Part 2: Building a Network and Fully-Connected Networks

Mikhail Romanov, Igor Slinko

Dataset

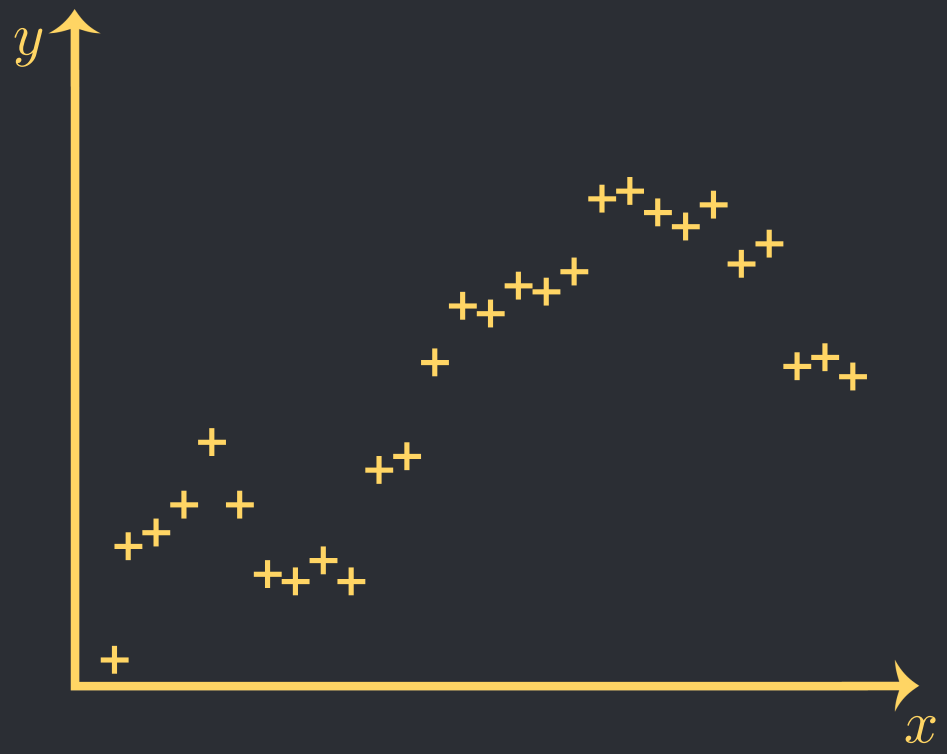
$\mathbf{x}_1$	$\mathbf{y}_1$
$\mathbf{x}_2$	$\mathbf{y}_2$
$\mathbf{x}_3$	$\mathbf{y}_3$
$\mathbf{x}_4$	$\mathbf{y}_4$
$\mathbf{x}_5$	$\mathbf{y}_5$
$\mathbf{x}_6$	$\mathbf{y}_6$
$\dots$	$\dots$
$\mathbf{x}_S$	$\mathbf{y}_S$

Dataset

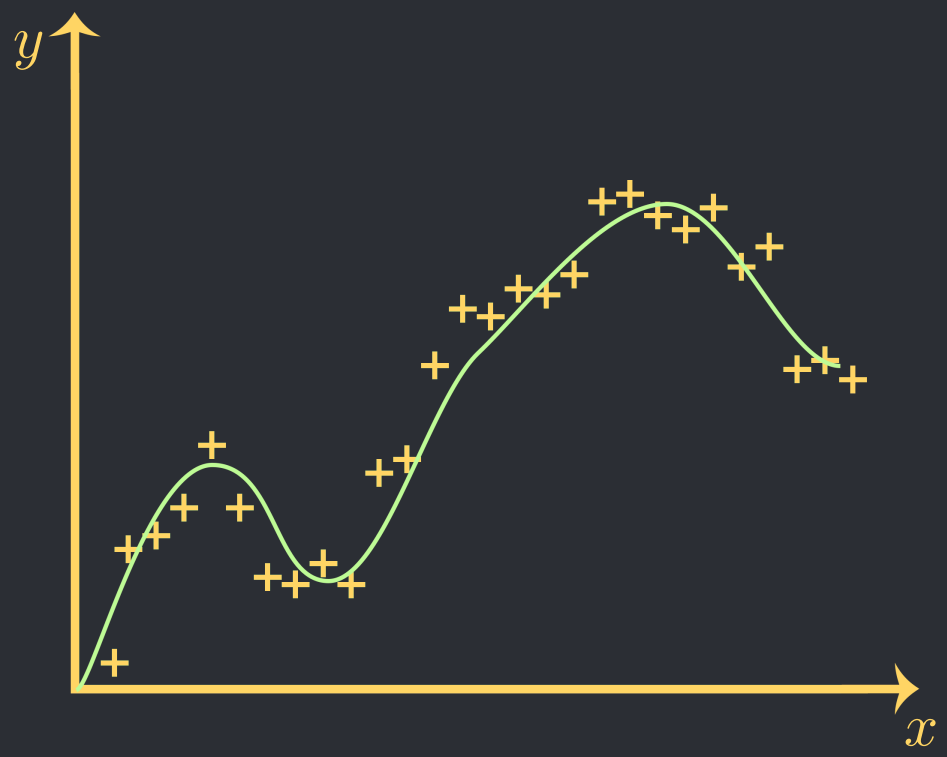
- Split in three subsets: train, validation, tests

$\mathbf{x}_1$	$\mathbf{y}_1$
$\mathbf{x}_2$	$\mathbf{y}_2$
$\mathbf{x}_3$	$\mathbf{y}_3$
$\mathbf{x}_4$	$\mathbf{y}_4$
$\mathbf{x}_5$	$\mathbf{y}_5$
$\mathbf{x}_6$	$\mathbf{y}_6$
...	...
$\mathbf{x}_S$	$\mathbf{y}_S$

Problem Example: Regression (Data Fitting)

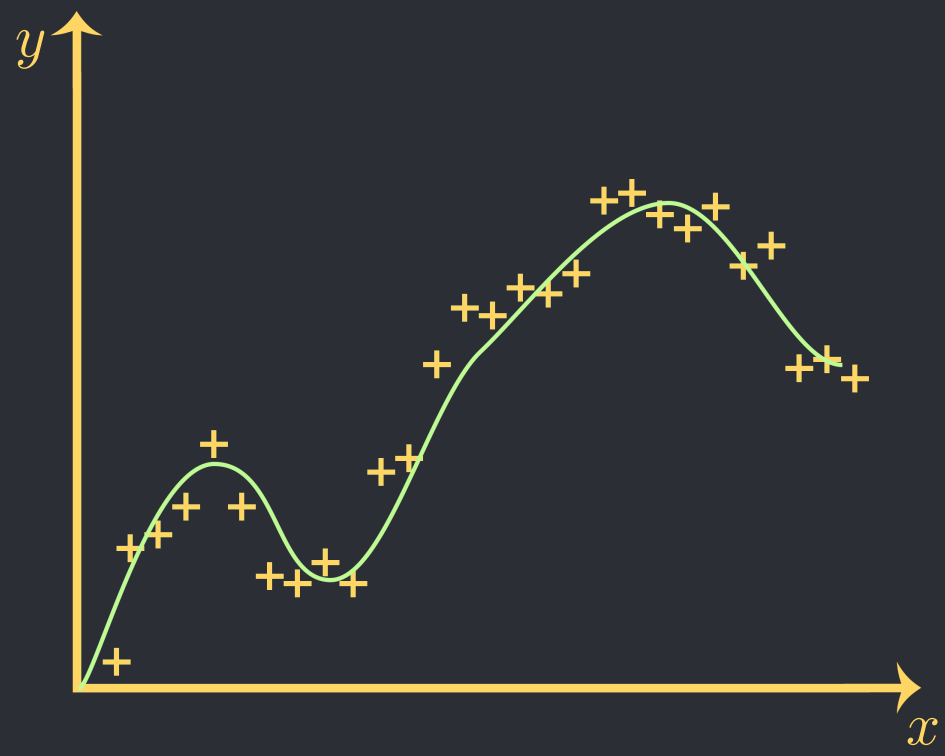


Problem Example: Regression (Data Fitting)



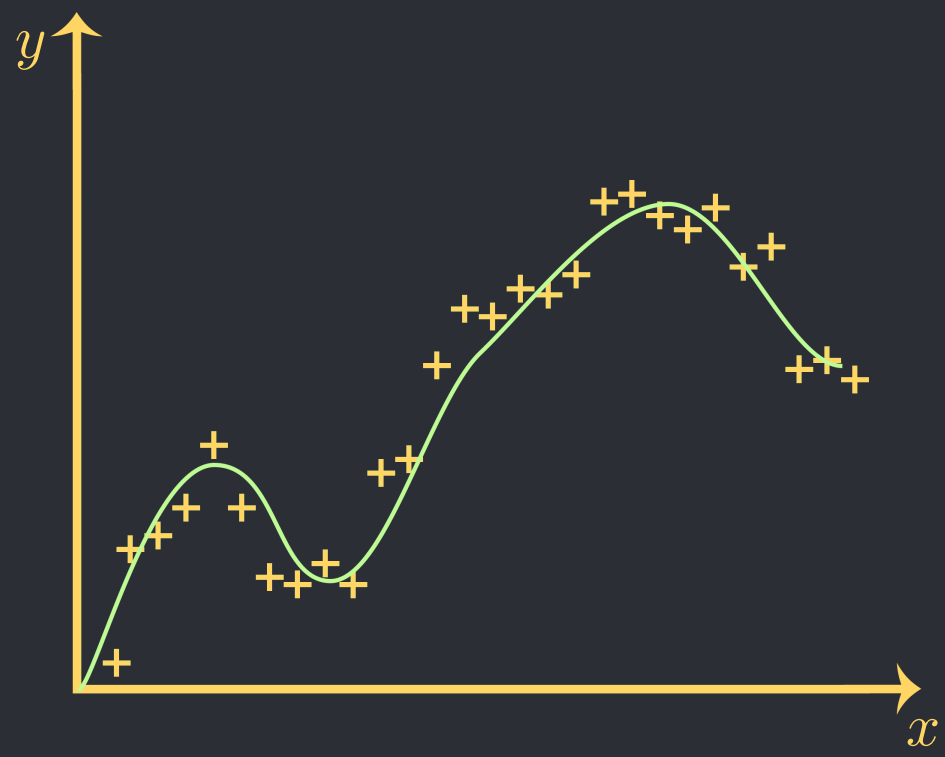
Problem Example: Regression (Data Fitting)

$$\mathbf{y}_s = f(\mathbf{x}_s)$$

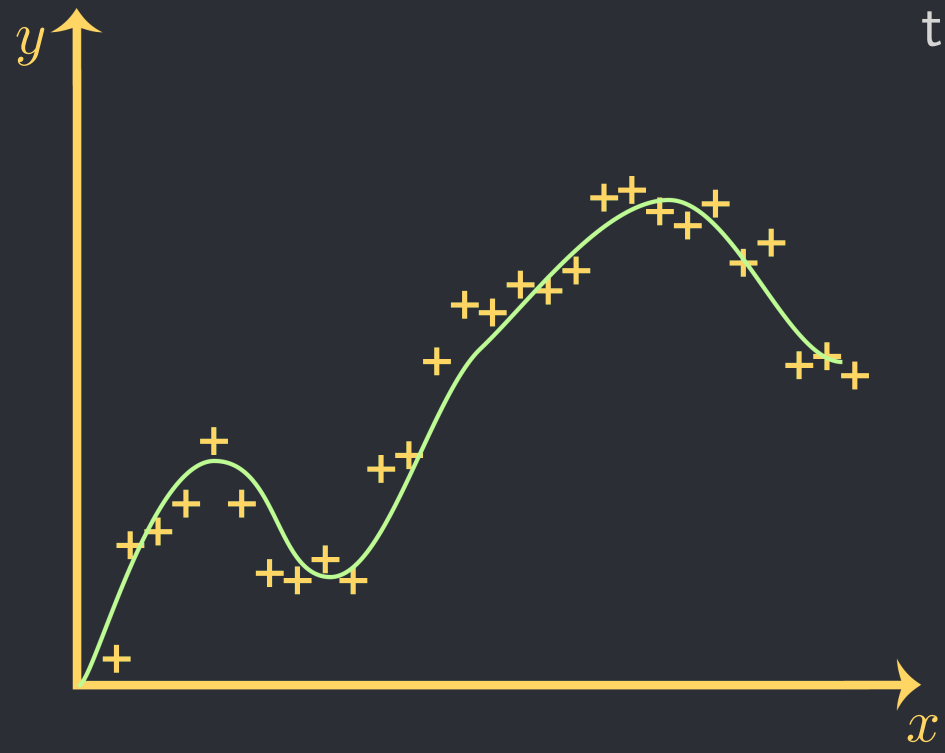


Problem Example: Regression (Data Fitting)

$$\mathbf{y}_s = f(\mathbf{x}_s) + \boldsymbol{\varepsilon}_s$$

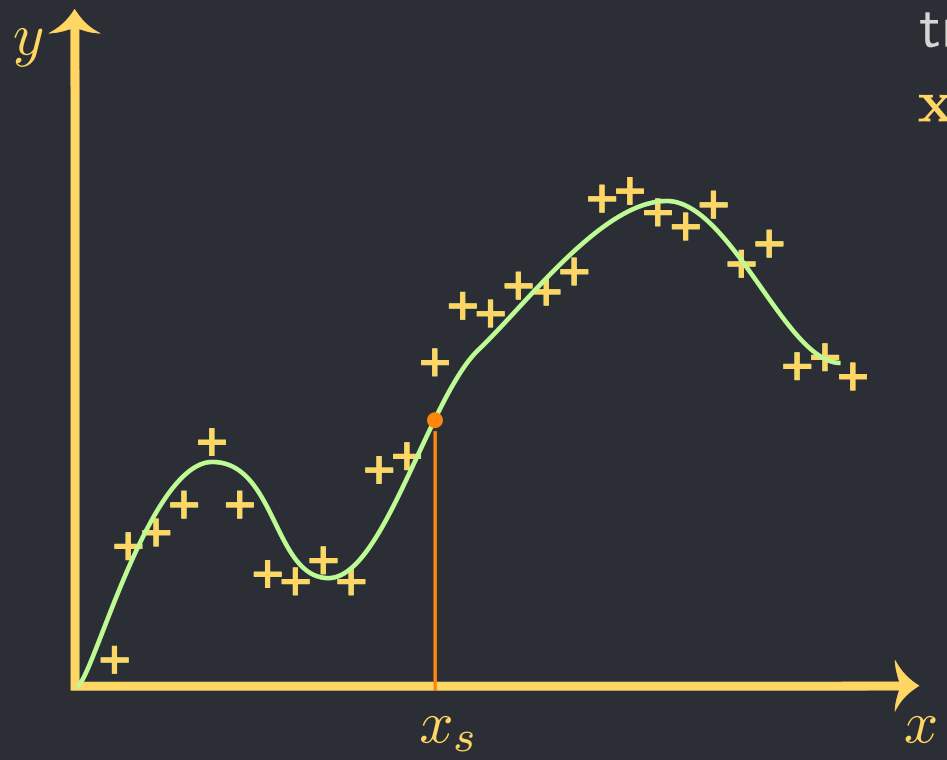


Problem Example: Regression (Data Fitting)



$\mathbf{y}_s = f(\mathbf{x}_s) + \boldsymbol{\varepsilon}_s$
$f(\mathbf{x}_s)$ – unobservable ground truth function

Problem Example: Regression (Data Fitting)

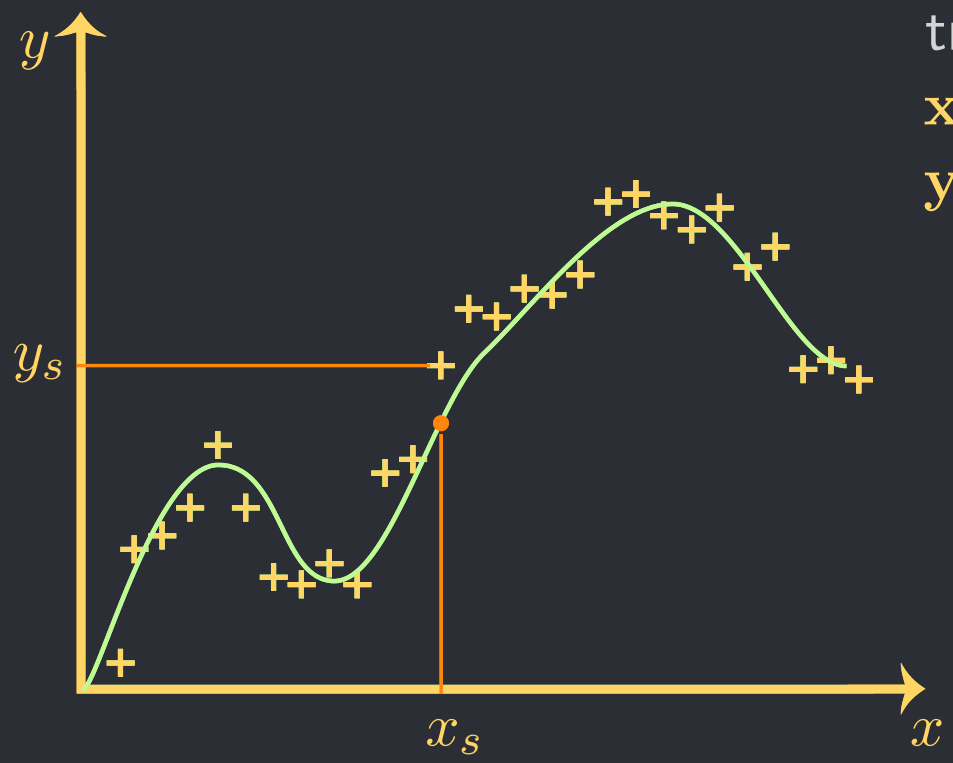


$\mathbf{y}_s = f(\mathbf{x}_s) + \boldsymbol{\varepsilon}_s$

$f(\mathbf{x}_s)$ – unobservable ground truth function

$\mathbf{x}_s$ – measurement conditions

Problem Example: Regression (Data Fitting)



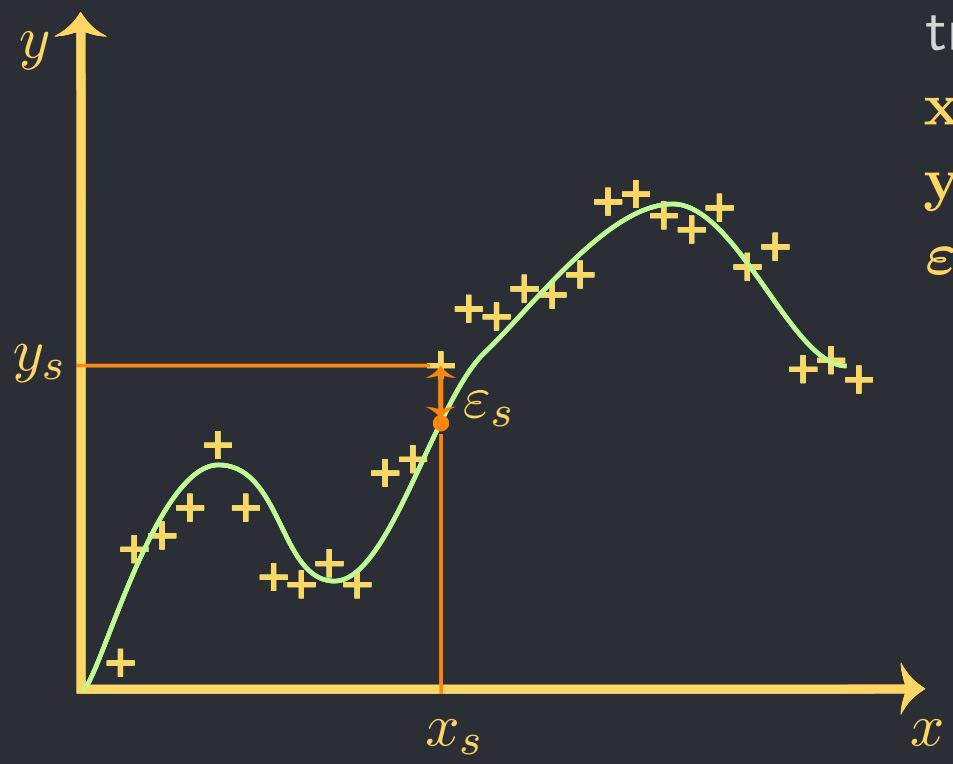
$\mathbf{y}_s = f(\mathbf{x}_s) + \boldsymbol{\varepsilon}_s$

$f(\mathbf{x}_s)$ – unobservable ground truth function

$\mathbf{x}_s$ – measurement conditions

$\mathbf{y}_s$ – measured value

Problem Example: Regression (Data Fitting)



$\mathbf{y}_s = f(\mathbf{x}_s) + \boldsymbol{\varepsilon}_s$

$f(\mathbf{x}_s)$ – unobservable ground truth function

$\mathbf{x}_s$ – measurement conditions

$\mathbf{y}_s$ – measured value

$\boldsymbol{\varepsilon}_s$ – noise

Starter Kit

Starter Kit

- Neural Network Architecture

Starter Kit

- Neural Network Architecture
- Loss Function

Starter Kit

- Neural Network Architecture
- Loss Function
- Numerical Optimization Method

Starter Kit

- Neural Network Architecture
- Loss Function
- Numerical Optimization Method
- Metrics

Neural Network

x_s

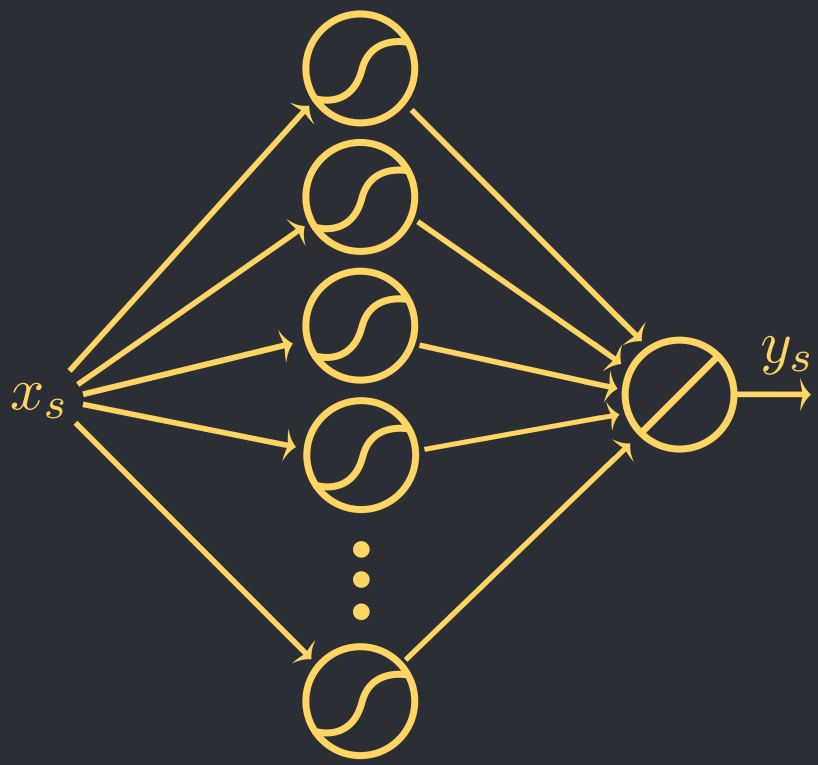
Neural Network

x_s

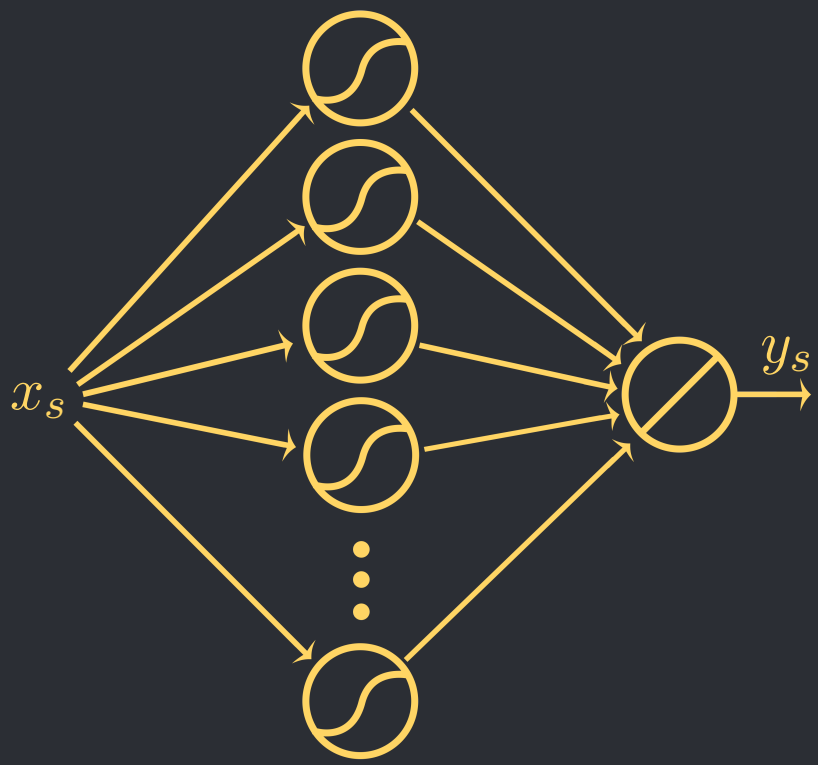
Neural Network

x_s

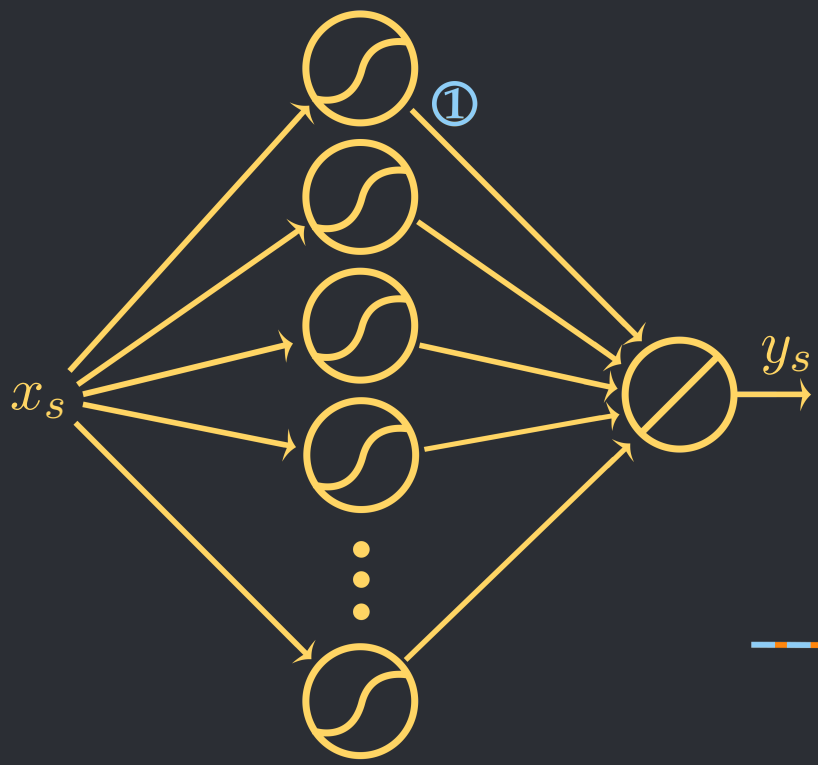
Neural Network

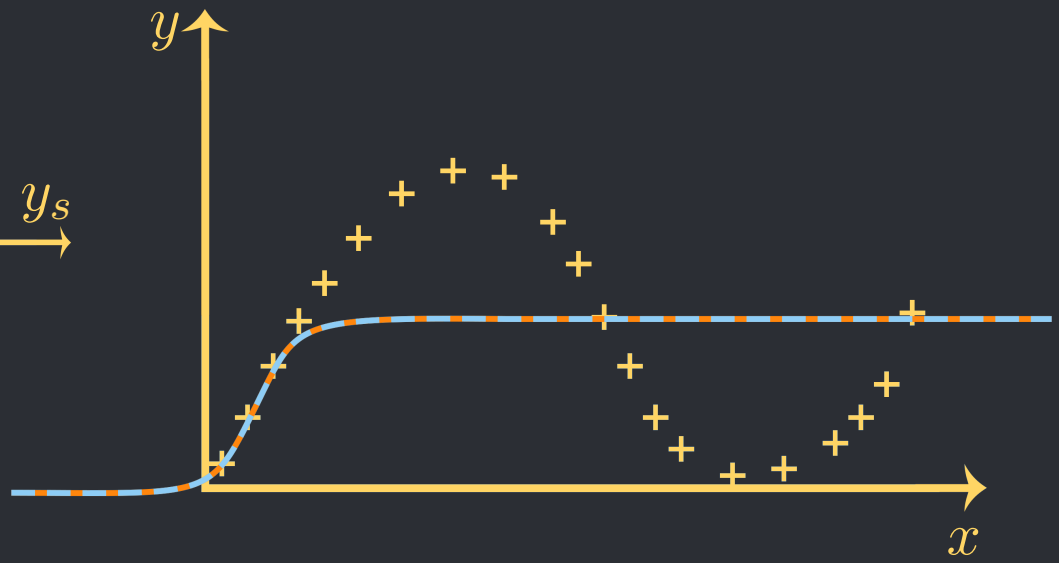


Neural Network

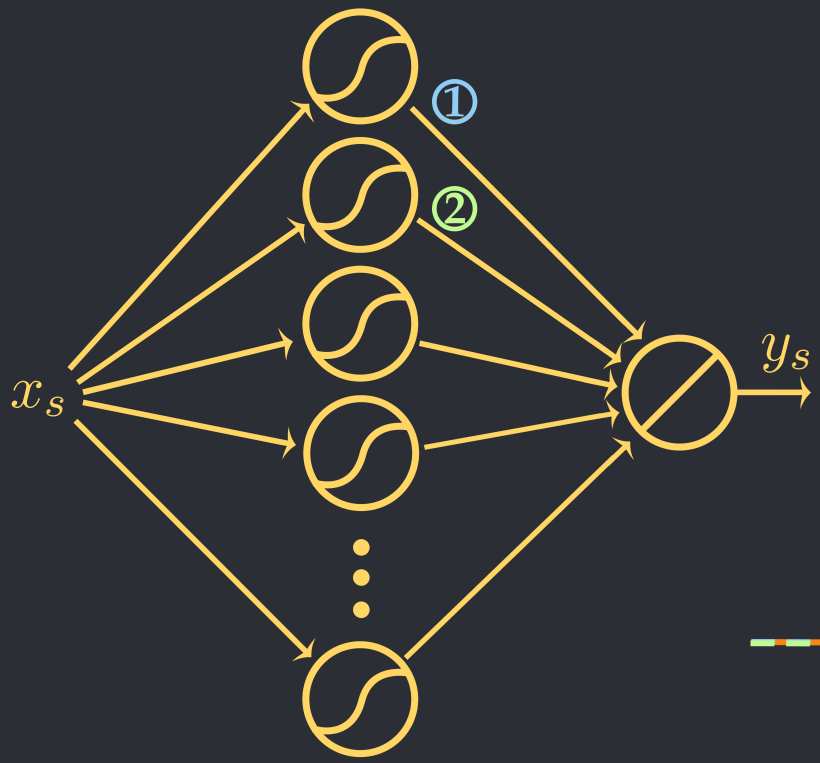


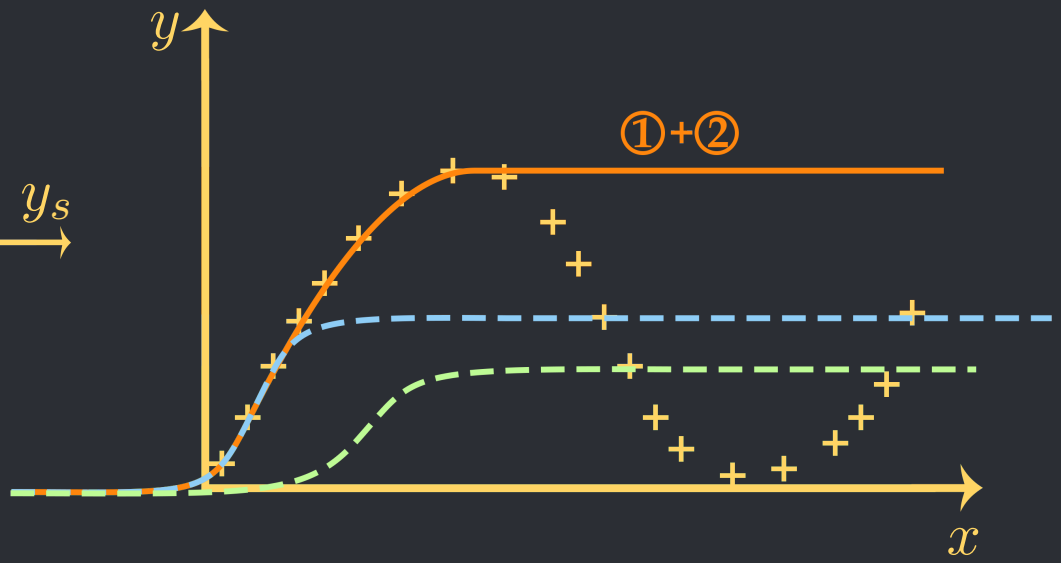
Neural Network



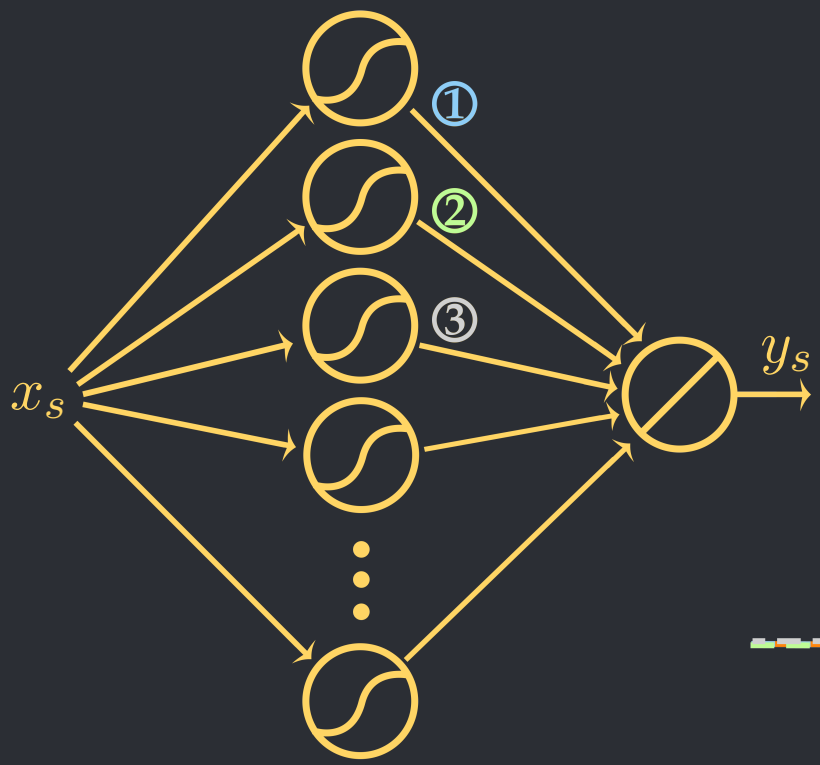


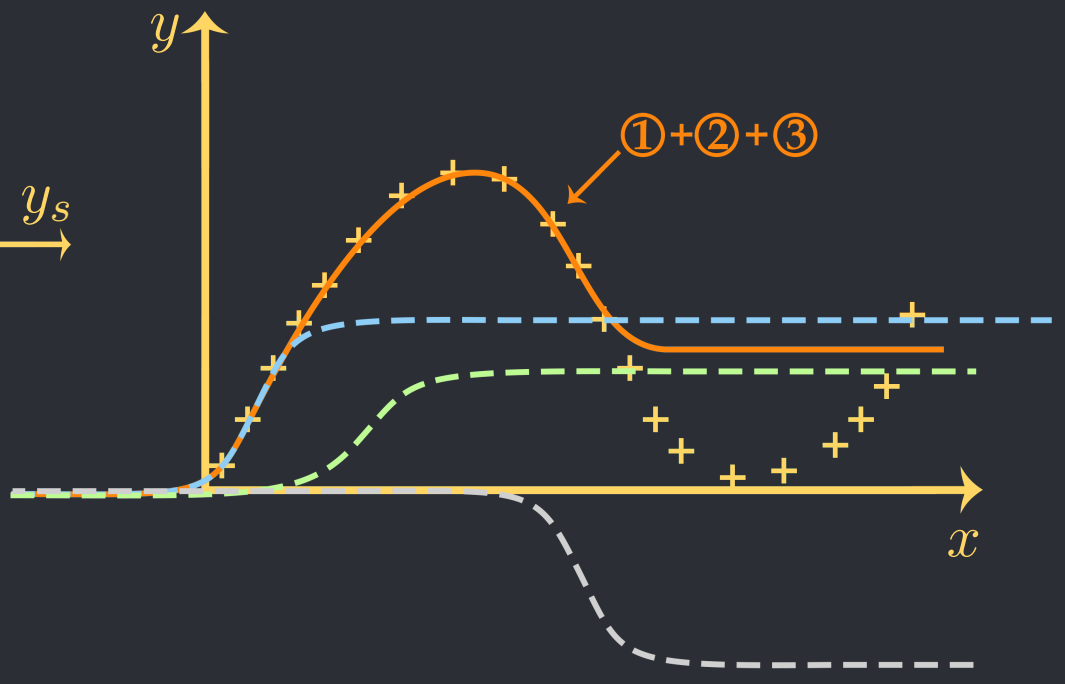
Neural Network



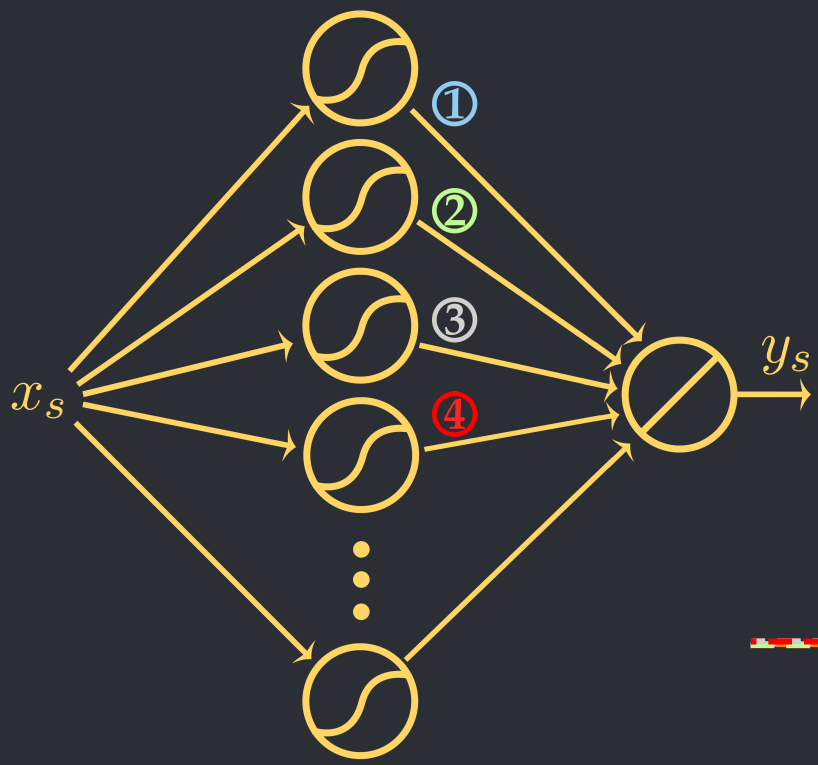


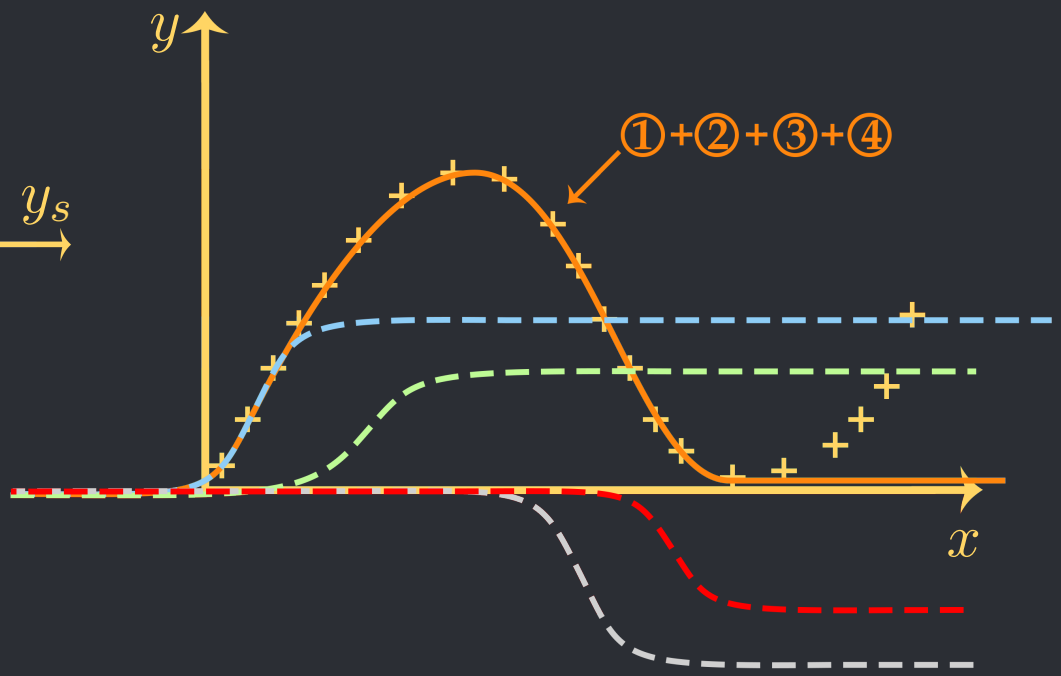
Neural Network



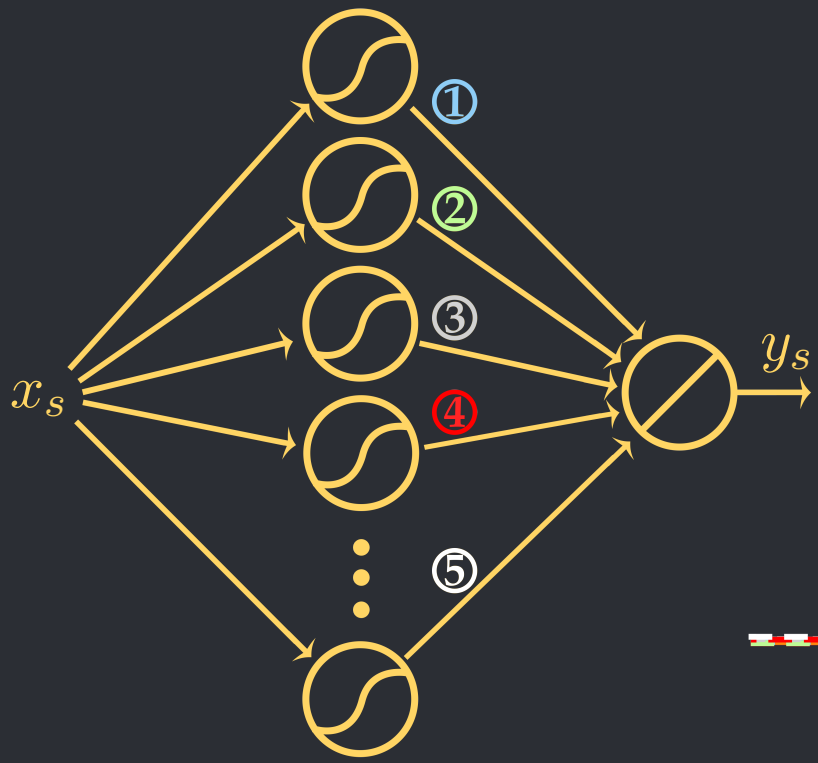


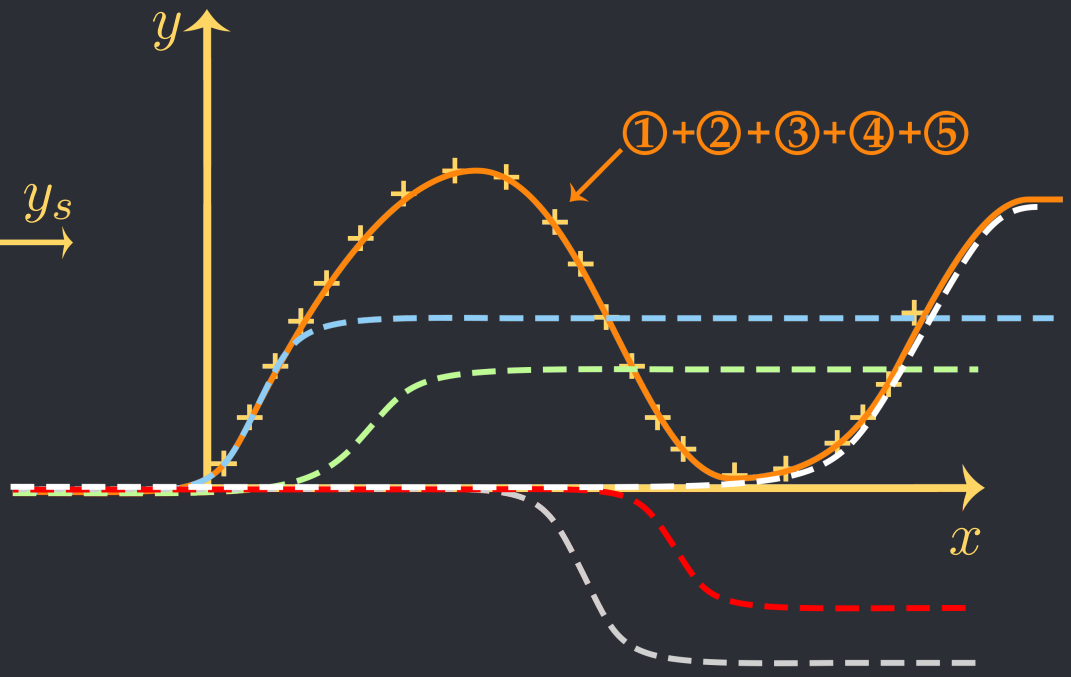
Neural Network



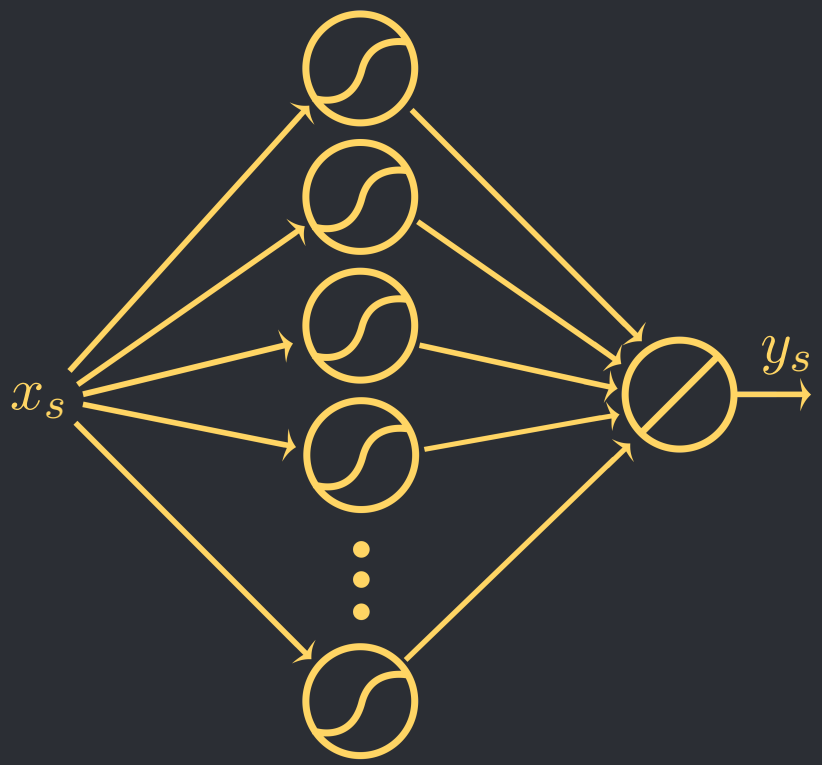


Neural Network



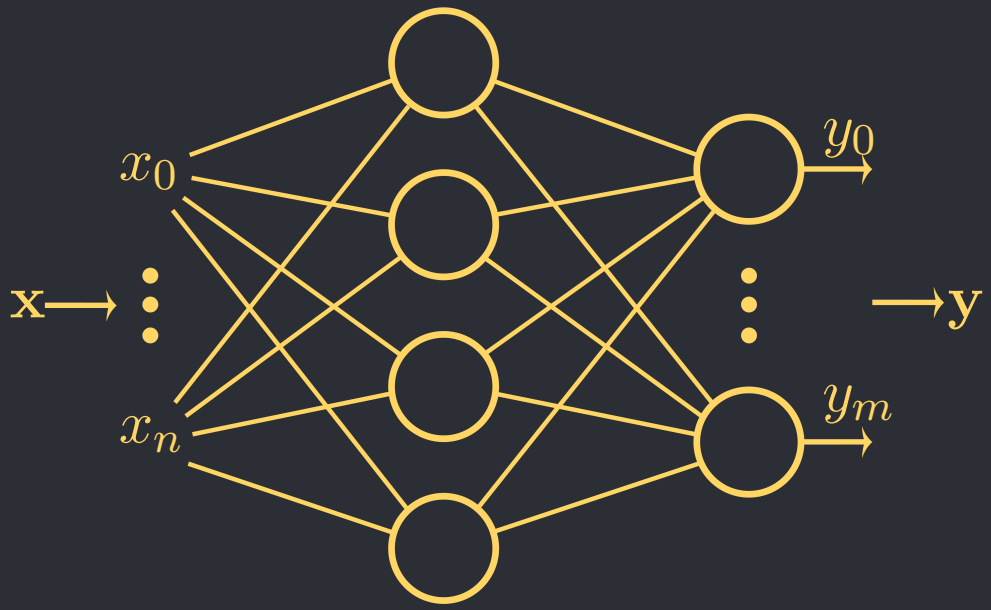


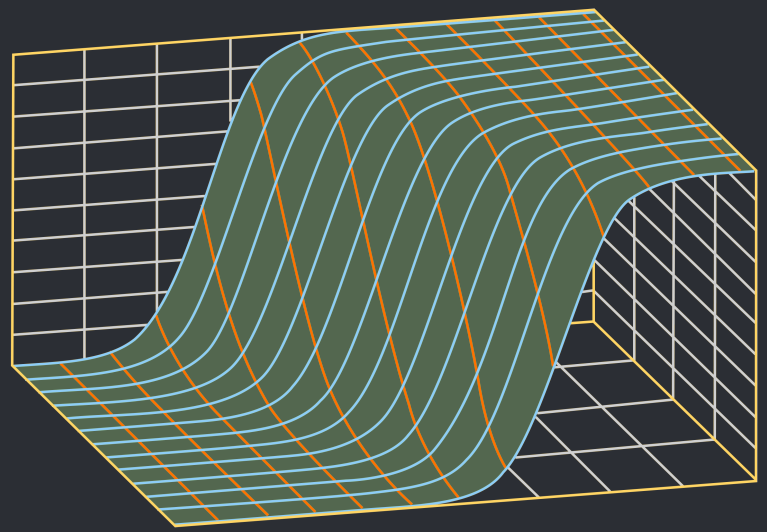
Neural Network



Using a linear combination of sigmoids it is possible to approximate with any accuracy any bounded function with limited amount of discontinuities.

Neural Network





Loss function

Consider one of the simpliest loss functions:
Mean Squared Error (MSE) for our problem.

Loss function

Consider one of the simpliest loss functions:
Mean Squared Error (MSE) for our problem.

$$MSE = \frac{1}{S} \sum_{s=1}^{S} (\widetilde{y}_s - y_s)^2$$

Loss function

Consider one of the simpliest loss functions:
Mean Squared Error (MSE) for our problem.

$$MSE = \frac{1}{S} \sum_{s=1}^{S} (\widetilde{y}_s - y_s)^2$$

Network output

Target values

Metrics

- It is possible to use any function as a metric, including the loss functions
- Examples: precision, recall, ROC AUC, top N
- Consider a metric «fraction of points with a deviation less than a specified value»

Metrics

- It is possible to use any function as a metric, including the loss functions
- Examples: precision, recall, ROC AUC, top N
- Consider a metric «fraction of points with a deviation less than a specified value»

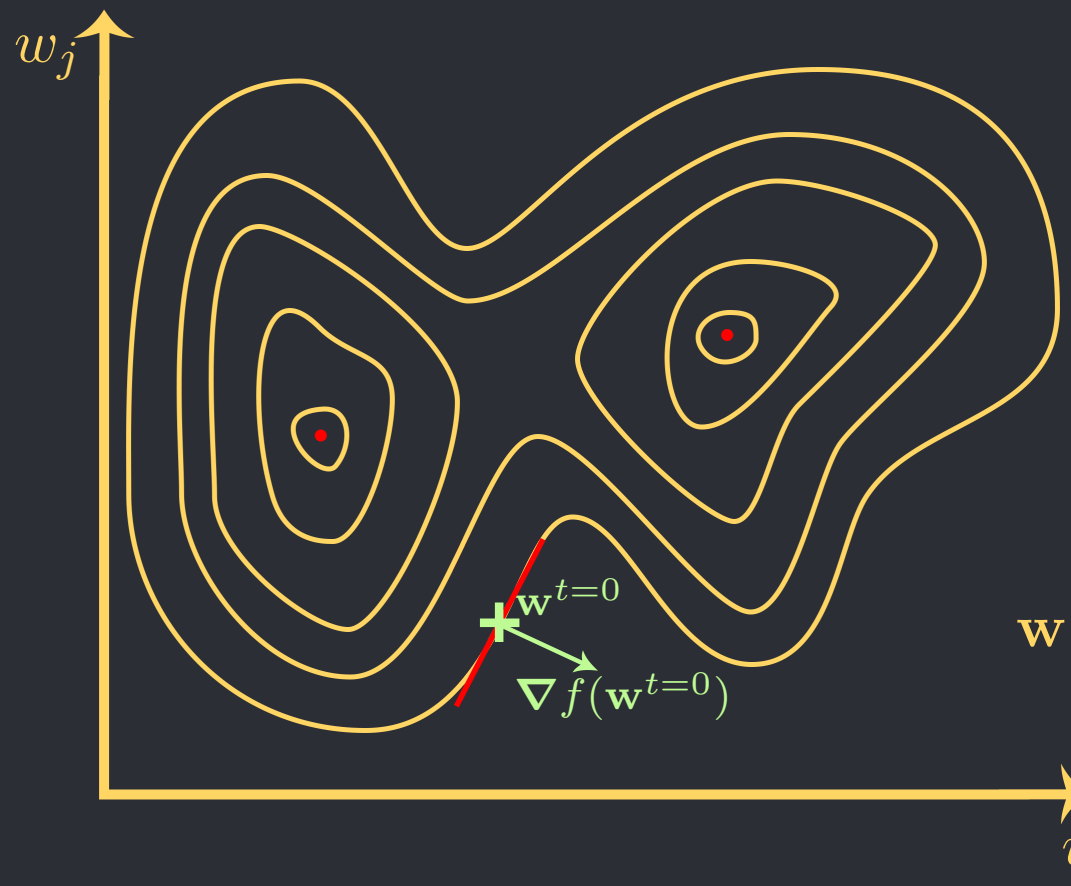
$$Acc_\varepsilon = \frac{1}{S}\sum_{s=1}^{S}[|y_s - \widetilde{y}_s| < \varepsilon]$$

Tuning the Parameters: Gradient Descent

All the weights of the network - $\mathbf{w}$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_i \\ w_{P-1} \end{bmatrix}$$

Tuning the Parameters: Gradient Descent

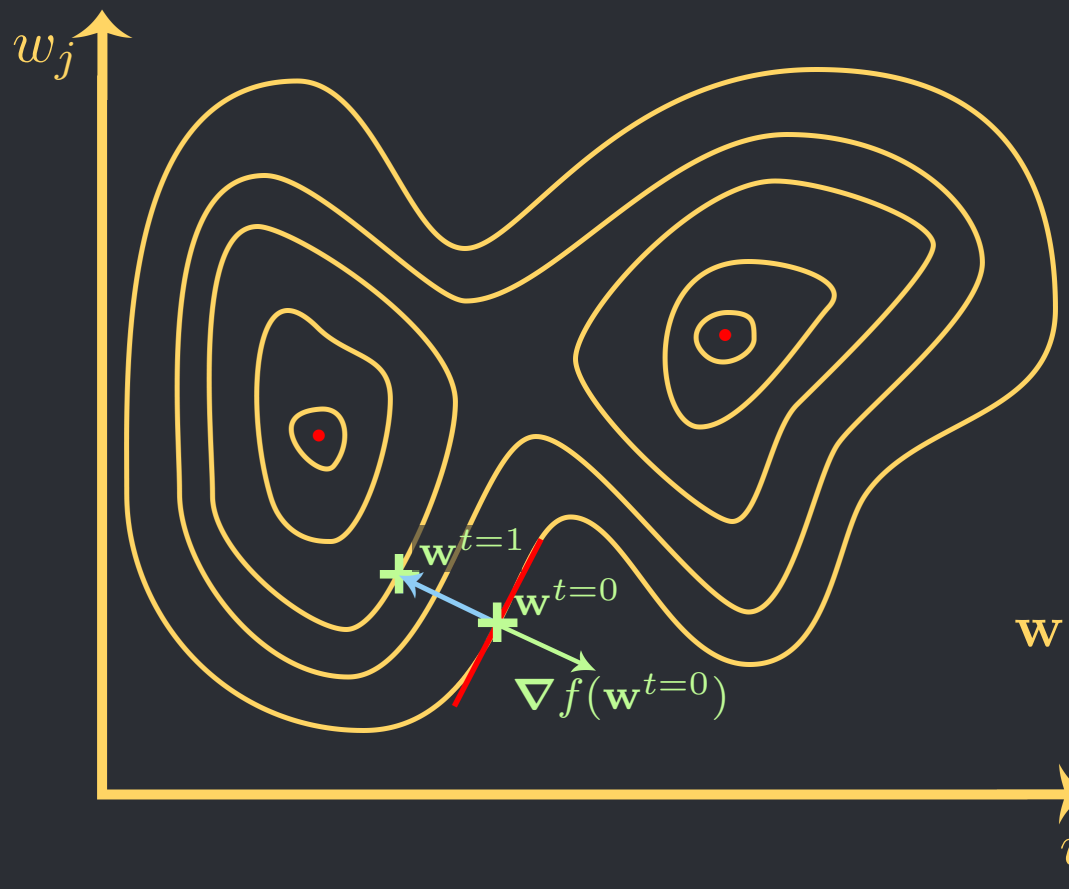


$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{P-1} \end{bmatrix} \qquad \nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_{P-1}} \end{bmatrix}$$

Tuning the Parameters: Gradient Descent

$$\mathbf{w}^{t=0}$$

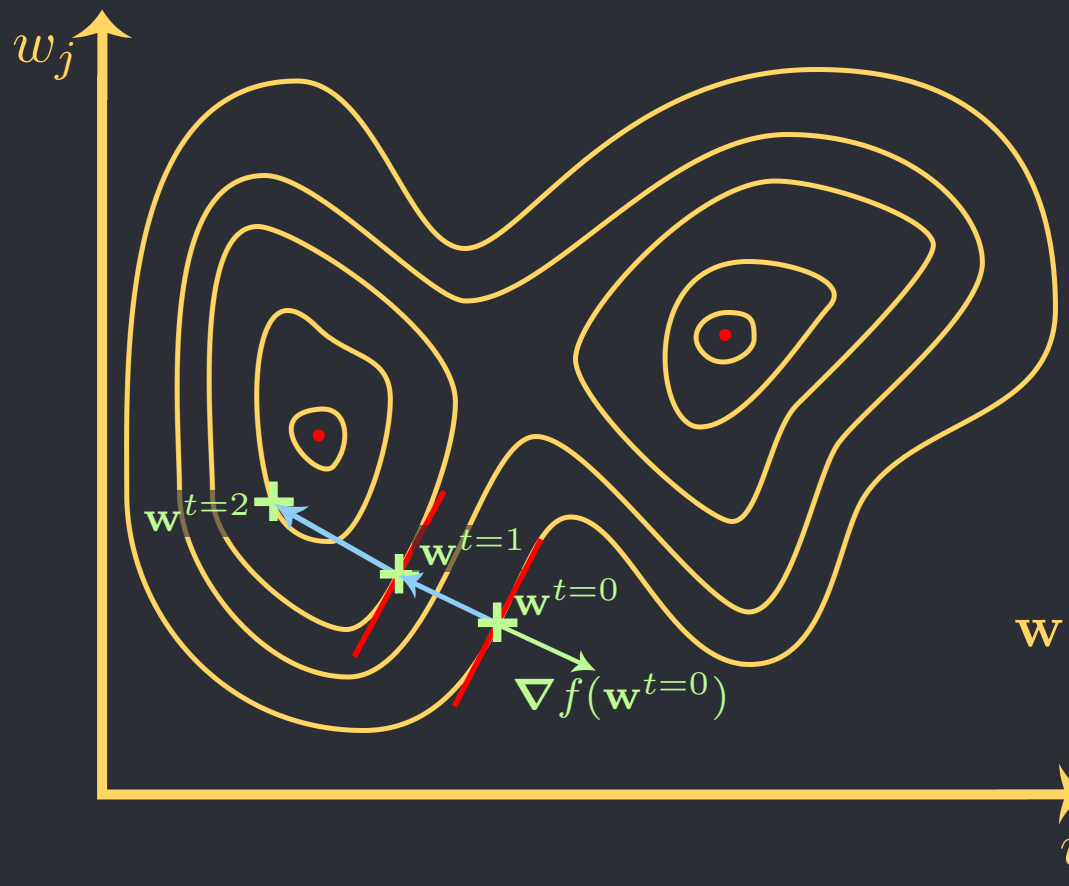
$$\mathbf{w}^{t=1} = \mathbf{w}^{t=0} - \alpha \boldsymbol{\nabla} f(\mathbf{w}^{t=0})$$



$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{P-1} \end{bmatrix} \qquad \boldsymbol{\nabla} f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_{P-1}} \end{bmatrix}$$

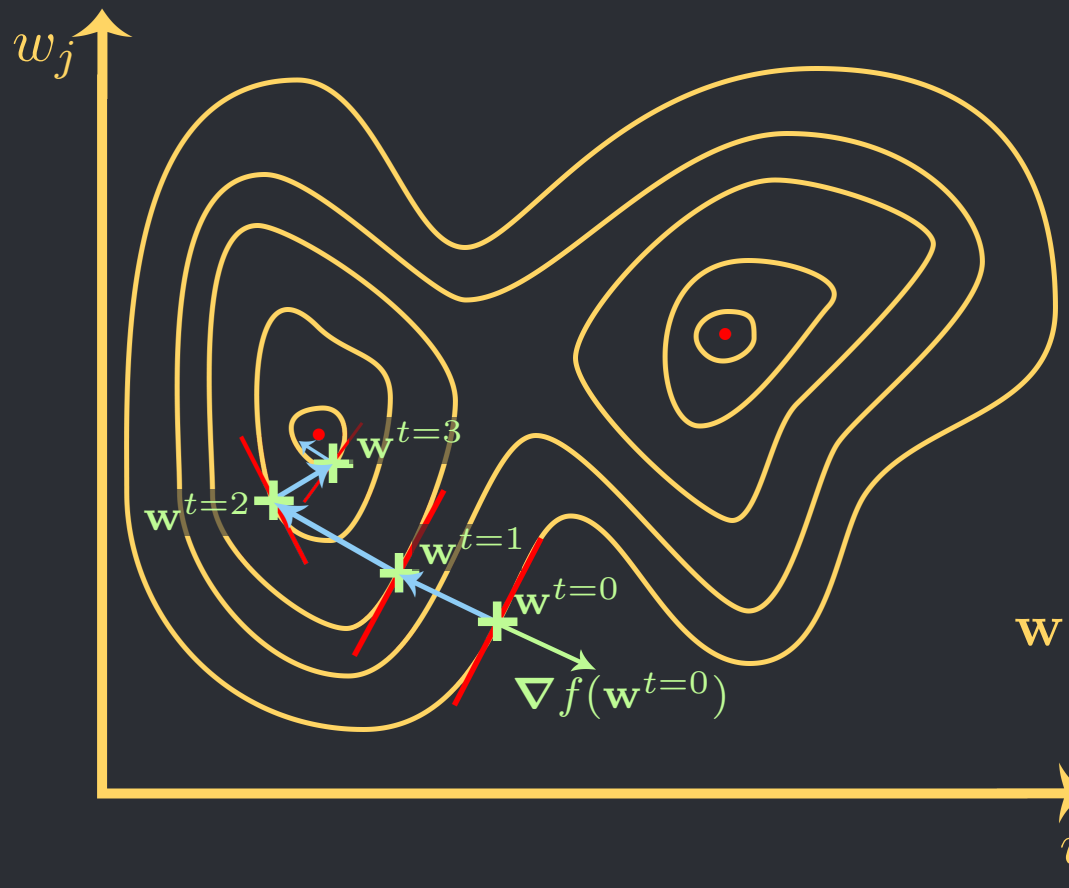
Tuning the Parameters: Gradient Descent

$$\mathbf{w}^{t=0}$$
$$\mathbf{w}^{t=1} = \mathbf{w}^{t=0} - \alpha \boldsymbol{\nabla} f(\mathbf{w}^{t=0})$$
$$\mathbf{w}^{t=2} = \mathbf{w}^{t=1} - \alpha \boldsymbol{\nabla} f(\mathbf{w}^{t=1})$$



$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{P-1} \end{bmatrix} \qquad \boldsymbol{\nabla} f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_{P-1}} \end{bmatrix}$$

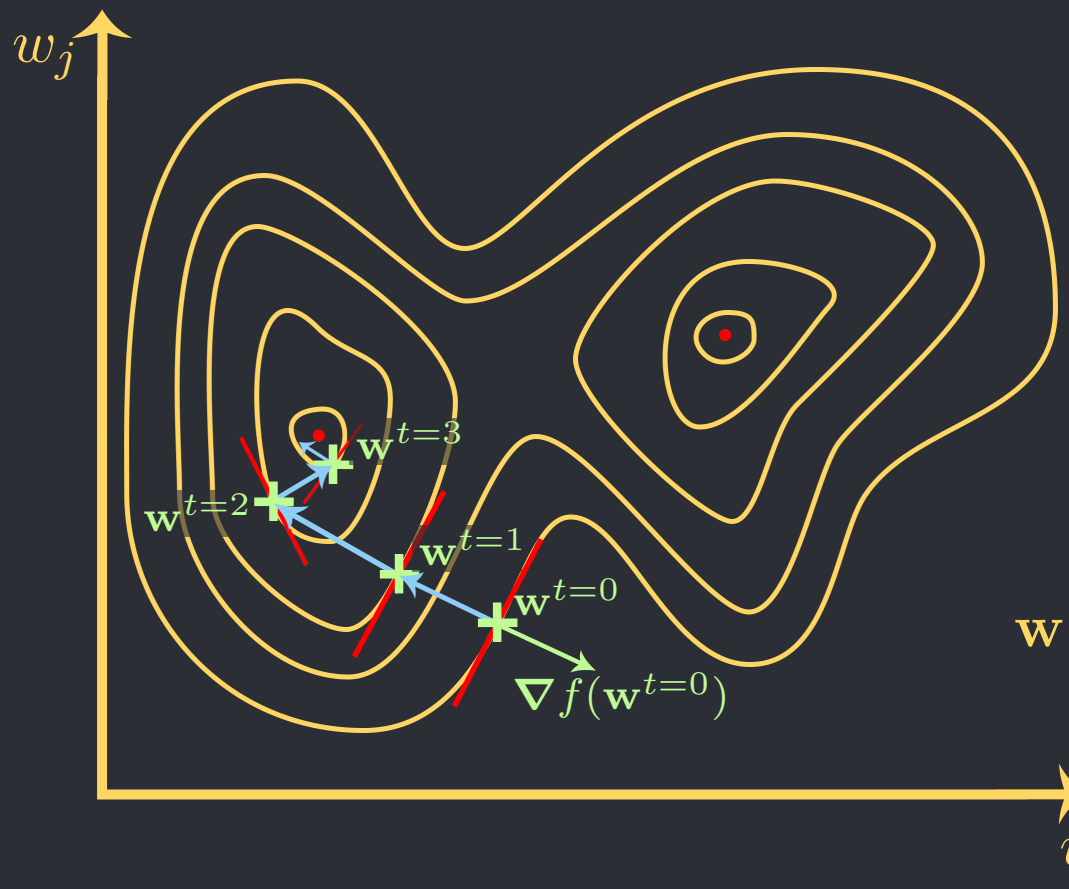
Tuning the Parameters: Gradient Descent



$$\mathbf{w}^{t=0}$$
$$\mathbf{w}^{t=1} = \mathbf{w}^{t=0} - \alpha \nabla f(\mathbf{w}^{t=0})$$
$$\mathbf{w}^{t=2} = \mathbf{w}^{t=1} - \alpha \nabla f(\mathbf{w}^{t=1})$$
$$\mathbf{w}^{t=3} = \mathbf{w}^{t=2} - \alpha \nabla f(\mathbf{w}^{t=2})$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{P-1} \end{bmatrix} \quad \nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_{P-1}} \end{bmatrix}$$

Tuning the Parameters: Gradient Descent



$$\mathbf{w}^{t=0}$$
$$\mathbf{w}^{t=1} = \mathbf{w}^{t=0} - \alpha \nabla f(\mathbf{w}^{t=0})$$
$$\mathbf{w}^{t=2} = \mathbf{w}^{t=1} - \alpha \nabla f(\mathbf{w}^{t=1})$$
$$\mathbf{w}^{t=3} = \mathbf{w}^{t=2} - \alpha \nabla f(\mathbf{w}^{t=2})$$
$$\dots$$
$$\mathbf{w}^{T} = \mathbf{w}^{T-1} - \alpha \nabla f(\mathbf{w}^{T-1})$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{P-1} \end{bmatrix} \qquad \nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_{P-1}} \end{bmatrix}$$

Tuning the Network: Gradient of a Loss Function

To be able to perform a Gradient Descent, we need a differentiable loss function

$$MSE = \frac{1}{S}\sum_{s=1}^{S}(y_s - \widetilde{y}_s)^2$$

Tuning the Network: Gradient of a Loss Function

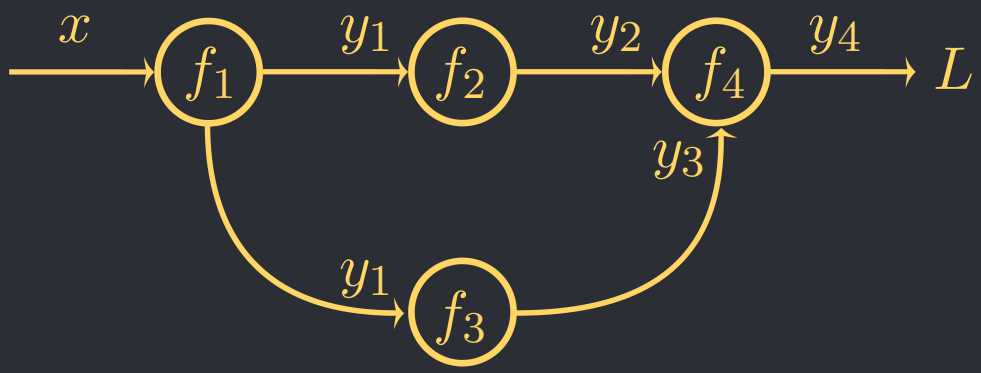
To be able to perform a Gradient Descent, we need a differentiable loss function

$$MSE = \frac{1}{S}\sum_{s=1}^{S}(y_s - \widetilde{y}_s)^2$$

$$\frac{\partial}{\partial \widetilde{y}_s} MSE = -\frac{2}{S}(y_s - \widetilde{y}_s)$$

Computing the Gradients: Chain Rule

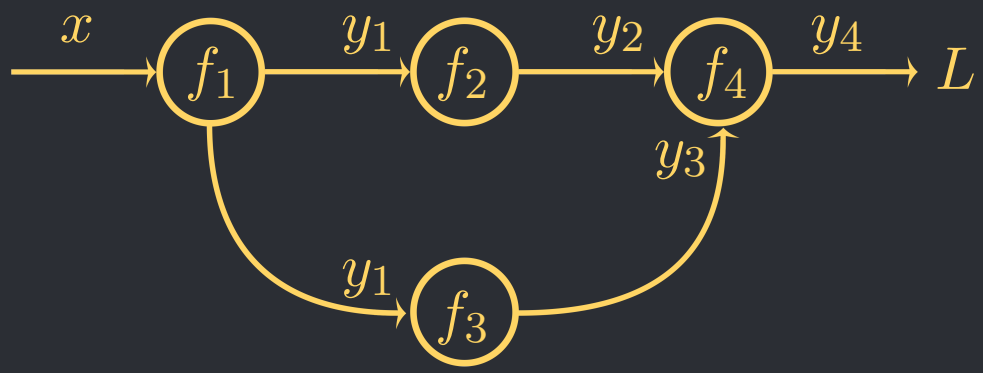
We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



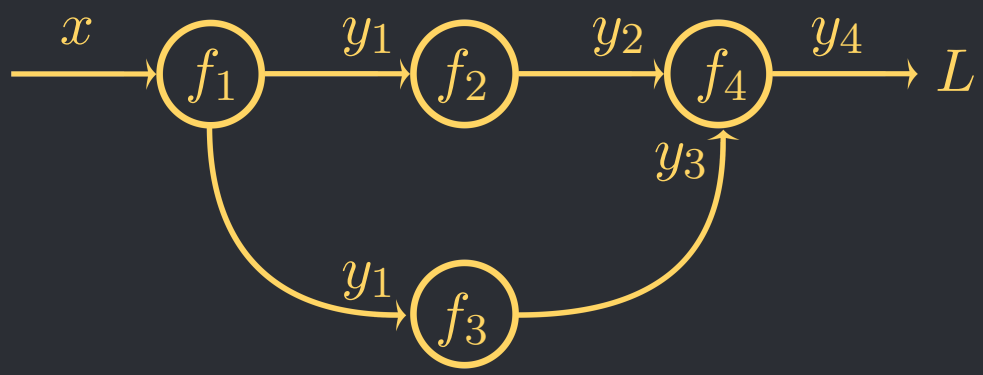
$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

Правило дифференцирования:

$$\frac{\partial f\big(g(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

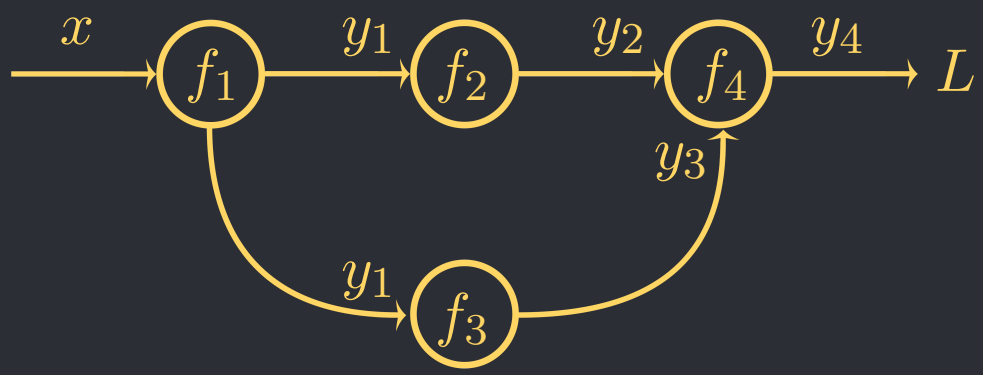
Правило дифференцирования:

$$\frac{\partial f\big(g(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\frac{\partial L}{\partial w_2} =$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

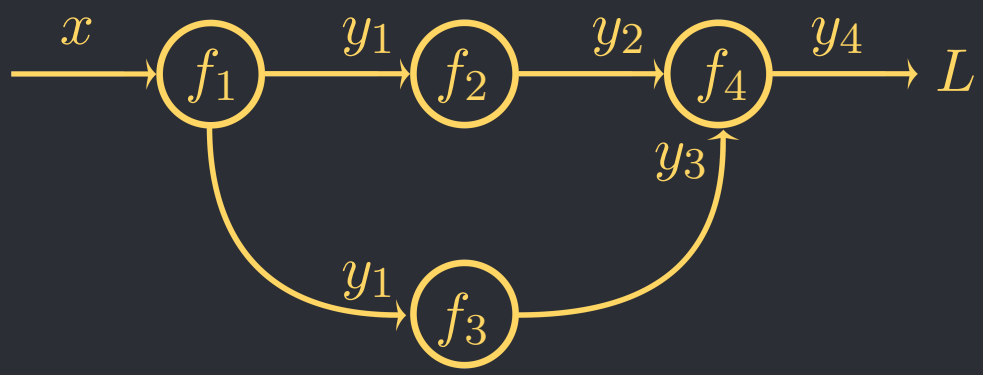
Правило дифференцирования:

$$\frac{\partial f\big(g(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_4}$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

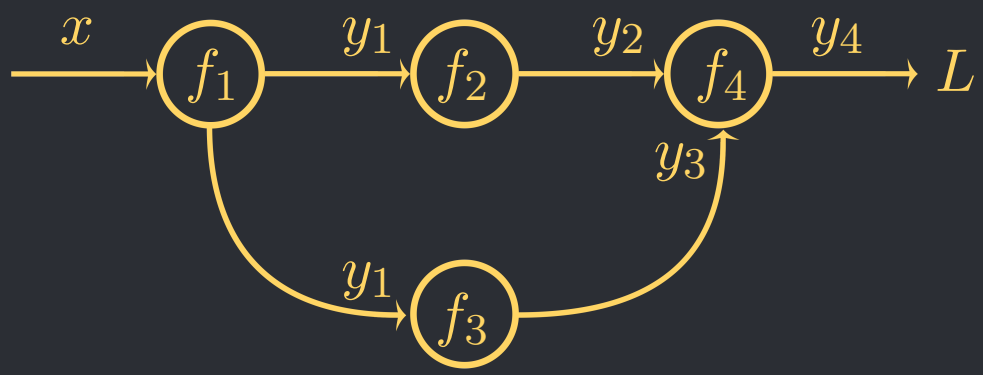
Правило дифференцирования:

$$\frac{\partial f\big(g(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_4} \cdot \frac{\partial f_4}{\partial y_2}$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

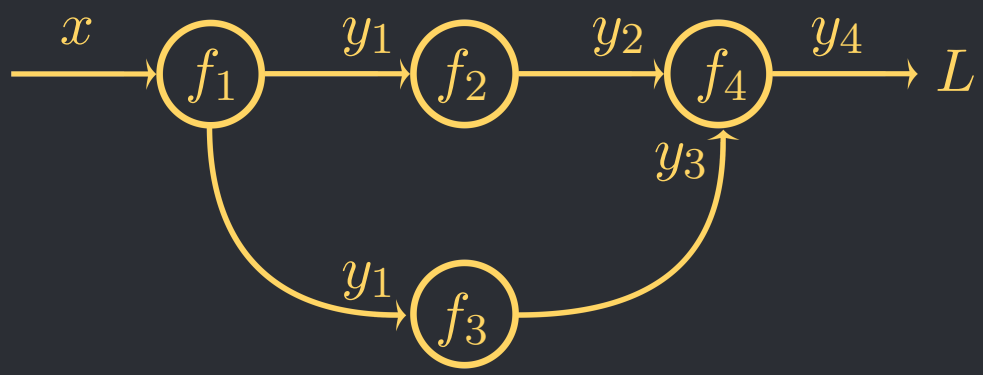
Правило дифференцирования:

$$\frac{\partial f\big(g(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_4} \cdot \frac{\partial f_4}{\partial y_2} \cdot \frac{\partial f_2}{\partial w_2}$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

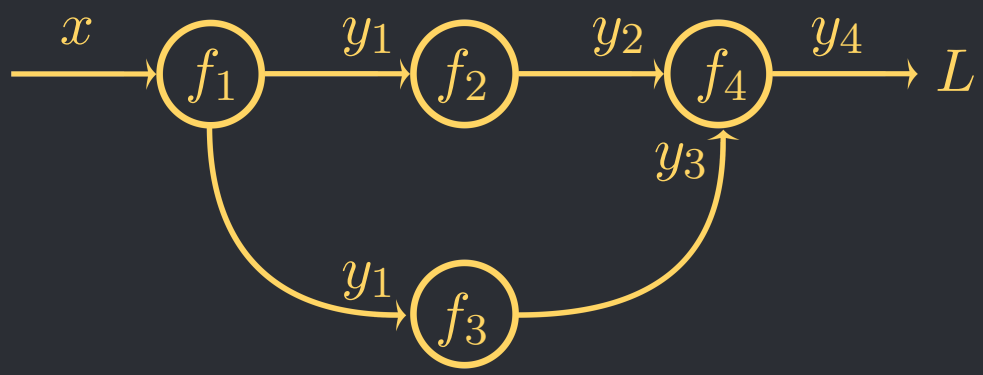
Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} + \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial x}$$

$$\frac{\partial L}{\partial w_1} =$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

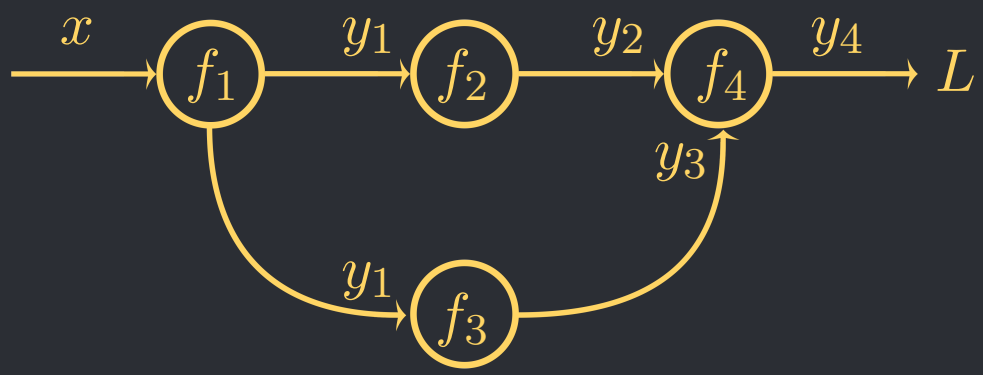
Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} + \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial x}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_4}$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

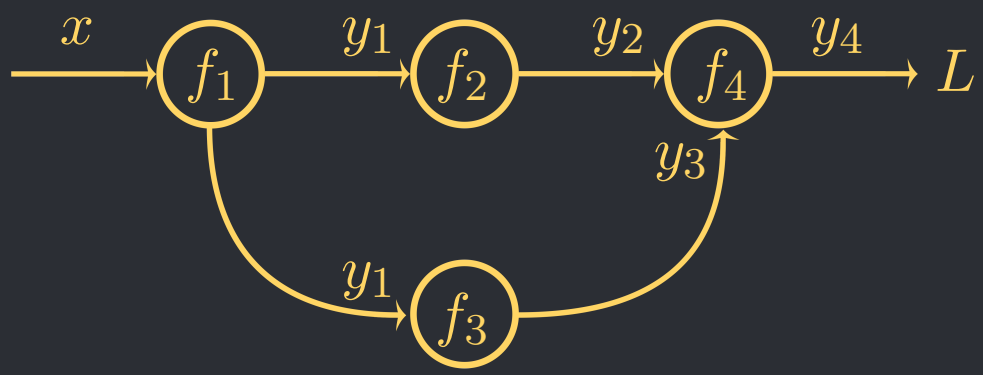
Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} + \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial x}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_4} \left[\frac{\partial f_4}{\partial y_2} \right.$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

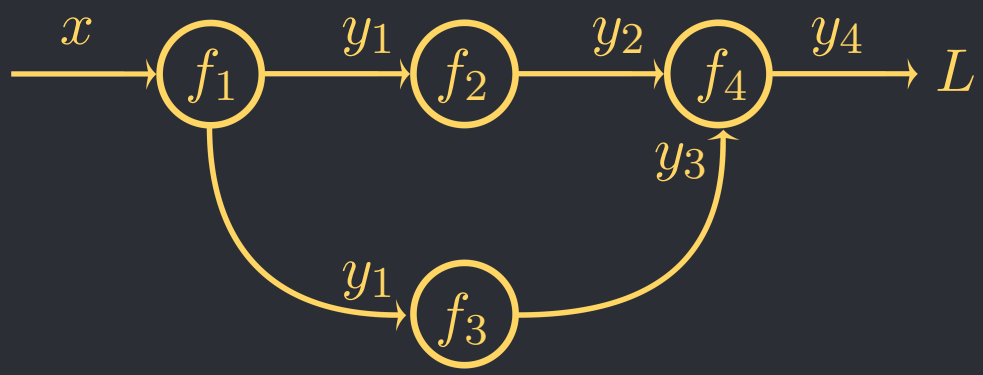
Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} + \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial x}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_4} \left[\frac{\partial f_4}{\partial y_2} \cdot \frac{\partial f_2}{\partial y_1}$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

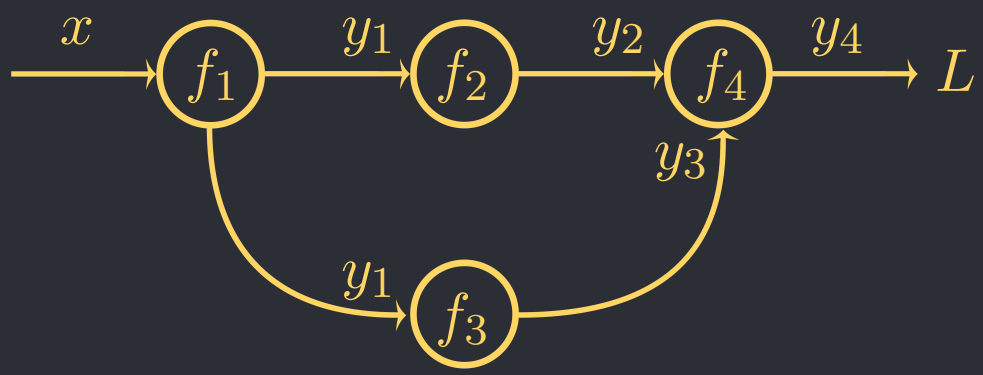
Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g}\cdot\frac{\partial g}{\partial x} + \frac{\partial f}{\partial e}\cdot\frac{\partial e}{\partial x}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_4}\left[\frac{\partial f_4}{\partial y_2}\cdot\frac{\partial f_2}{\partial y_1} + \frac{\partial f_4}{\partial y_3}\right.$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

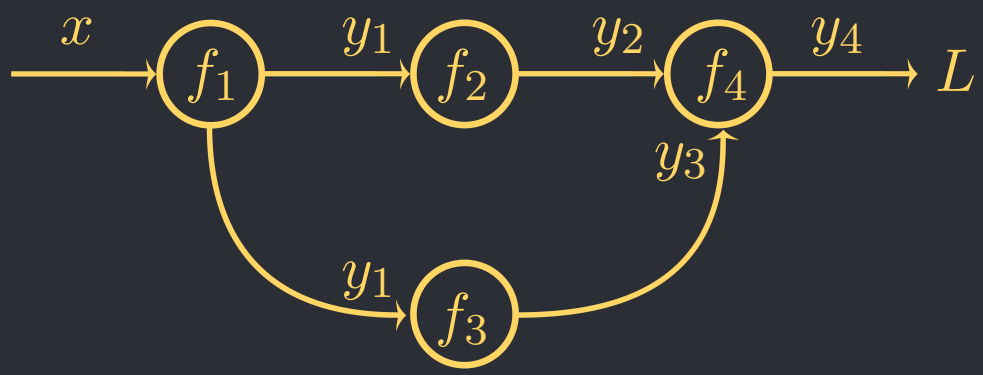
Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g}\cdot\frac{\partial g}{\partial x} + \frac{\partial f}{\partial e}\cdot\frac{\partial e}{\partial x}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_4}\left[\frac{\partial f_4}{\partial y_2}\cdot\frac{\partial f_2}{\partial y_1} + \frac{\partial f_4}{\partial y_3}\cdot\frac{\partial f_3}{\partial y_1}\right]$$

Computing the Gradients: Chain Rule

We know a derivative of the Loss Function. How to compute the derivatives of the Loss Function over the all the parameters of the network?



$$L\Big(f_4\Big(f_2\big(f_1(x)\big),\ f_3\big(f_1(x)\big)\Big)\Big)$$

Правило дифференцирования:

$$\frac{\partial f\big(g(x),\, e(x)\big)}{\partial x} = \frac{\partial f}{\partial g}\cdot\frac{\partial g}{\partial x} + \frac{\partial f}{\partial e}\cdot\frac{\partial e}{\partial x}$$

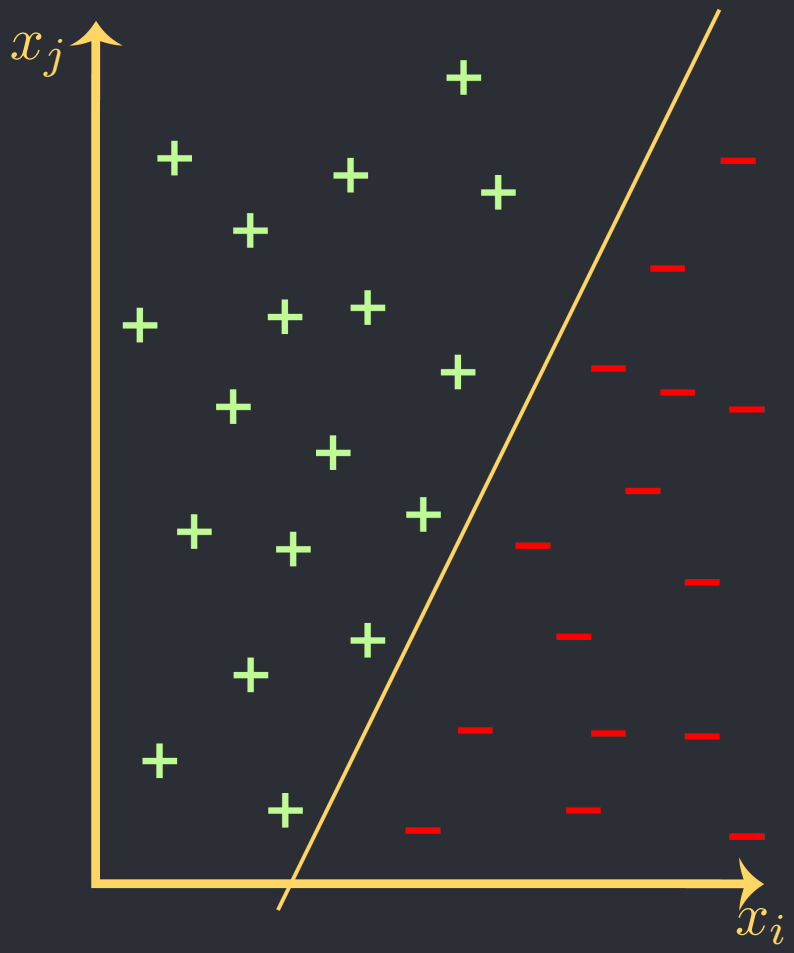
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_4}\left[\frac{\partial f_4}{\partial y_2}\cdot\frac{\partial f_2}{\partial y_1} + \frac{\partial f_4}{\partial y_3}\cdot\frac{\partial f_3}{\partial y_1}\right]\frac{\partial f_1}{\partial x}$$

Example: Logistic Regression

Linear classification algorithm:

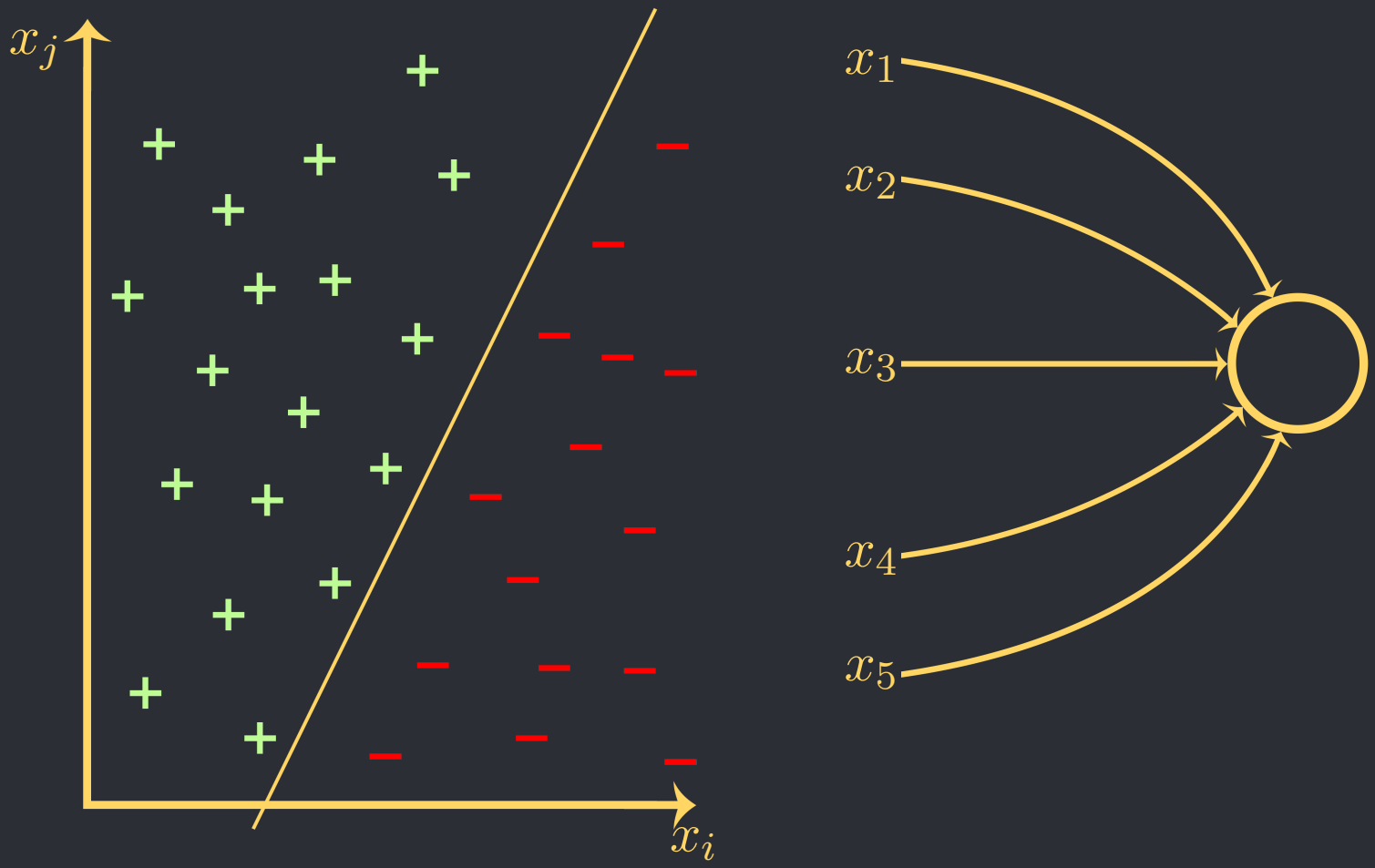
Example: Logistic Regression

Linear classification algorithm:



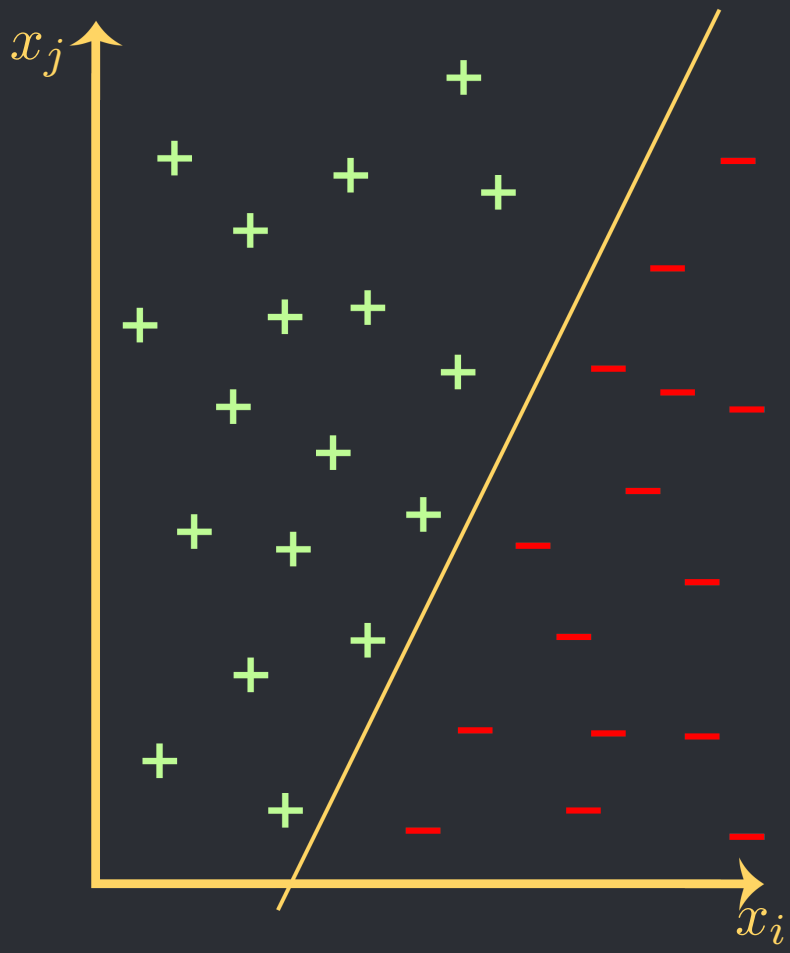
Example: Logistic Regression

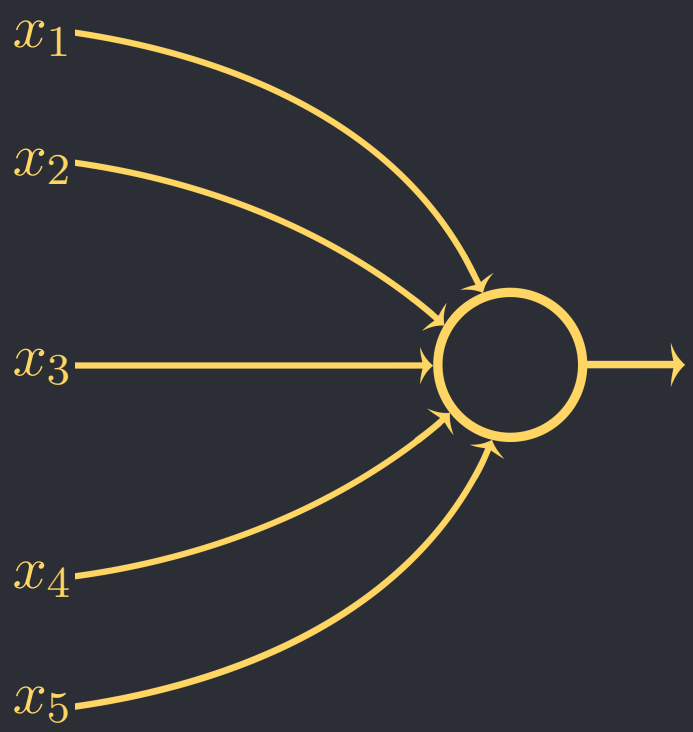
Linear classification algorithm:



Example: Logistic Regression

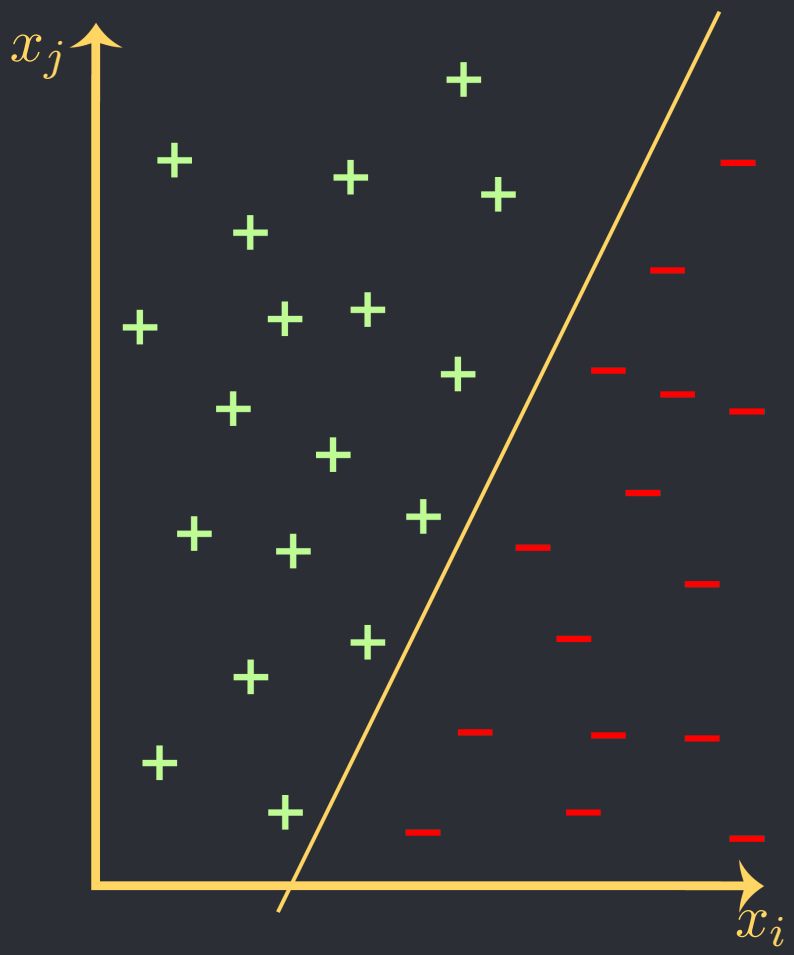
Linear classification algorithm:



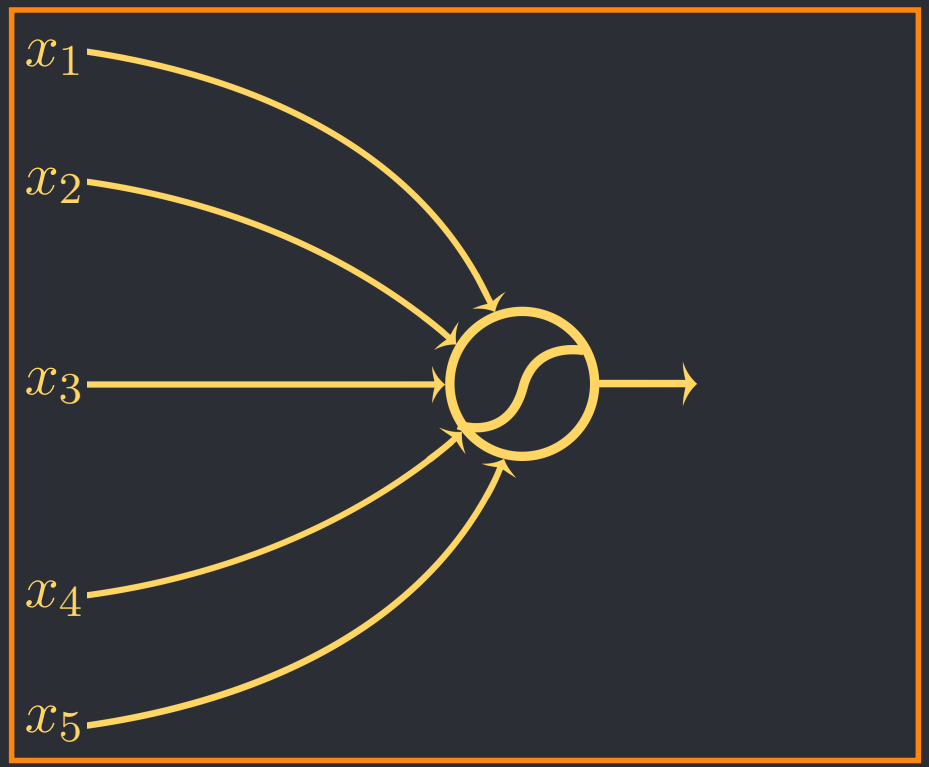


Example: Logistic Regression

Linear classification algorithm:

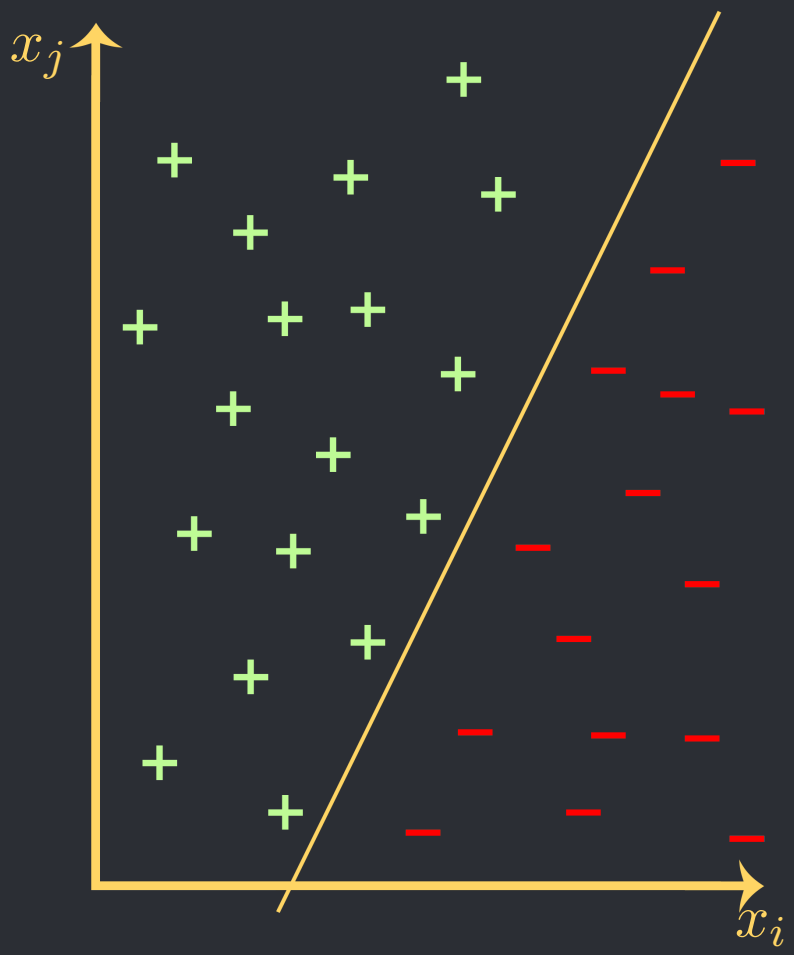


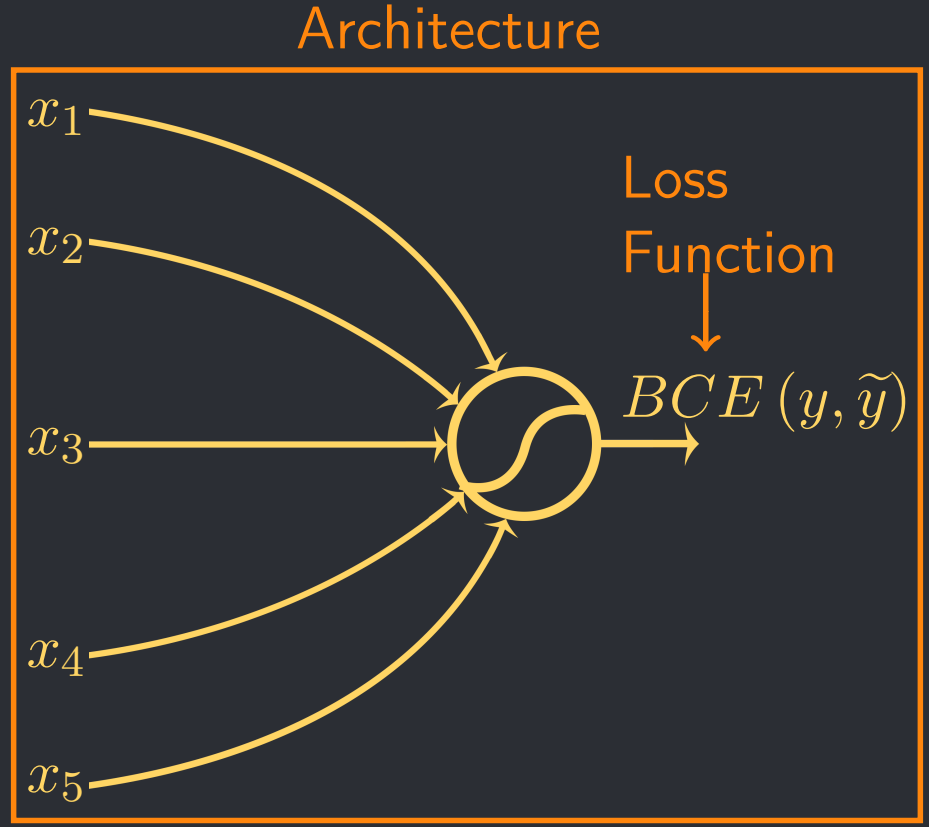
Architecture



Example: Logistic Regression

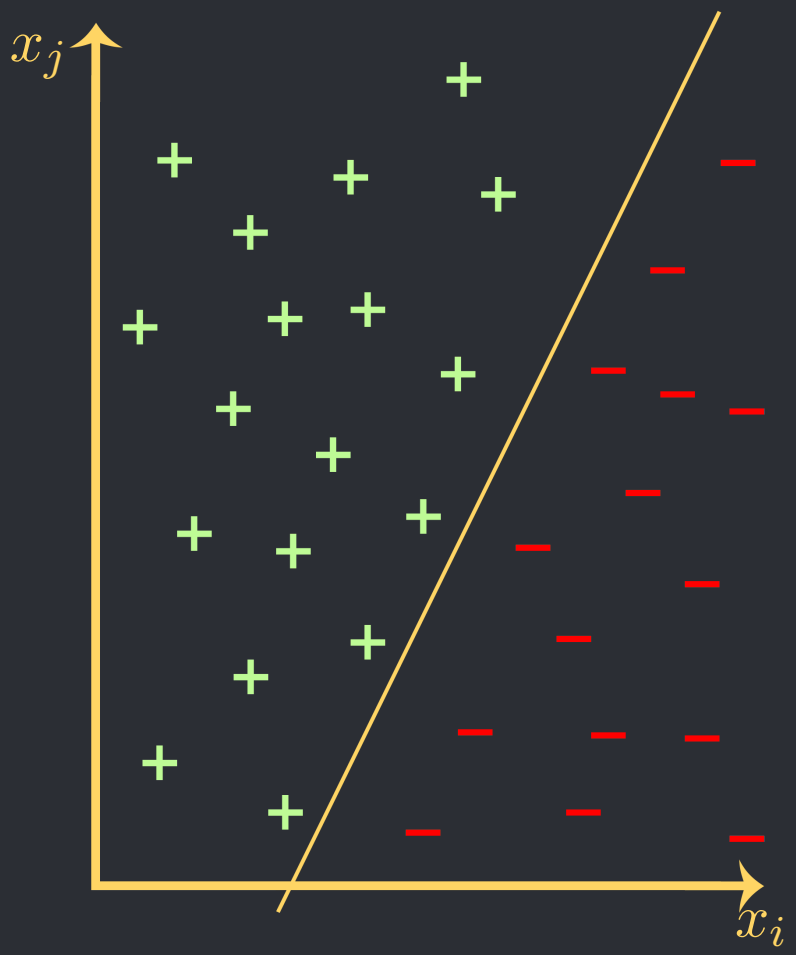
Linear classification algorithm:

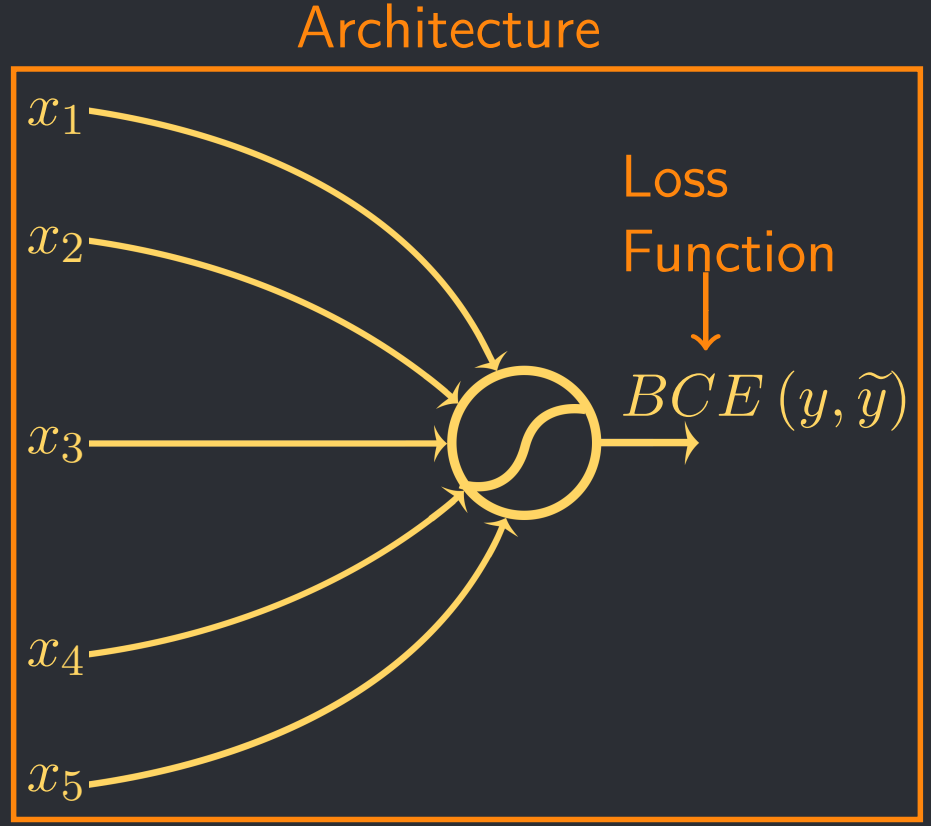




Example: Logistic Regression

Linear classification algorithm:





Optimization – stochastic gradient descent

Summary

- Basic problem example for a neural network

Summary

- Basic problem example for a neural network
- Basic neural network

Summary

- Basic problem example for a neural network
- Basic neural network
- Dataset

Summary

- Basic problem example for a neural network
- Basic neural network
- Dataset
- Loss Function

Summary

- Basic problem example for a neural network
- Basic neural network
- Dataset
- Loss Function
- Metrics

Summary

- Basic problem example for a neural network
- Basic neural network
- Dataset
- Loss Function
- Metrics
- Optimization of Neural Network (Learning procedure)

Summary

- Basic problem example for a neural network
- Basic neural network
- Dataset
- Loss Function
- Metrics
- Optimization of Neural Network (Learning procedure)
- Derivative of a Loss Function over the network parameters

Summary

- Basic problem example for a neural network
- Basic neural network
- Dataset
- Loss Function
- Metrics
- Optimization of Neural Network (Learning procedure)
- Derivative of a Loss Function over the network parameters
- Логистическая регрессия – тоже нейронная сеть