Gradient-Based Optimization

Elena Congeduti, 14-11-2024

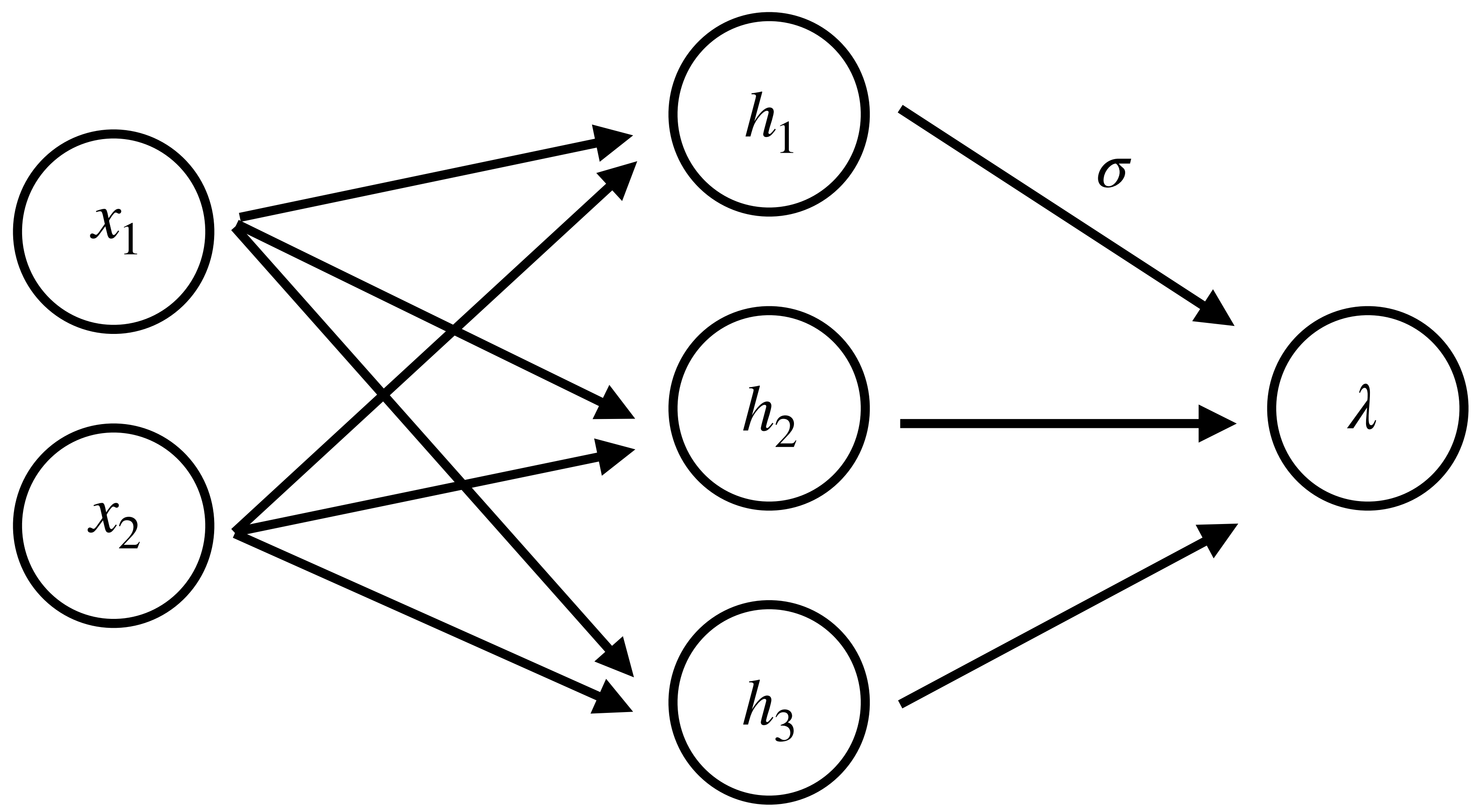
TU Delft

Lecture's Agenda

1. Loss Functions
2. Maximum Likelihood
3. Gradient Descent
4. Backpropagation

Training loop for classification

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs



Iterate until convergence

1. Forward pass: for batch x compute output $\hat{\lambda} = f(x; \theta)$
2. **Evaluate: compare the $\hat{\lambda}$ with the class label y**
3. Backward pass: update the parameters θ

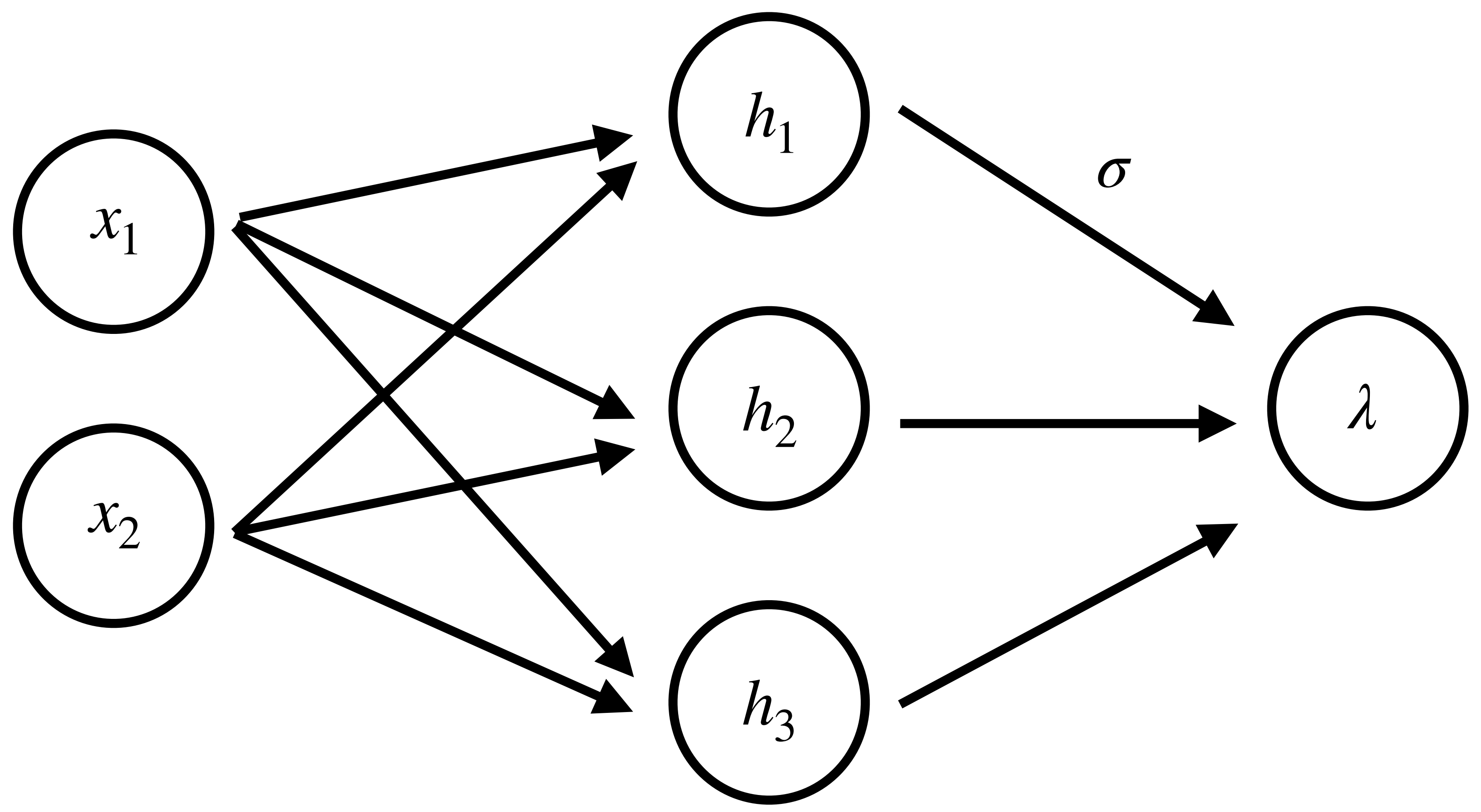
Scalar loss function $L(\theta)$

Training loop for classification

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

Iterate until convergence

1. Forward pass: for batch x compute output $\hat{\lambda} = f(x; \theta)$
2. **Evaluate: compare the $\hat{\lambda}$ with the class label y**
3. Backward pass: update the parameters θ



Scalar loss function $L(\theta)$

- Error between ground truth y and network output $\hat{\lambda}$
- Suitable for gradient learning ('usable' gradient)

Binary Classification Loss

Which is a good candidate loss when the last activation returns $\hat{\lambda} = \sigma(\ldots)$?

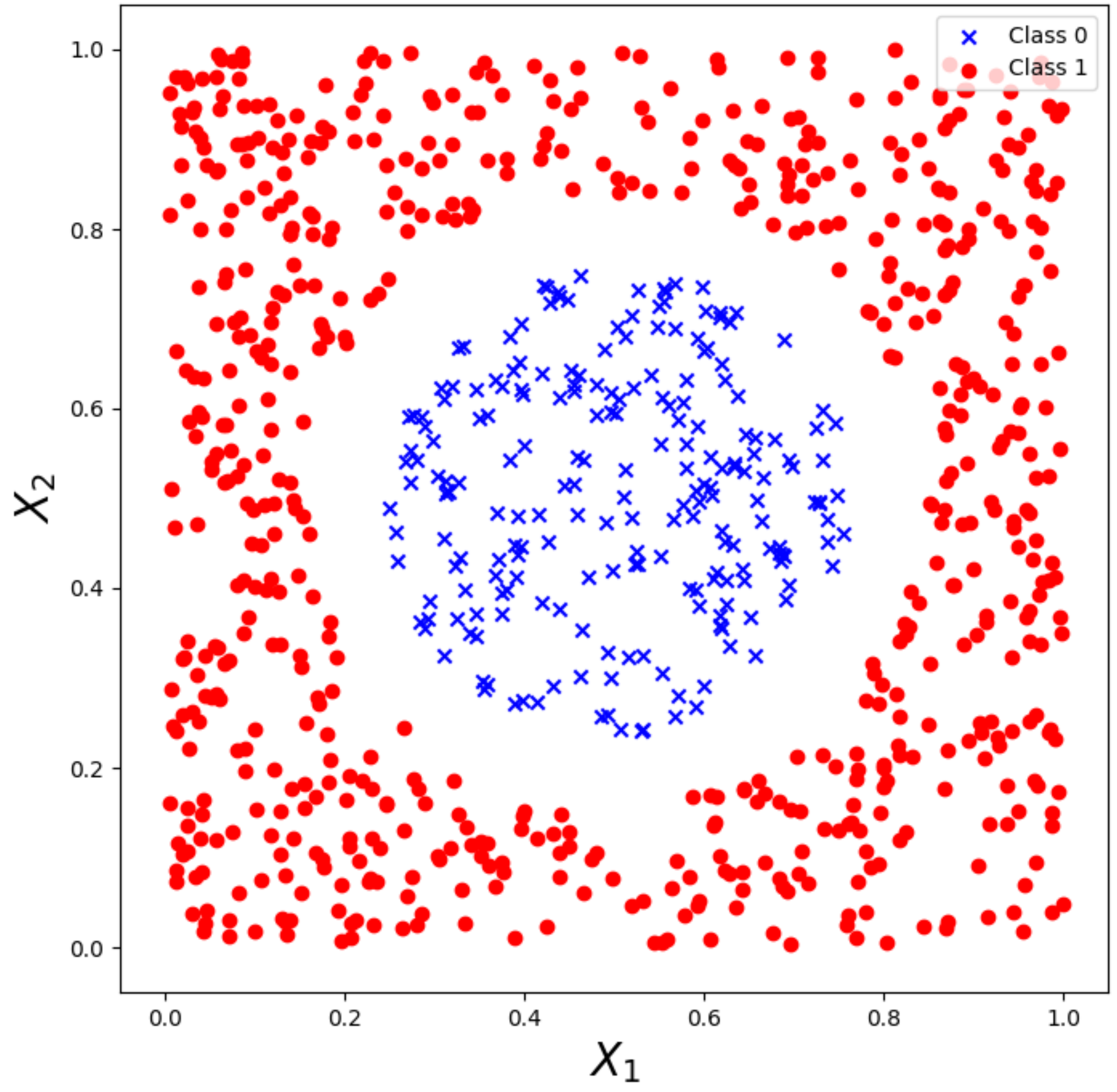
A. $L_{ratio} = \dfrac{y}{\hat{\lambda}}$

B. $L_{0-1} = \begin{cases} 0 & \text{if } y = \hat{\lambda} \\ 1 & \text{if } y \neq \hat{\lambda} \end{cases}$

C. $L_{SE} = \dfrac{1}{2}(y - \hat{\lambda})^2$

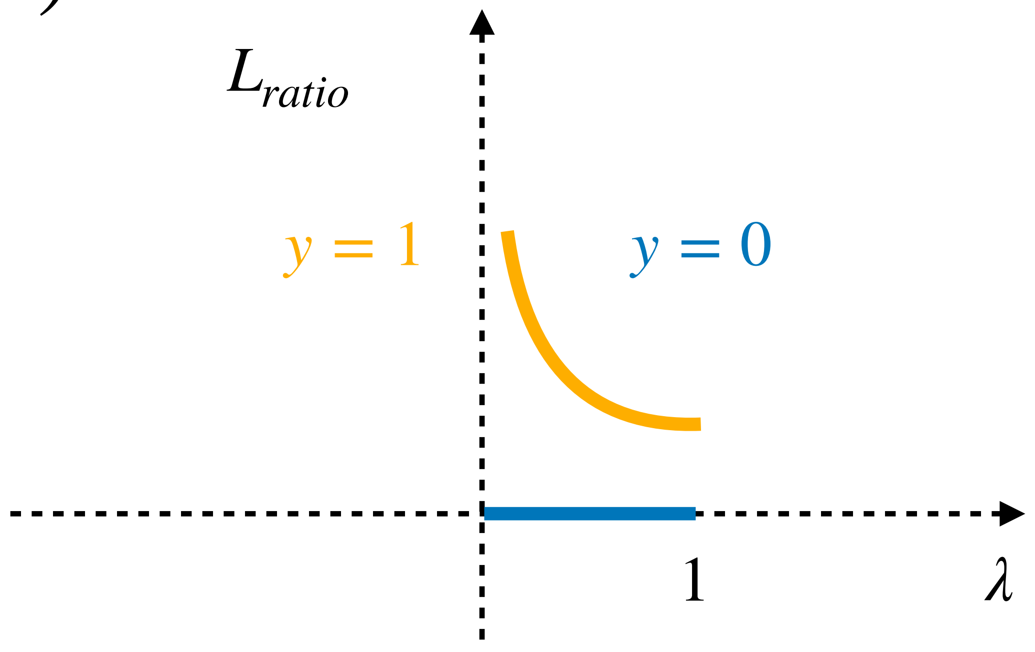
D. $L_1 = |y - \hat{\lambda}|$

E. $L_{CE} = -y\log(\hat{\lambda}) - (1-y)\log(1-\hat{\lambda})$



Binary Classification Loss

Which is a good candidate loss when the last activation returns $\hat{\lambda} = \sigma(\ldots)$?

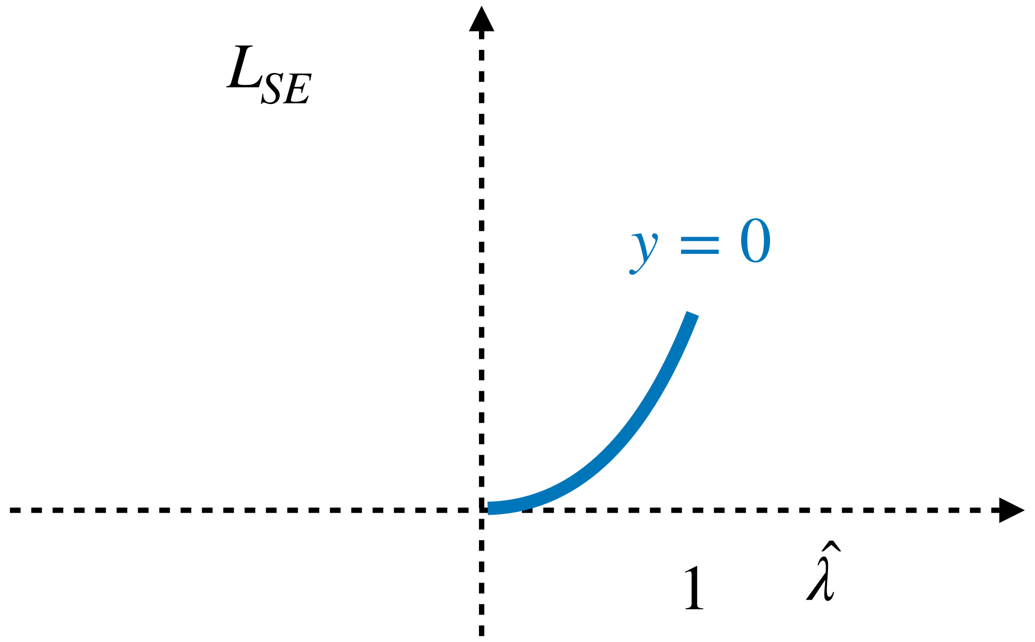


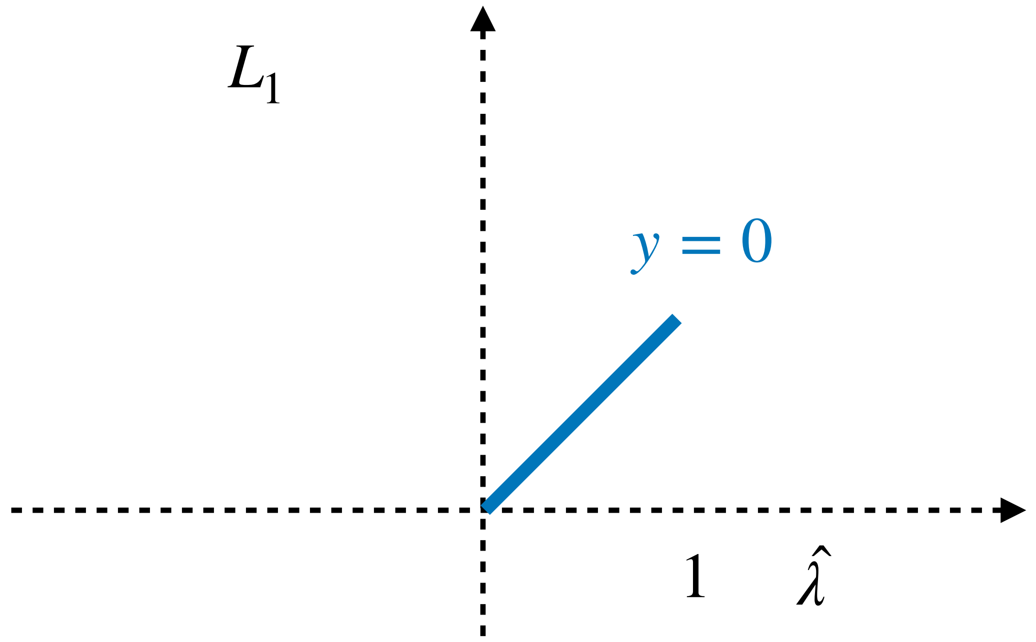
A. $\cancel{L_{ratio} = \frac{y}{\hat{\lambda}}}$ → No error measure for $y = 0$

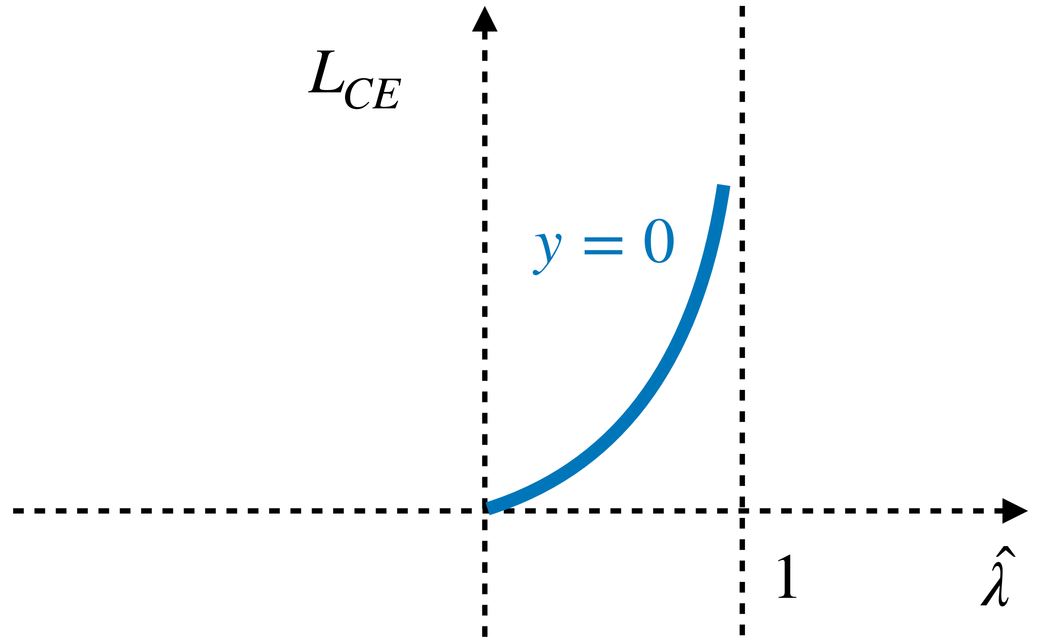
B. $\cancel{L_{0-1} = \begin{cases} 0 & \text{if } y = \hat{\lambda} \\ 1 & \text{if } y \neq \hat{\lambda} \end{cases}}$ → always 0

C. $L_{SE} = \frac{1}{2}(y - \hat{\lambda})^2$

D. $L_1 = |y - \hat{\lambda}|$



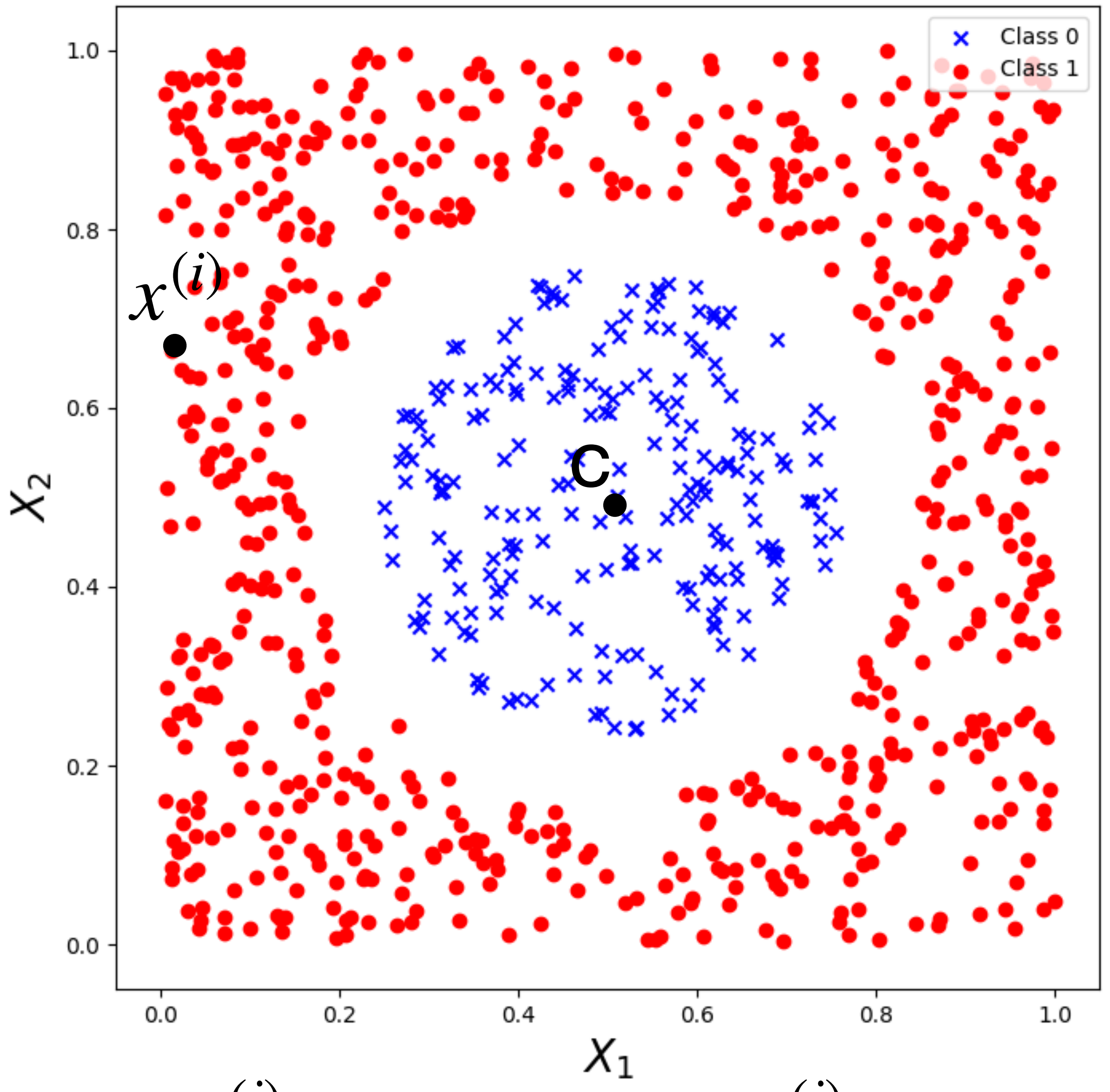




E. $L_{CE} = -y\log(\hat{\lambda}) - (1 - y)\log(1 - \hat{\lambda})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

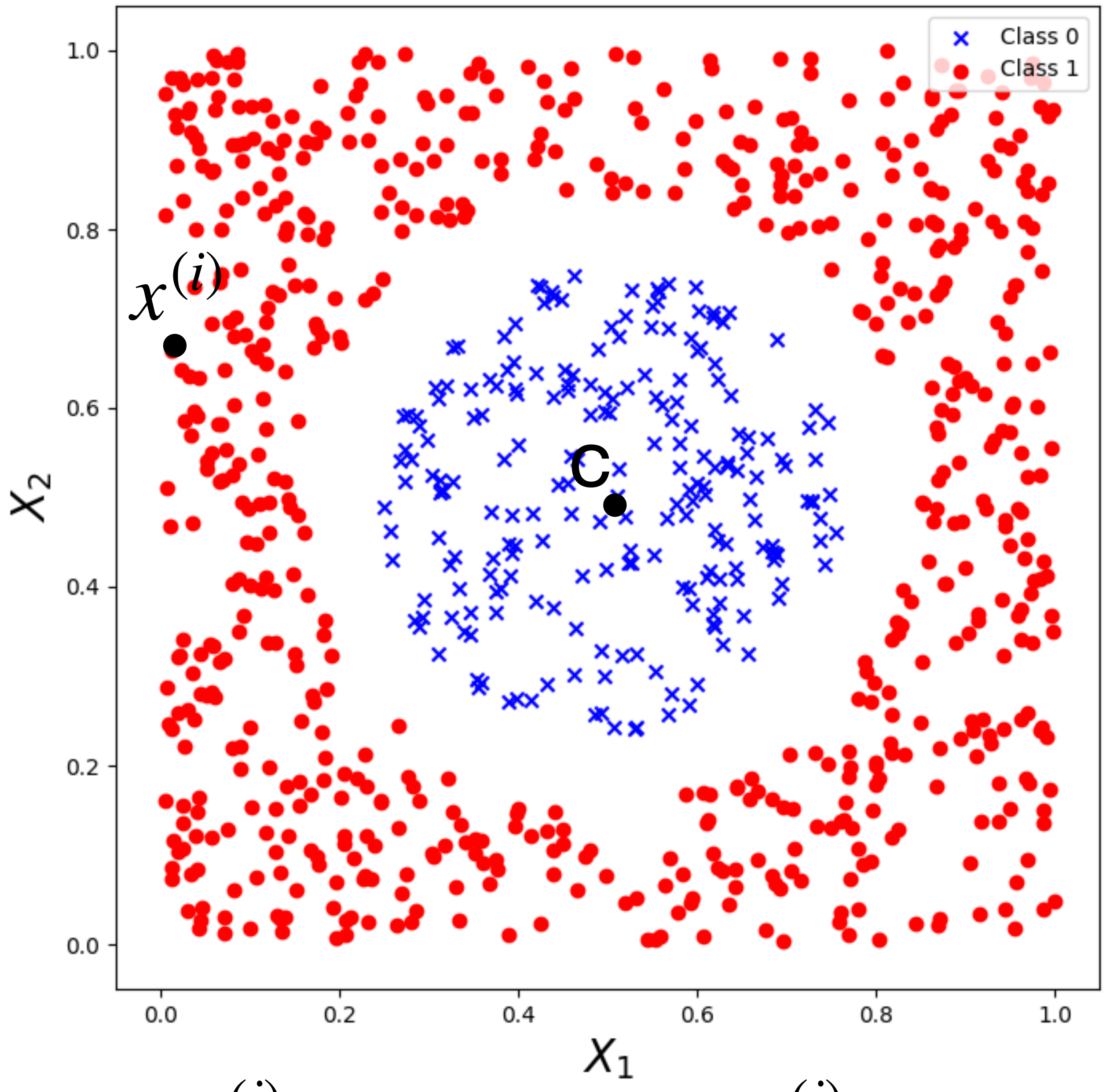


$x^{(i)} = (x_1, x_2)$ $\quad y^{(i)} \in \{0,1\}$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$



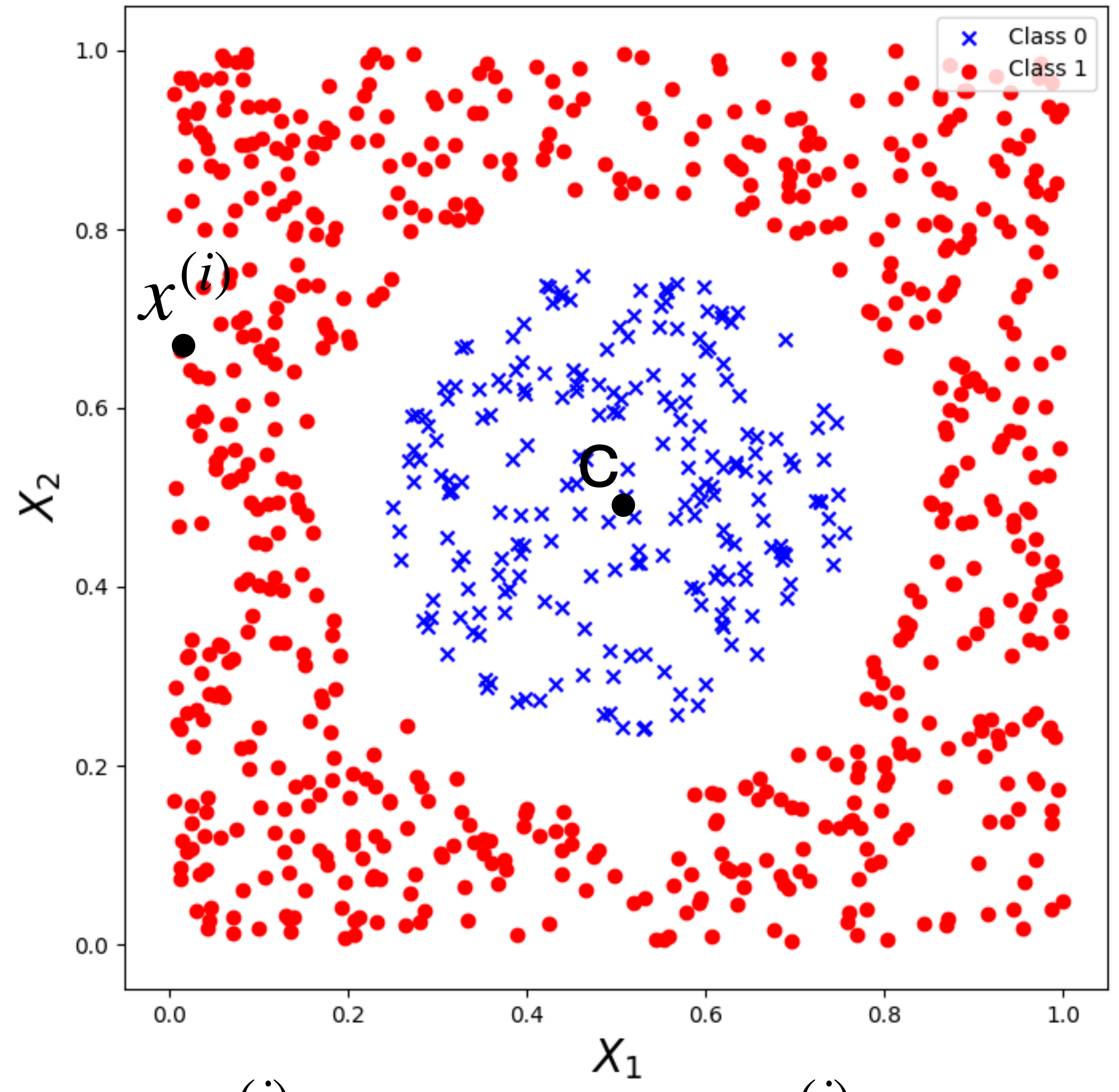
$x^{(i)} = (x_1, x_2) \qquad y^{(i)} \in \{0,1\}$

$$p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$



$x^{(i)} = (x_1, x_2)$ $\quad$ $y^{(i)} \in \{0,1\}$

$$p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

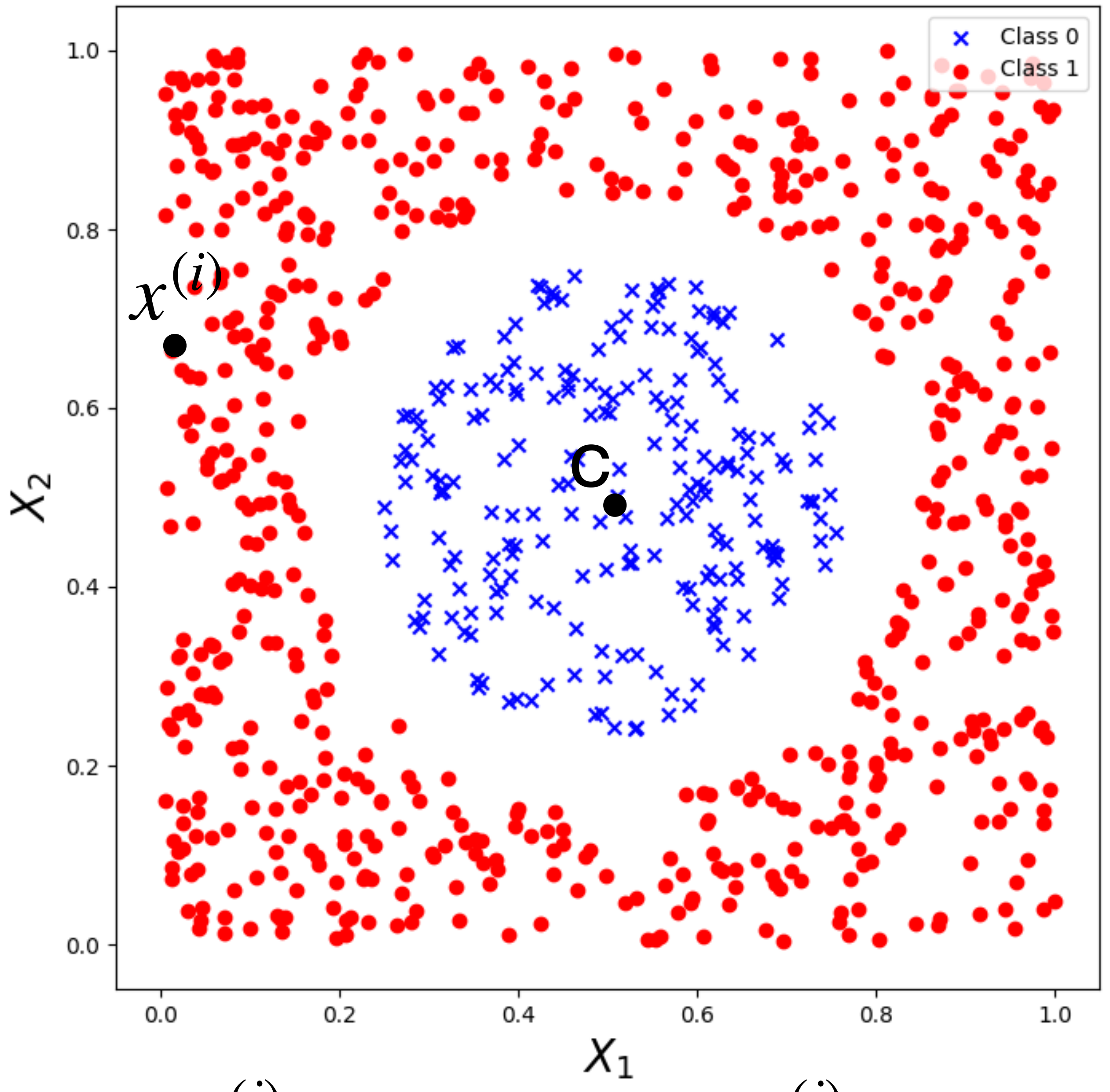
Deterministic distribution $\quad P(y=0) = 1$, $P(y=1) = 0$

Euclidean distance

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$



$x^{(i)} = (x_1, x_2) \qquad y^{(i)} \in \{0,1\}$

$$p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

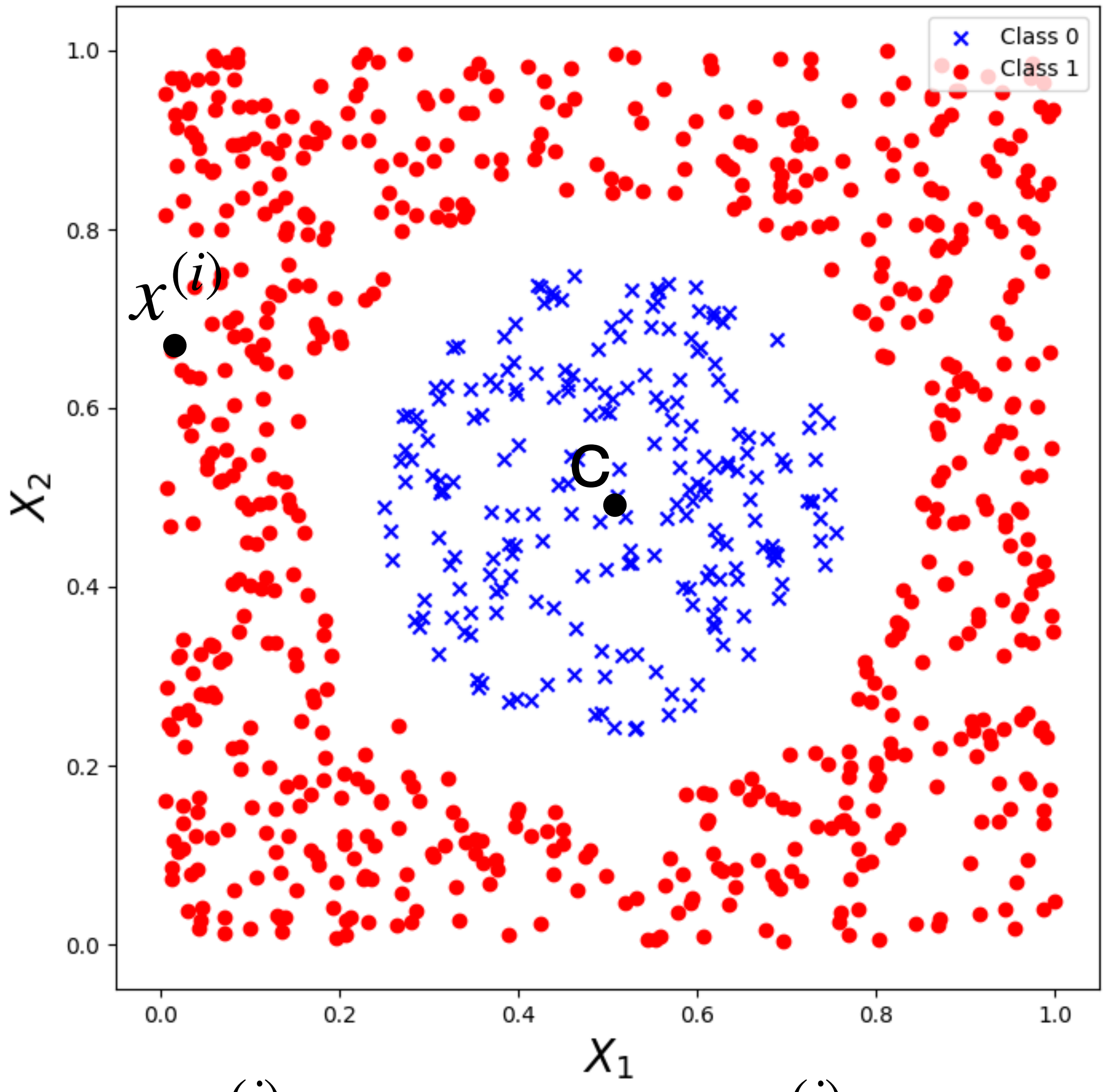
$\{x^{(1)}, \ldots, x^{(n)}\}$, classes $y^{(i)} \sim p_{data}(\cdot \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$

2. Learn an approximation p_{model} of the data distribution



$x^{(i)} = (x_1, x_2)$ $\quad y^{(i)} \in \{0,1\}$

$$p_{model} \approx \; p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

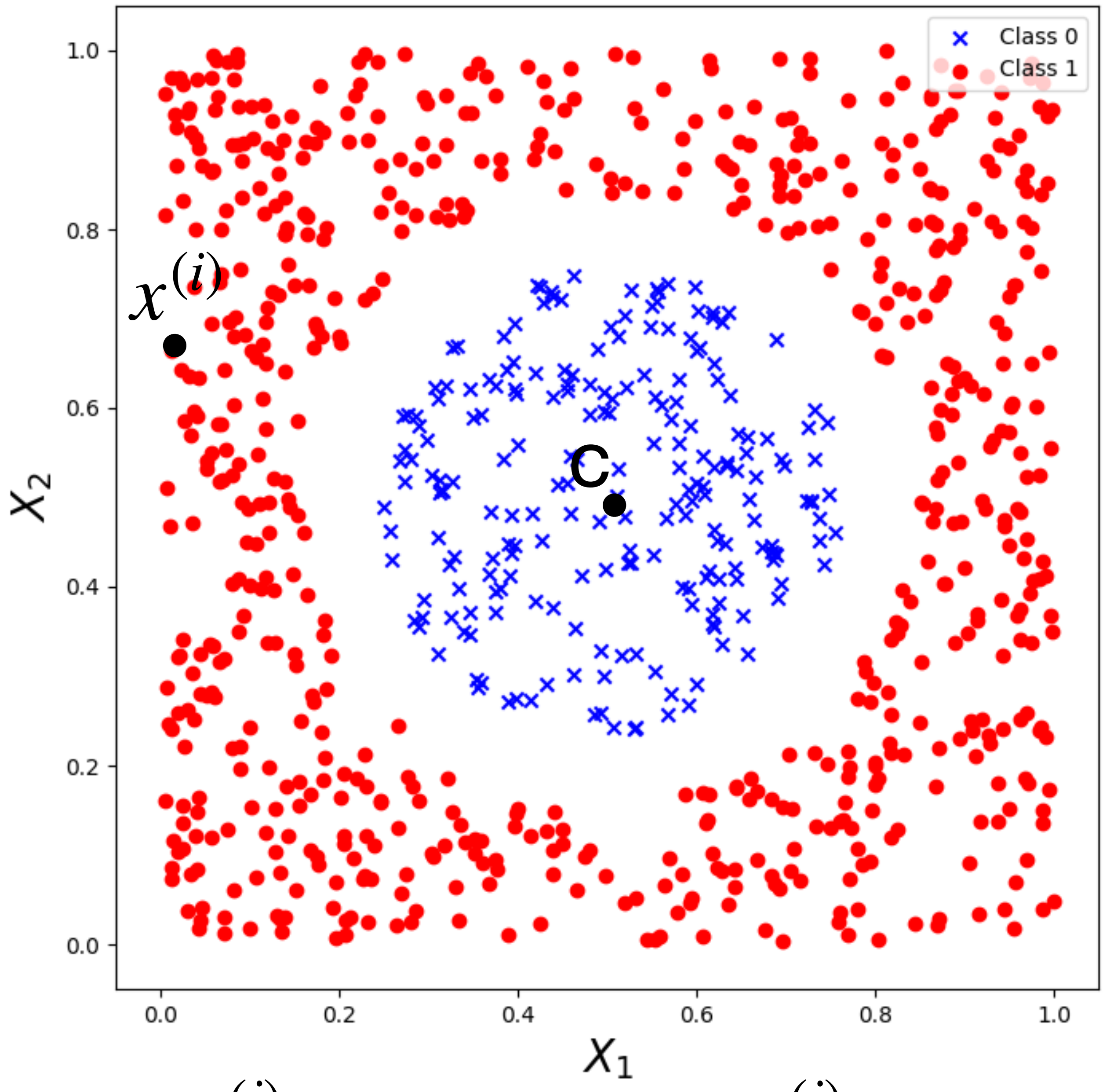
Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$

2. Learn an approximation p_{model} of the data distribution

3. Distribution $p_{model}(y \mid \lambda)$ over the output space



$x^{(i)} = (x_1, x_2) \qquad y^{(i)} \in \{0,1\}$

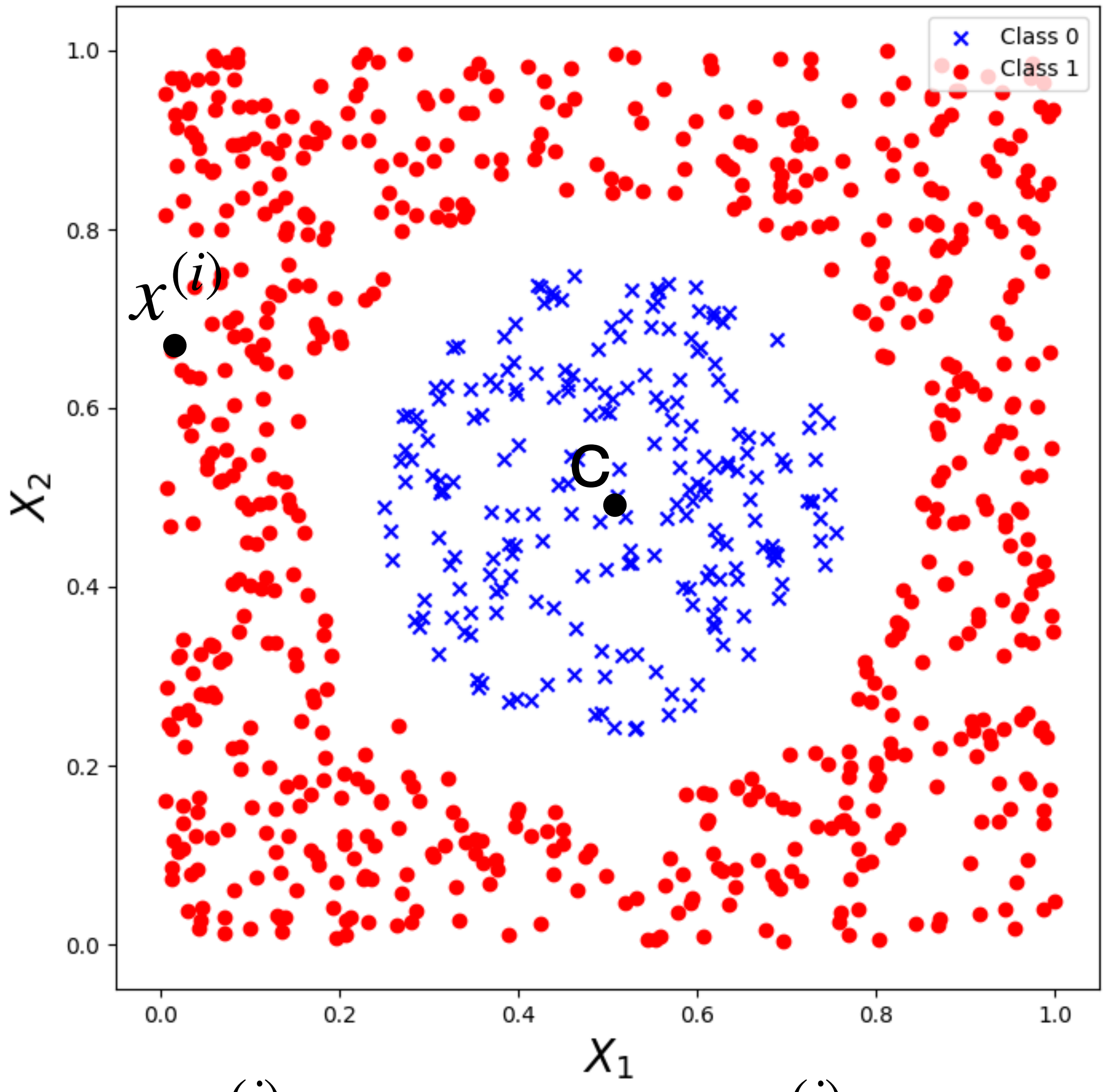
$$p_{model} \approx \; p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$
2. Learn an approximation p_{model} of the data distribution
3. Distribution $p_{model}(y \mid \lambda)$ over the output space



$x^{(i)} = (x_1, x_2) \qquad y^{(i)} \in \{0,1\}$

$$p_{model} \approx \; p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \le r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

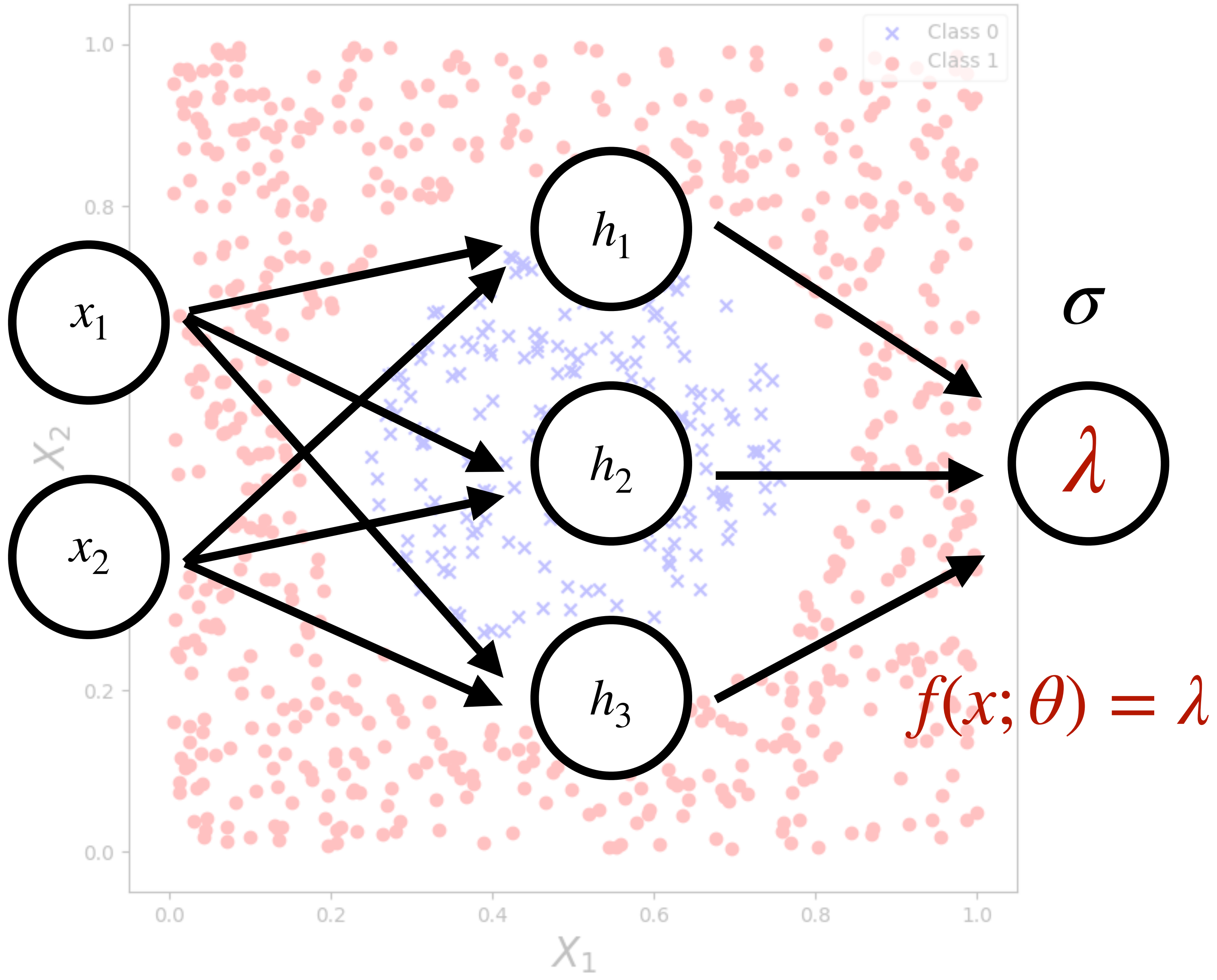
$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\boxed{\lambda^{(i)}} = P(y = 1 \mid x^{(i)})$

λ depends on the input x

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$
2. Learn an approximation p_{model} of the data distribution
3. Distribution $p_{model}(y \mid \lambda)$ over the output space
4. Set the network $f(x; \theta)$ to output (some of) the parameters λ



$$p_{model} \approx p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$

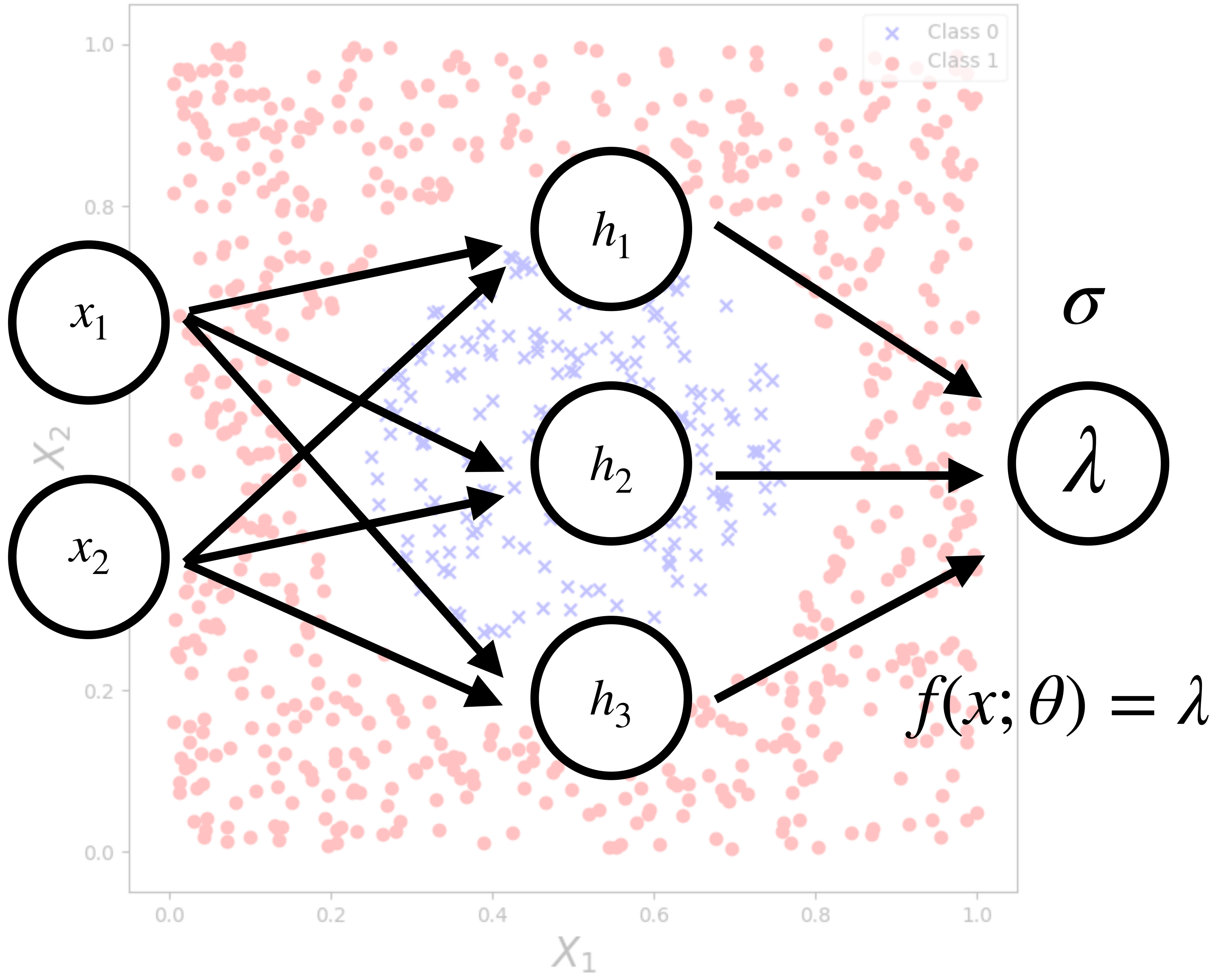
2. Learn an approximation p_{model} of the data distribution

3. Distribution $p_{model}(y \mid \lambda)$ over the output space

4. Set the network $f(x; \theta)$ to output (some of) the parameters λ

5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data

$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta)$$



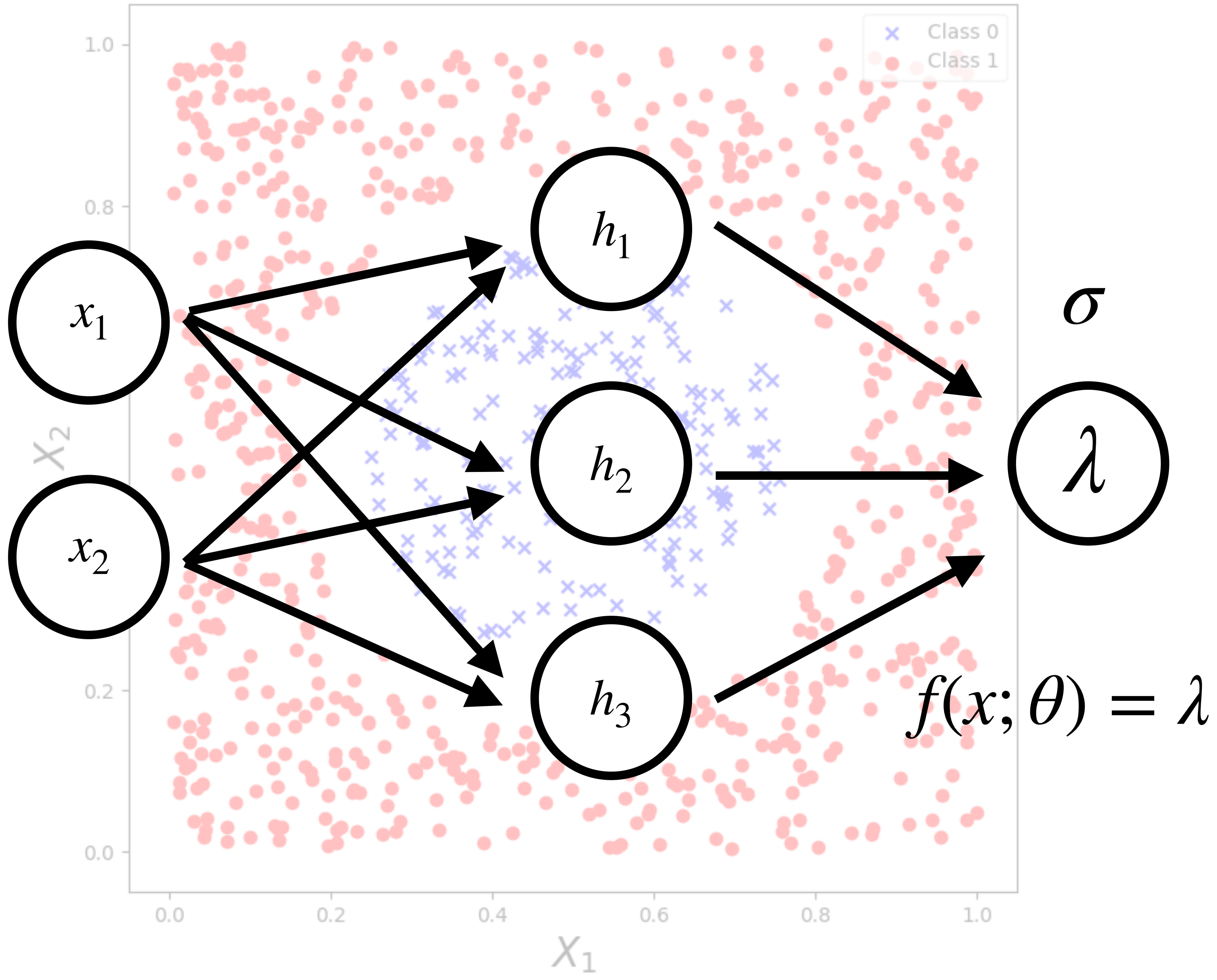
$$p_{model} \approx p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$
2. Learn an approximation p_{model} of the data distribution
3. Distribution $p_{model}(y \mid \lambda)$ over the output space
4. Set the network $f(x;\theta)$ to output (some of) the parameters λ
5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data



$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta)$$

$0, 1, 1, 1\ldots \quad (0.5,0.5), (0.2,0.2), (0.8,0.8), (0.8,0.1)\ldots$

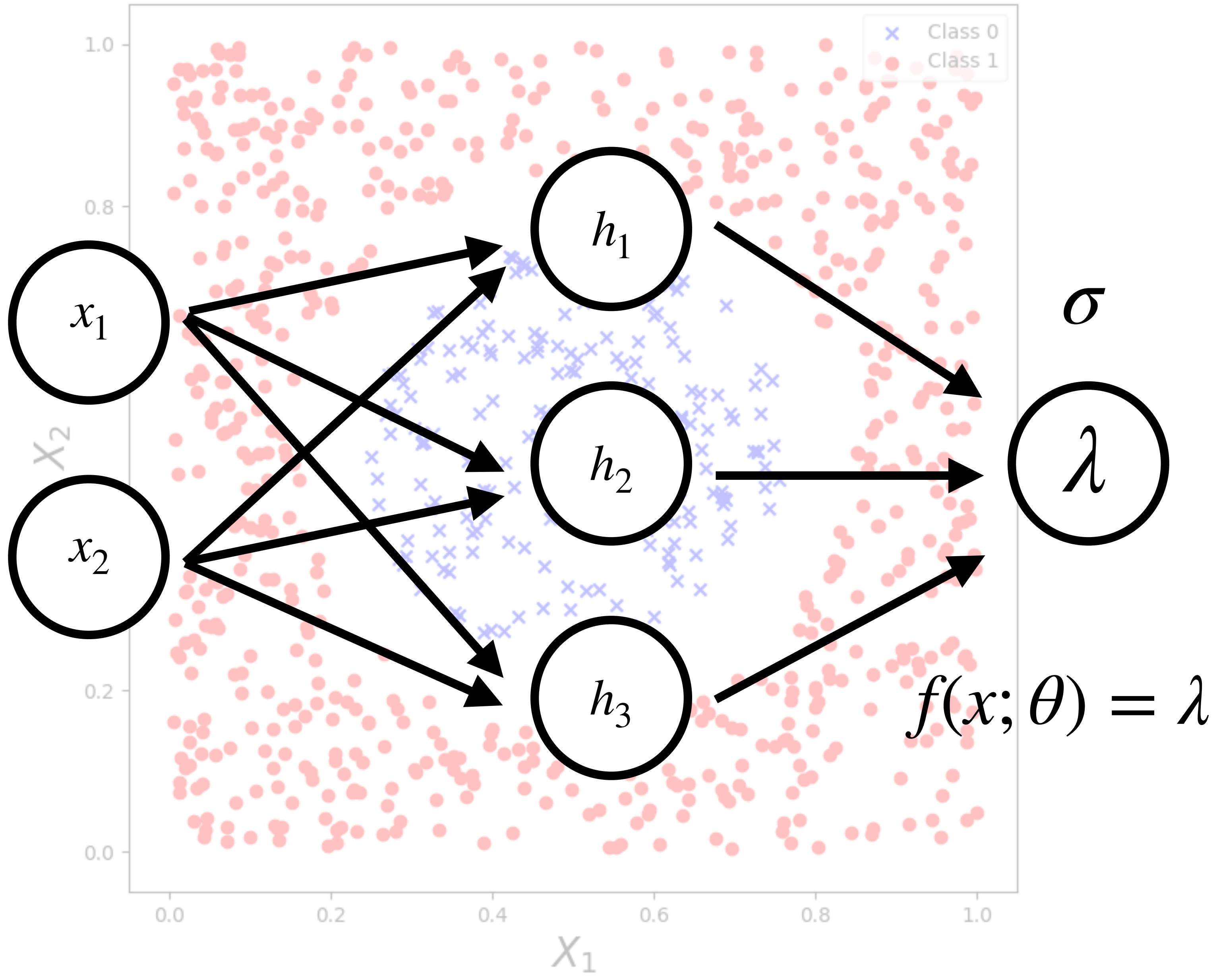
$$p_{model} \approx p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \le r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$
2. Learn an approximation p_{model} of the data distribution
3. Distribution $p_{model}(y \mid \lambda)$ over the output space
4. Set the network $f(x;\theta)$ to output (some of) the parameters λ
5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data



$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta)$$

$$P(0, 1, 1, 1\ldots \mid (0.5,0.5), (0.2,0.2), (0.8,0.8), (0.8,0.1)\ldots; \theta)$$

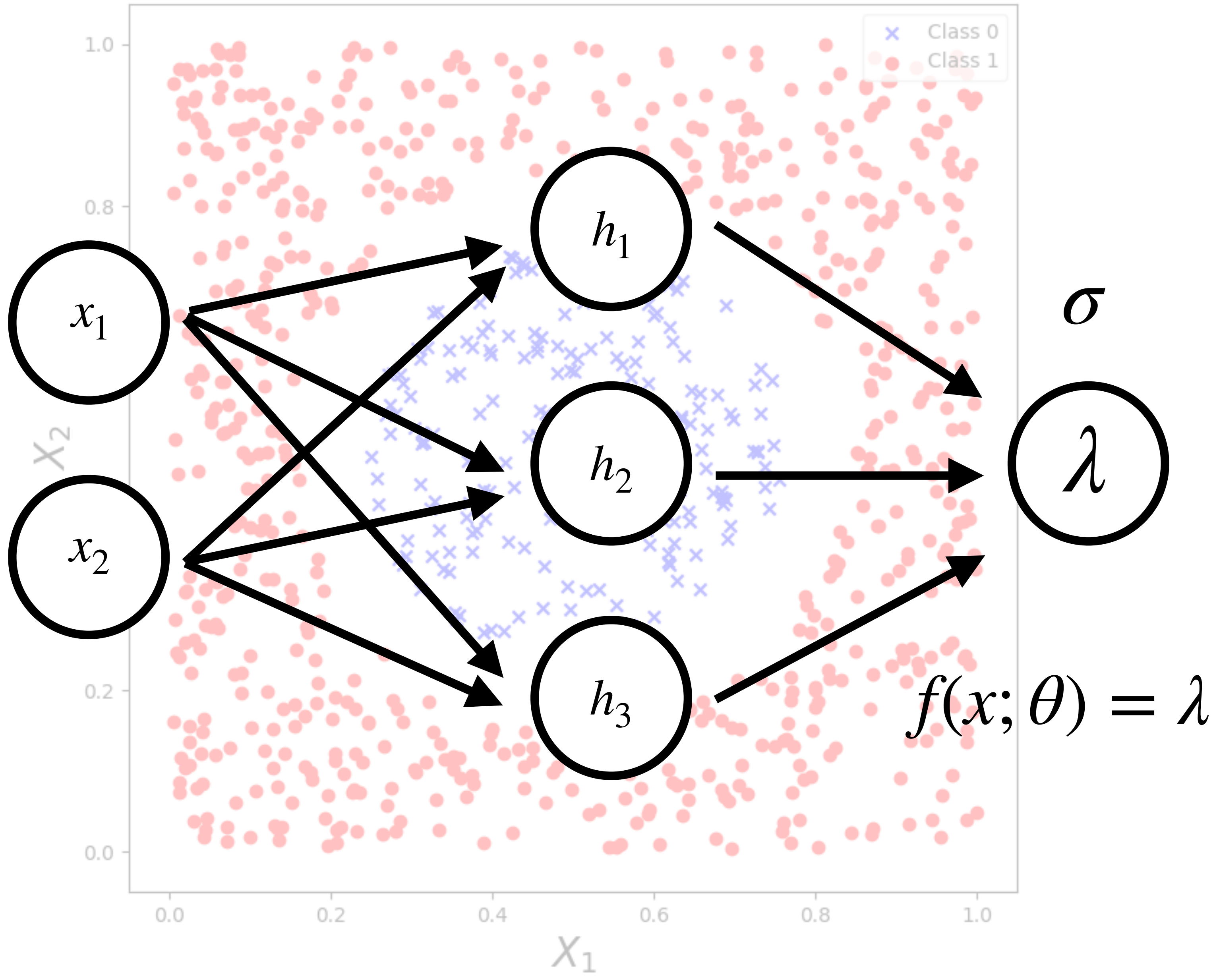
$$p_{model} \approx p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$
2. Learn an approximation p_{model} of the data distribution
3. Distribution $p_{model}(y \mid \lambda)$ over the output space
4. Set the network $f(x;\theta)$ to output (some of) the parameters λ
5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data



$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta) = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \theta)$$

Multiplication $P(y^{(1)} \mid x^{(1)}; \theta) \times P(y^{(2)} \mid x^{(2)}; \theta) \times \ldots \times P(y^{(n)} \mid x^{(n)}; \theta)$

$$p_{model} \approx \; p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of input/output pairs

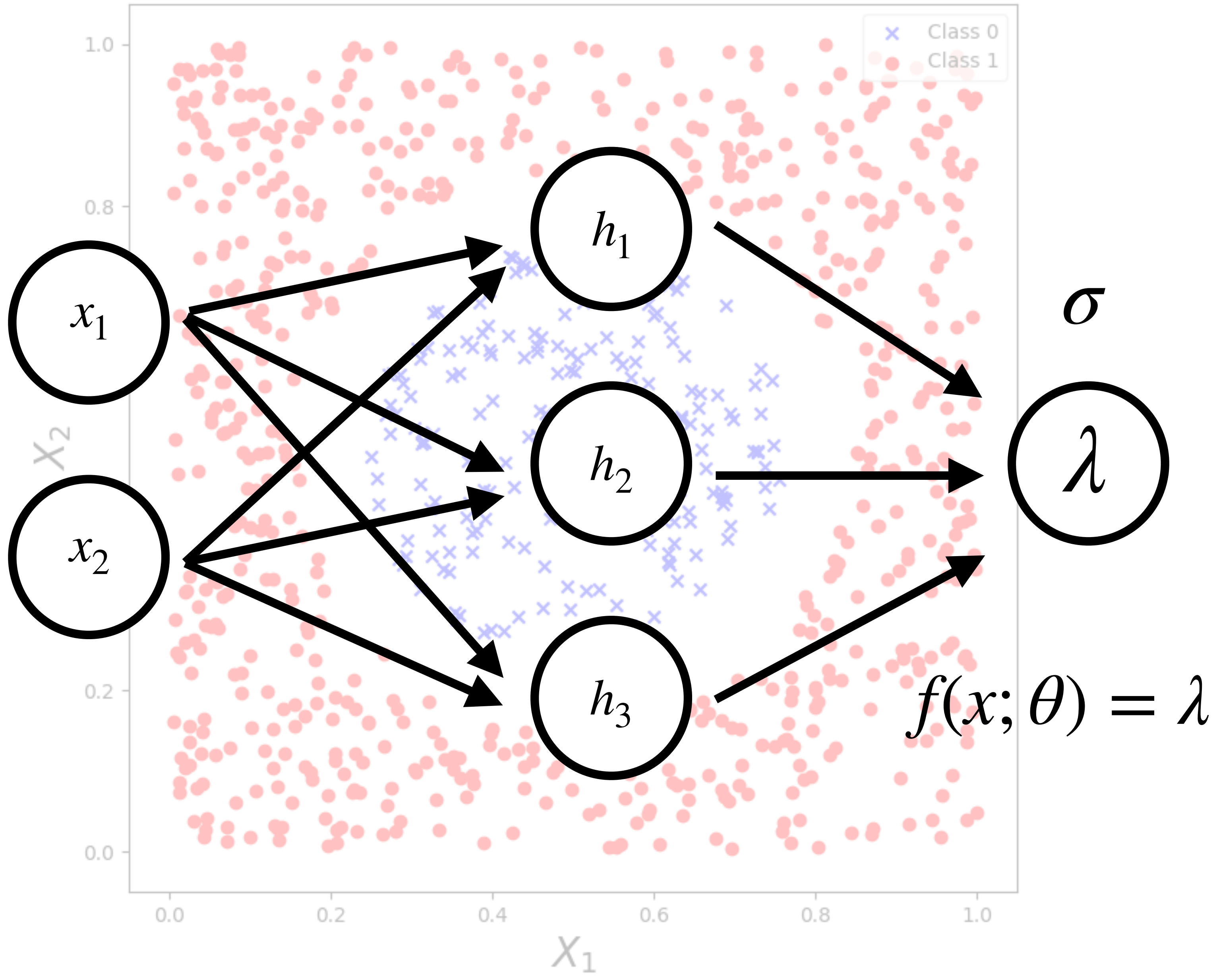
1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$

2. Learn an approximation p_{model} of the data distribution

3. Distribution $p_{model}(y \mid \lambda)$ over the output space

4. Set the network $f(x;\theta)$ to output (some of) the parameters λ

5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data



$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta) = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \theta)$$

Can we do that?

$$p_{model} \approx p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of i.i.d. input/output pairs

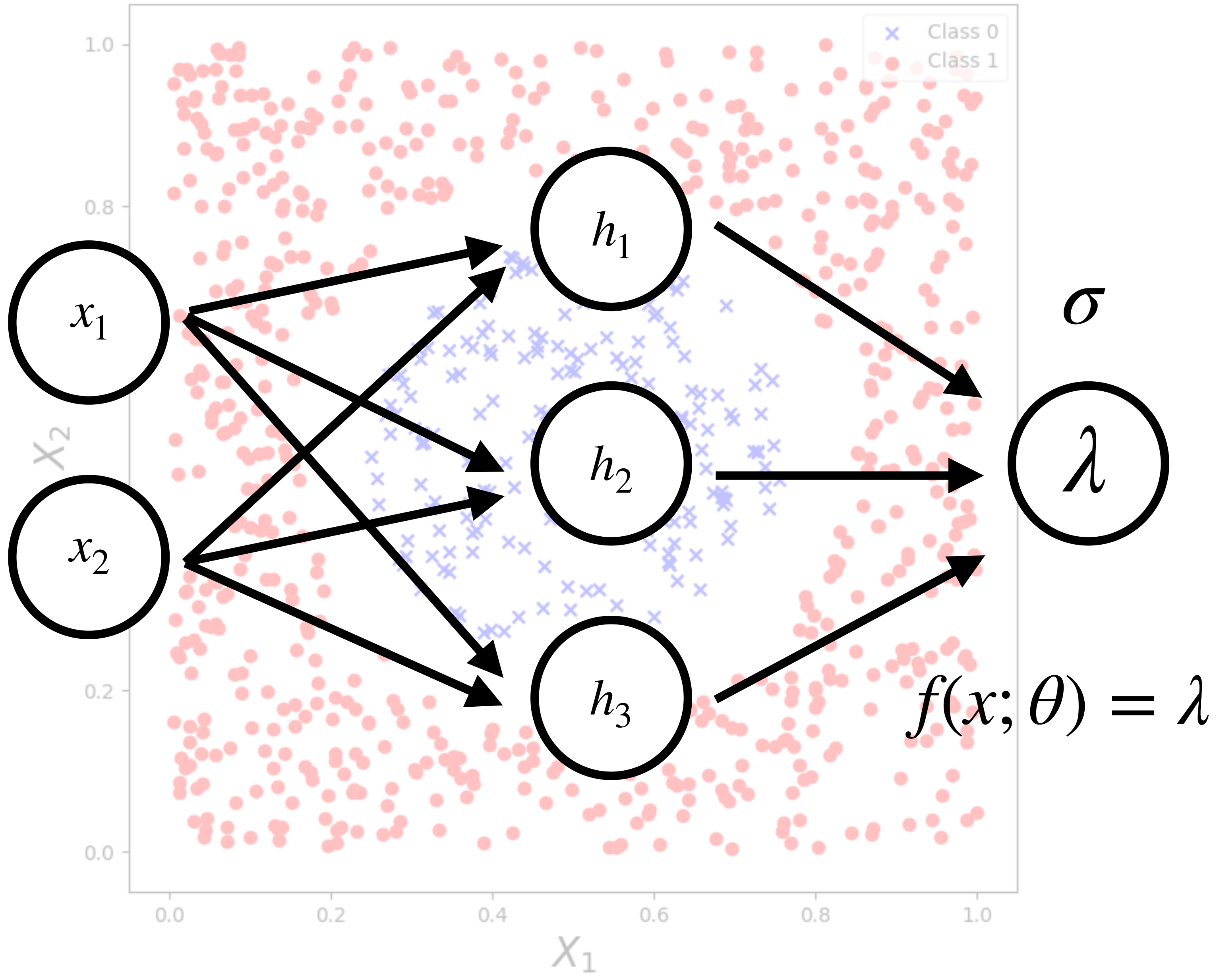
1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$

2. Learn an approximation p_{model} of the data distribution

3. Distribution $p_{model}(y \mid \lambda)$ over the output space

4. Set the network $f(x;\theta)$ to output (some of) the parameters λ

5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data



$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta) = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \theta)$$

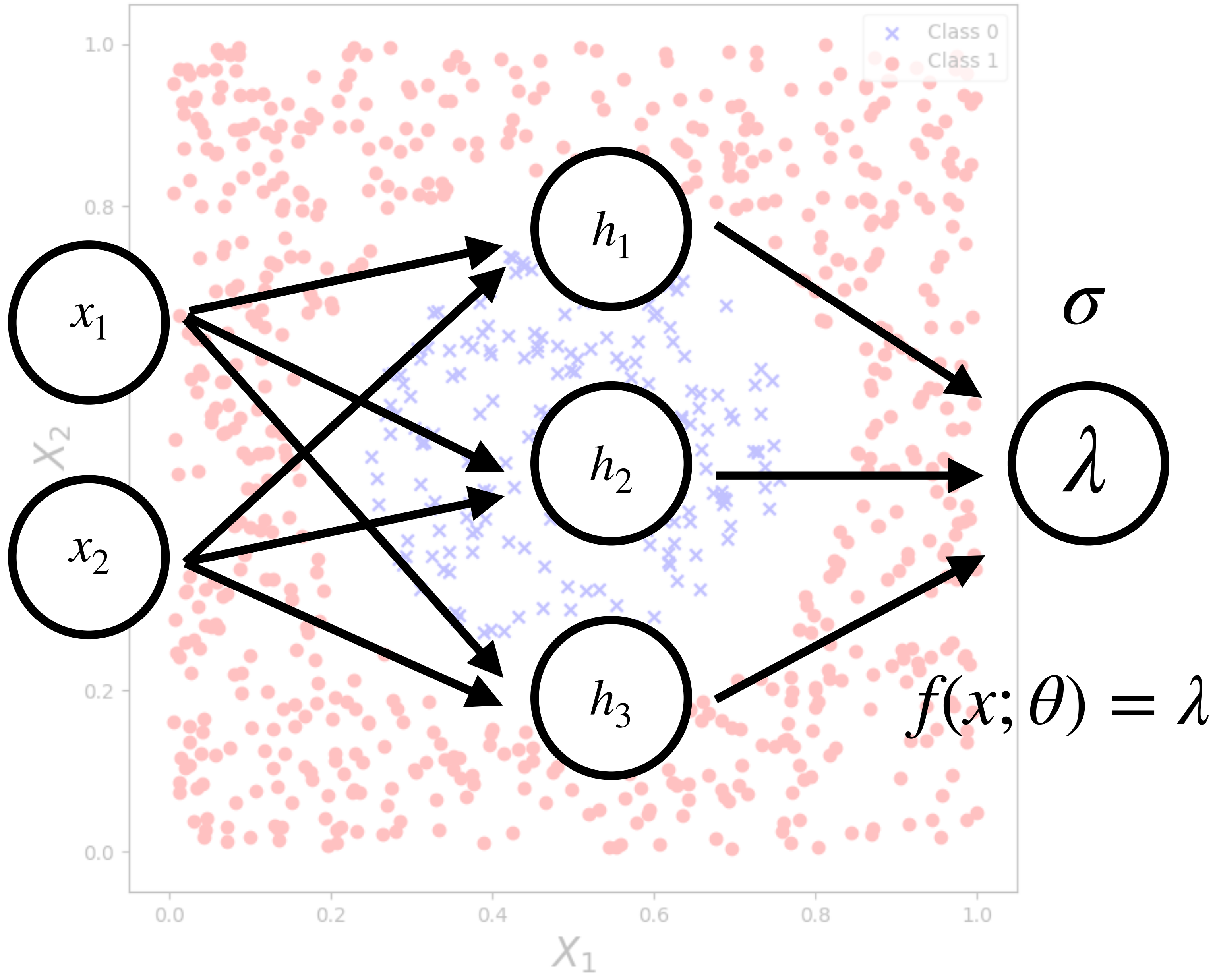
$$p_{model} \approx \; p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

Assuming y conditional independent on x and identically distributed

Maximum Likelihood

Training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ of i.i.d. input/output pairs



1. $y^{(i)}$ drawn from an unknown distribution $p_{data}(y \mid x^{(i)})$

2. Learn an approximation p_{model} of the data distribution

3. Distribution $p_{model}(y \mid \lambda)$ over the output space

4. Set the network $f(x;\theta)$ to output (some of) the parameters λ

5. Look for parameters $\hat{\theta}$ to maximize the total likelihood of the training data

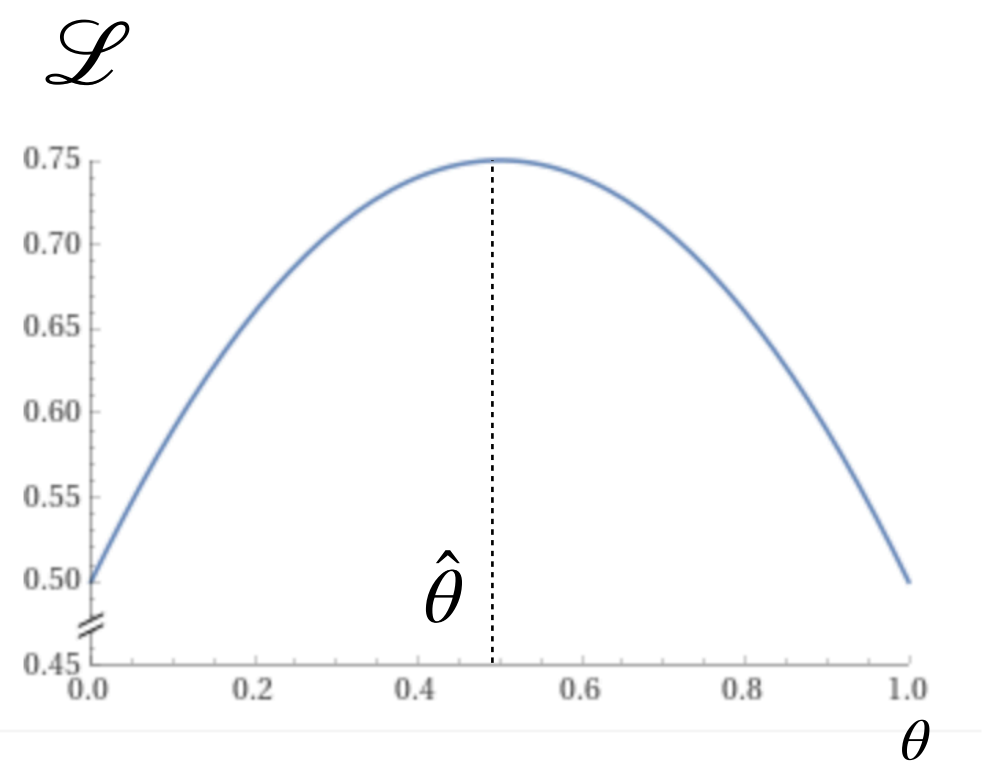
$$P(y^{(1)}, \ldots, y^{(n)} \mid x^{(1)}, \ldots, x^{(n)}; \theta) = \prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)}; \theta))$$

$$p_{model} \approx \; p_{data}(y \mid x^{(i)}) = \begin{cases} \delta_0(y) & dist(x^{(i)}, c) \leq r \\ \delta_1(y) & dist(x^{(i)}, c) > r \end{cases}$$

$p_{model}(y \mid \lambda)$ over $\{0,1\}$ only one parameter $\lambda^{(i)} = P(y = 1 \mid x^{(i)})$

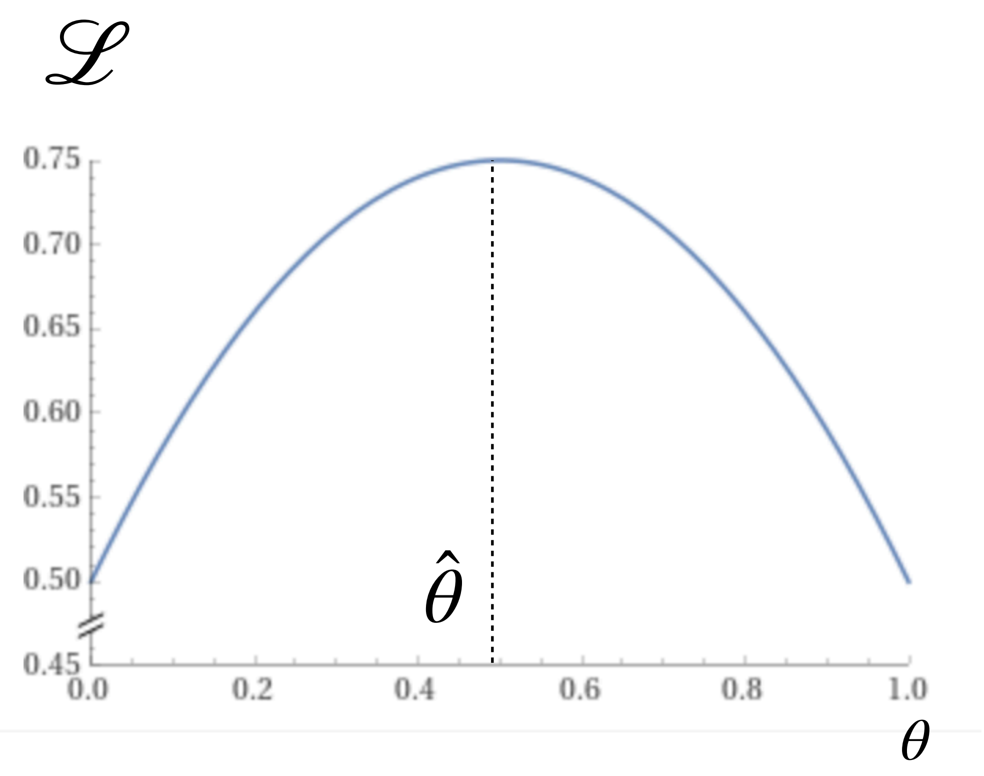
Maximum Likelihood

Maximum Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\overbrace{\prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)};\theta))}^{likelihood}\right)$



Maximum Likelihood

Maximum Likelihood criterion: $\hat{\theta} = argmax_\theta \left(\overbrace{\prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)}; \theta))}^{\textit{likelihood}} \right)$



Multiplying many probabilities (numbers lower than 1): the product can become very small!

$$0.01 \times 0.1 \times \ldots \approx 0$$

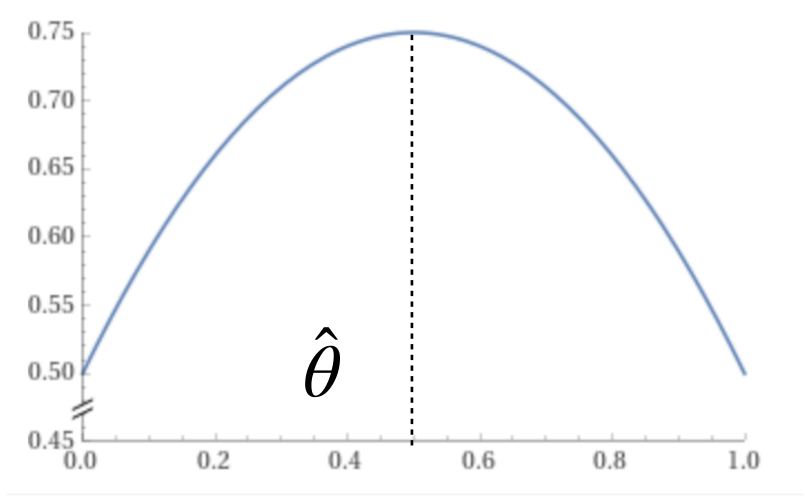
Maximum Likelihood

Maximum Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)};\theta))\right)$

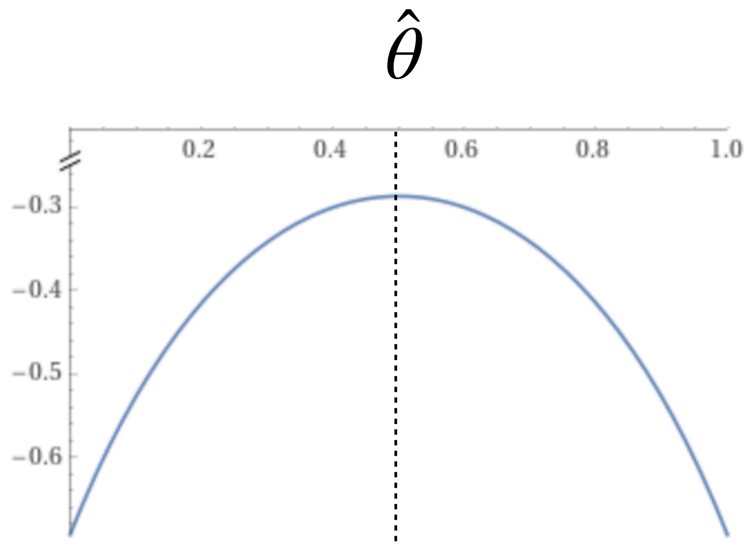
$\Updownarrow$

Maximum log-Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\overbrace{\log\left(\prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)};\theta))\right)}^{\textit{log- likelihood}}\right)$

$\mathscr{L}$



$log(\mathscr{L})$



θ

Log is an increasing function

Maximum Likelihood

Maximum Likelihood criterion: $\hat{\theta} = argmax_\theta \left(\prod_{i=1}^{n} p_{model}(y^{(i)} \,|\, f(x^{(i)}; \theta)) \right)$

$\Updownarrow$

Sum of log probabilities!

Maximum log-Likelihood criterion: $\hat{\theta} = argmax_\theta \left(\overbrace{\log \left(\prod_{i=1}^{n} p_{model}(y^{(i)} \,|\, f(x^{(i)}; \theta)) \right)}^{\textit{log- likelihood}} \right) = argmax_\theta \left(\sum_{i=1}^{n} \log(p_{model}(y^{(i)} \,|\, f(x^{(i)}; \theta))) \right)$

Maximum Likelihood

Maximum Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)};\theta))\right)$

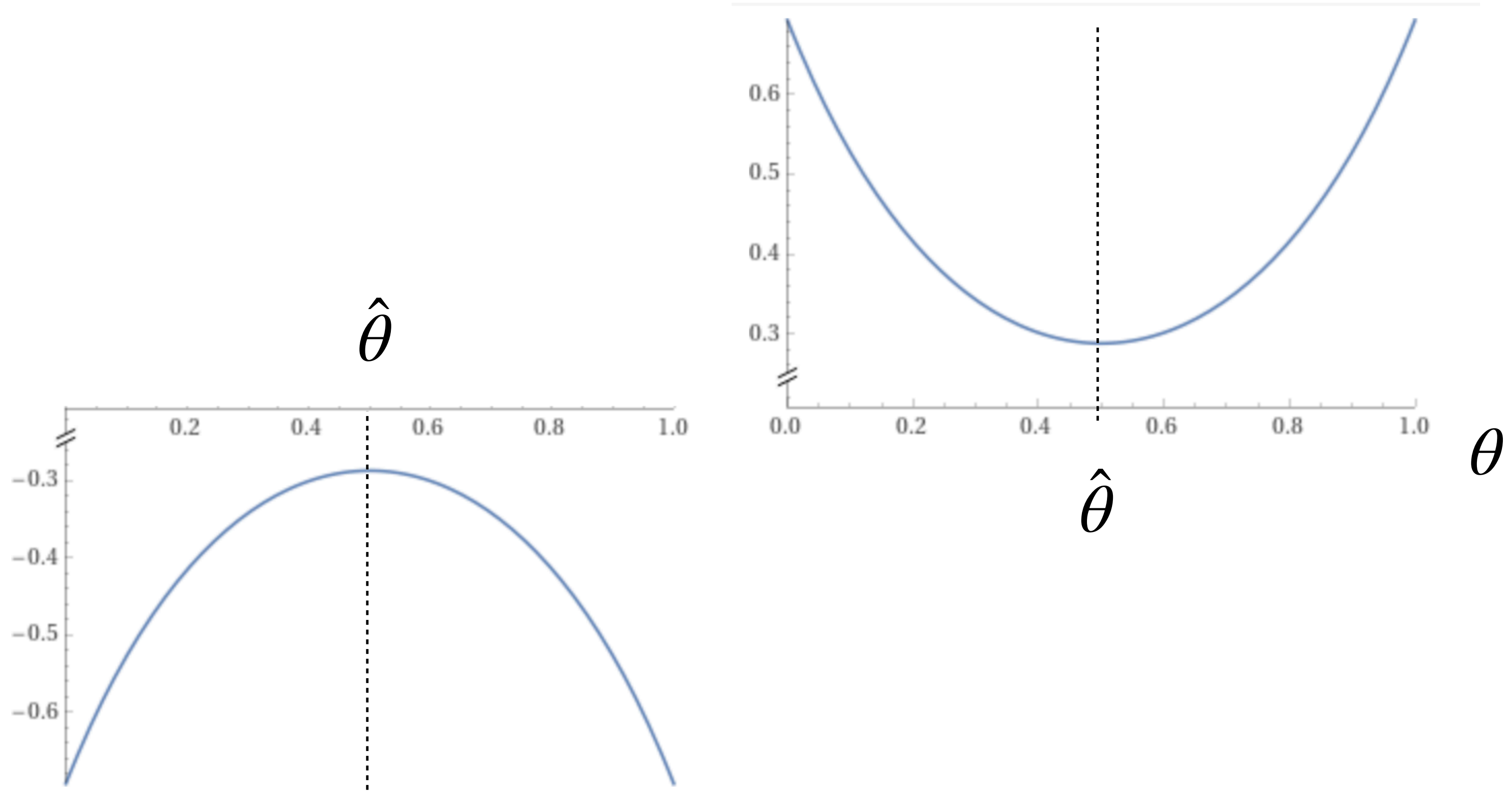
$\Updownarrow$

Maximum log-Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\sum_{i=1}^{n} \log(p_{model}(y^{(i)} \mid f(x^{(i)};\theta)))\right)$

$\Updownarrow$

Minimum negative log-Likelihood criterion: $\hat{\theta} = argmin_{\theta}\left(\overbrace{\sum_{i=1}^{n} -\log(p_{model}(y^{(i)} \mid f(x^{(i)};\theta)))}^{\textit{negative log-likelihood}}\right)$

$log(\mathscr{L})$ $\quad$ $-log(\mathscr{L})$



$argmin(-\mathscr{L}) = argmax(\mathscr{L})$

Maximum Likelihood

Maximum Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\prod_{i=1}^{n} p_{model}(y^{(i)} \mid f(x^{(i)}; \theta))\right)$

$$\Updownarrow$$

Maximum log-Likelihood criterion: $\hat{\theta} = argmax_{\theta}\left(\sum_{i=1}^{n} \log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))\right)$

$$\Updownarrow$$

Minimum negative log-Likelihood criterion: $\hat{\theta} = argmin_{\theta}\left(\sum_{i=1}^{n} -\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))\right)$

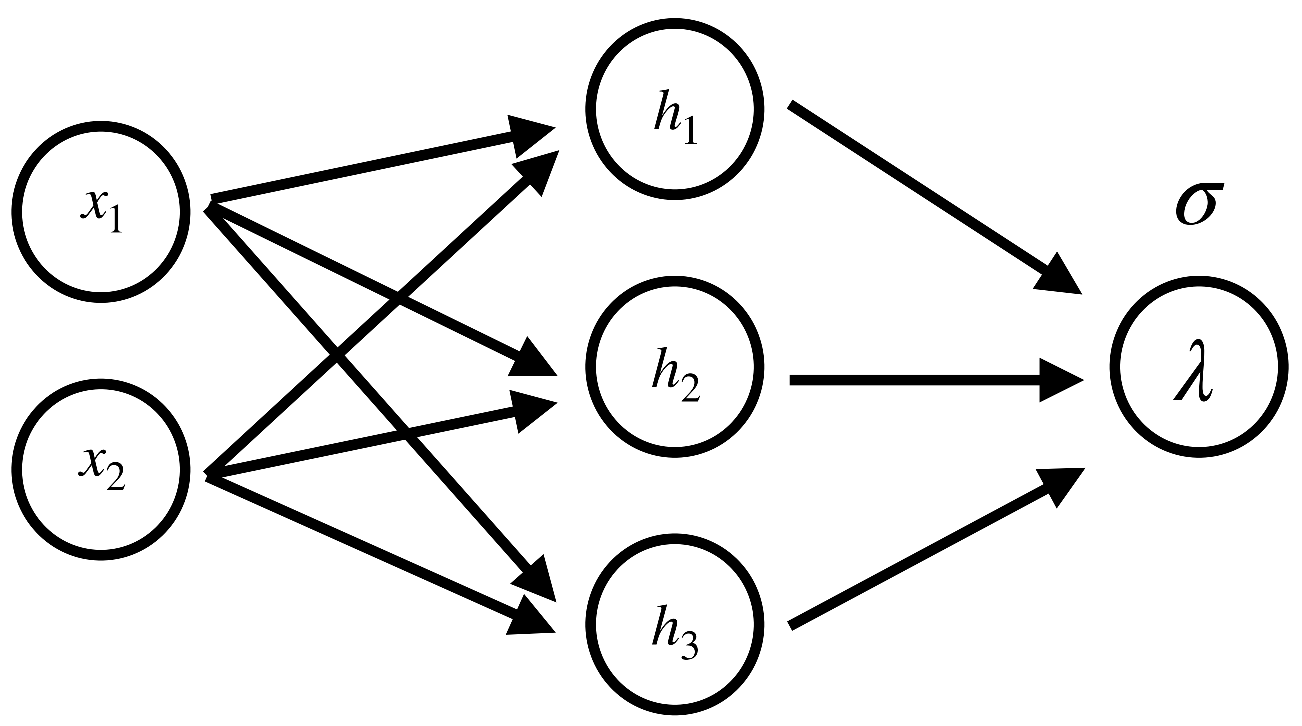
$$\Updownarrow$$

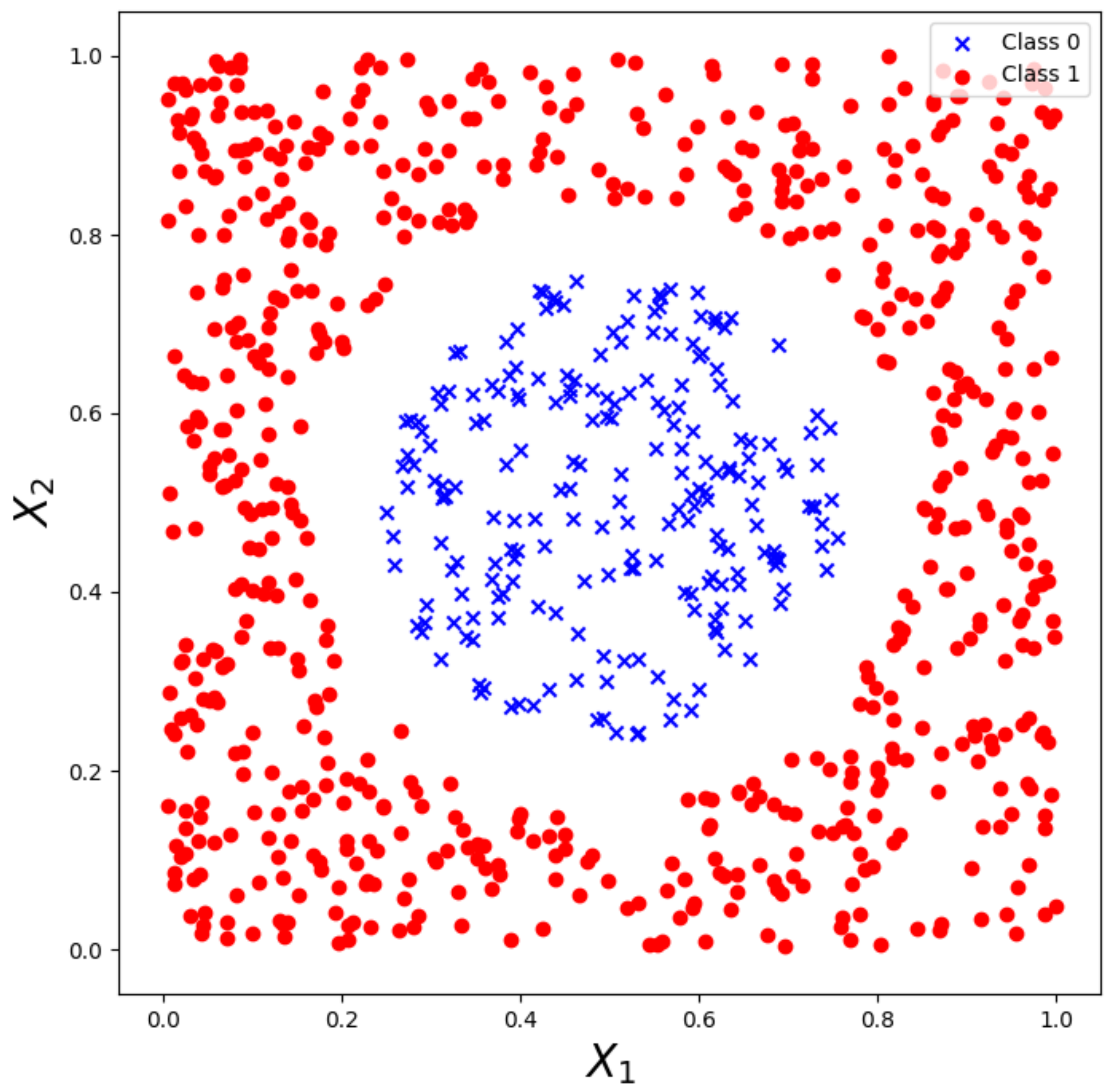
Minimum loss function: $\hat{\theta} = argmin_{\theta}\left(\overbrace{\frac{1}{n}\sum_{i=1}^{n} -\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))}^{\text{Loss}}\right) = argmin_{\theta}L(\theta)$

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1



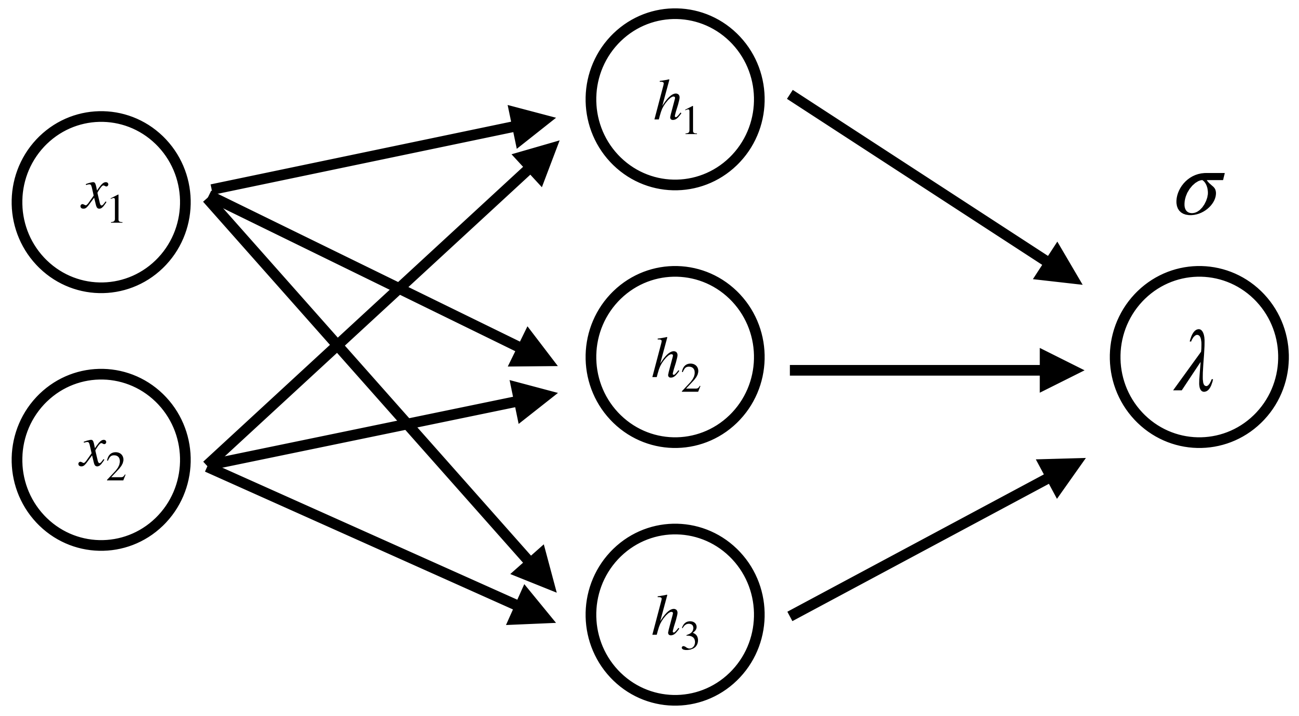


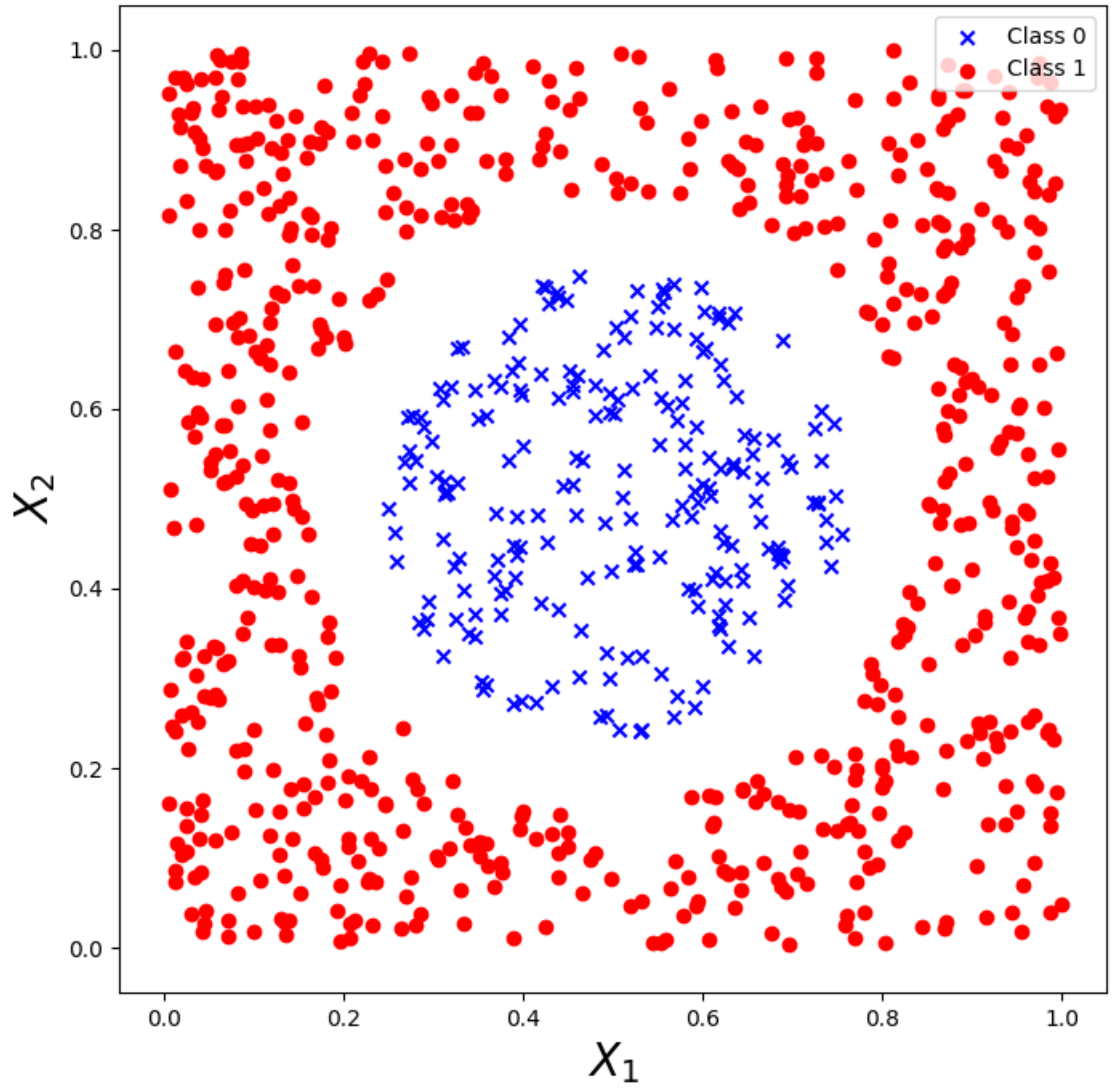
Loss function: $L(\theta) = \dfrac{1}{n}\sum_{i=1}^{n} -\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))$

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1



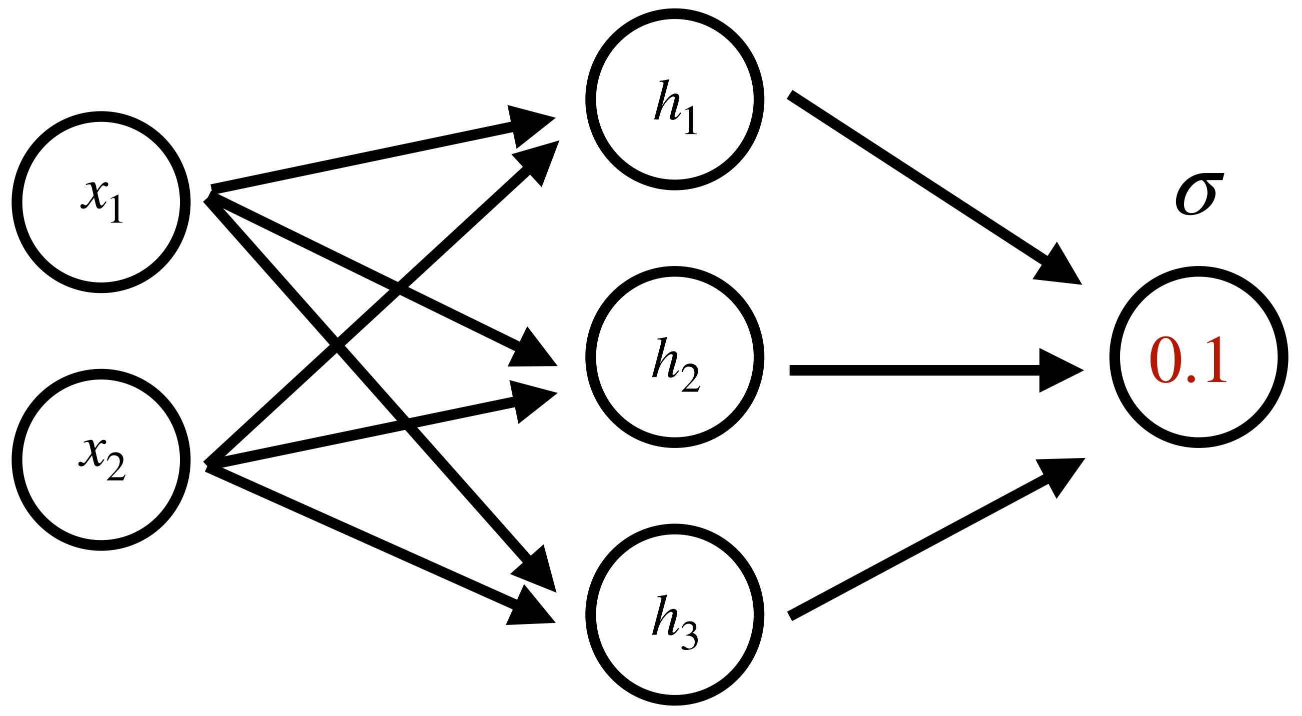


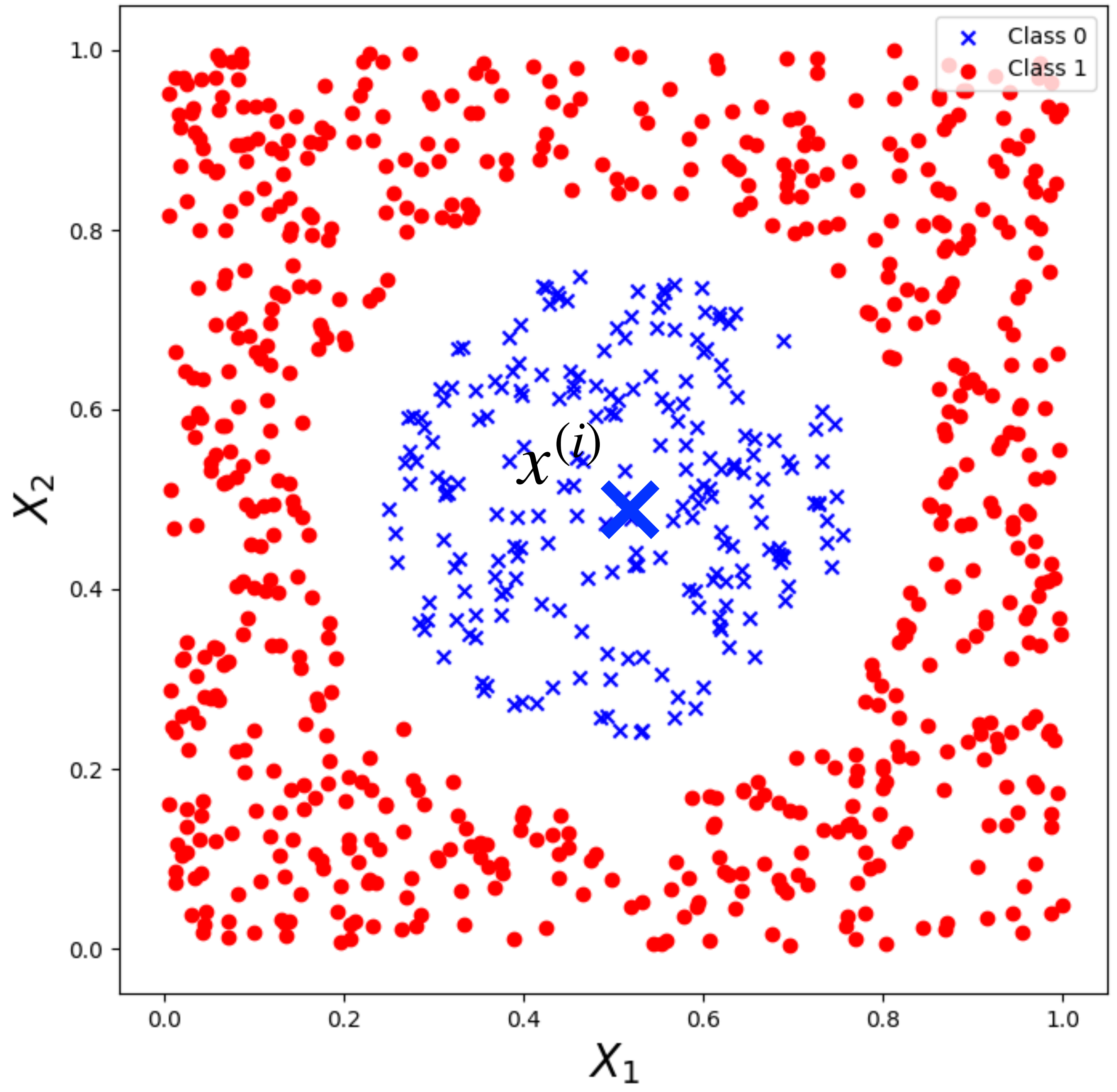
Loss function: $L(\theta) = \frac{1}{n} \sum_{i=1}^{n} -\log\left(\boxed{p_{model}(y^{(i)} \mid f(x^{(i)}; \theta))}\right) \longrightarrow p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)) = \begin{cases} \lambda^{(i)} & if\ y^{(i)} = 1 \\ 1 - \lambda^{(i)} & if\ y^{(i)} = 0 \end{cases}$

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





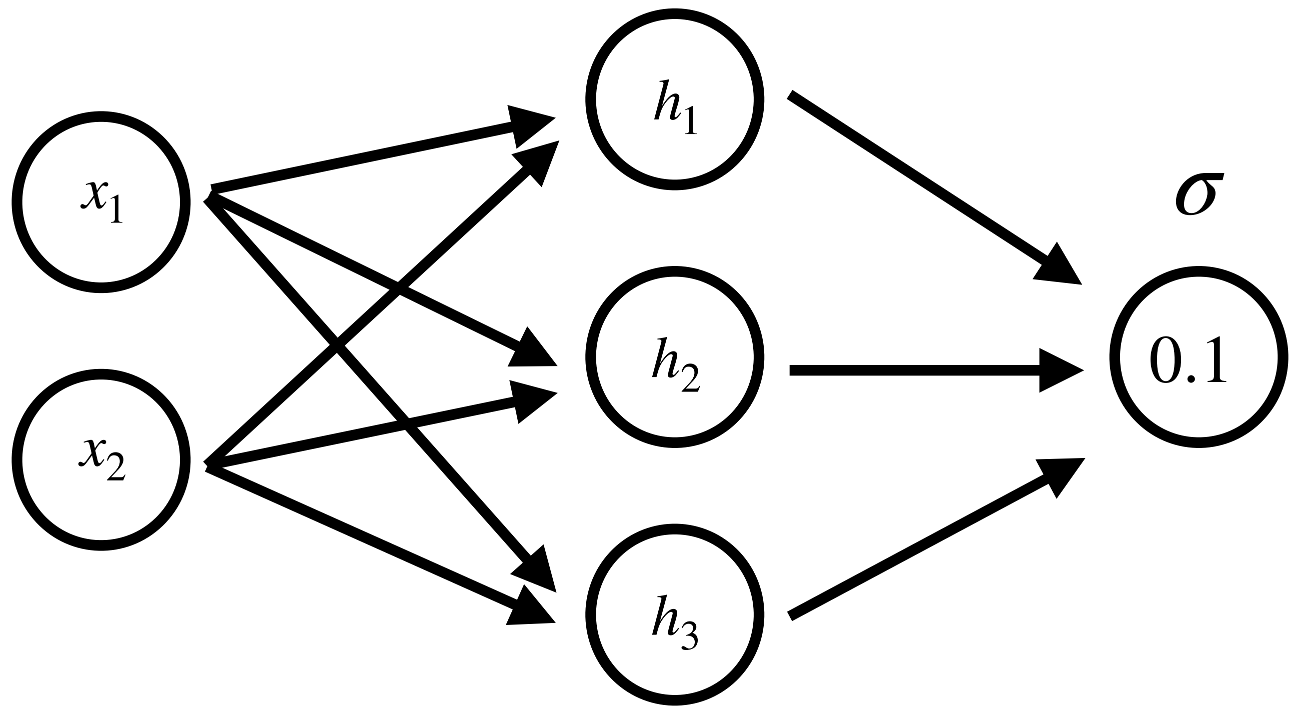
Loss function: $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} -\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))$

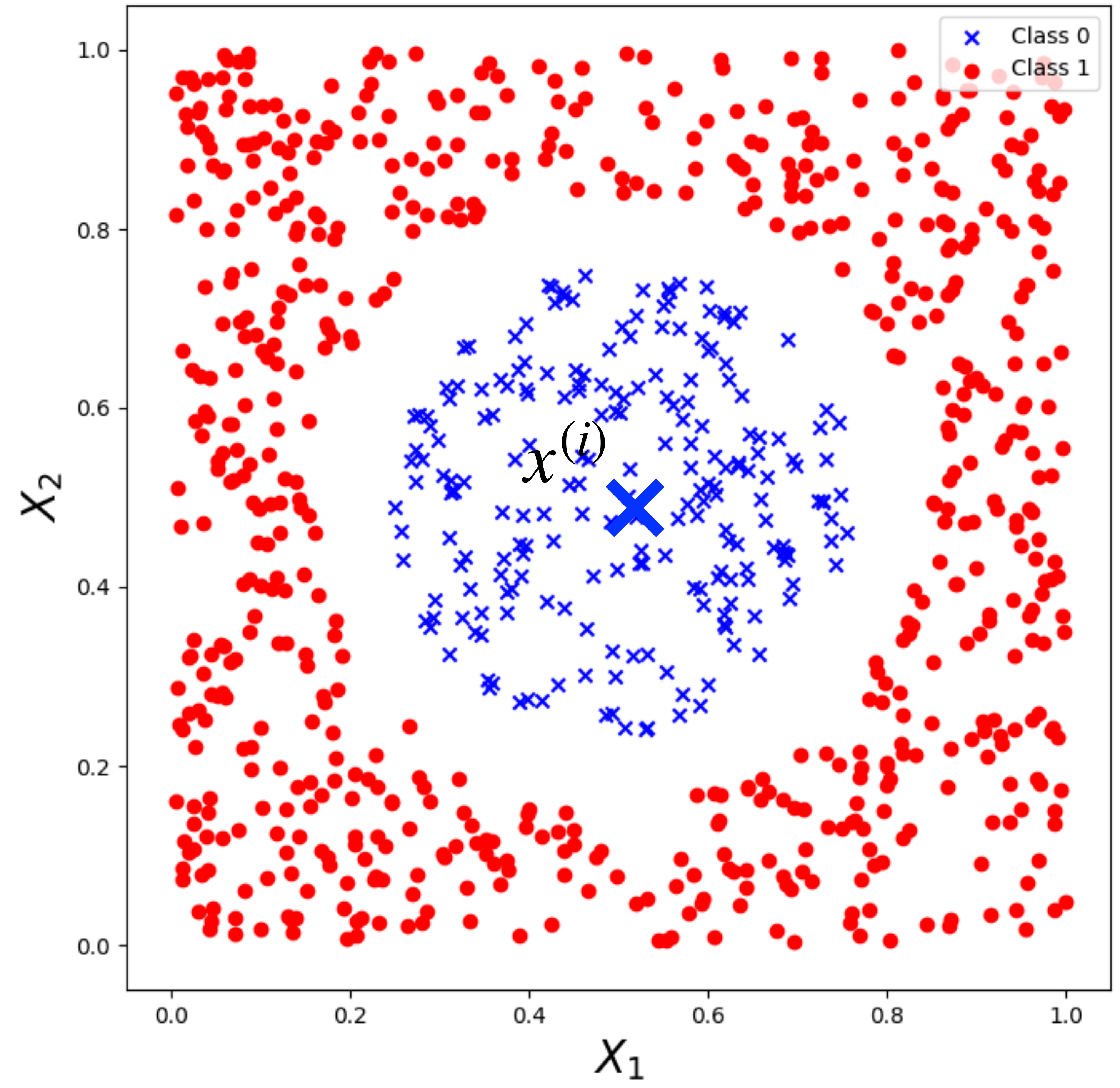
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	?	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \boxed{-\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))}$

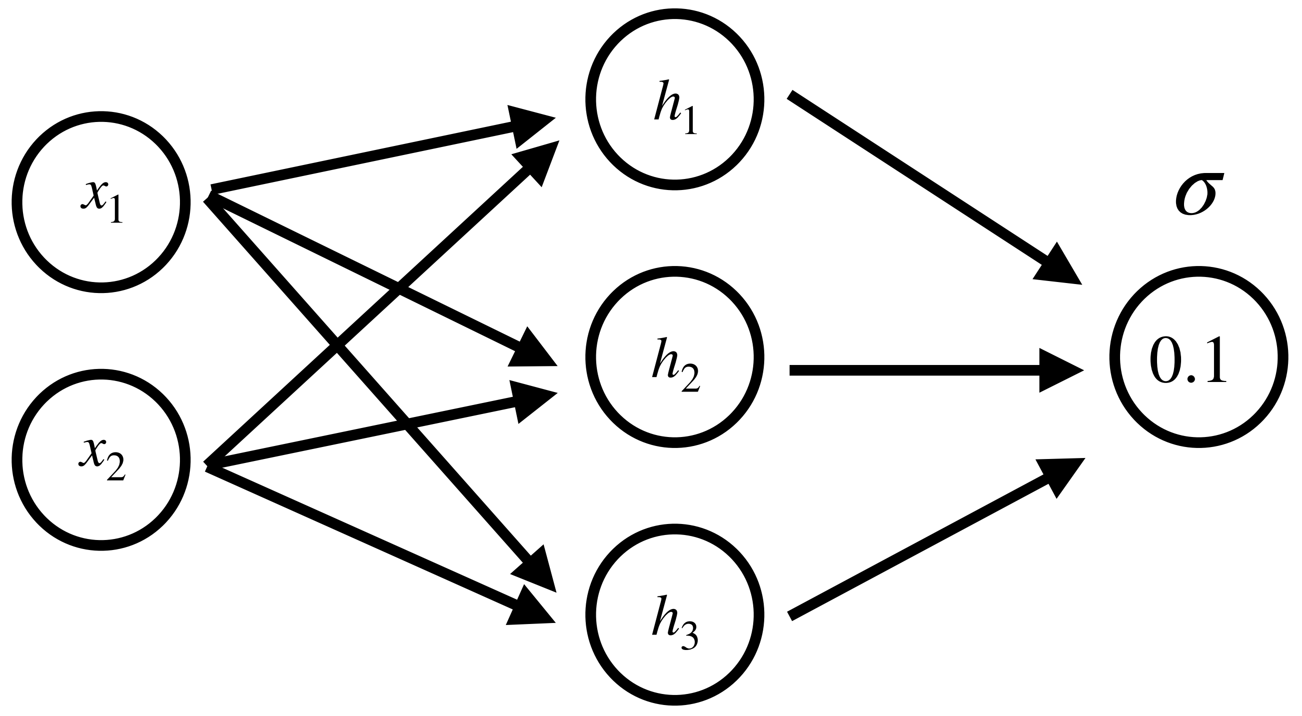
$$l^{(i)} = -\log(p_{model}(y^{(i)} = 0 \mid \lambda^{(i)} = 0.1)) = -\log(0.9)$$

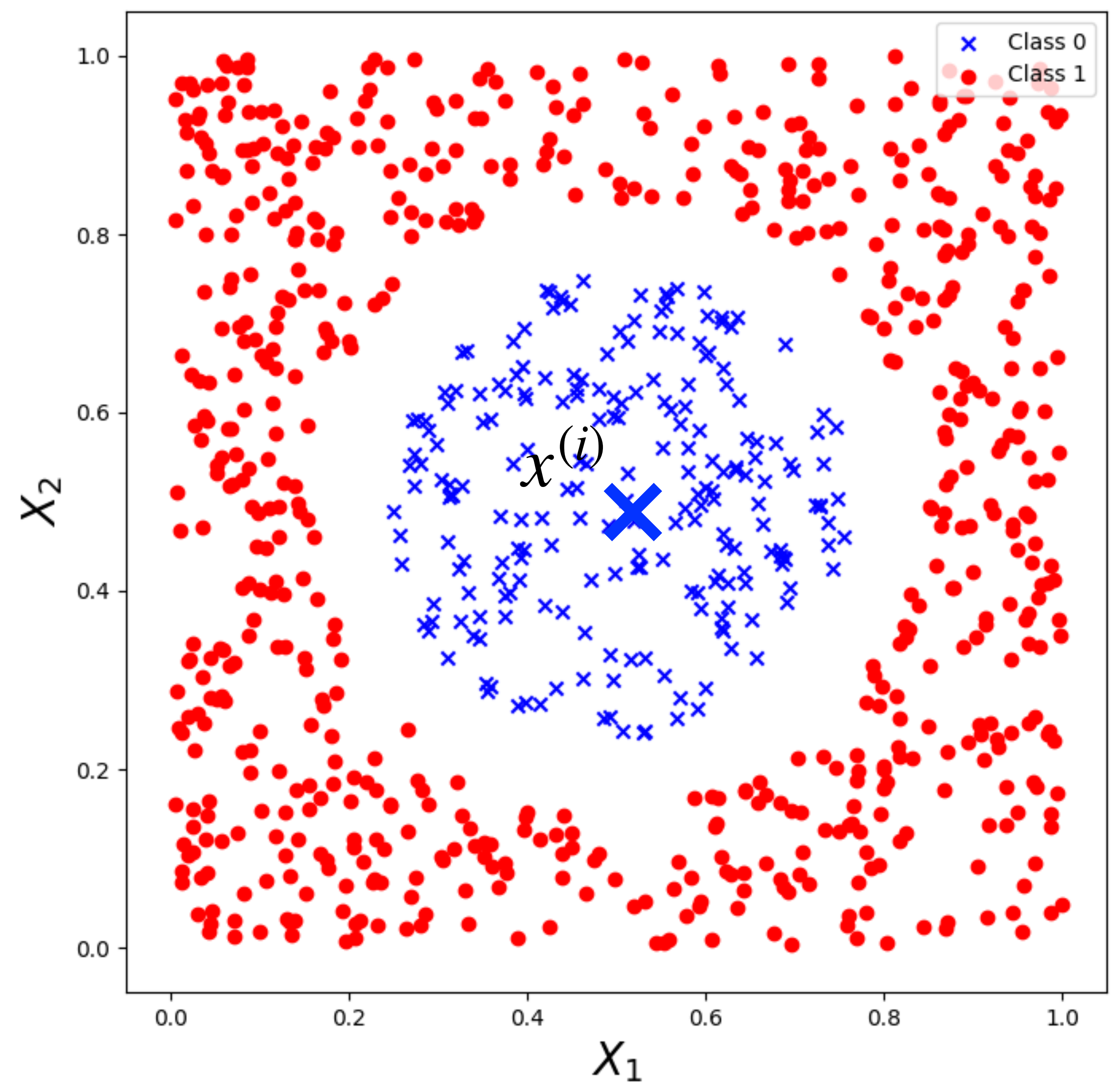
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{-\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))}$

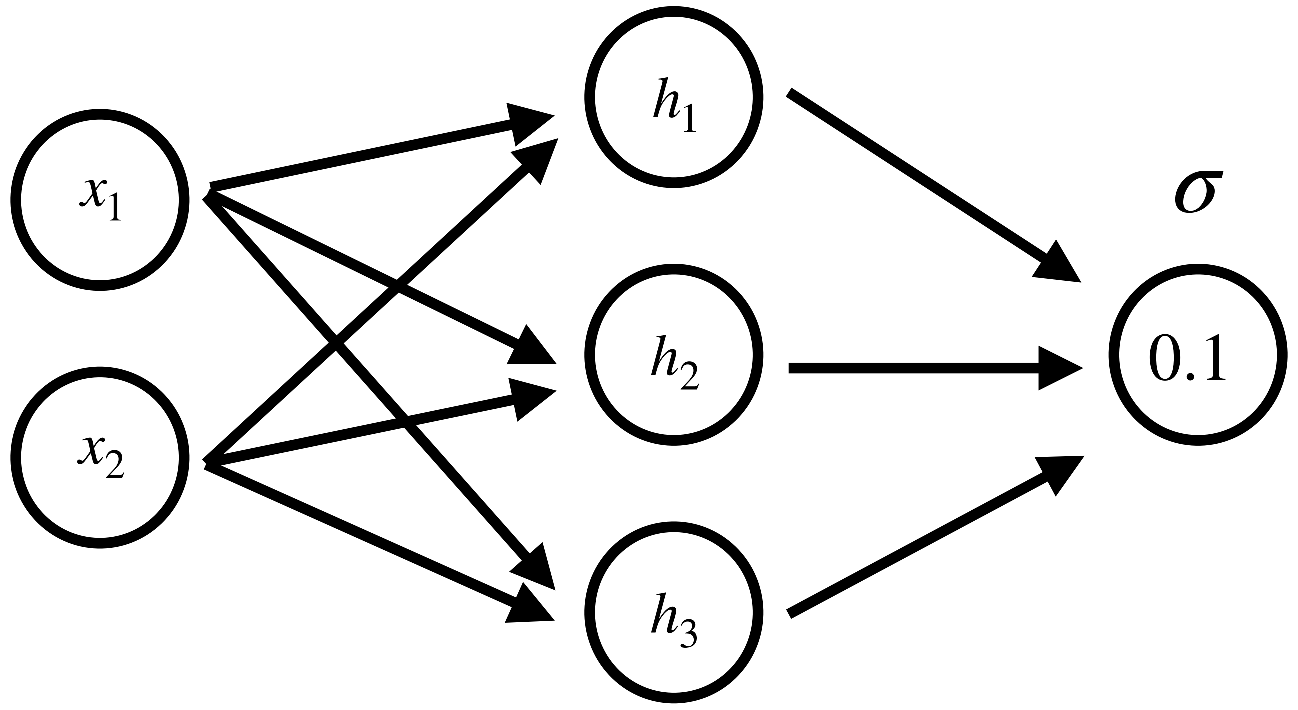
$$l^{(i)} = -y^{(i)}\log(\lambda^{(i)}) - (1 - y^{(i)})\log(1 - \lambda^{(i)})$$

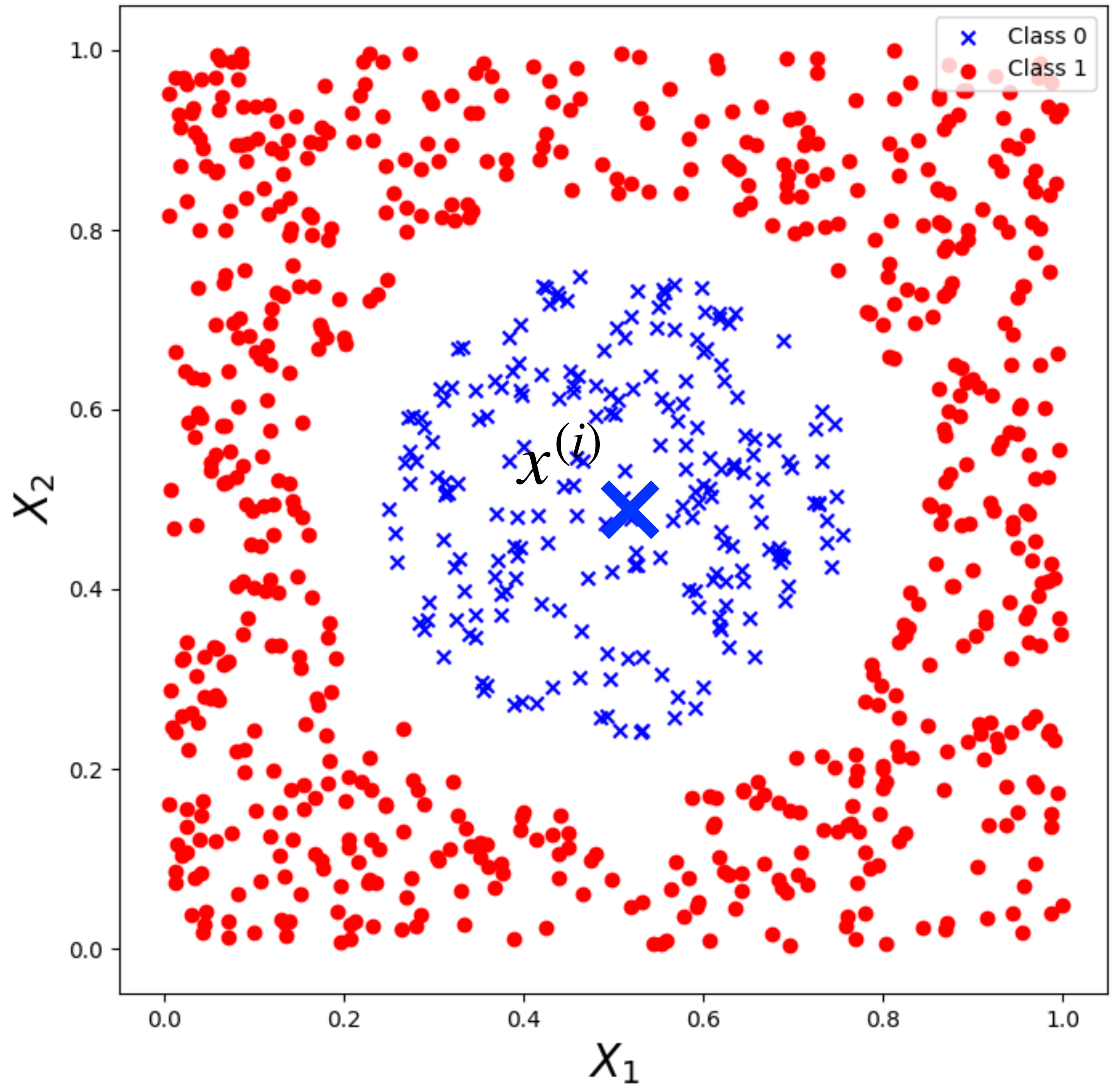
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{-\log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))}$

$$l^{(i)} = -\boxed{y^{(i)}}\log(\lambda^{(i)}) - (1 - \boxed{y^{(i)}})\log(1 - \lambda^{(i)})$$

$\downarrow$ $\downarrow$

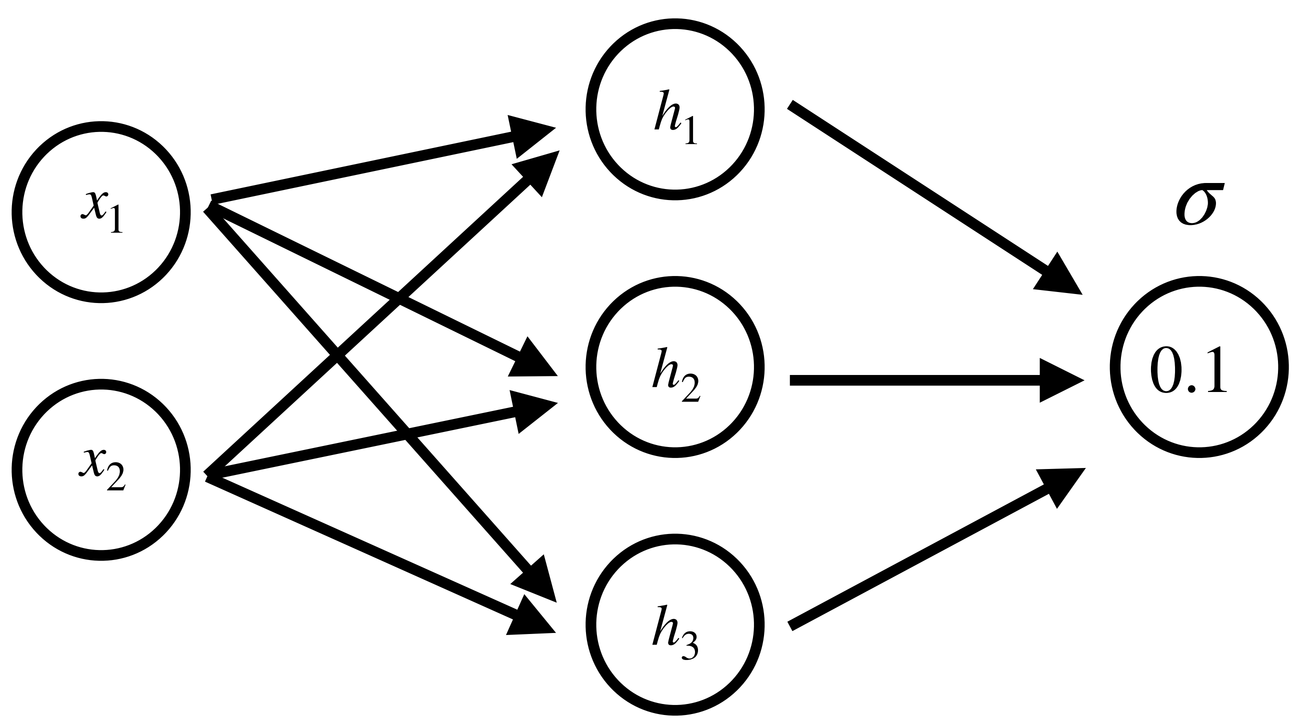
0 0

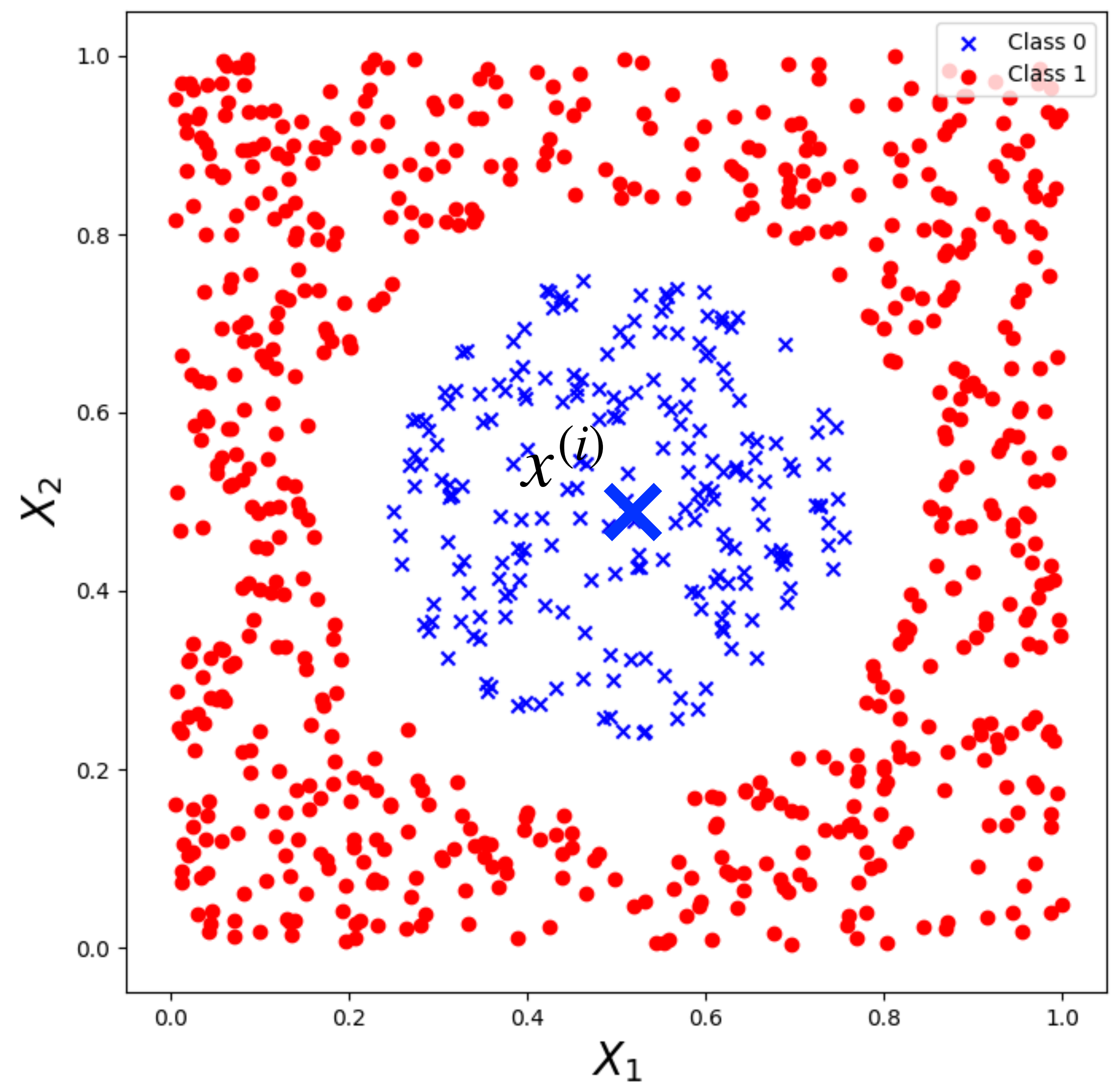
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{- \log(p_{model}(y^{(i)} \mid f(x^{(i)}; \theta)))}$

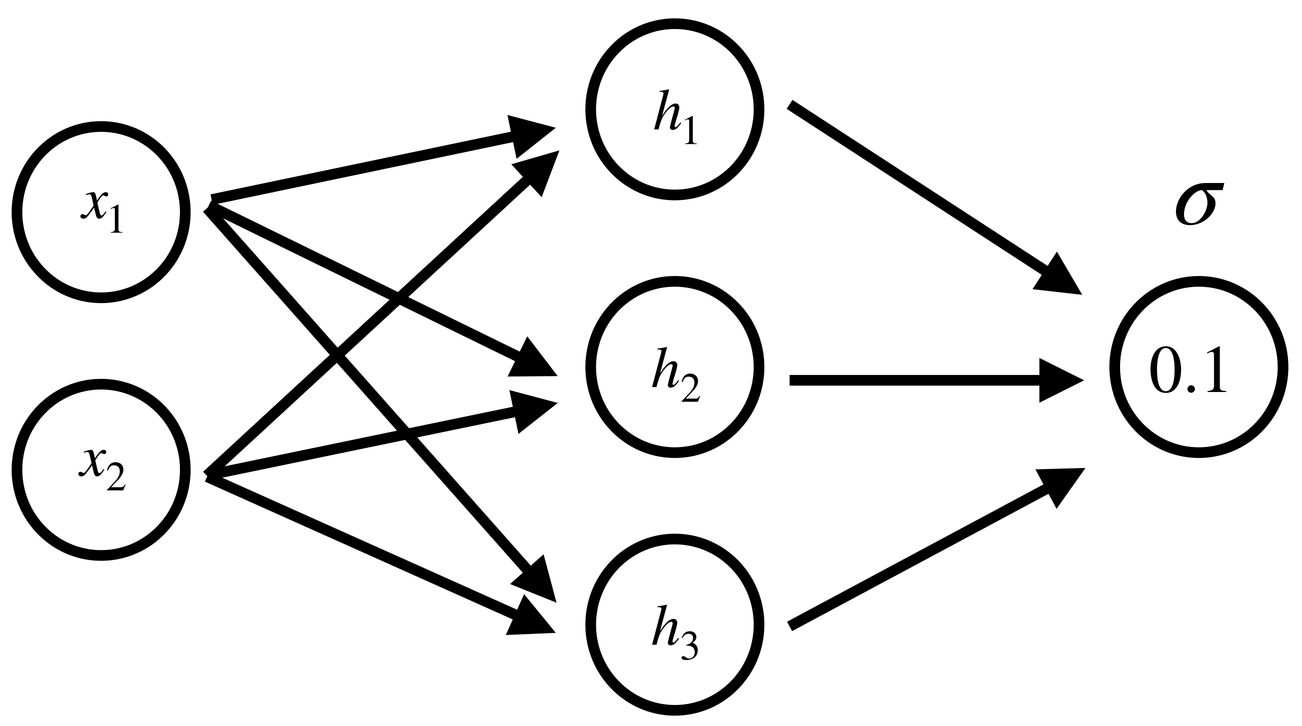
$$l^{(i)} = -\log(1 - \lambda^{(i)}) = -\log(0.9)$$

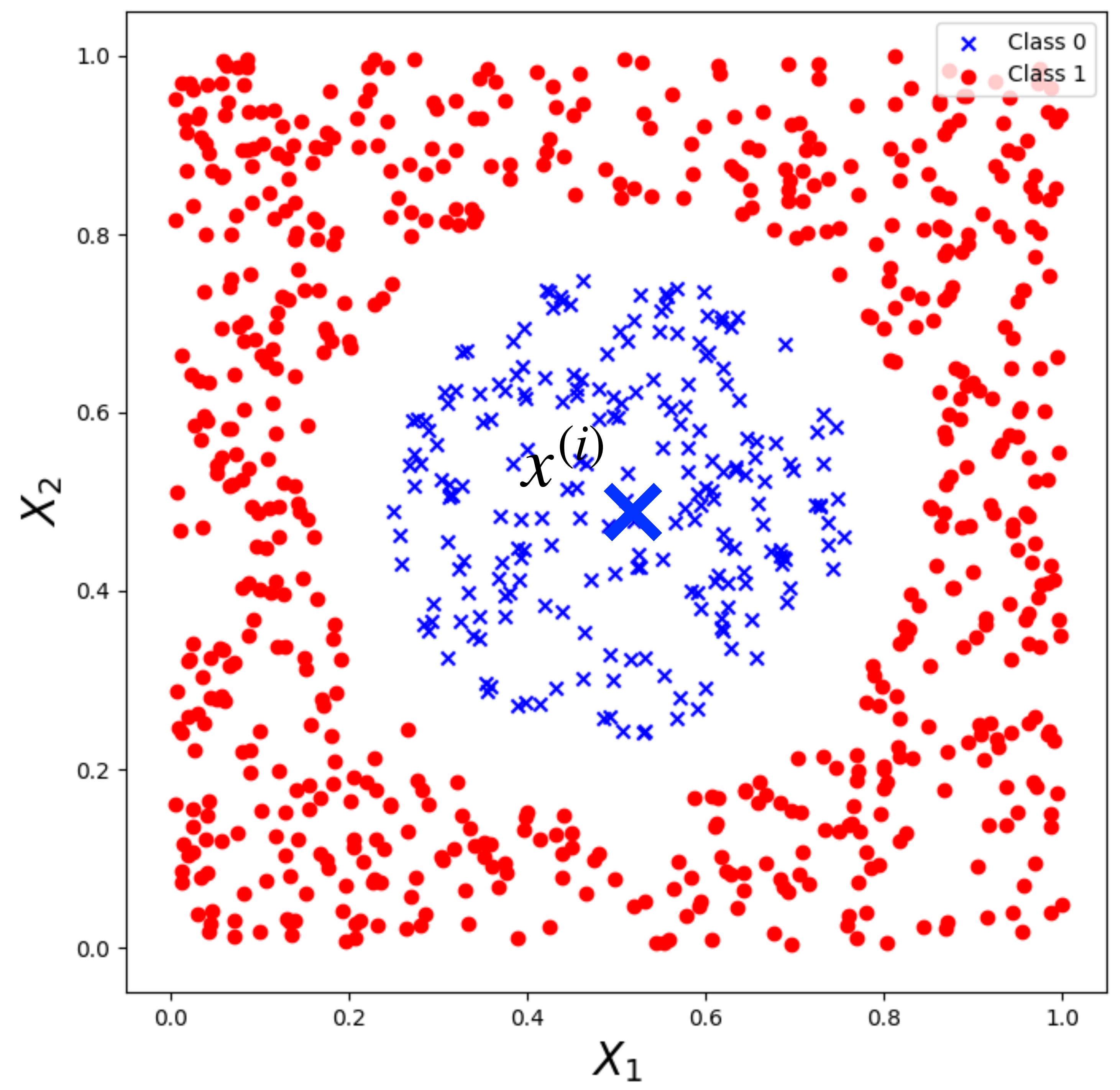
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





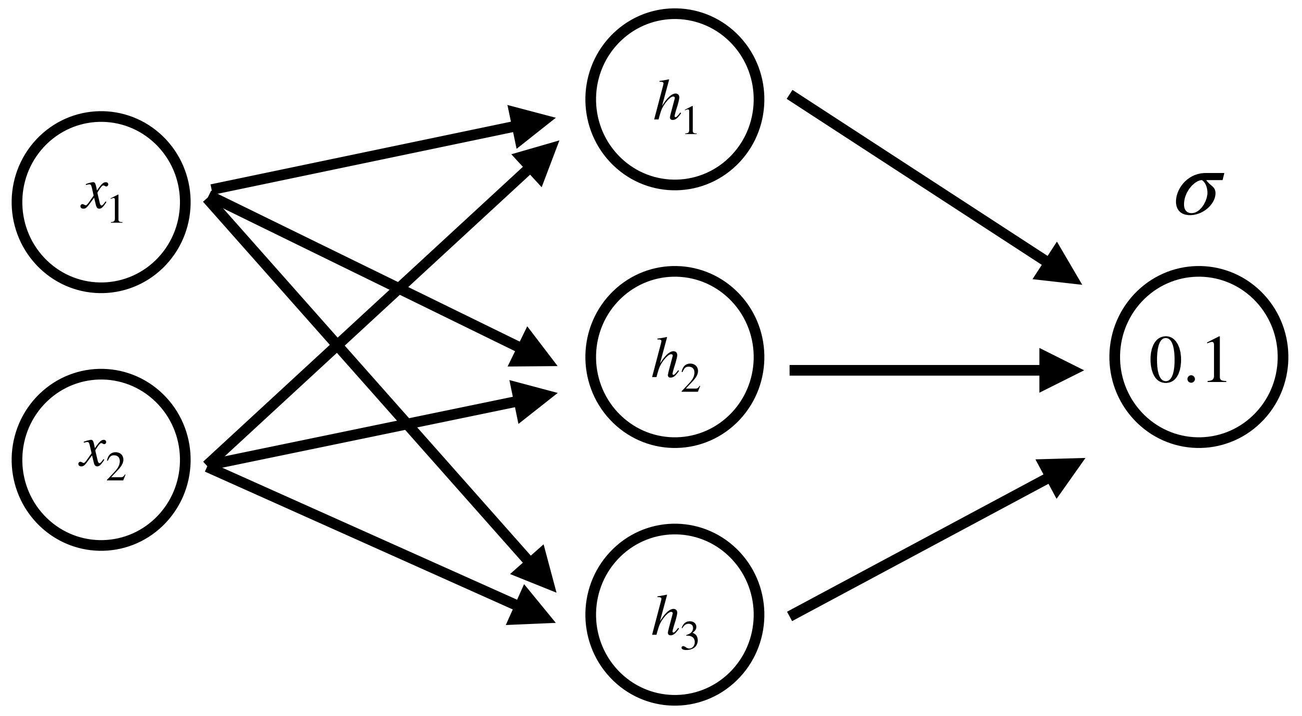
Loss function: $L_{CE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} - y^{(i)}\log(\lambda^{(i)}) - (1 - y^{(i)})\log(1 - \lambda^{(i)})$

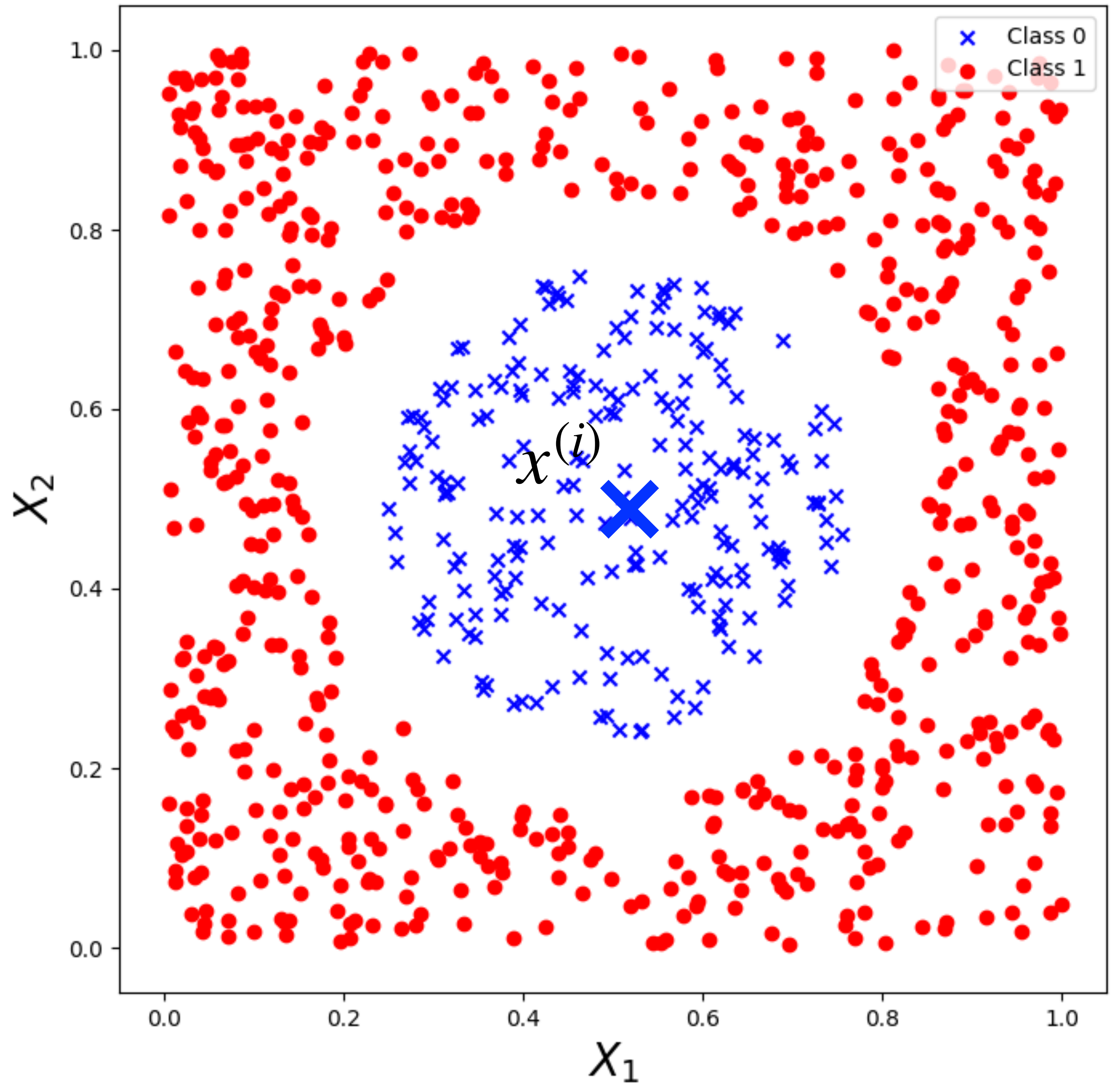
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L_{CE}(\theta) = \frac{1}{n} \sum_{i=1}^{n} -y^{(i)} \log(\lambda^{(i)}) - (1 - y^{(i)}) \log(1 - \lambda^{(i)})$

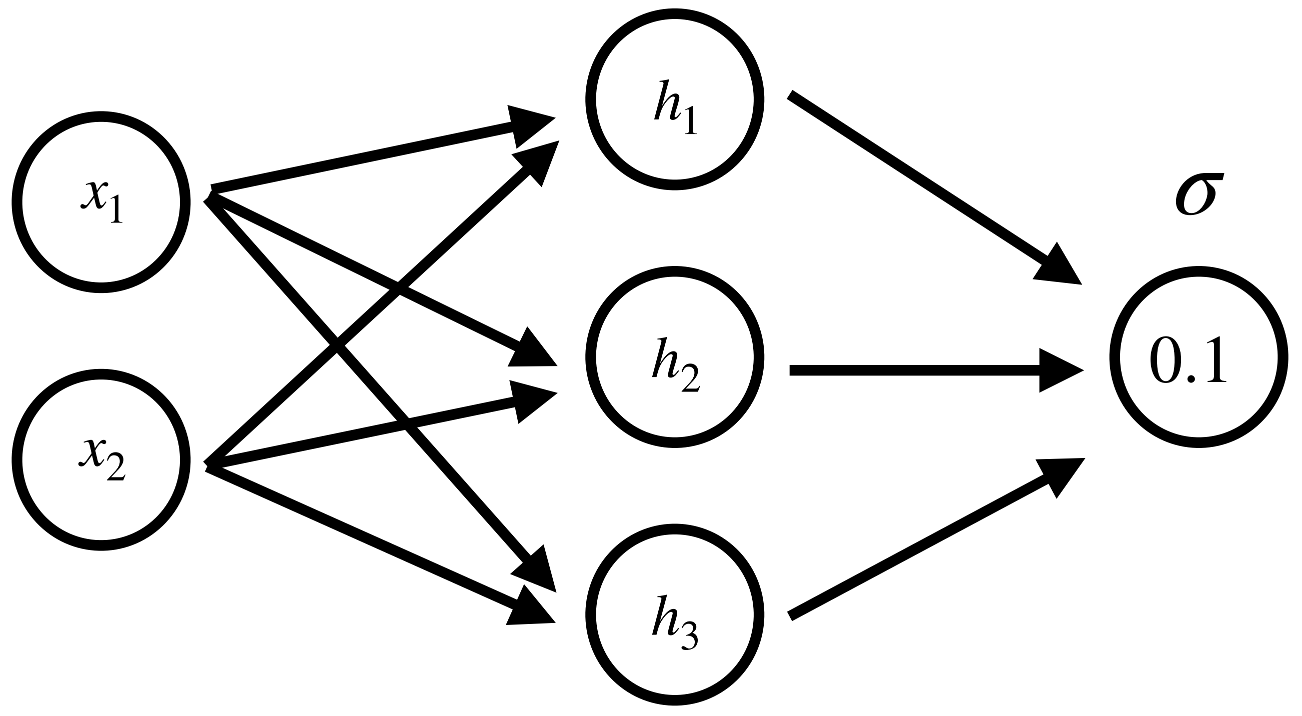
$$\hat{y}^{(i)} = argmax_y \left(p_{model}(y \,|\, \lambda^{(i)}) \right)$$

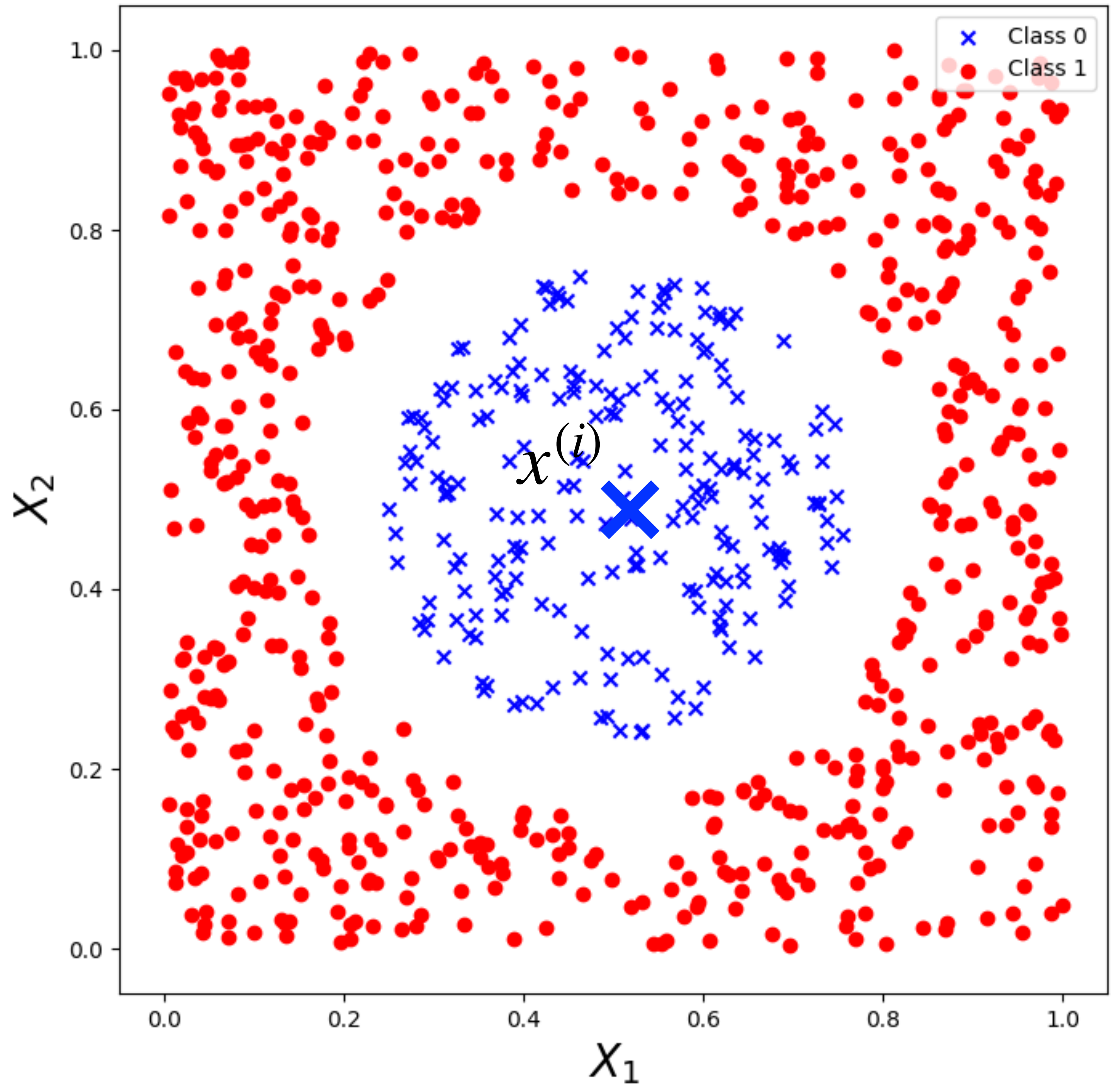
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L_{CE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} -y^{(i)}\log(\lambda^{(i)}) - (1-y^{(i)})\log(1-\lambda^{(i)})$

$$\hat{y}^{(i)} = argmax_y\left(p_{model}(y\,|\,\lambda^{(i)})\right)$$

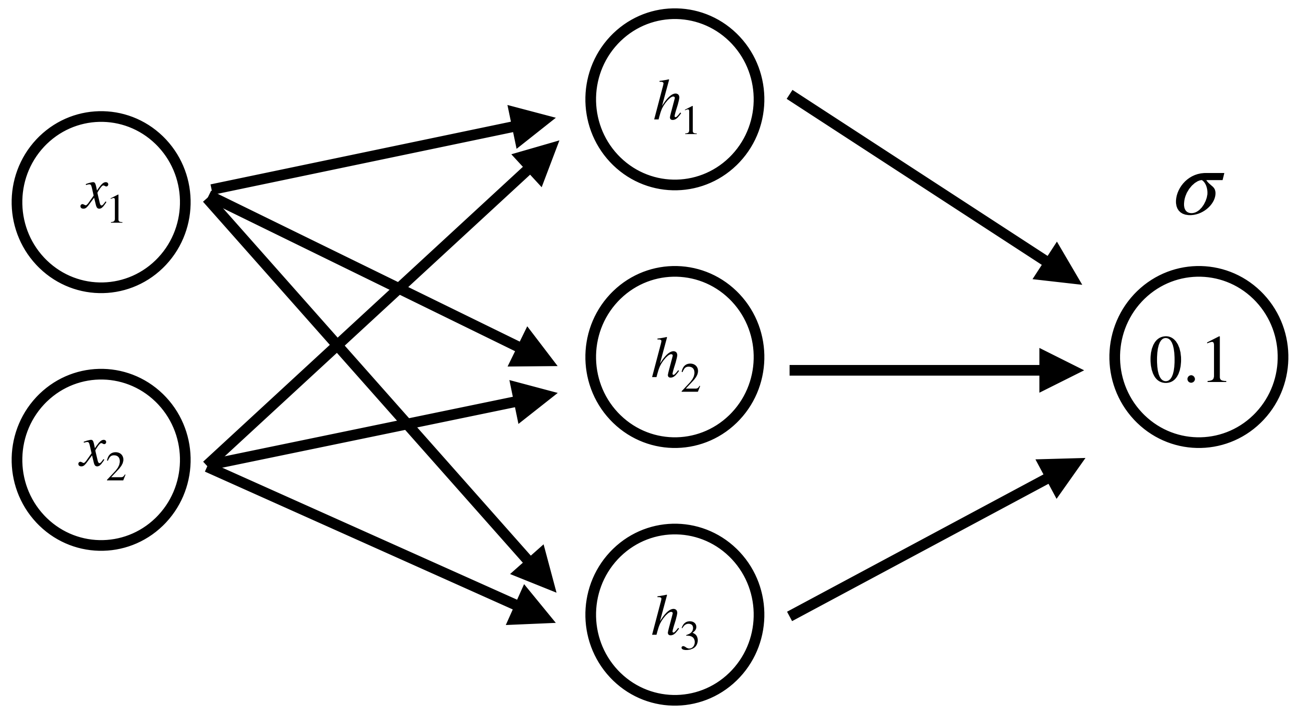
$$p_{model}(y\,|\,\lambda^{(i)} = 0.1) = (\underset{y=0}{0.9},\ \underset{y=1}{0.1})$$

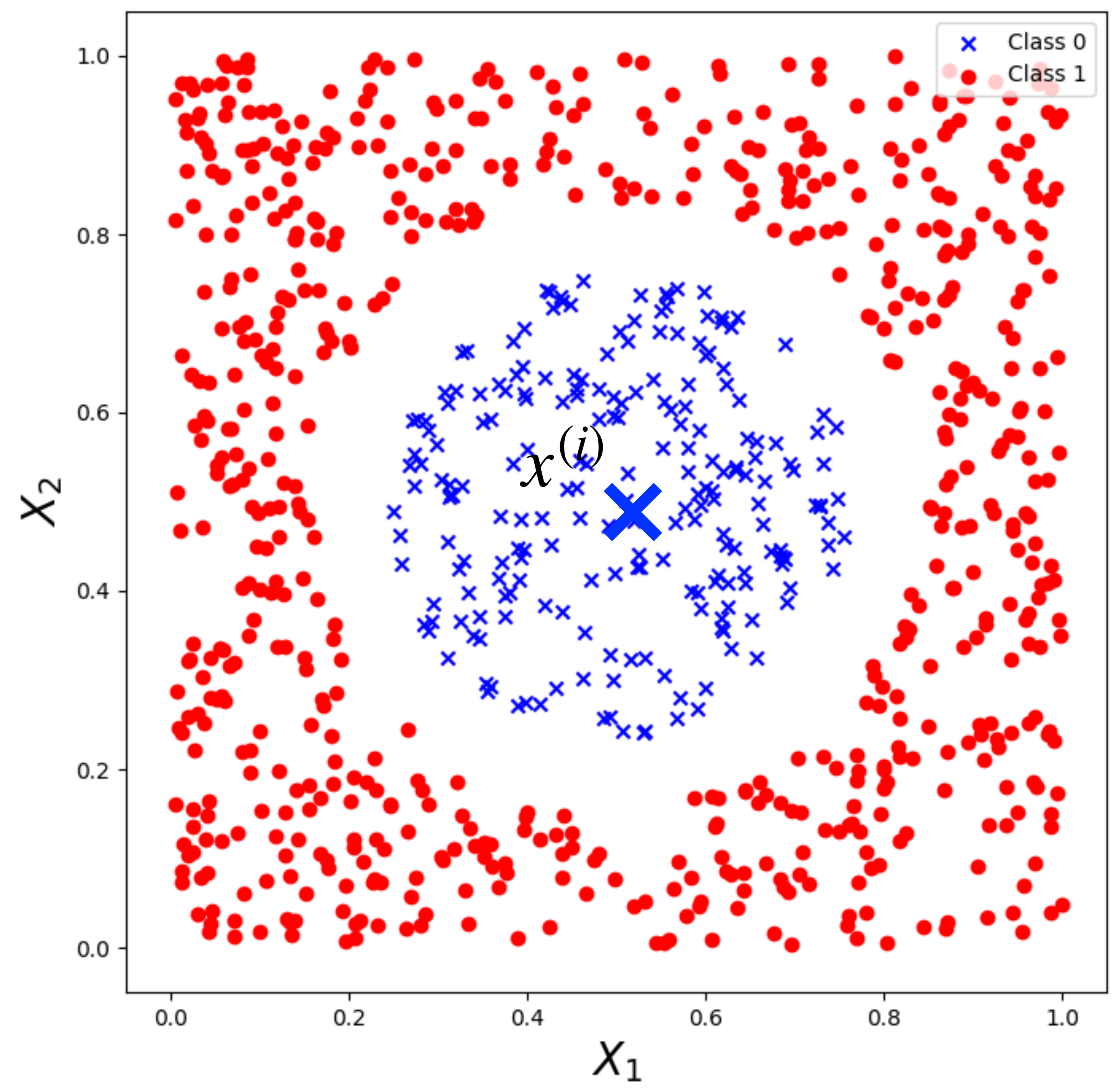
Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	?

Building the Loss Function: Binary Classification

Training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$

$f(x^{(i)}; \theta) = \lambda^{(i)}$ probability that an input $x^{(i)}$ belongs to class 1





Loss function: $L_{CE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} - y^{(i)}\log(\lambda^{(i)}) - (1 - y^{(i)})\log(1 - \lambda^{(i)})$

$$\hat{y}^{(i)} = argmax_y\left((0.9{,}0.1)\right) = 0$$

Input x	Class y	Output λ	Error l	Predicted $\hat{y}$
(0.5, 0.5)	0	0.1	-log(0.9)	0

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_{y} p_{data}(y) \log(p_{model}(y))^{*}$$

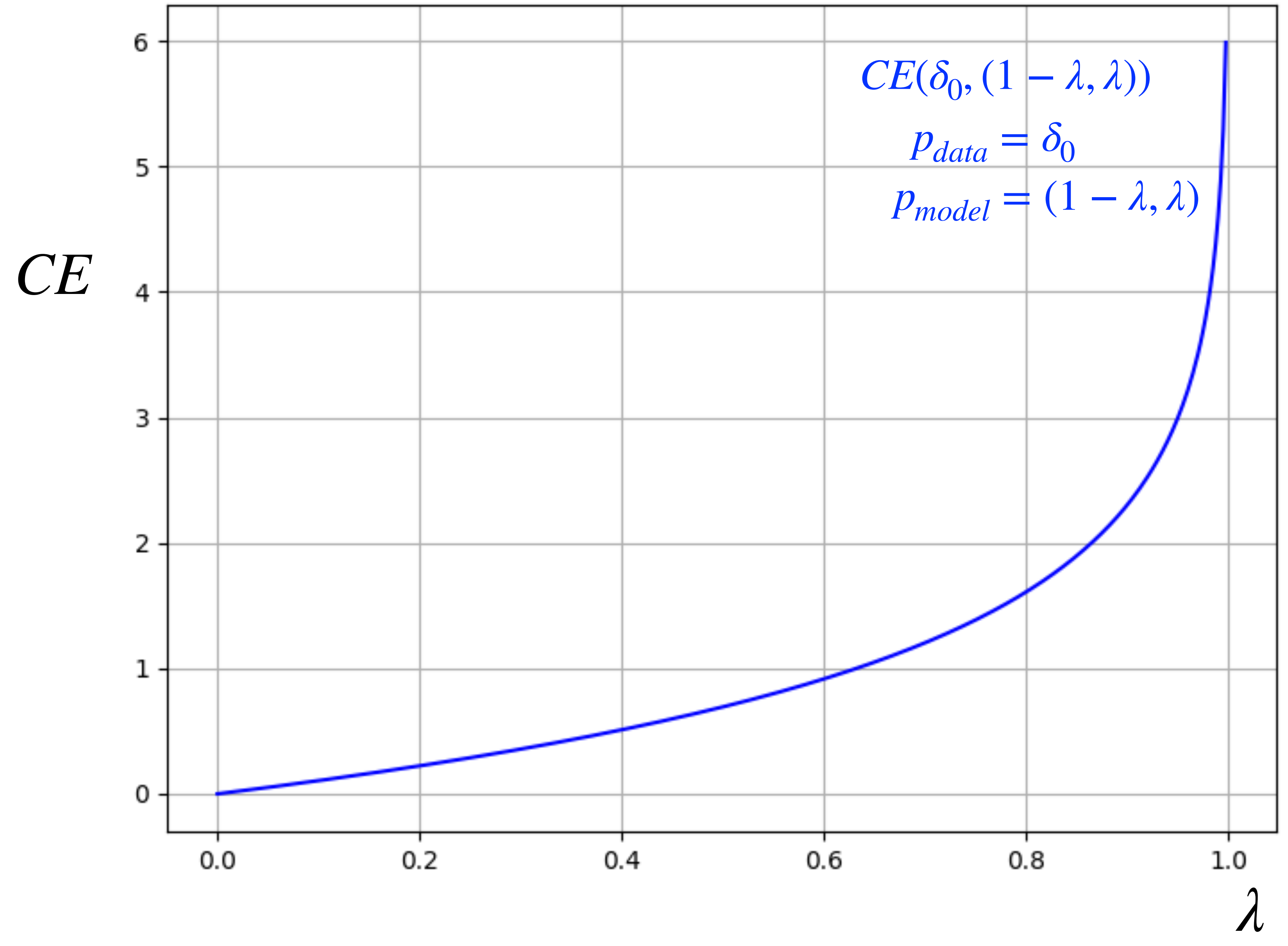
*Colah's blog: Visual Information Theory

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_y p_{data}(y) \log(p_{model}(y))^*$$

The loss L estimates $CE(p_{data}, p_{model})$



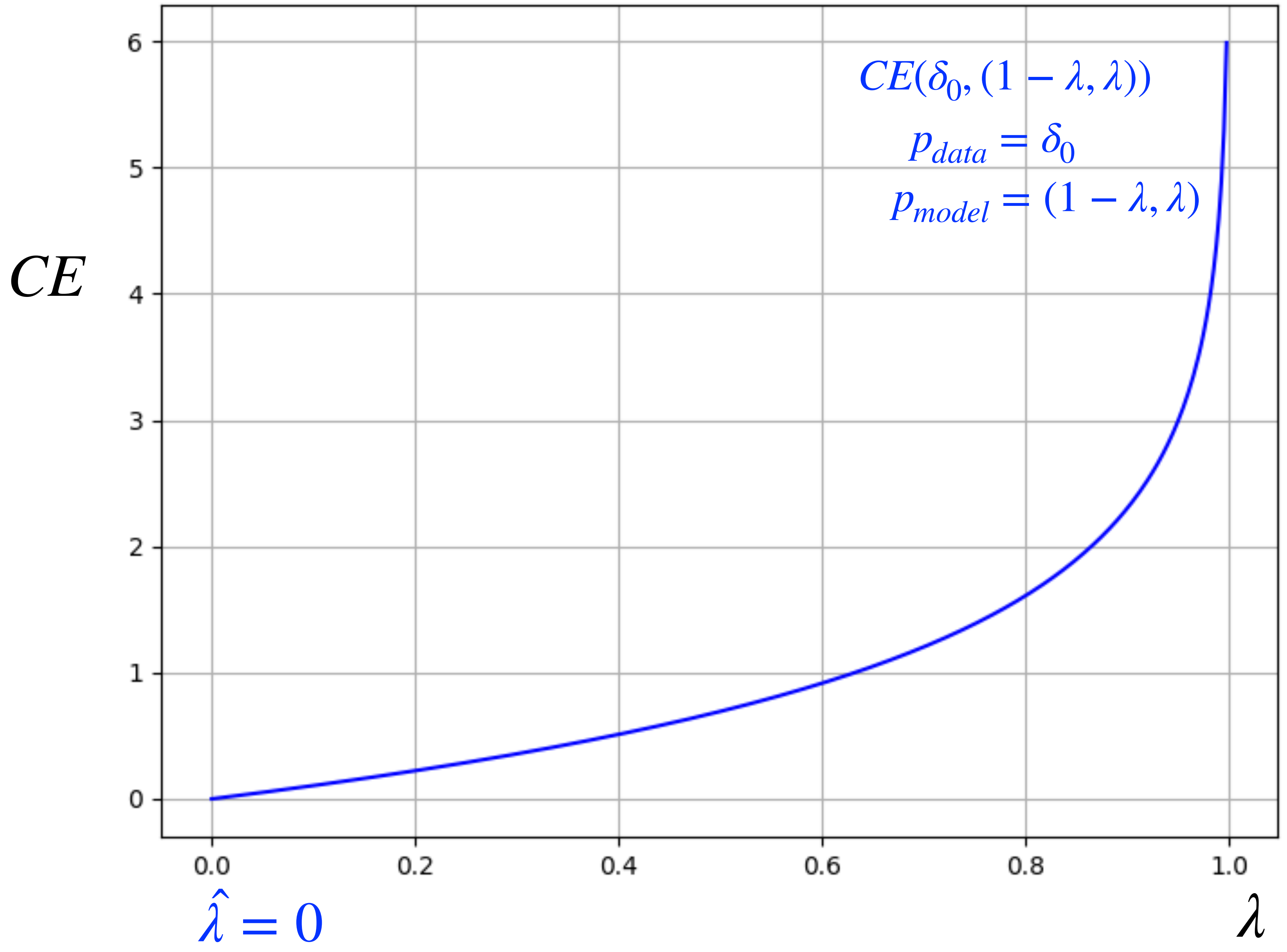
*Colah's blog: Visual Information Theory

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_y p_{data}(y)\ \log(p_{model}(y))^*$$

The loss L estimates $CE(p_{data}, p_{model})$



Optimal output

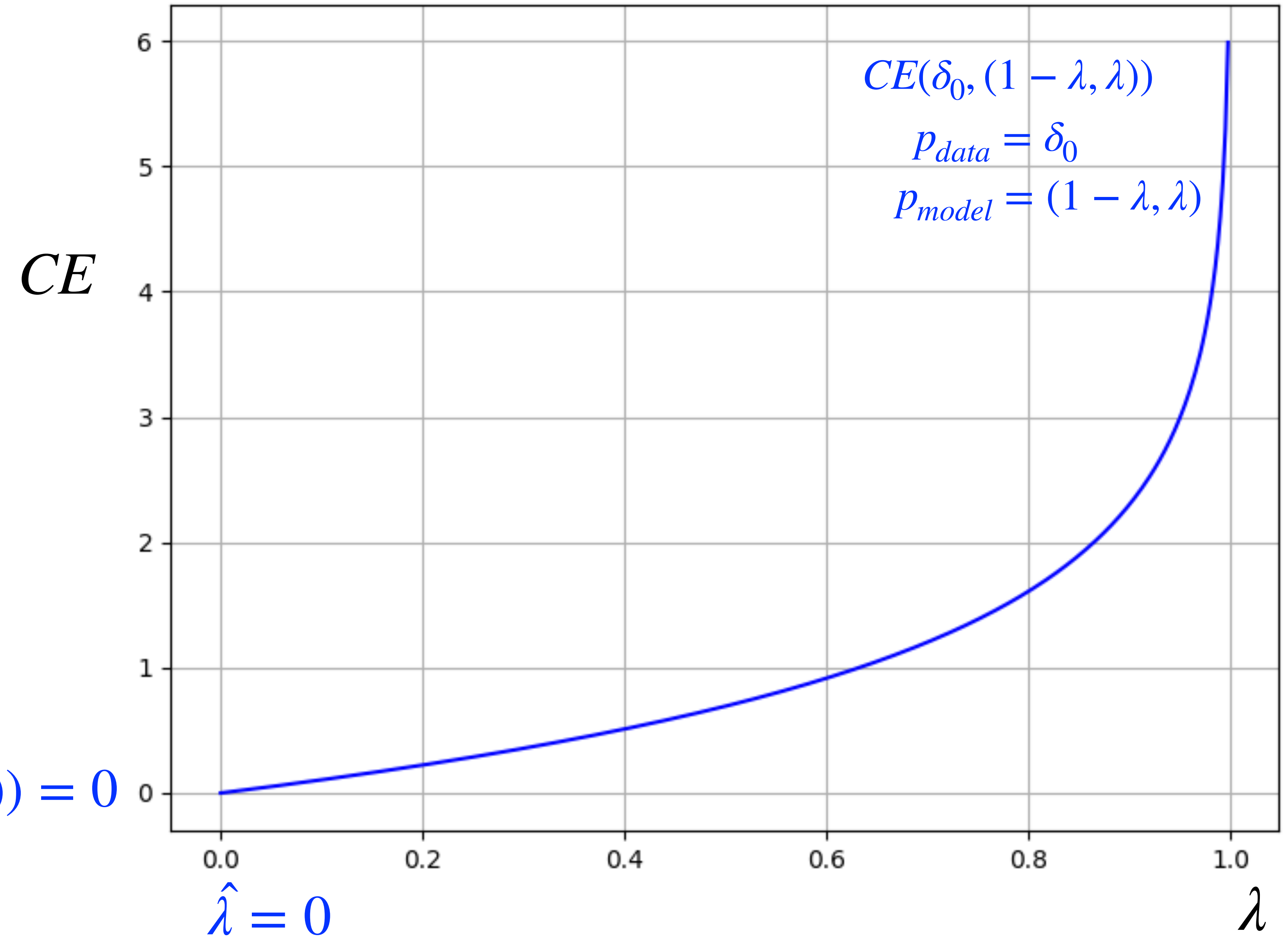
*Colah's blog: Visual Information Theory

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_{y} p_{data}(y)\ \log(p_{model}(y))^{*}$$

The loss L estimates $CE(p_{data}, p_{model})$



$CE(\delta_0, (1{,}0)) = 0$

$\hat{\lambda} = 0$

$\hat{p}_{model} = (1{,}0) = \delta_0 = p_{data}$

Optimal output

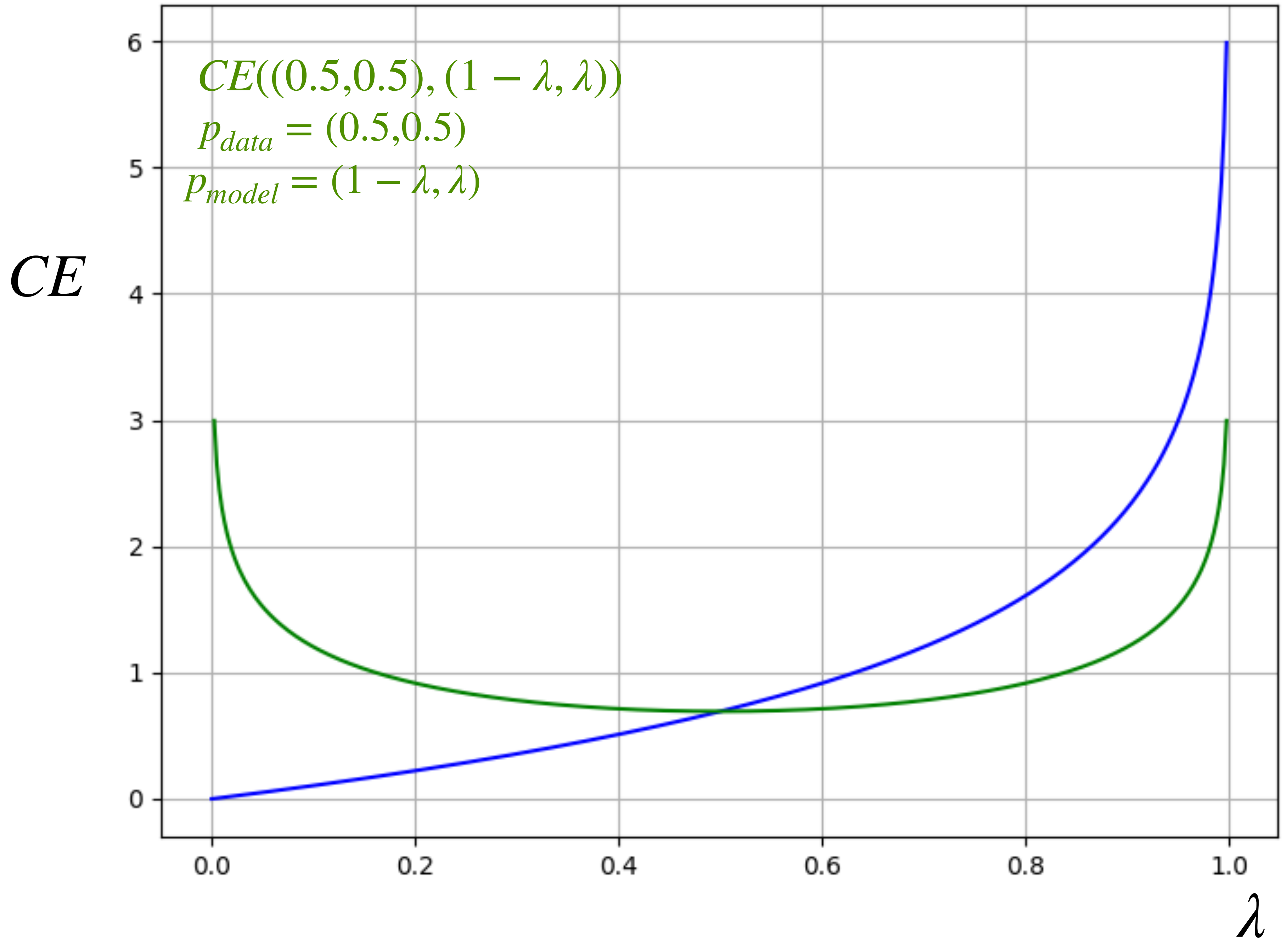
*Colah's blog: Visual Information Theory

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_{y} p_{data}(y)\ \log(p_{model}(y))^{*}$$

The loss L estimates $CE(p_{data}, p_{model})$



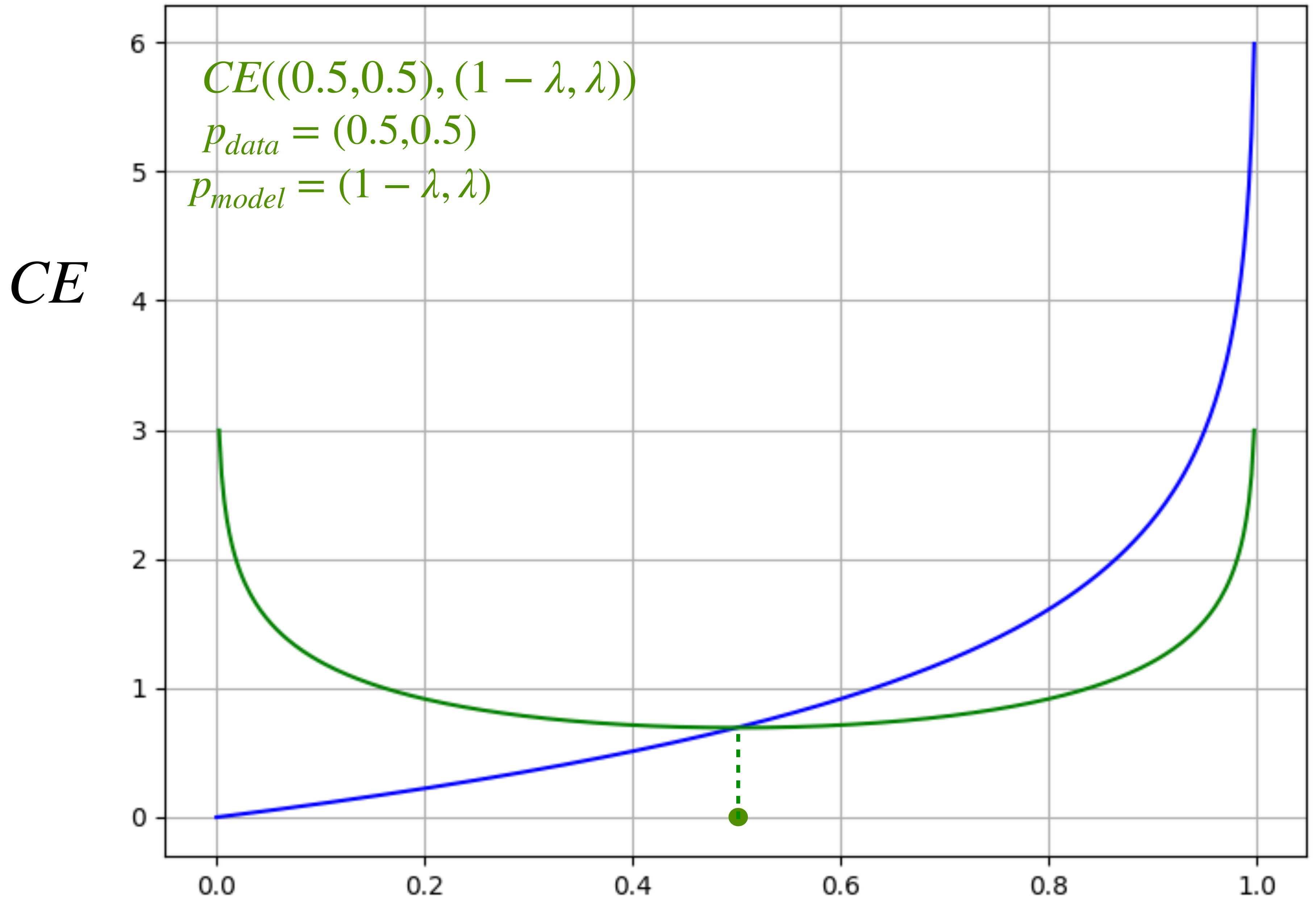
*Colah's blog: Visual Information Theory

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_{y} p_{data}(y)\ \log(p_{model}(y))^{*}$$

The loss L estimates $CE(p_{data}, p_{model})$



$\hat{\lambda} = 0.5$ λ

$\hat{p}_{model} = (0.5, 0.5) = p_{data}$

Optimal output

*Colah's blog: Visual Information Theory

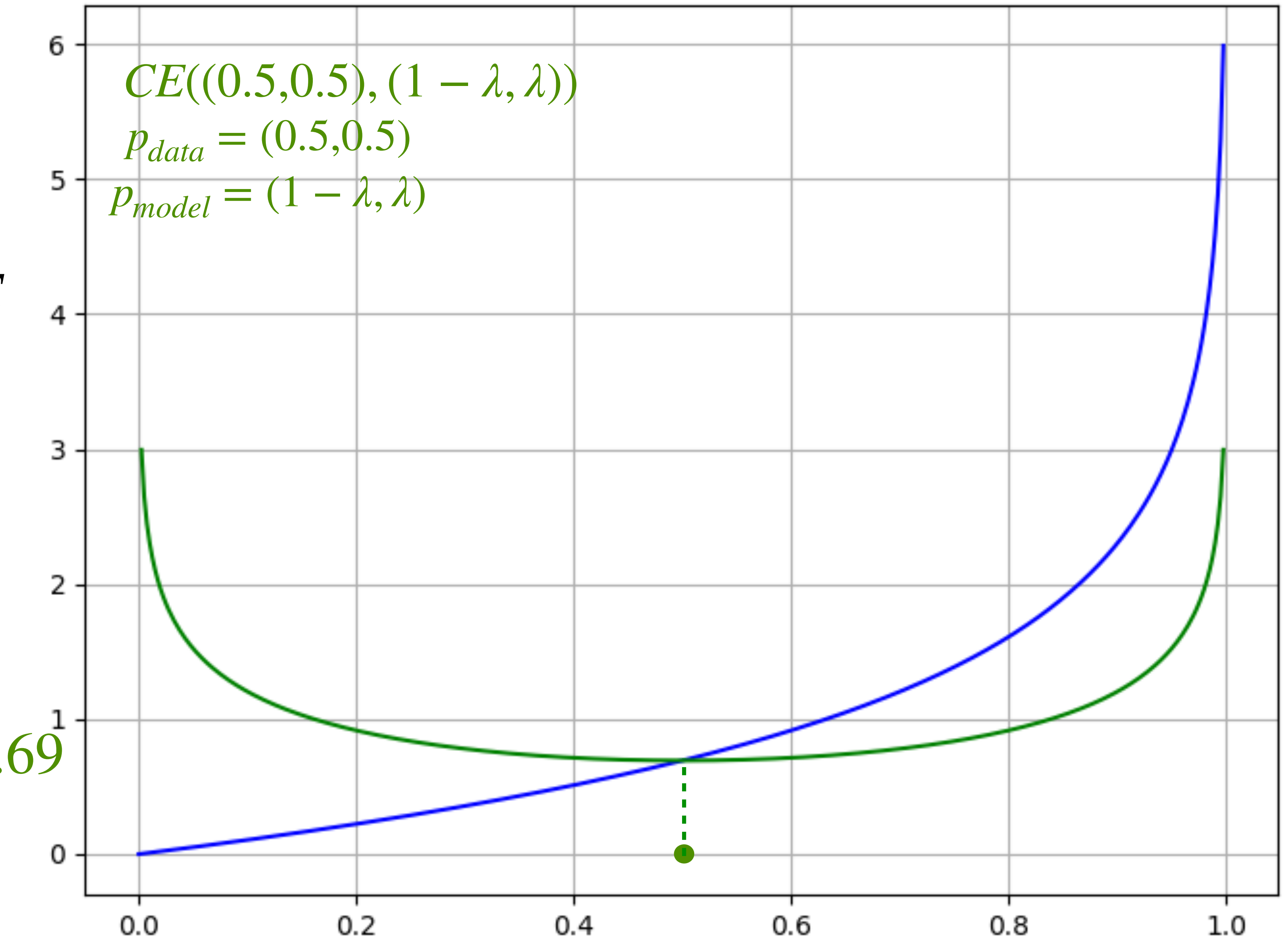
Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

$$CE(p_{data}, p_{model}) = -\sum_{y} p_{data}(y)\ \log(p_{model}(y))^{*}$$

The loss L estimates $CE(p_{data}, p_{model})$

CE



$CE((0.5{,}0.5), (0.5{,}0.5)) = 0.69$

λ

$\hat{\lambda} = 0.5$

Minimum loss is not zero!

$\hat{p}_{model} = (0.5, 0.5) = p_{data}$

Optimal output

*Colah's blog: Visual Information Theory

Cross Entropy

Given the ground-truth distribution p_{data} and an approximation p_{model}

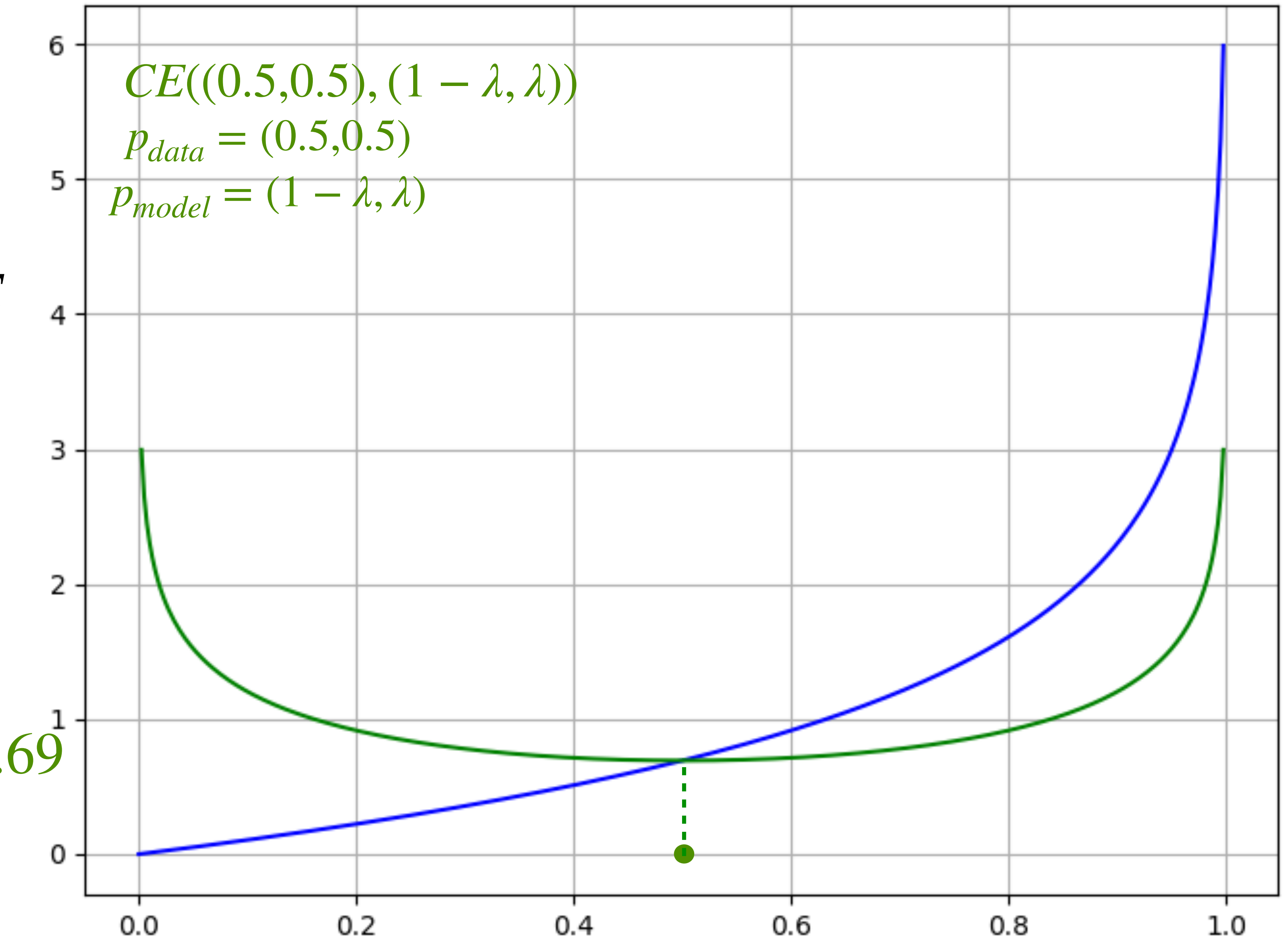
$$CE(p_{data}, p_{model}) = -\sum_{y} p_{data}(y) \log(p_{model}(y))^{*}$$

The loss L estimates $CE(p_{data}, p_{model})$

A prefect model $p_{data} = p_{model}$

$$CE(p_{data}, p_{model}) = H(p_{data}) \text{ entropy}$$

CE



$CE((0.5{,}0.5), (0.5{,}0.5)) = 0.69$

Minimum loss is not zero!

$\hat{\lambda} = 0.5$ λ

$\hat{p}_{model} = (0.5, 0.5) = p_{data}$

Optimal output

*Colah's blog: Visual Information Theory

Maximum Likelihood

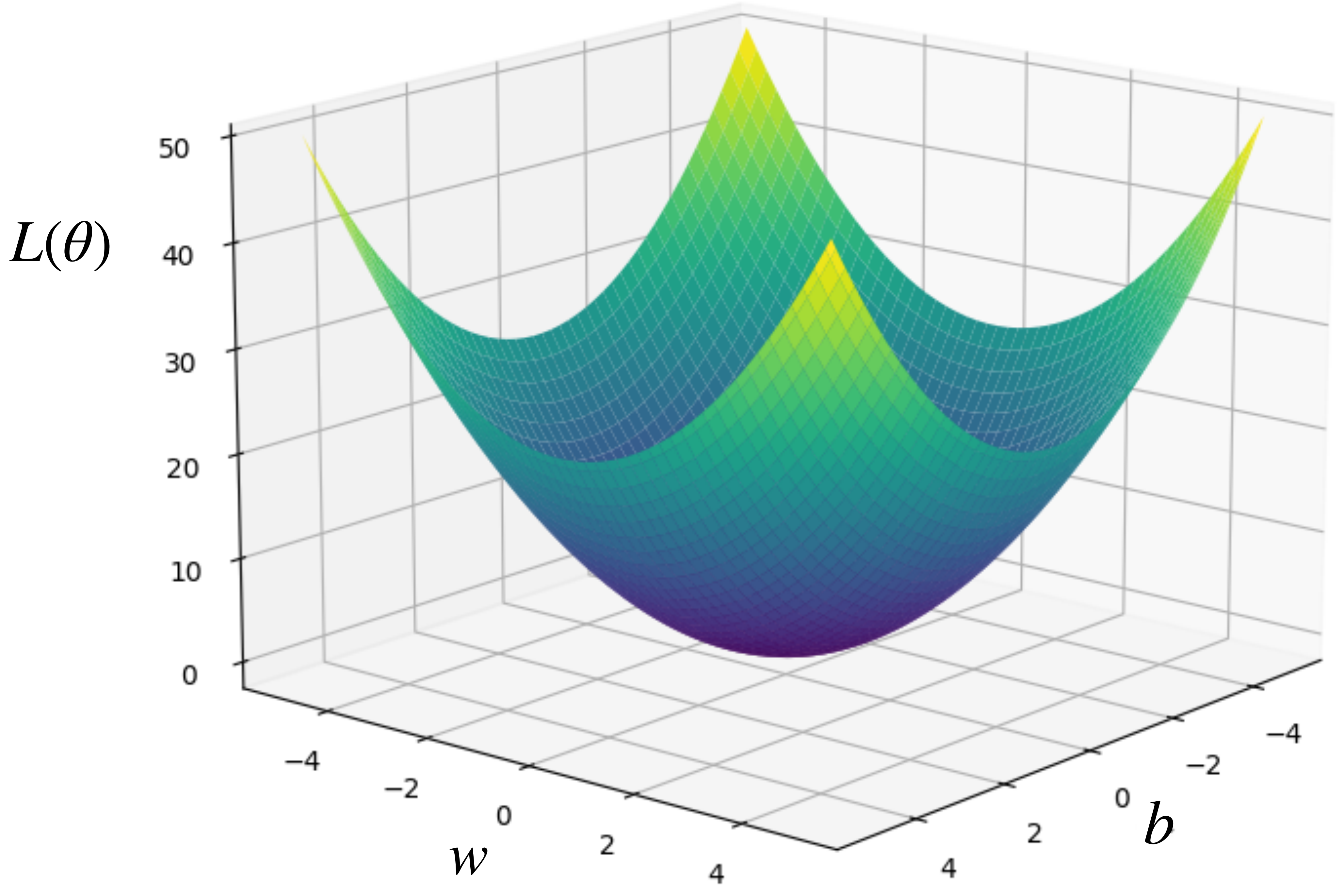
- Open the notebook in Brightspace → Gradient-based Optimization → Maximum Likelihood - Bandits
- Work in group of 2 or 3
- Go through the notebook and answer the questions with your peers
- If you have doubts, raise your hand or come to me
- Around 15 minutes

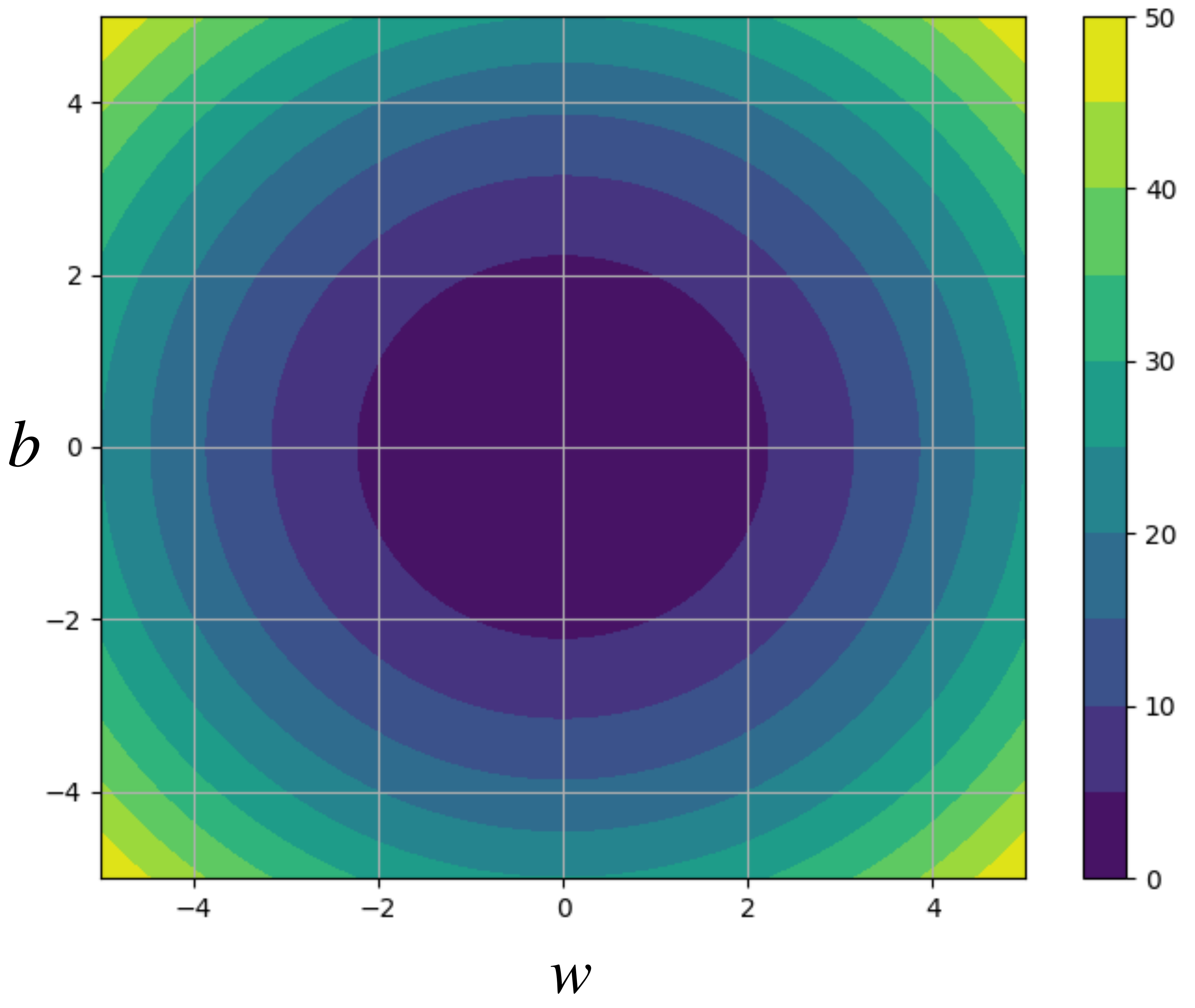
Discussion next Monday

Loss Function Minimization

Solve the minimization problem $\hat{\theta} = argmin_{\theta} L(\theta)$ through gradient descent.

$\theta = \{w, b\}$

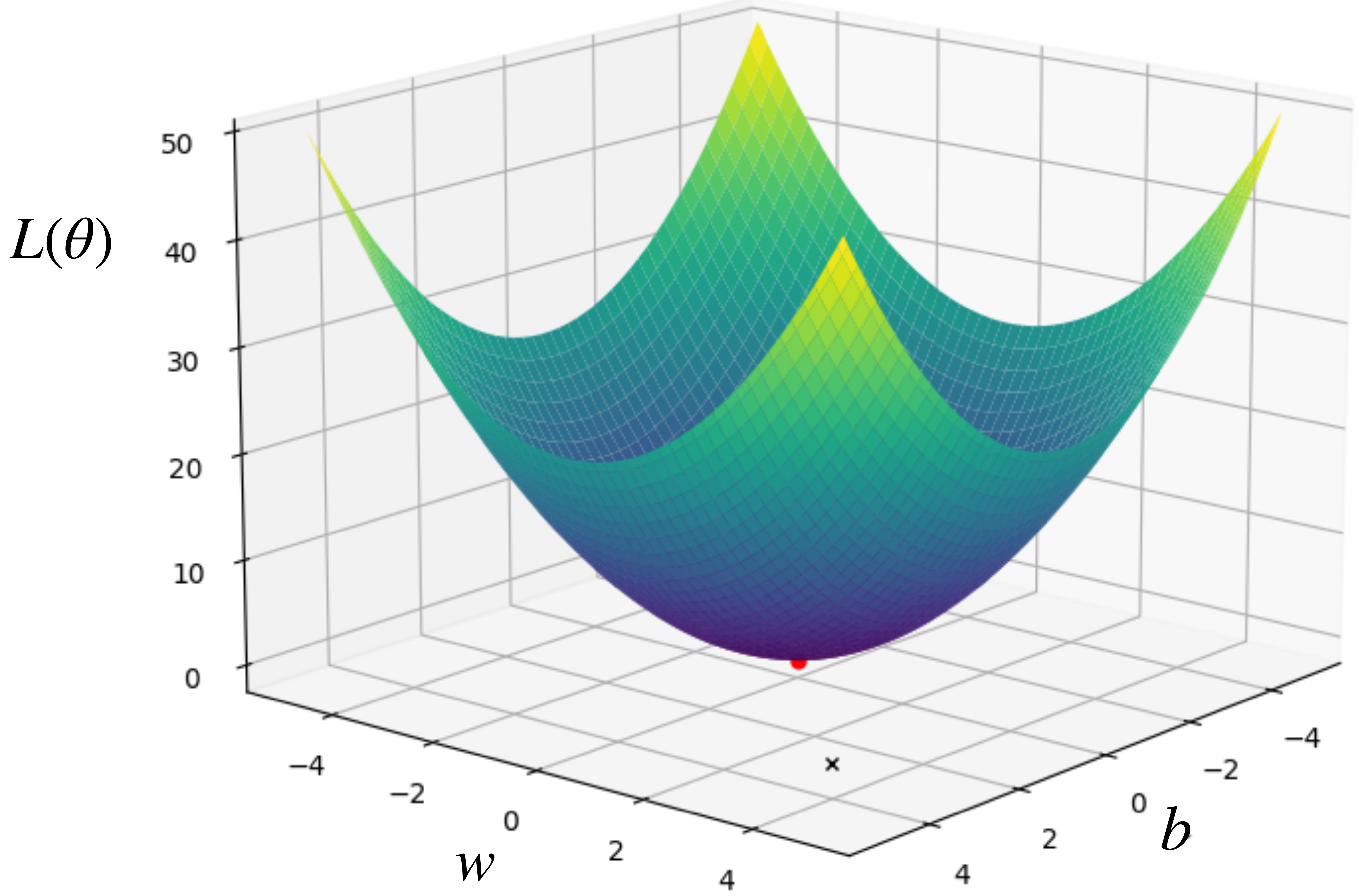


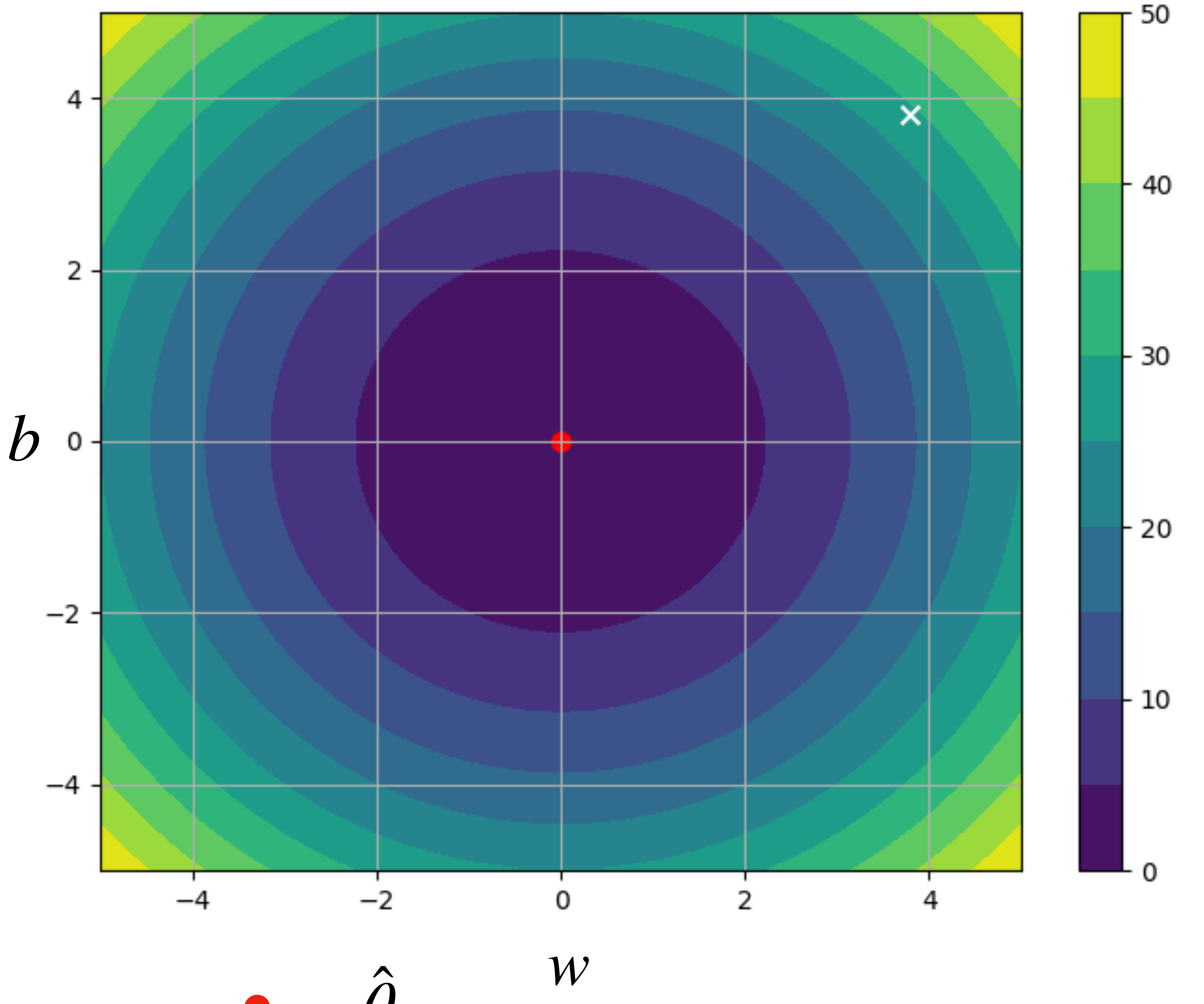


Loss Function Minimization

Solve the minimization problem $\hat{\theta} = argmin_{\theta} L(\theta)$ through gradient descent.

$\theta = \{w, b\}$





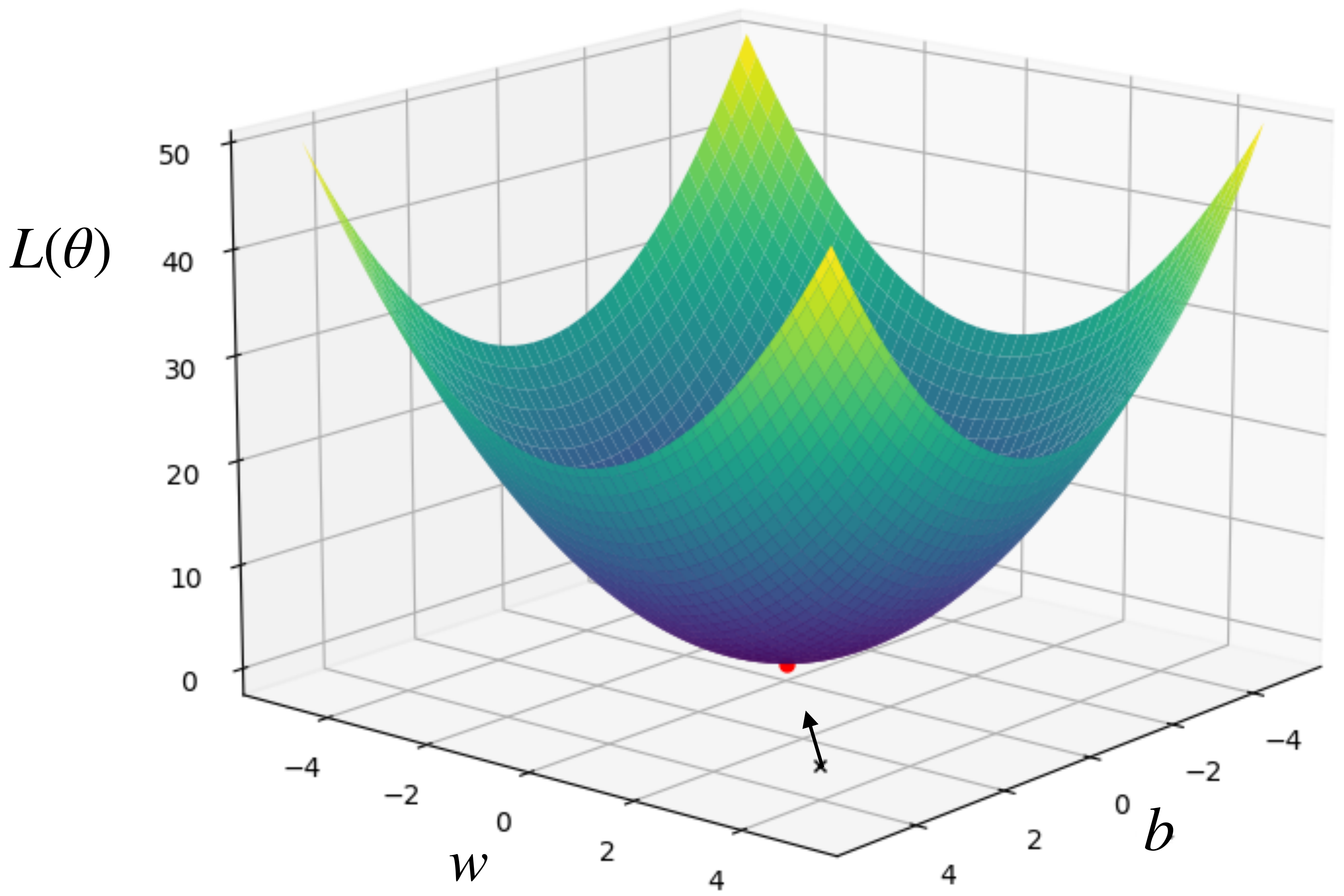
● $\hat{\theta}$

✕ $\theta_0 = (w_0, b_0)$

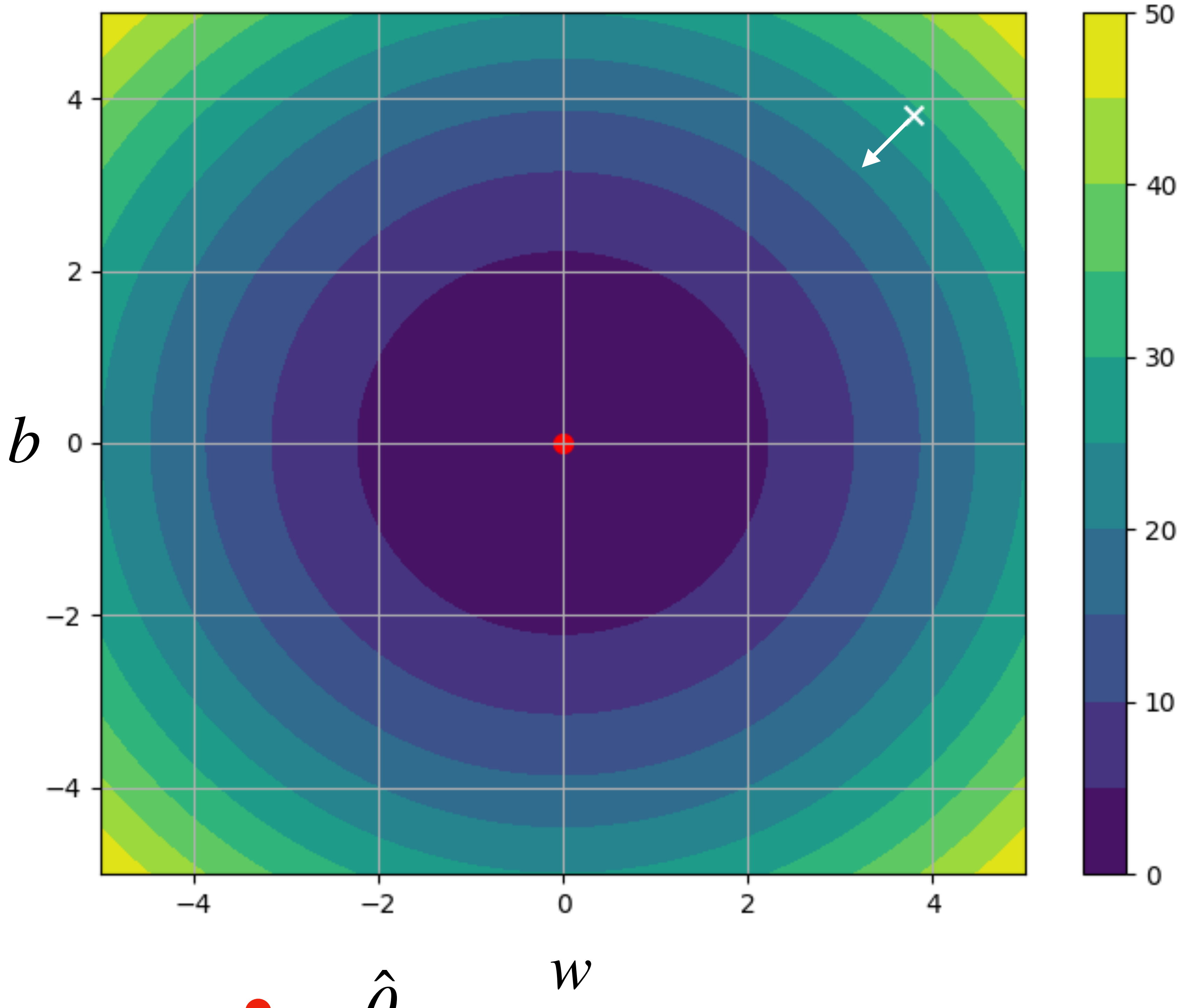
Loss Function Minimization

Solve the minimization problem $\hat{\theta} = argmin_{\theta} L(\theta)$ through gradient descent.

$\theta = \{w, b\}$



$$-\nabla L(\theta_0) = -\left(\frac{\partial L}{\partial w}(\theta_0), \frac{\partial L}{\partial b}(\theta_0)\right)$$



● $\hat{\theta}$

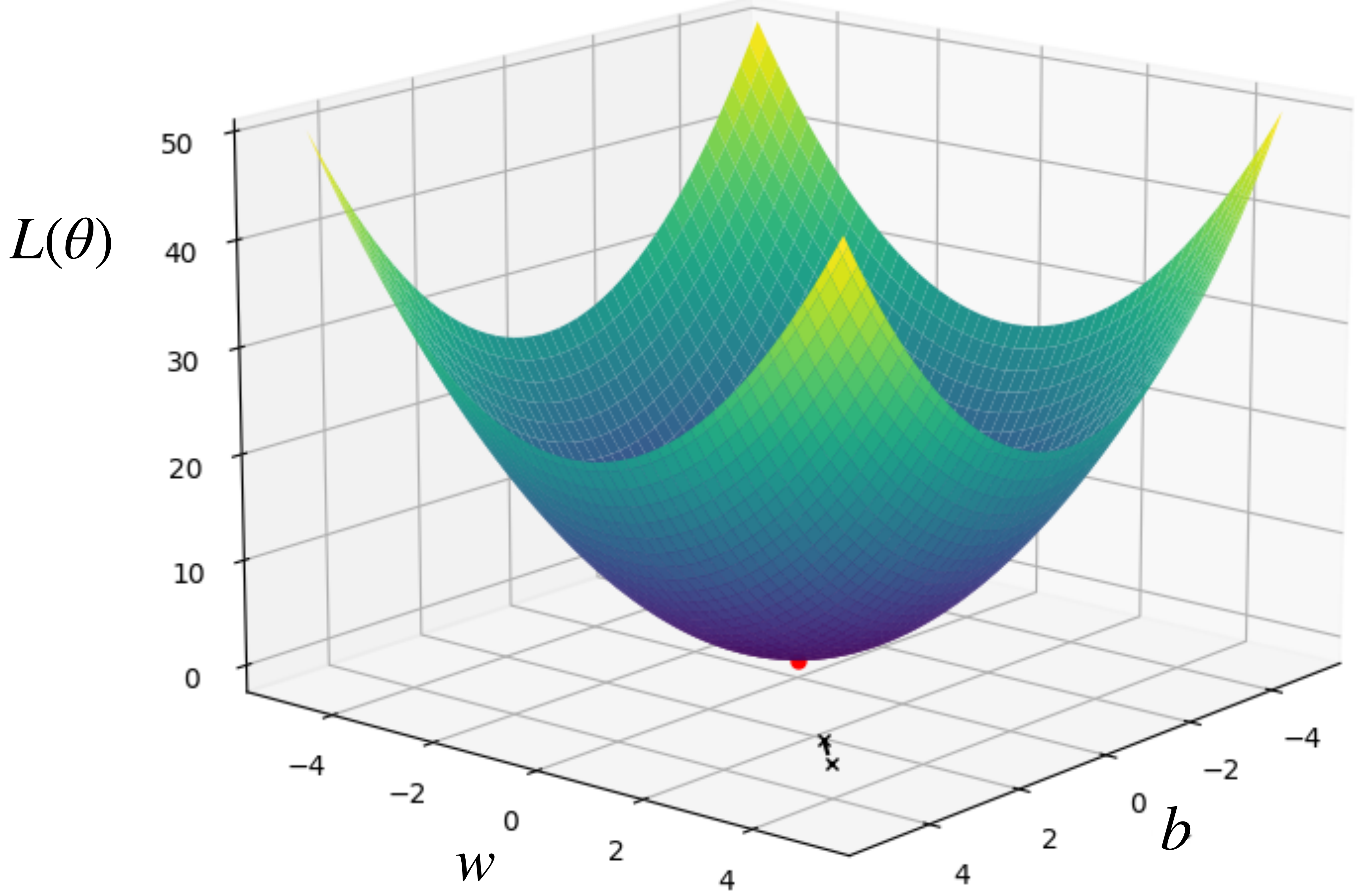
✕ $\theta_0 = (w_0, b_0)$

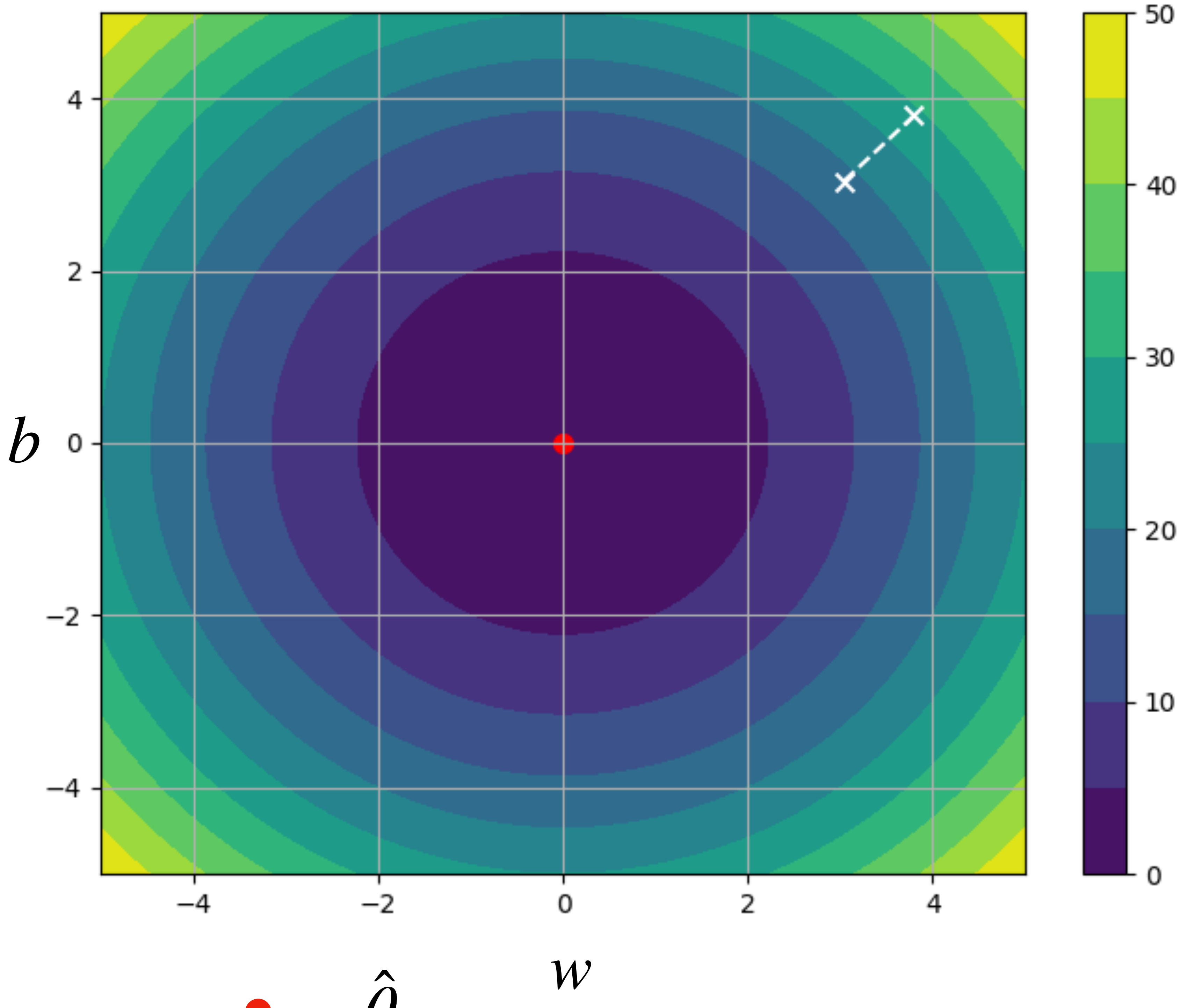
↖ $-\alpha \nabla L(\theta_0)$

Loss Function Minimization

Solve the minimization problem $\hat{\theta} = argmin_{\theta} L(\theta)$ through gradient descent.

$\theta = \{w, b\}$





$$\theta_1 = \theta_0 - \alpha \nabla L(\theta_0)$$

● $\hat{\theta}$

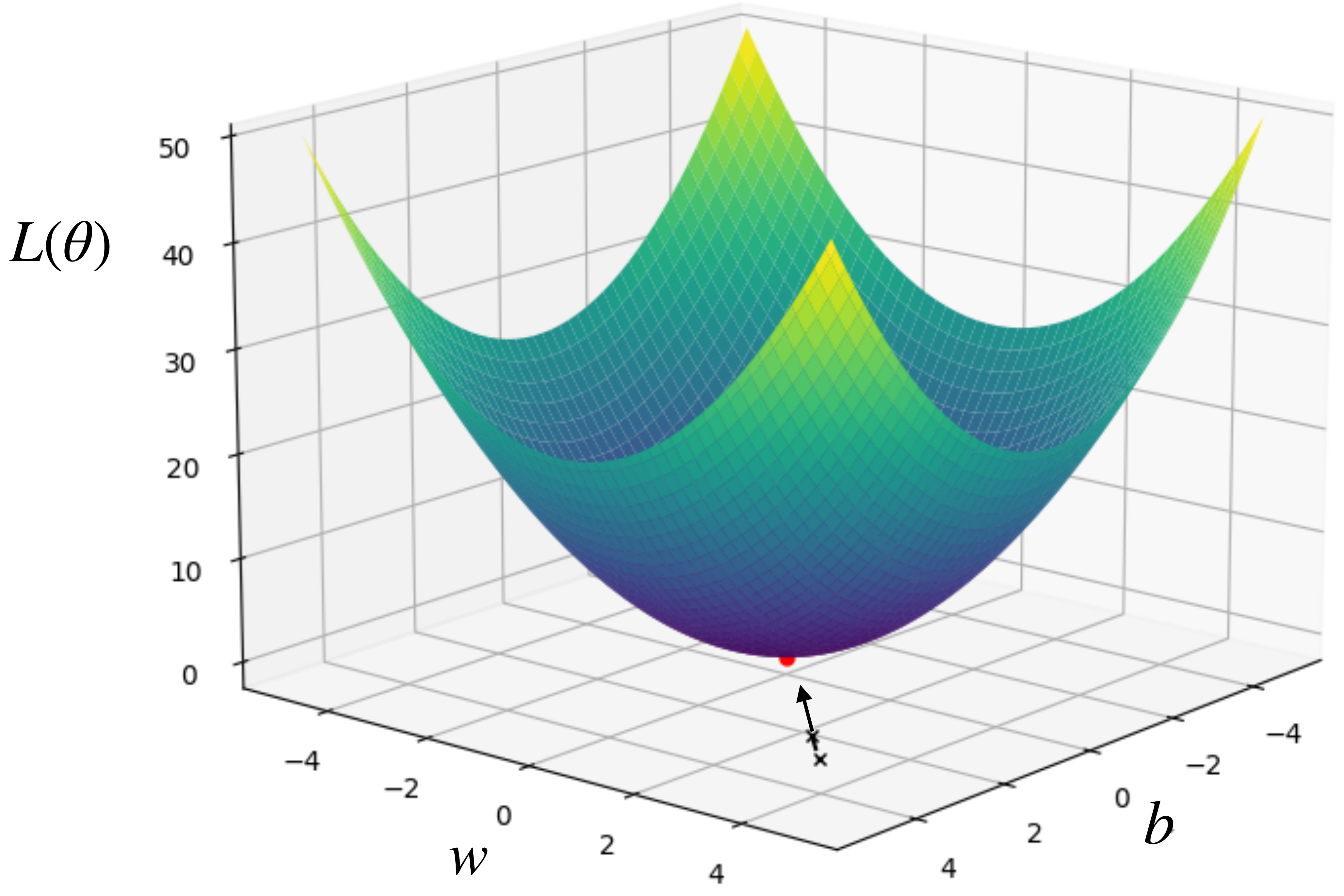
✕ $\theta_1 = (w_1, b_1)$

↑ $-\alpha \nabla L(\theta_0)$

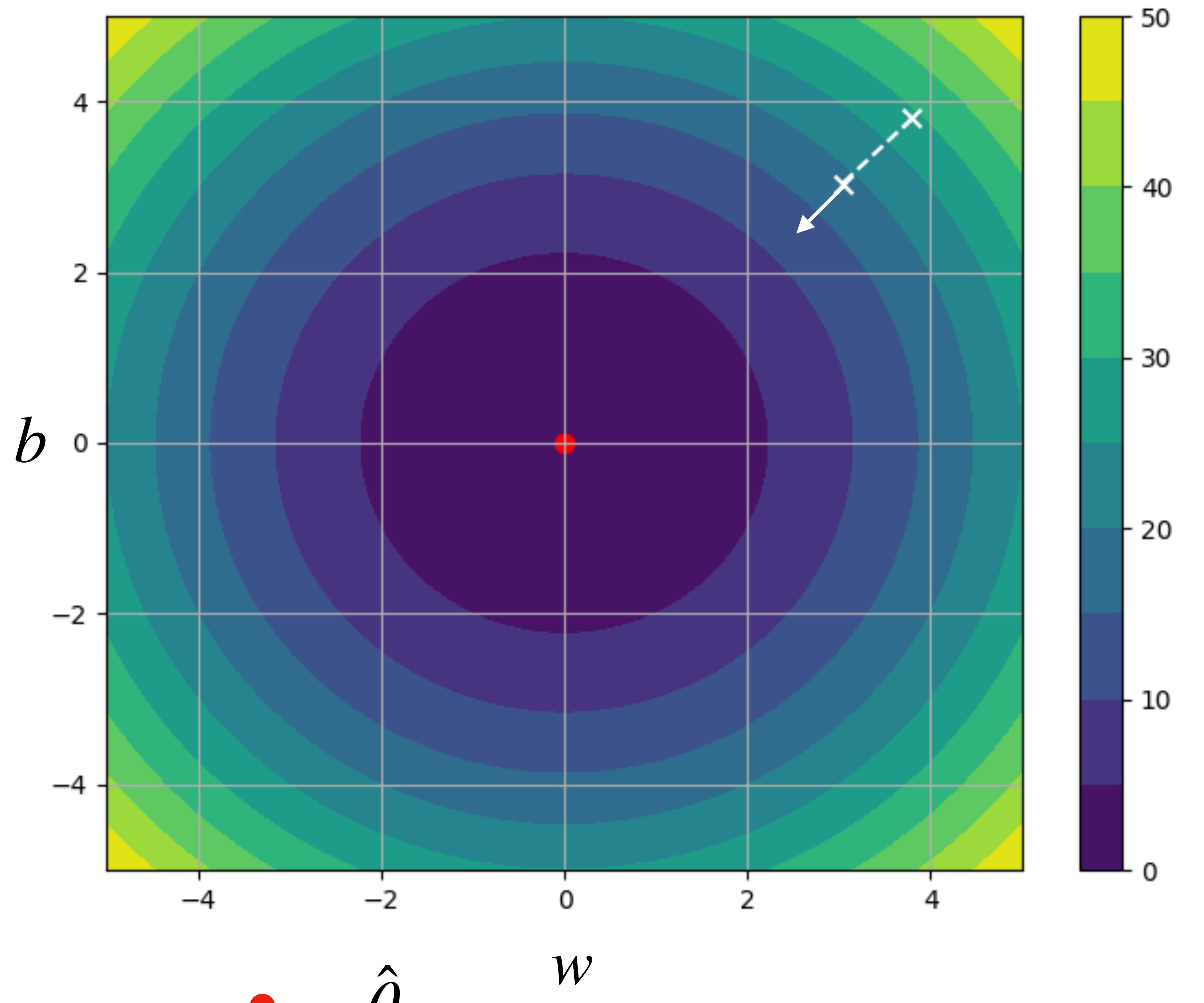
Loss Function Minimization

Solve the minimization problem $\hat{\theta} = argmin_{\theta} L(\theta)$ through gradient descent.

$\theta = \{w, b\}$



$$-\nabla L(\theta_1) = -\left(\frac{\partial L}{\partial w}(\theta_1), \frac{\partial L}{\partial b}(\theta_1)\right)$$



● $\hat{\theta}$

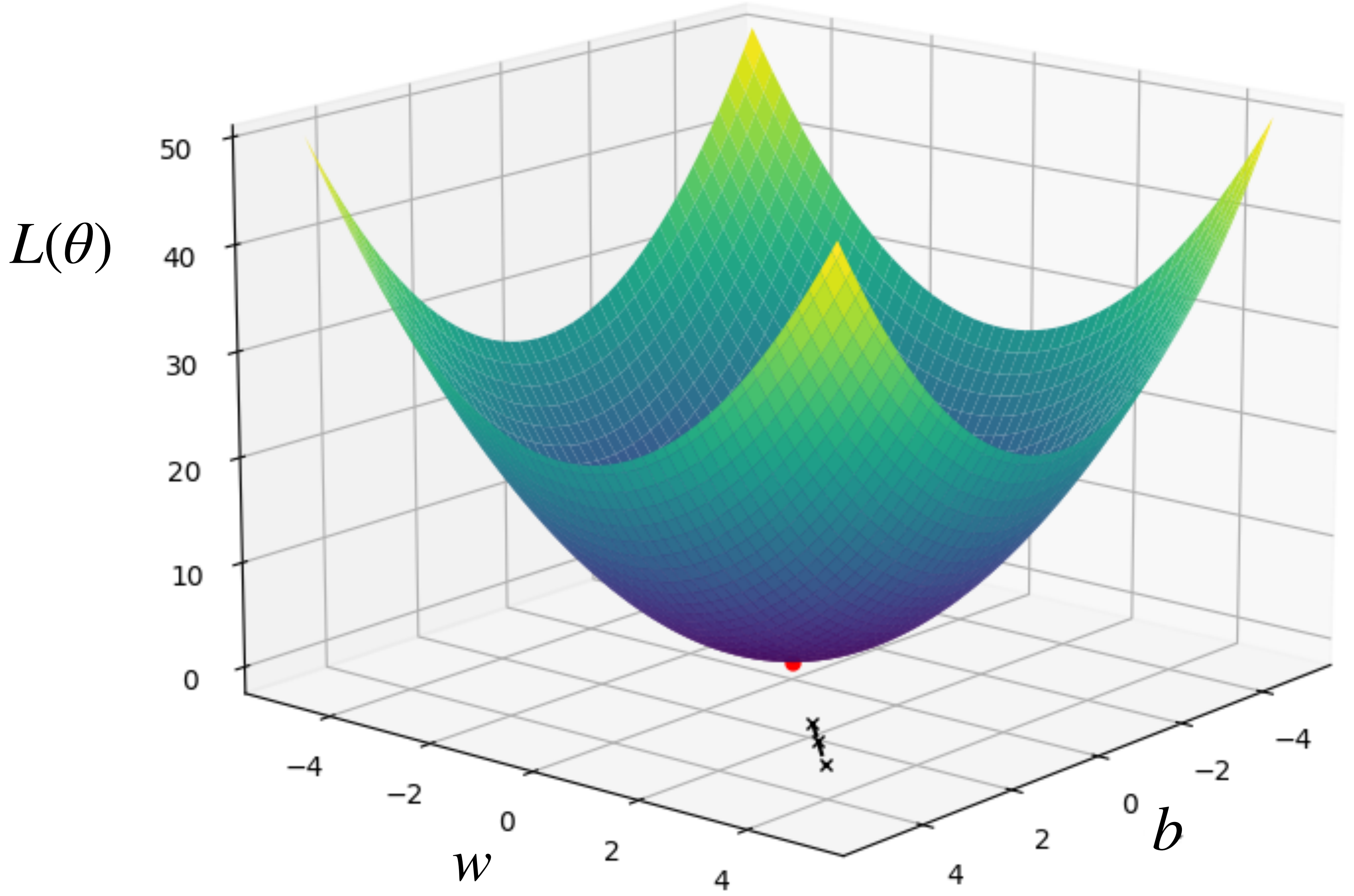
✕ $\theta_1 = (w_1, b_1)$

↖ $-\alpha \nabla L(\theta_1)$

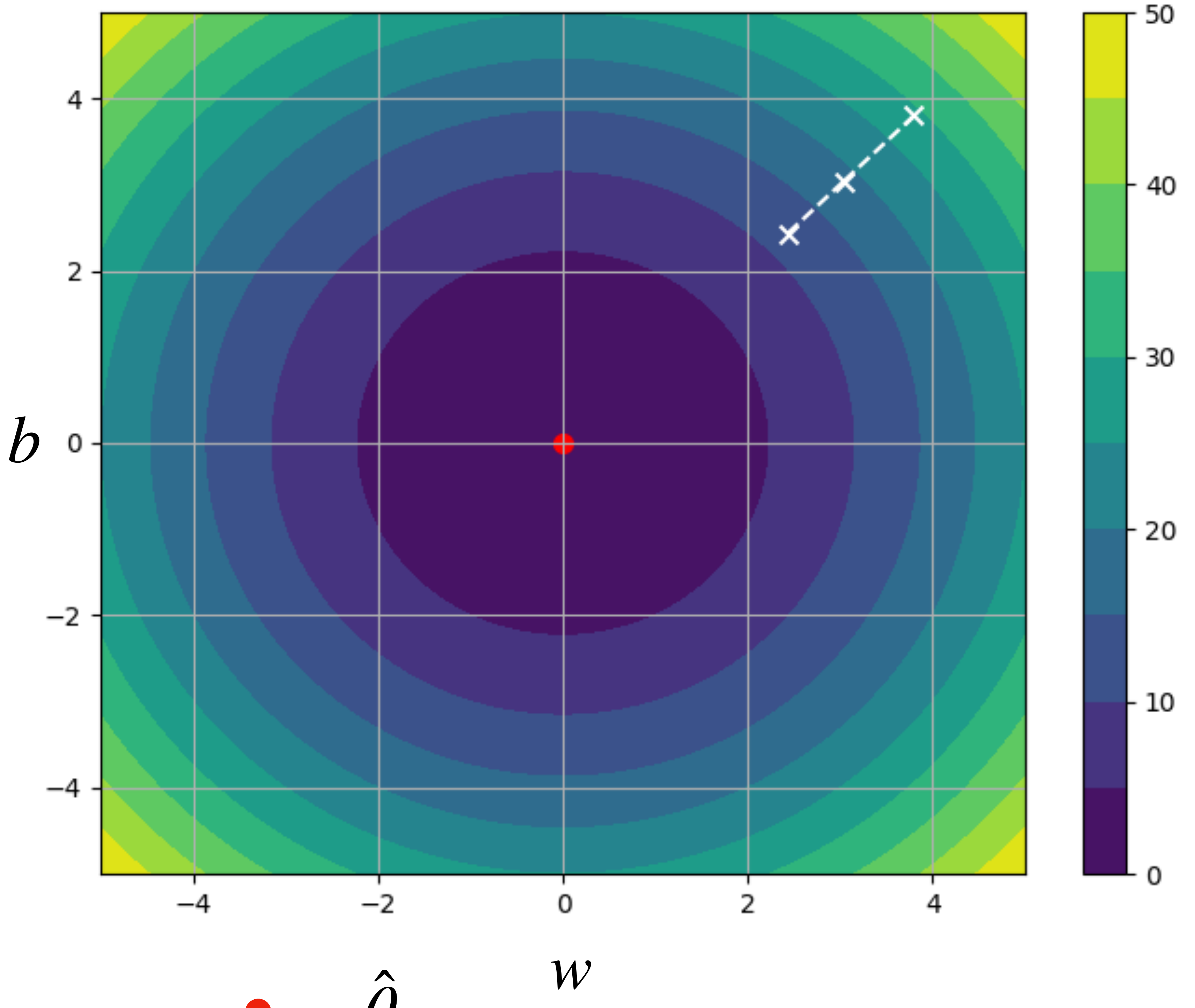
Loss Function Minimization

Solve the minimization problem $\hat{\theta} = argmin_{\theta} L(\theta)$ through gradient descent.

$\theta = \{w, b\}$



$\theta_2 = \theta_1 - \alpha \nabla L(\theta_1)$

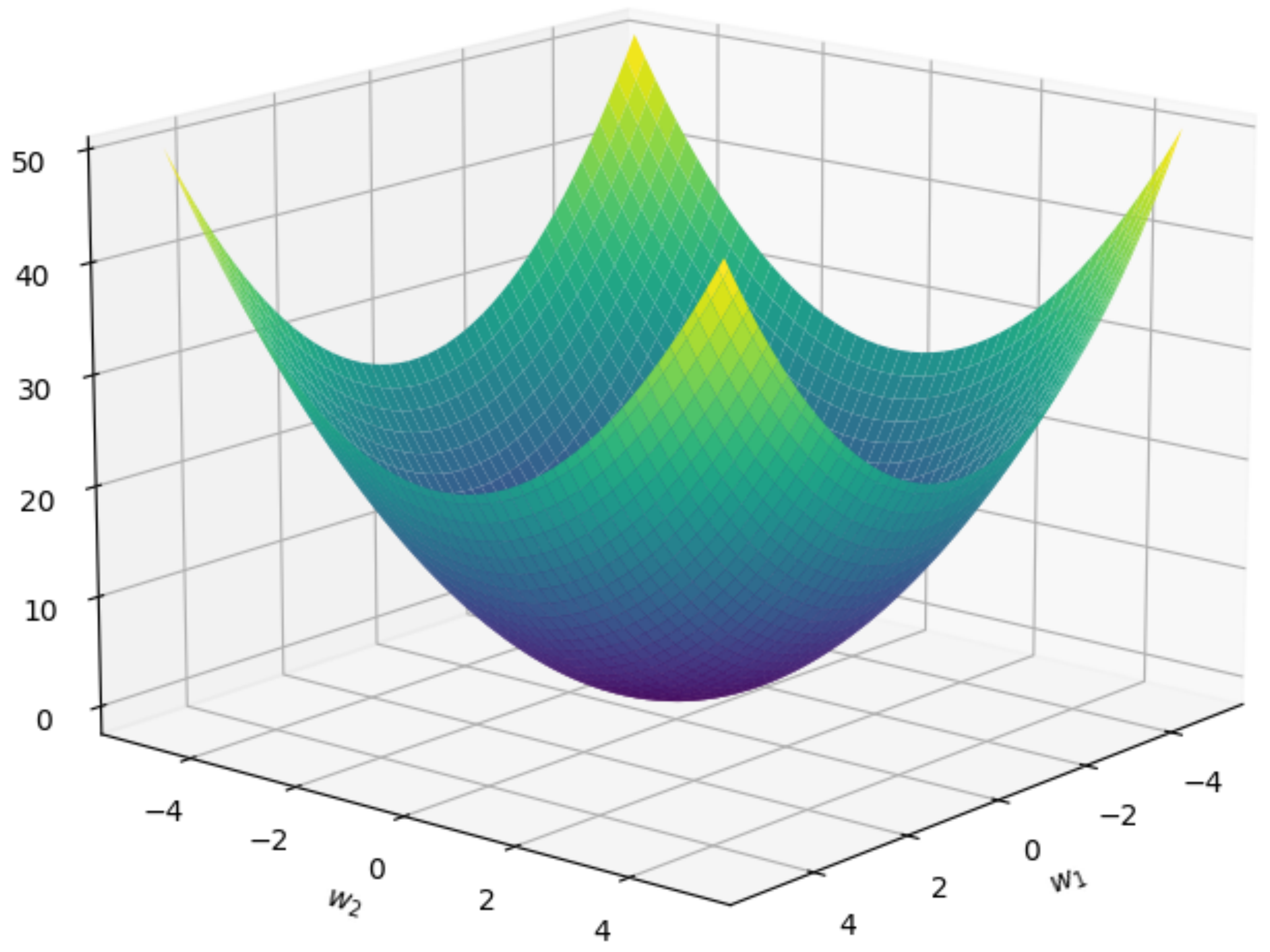


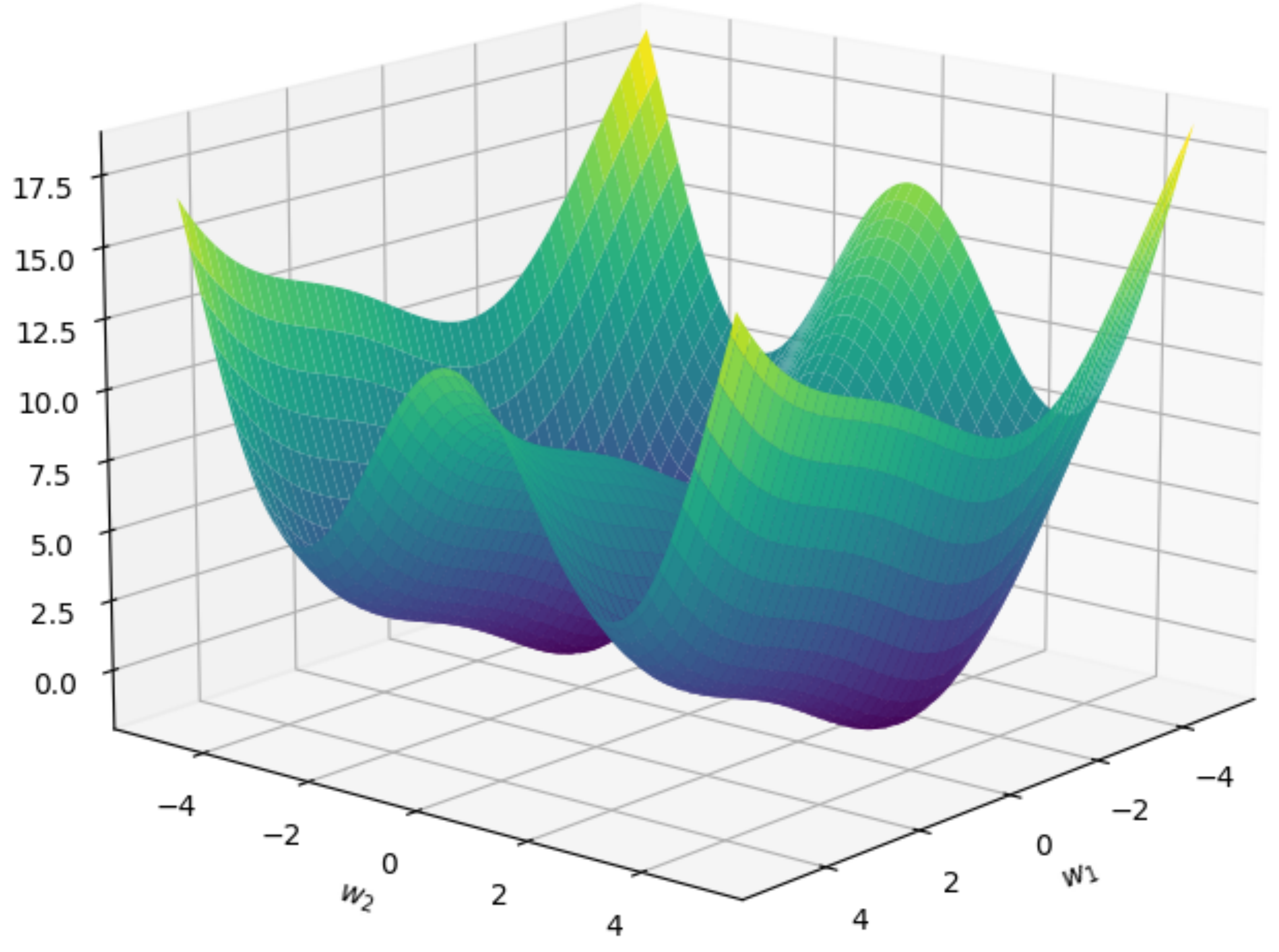
● $\hat{\theta}$

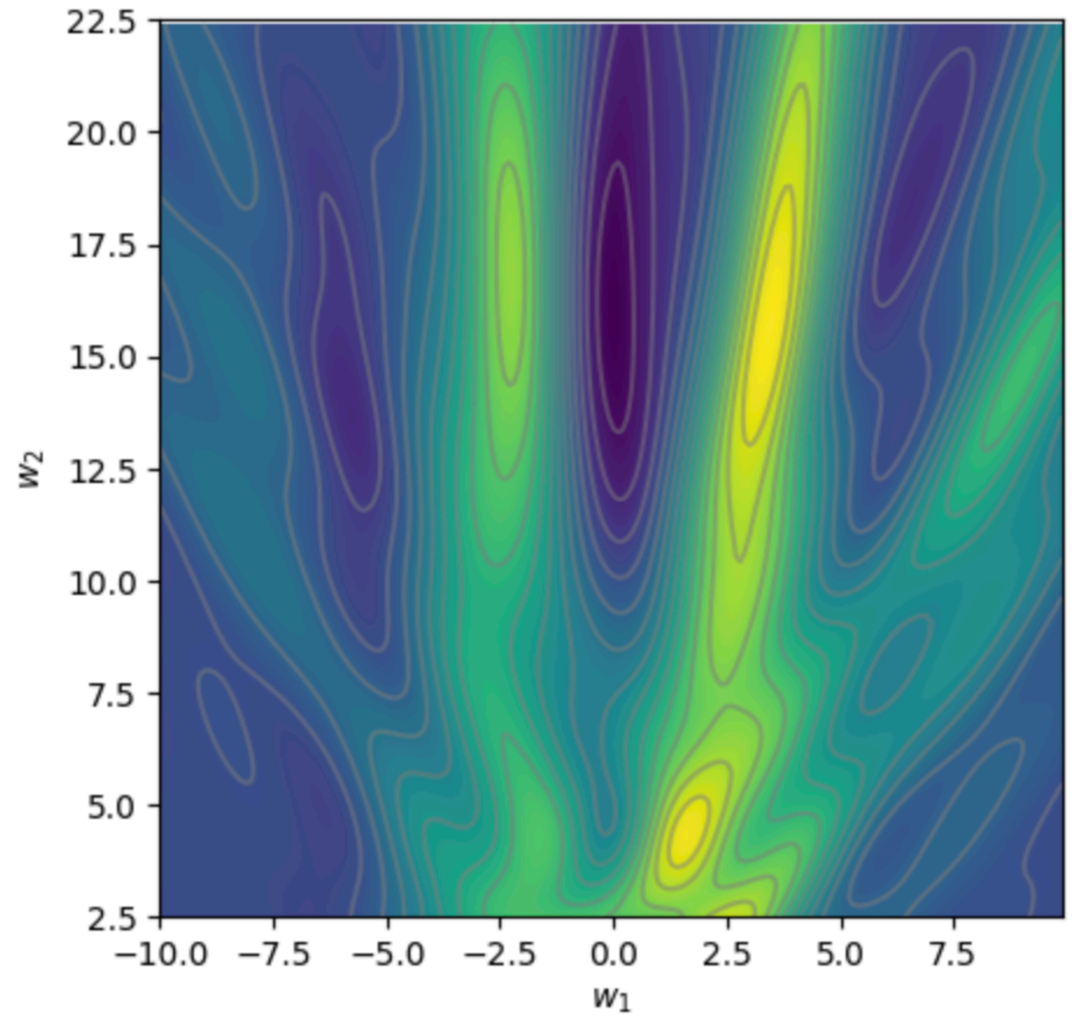
✕ $\theta_2 = (w_2, b_2)$

$-\alpha \nabla L(\theta_1)$

Loss Function Minima







Convex function ⟶ global minimum

Realistic loss functions are not-convex ⟶ search a local minimum

Gradient Descent

Iterative optimization algorithm to find $\hat{\theta} = argmin_{\theta} L(\theta)$

Step 1. Compute the derivatives $\nabla L(\theta) = \left(\frac{\partial L}{\partial w_1}(\theta), \frac{\partial L}{\partial w_2}(\theta) \ldots \right)$

Step 2. Update the parameters $\theta \leftarrow \theta - \alpha \nabla L(\theta)$ $\longleftrightarrow$ $w_j \leftarrow w_j - \alpha \frac{\partial L}{\partial w_j}(\theta)$

Terminate if $\|\nabla L(\theta)\|$ is small enough

The *learning rate* α controls how fast we are changing the weights.

Deterministic Gradient Descent

Iterative optimization algorithm to find $\hat{\theta} = argmin_{\theta} L(\theta)$

Step 1. Compute the derivatives $\nabla L(\theta) = \left(\frac{\partial L}{\partial w_1}(\theta), \frac{\partial L}{\partial w_2}(\theta) \dots \right)$

(Deterministic/full-batch)

$$\frac{\partial L}{\partial w_j}(\theta) = \frac{\partial}{\partial w_j}\left(\frac{1}{n} \sum_{i=1}^{n} l^{(i)} \right)(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial l^{(i)}}{\partial w_j}(\theta)$$

Deterministic Gradient Descent

Iterative optimization algorithm to find $\hat{\theta} = argmin_{\theta} L(\theta)$

Step 1. Compute the derivatives $\nabla L(\theta) = \left(\frac{\partial L}{\partial w_1}(\theta), \frac{\partial L}{\partial w_2}(\theta) \ldots \right)$

(Deterministic/full-batch)

$$\frac{\partial L}{\partial w_j}(\theta) = \frac{\partial}{\partial w_j}\left(\frac{1}{n} \sum_{i=1}^{n} l^{(i)} \right)(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial l^{(i)}}{\partial w_j}(\theta)$$

Problems:

- One update might be time consuming and require a lot of memory for large datasets
- The local minimum found highly depends on the initial θ_0

Stochastic Gradient Descent

Iterative optimization algorithm to find $\hat{\theta} = argmin_{\theta} L(\theta)$

Step 1. Compute the derivatives $\nabla L(\theta) = \left(\frac{\partial L}{\partial w_1}(\theta), \frac{\partial L}{\partial w_2}(\theta) \dots \right)$

(Stochastic) Gradient Descent

$$\frac{\partial L}{\partial w_j}(\theta) \approx \frac{1}{m} \sum_{k=1}^{m} \frac{\partial l^{(i_k)}}{\partial w_j}(\theta)$$

Randomly sample mini-batches with m data points $\{x^{(i_k)}, y^{(i_k)}\}_{k=1,\dots,m}$

Problems:

- One update might be time consuming and require a lot of memory for large datasets
- The local minimum found highly depends on the initial θ_0

Stochastic Gradient Descent

Iterative optimization algorithm to find $\hat{\theta} = argmin_{\theta} L(\theta)$

Step 1. Compute the derivatives $\nabla L(\theta) = \left(\frac{\partial L}{\partial w_1}(\theta), \frac{\partial L}{\partial w_2}(\theta) \ldots \right)$

(Stochastic) Gradient Descent

$$\frac{\partial L}{\partial w_j}(\theta) \approx \frac{1}{m} \sum_{k=1}^{m} \frac{\partial l^{(i_k)}}{\partial w_j}(\theta)$$

Randomly sample mini-batches with m data points $\{x^{(i_k)}, y^{(i_k)}\}_{k=1,\ldots,m}$

How to compute the derivatives?

Computing Derivatives

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

$$x \xrightarrow{\cdot w + b} h \xrightarrow{\sigma} \lambda$$

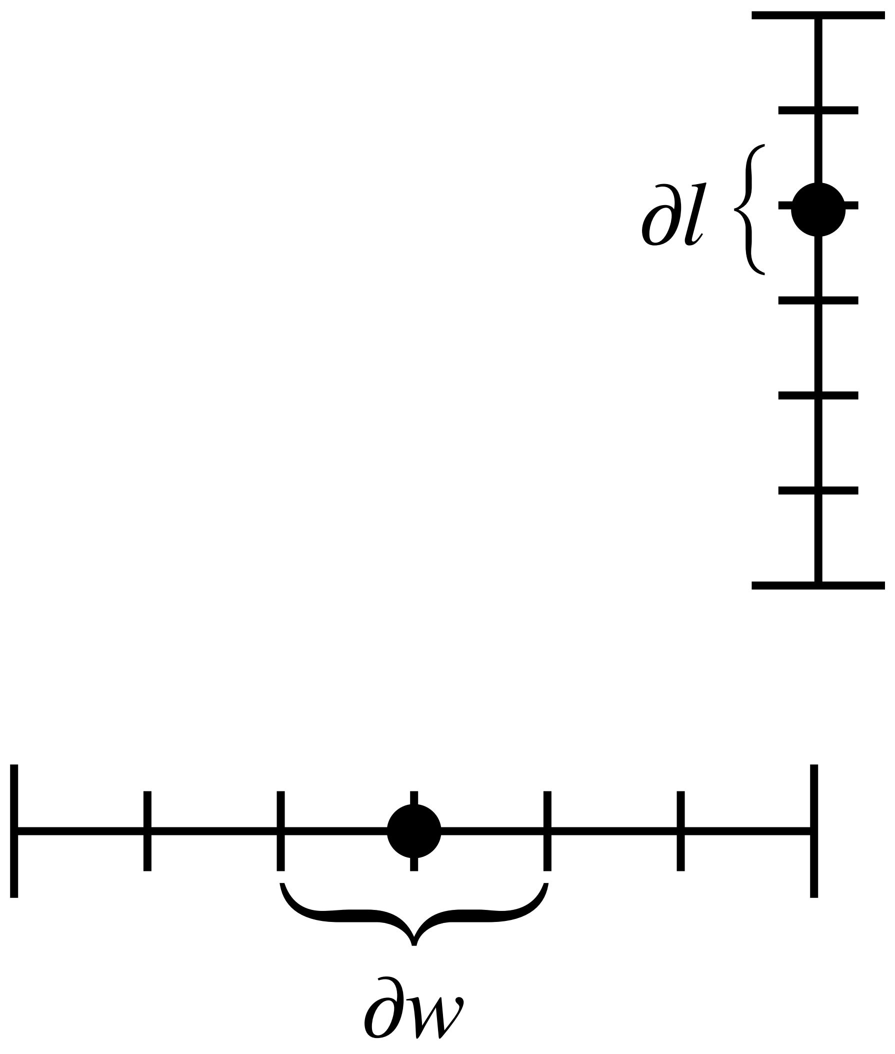
Computing Derivatives

Example: univariate regression

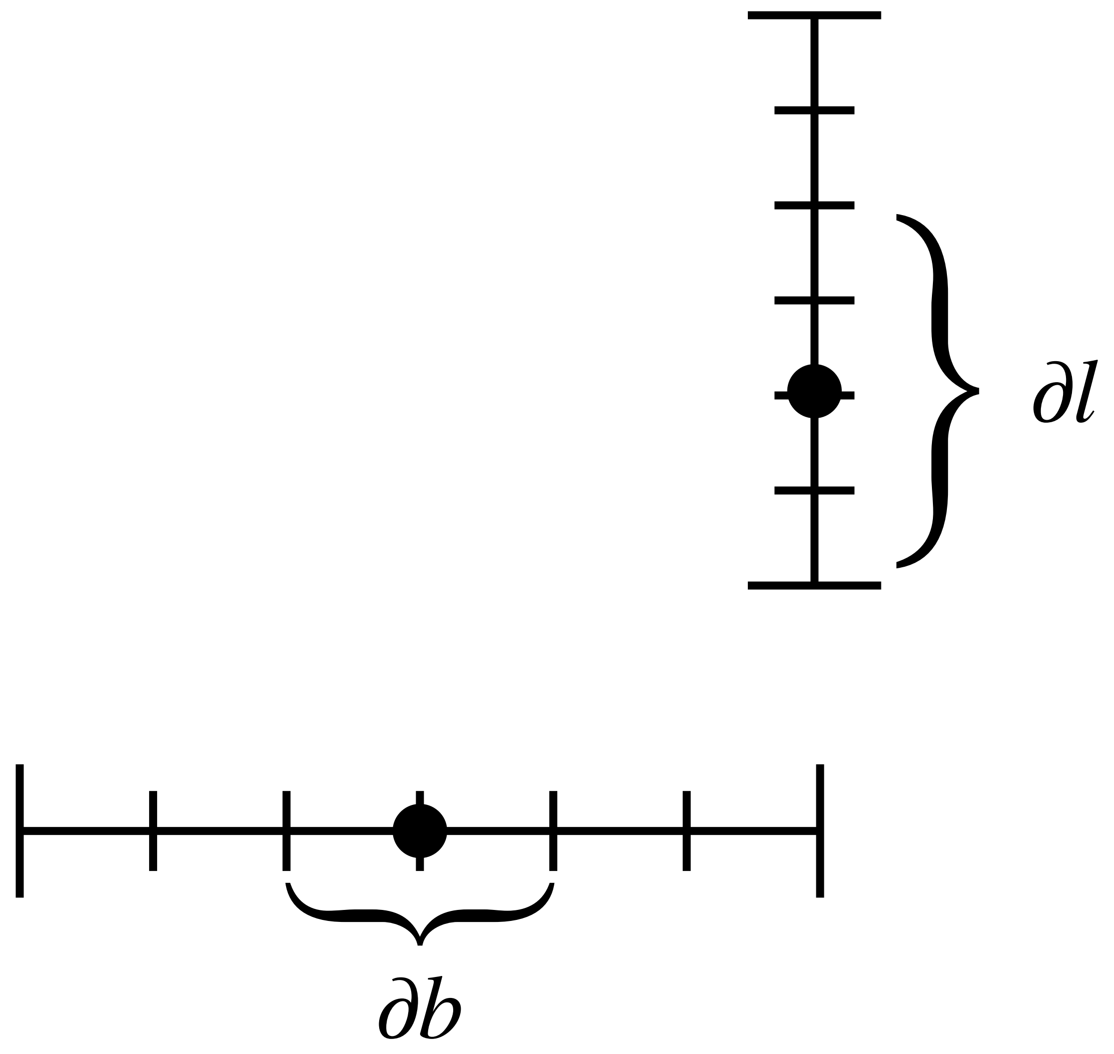
$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

$$x \xrightarrow{\cdot\, w + b} h \xrightarrow{\sigma} \lambda$$

$$\frac{\partial l}{\partial w} =$$



$$\frac{\partial l}{\partial b} =$$



How sensitive is the loss to small changes of a specific parameter

Computing Derivatives

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

$$x \xrightarrow{\cdot w + b} h \xrightarrow{\sigma} \lambda$$

$$\begin{aligned}
\frac{\partial l}{\partial w} &= \frac{\partial}{\partial w}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right) \\
&= (y - \sigma(xw + b))\frac{\partial}{\partial w}\left(y - \sigma(xw + b)\right) \\
&= -(y - \sigma(xw + b))\sigma'(xw + b)x
\end{aligned}$$

Why?

$$\begin{aligned}
\frac{\partial l}{\partial b} &= \frac{\partial}{\partial b}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right) \\
&= (y - \sigma(xw + b))\frac{\partial}{\partial b}\left(y - \sigma(xw + b)\right) \\
&= -(y - \sigma(xw + b))\sigma'(xw + b)
\end{aligned}$$

Computing Derivatives

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

$$x \xrightarrow{\cdot w + b} h \xrightarrow{\sigma} \lambda$$

$$\frac{\partial l}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right)$$
$$= (y - \sigma(xw + b))\frac{\partial}{\partial w}\left(y - \sigma(xw + b)\right)$$
$$= -(y - \sigma(xw + b))\sigma'(xw + b)x$$

Chain Rule*

$$\frac{\partial l}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right)$$
$$= (y - \sigma(xw + b))\frac{\partial}{\partial b}\left(y - \sigma(xw + b)\right)$$
$$= -(y - \sigma(xw + b))\sigma'(xw + b)$$

**To refresh the chain rule check* *MIT-Multivariable Calculus: ChainRule, Gradient and Directional Derivatives*

Computing Derivatives

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

$$x \xrightarrow{\cdot\, w + b} h \xrightarrow{\sigma} \lambda$$

$$\begin{aligned}\frac{\partial l}{\partial w} &= \frac{\partial}{\partial w}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right)\\ &= \boxed{(y - \sigma(xw + b))\frac{\partial}{\partial w}\big(y - \sigma(xw + b)\big)}\\ &= -(y - \sigma(xw + b))\sigma'(xw + b)x\end{aligned}$$

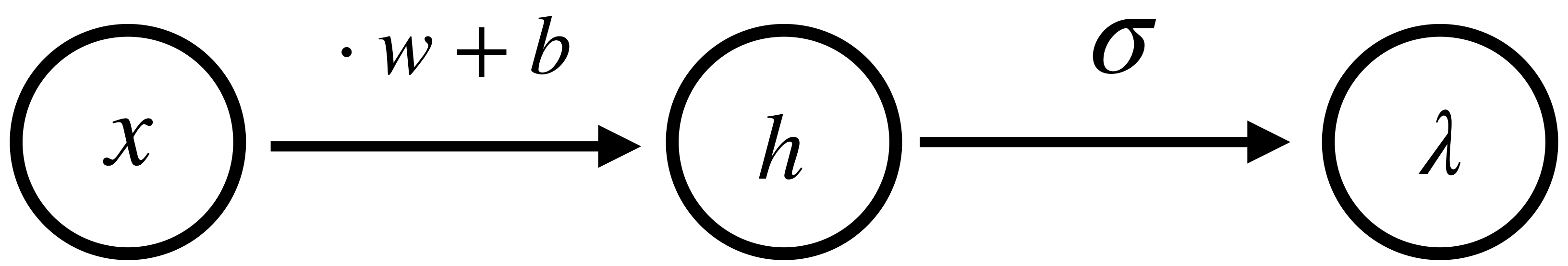
$$\begin{aligned}\frac{\partial l}{\partial b} &= \frac{\partial}{\partial b}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right)\\ &= \boxed{(y - \sigma(xw + b))\frac{\partial}{\partial b}\big(y - \sigma(xw + b)\big)}\\ &= -(y - \sigma(xw + b))\sigma'(xw + b)\end{aligned}$$

- Repeated computations

Computing Derivatives

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



$$\begin{aligned}\frac{\partial l}{\partial w} &= \frac{\partial}{\partial w}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right) \\ &= (y - \sigma(xw + b))\frac{\partial}{\partial w}\left(y - \sigma(xw + b)\right) \\ &= \boxed{-(y - \sigma(xw + b))\sigma'(xw + b)}x\end{aligned}$$

$$\begin{aligned}\frac{\partial l}{\partial b} &= \frac{\partial}{\partial b}\left(\frac{1}{2}(y - \sigma(xw + b))^2\right) \\ &= (y - \sigma(xw + b))\frac{\partial}{\partial b}\left(y - \sigma(xw + b)\right) \\ &= \boxed{-(y - \sigma(xw + b))\sigma'(xw + b)}\end{aligned}$$

- Repeated computations
- Identical terms

Backpropagation

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

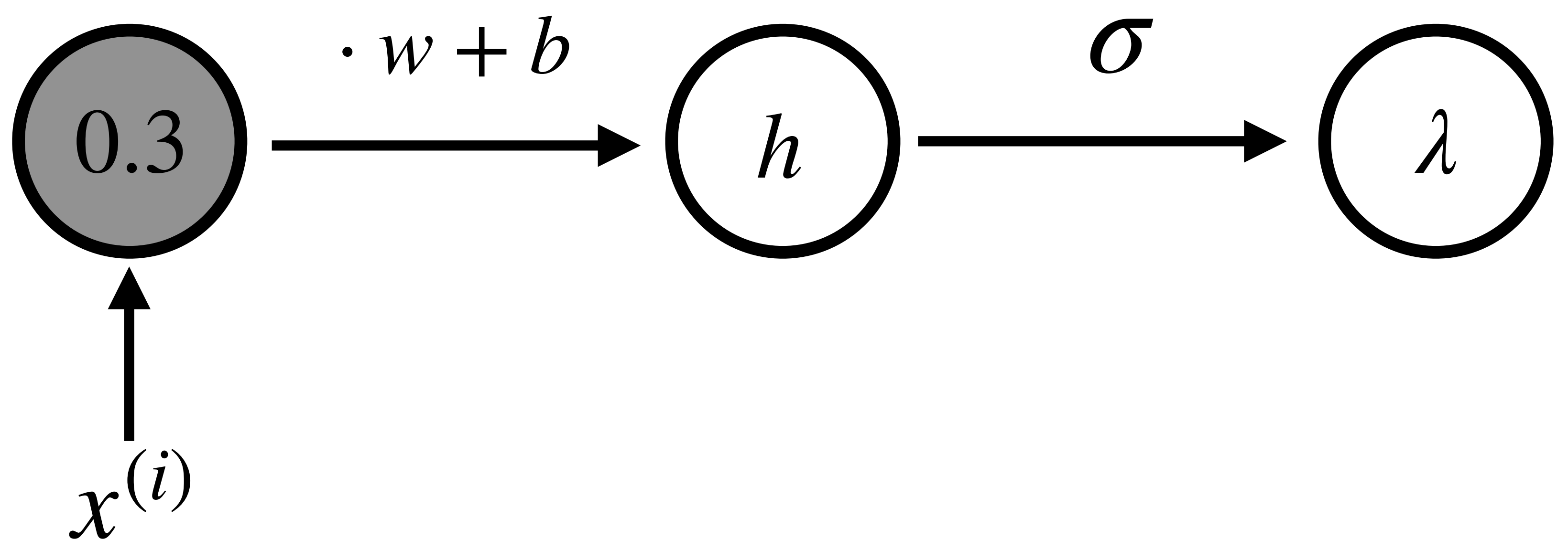
$$x \xrightarrow{\cdot w + b} h \xrightarrow{\sigma} \lambda$$

Efficient way to use chain rule to compute derivatives

Backpropagation

Example: univariate regression

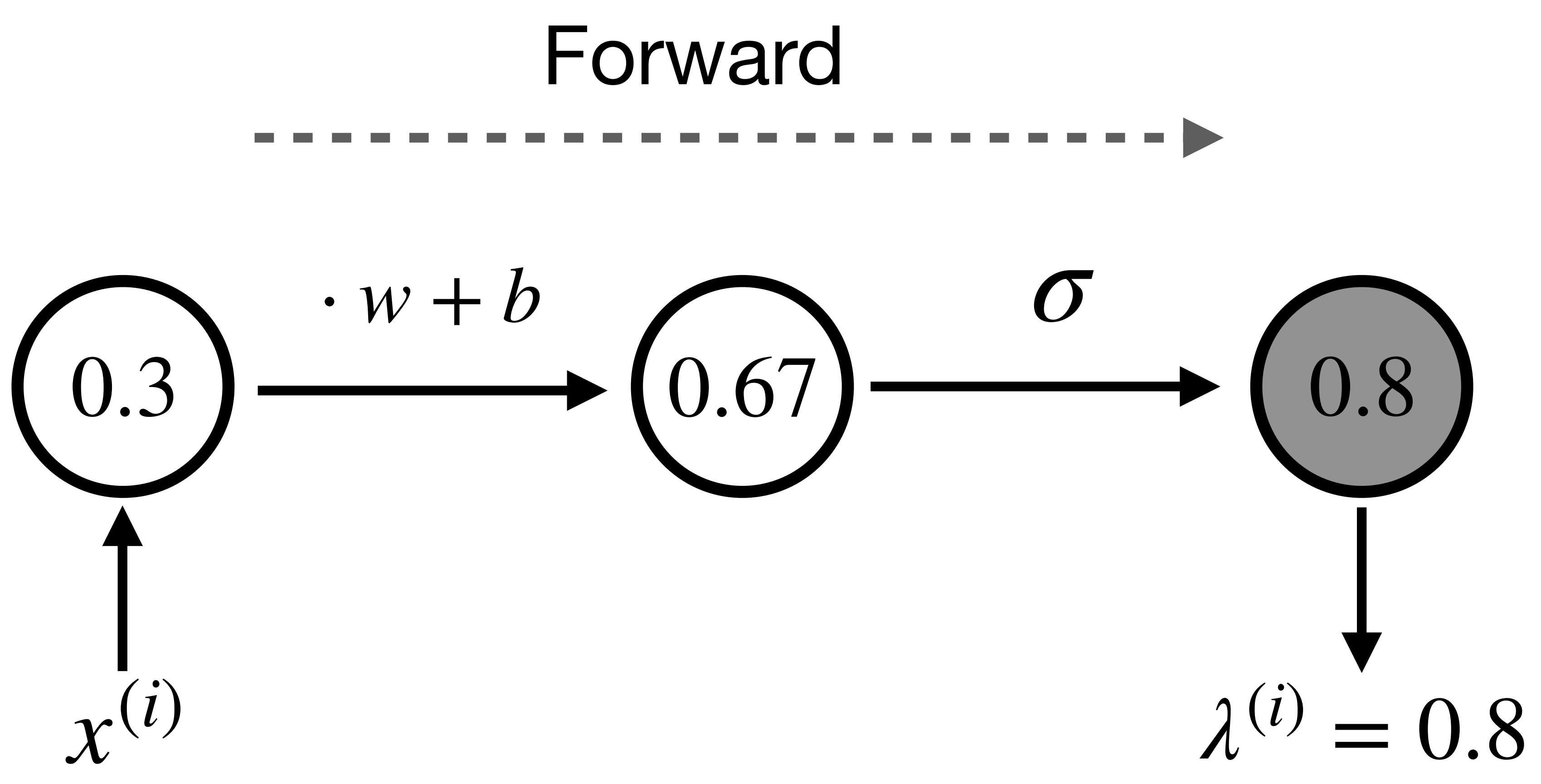
$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Backpropagation

Example: univariate regression

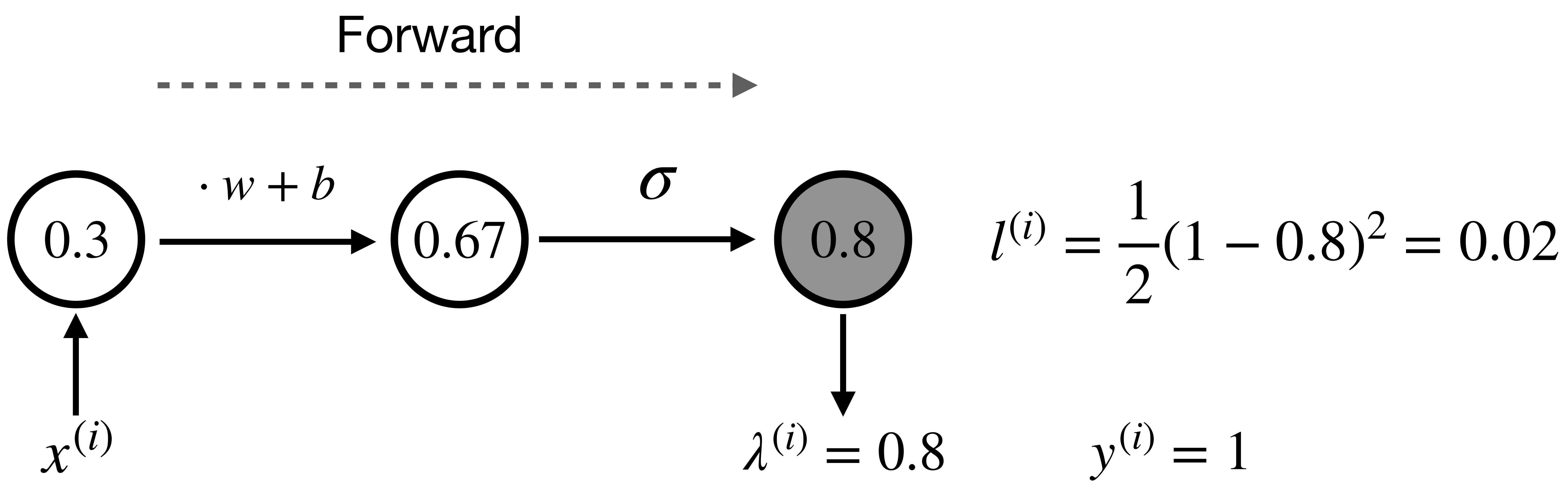
$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Backpropagation

Example: univariate regression

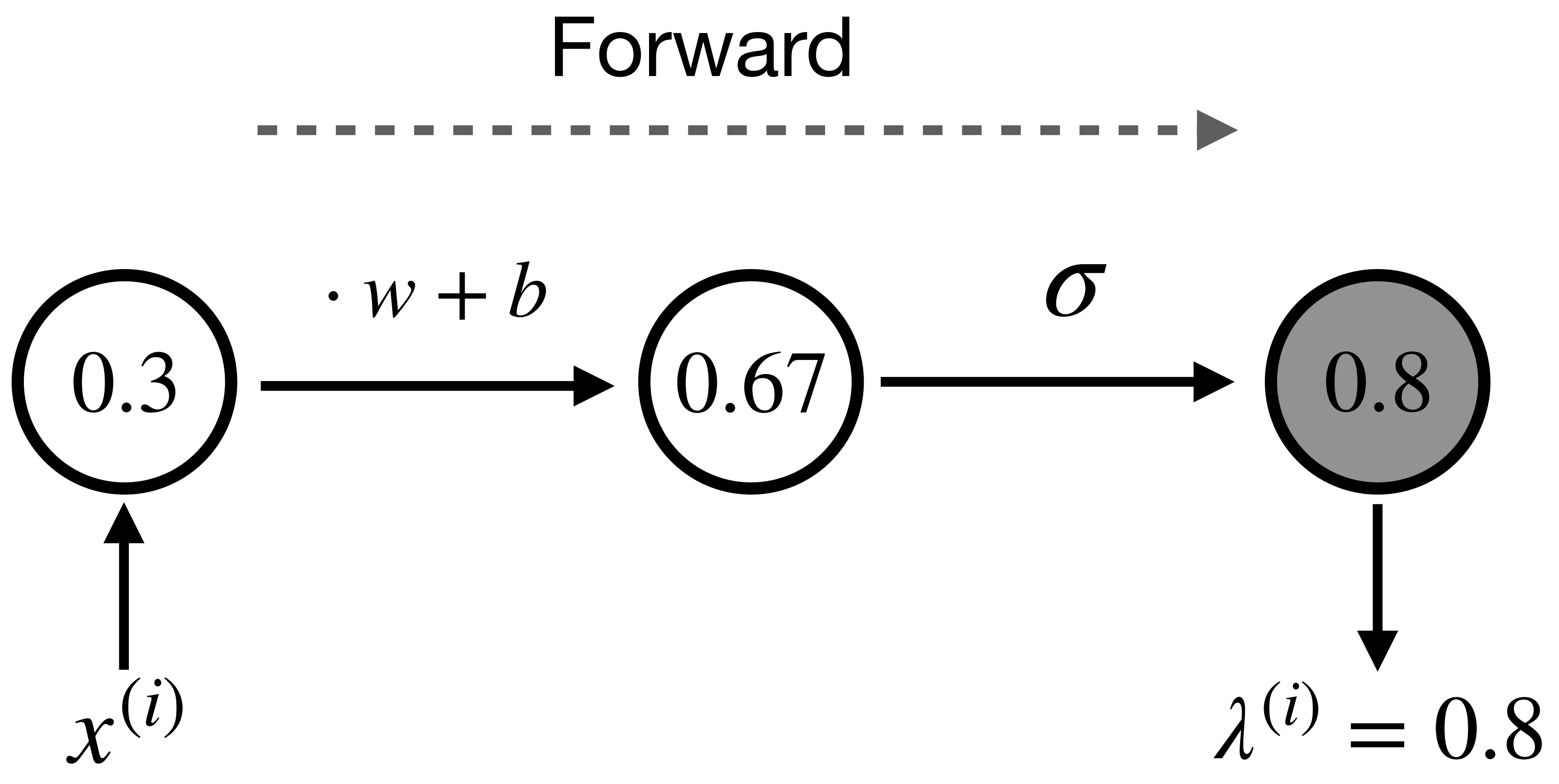
$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Backpropagation

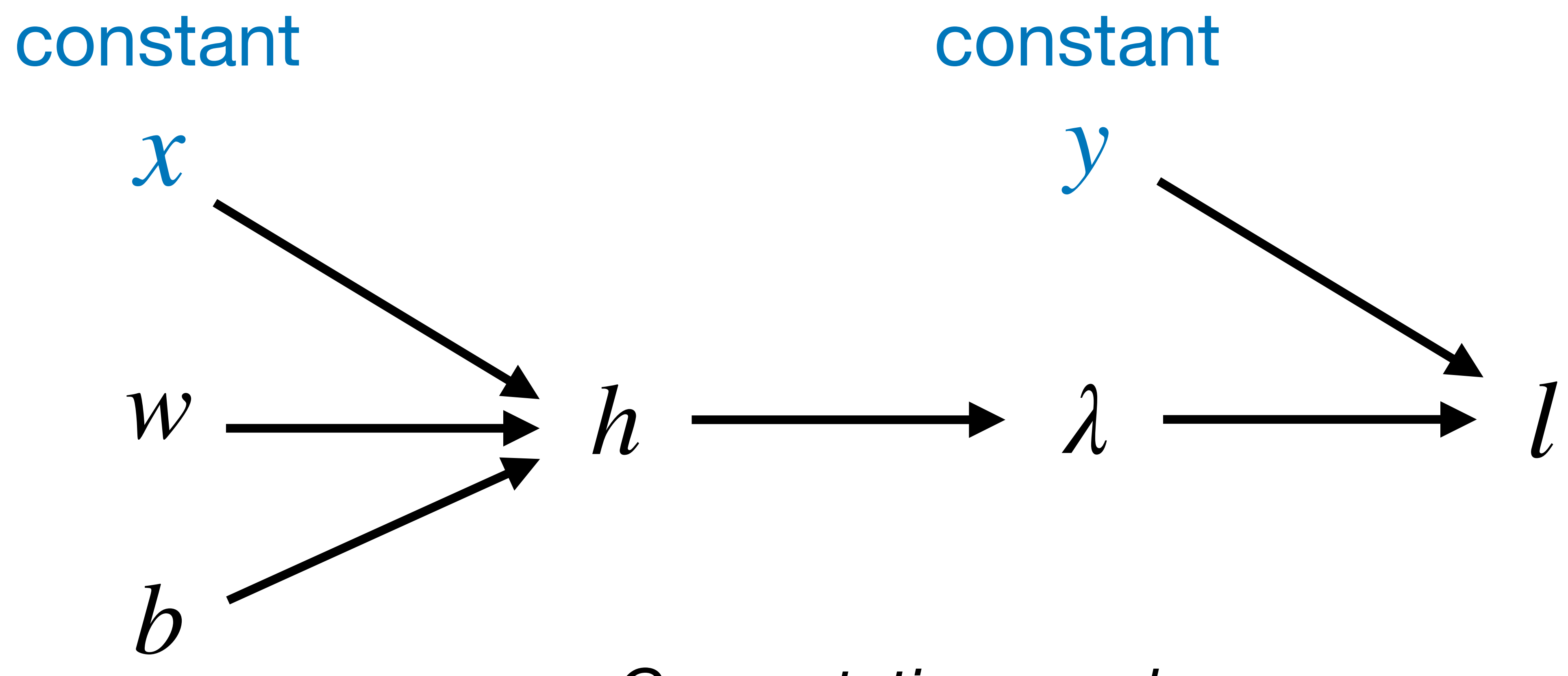
Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



$$l^{(i)} = \frac{1}{2}(1 - 0.8)^2 = 0.02$$

$$y^{(i)} = 1$$



Computation graph

Backpropagation

Example: univariate regression

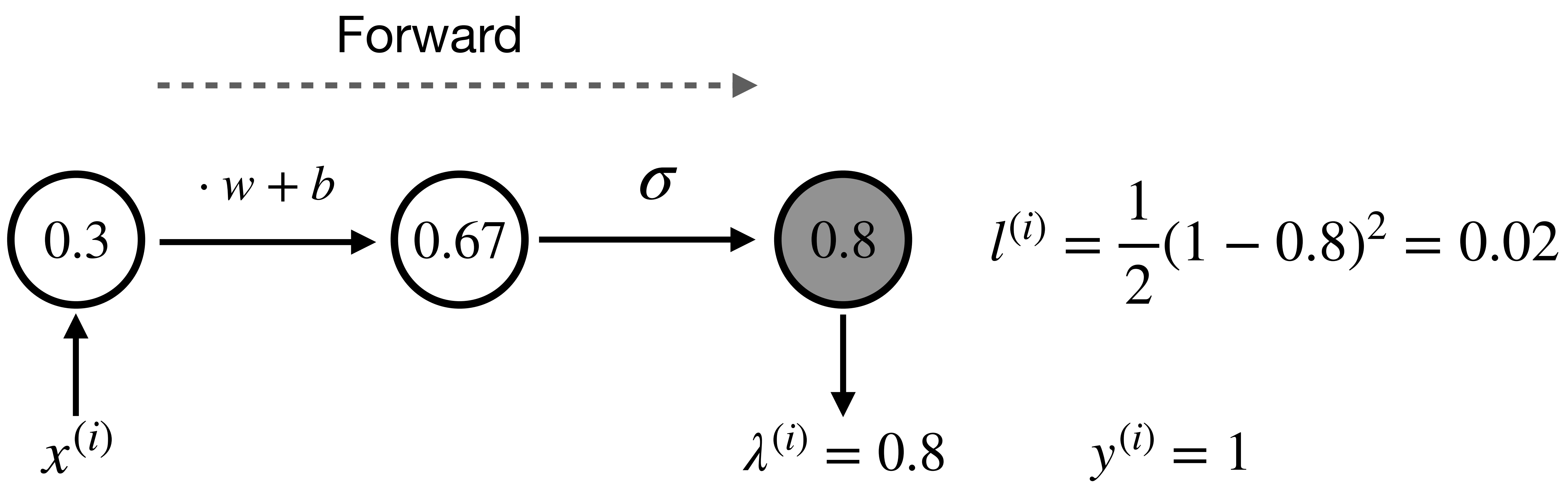
$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

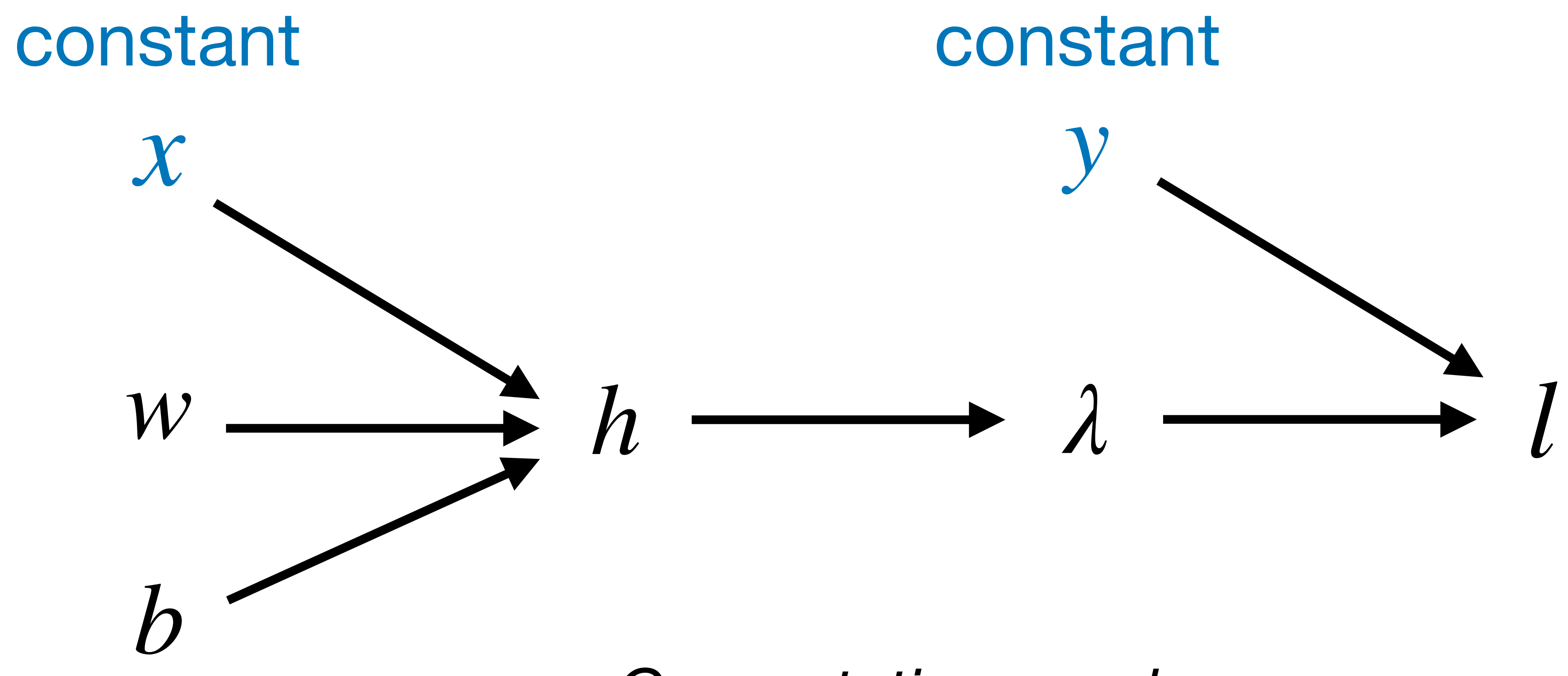
Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$





Computation graph

Chain Rule in Neural Network

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

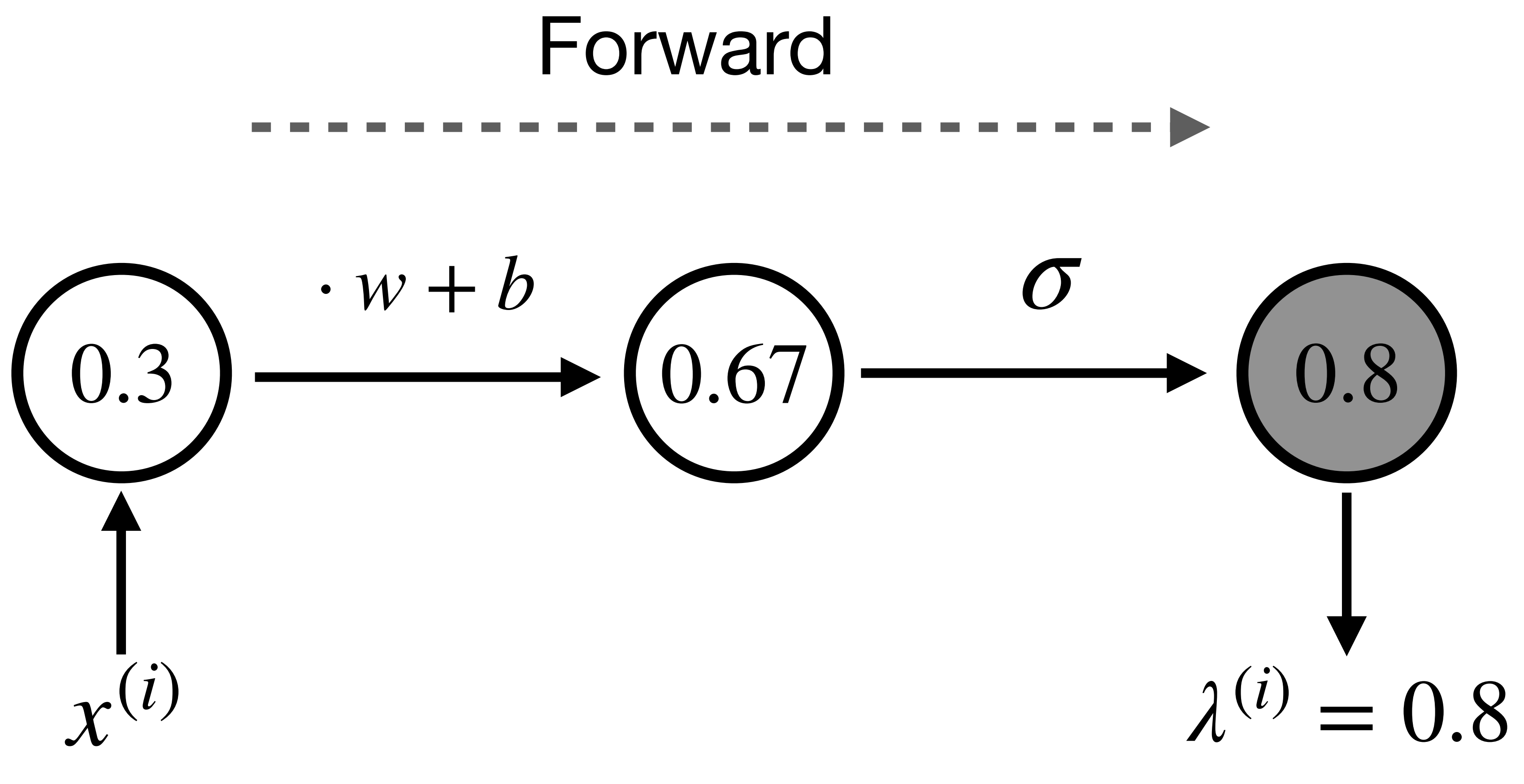
$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

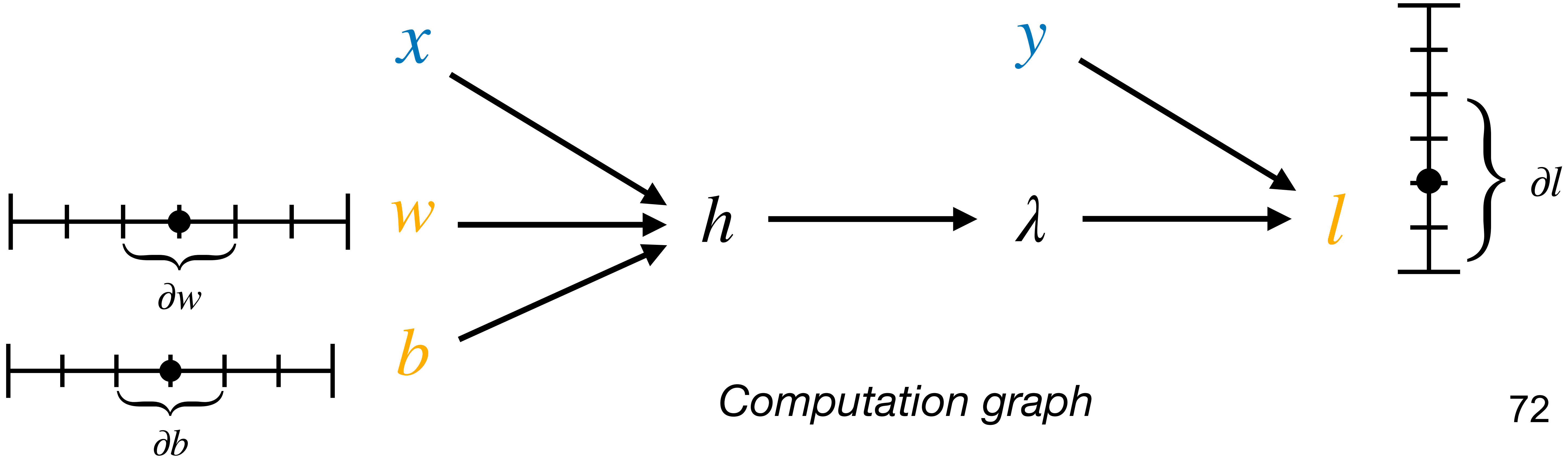
$$\frac{\partial l}{\partial w} = \cdots$$

$$\frac{\partial l}{\partial b} = \cdots$$



$$l^{(i)} = \frac{1}{2}(1 - 0.8)^2 = 0.02$$

$$y^{(i)} = 1$$



Computation graph

Chain Rule in Neural Network

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

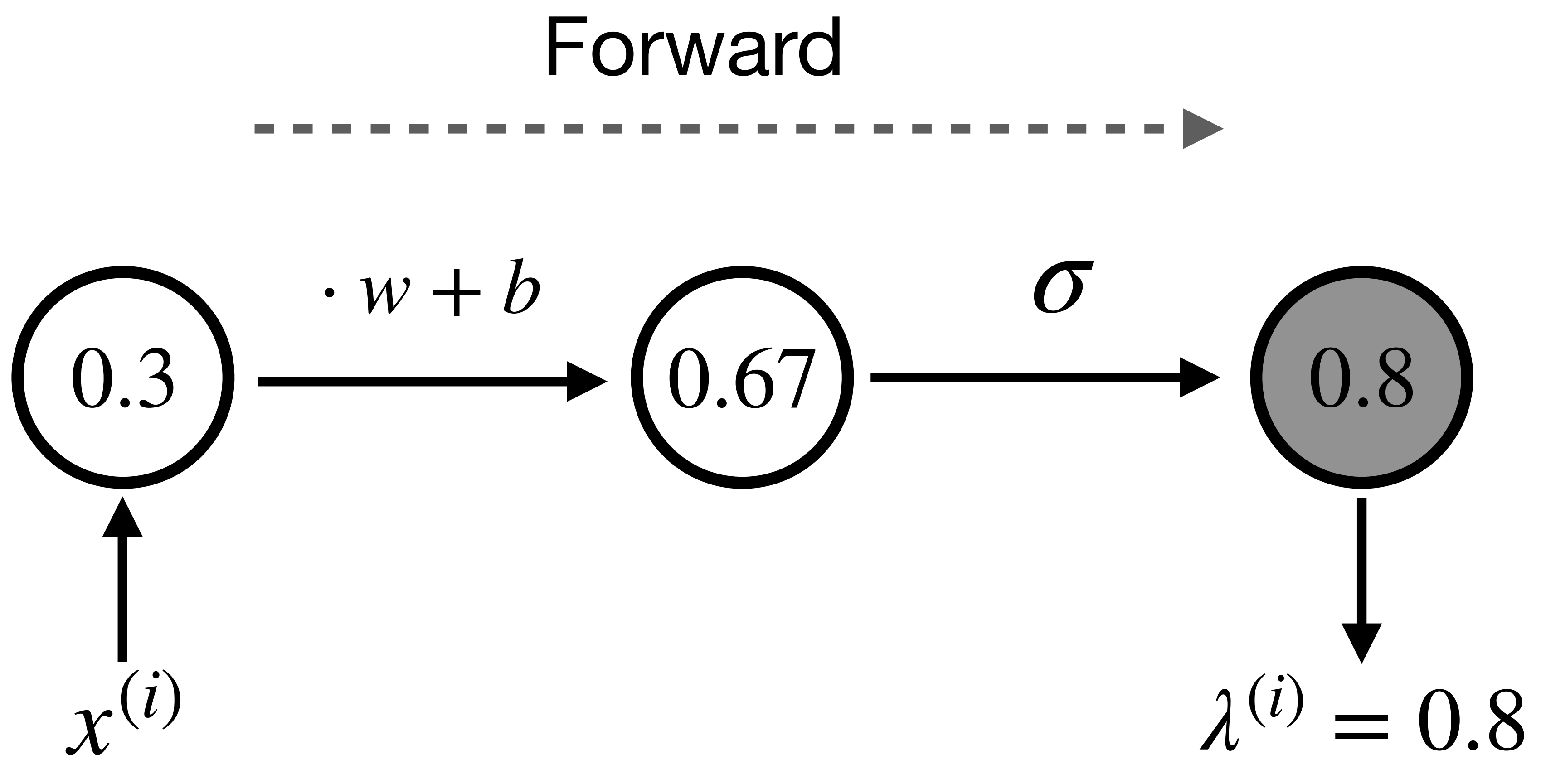
$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

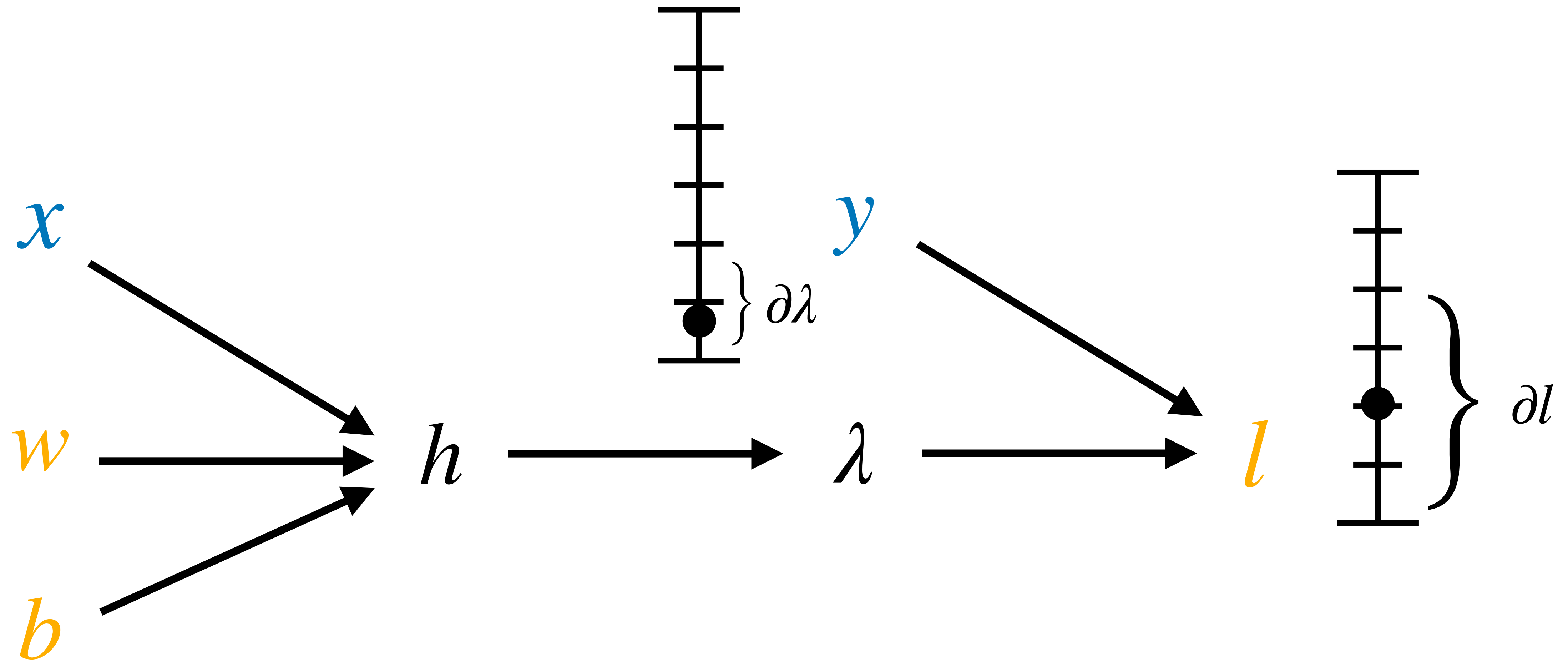
$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda} \cdots$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda} \cdots$$



$$l^{(i)} = \frac{1}{2}(1 - 0.8)^2 = 0.02$$

$$y^{(i)} = 1$$



Computation graph

Chain Rule in Neural Network

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

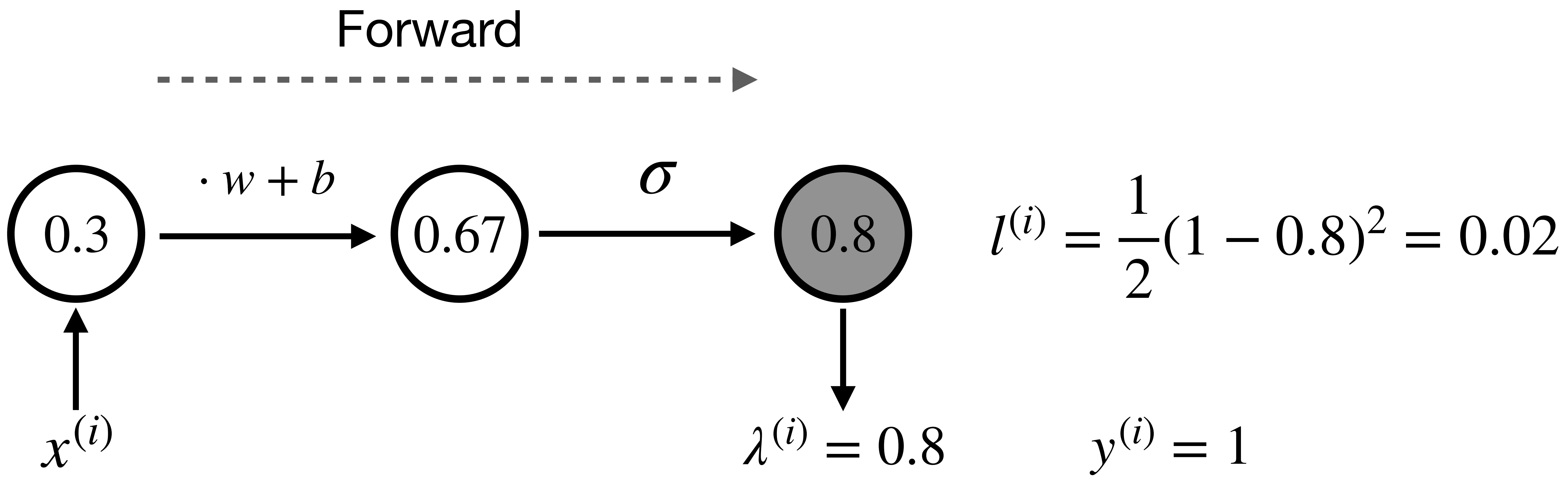
$$l = \frac{1}{2}(y - \lambda)^2$$

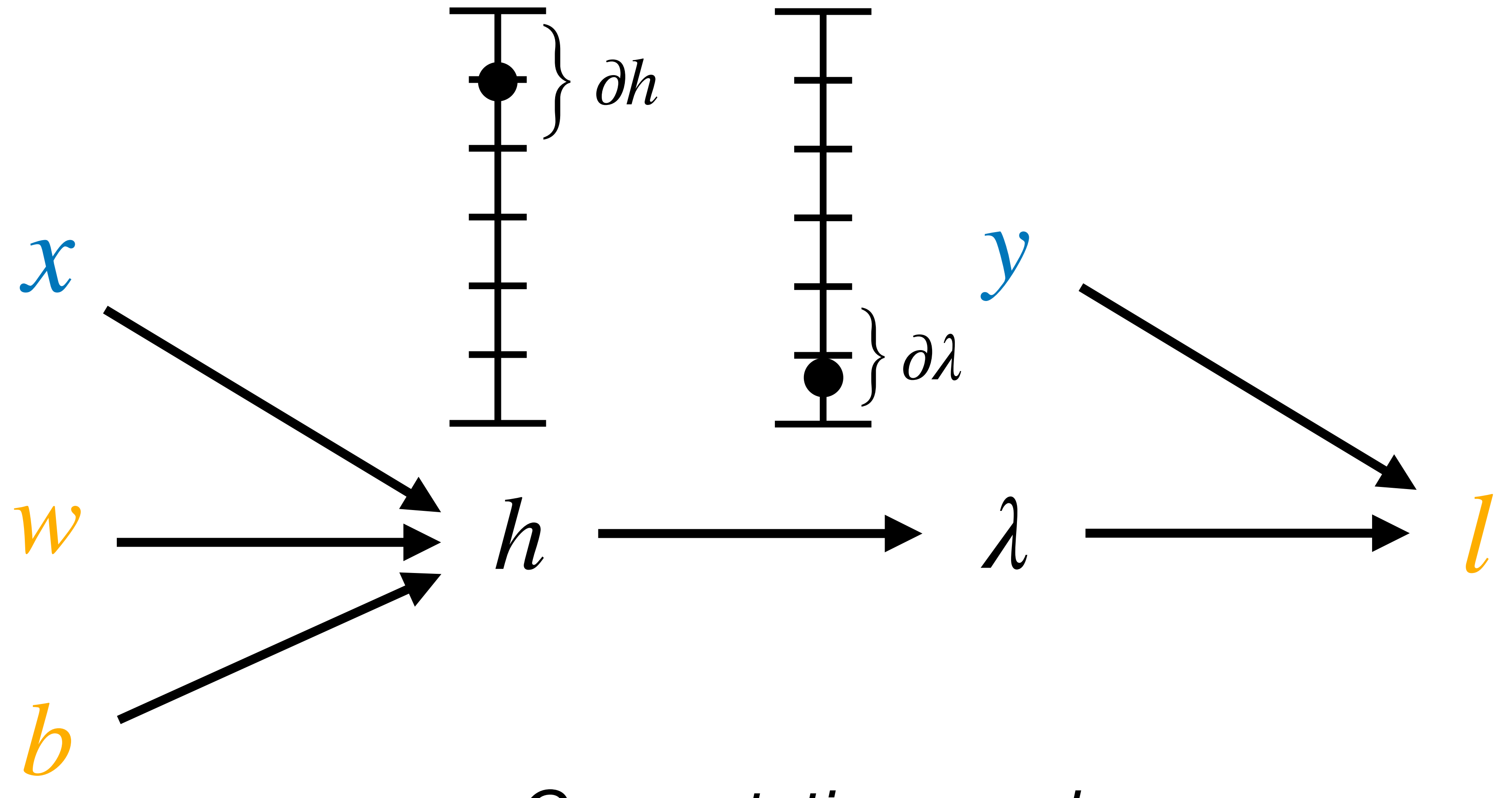
$$\lambda = \sigma(h)$$

$$h = xw + b$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\cdots$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\cdots$$





Computation graph

Chain Rule in Neural Network

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

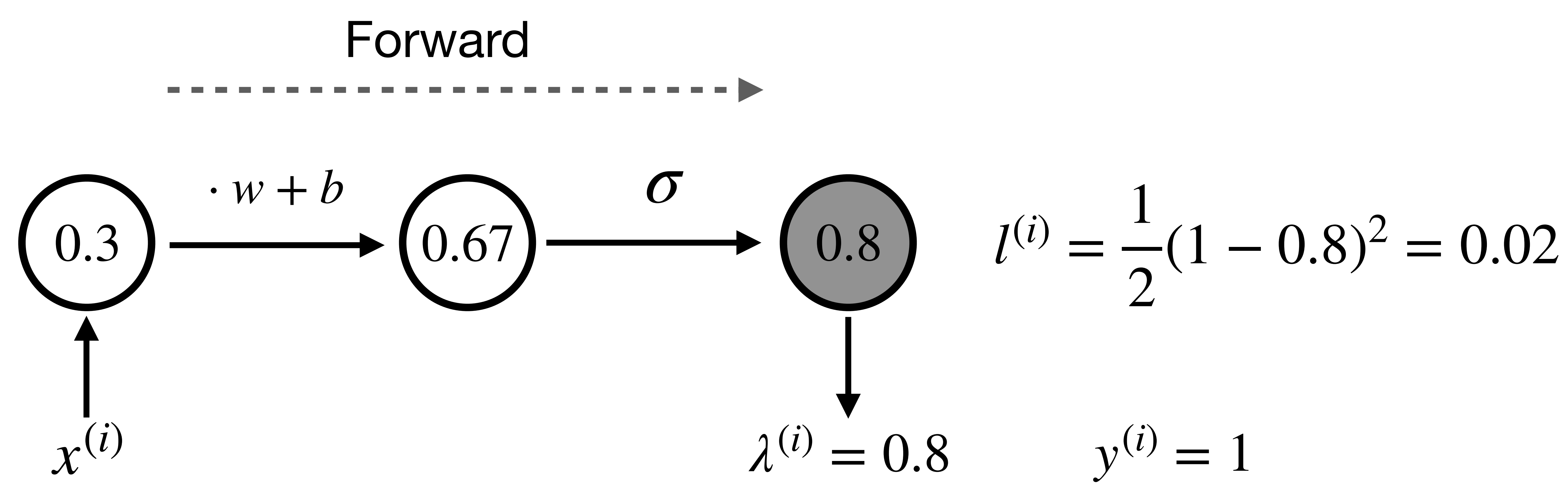
$$l = \frac{1}{2}(y - \lambda)^2$$

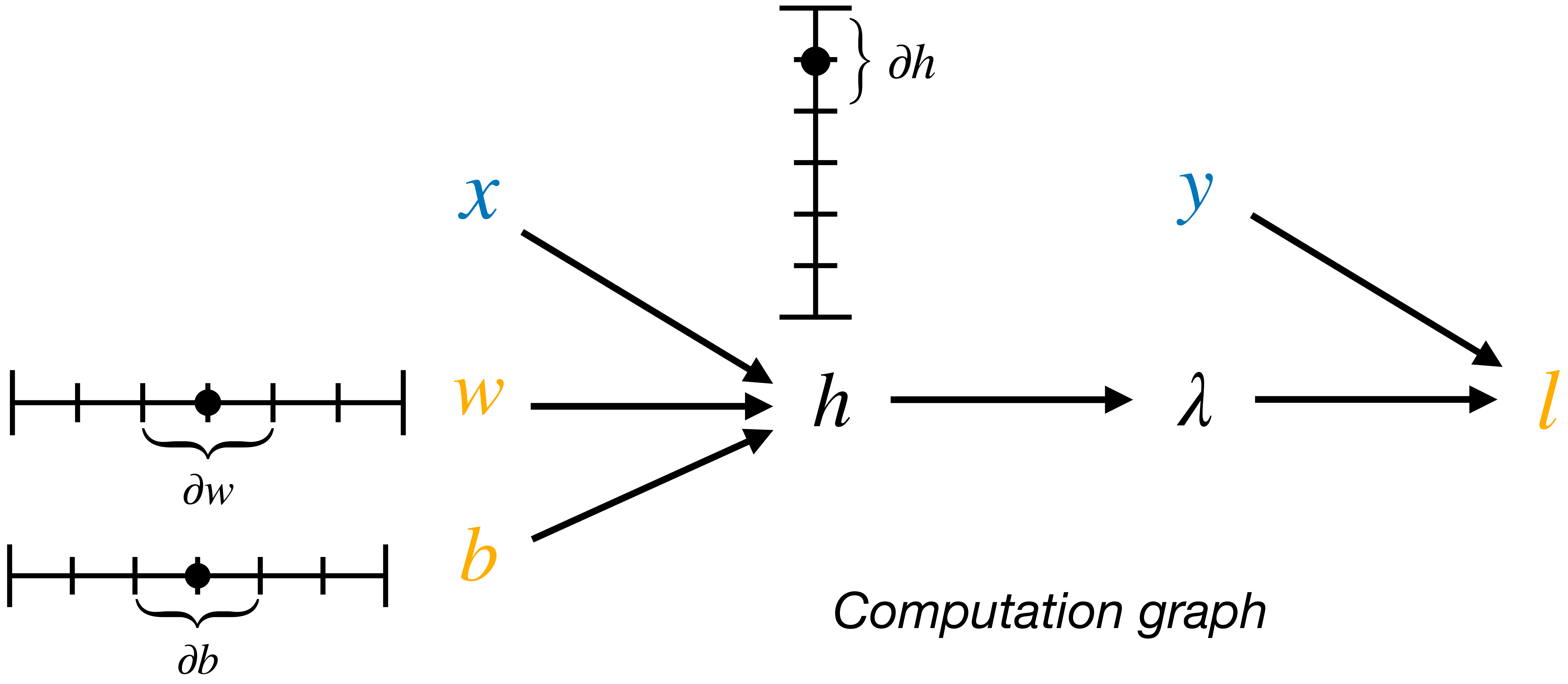
$$\lambda = \sigma(h)$$

$$h = xw + b$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$





Computation graph

Chain Rule in Neural Network

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

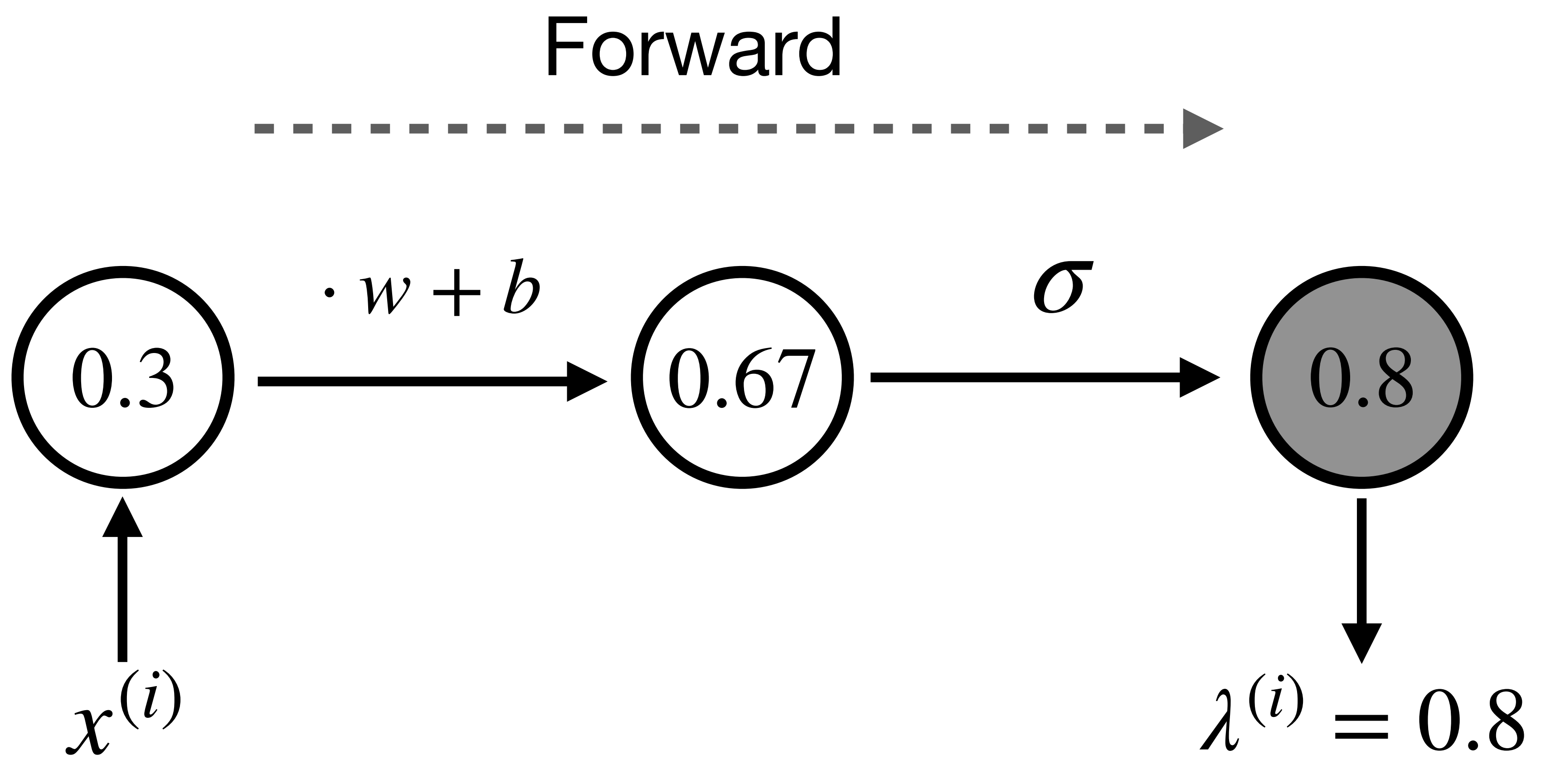
$$\frac{\partial l}{\partial w} = \boxed{\frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$

3 derivatives

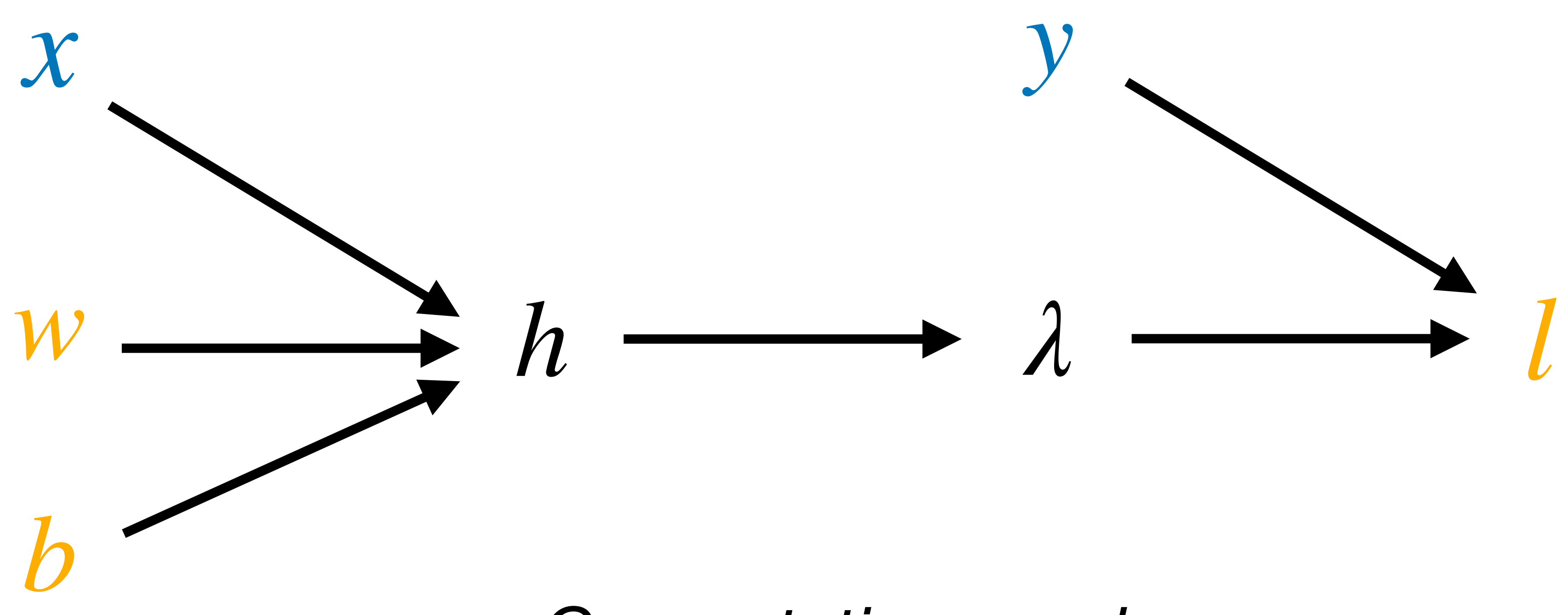
\+

?



$$l^{(i)} = \frac{1}{2}(1 - 0.8)^2 = 0.02$$

$$y^{(i)} = 1$$



Computation graph

Chain Rule in Neural Network

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

$$\frac{\partial l}{\partial w} = \underbrace{\frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}}\frac{\partial h}{\partial w}$$

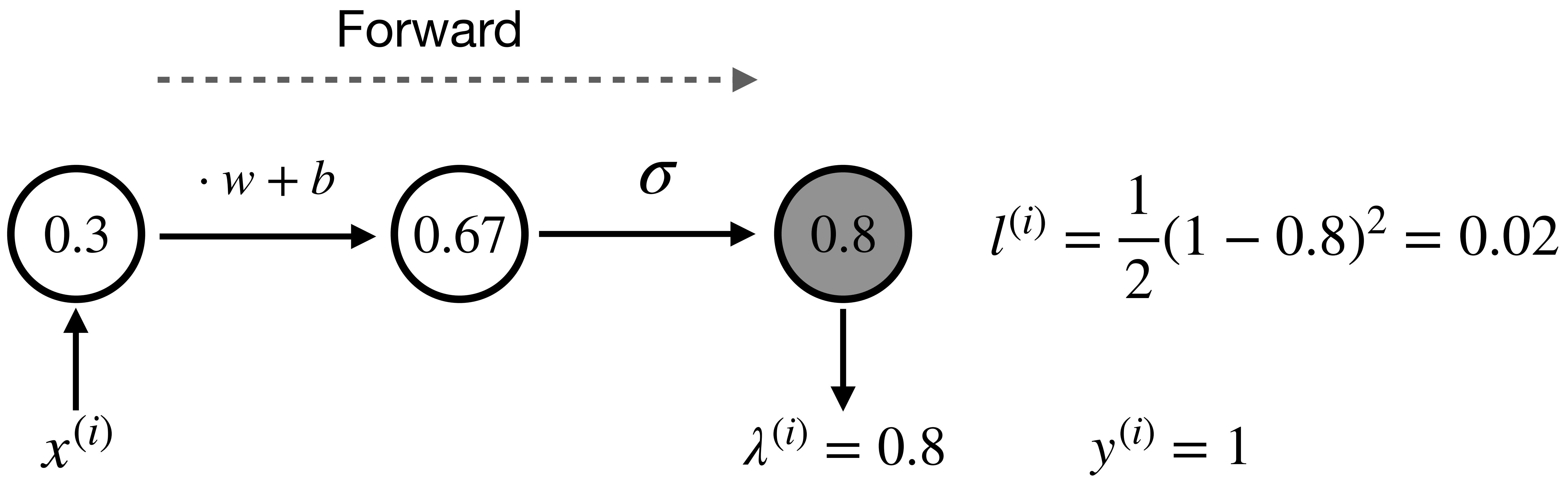
$$=$$

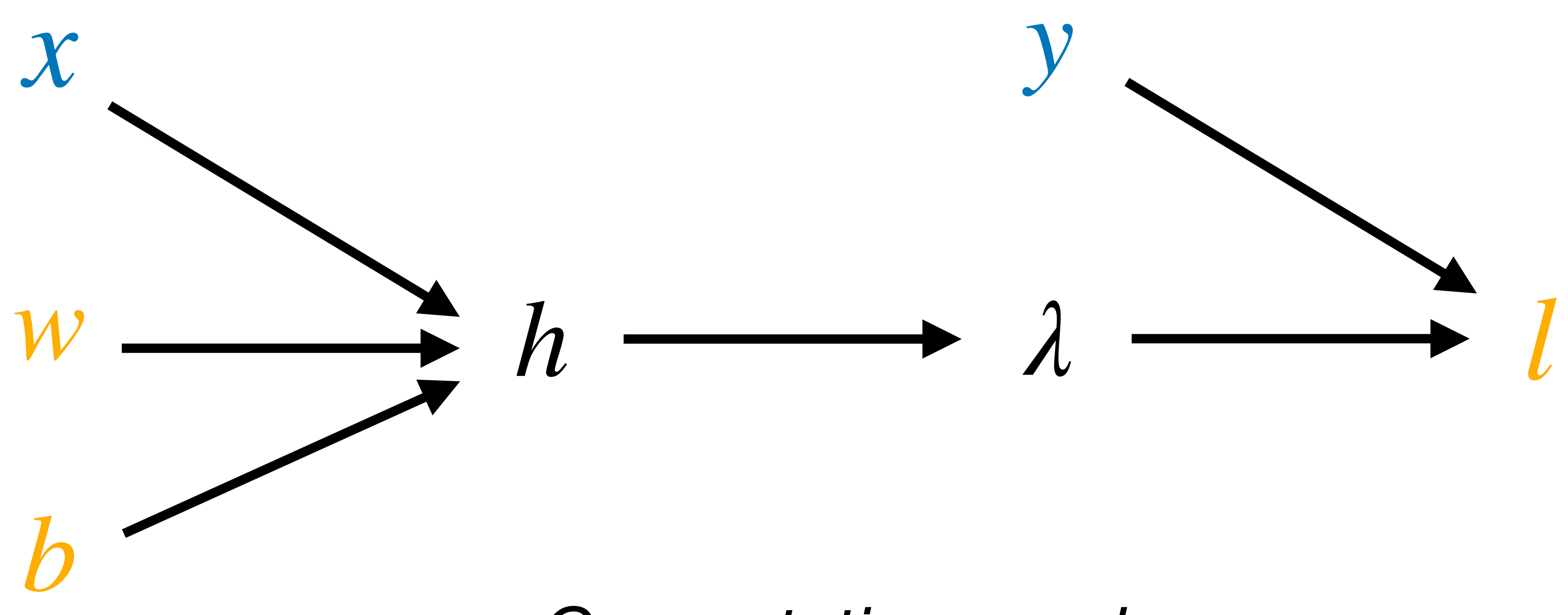
$$\frac{\partial l}{\partial b} = \overbrace{\frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}}\boxed{\frac{\partial h}{\partial b}}$$

3 derivatives

+

1 derivative





Computation graph

Forward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

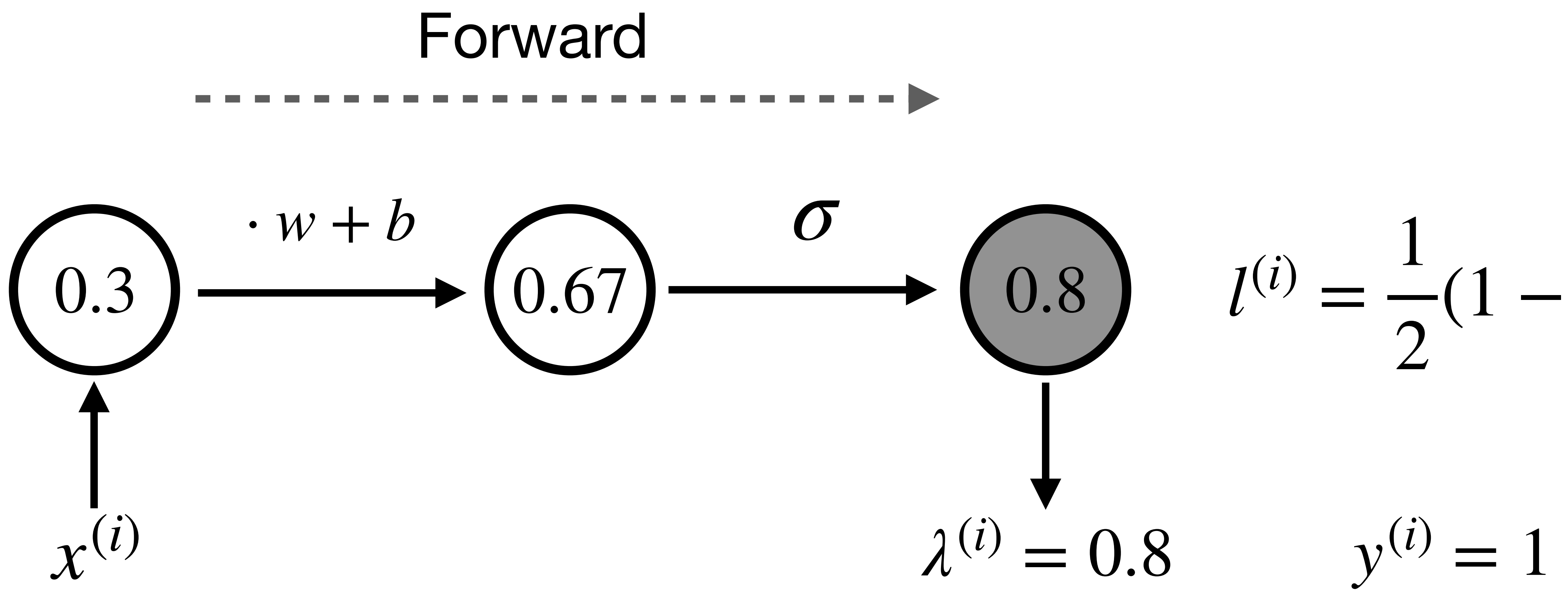
$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

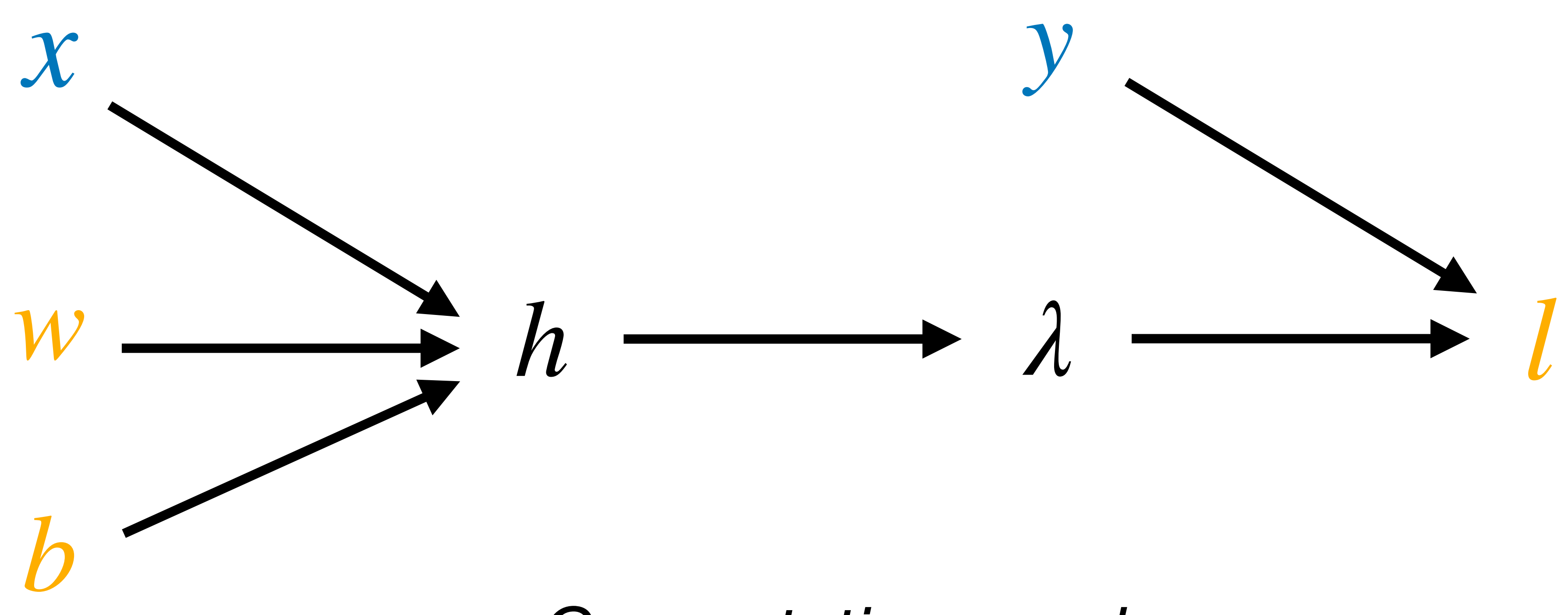
$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$



$$l^{(i)} = \frac{1}{2}(1 - 0.8)^2 = 0.02$$

Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

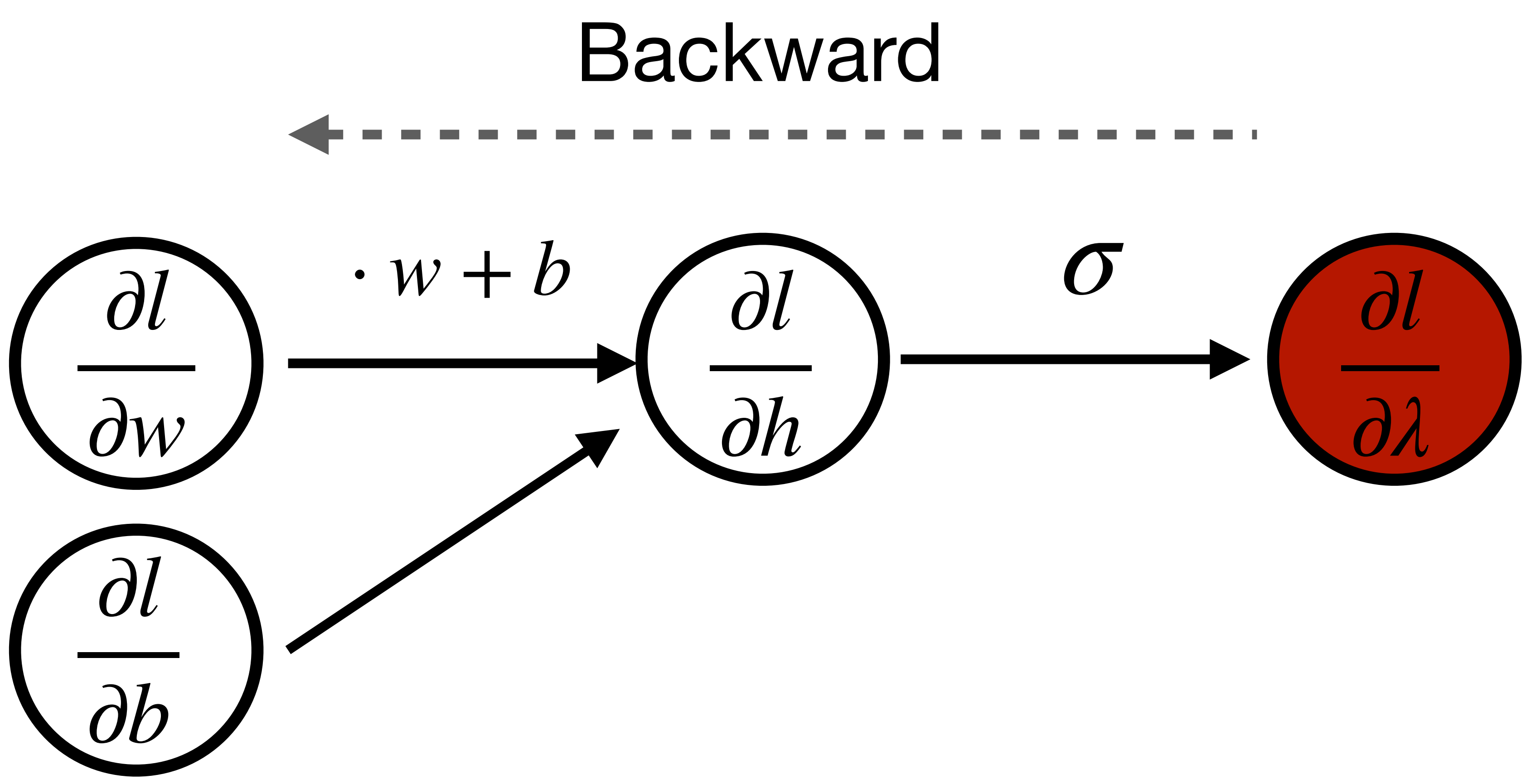


Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Forward pass equations **Backward pass equations**

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

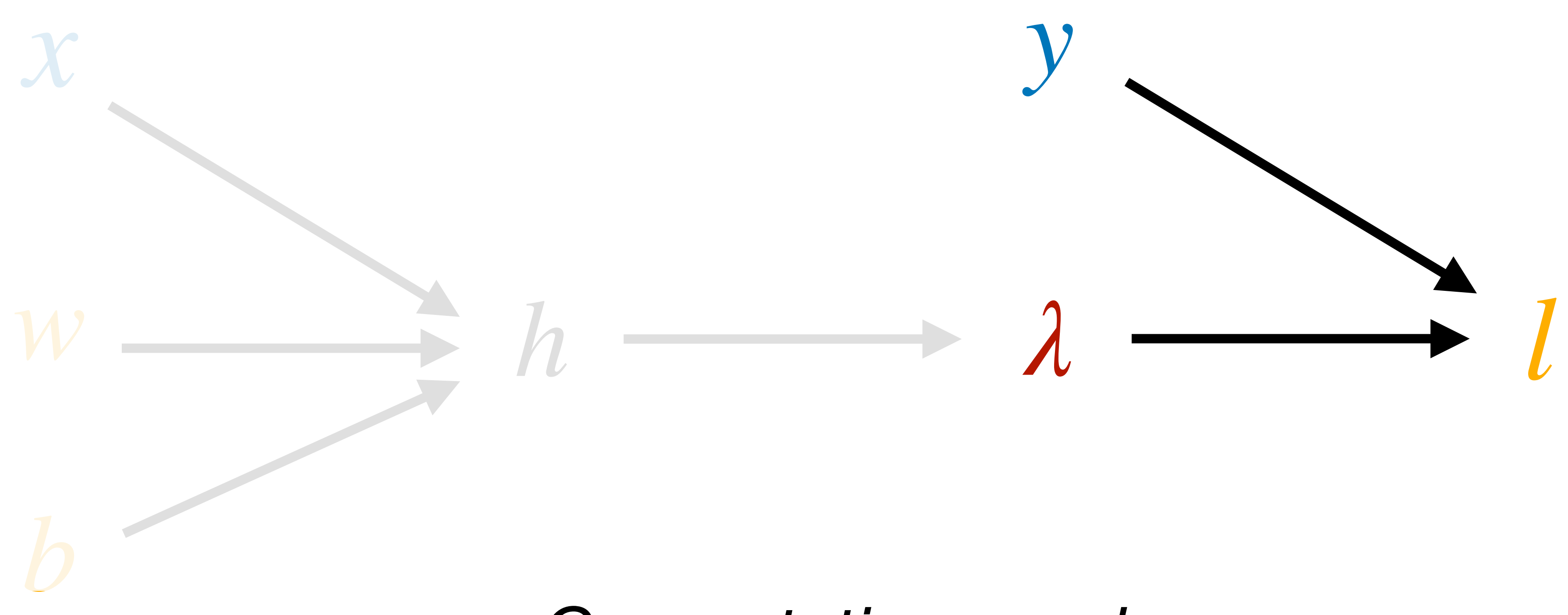
$$h = xw + b$$

Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda} \frac{\partial \lambda}{\partial h} \frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda} \frac{\partial \lambda}{\partial h} \frac{\partial h}{\partial b}$$



Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

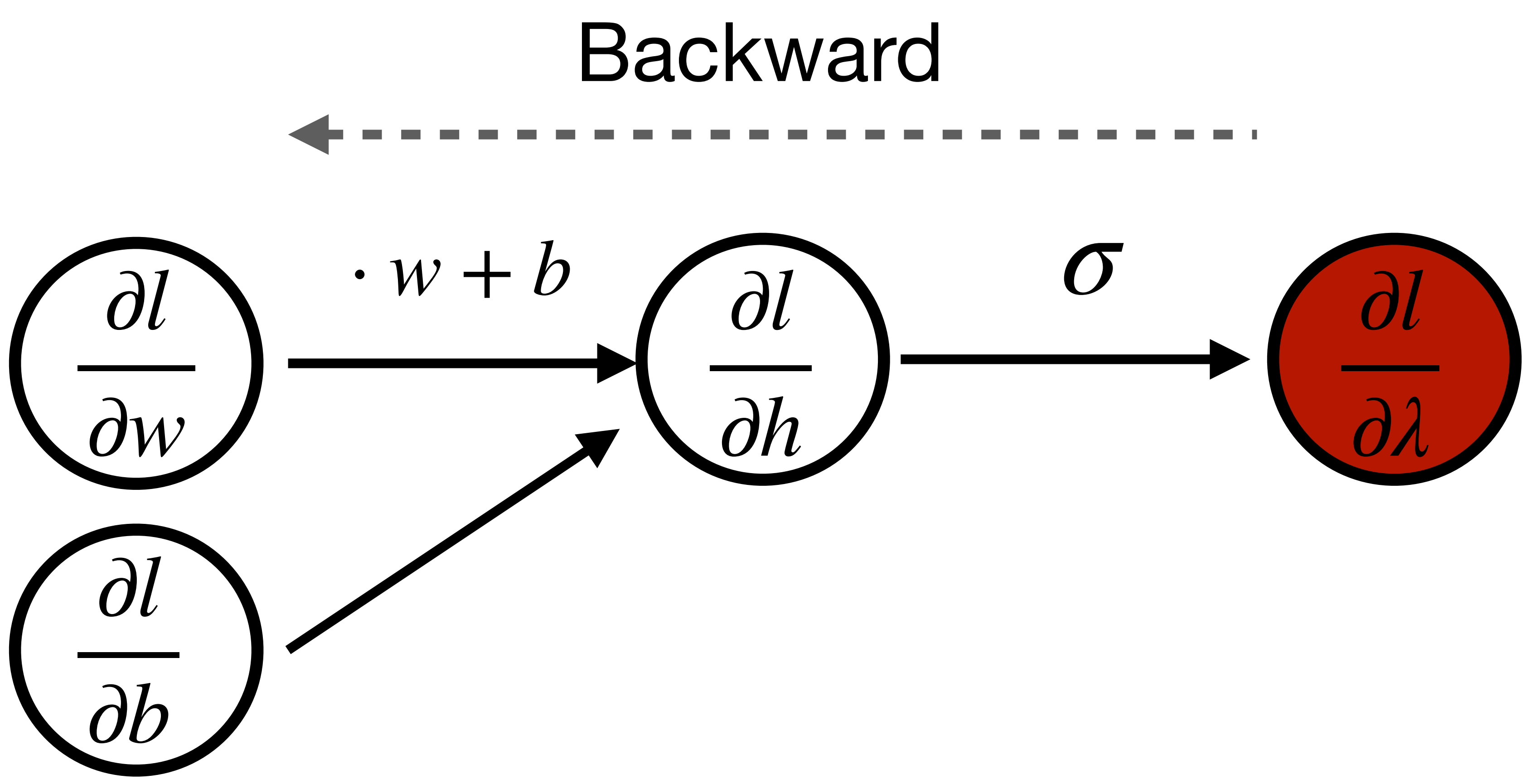
$$h = xw + b$$

Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

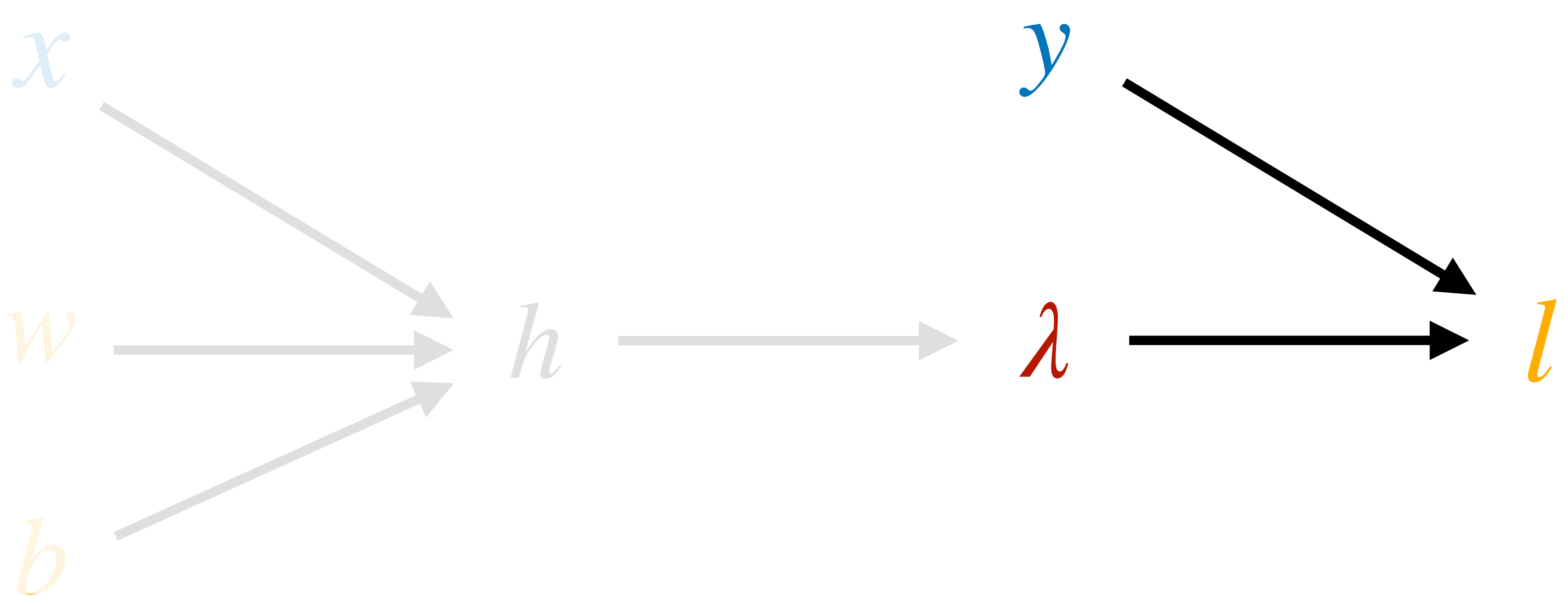
$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$



Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

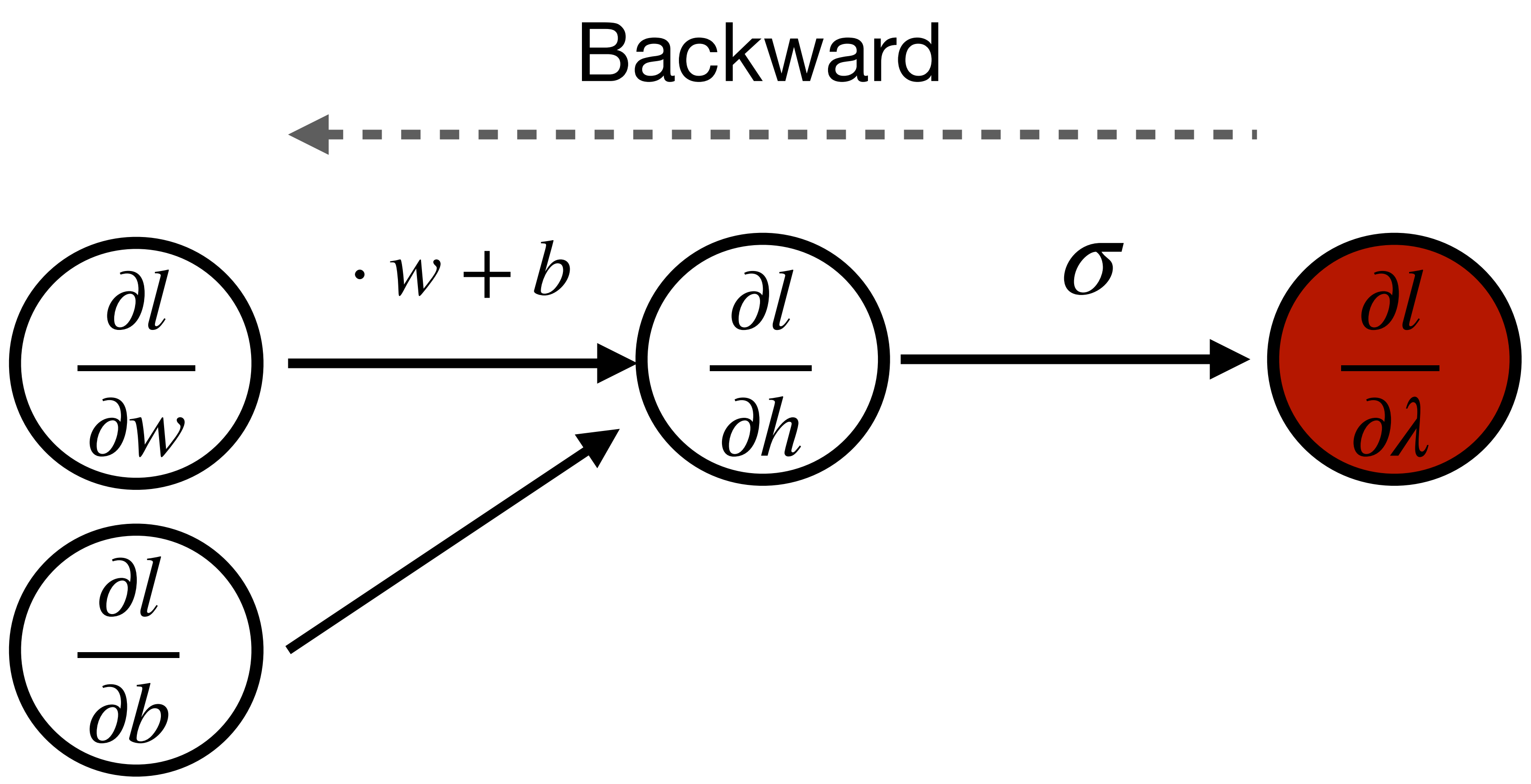


Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

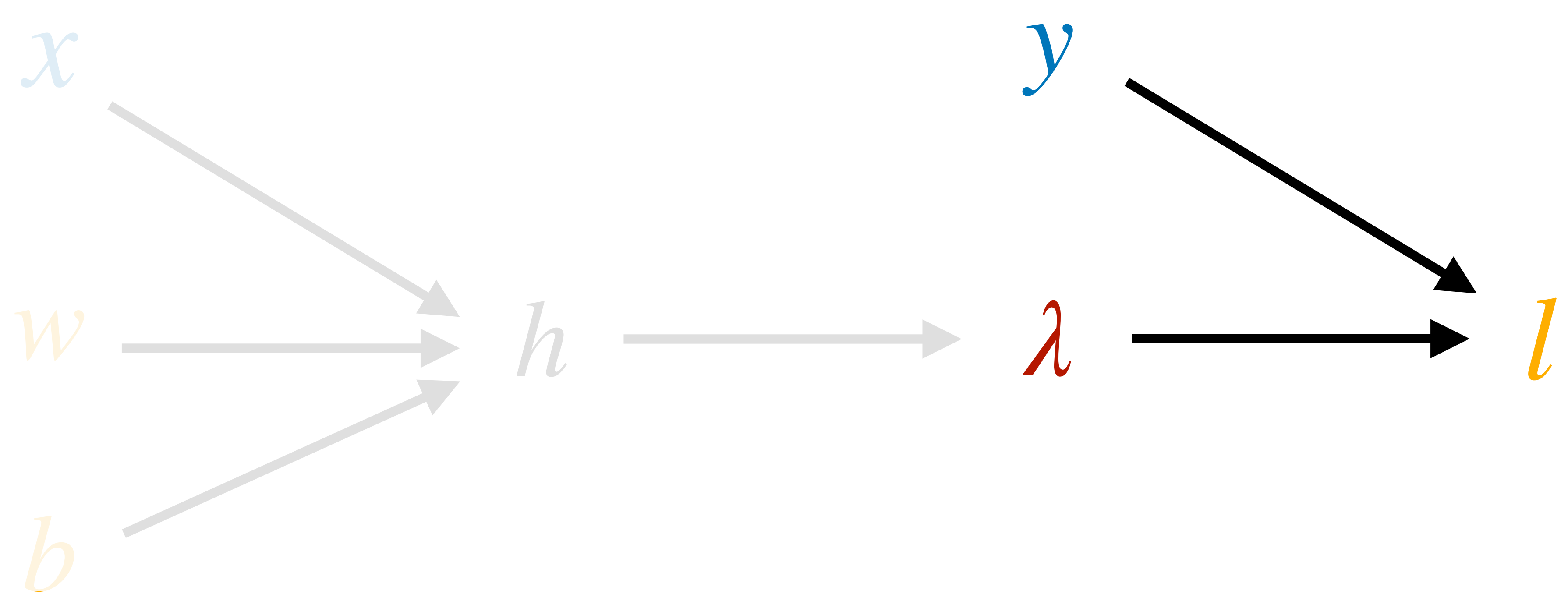
Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l^{(i)}}{\partial \lambda} \leftarrow (\lambda^{(i)} - y^{(i)}) = -0.2$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$

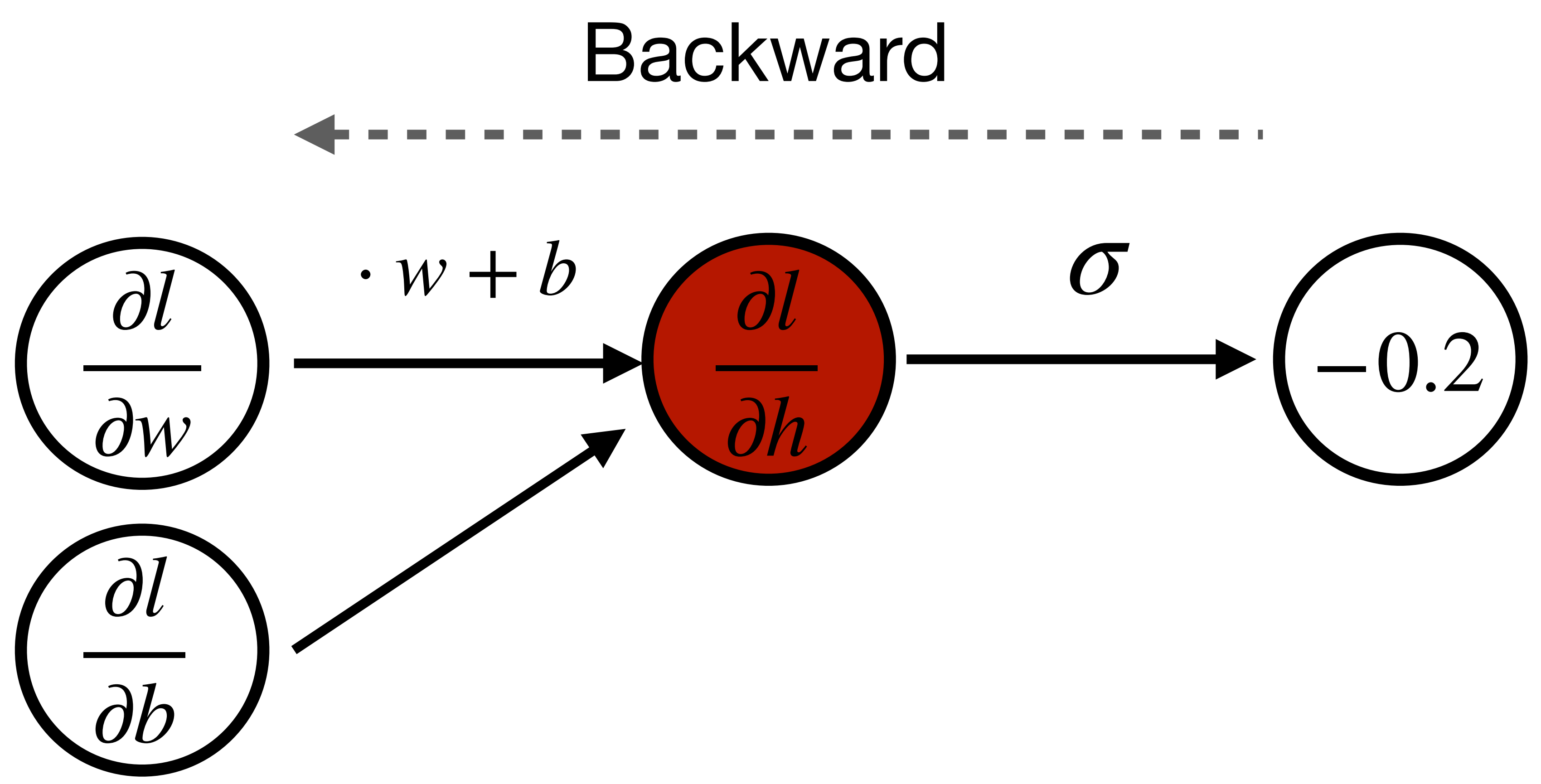


Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

$$\frac{\partial l}{\partial h} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}$$

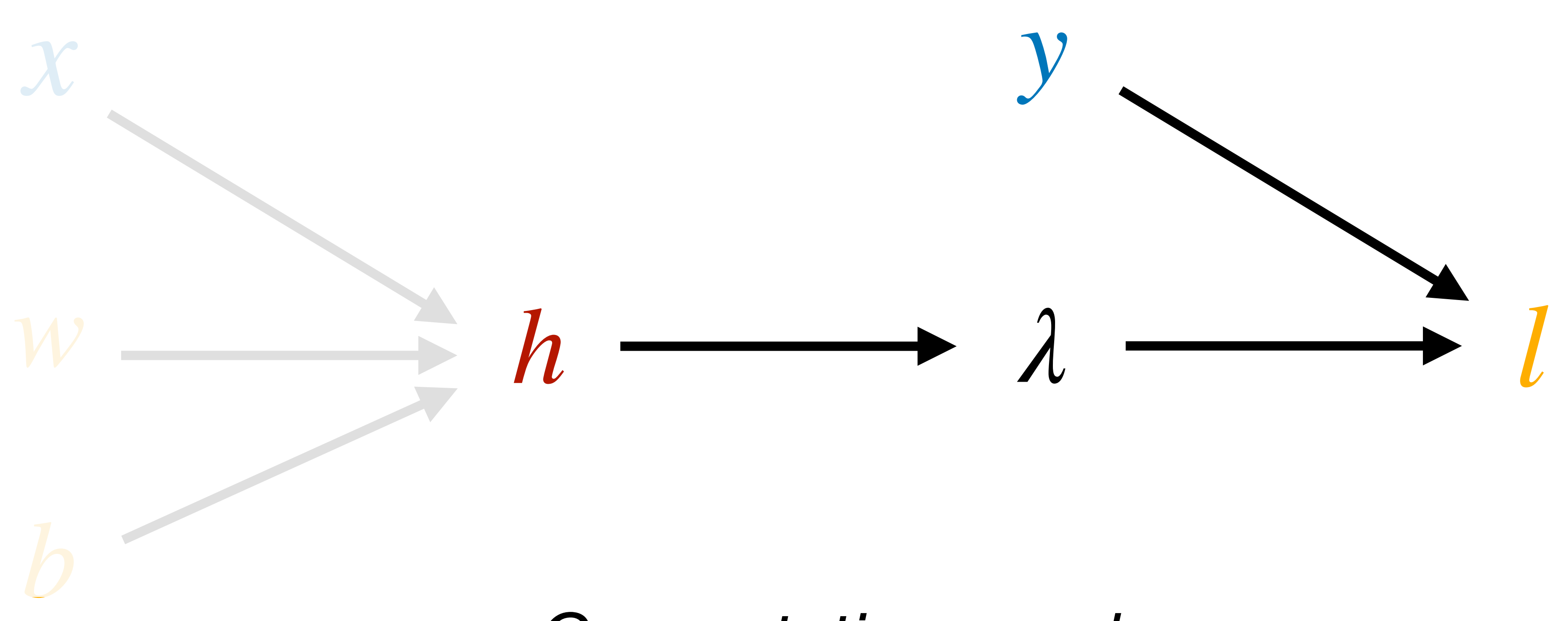
Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l^{(i)}}{\partial \lambda} \leftarrow -0.2$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$



Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

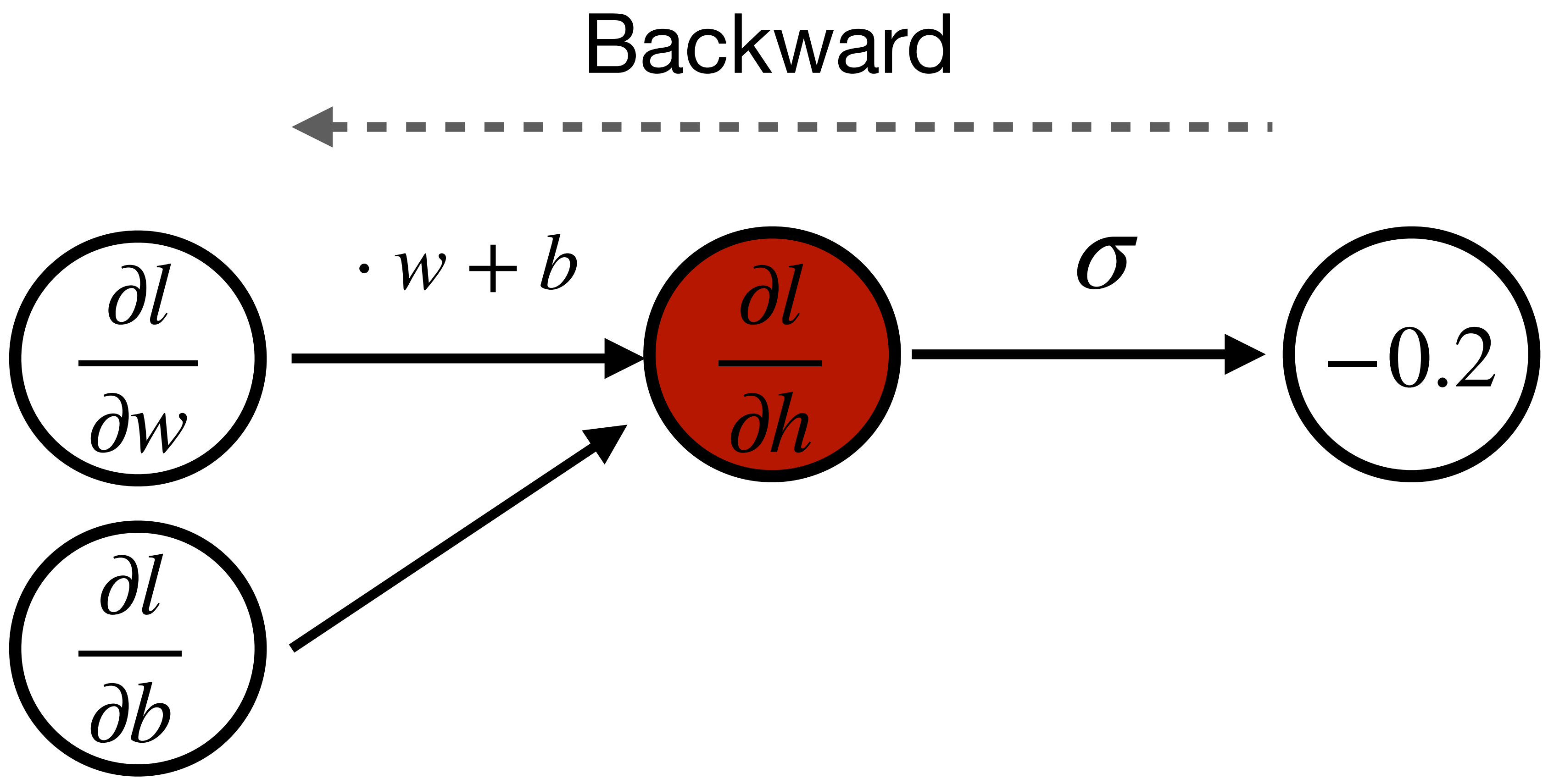
Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

$$\frac{\partial l}{\partial h} = \frac{\partial l}{\partial \lambda}\sigma'(h)$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

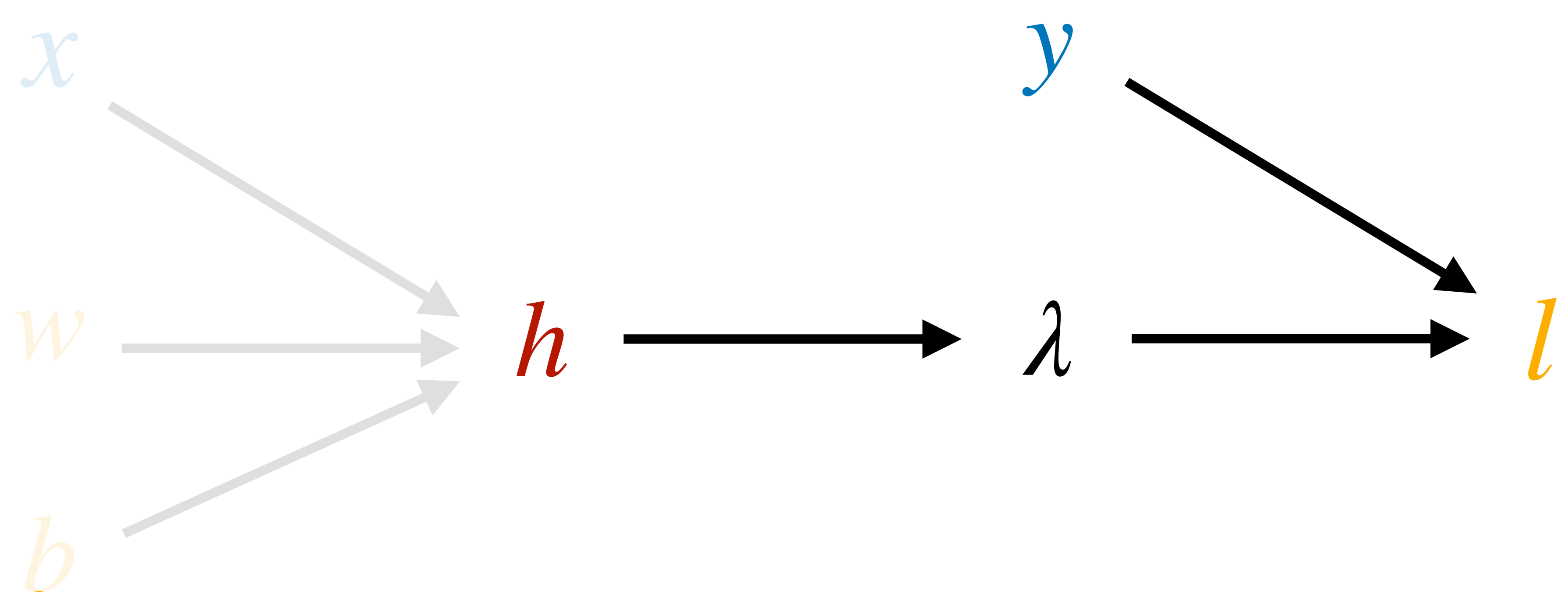
$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$



Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l^{(i)}}{\partial \lambda} \leftarrow -0.2$$

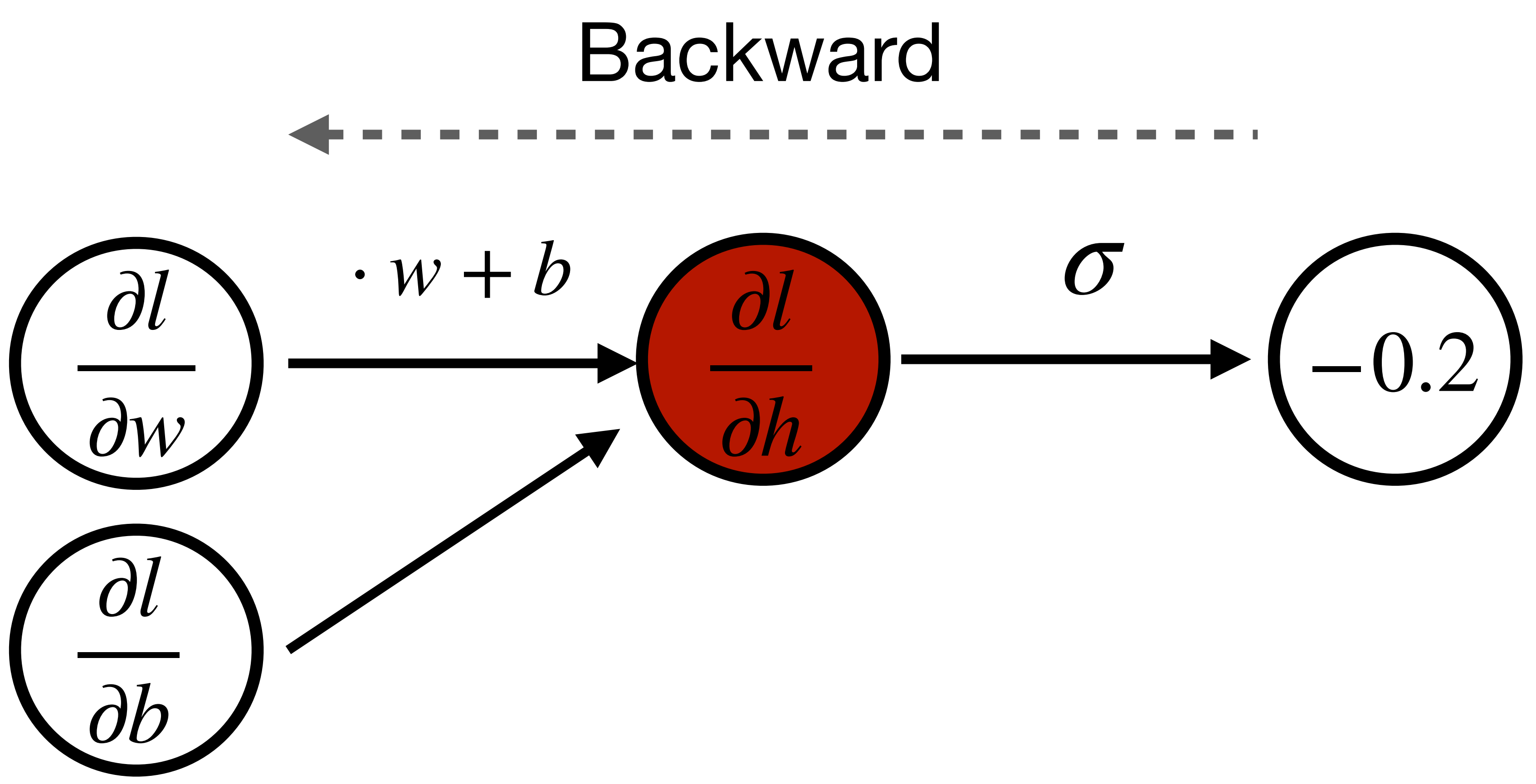


Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

$$\frac{\partial l}{\partial h} = \frac{\partial l}{\partial \lambda}\lambda(1 - \lambda)$$

Stored

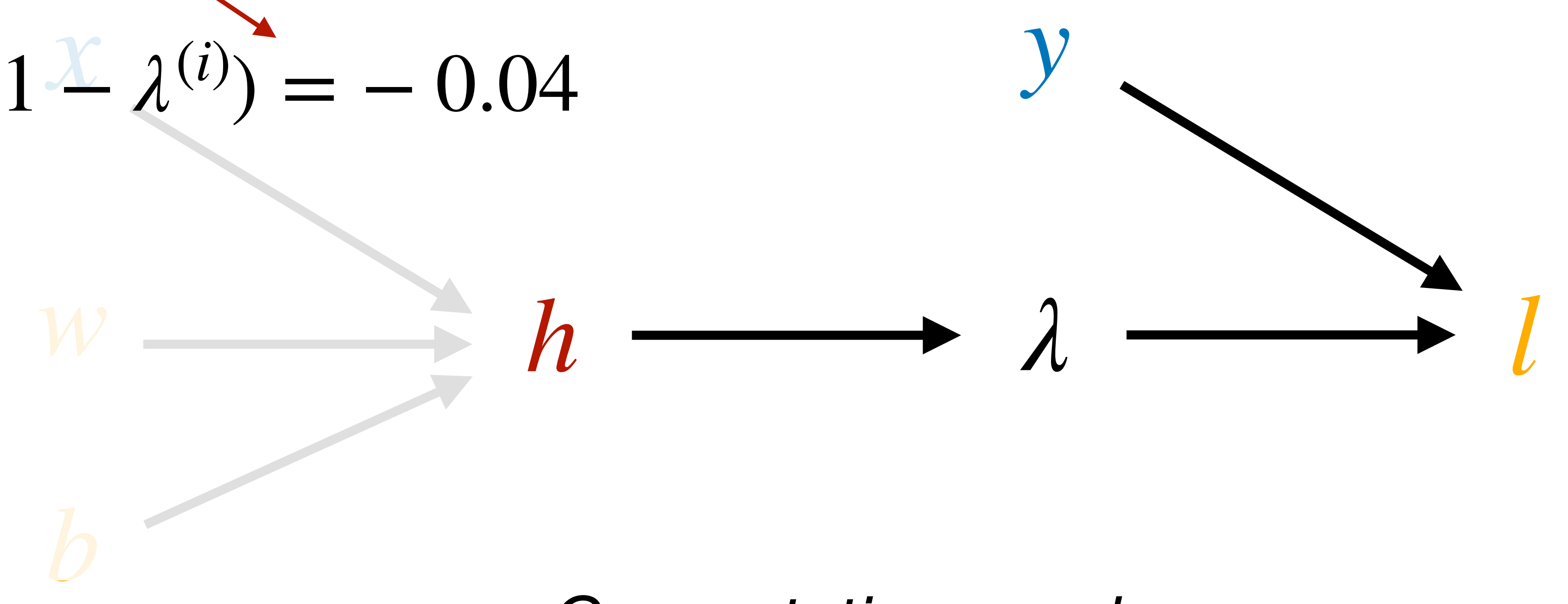
$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l^{(i)}}{\partial \lambda} \leftarrow -0.2$$

$$\frac{\partial l^{(i)}}{\partial h} \leftarrow \frac{\partial l^{(i)}}{\partial \lambda}\lambda^{(i)}(1 - \lambda^{(i)}) = -0.04$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$



Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$

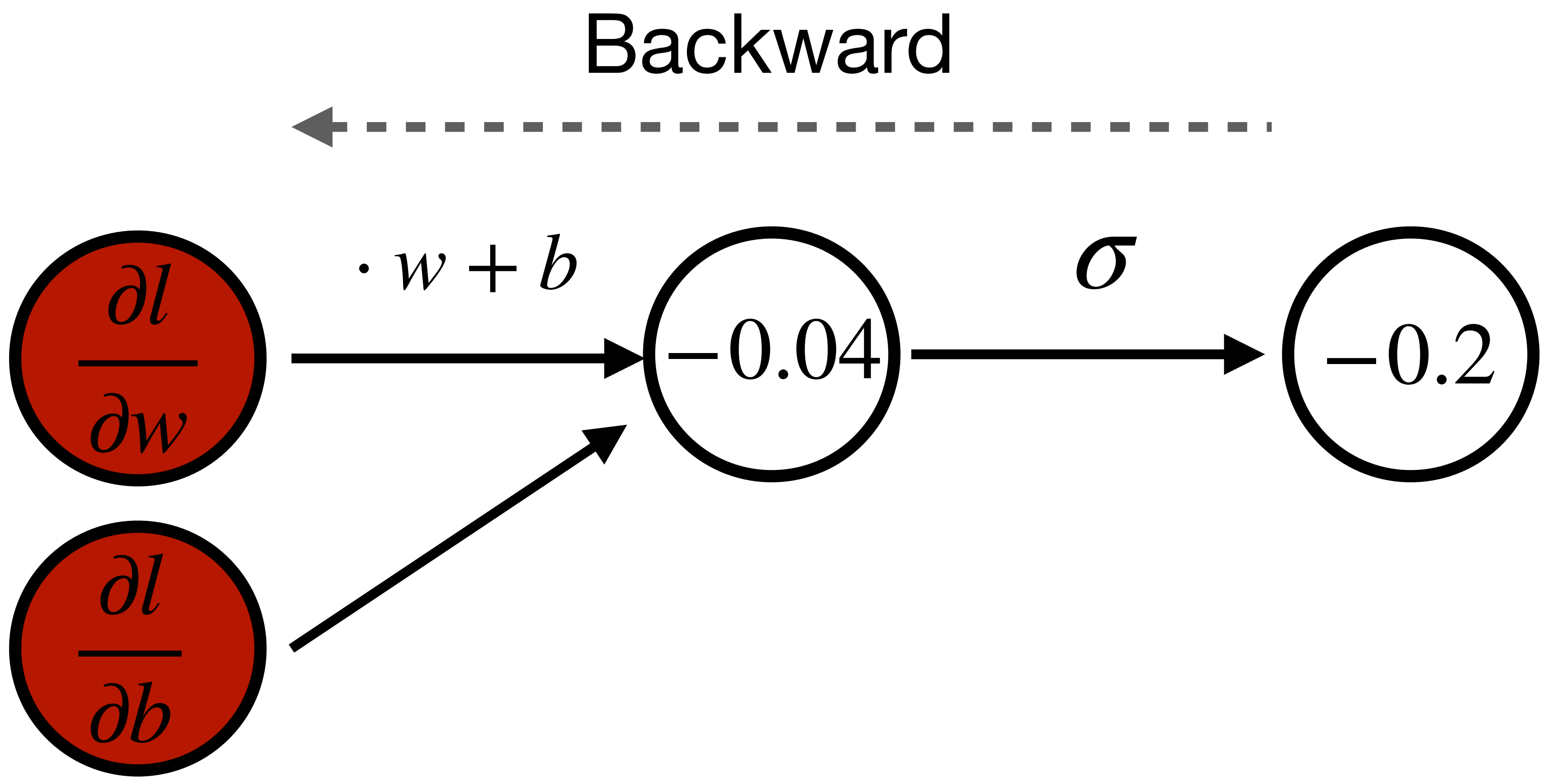
Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

$$\frac{\partial l}{\partial h} = \frac{\partial l}{\partial \lambda}\lambda(1 - \lambda)$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial h}\frac{\partial h}{\partial b}$$

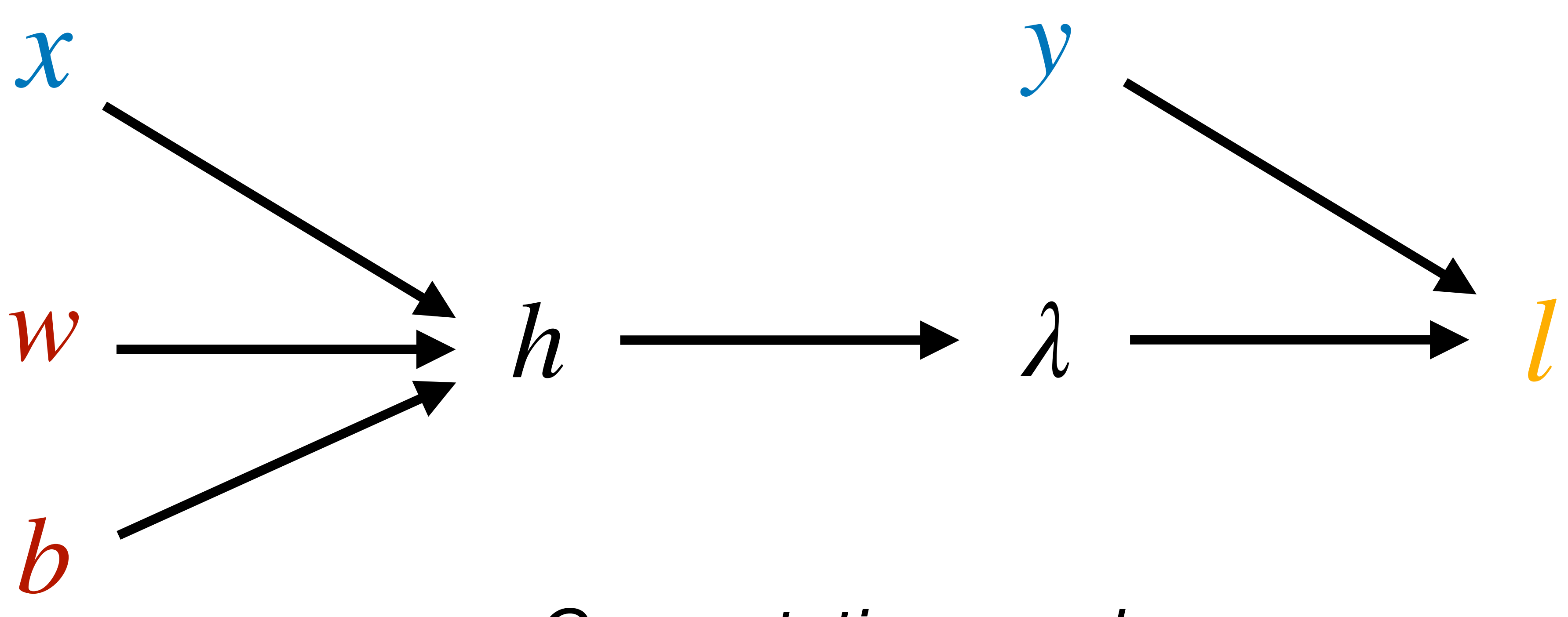


Stored

$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l^{(i)}}{\partial \lambda} \leftarrow -0.2$$

$$\frac{\partial l^{(i)}}{\partial h} \leftarrow -0.04$$

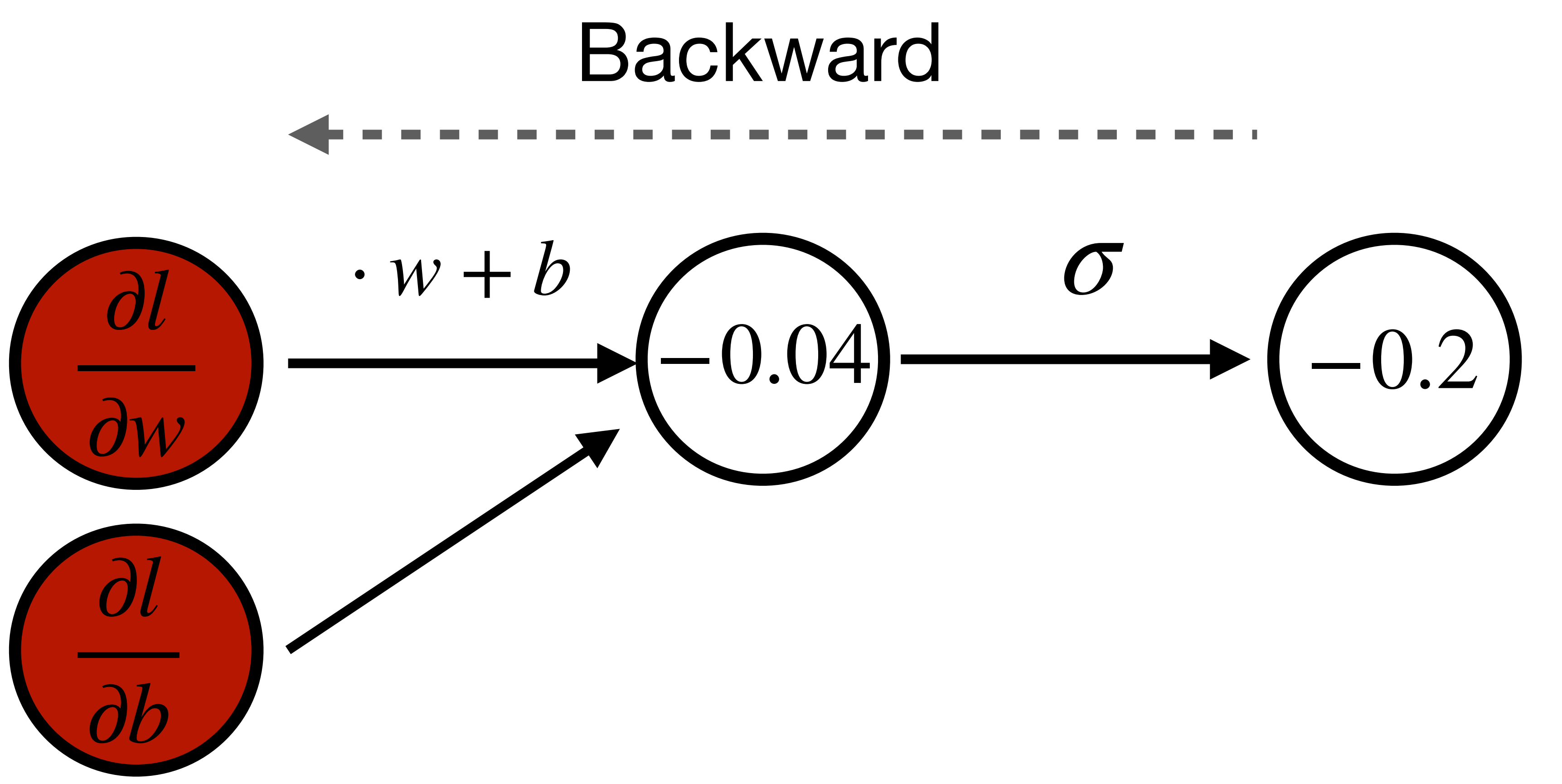


Computation graph

Backward Pass

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$



Forward pass equations

$$l = \frac{1}{2}(y - \lambda)^2$$

$$\lambda = \sigma(h)$$

$$h = xw + b$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial \lambda}\frac{\partial \lambda}{\partial h}\frac{\partial h}{\partial b}$$

Backward pass equations

$$\frac{\partial l}{\partial \lambda} = (\lambda - y)$$

$$\frac{\partial l}{\partial h} = \frac{\partial l}{\partial \lambda}\lambda(1 - \lambda)$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial h}x$$

$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial h}$$

Stored

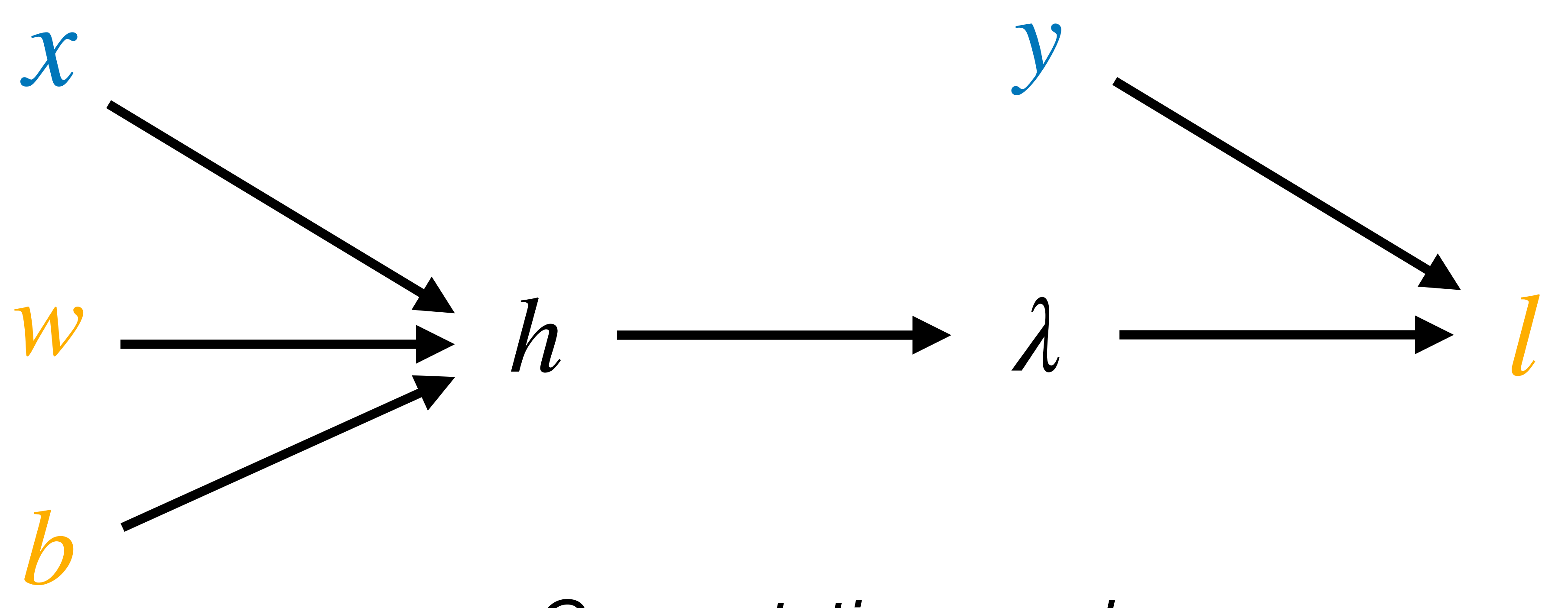
$$(x^{(i)}, h^{(i)}, \lambda^{(i)}, y^{(i)}, l^{(i)}) \leftarrow (0.3, 0.67, 0.8, 1, 0.02)$$

$$\frac{\partial l^{(i)}}{\partial \lambda} \leftarrow -0.2$$

$$\frac{\partial l^{(i)}}{\partial h} \leftarrow -0.04$$

$$\frac{\partial l^{(i)}}{\partial w} \leftarrow -0.26$$

$$\frac{\partial l^{(i)}}{\partial b} \leftarrow -0.04$$



Computation graph

Stochastic Gradient Descent

Example: univariate regression

$$l = \frac{1}{2}(y - \sigma(xw + b))^2$$

Step 1. Compute the derivatives $\nabla L = \left(\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}\right)$

$$\frac{\partial l^{(i)}}{\partial w} = -0.26$$

$$\frac{\partial l^{(i)}}{\partial b} = -0.04$$

$$\rightarrow \left\{\frac{\partial l^{(i)}}{\partial w}, \frac{\partial l^{(i)}}{\partial b}\right\}_i$$ for the mini-batch $$\rightarrow$$

$$\frac{\partial L}{\partial w} \approx \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}$$

$$\frac{\partial L}{\partial b} \approx \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial b}$$

Step 2. Update the parameters $w \leftarrow w - \alpha\frac{\partial L}{\partial w}$

$$b \leftarrow b - \alpha\frac{\partial L}{\partial b}$$

Backpropagation: exercise

Exercise 1

Consider the following forward pass equation for a two layer network

$$
\begin{aligned}
z_1 &= x \cdot W_1 + b_1 \\
a_1 &= ReLU(z_1) \\
z_2 &= a_1 \cdot W_2 + b_2 \\
\lambda &= \sigma(z_2) \\
l &= -y\log(\lambda) - (1-y)\log(1-\lambda)
\end{aligned}
$$

$$x^{(i)} \in \mathbb{R}^N,\ y^{(i)} \in \mathbb{R}^1,\ z_1 \in \mathbb{R}^{D_1}$$

1. What are the shapes of W_1, b_1, W_2, b_2? If we use the network across a mini-batch with m samples, what would be the shape of W_1, b_1, W_2, b_2? and the shape of the input x and output y?

2. Depict the computation graph

3. Write down the backward pass equations for $\dfrac{\partial l}{\partial W_2}, \dfrac{\partial l}{\partial b_2}$ *[hint: be careful with the dimensions!]**

Backpropagation: solution

Exercise 1

Consider the following forward pass equation for a two layer network

$$z_1 = x \cdot W_1 + b_1$$
$$a_1 = ReLU(z_1)$$
$$z_2 = a_1 \cdot W_2 + b_2$$
$$\lambda = \sigma(z_2)$$
$$l = -y\log(\lambda) - (1-y)\log(1-\lambda)$$

$x^{(i)} \in \mathbb{R}^N,\ y^{(i)} \in \mathbb{R}^1,\ z_1 \in \mathbb{R}^{D_1}$

1. What are the shapes of W_1, b_1, W_2, b_2? If we use the network across a mini-batch with m samples, what would be the shape of W_1, b_1, W_2, b_2? and the shape of the input x and output y?

1. $W_1 \in \mathbb{R}^{N \times D_1},\ b_1 \in \mathbb{R}^{D_1},\ W_2 \in \mathbb{R}^{D_1},\ b_2 \in \mathbb{R}^1$. Weights and bias do not change shape with number of input samples.
$x \in \mathbb{R}^{m,N},\ y \in \mathbb{R}^m$

Backpropagation: solution

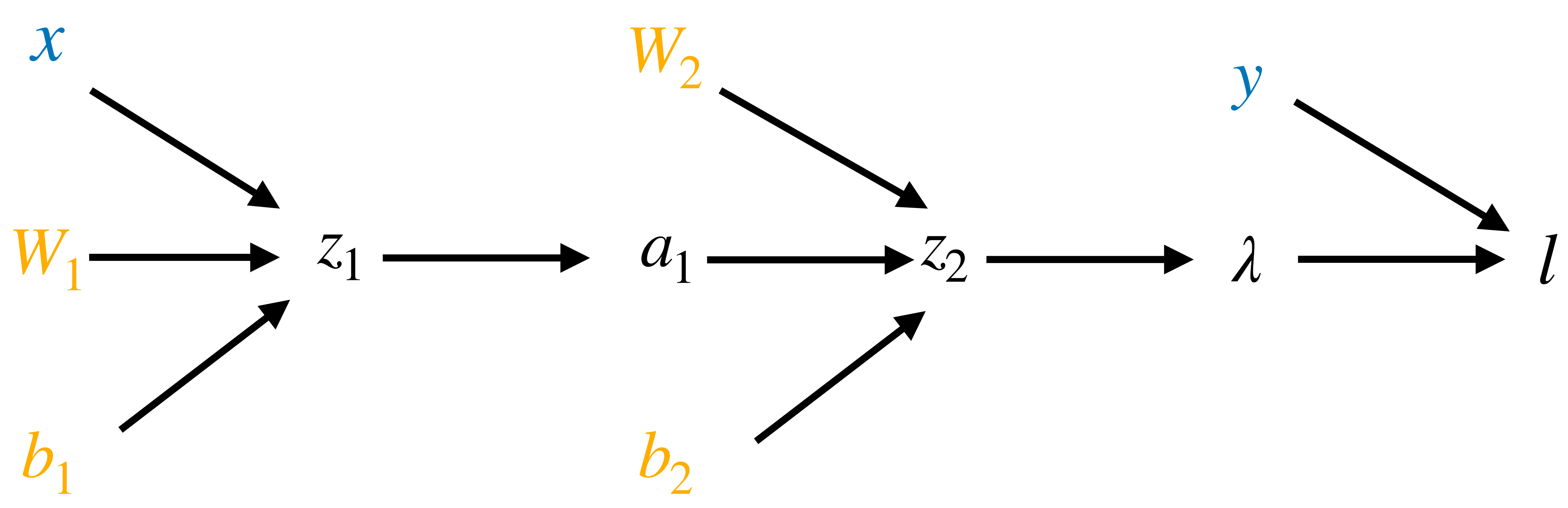
Exercise 1

Consider the following forward pass equation for a two layer network

$$
\begin{aligned}
z_1 &= x \cdot W_1 + b_1 \\
a_1 &= ReLU(z_1) \\
z_2 &= a_1 \cdot W_2 + b_2 \\
\lambda &= \sigma(z_2) \\
l &= -y\log(\lambda) - (1-y)\log(1-\lambda)
\end{aligned}
$$

$$x^{(i)} \in \mathbb{R}^N,\ y^{(i)} \in \mathbb{R}^1,\ z_1 \in \mathbb{R}^{D_1}$$

2. Depict the computation graph



Backpropagation: solution

Exercise 1

Consider the following forward pass equation for a two layer network

$$
\begin{aligned}
z_1 &= x \cdot W_1 + b_1 \\
a_1 &= ReLU(z_1) \\
z_2 &= a_1 \cdot W_2 + b_2 \\
\lambda &= \sigma(z_2) \\
l &= -y\log(\lambda) - (1-y)\log(1-\lambda)
\end{aligned}
$$

$x^{(i)} \in \mathbb{R}^N,\ y^{(i)} \in \mathbb{R}^1,\ z_1 \in \mathbb{R}^{D_1}$

3. Write down the backward pass equations for $\dfrac{\partial l}{\partial W_2}, \dfrac{\partial l}{\partial b_2}$

$$\frac{\partial l}{\partial \lambda} = -\frac{y}{\lambda} + \frac{(1-y)}{1-\lambda} \in \mathbb{R}$$

$$\frac{\partial l}{\partial z_2} = \frac{\partial l}{\partial \lambda}\sigma(z_2)(1-\sigma(z_2)) \in \mathbb{R}$$

$$\frac{\partial l}{\partial W_2} = \frac{\partial l}{\partial z_2} a_1 \in \mathbb{R}^{D_1} \qquad \frac{\partial l}{\partial b_2} = \frac{\partial l}{\partial z_2} \cdot 1 \in \mathbb{R}$$

Maximum Likelihood for Regression (optional)

Assume you have a training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ for $x, y \in \mathbb{R}$ and you want to train a neural network to predict the outcome y given the value of x.
We follow the steps to build the loss function according to the maximum likelihood criterion:

1. We assume the data are generated by a Gaussian distribution $p_{data}(y\,|\,x) \sim \mathcal{N}(\mu_x, \sigma = 1)$
2. We want to learn $p_{model}(y\,|\,\lambda_x) \approx p_{data}(y\,|\,x)$
3. We set the network to output an approximation of the mean of $f(x;\theta) = \lambda_x \approx \mu_x$

What is the loss function build according to maximum likelihood criterion for this problem?

*where p_{model} as p_{data} are intended as probability density functions (it's a continuous space problem)

Maximum Likelihood for Regression (optional)

Assume you have a training set $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,n}$ for $x, y \in \mathbb{R}$ and you want to train a neural network to predict the outcome y given the value of x.
We follow the steps to build the loss function according to the maximum likelihood criterion:

1. We assume the data are generated by a Gaussian distribution $p_{data}(y\,|\,x) \sim \mathcal{N}(\mu_x, \sigma = 1)$
2. We want to learn $p_{model}(y\,|\,\lambda_x) \approx p_{data}(y\,|\,x)$
3. We set the network to output an approximation of the mean of $f(x;\theta) = \lambda_x \approx \mu_x$

What is the loss function build according to maximum likelihood criterion for this problem?

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} -\log(p_{model}(y^{(i)}\,|\,f(x^{(i)};\theta))) = \frac{1}{n}\sum_{i=1}^{n} -\log\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{\left(y^{(i)}-\lambda^{(i)}\right)^2}{2}}\right) = \frac{1}{n}\sum_{i=1}^{n}\log(\sqrt{2\pi}) + \frac{\left(y^{(i)}-\lambda^{(i)}\right)^2}{2}$$

$$= \text{const} + \frac{1}{n}\sum_{i=1}^{n}\frac{\left(y^{(i)}-\lambda^{(i)}\right)^2}{2} = L_{SE}$$

This is the maximum likelihood formulation for a regression problem and indeed the loss is the mean squared error :)

*where p_{model} as p_{data} are intended as probability density functions (it's a continuous space problem)

Summary

Topics

- Loss function
- Maximum-likelihood
- Stochastic Gradient Discent
- Backpropagation algorithm

Reading material

- *Understanding Deep Learning* - Chapter 5 (5.1, 5.2, 5.7), Chapter 6 (6.1)
- *Deep Learning book* - Chapter 5.5
- Backpropagation1 Backpropagation2