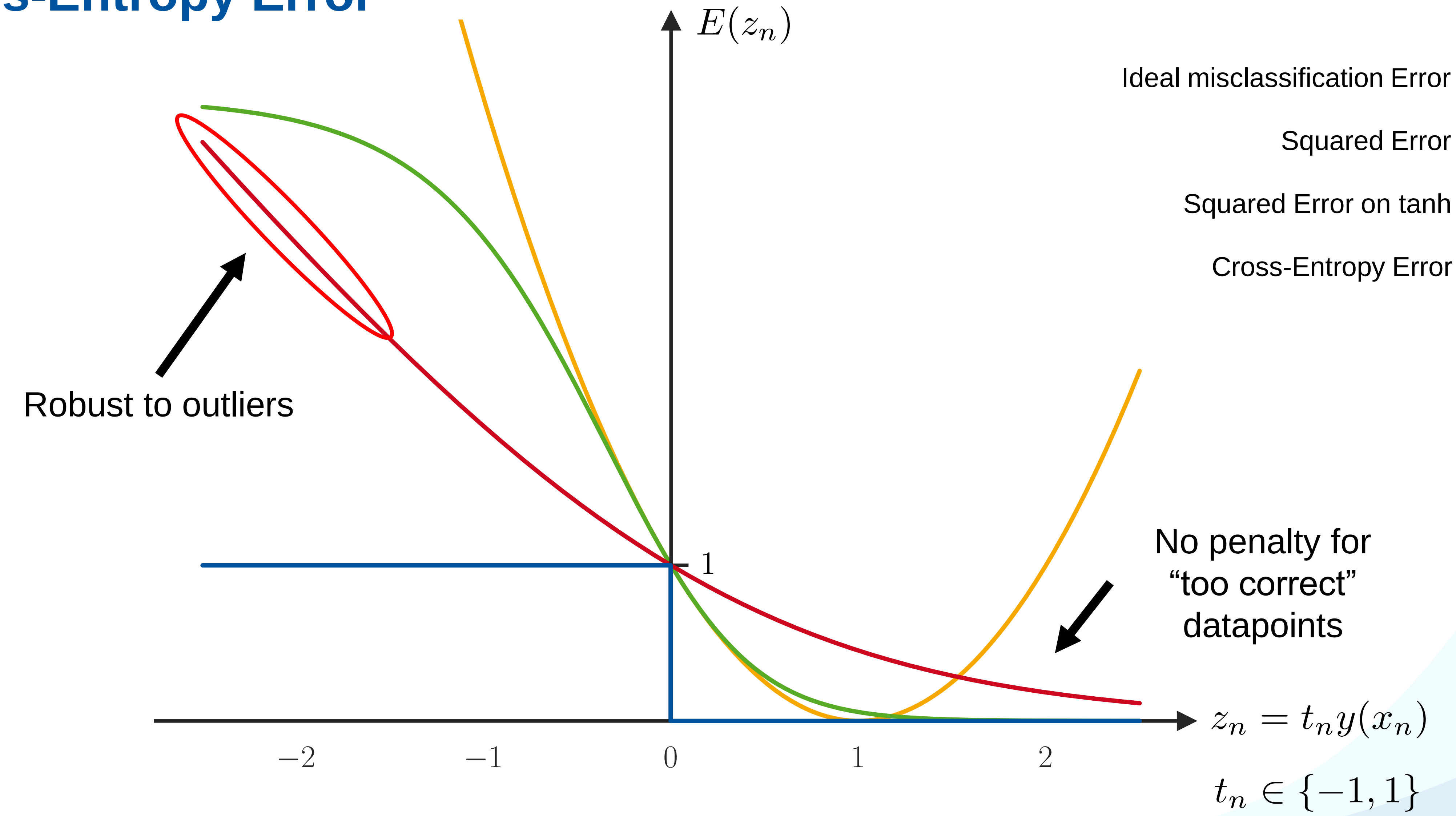
Squared Error



Squared Error on Tanh



Cross-Entropy Error



Discussion: Cross-Entropy Error

Advantages

- Minimizer of this error corresponds to class posteriors
- Convex function, unique minimum exists
- Robust to outliers

Limitations

- No closed-form solution, requires iterative estimation

References and Further Reading

- More information about Logistic Regression is available in Chapter 4.3 of Bishop's book.

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