Deep learning with Neural Networks

Logistic Regression, Stochastic Optimization & Regularization

Pablo Martínez Olmos, pamartin@ing.uc3m.es

A binary classification problem

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{N} \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{0, 1\}$

- Goal: to propose an **hypothesis function** to estimate the **most likely class** of a new data point

$$y^{(*)} \approx \arg \max_{y \in \{0,1\}} h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(*)}, y)$$

Parameters to be optimized using the training set

Logistic Regression: the model

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

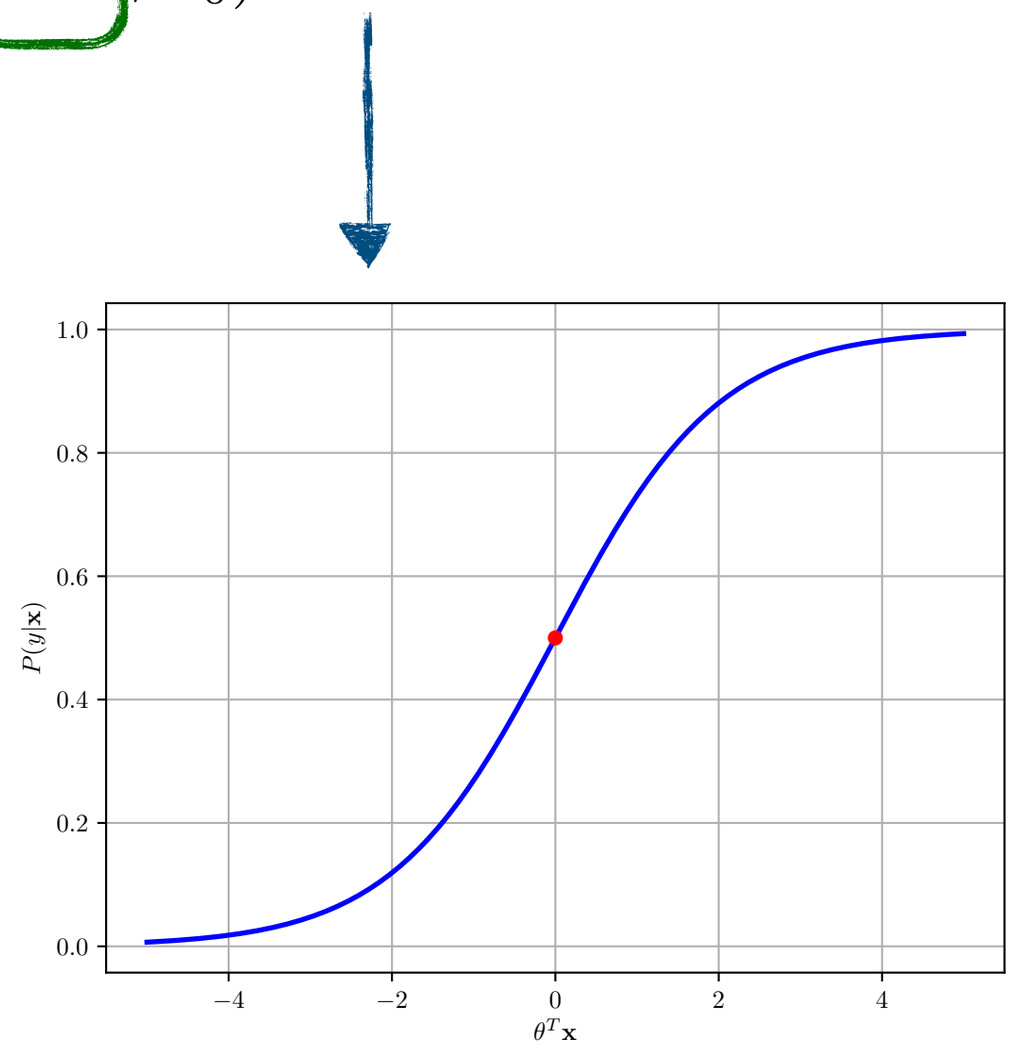
Generalized linear model

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Generalized linear model



Binary Logistic Regression

- Discriminative classifier:

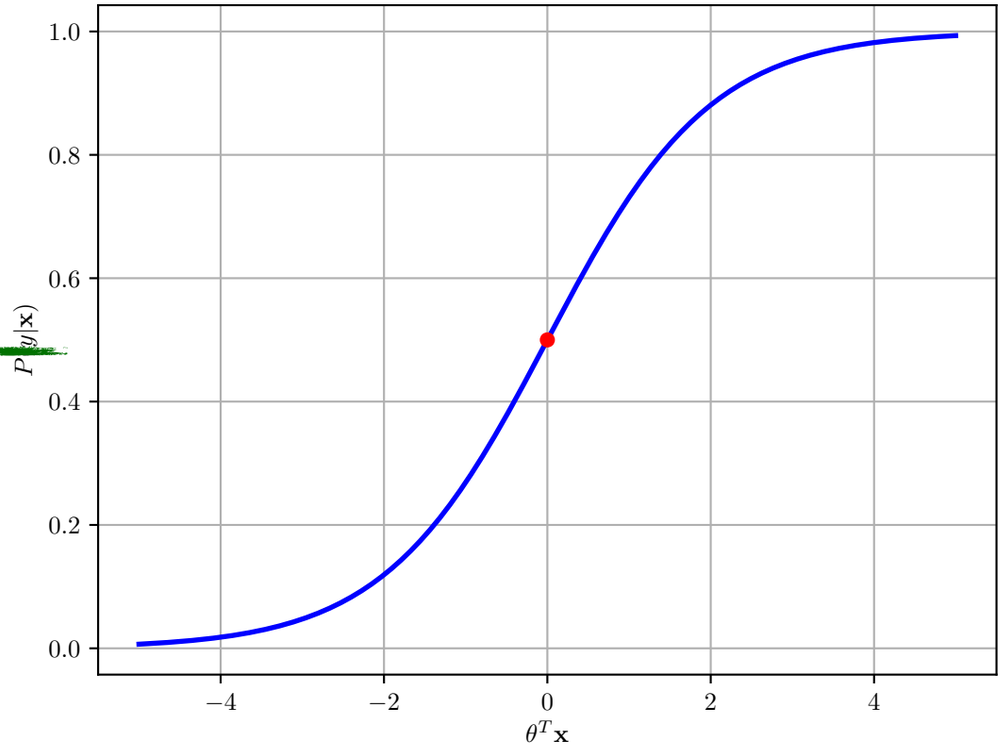
$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Generalized linear model

- Decision boundary:

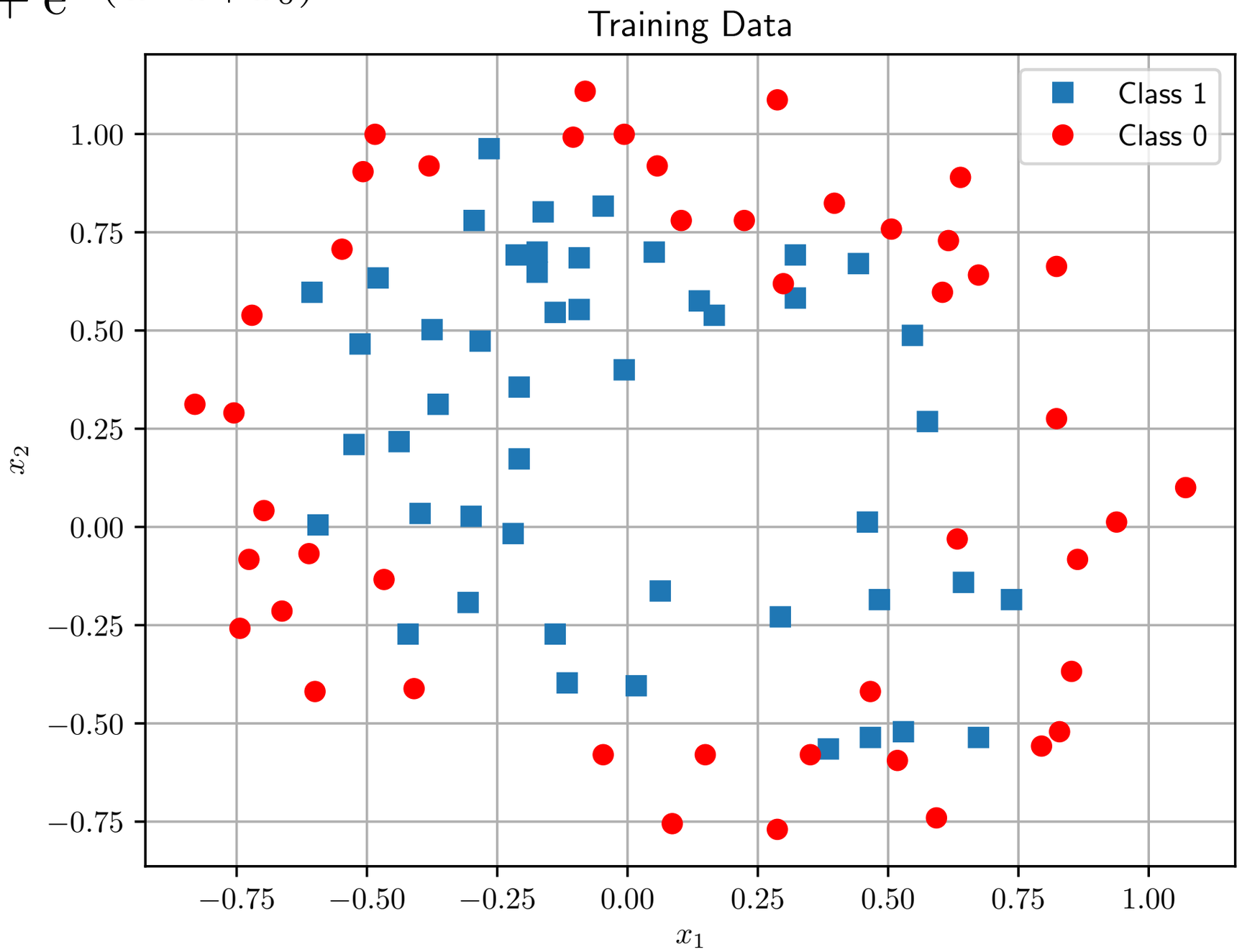
$$\{\boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{w}^T\boldsymbol{x} + w_0 = \mathbf{0}\}$$

Hyperplane! Linear classifier



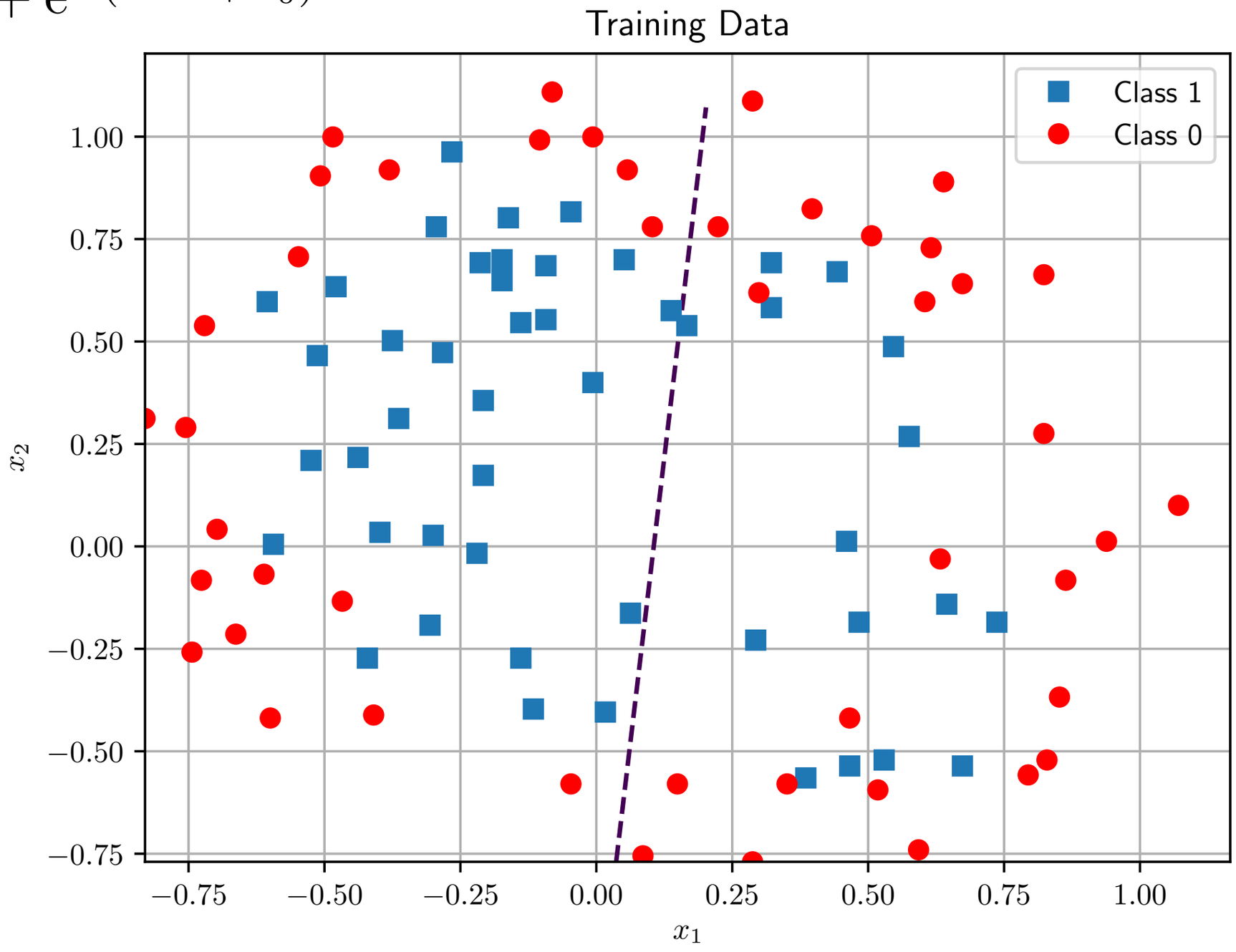
Binary Logistic Regression

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$



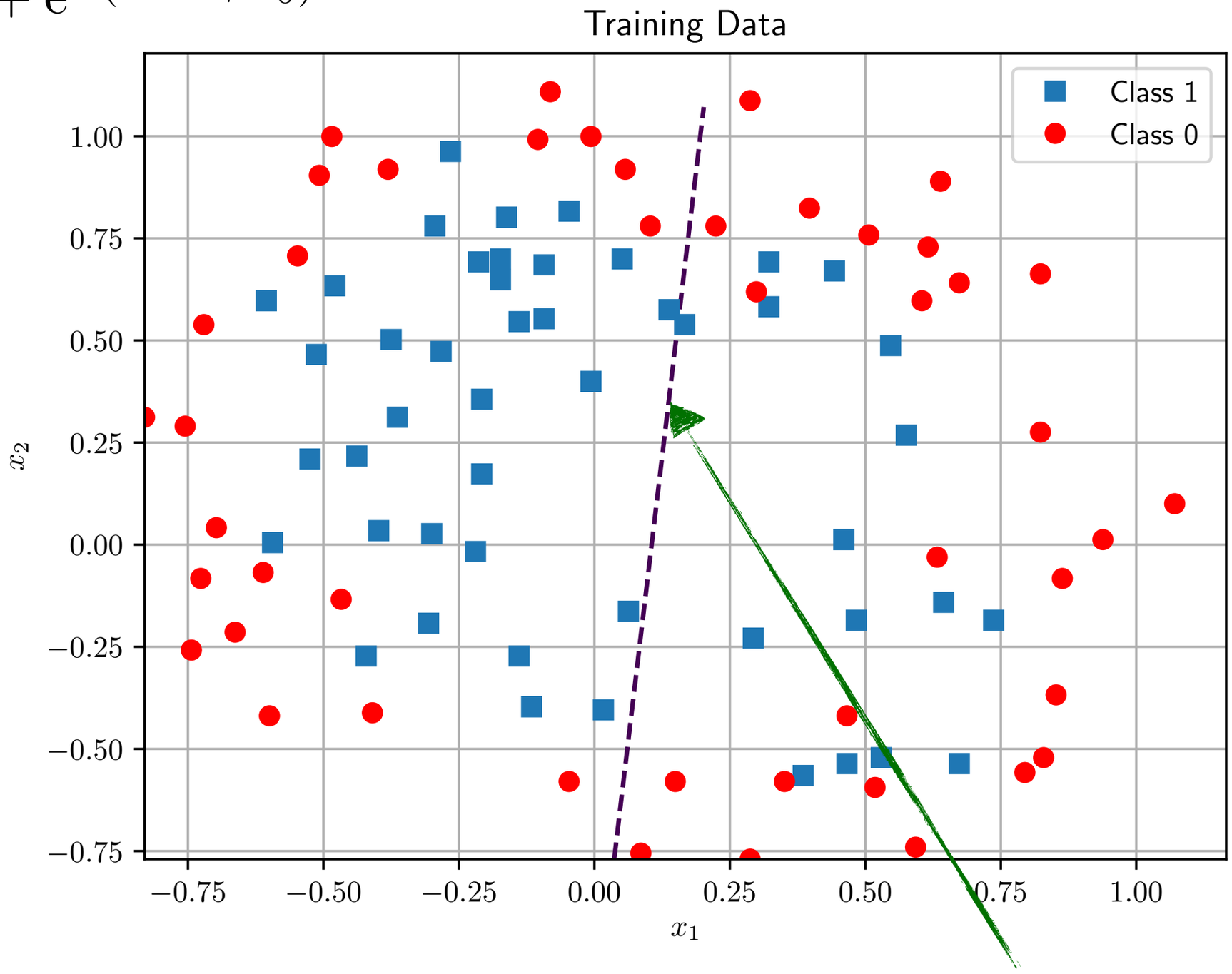
Binary Logistic Regression

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$



Binary Logistic Regression

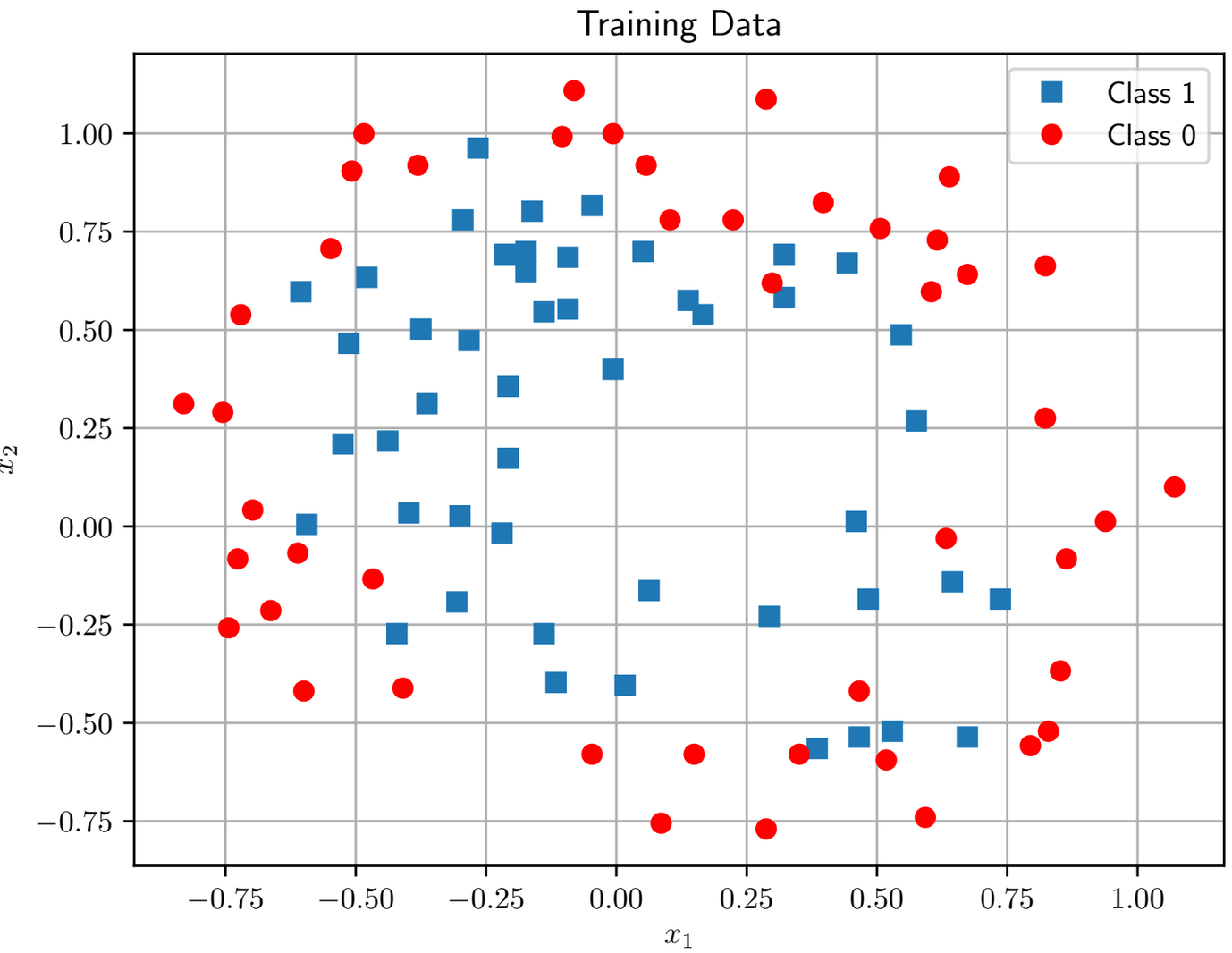
$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$



LR decision boundary with linear features

Binary Logistic Regression. Adding new features

- The problem is easy to visualize
- An ellipsoid clearly fits better as decision boundary than a straight line



- Expand input space with extra degrees of freedom
- Quadratic features

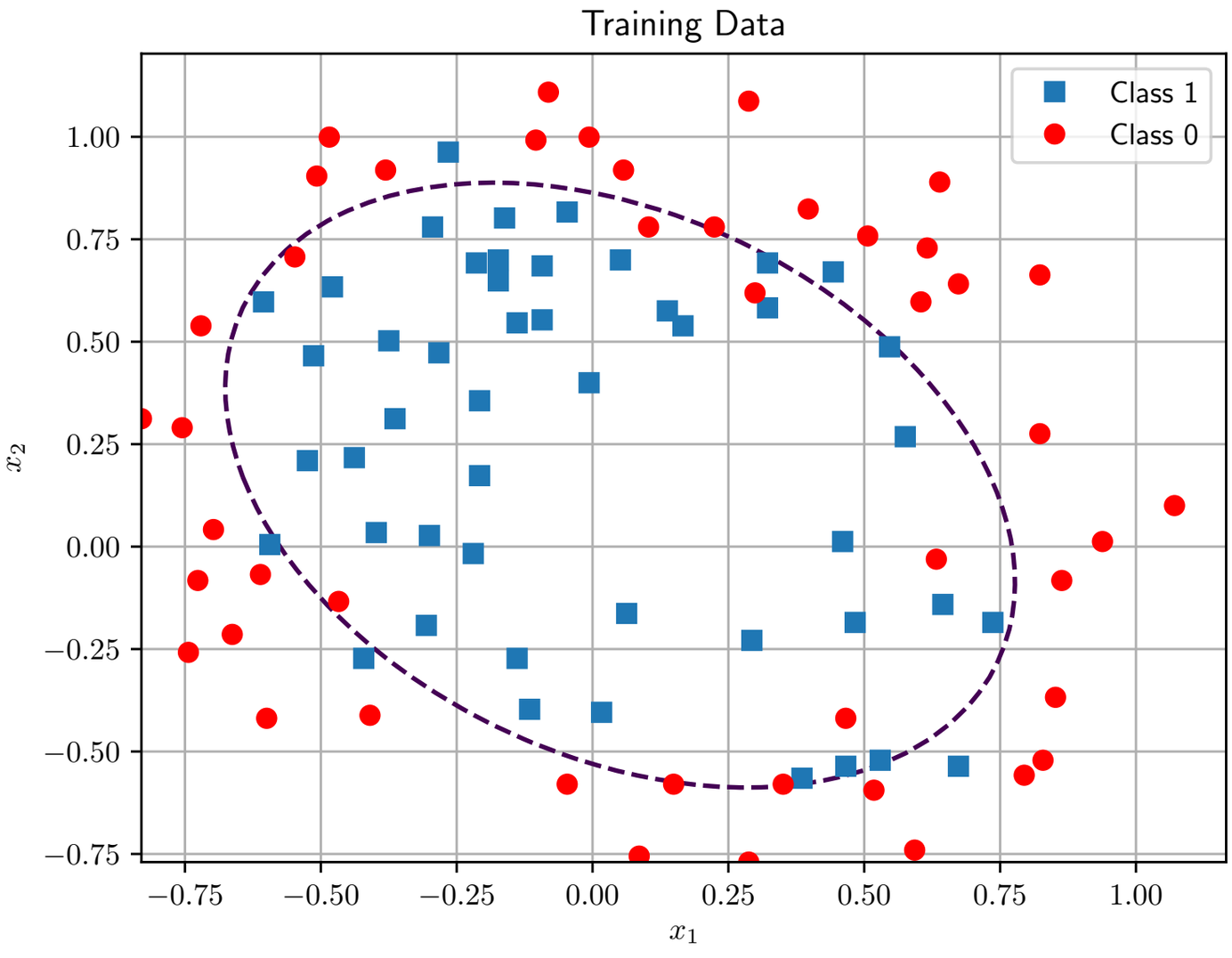
$$\boldsymbol{x} = [x_1,\ x_2]$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

$$P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + w_0)$$

Binary Logistic Regression. Adding new features

- The problem is easy to visualize
- An ellipsoid clearly fits better as decision boundary than a straight line



- Expand input space with extra degrees of freedom
- Quadratic features

$$\boldsymbol{x} = [x_1,\ x_2]$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

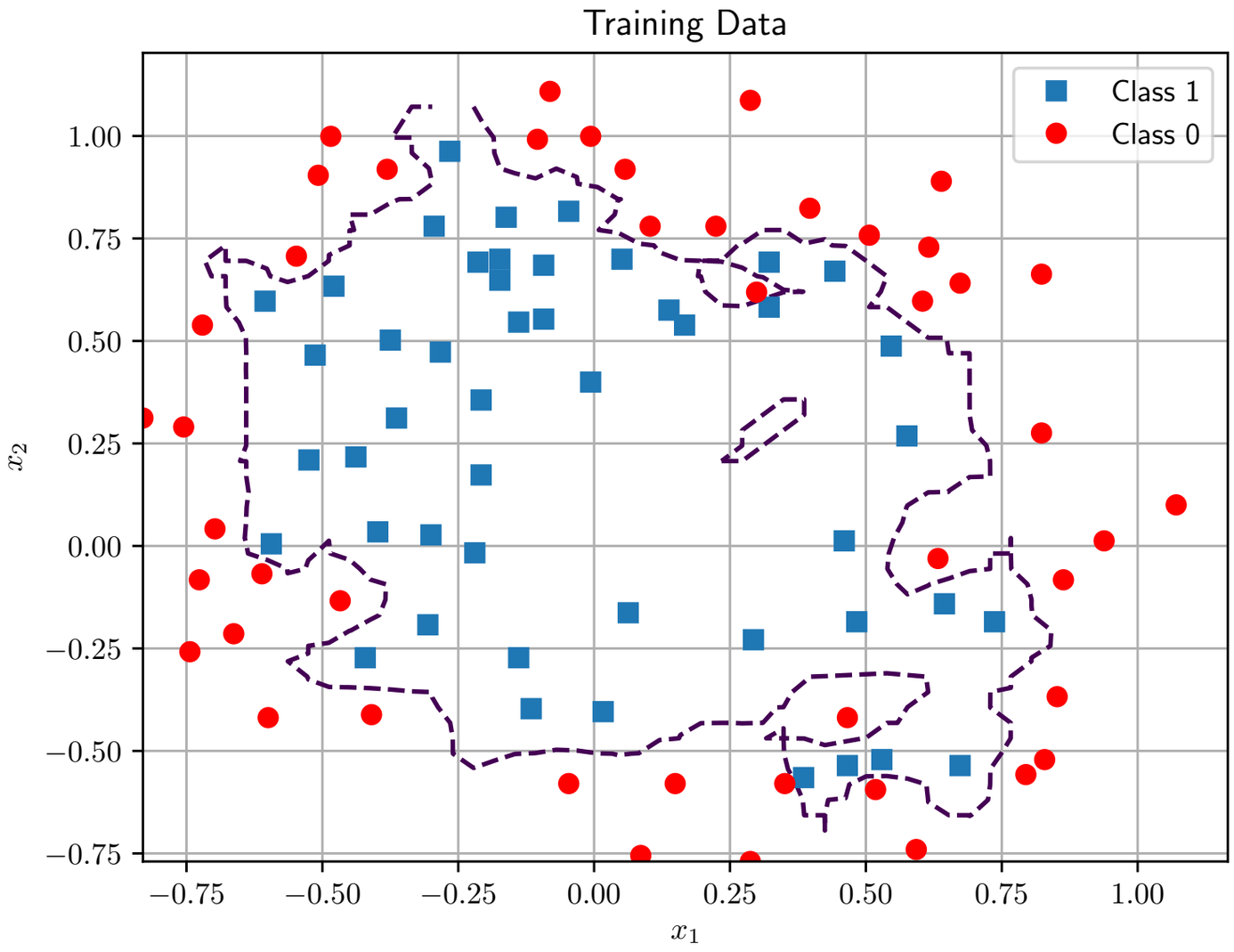
$$P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + w_0)$$

Binary Logistic Regression. Adding more and more features

- By adding new features we allow the model to fit (even overfit) to the training data
- We need a validation set to evaluate the model's exposure to overfitting

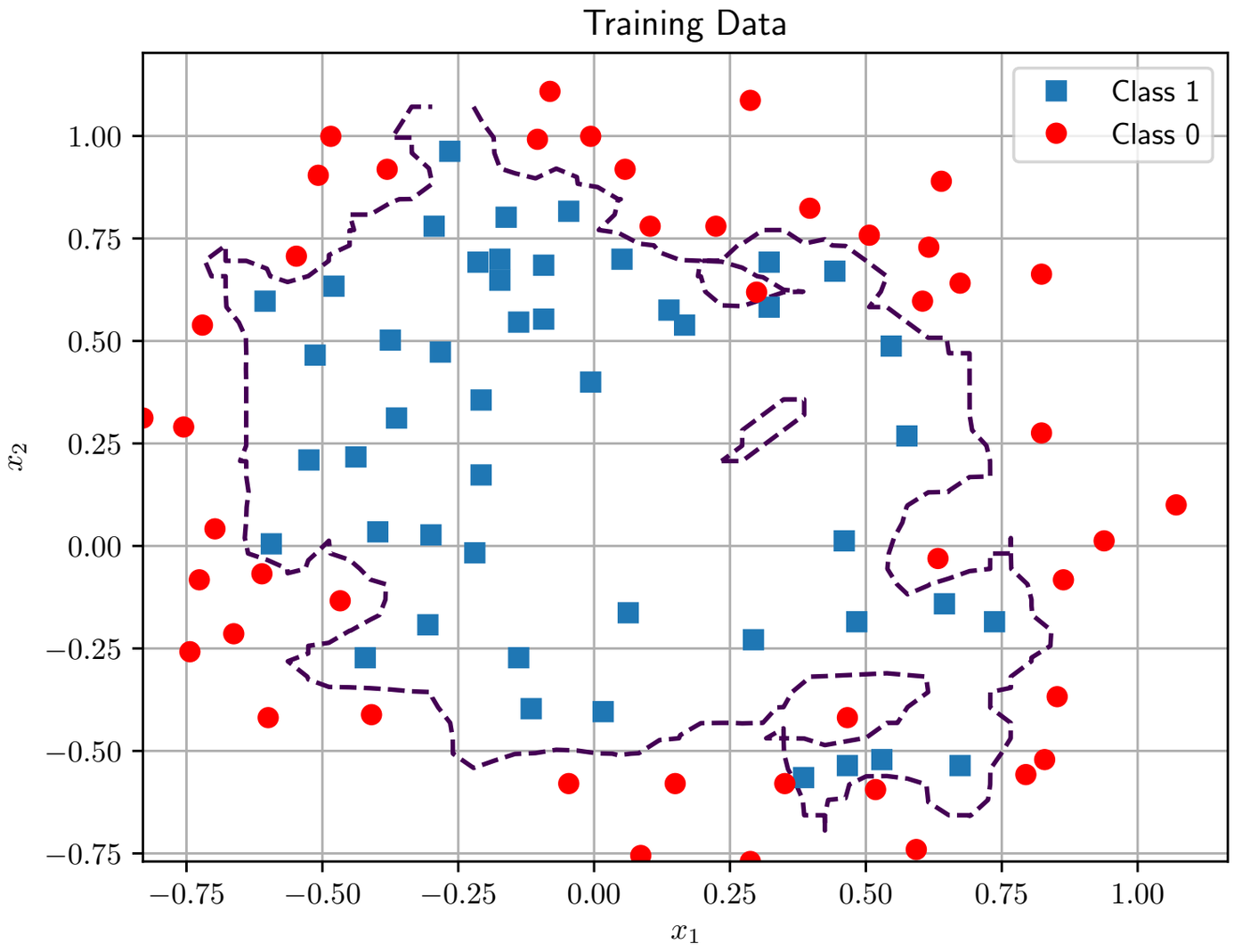
Binary Logistic Regression. Adding more and more features

- By adding new features we allow the model to fit (even overfit) to the training data
- We need a validation set to evaluate the model's exposure to overfitting



Binary Logistic Regression. Adding more and more features

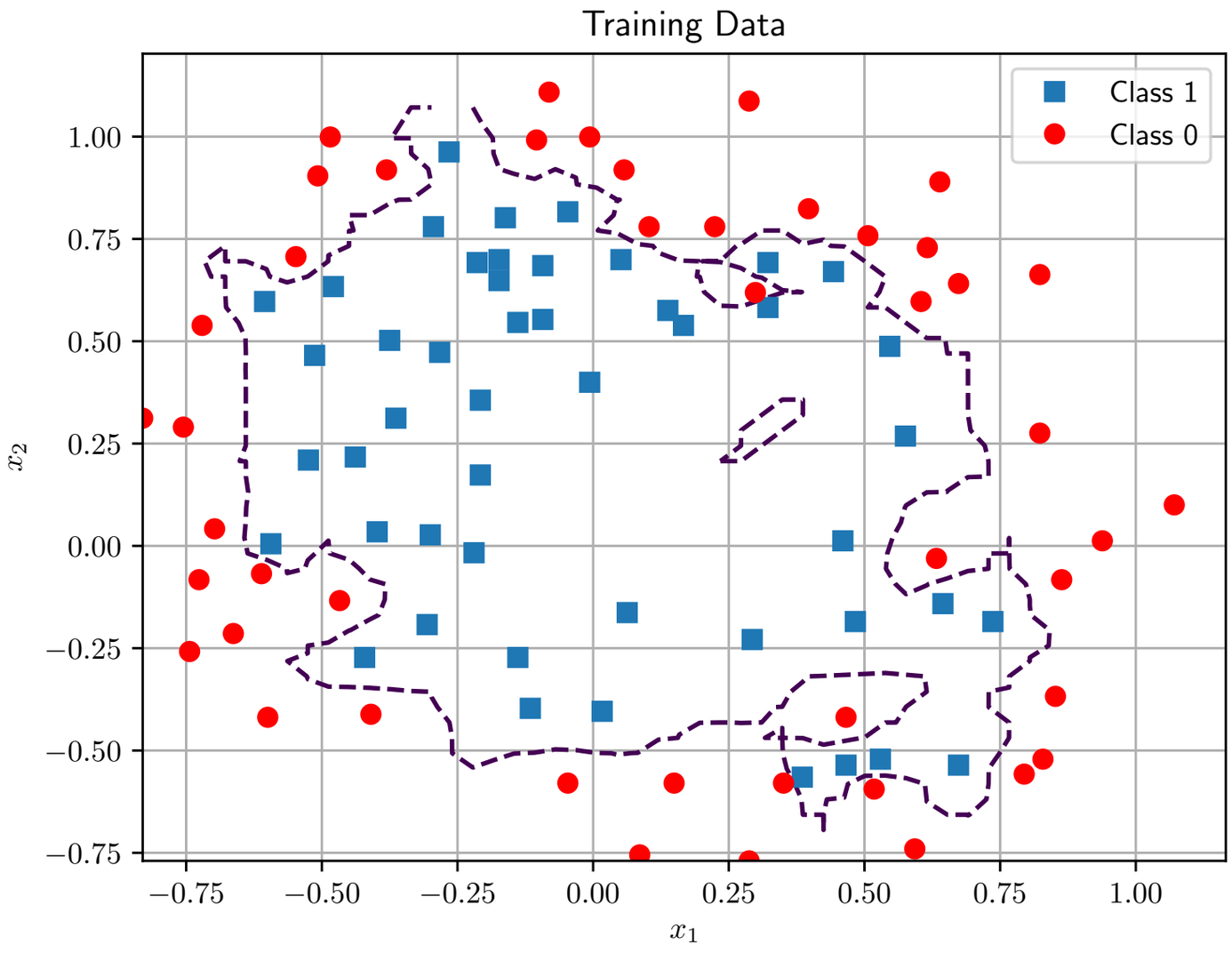
- By adding new features we allow the model to fit (even overfit) to the training data
- We need a validation set to evaluate the model's exposure to overfitting



Provided you find the right set of features!

Binary Logistic Regression. Adding more and more features

- By adding new features we allow the model to fit (even overfit) to the training data
- We need a validation set to evaluate the model's exposure to overfitting

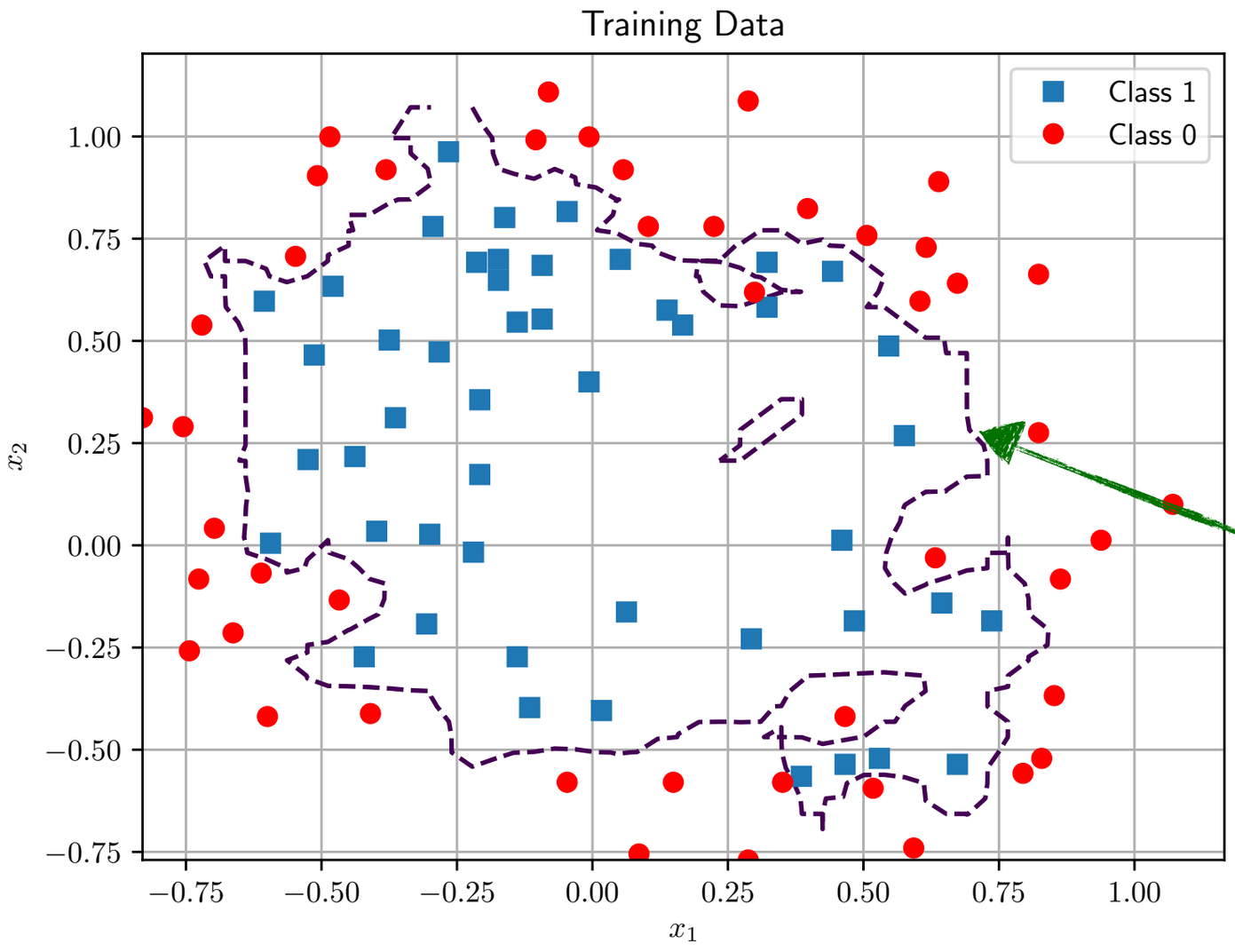


Provided you find the right set of features!

VERY hard problem in general. Requires domain-specific knowledge.

Binary Logistic Regression. Adding more and more features

- By adding new features we allow the model to fit (even overfit) to the training data
- We need a validation set to evaluate the model's exposure to overfitting



Provided you find the right set of features!

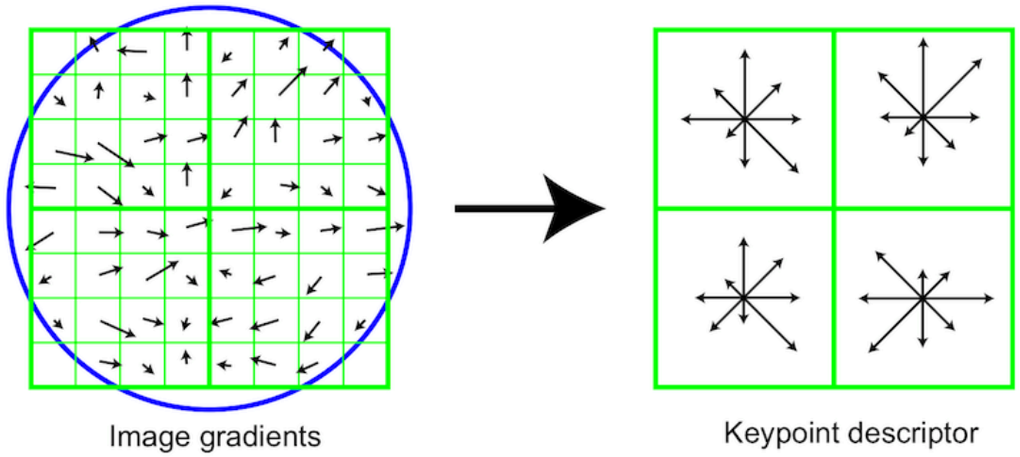
VERY hard problem in general. Requires domain-specific knowledge.

What is the equation of such a weird boundary?

Ways to go …

Ways to go …

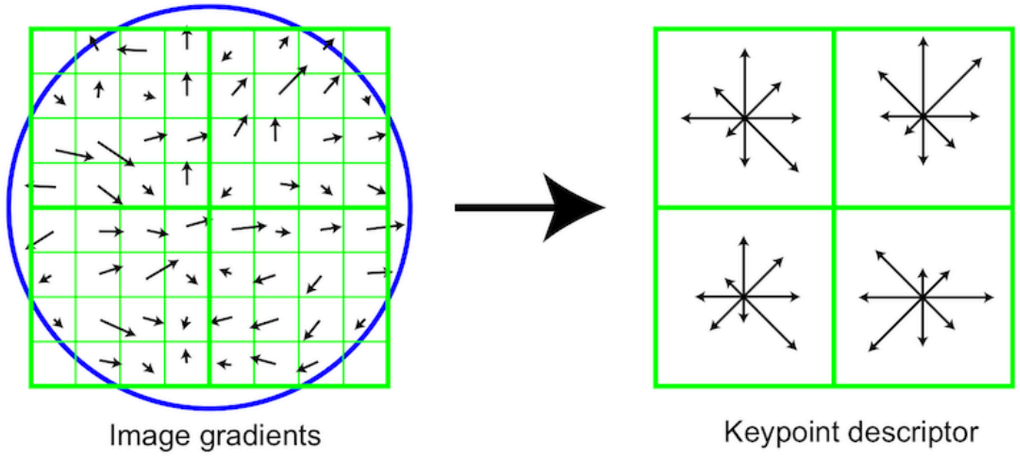
- Manual **feature design**: complex in high-dimensional settings, domain-specific knowledge



E.g. SIFT features in an image
(Image taken from this excellent blog)

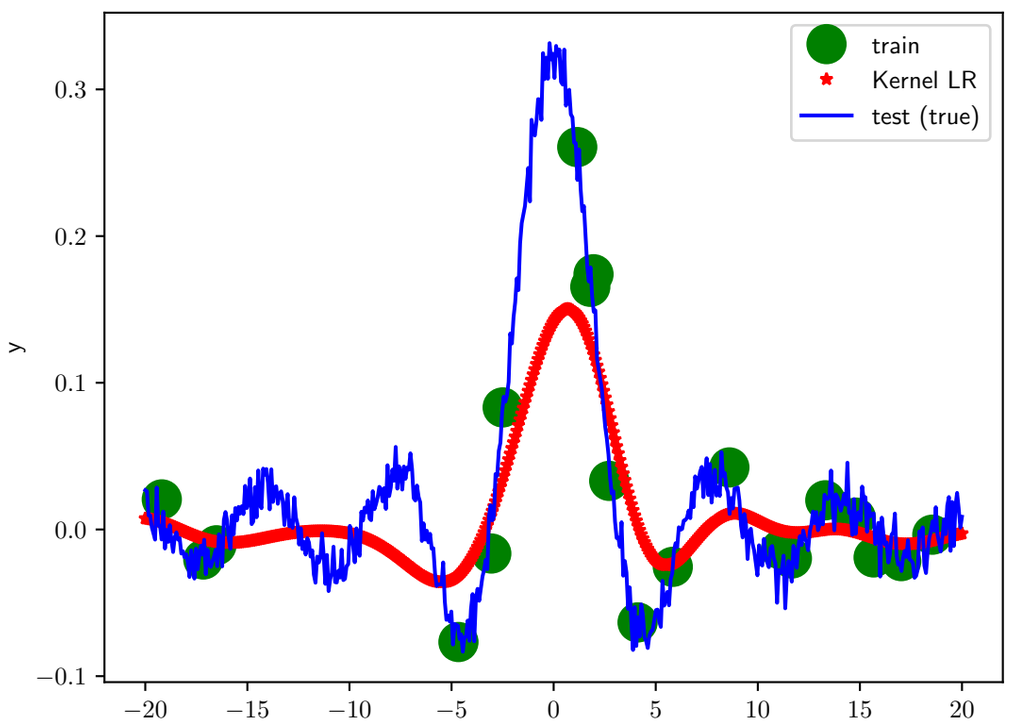
Ways to go …

- Manual **feature design**: complex in high-dimensional settings, domain-specific knowledge



E.g. SIFT features in an image
(Image taken from this excellent blog)

- **Kernel methods**: you do not think on feature space, but on the data space (~ interpolate with your neighbours). Computationally very complex for large databases.

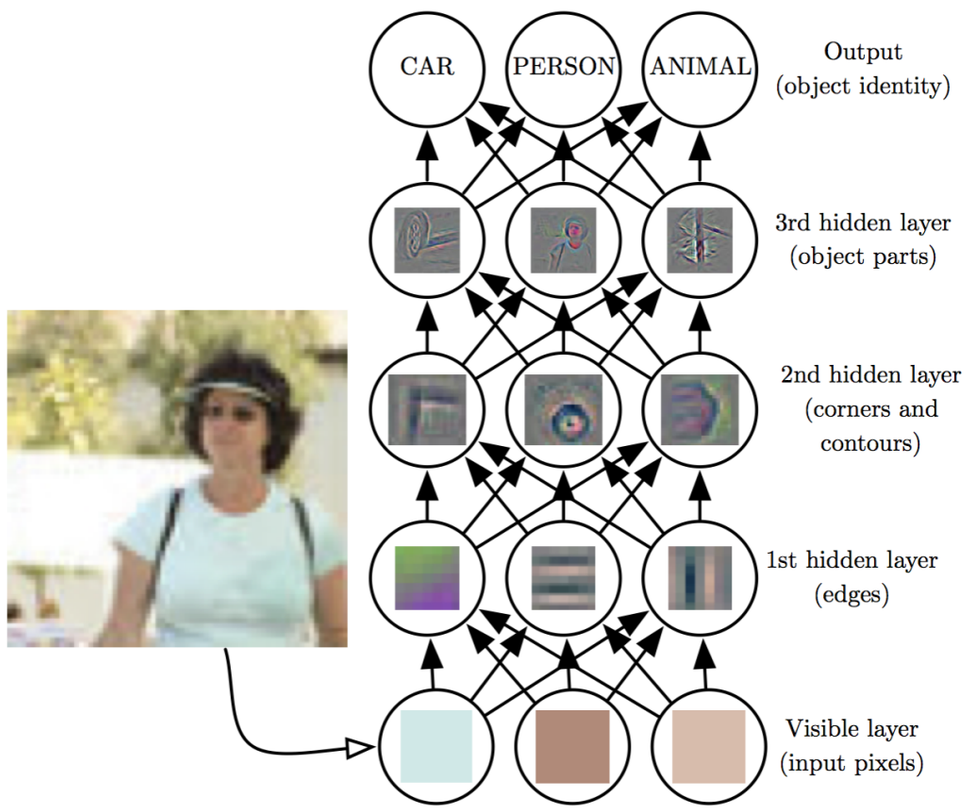


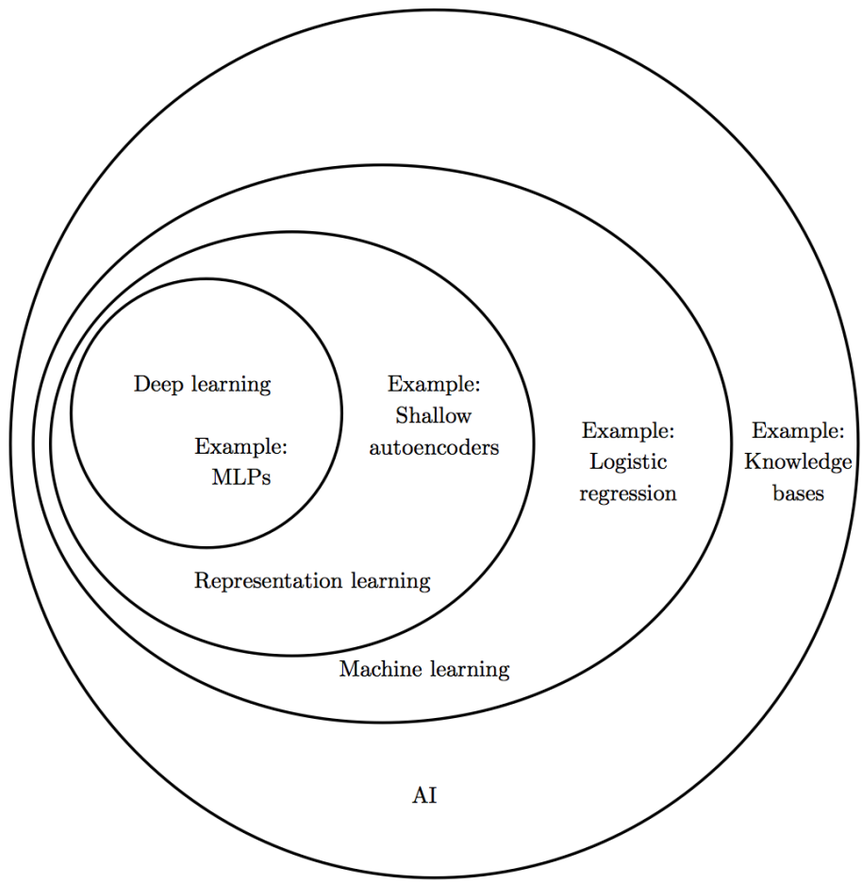
Example of kernel regression with Gaussian kernel

Ways to go ...

- **Representation Learning**: use raw-data and a highly flexible family of transformation functions.

Neural networks belong to this class of methods!!





Source: Deep Learning, Good Fellow et al. 2017

Logistic Regression: training

Binary Logistic Regression. Maximum likelihood training

- Hypothesis function: $P(y=1|\boldsymbol{x}) = \dfrac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$

- Training database:

- (Log)-Likelihood function:

Binary Logistic Regression. Maximum likelihood training

- Hypothesis function: $P(y=1|\boldsymbol{x}) = \dfrac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^N \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{0,1\}$

- (Log)-Likelihood function:

Binary Logistic Regression. Maximum likelihood training

- Hypothesis function: $P(y=1|\boldsymbol{x}) = \dfrac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^N \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{0,1\}$

- (Log)-Likelihood function:

$$P(y|\boldsymbol{x}) = P(y=1|\boldsymbol{x})^{\mathbb{1}[y=1|\boldsymbol{x}]} P(y=0|\boldsymbol{x})^{\mathbb{1}[y=0|\boldsymbol{x}]}$$

Binary Logistic Regression. Maximum likelihood training

- Hypothesis function: $P(y=1|\boldsymbol{x}) = \dfrac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^N \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{0,1\}$

- (Log)-Likelihood function:

$$P(y|\boldsymbol{x}) = P(y=1|\boldsymbol{x})^{\mathbb{1}[y=1|\boldsymbol{x}]} P(y=0|\boldsymbol{x})^{\mathbb{1}[y=0|\boldsymbol{x}]}$$

$$P(\boldsymbol{y}|\boldsymbol{X}) = \prod_{i=1}^{N} P(y^{(i)}=1|\boldsymbol{x}^{(i)})$$

Binary Logistic Regression. Maximum likelihood training

- Hypothesis function: $P(y=1|\boldsymbol{x}) = \dfrac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^N \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{0,1\}$

- (Log)-Likelihood function:

$$P(y|\boldsymbol{x}) = P(y=1|\boldsymbol{x})^{\mathbb{1}[y=1|\boldsymbol{x}]} P(y=0|\boldsymbol{x})^{\mathbb{1}[y=0|\boldsymbol{x}]}$$

$$P(\boldsymbol{y}|\boldsymbol{X}) = \prod_{i=1}^{N} P(y^{(i)}=1|\boldsymbol{x}^{(i)})$$

$$\log P(\boldsymbol{y}|\boldsymbol{X}) = \sum_{i=1}^{N} \log P(y^{(i)}|\boldsymbol{x}^{(i)})$$

$$= \sum_{i=1}^{N} \mathbb{1}[y^{(i)}=1] \log P(y^{(i)}=1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)}=0] \log P(y^{(i)}=0|\boldsymbol{x}^{(i)})$$

Binary Logistic Regression. Binary Cross Entropy Function

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Binary Cross Entropy Function

$$\begin{aligned}\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\ &= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)}=1]\log P(y^{(i)}=1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)}=0]\log P(y^{(i)}=0|\boldsymbol{x}^{(i)})\end{aligned}$$

Binary Logistic Regression. Binary Cross Entropy Function

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Binary Cross Entropy Function

$$\begin{aligned}\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\ &= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)}=1]\log P(y^{(i)}=1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)}=0]\log P(y^{(i)}=0|\boldsymbol{x}^{(i)})\end{aligned}$$

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L}$$

Binary Logistic Regression. Binary Cross Entropy Function

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Binary Cross Entropy Function

$$\begin{aligned}\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\ &= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)}=1]\log P(y^{(i)}=1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)}=0]\log P(y^{(i)}=0|\boldsymbol{x}^{(i)})\end{aligned}$$

$\mathcal{L}$ **is convex!!**

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L}$$

Binary Logistic Regression. Binary Cross Entropy Function

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

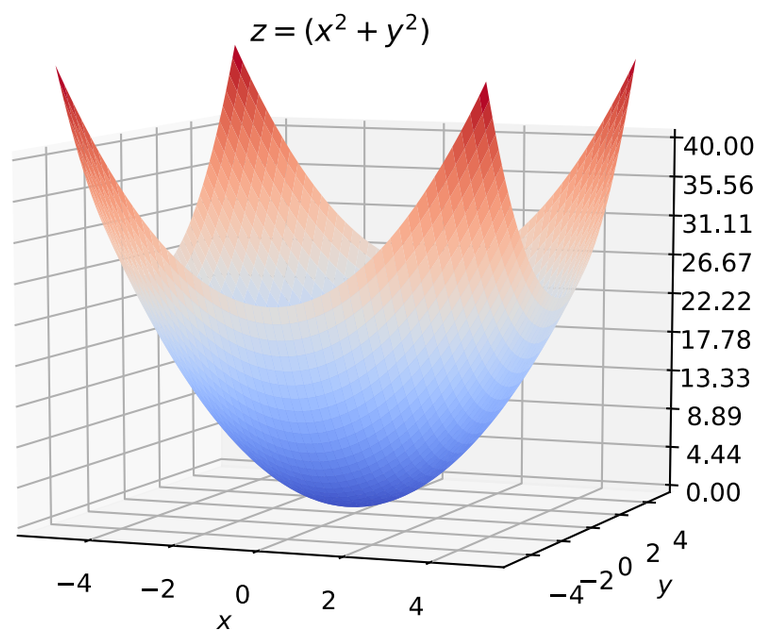
Binary Cross Entropy Function

$$\begin{aligned}\mathcal{L} &= -\log P(\boldsymbol{y}|\boldsymbol{X}) \\ &= -\sum_{i=1}^{N} \mathbb{1}[y^{(i)}=1]\log P(y^{(i)}=1|\boldsymbol{x}^{(i)}) + \mathbb{1}[y^{(i)}=0]\log P(y^{(i)}=0|\boldsymbol{x}^{(i)})\end{aligned}$$

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L}$$

$\mathcal{L}$ is convex!!



Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L}$$

Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

By using that $\dfrac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$ it can be shown that

$$\nabla_{\boldsymbol{w}, w_0} \mathcal{L} = \sum_{i=1}^{N} \left(\sigma \left(\begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right) \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

By using that $\dfrac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$ it can be shown that

$$\nabla_{\boldsymbol{w}, w_0} \mathcal{L} = \sum_{i=1}^{N} \underbrace{\left(\sigma\left(\begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right)}_{\text{Prediction error (real scalar)}} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

By using that $\dfrac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$ it can be shown that

$$\nabla_{\boldsymbol{w}, w_0} \mathcal{L} = \sum_{i=1}^{N} \underbrace{\left(\sigma\left(\begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right)}_{\text{Prediction error (real scalar)}} \underbrace{\begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}}_{\substack{\text{Vector} \\ \text{(bias + data point)}}}$$

Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

By using that $\dfrac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$ it can be shown that

$$\nabla_{\boldsymbol{w}, w_0} \mathcal{L} = \sum_{i=1}^{N} \Big(\sigma\Big(\begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \Big) - y^{(i)} \Big) \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

Prediction error
(real scalar)

Vector
(bias + data point)

Gradient is a weighted-sum of training data points

Logistic Regression Training

Logistic Regression Training

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \arg\min_{\boldsymbol{w}, w_0} \mathcal{L} \quad \Longrightarrow \quad \nabla_{\boldsymbol{w}, w_0} \mathcal{L}(\hat{\boldsymbol{w}}, \hat{w}_0) = \mathbf{0}$$

No closed-form solution!

By using that $\dfrac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$ it can be shown that

$$\nabla_{\boldsymbol{w}, w_0} \mathcal{L} = \sum_{i=1}^{N} \underbrace{\left(\sigma\left(\begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right)}_{\text{Prediction error (real scalar)}} \underbrace{\begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}}_{\text{Vector (bias + data point)}}$$

Gradient is a weighted-sum of training data points

Gradient Descent (GD)

$$\boldsymbol{w}_+ \doteq \begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - \left(\alpha \nabla \mathcal{L}\right)\big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

$$\nabla_{\boldsymbol{w}_+} \mathcal{L} = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w}_+ \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right) \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

Gradient Descent (GD)

$$\boldsymbol{w}_+ \doteq \begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \Rightarrow \nabla_{\boldsymbol{w}_+}\mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - (\alpha\nabla\mathcal{L})\,|_{\boldsymbol{w}_+^{(\ell-1)}}$$

the step size

$$\nabla_{\boldsymbol{w}_+}\mathcal{L} = \sum_{i=1}^{N}\left(\sigma\left(\boldsymbol{w}_+\begin{bmatrix}1.0\\ \boldsymbol{x}^{(i)}\end{bmatrix}\right) - y^{(i)}\right)\begin{bmatrix}1.0\\ \boldsymbol{x}^{(i)}\end{bmatrix}$$

Gradient Descent (GD)

$$\boldsymbol{w}_+ \doteq \begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - (\alpha \nabla \mathcal{L}) \,|_{\boldsymbol{w}_+^{(\ell-1)}}$$

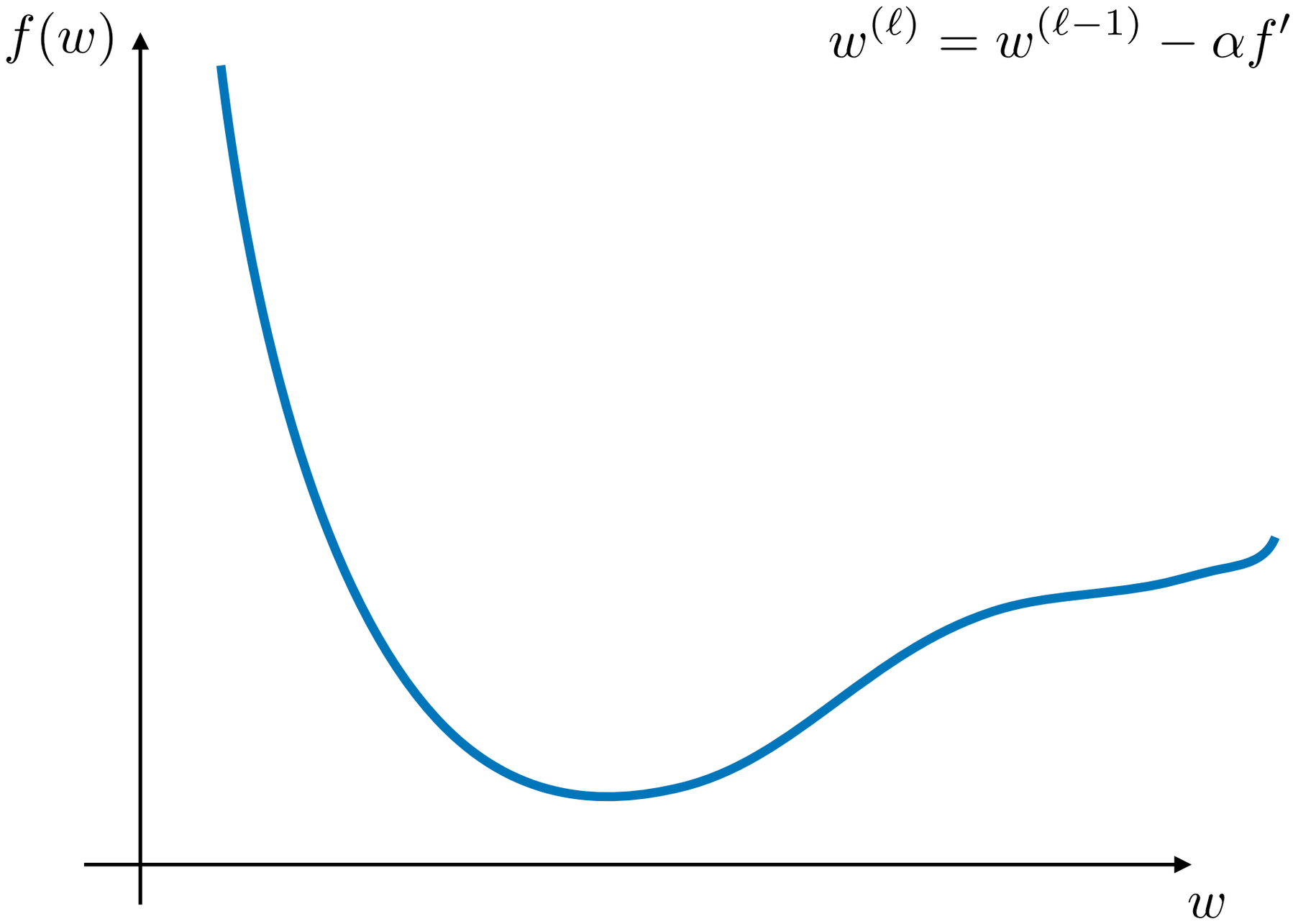
the step size

$$\nabla_{\boldsymbol{w}_+} \mathcal{L} = \sum_{i=1}^{N} \left(\sigma \left(\boldsymbol{w}_+ \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right) \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

Choosing the step size manually is a **hard problem!**

Gradient Descent

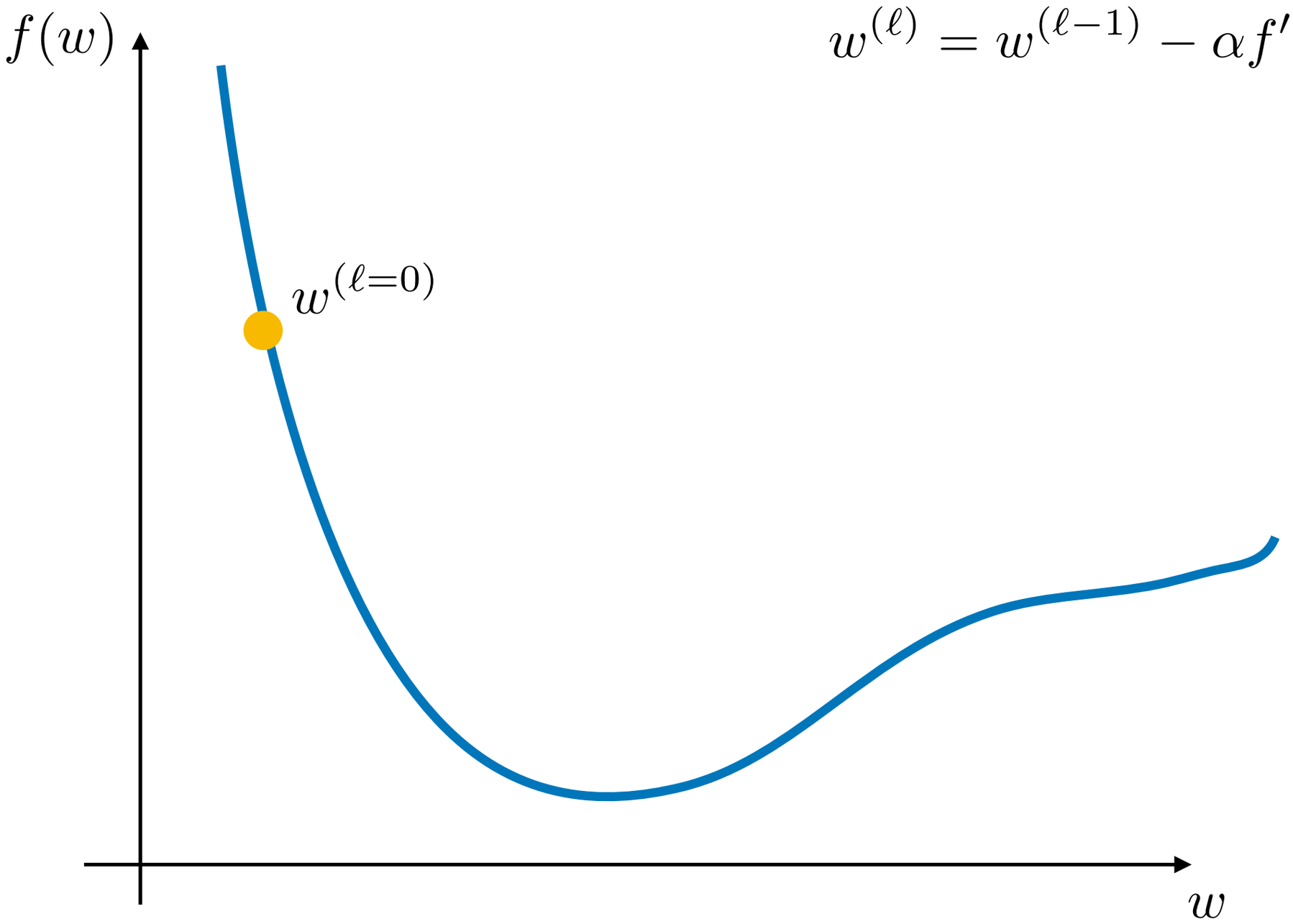
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too large ….

Gradient Descent

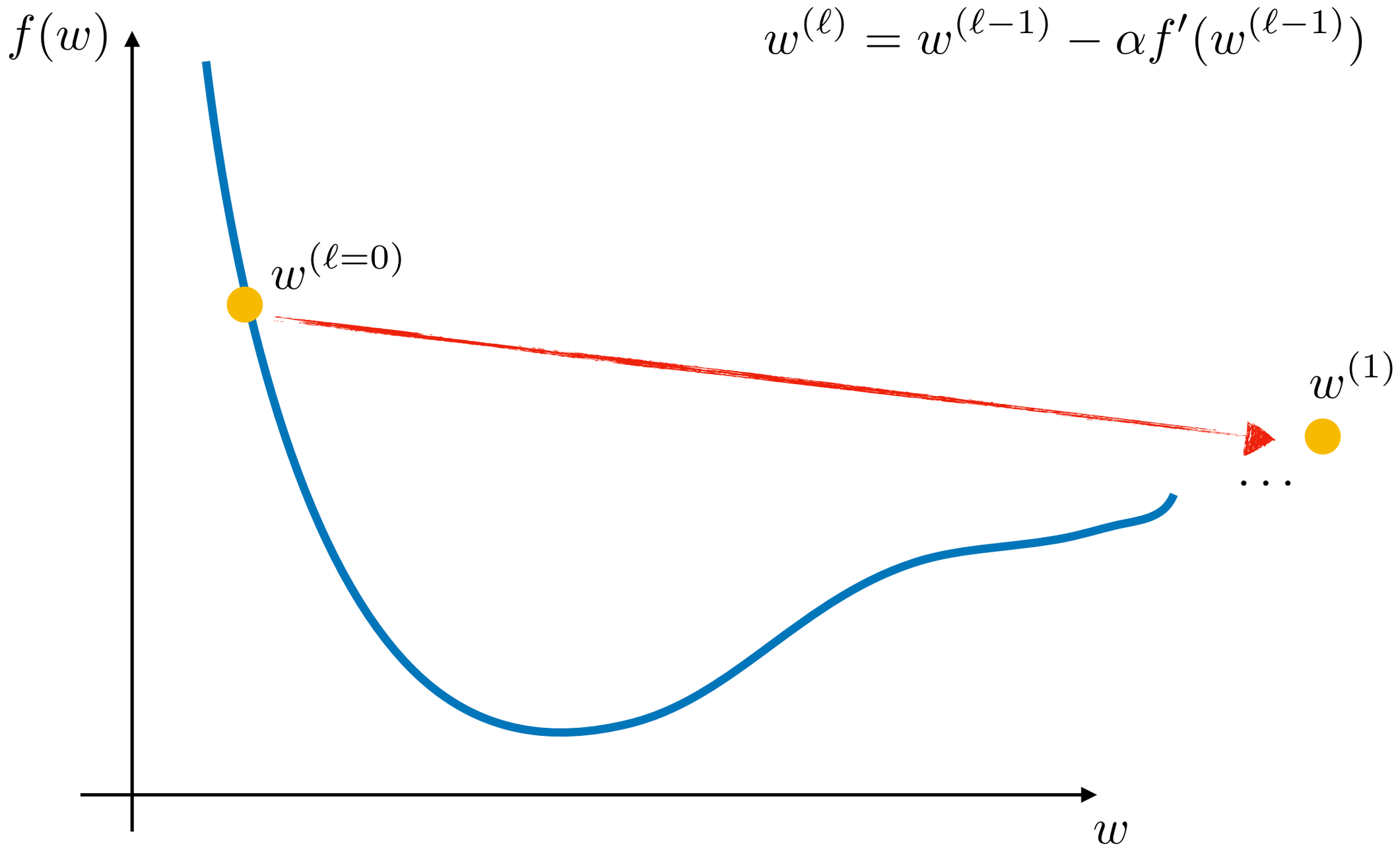
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too large ….

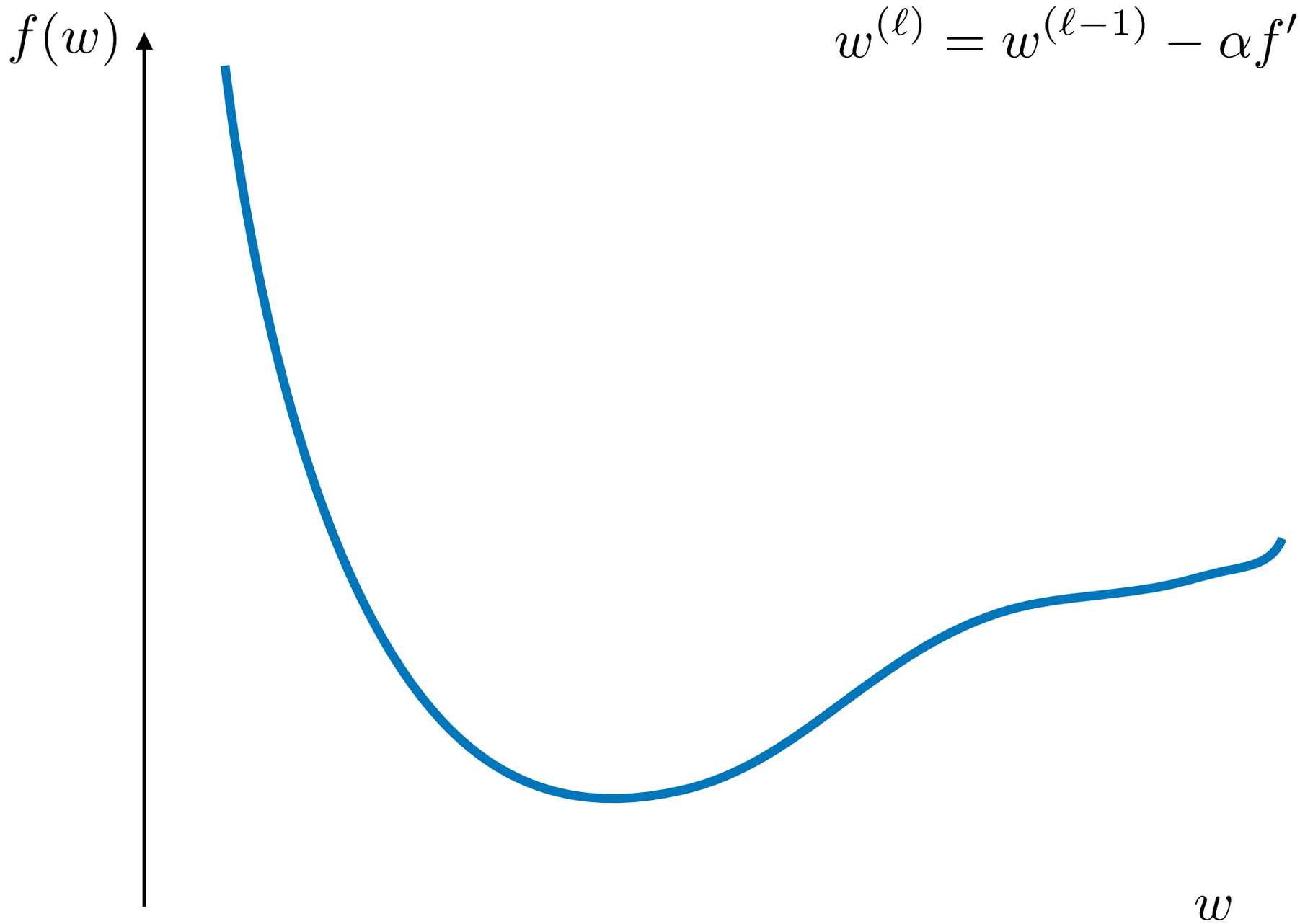
Gradient Descent

$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too large ….

Gradient Descent

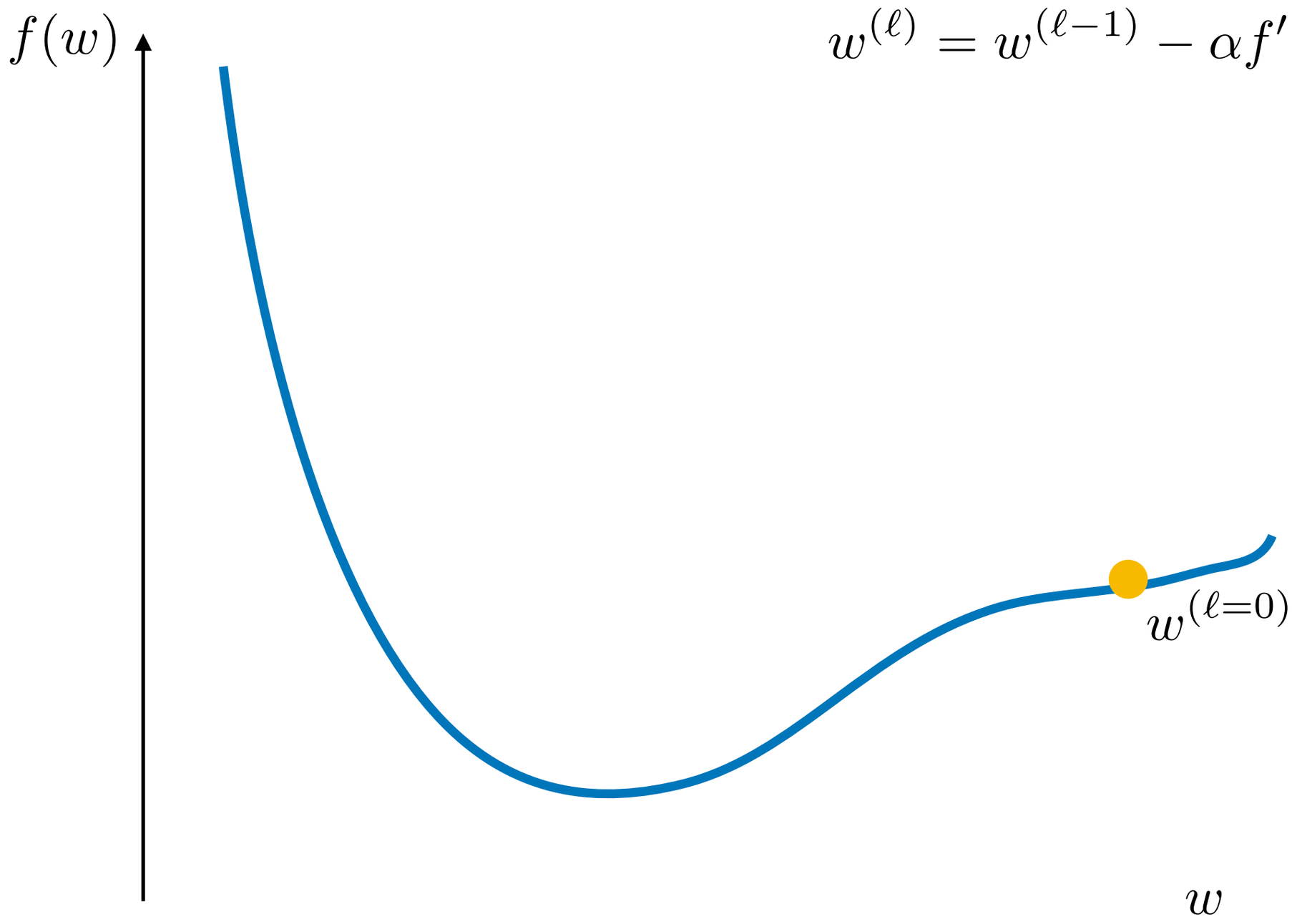


$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$

If α is too small ….

Gradient Descent

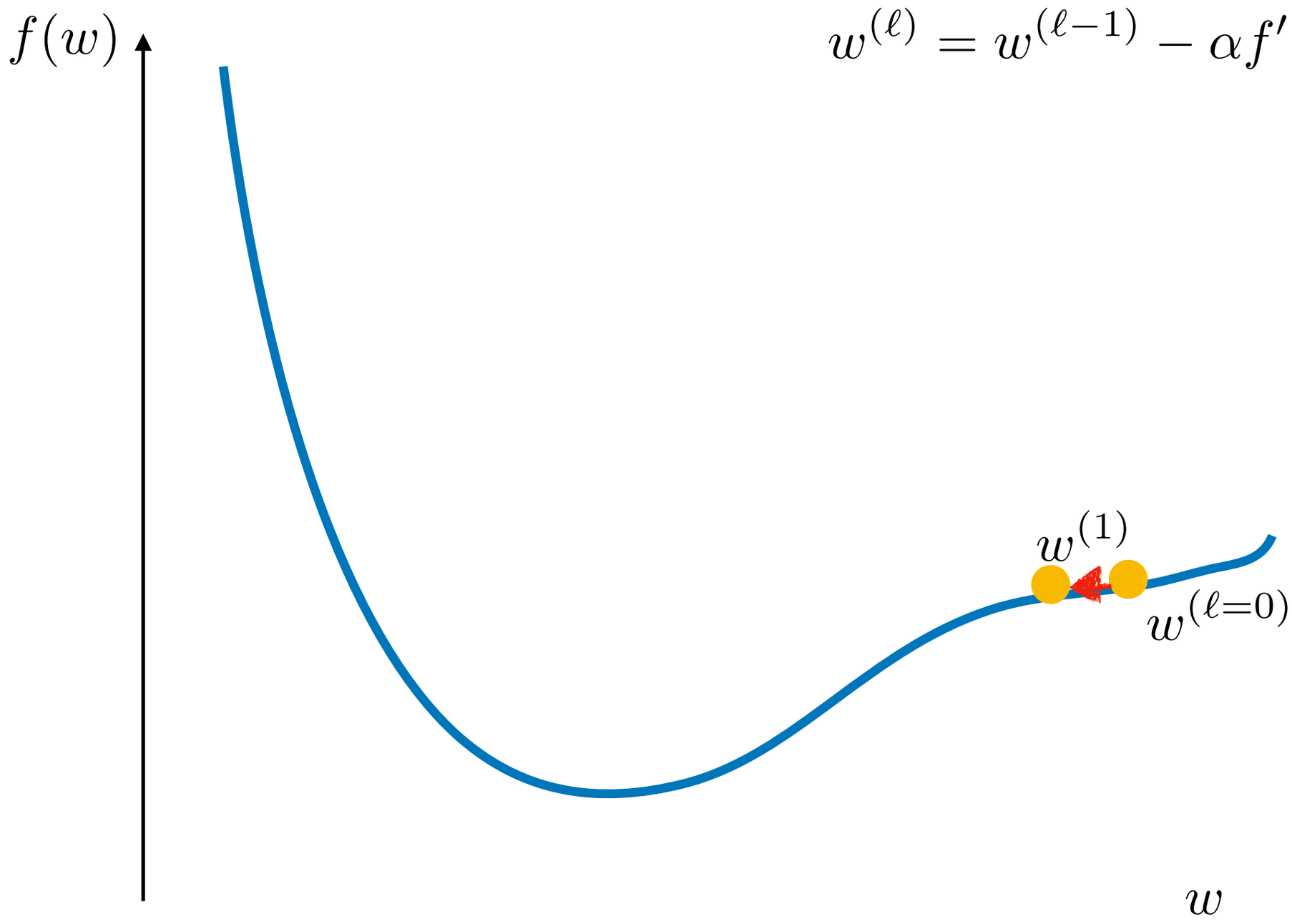
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

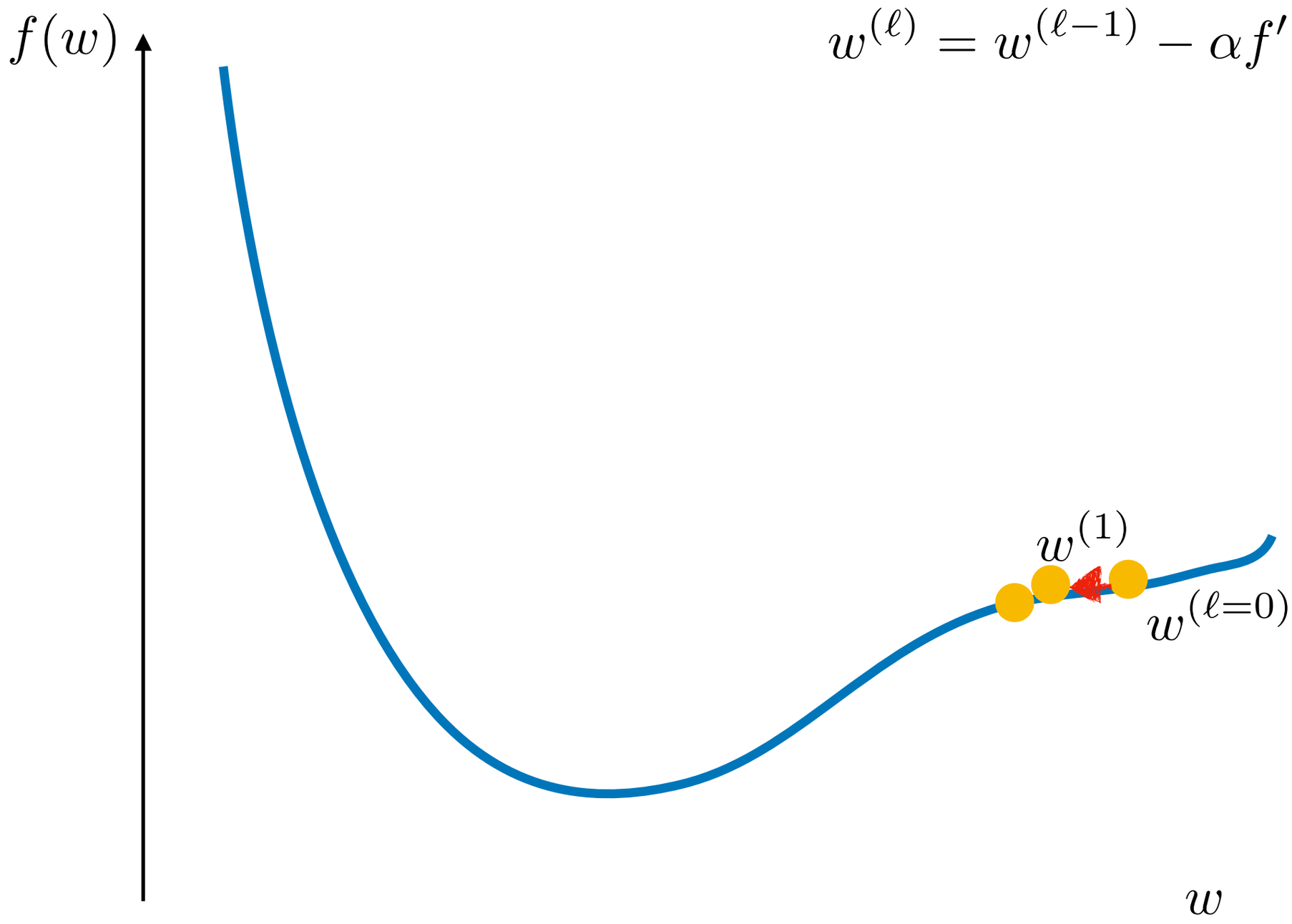
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

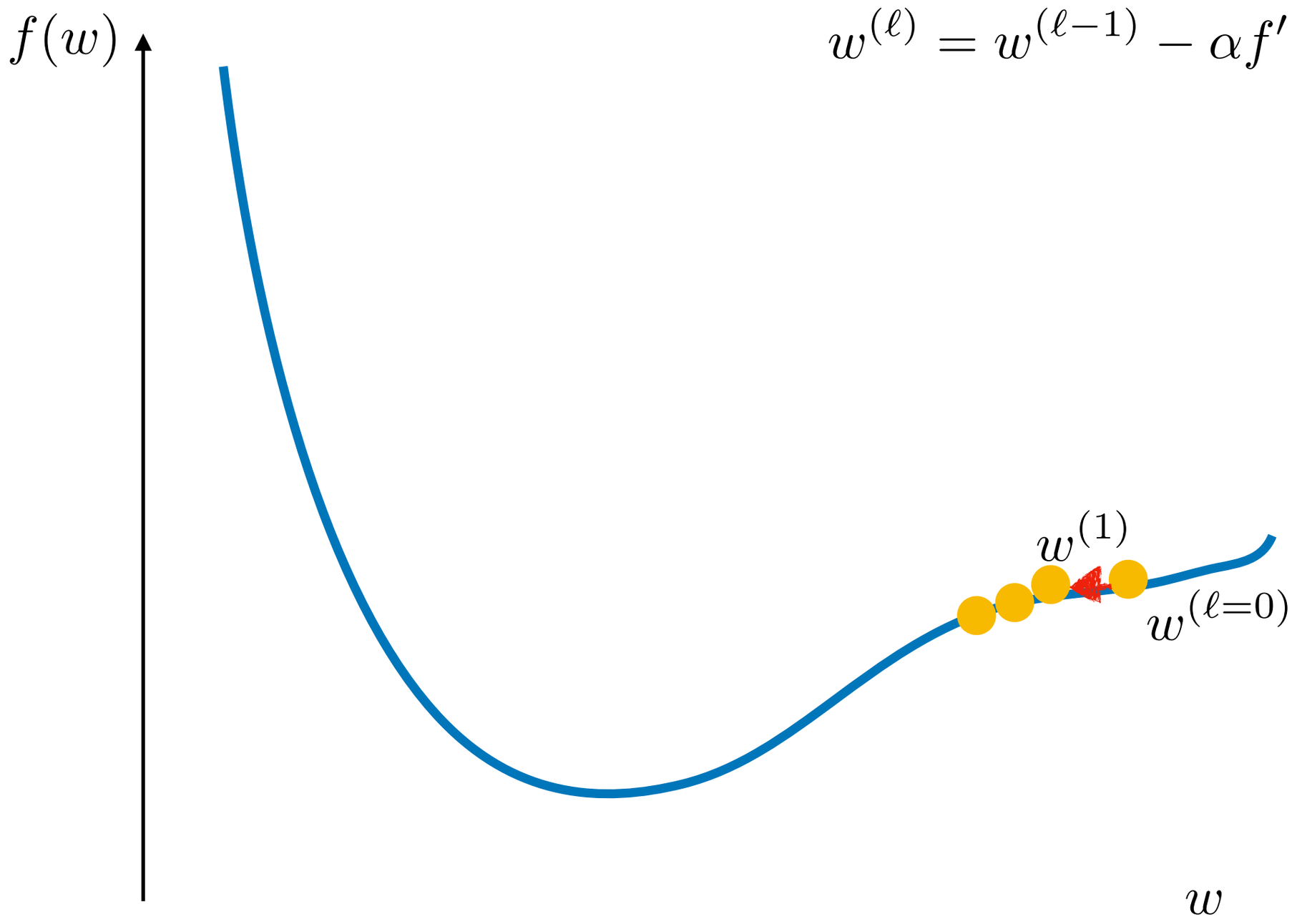
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

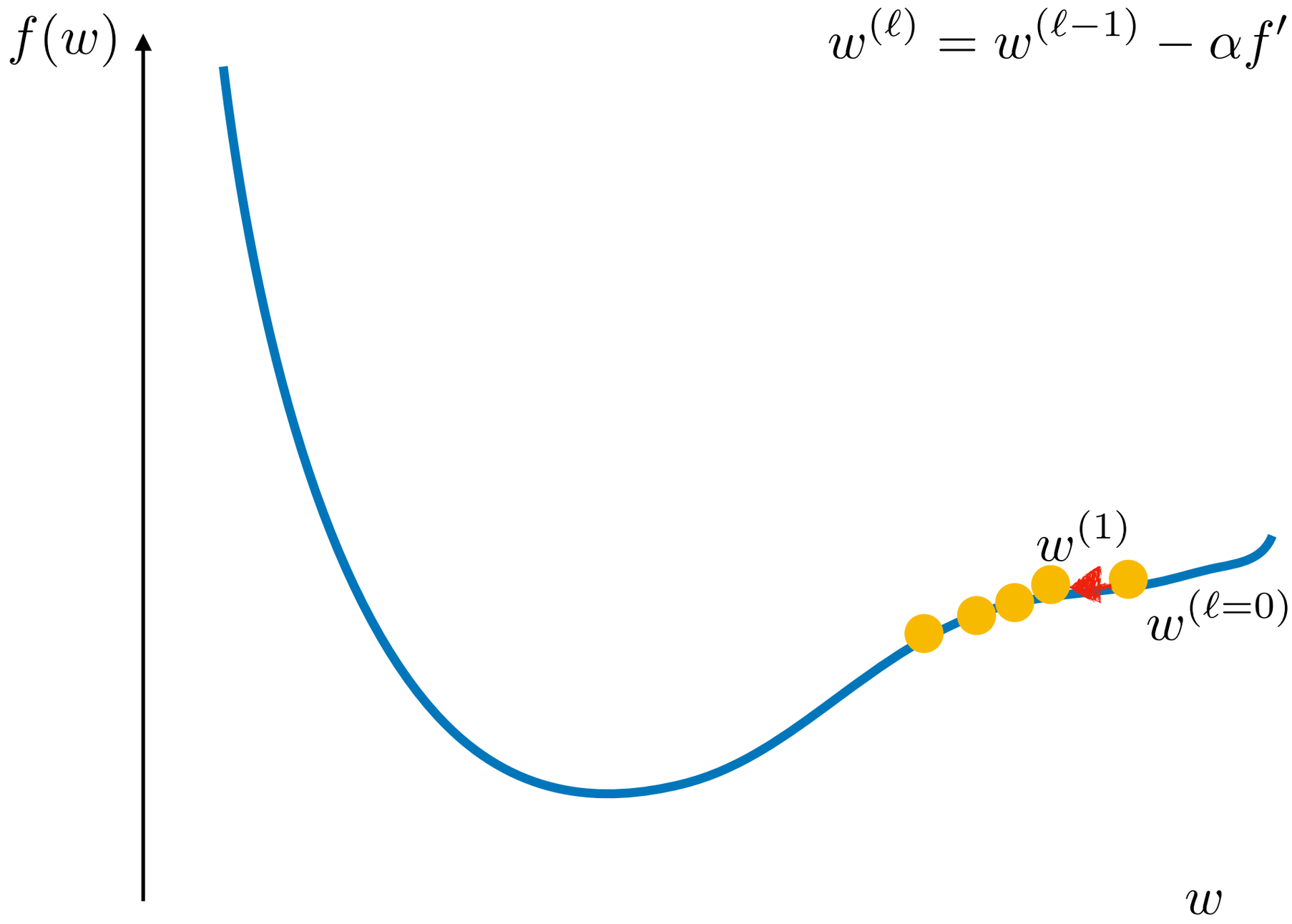
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

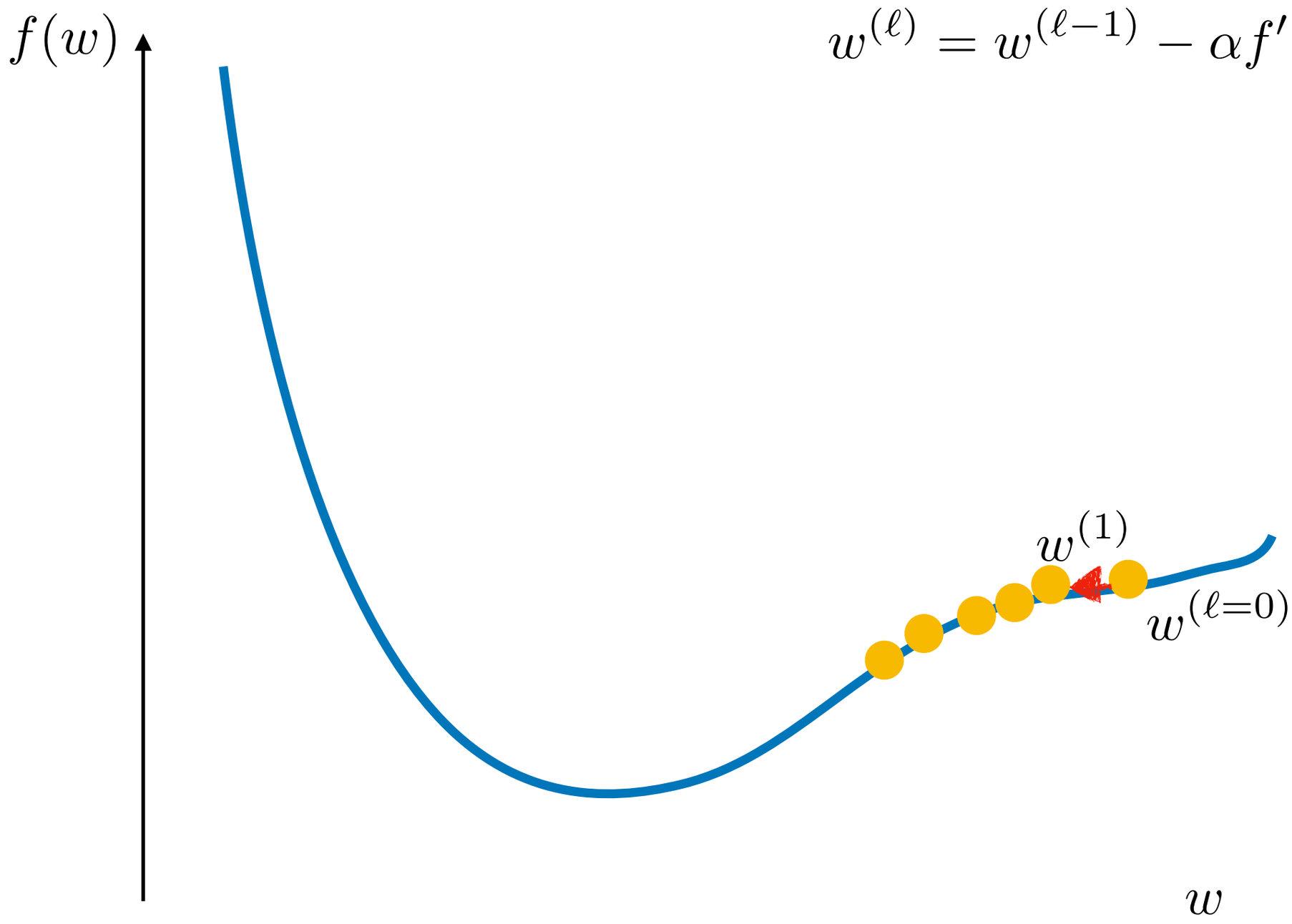
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

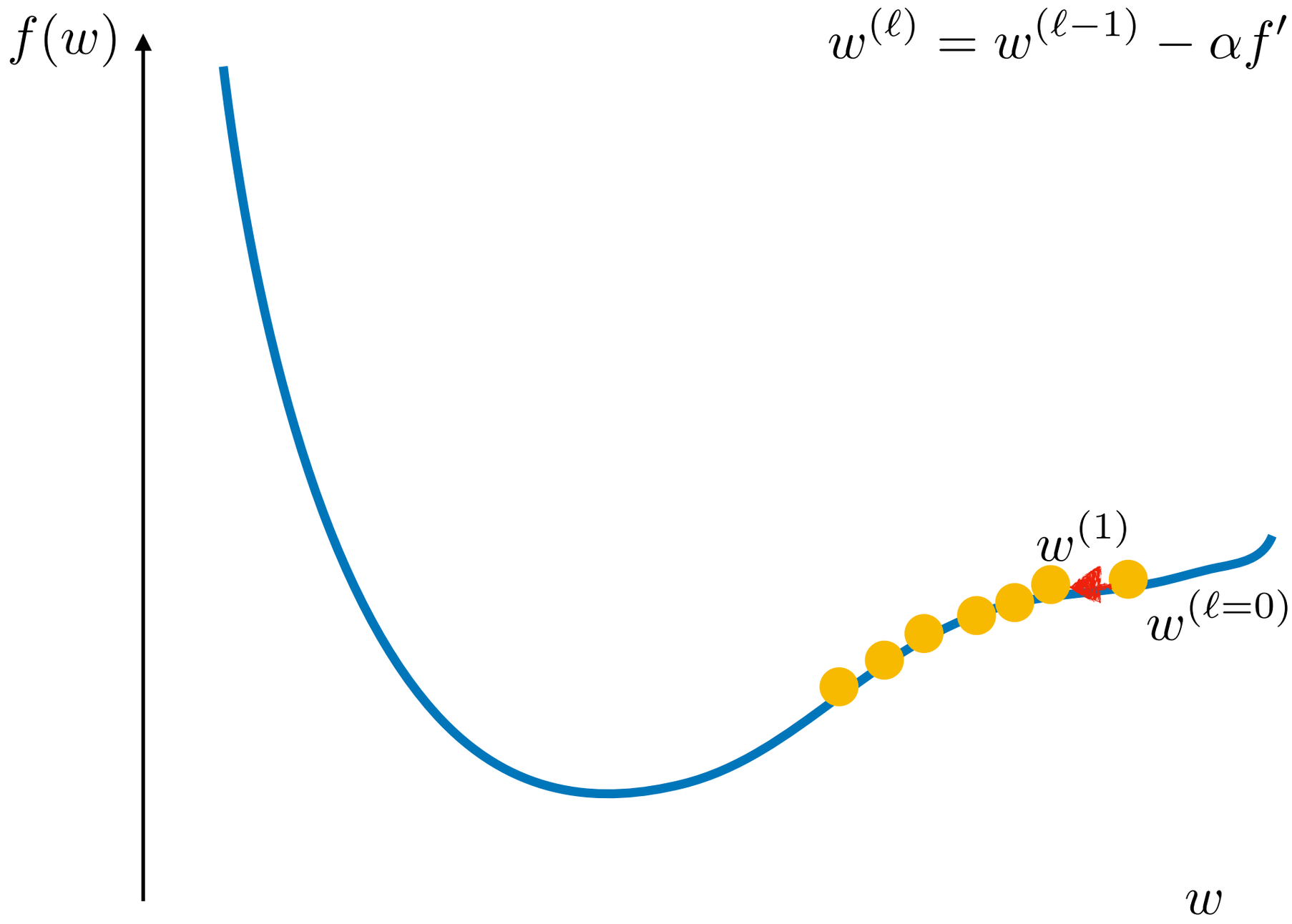
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

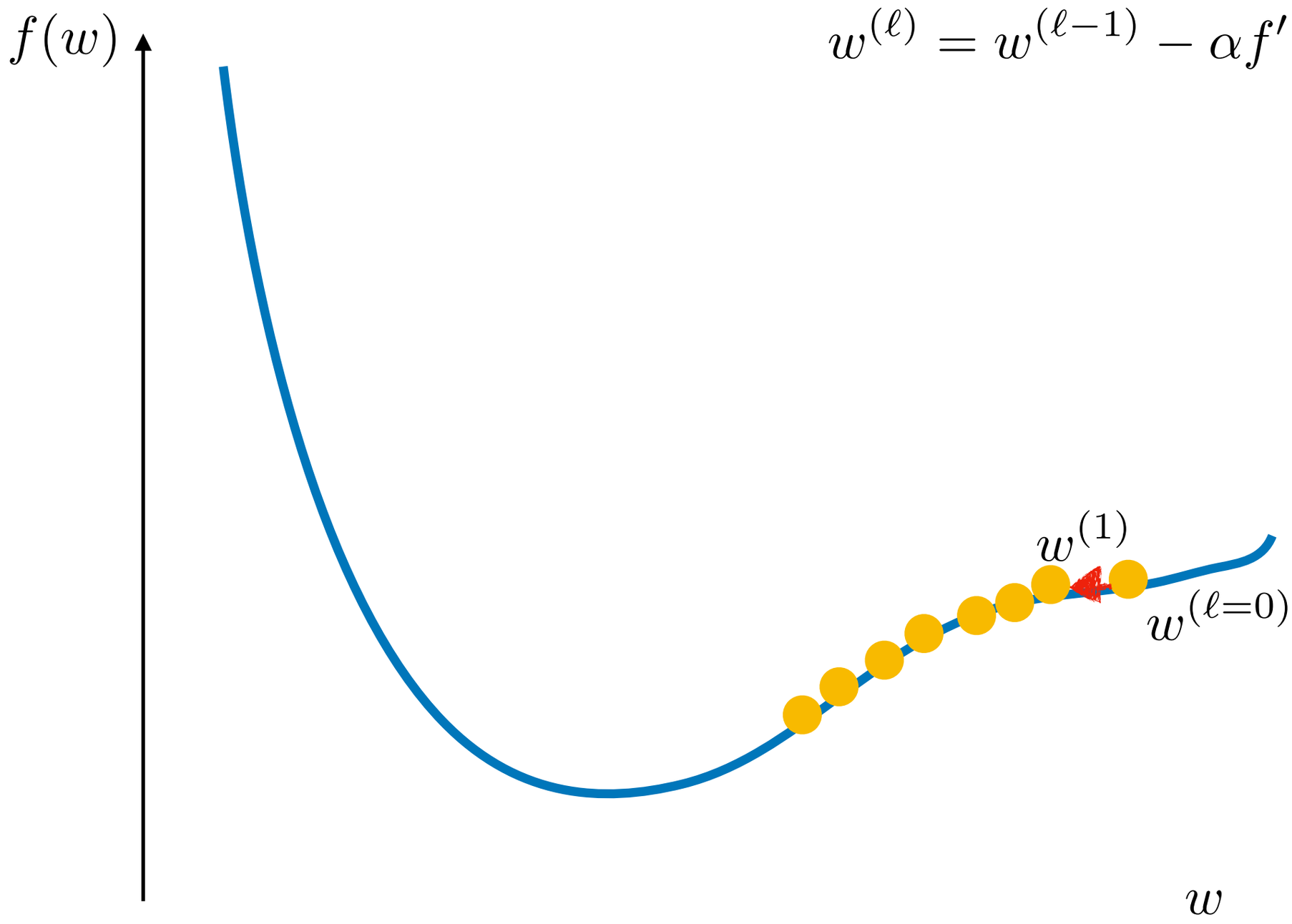
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

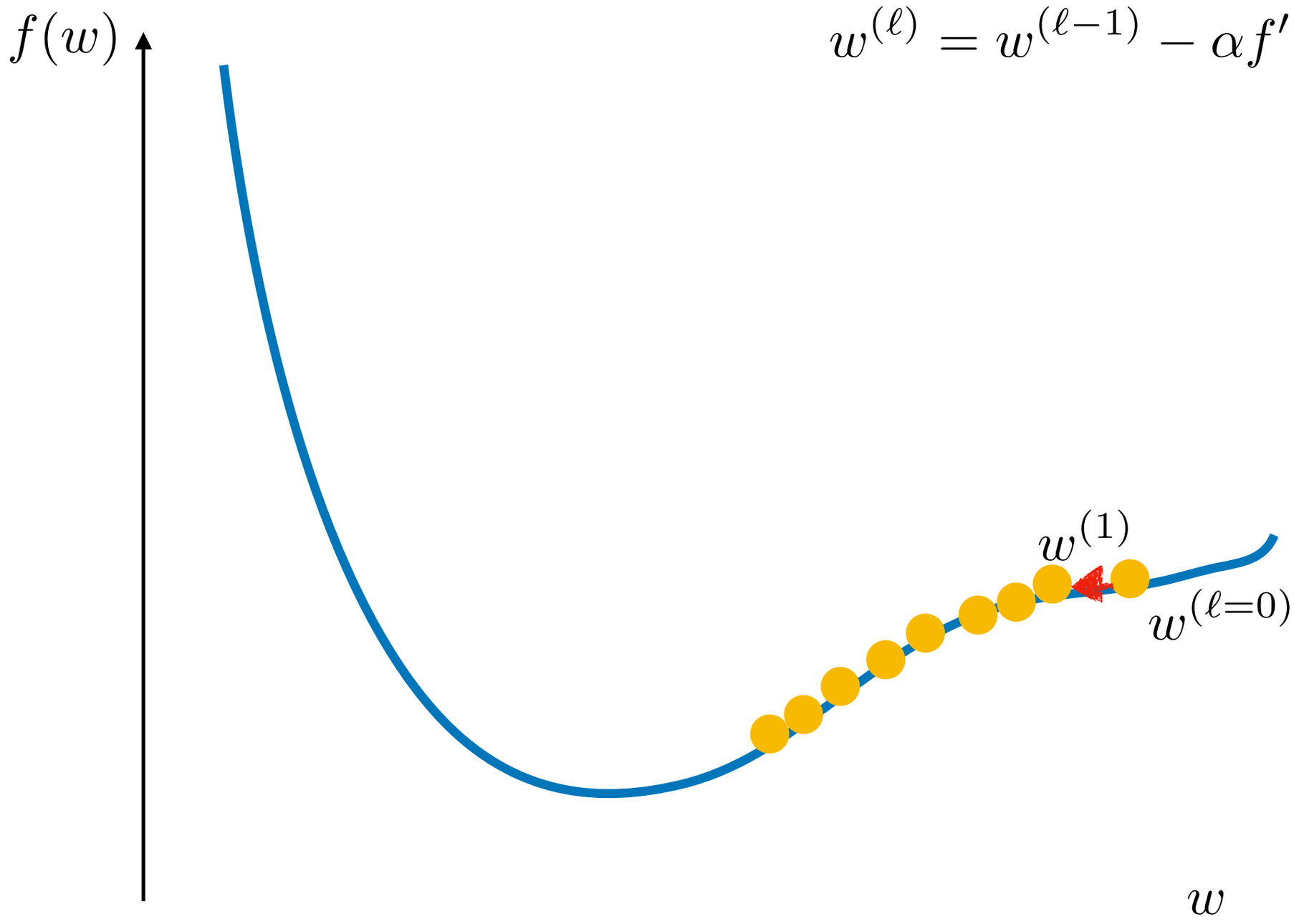
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

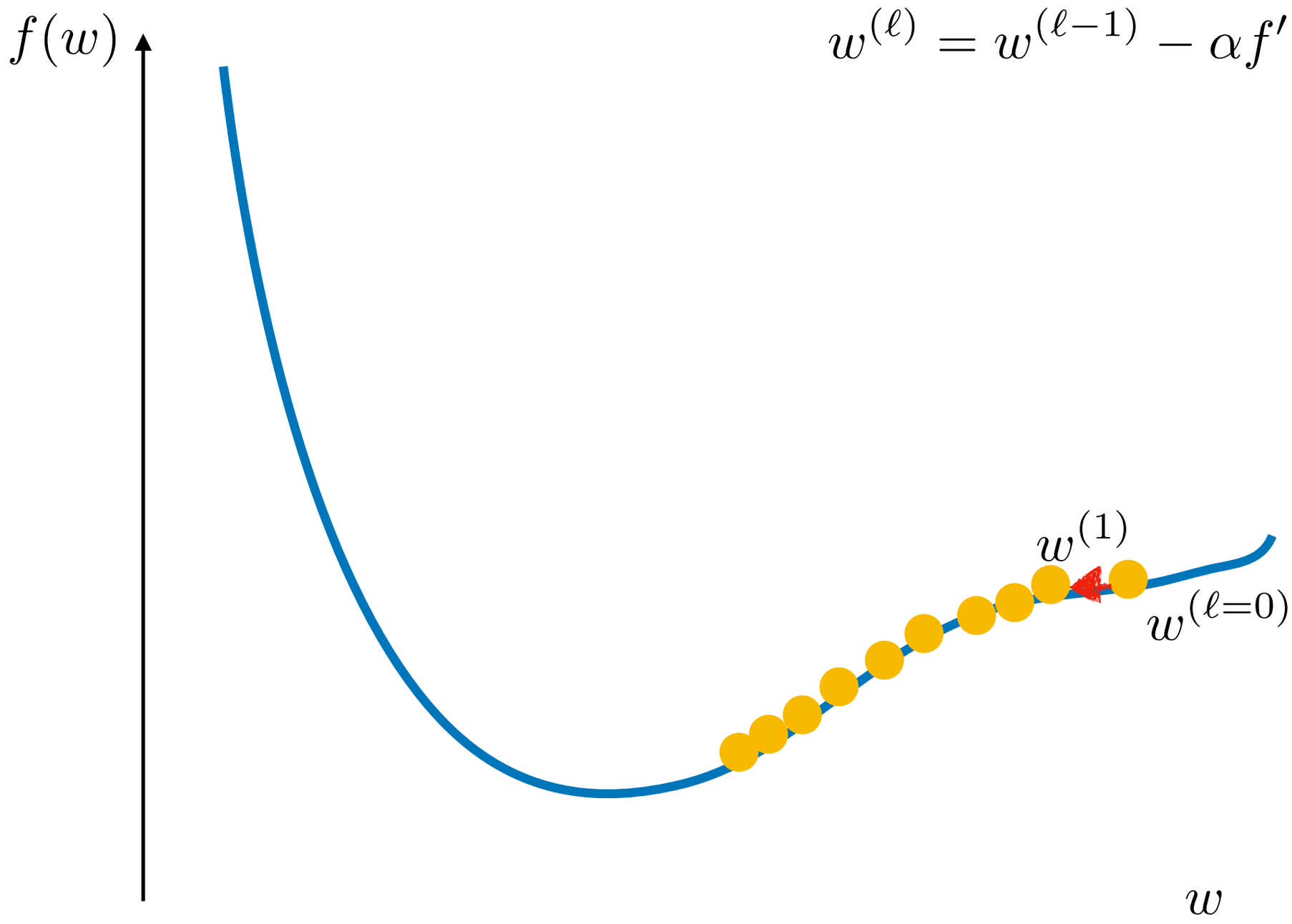
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

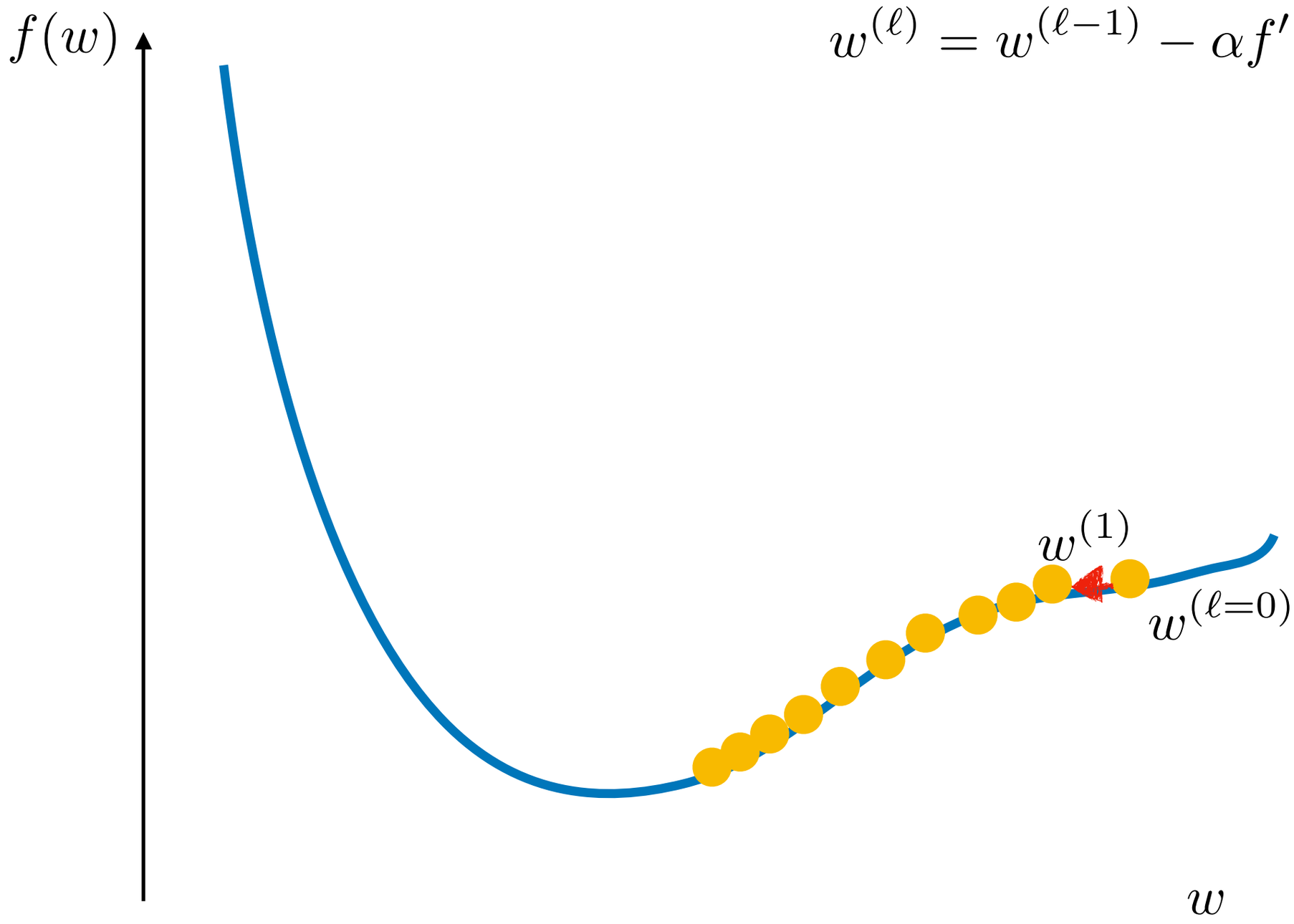
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

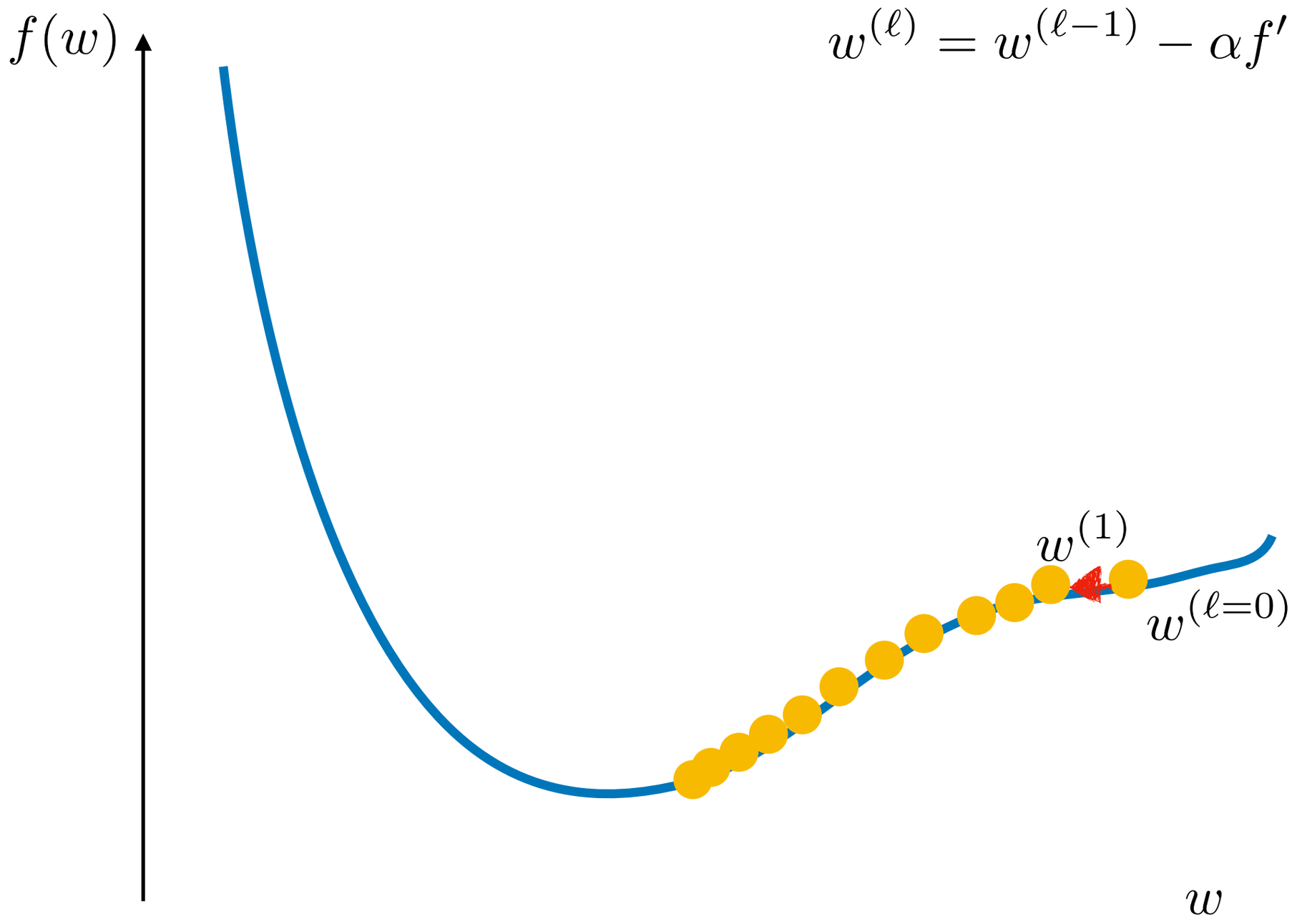
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

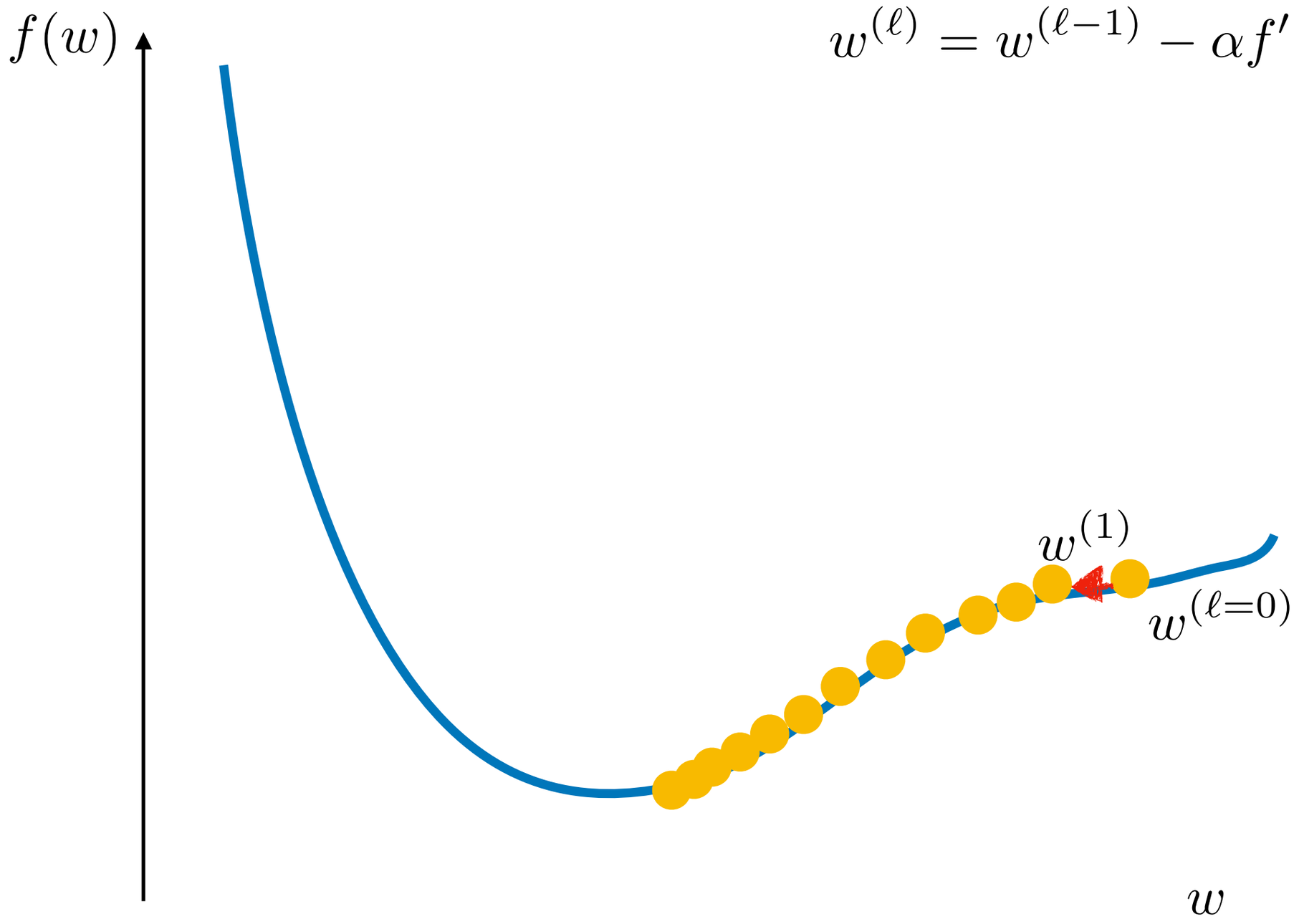
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

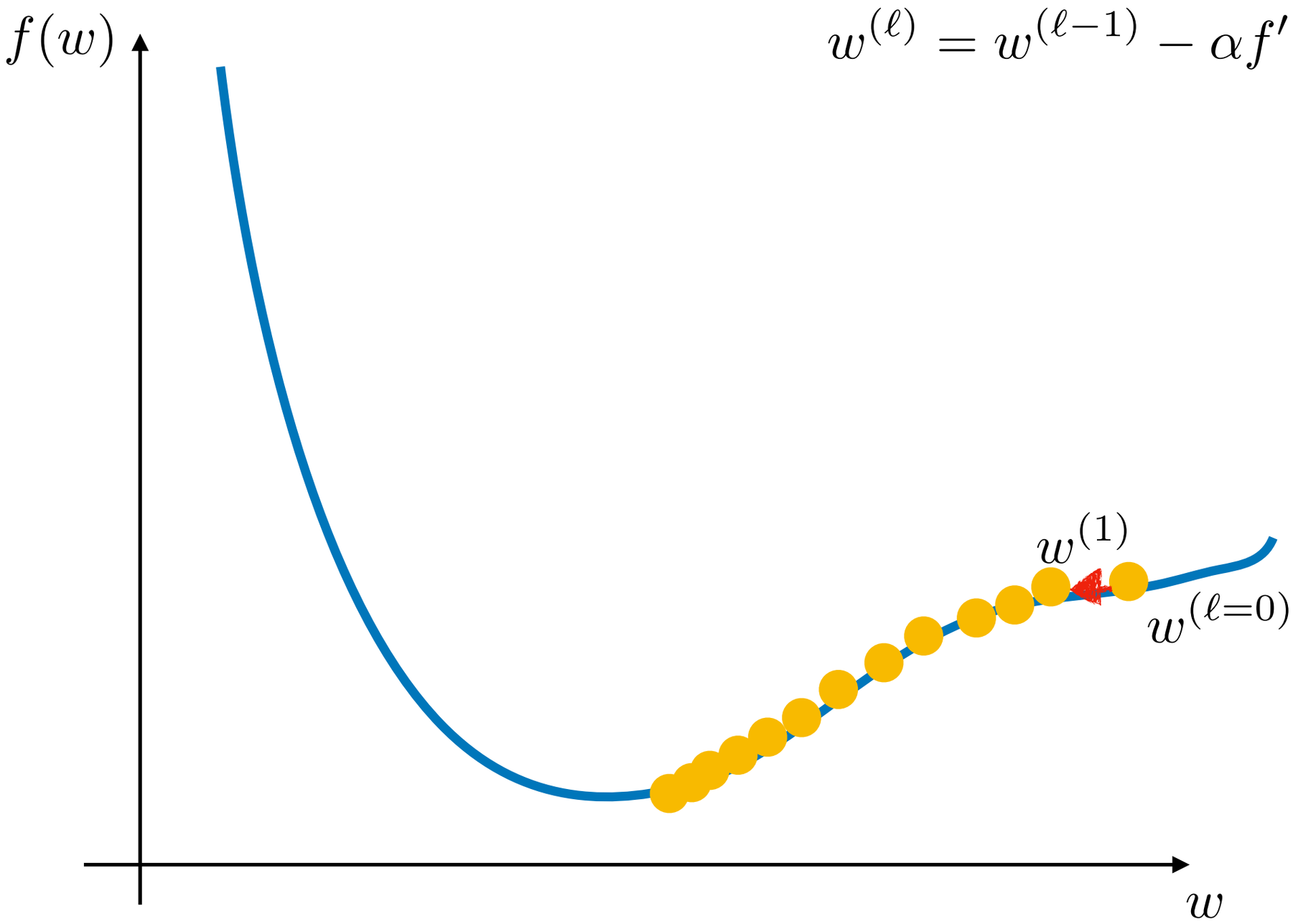
$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Gradient Descent

$$w^{(\ell)} = w^{(\ell-1)} - \alpha f'(w^{(\ell-1)})$$



If α is too small ….

Line Search methods: automatically running the step size

$$\boldsymbol{w}_{+}^{(\ell)} = \boldsymbol{w}_{+}^{(\ell-1)} - \left(\alpha^{(\ell)} \nabla \mathcal{L}\right) |_{\boldsymbol{w}_{+}^{(\ell-1)}}$$

- Estimate the largest possible step size that guarantees that the function decreases
- Every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent
- Momentum, Adam, Adagrad, …

Check out in Aula Global two excellent posts on SGD methods for Deep Learning

(Link1, Link2)

Logistic Regression:
Stochastic gradient descent

uc3m

Stochastic Optimization (Mini-batch Optimization)

$$\boldsymbol{w}_+ \doteq \left[\begin{array}{cc} w_0 & \boldsymbol{w} \end{array} \right] \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - \left(\alpha \nabla \mathcal{L} \right) \big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

$$\nabla_{\boldsymbol{w}_+} \mathcal{L} = \sum_{i=1}^{N} \left(\sigma \left(\boldsymbol{w}_+ \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array} \right] \right) - y^{(i)} \right) \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array} \right]$$

Stochastic Optimization (Mini-batch Optimization)

$$\boldsymbol{w}_+ \doteq \left[\begin{array}{cc} w_0 & \boldsymbol{w} \end{array} \right] \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - \left(\alpha \nabla \mathcal{L}\right)\Big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

Expensive for very large databases!!

$$\nabla_{\boldsymbol{w}_+} \mathcal{L} = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w}_+ \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array} \right] \right) - y^{(i)} \right) \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array} \right]$$

Stochastic Optimization (Mini-batch Optimization)

$$\boldsymbol{w}_+ \doteq \left[\begin{array}{cc} w_0 & \boldsymbol{w} \end{array} \right] \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - \left(\alpha \nabla \mathcal{L}\right)\big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

Expensive for very large databases!!

$$\nabla_{\boldsymbol{w}_+} \mathcal{L} = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w}_+ \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array} \right] \right) - y^{(i)} \right) \left[\begin{array}{c} 1.0 \\ \boldsymbol{x}^{(i)} \end{array} \right]$$

Mini-batch optimization

Select at random a mini-batch $\mathcal{B}$ of data at every SGD iteration

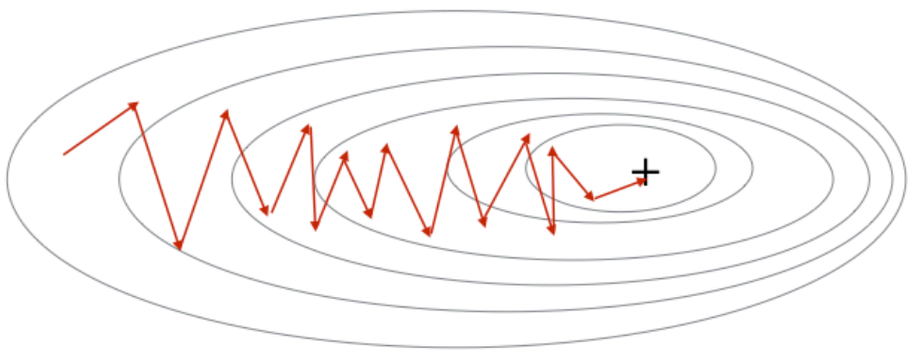
Stochastic Optimization (Mini-batch Optimization)

$$\boldsymbol{w}_+ \doteq \begin{bmatrix} w_0 & \boldsymbol{w} \end{bmatrix} \Rightarrow \nabla_{\boldsymbol{w}_+} \mathcal{L} = \mathbf{0}$$

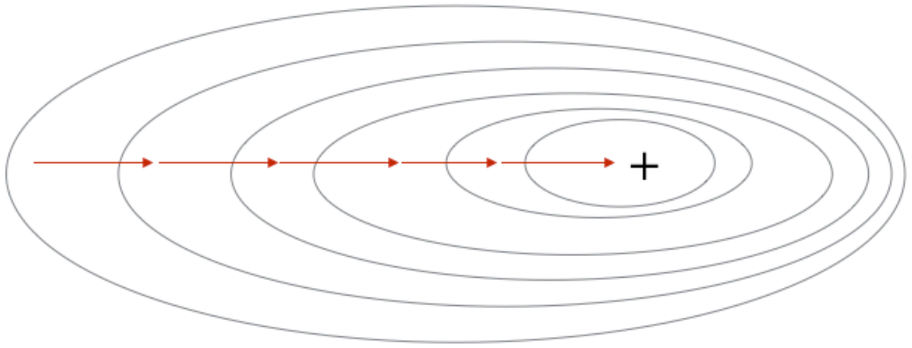
$$\boldsymbol{w}_+^{(\ell)} = \boldsymbol{w}_+^{(\ell-1)} - \left(\alpha \nabla \mathcal{L}\right)\big|_{\boldsymbol{w}_+^{(\ell-1)}}$$

$$\nabla_{\boldsymbol{w}_+} \mathcal{L} \approx \sum_{i \in \mathcal{B}} \left(\sigma\left(\boldsymbol{w}_+ \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix} \right) - y^{(i)} \right) \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

Stochastic Gradient Descent



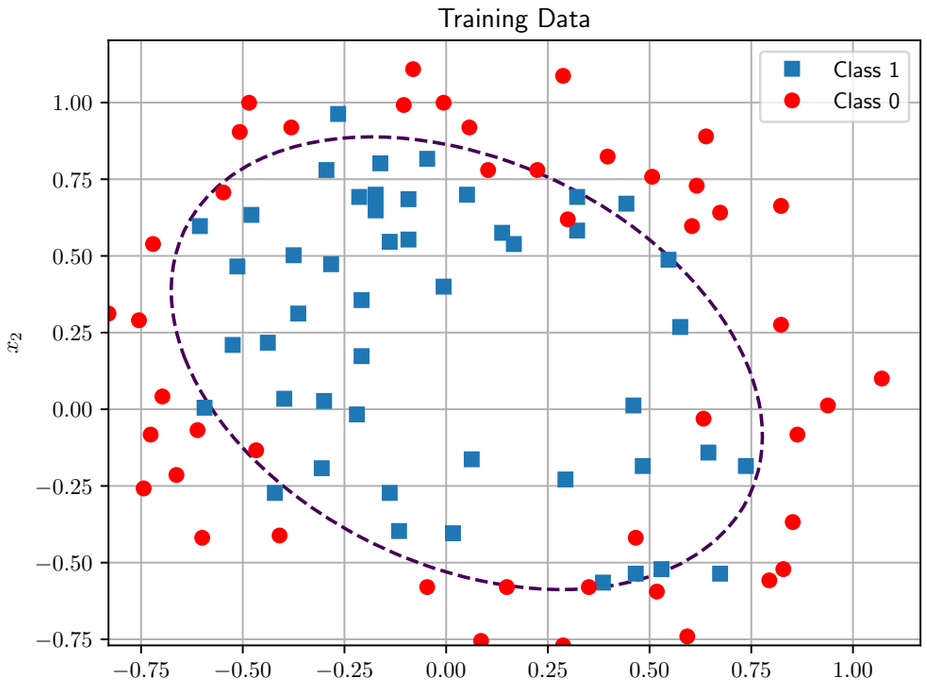
Gradient Descent

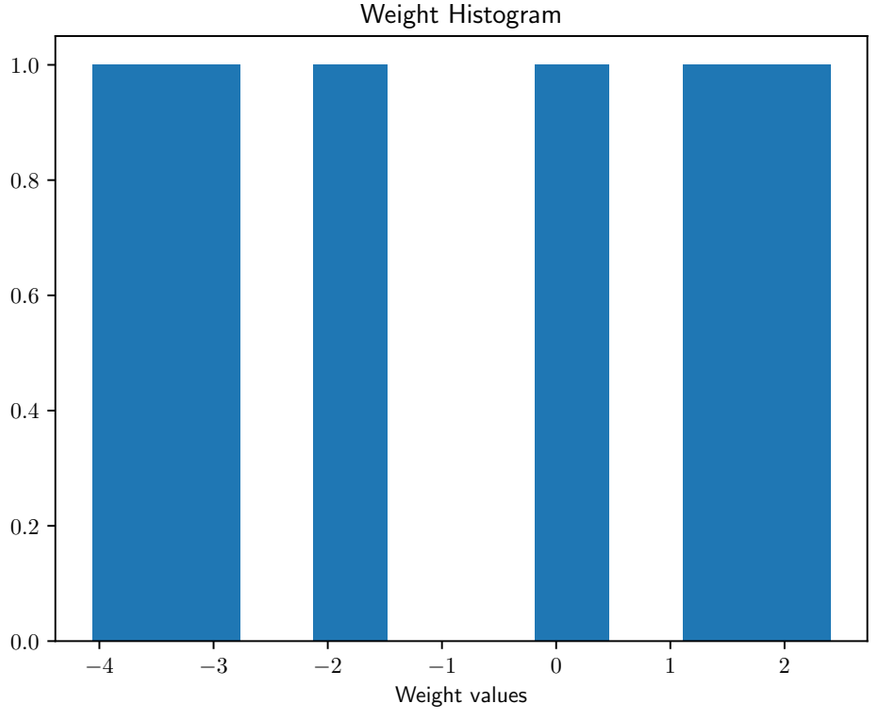


Source: this post

Logistic Regression: regularizing the weight norm

Binary Logistic Regression. Adding new features

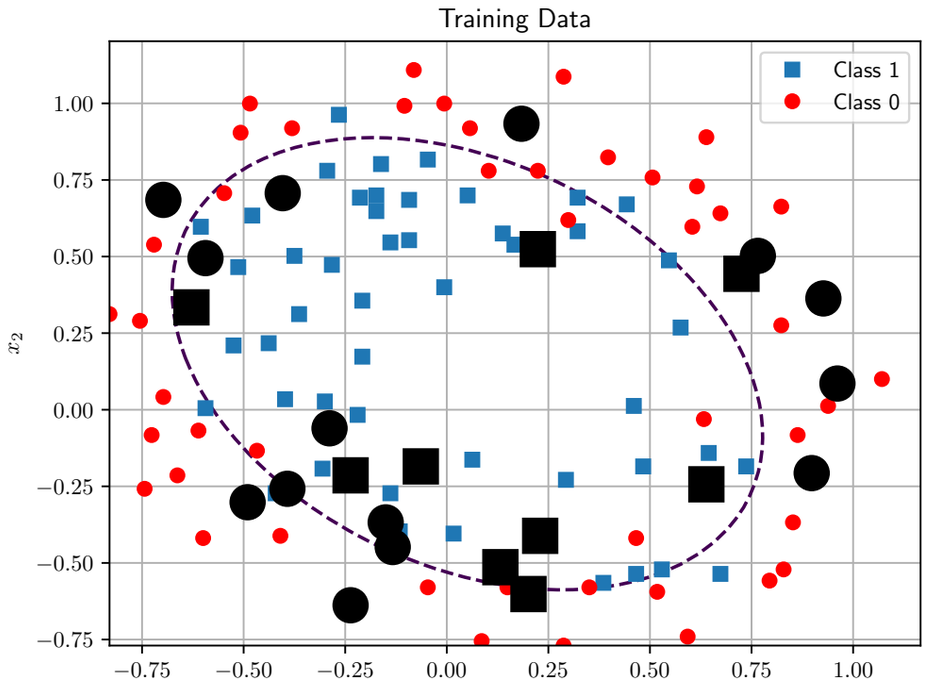


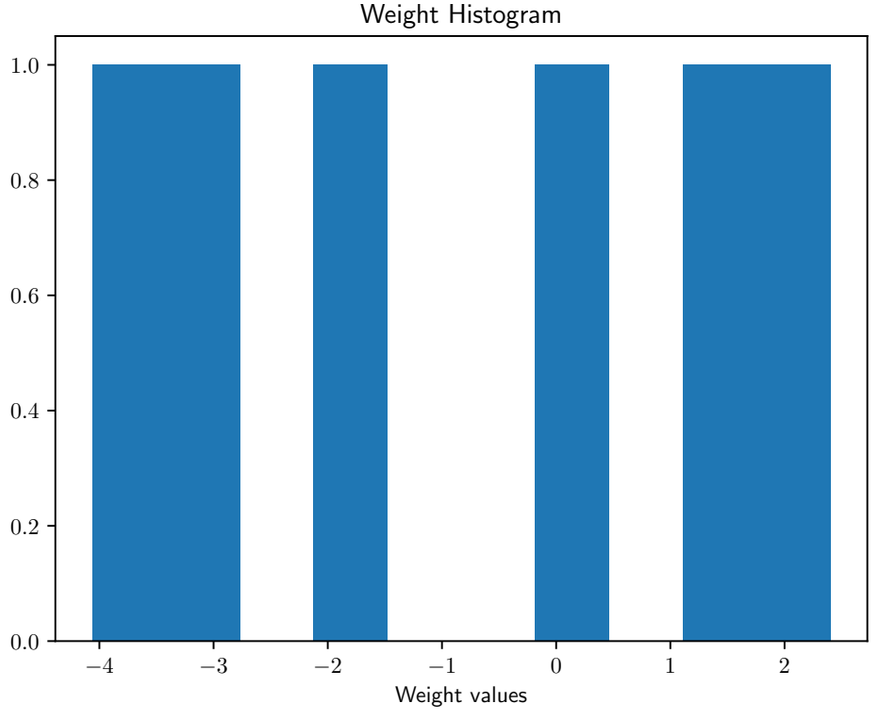


$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

Binary Logistic Regression. Adding new features

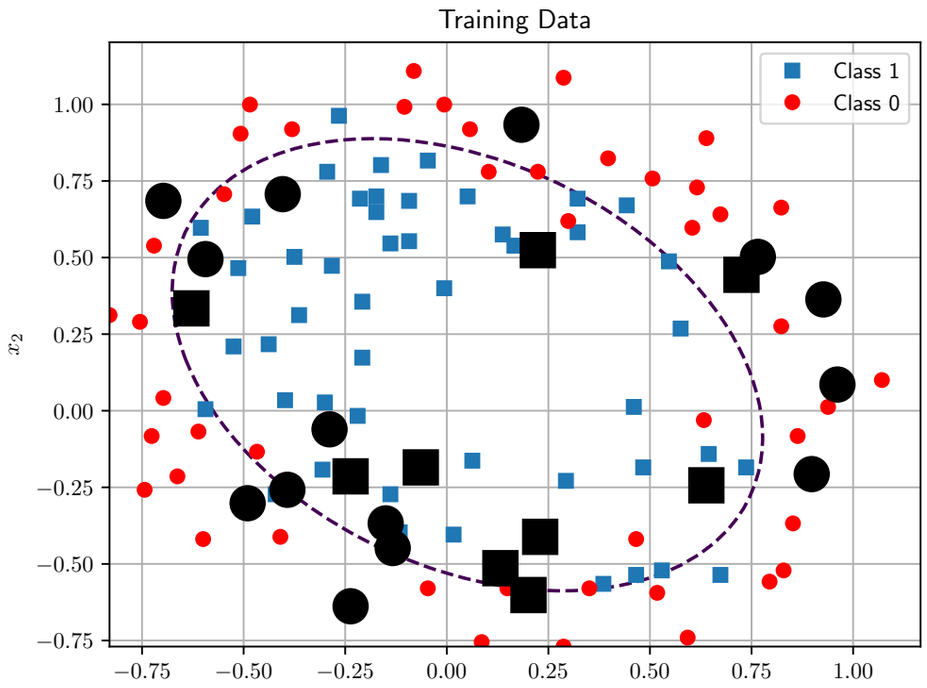




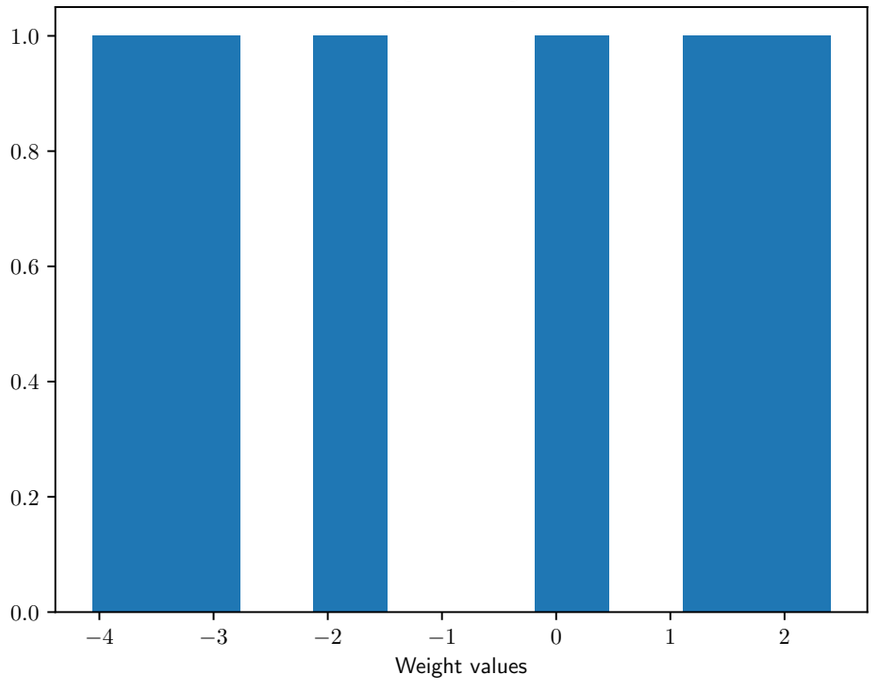
$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

Binary Logistic Regression. Adding new features



Weight Histogram

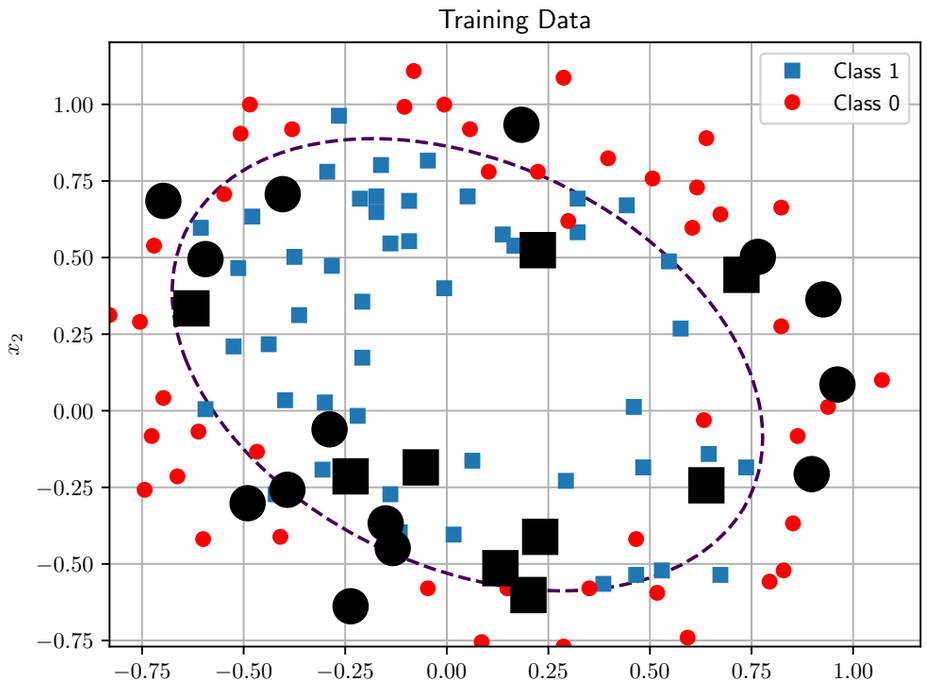


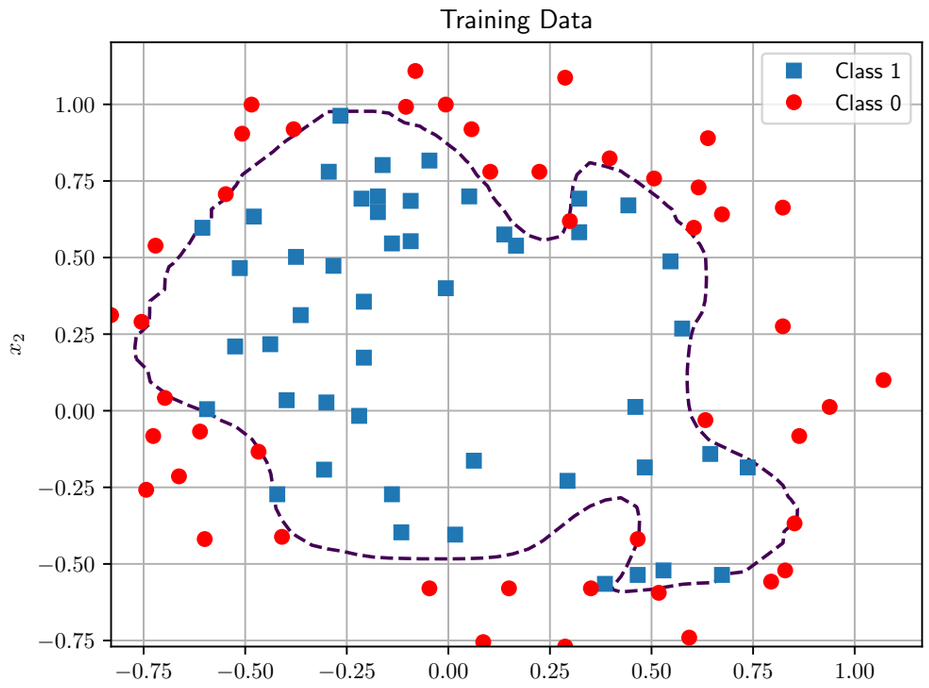
Test accuracy = 65.2 %

$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$

$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

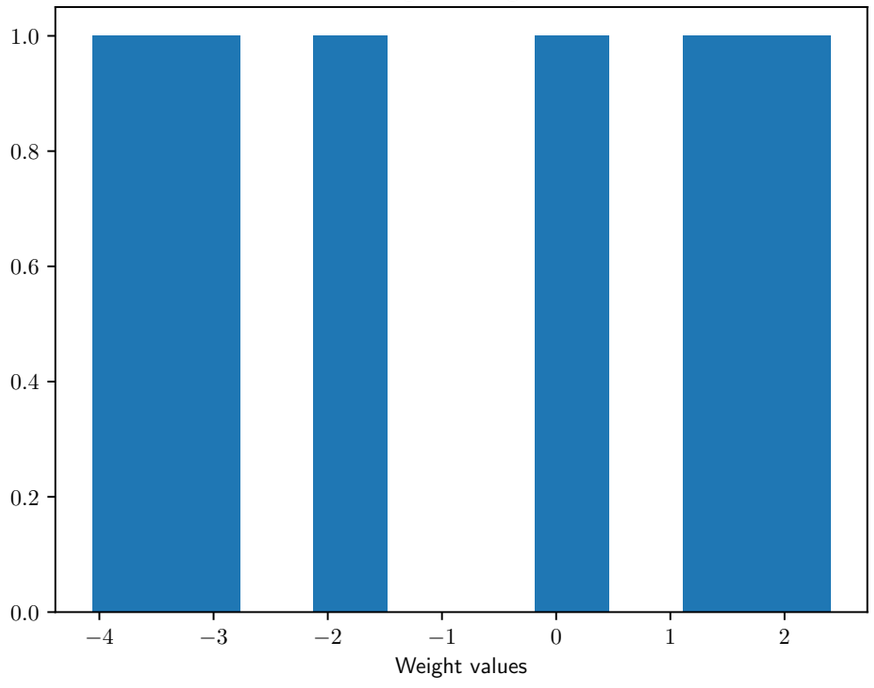
Binary Logistic Regression. Adding new features



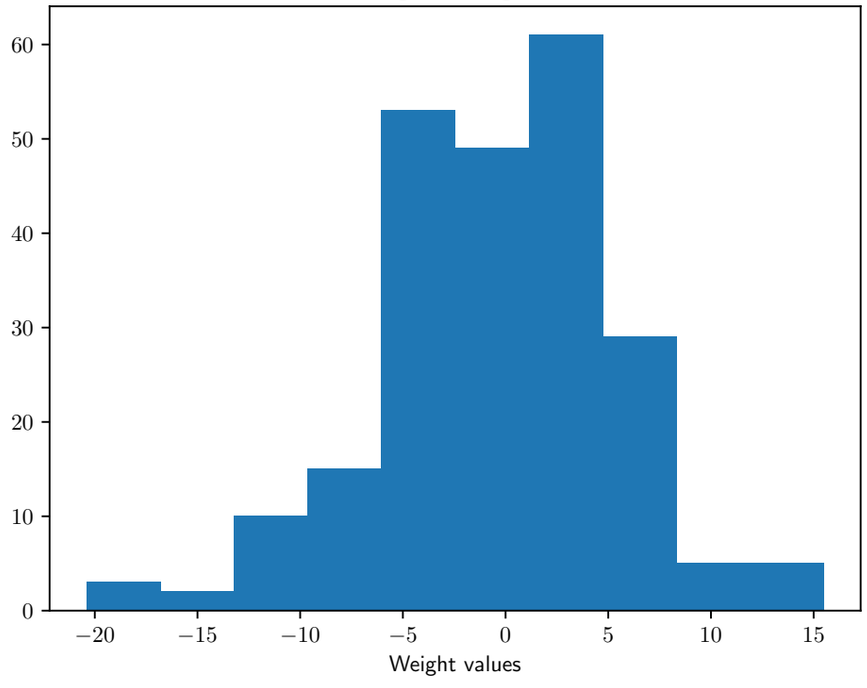


Test accuracy = 65.2 %

Weight Histogram



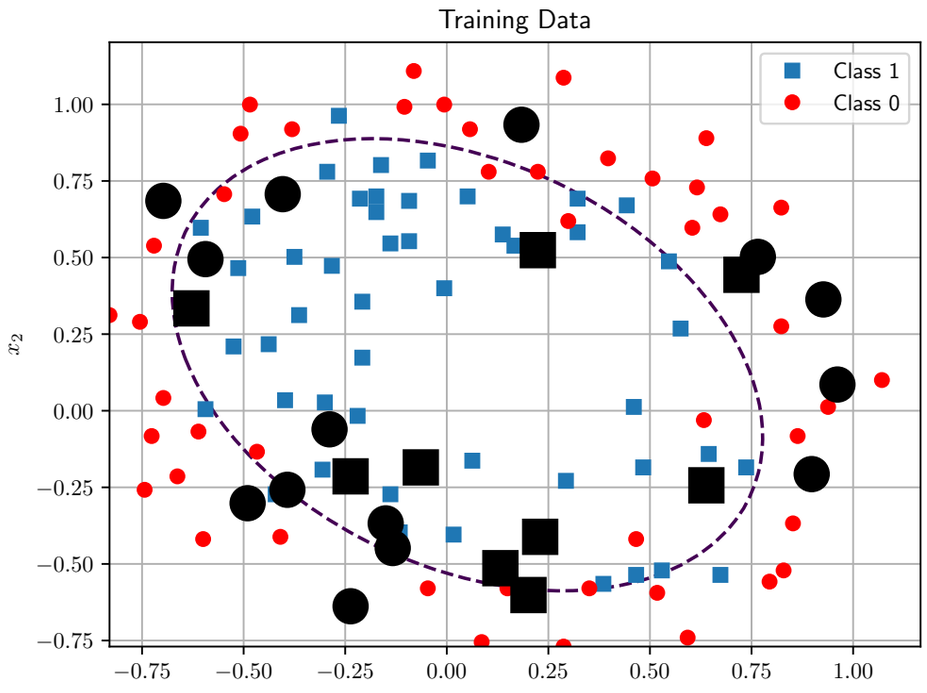
Weight Histogram

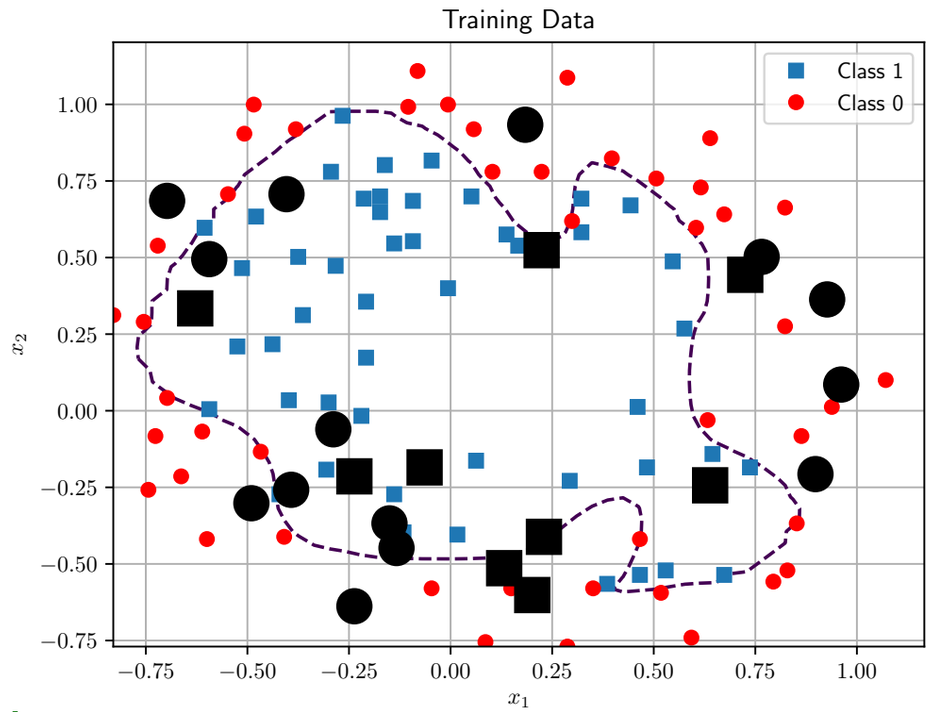


$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

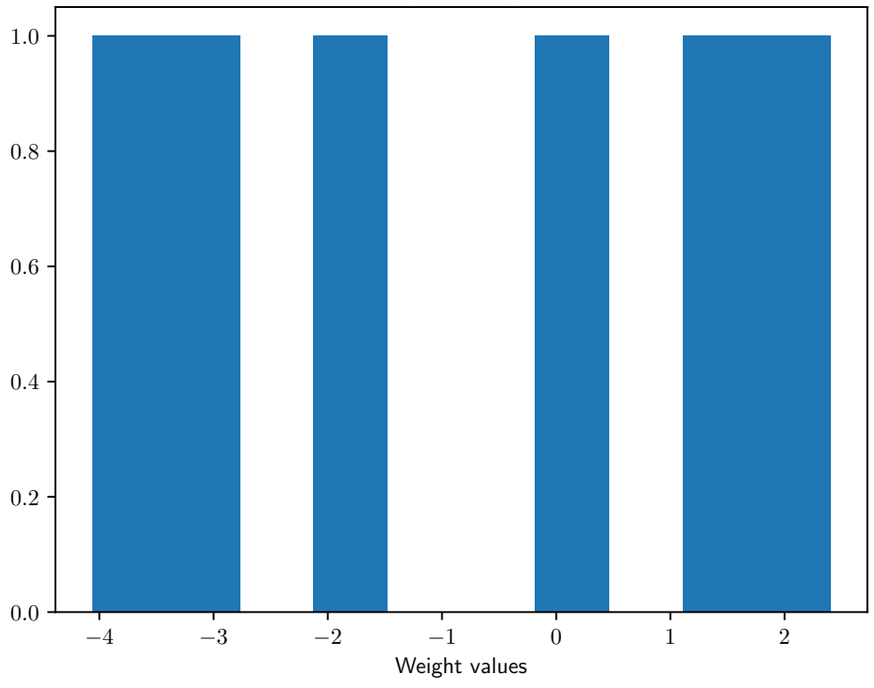
Binary Logistic Regression. Adding new features



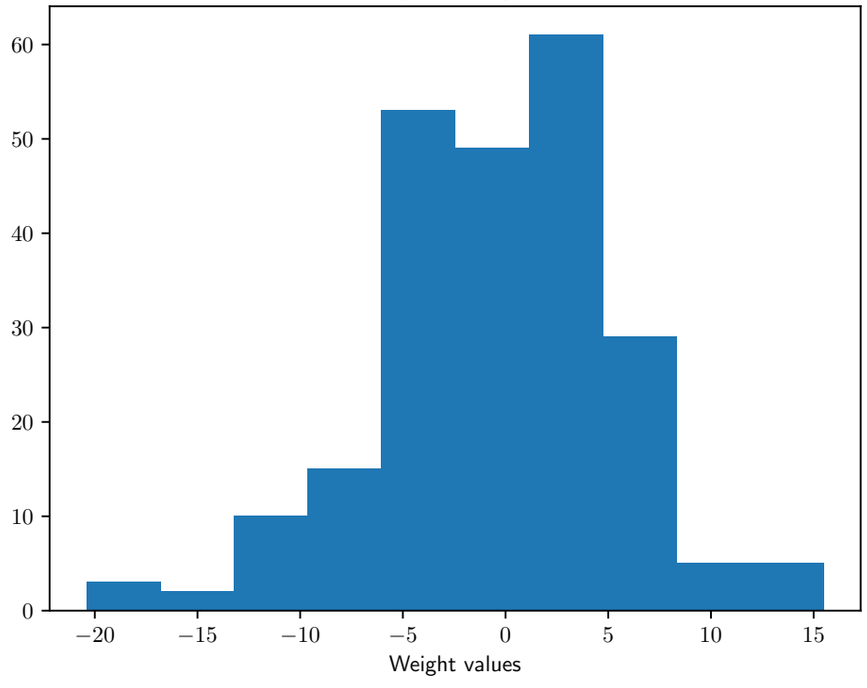


Test accuracy = 65.2 %

Weight Histogram



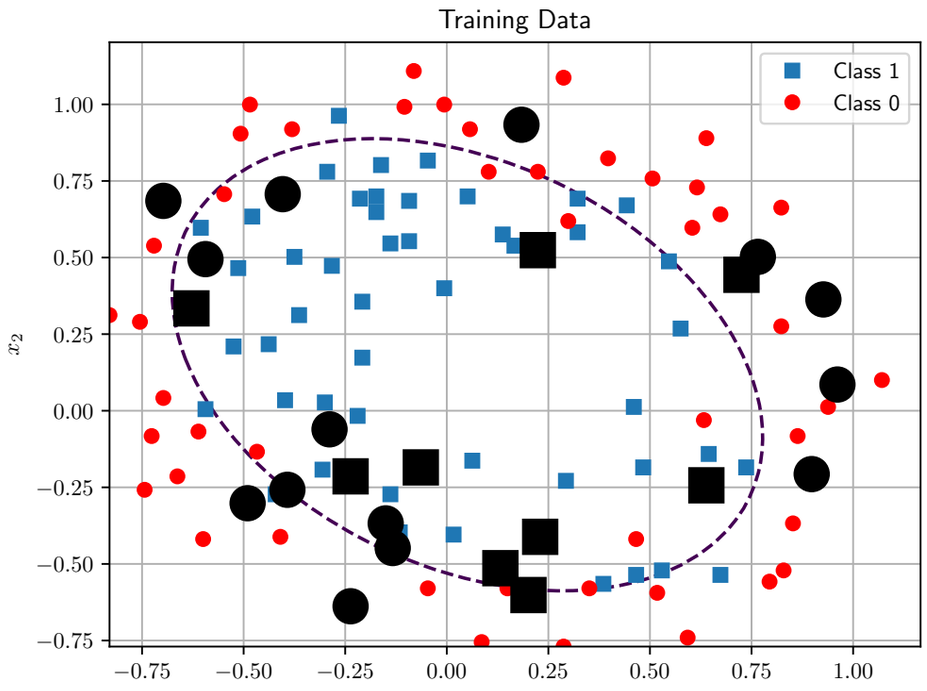
Weight Histogram

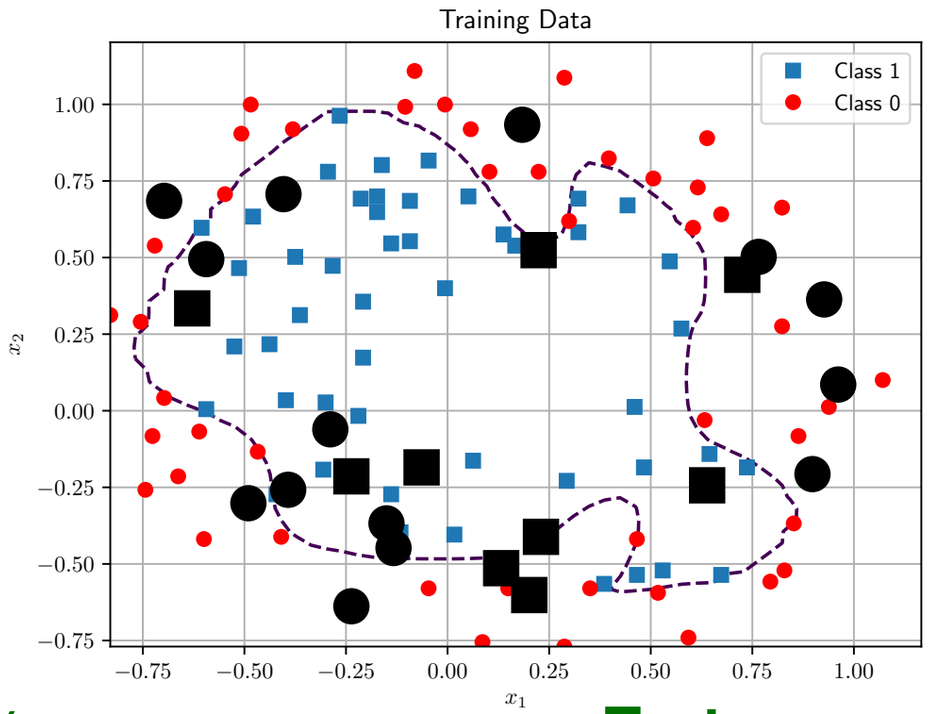


$$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$$

$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

Binary Logistic Regression. Adding new features

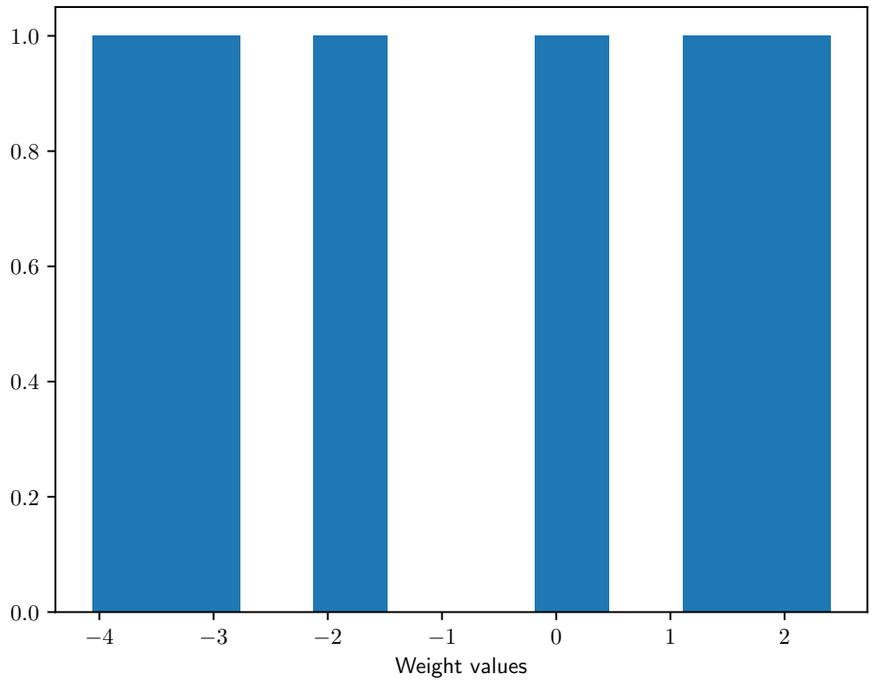




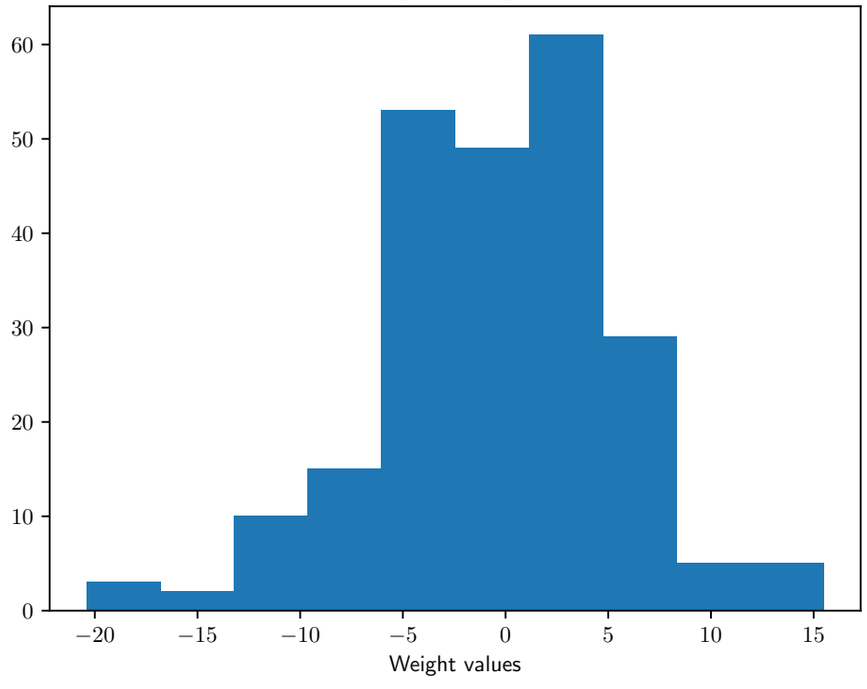
Test accuracy = 65.2 %

Test accuracy = 60.8 %

Weight Histogram



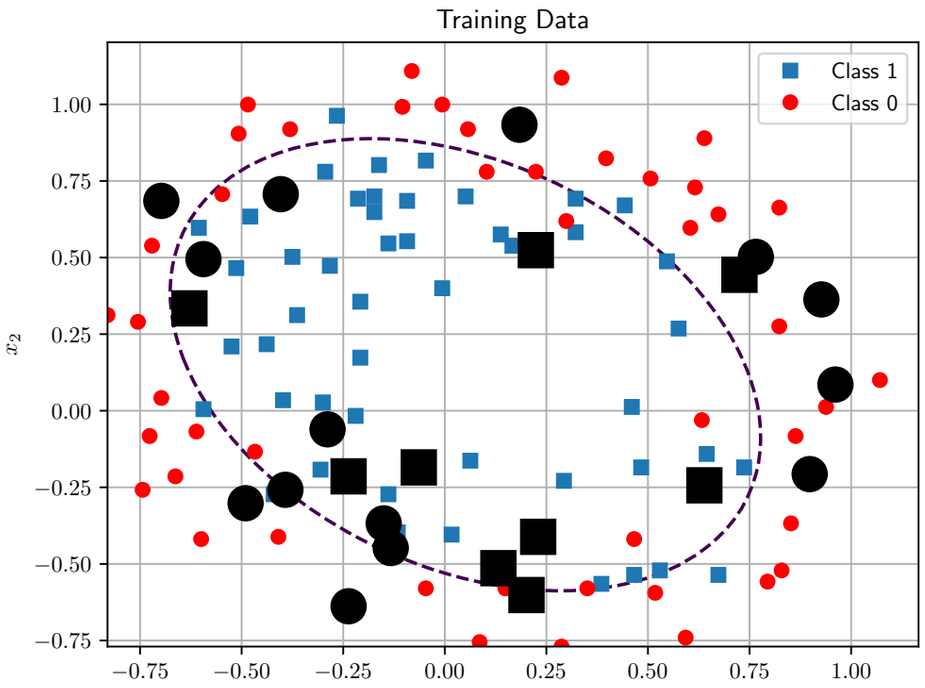
Weight Histogram

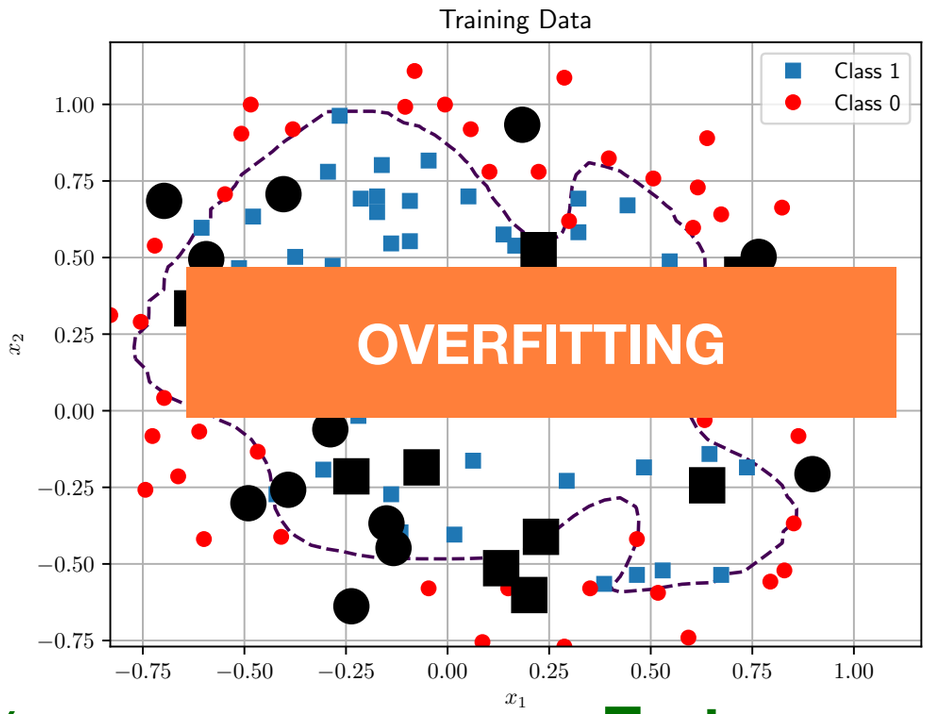


$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$

$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

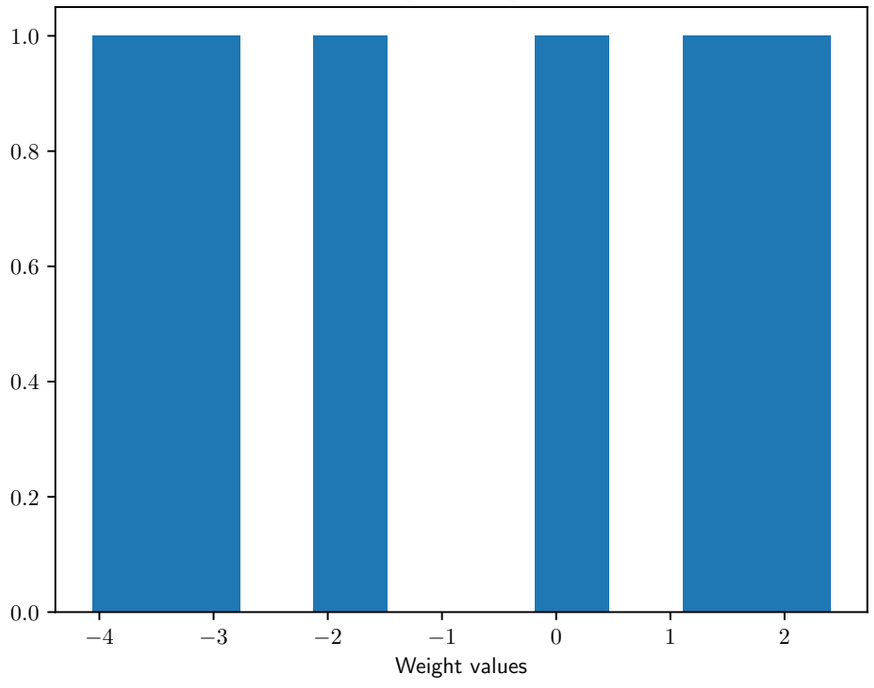
Binary Logistic Regression. Adding new features





Test accuracy = 65.2 %

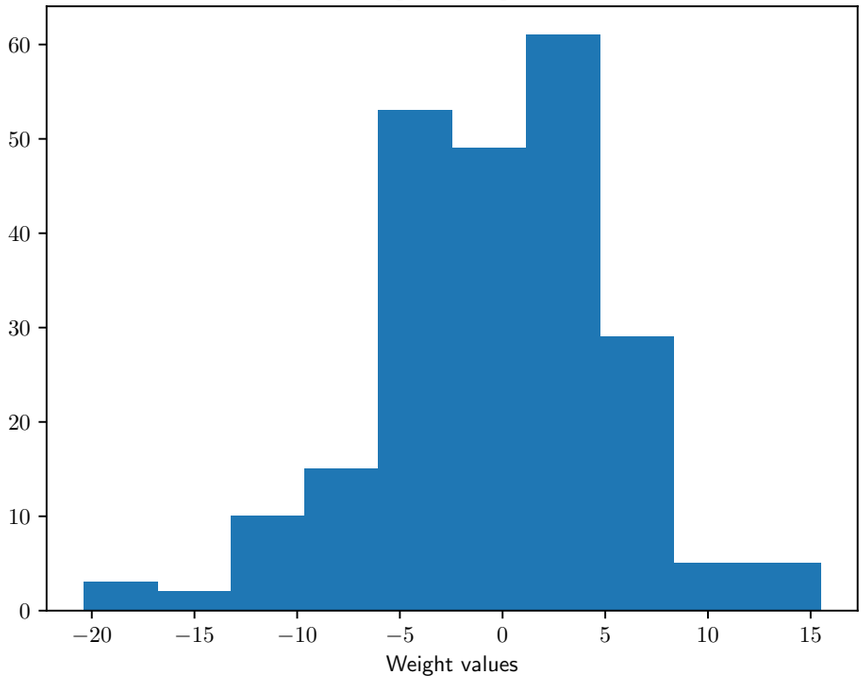
Weight Histogram



$\boldsymbol{\phi}(\boldsymbol{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2]$

Test accuracy = 60.8 %

Weight Histogram



$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

Weight norm regularization

- Large weights are associated to overfitting issues

- Instead of manually reducing the number of features,
  it is easier to pelanize solution for which

$$||\boldsymbol{w}||_2^2 = \sum w_j^2$$ **L2 regularization**

or

$$||\boldsymbol{w}||_1 = \sum |w_j|$$ **L1 regularization**

are very large.

Weight norm regularization

- Large weights are associated to overfitting issues

- Instead of manually reducing the number of features,
  it is easier to pelanize solution for which

$$||\boldsymbol{w}||_2^2 = \sum w_j^2$$ **L2 regularization**

or

$$||\boldsymbol{w}||_1 = \sum |w_j|$$ **L1 regularization**

  are very large.

- Regularized Logistic Regression:

$$\mathcal{L} = -\log P(\boldsymbol{y}|\boldsymbol{X}) + \lambda||\boldsymbol{w}||_1$$

$$\mathcal{L} = -\log P(\boldsymbol{y}|\boldsymbol{X}) + \lambda||\boldsymbol{w}||_2^2$$

Weight norm regularization

- Large weights are associated to overfitting issues

- Instead of manually reducing the number of features,
  it is easier to pelanize solution for which

$$||\boldsymbol{w}||_2^2 = \sum w_j^2$$ **L2 regularization**

or

$$||\boldsymbol{w}||_1 = \sum |w_j|$$ **L1 regularization**

are very large.

- Regularized Logistic Regression:

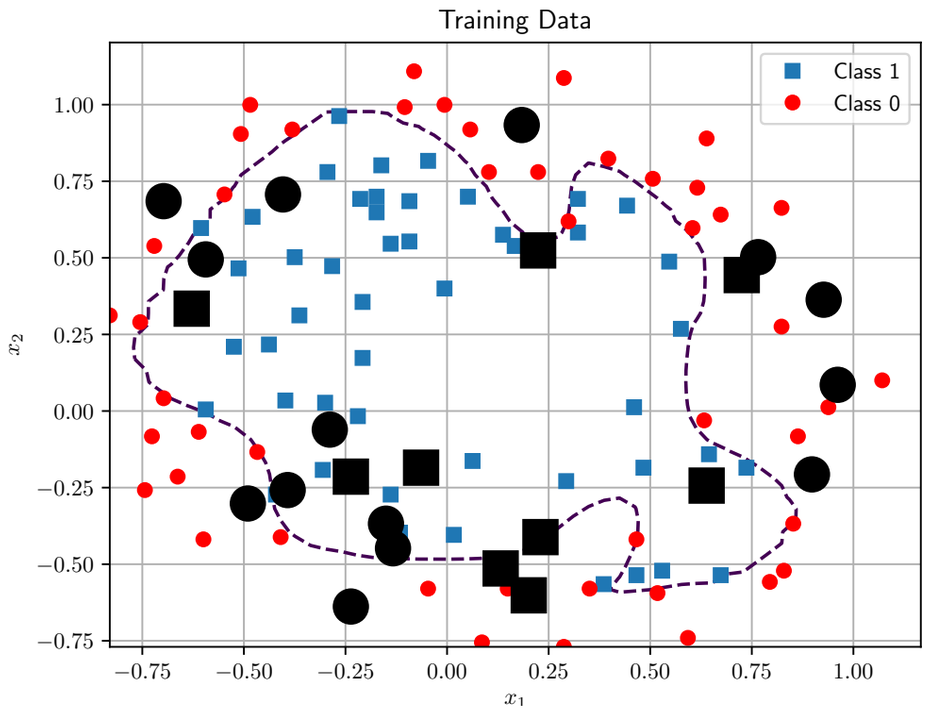
$$\mathcal{L} = -\log P(\boldsymbol{y}|\boldsymbol{X}) + \lambda||\boldsymbol{w}||_1$$

Regularization Parameter

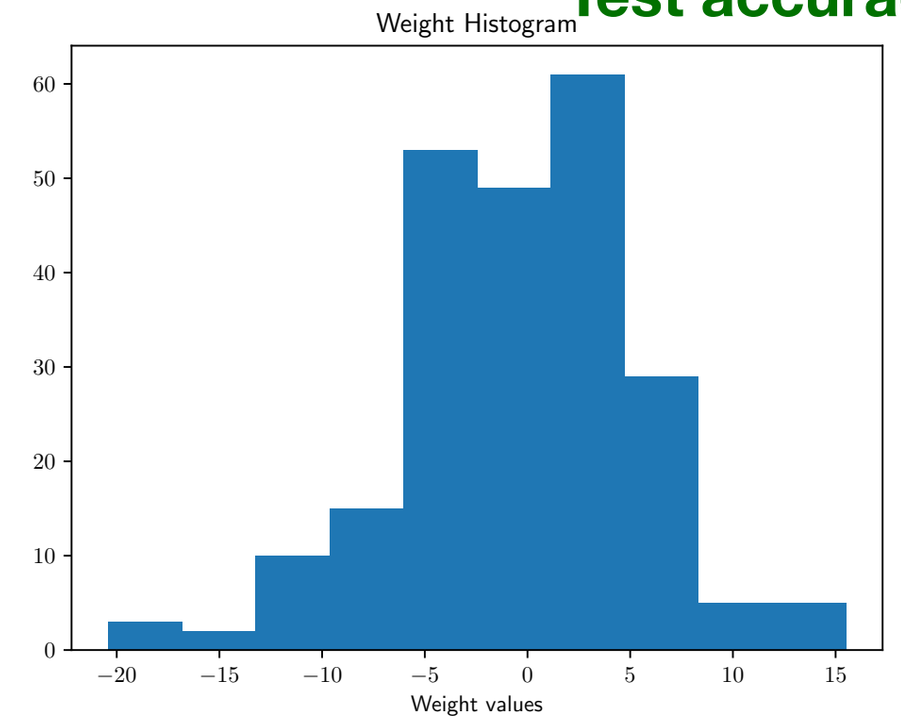
Cross validation!!

$$\mathcal{L} = -\log P(\boldsymbol{y}|\boldsymbol{X}) + \lambda||\boldsymbol{w}||_2^2$$

Weight norm regularization

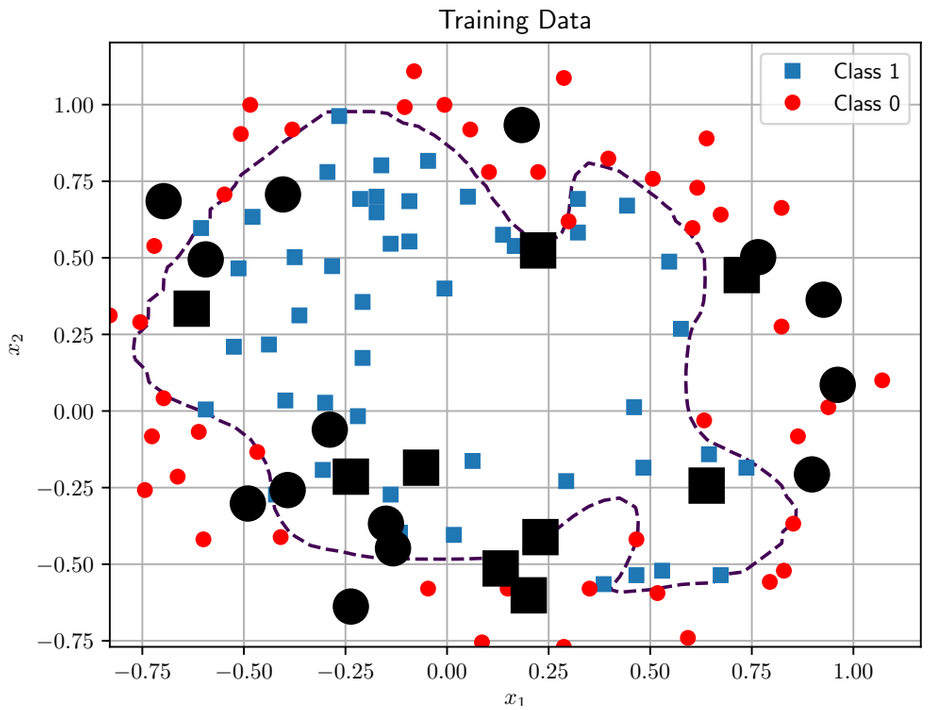


Test accuracy = 60.8 %

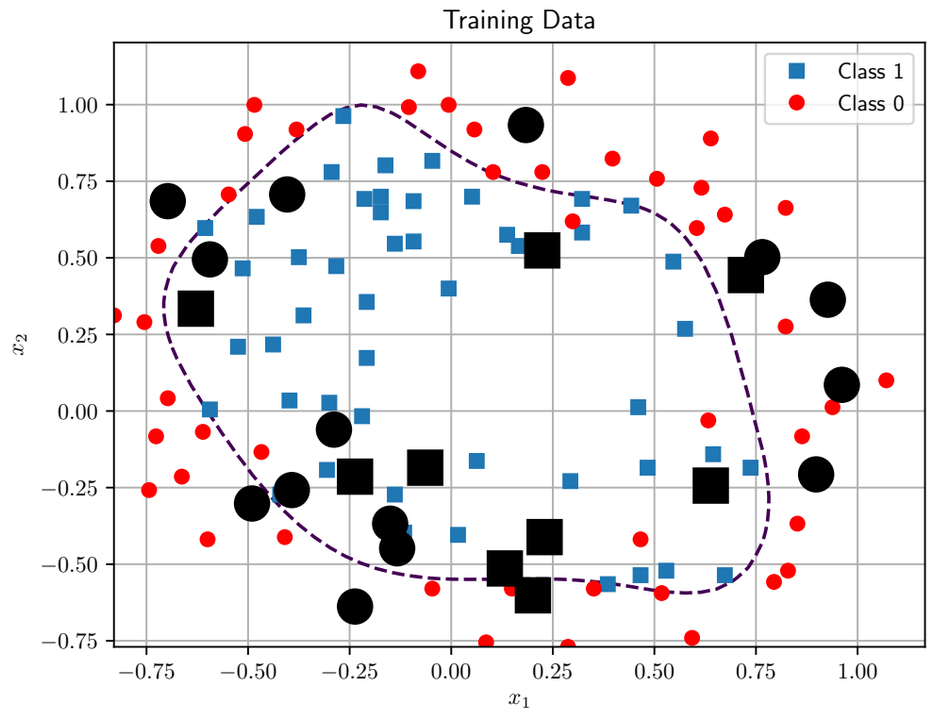


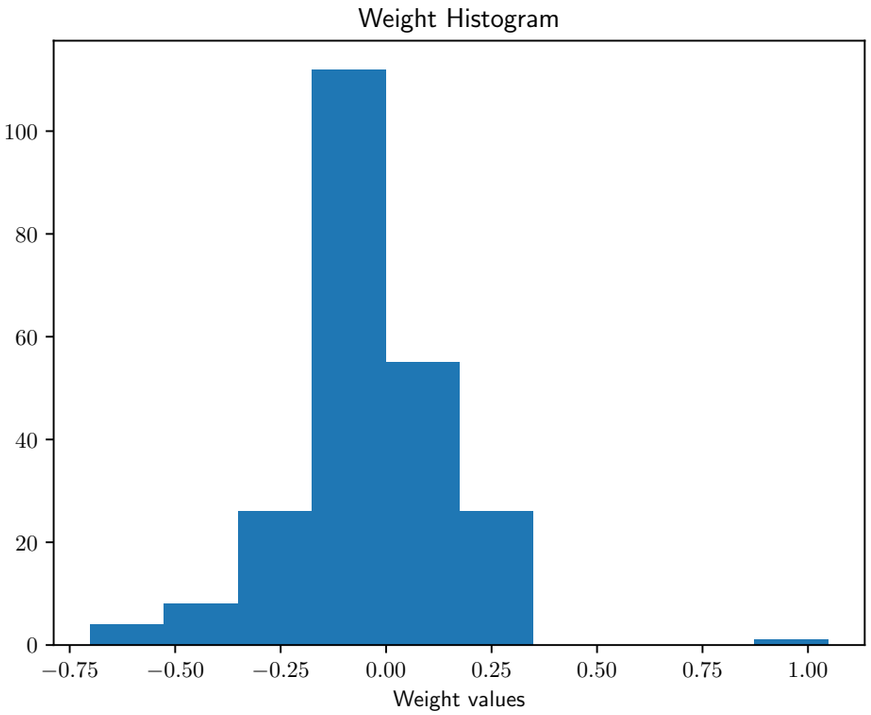
$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

Weight norm regularization



Test accuracy = 60.8 %

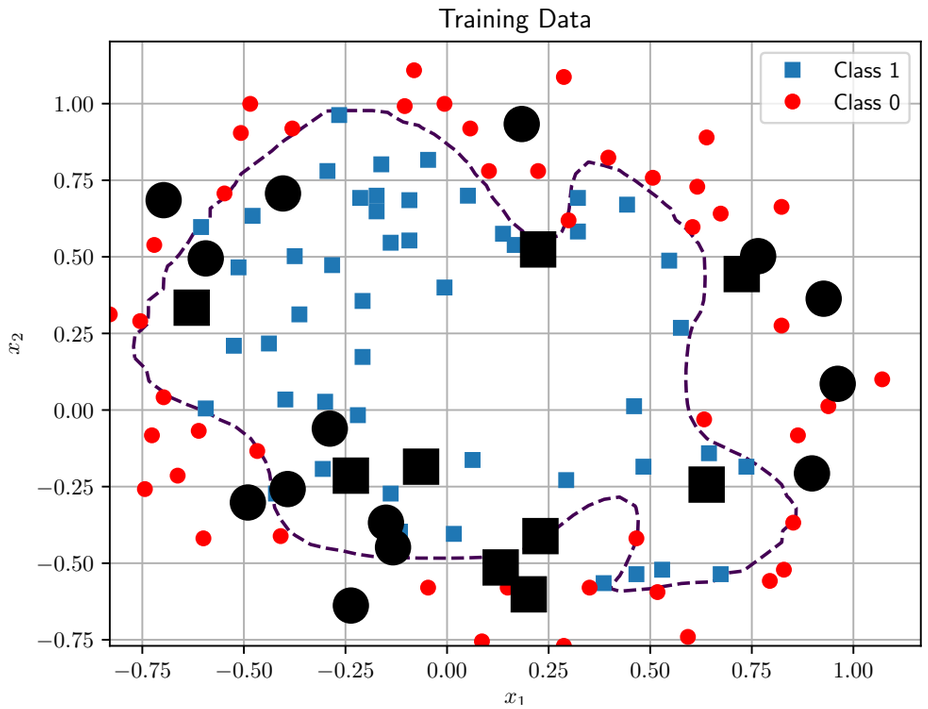




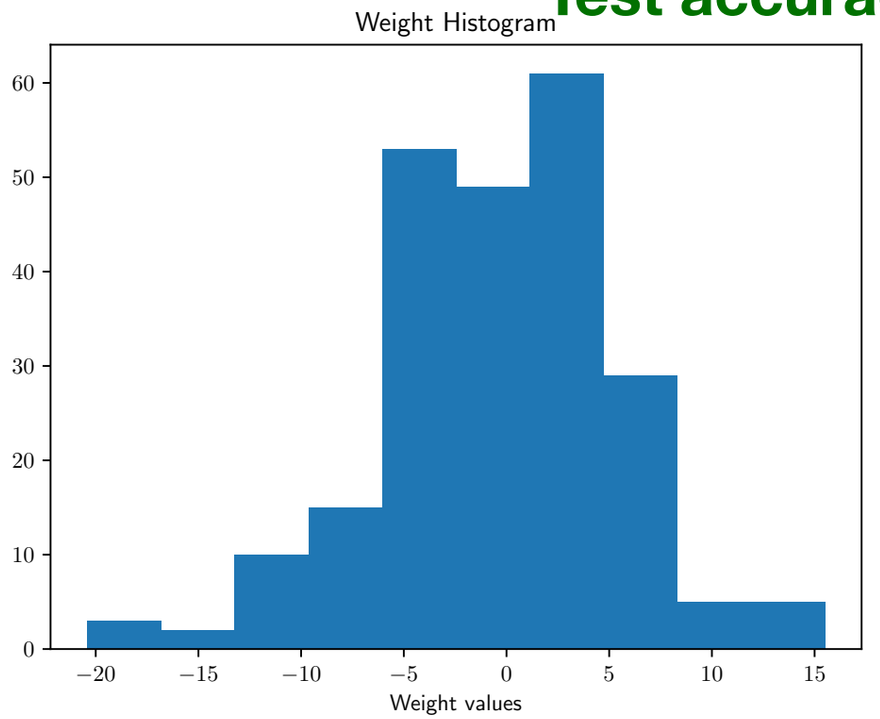
$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

$\lambda = 1.0, \quad L_2$

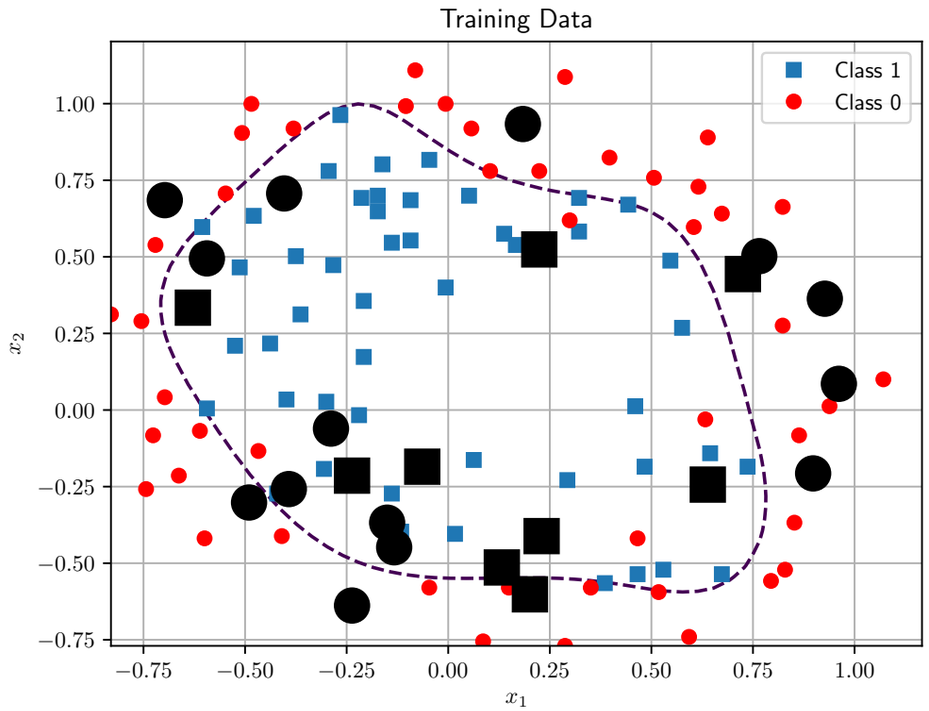
Weight norm regularization



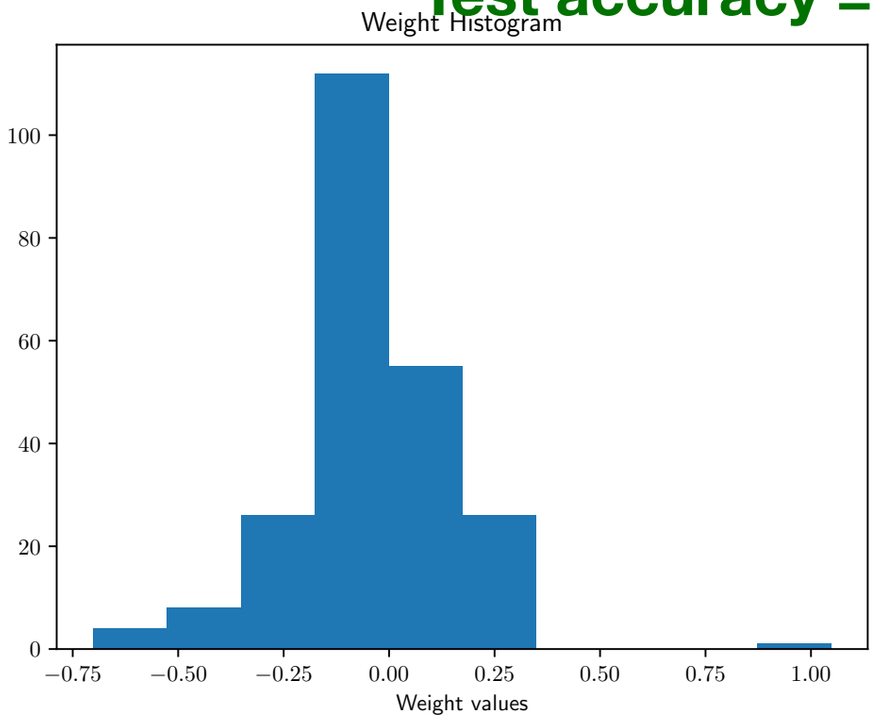
Test accuracy = 60.8 %



$\boldsymbol{\phi}(\boldsymbol{x})$ All polynomial terms up to degree 6

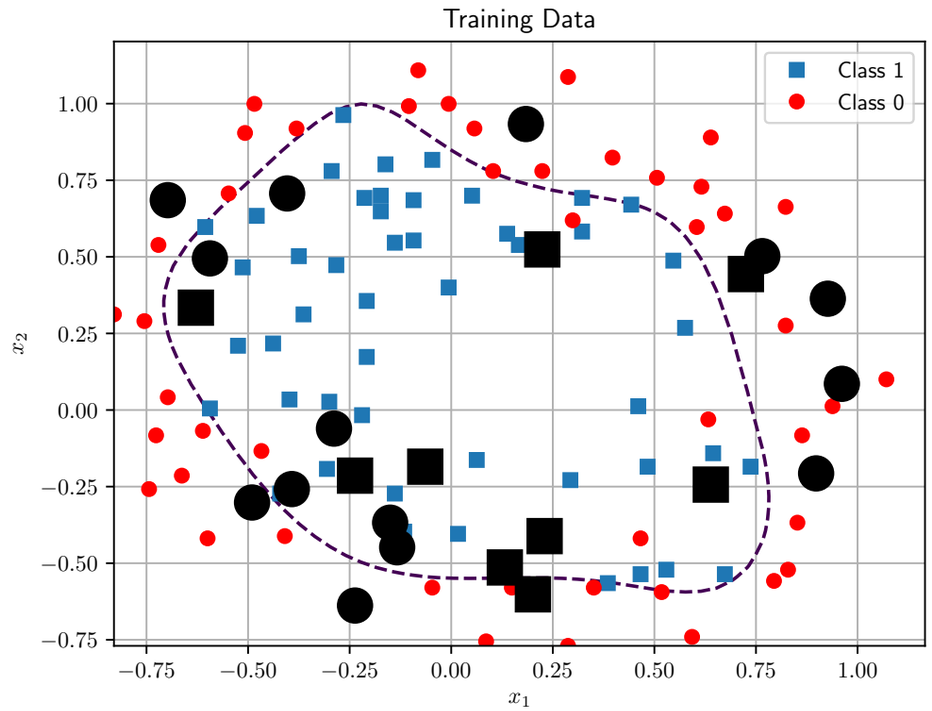


Test accuracy = 65.2 %

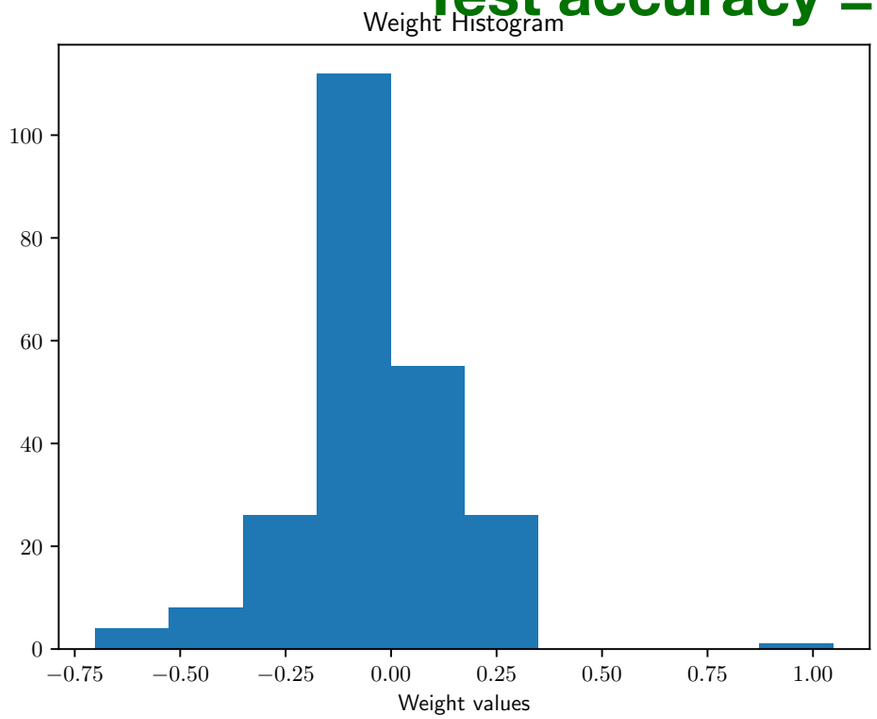


$\lambda = 1.0, \quad L_2$

Weight norm regularization

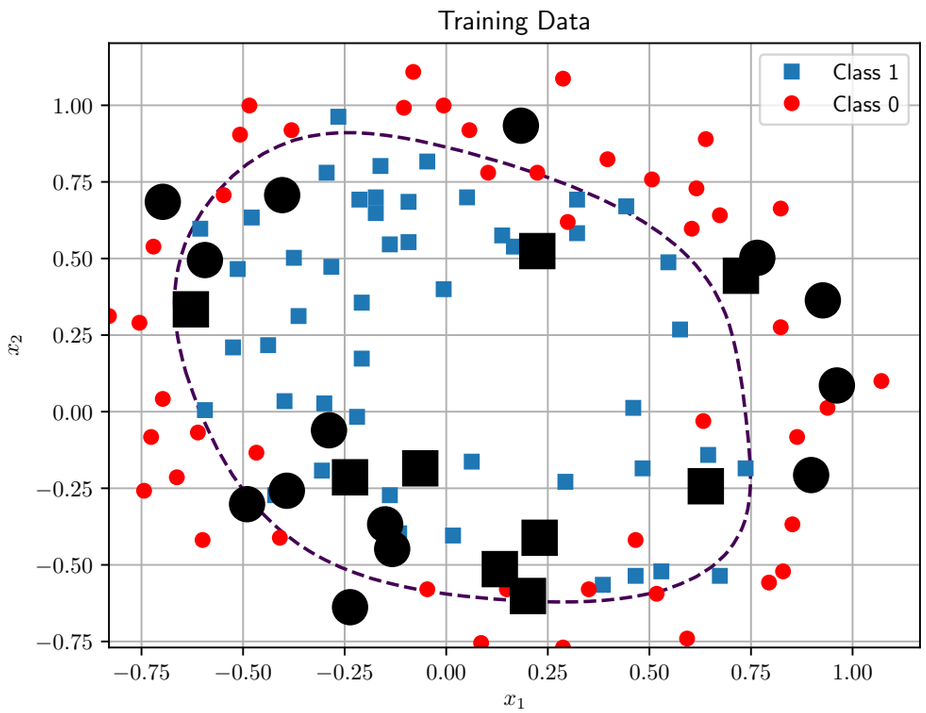


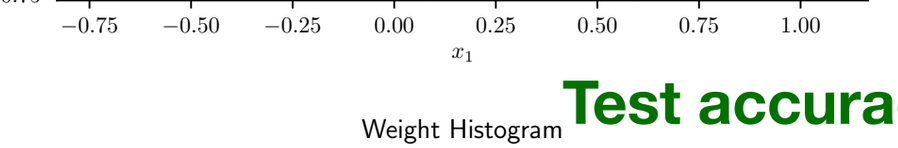
Test accuracy = 65.2 %



$$\lambda = 1.0, \quad L_2$$

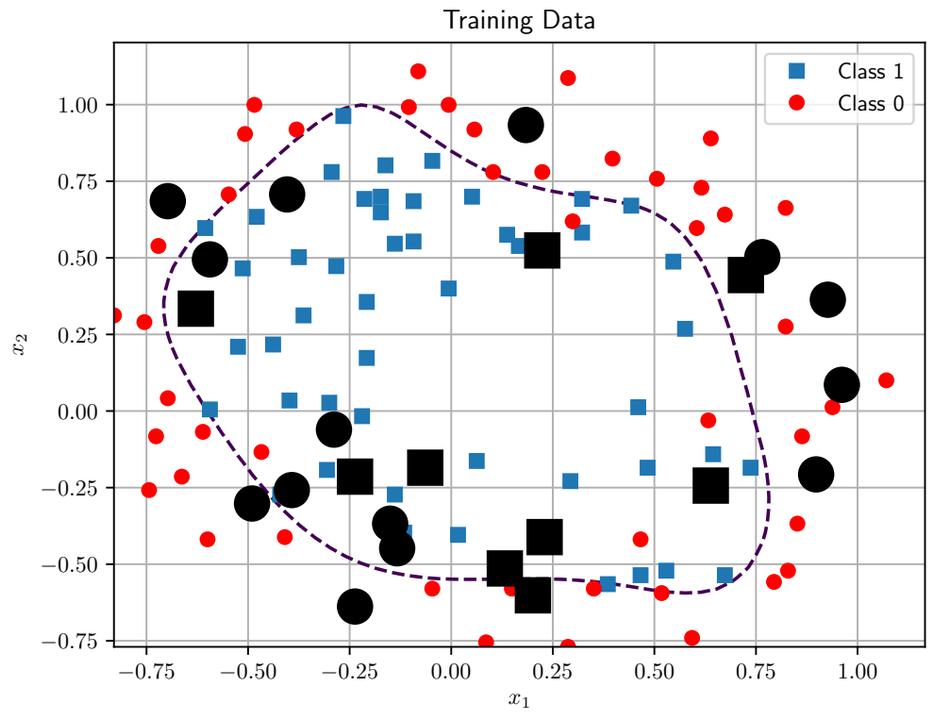
Weight norm regularization

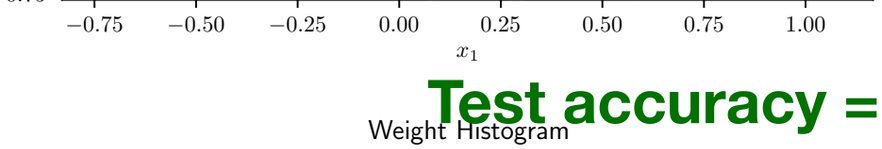




Test accuracy = 69.5 %

$\lambda = 1.0, \quad L_1$





Test accuracy = 65.2 %

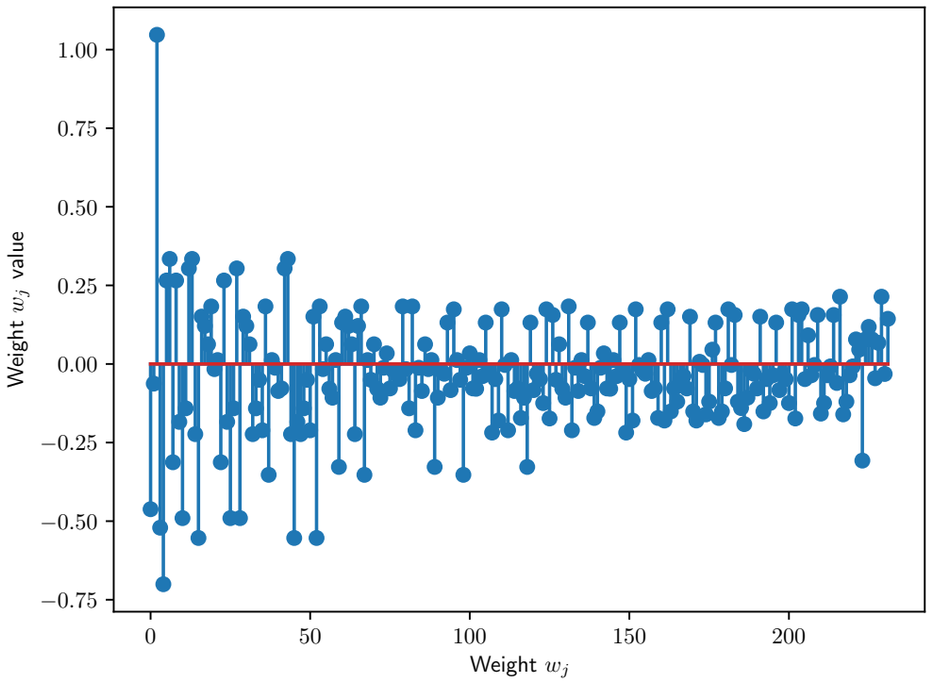
$\lambda = 1.0, \quad L_2$

L1 norm regularization induces sparsity

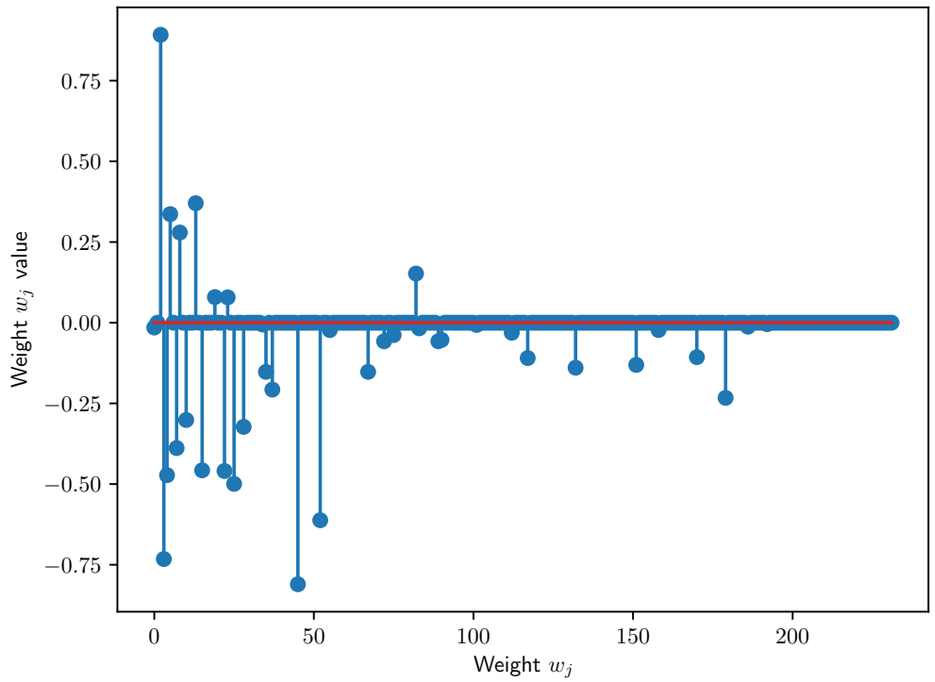
- With L1 regularization, many weights go to zero. Good for feature selection!
- Numerical optimization with L1-type of regularizers is more challenging

Check out in Aula Global this explanation about sparsity with L1 (by Emily Fox):

Link



$\lambda = 1.0, \quad L_2$



$\lambda = 1.0, \quad L_1$

Logistic Regression: multiple classes

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

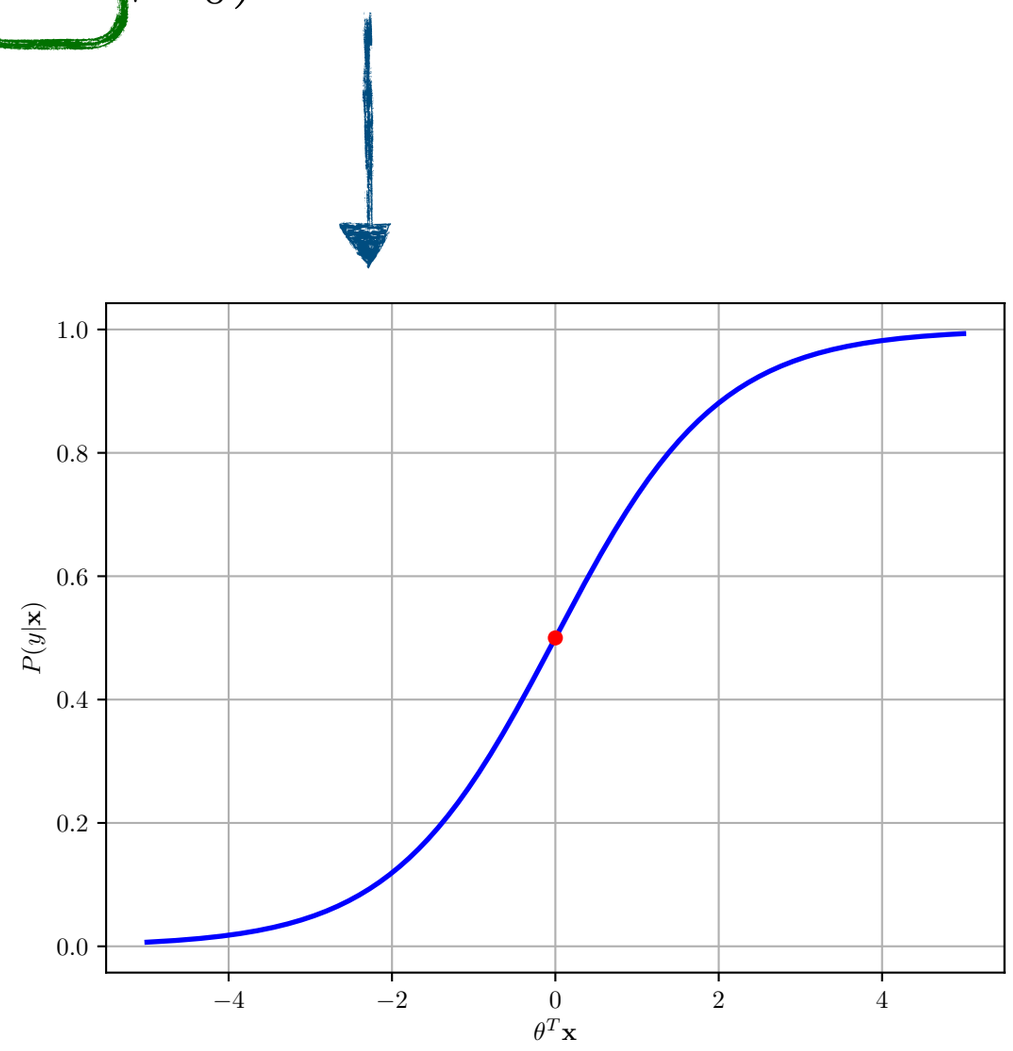
Generalized linear model

Binary Logistic Regression

- Discriminative classifier:

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Generalized linear model



Binary Logistic Regression

- Discriminative classifier:

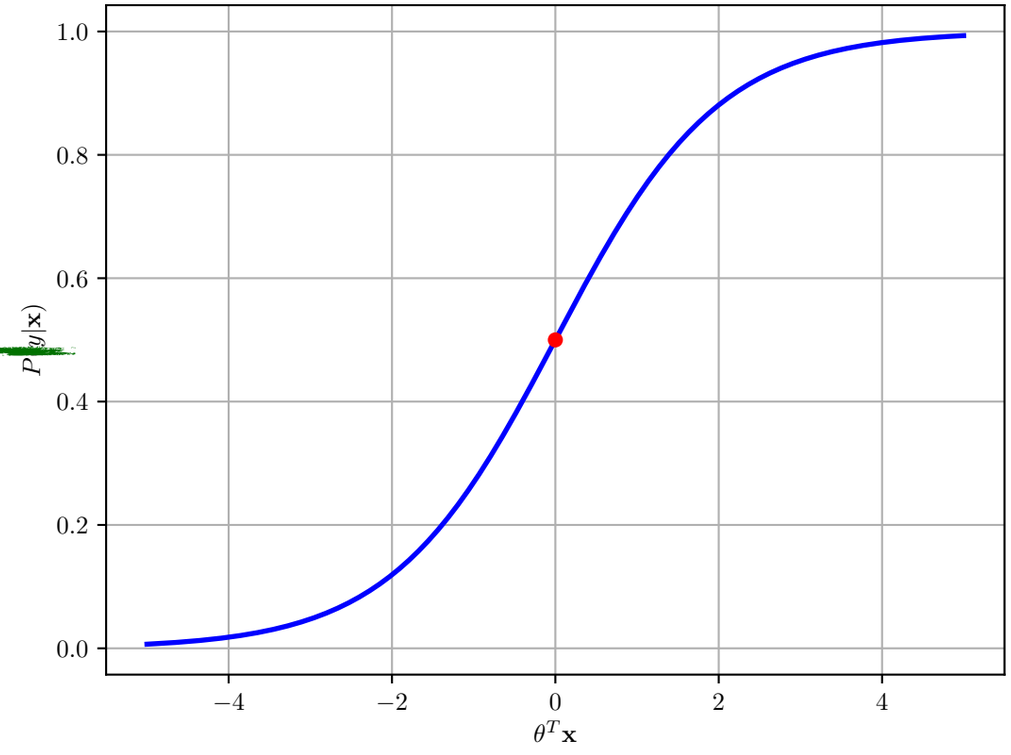
$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^T\boldsymbol{x}+w_0)}} \doteq \sigma(\boldsymbol{w}^T\boldsymbol{x}+w_0)$$

Generalized linear model

- Decision boundary:

$$\{\boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{w}^T\boldsymbol{x} + w_0 = \mathbf{0}\}$$

Hyperplane! Linear classifier



Multiclass Logistic Regression

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{N} \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{1, \ldots, K\}$

- Discriminative classifier based on the **Softmax function**:

$$P(y = k|\boldsymbol{x}) = \frac{\mathrm{e}^{-(z_k)}}{\sum_{j=1}^{K} \mathrm{e}^{-(z_j)}}, \quad \boldsymbol{z} = \boldsymbol{W} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

- Similar gradient expression

Multiclass Logistic Regression

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{N} \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{1, \ldots, K\}$

- Discriminative classifier based on the **Softmax function**:

$$P(y = k|\boldsymbol{x}) = \frac{\mathrm{e}^{-(z_k)}}{\sum_{j=1}^{K} \mathrm{e}^{-(z_j)}}, \quad \boldsymbol{z} = \boldsymbol{W} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

$$K \times (m+1)$$

- Similar gradient expression

Multiclass Logistic Regression

- Training database: $\mathcal{D} \doteq (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{N} \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^m \qquad y^{(i)} \in \{1, \ldots, K\}$

- Discriminative classifier based on the **Softmax function**:

$$P(y = k|\boldsymbol{x}) = \frac{\mathrm{e}^{-(z_k)}}{\sum_{j=1}^{K} \mathrm{e}^{-(z_j)}}, \quad \boldsymbol{z} = \boldsymbol{W} \begin{bmatrix} 1.0 \\ \boldsymbol{x}^{(i)} \end{bmatrix}$$

$$K \times (m + 1)$$

- Loss function (Cross entropy):

$$\mathcal{L} = -\log P(\boldsymbol{y}|\boldsymbol{X}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}[y^{(n)} == k] \log P(y = k|\boldsymbol{x})$$

- Similar gradient expression