NEURAL NETWORKS

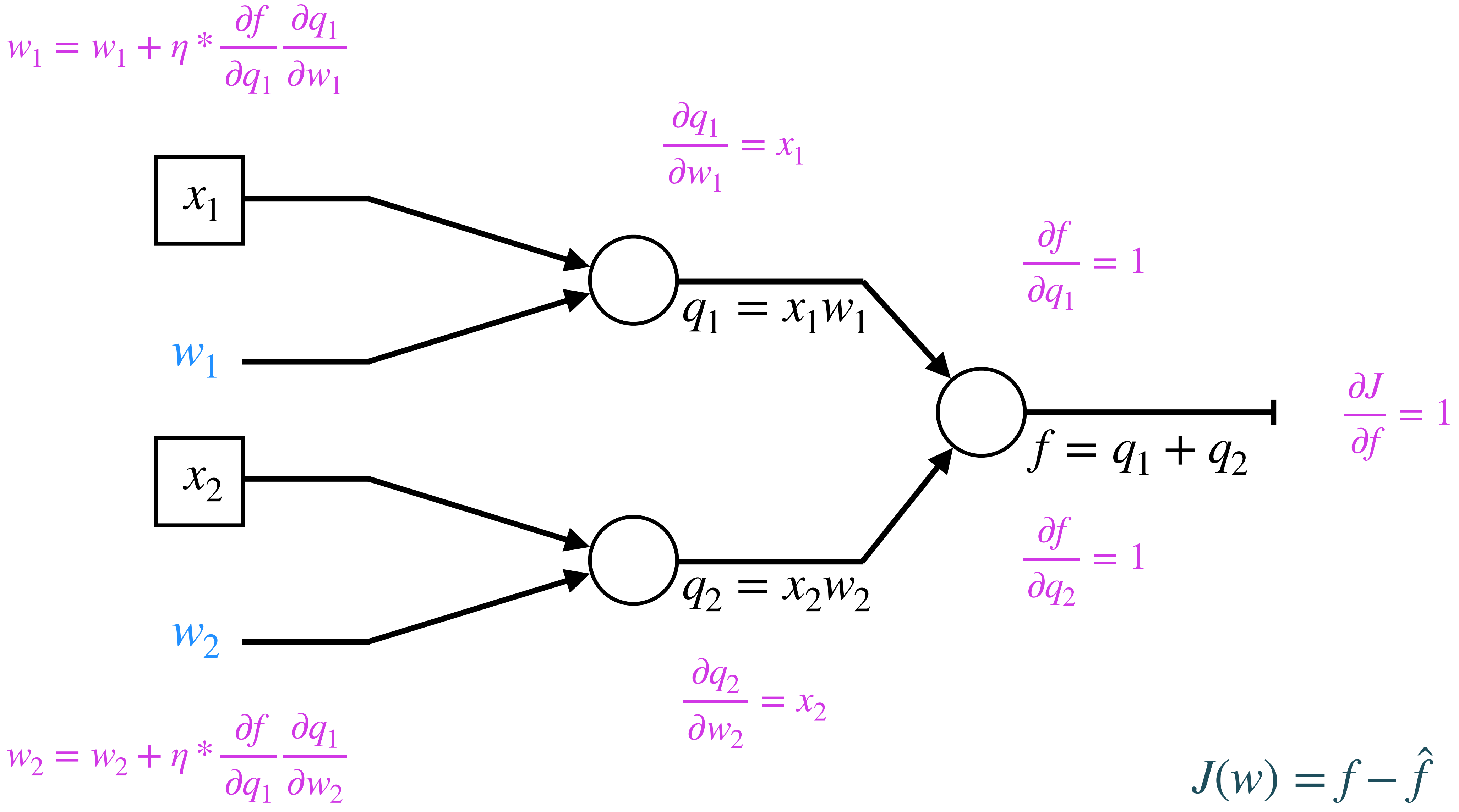
AND DEEP LEARNING

MICHELLE KUCHERA
DAVIDSON COLLEGE

CPS-FR
MIT
25 AUGUST 2023

MICHELLE KUCHERA

B.S., M.S. PHYSICS
M.S., PH.D. COMPUTATIONAL SCIENCE



ALPhA
DAVIDSON COLLEGE

CERN

CMS
Compact Muon Solenoid

Jefferson Lab

FRIB

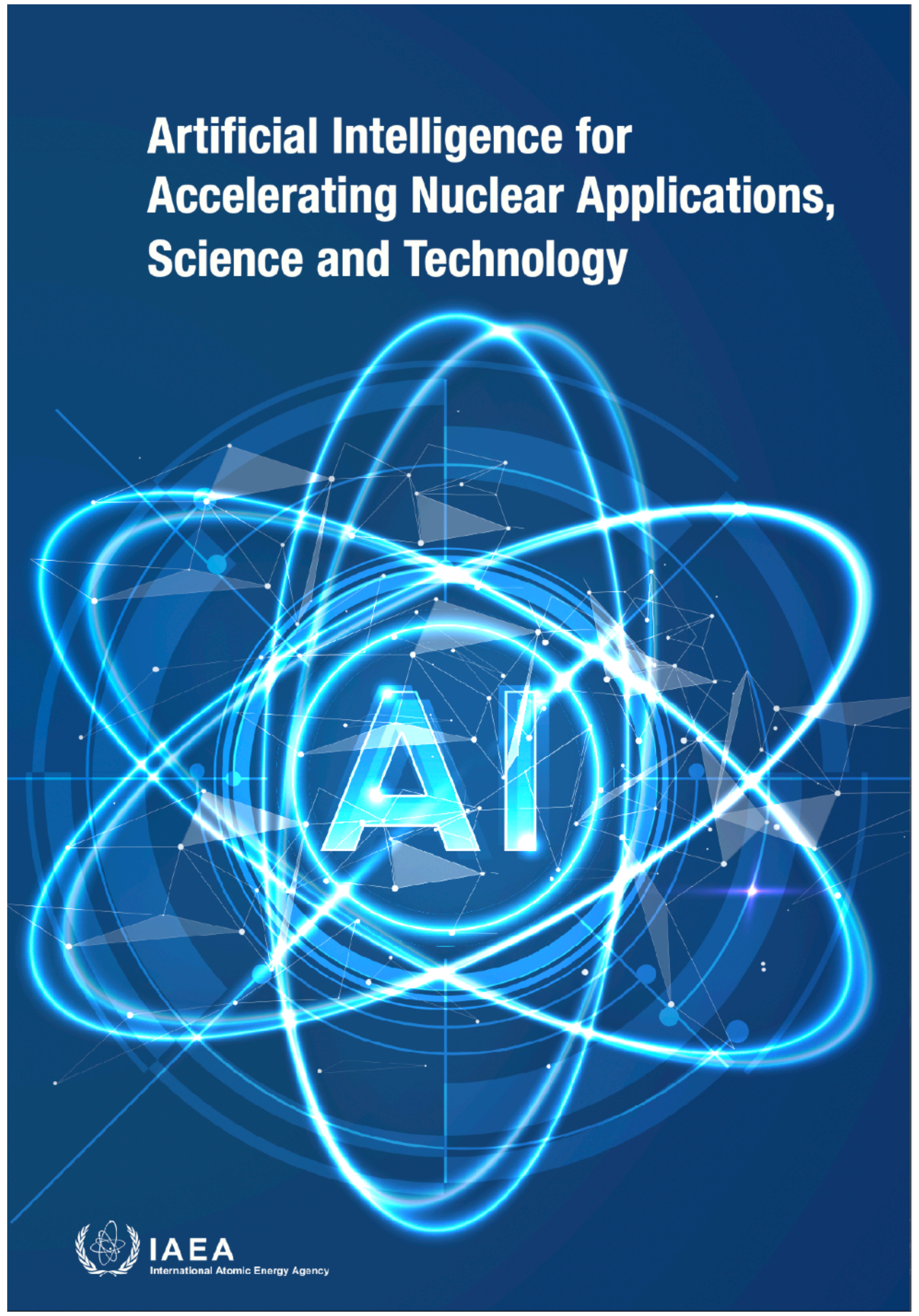
DAVIDSON

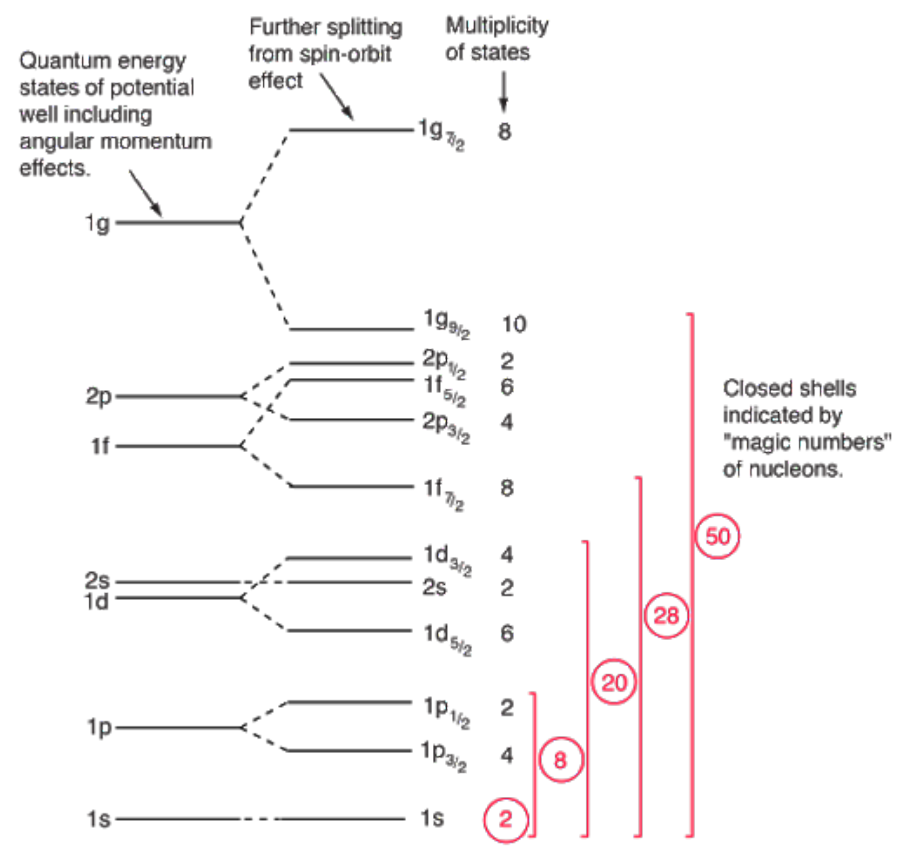
NSCL

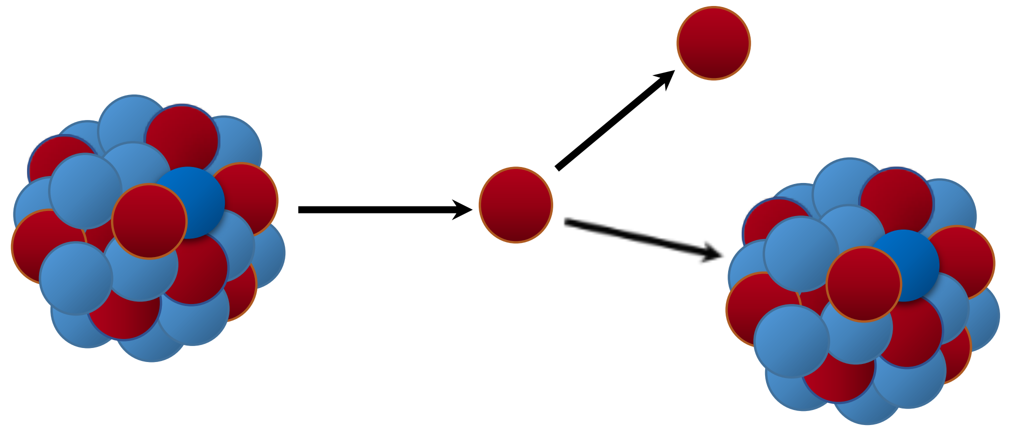
PhD: GPUs for Bayesian Neural Networks (😟)

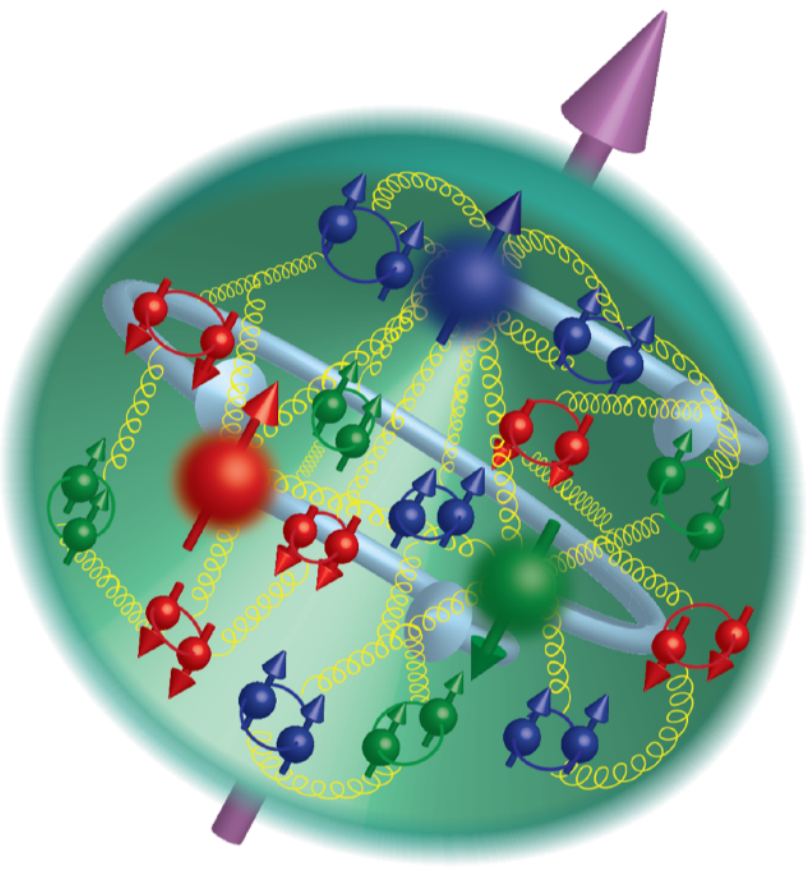
MICHELLE KUCHERA

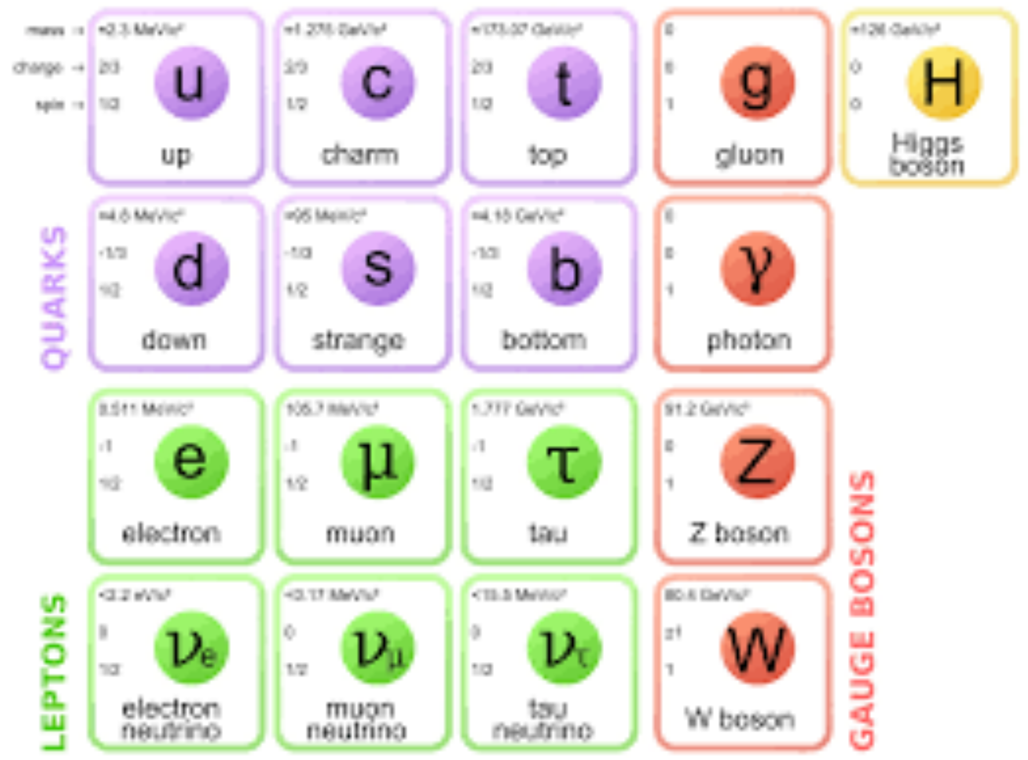
B.S., M.S. PHYSICS
M.S., PH.D. COMPUTATIONAL SCIENCE









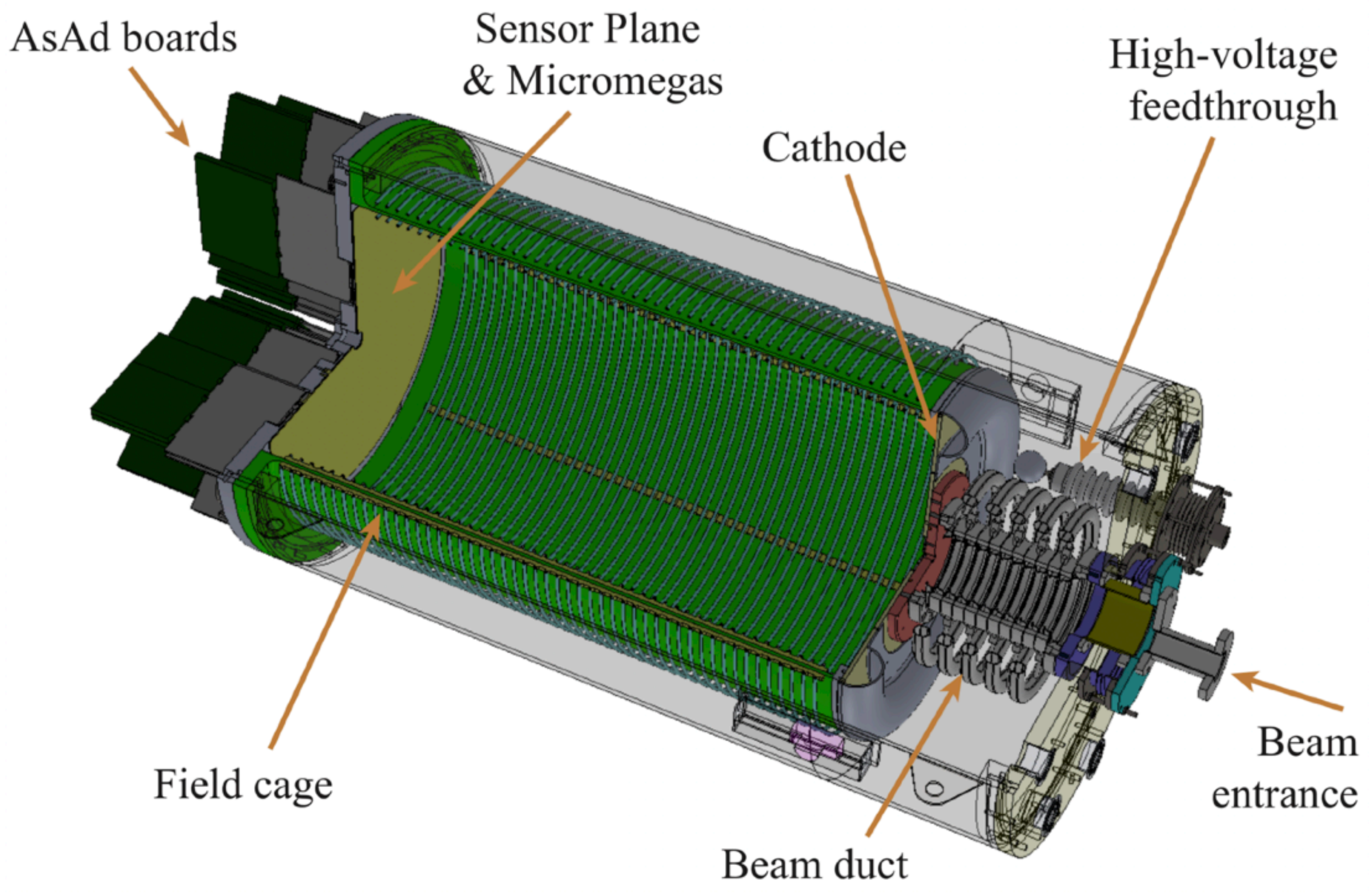


FRIB

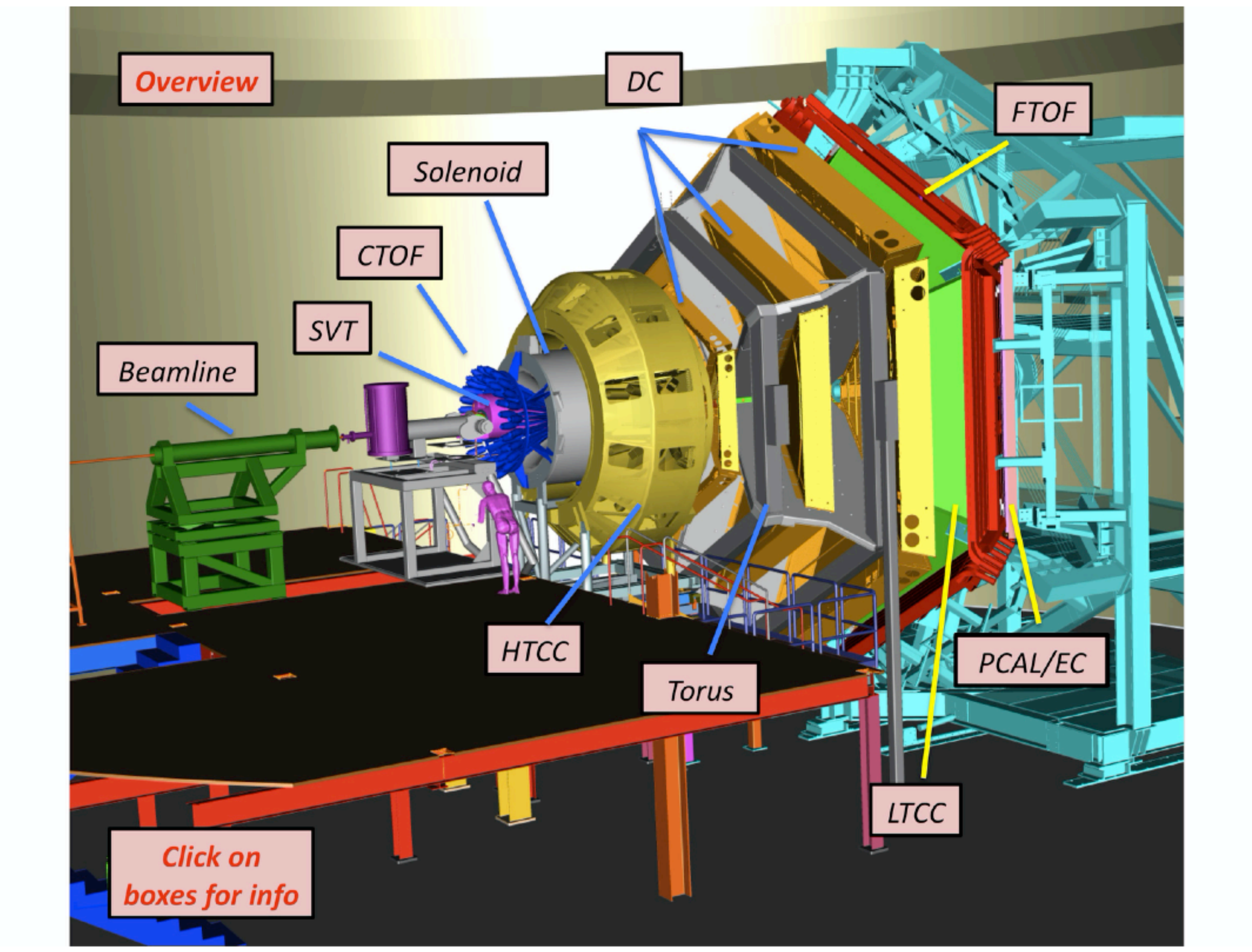
Jefferson Lab

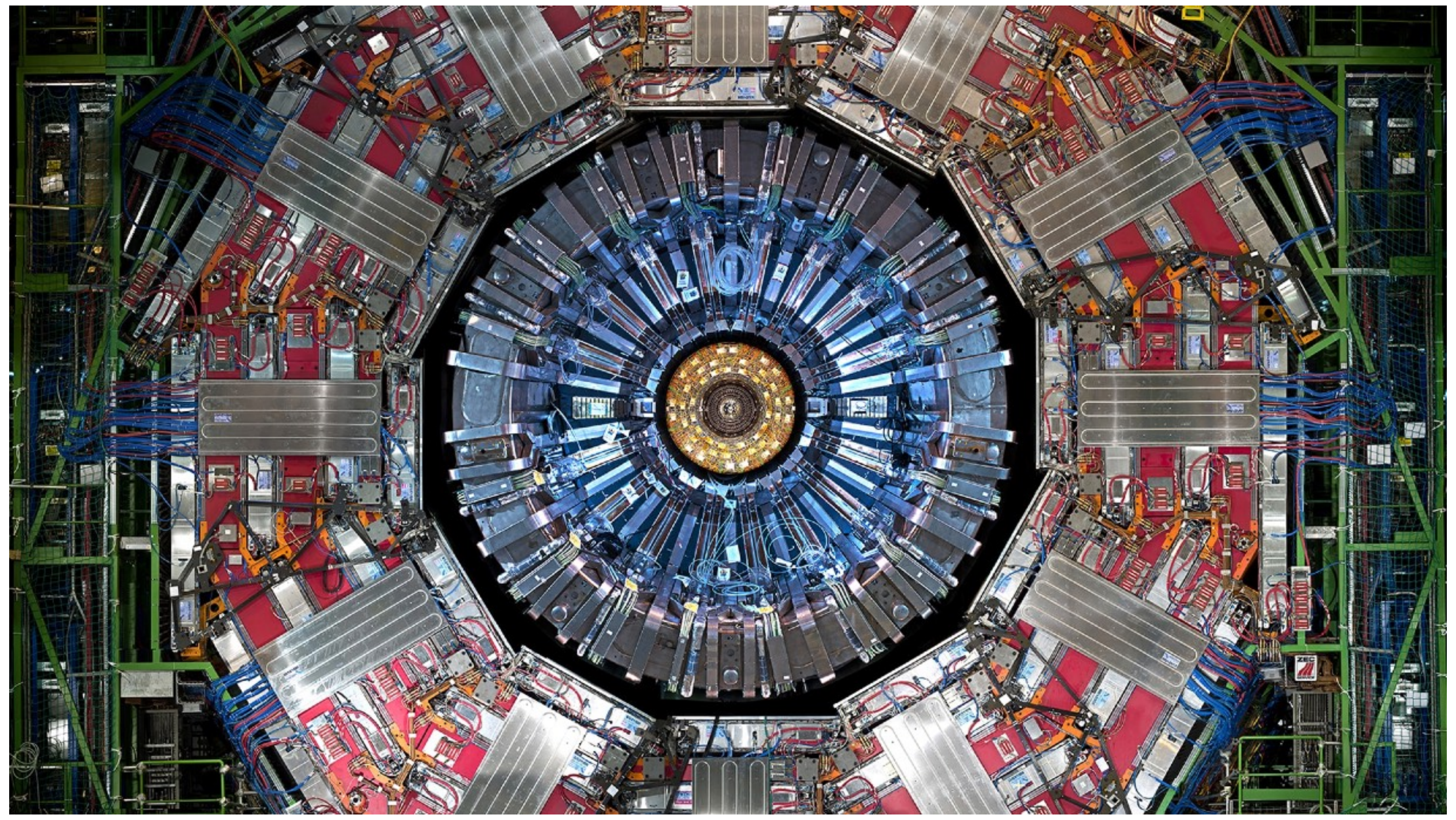
CERN

EXPERIMENTAL DATA



J. BRADT ET. AL., *NUCLEAR INSTRUMENTS AND METHODS*, 2017.



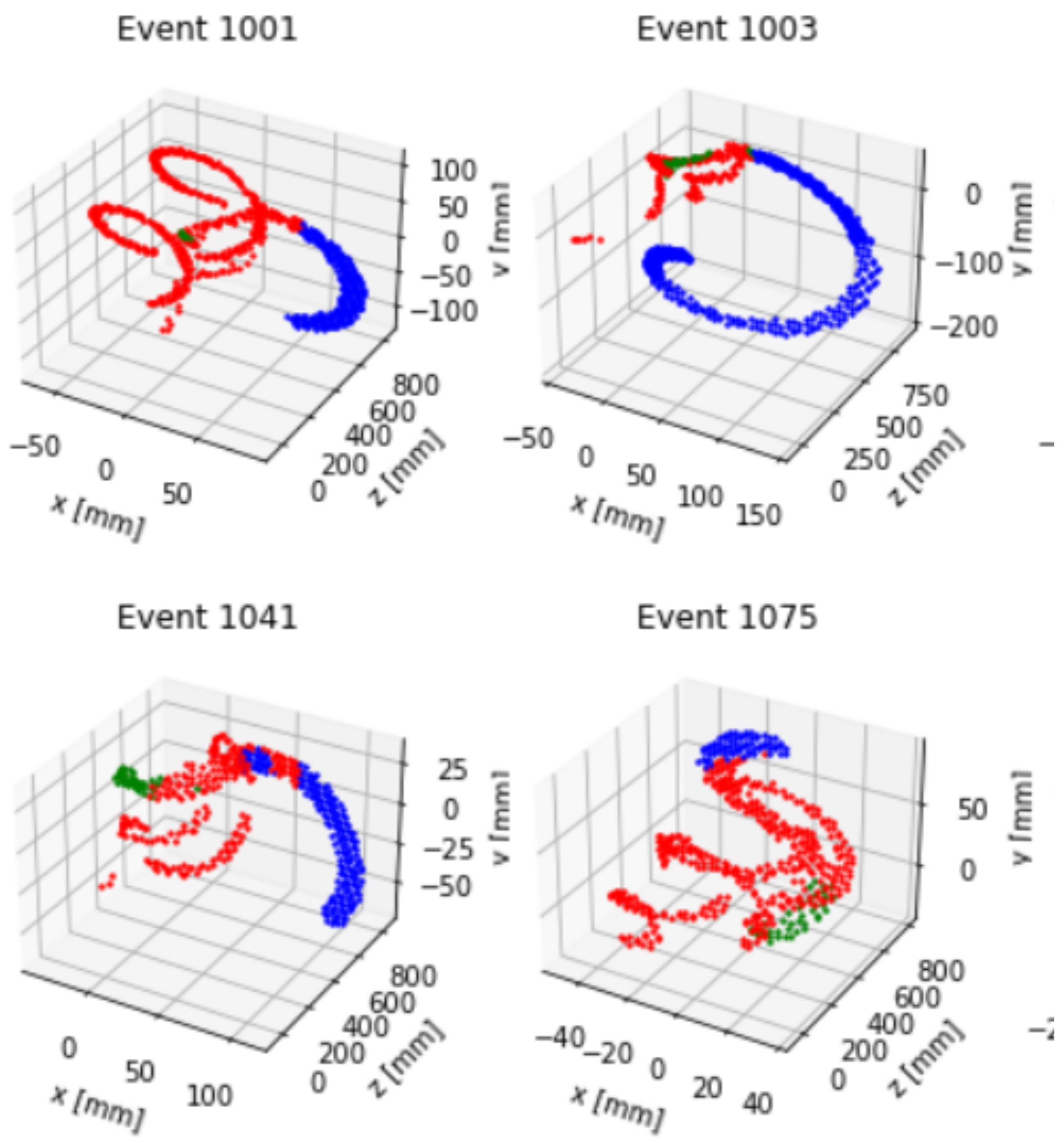


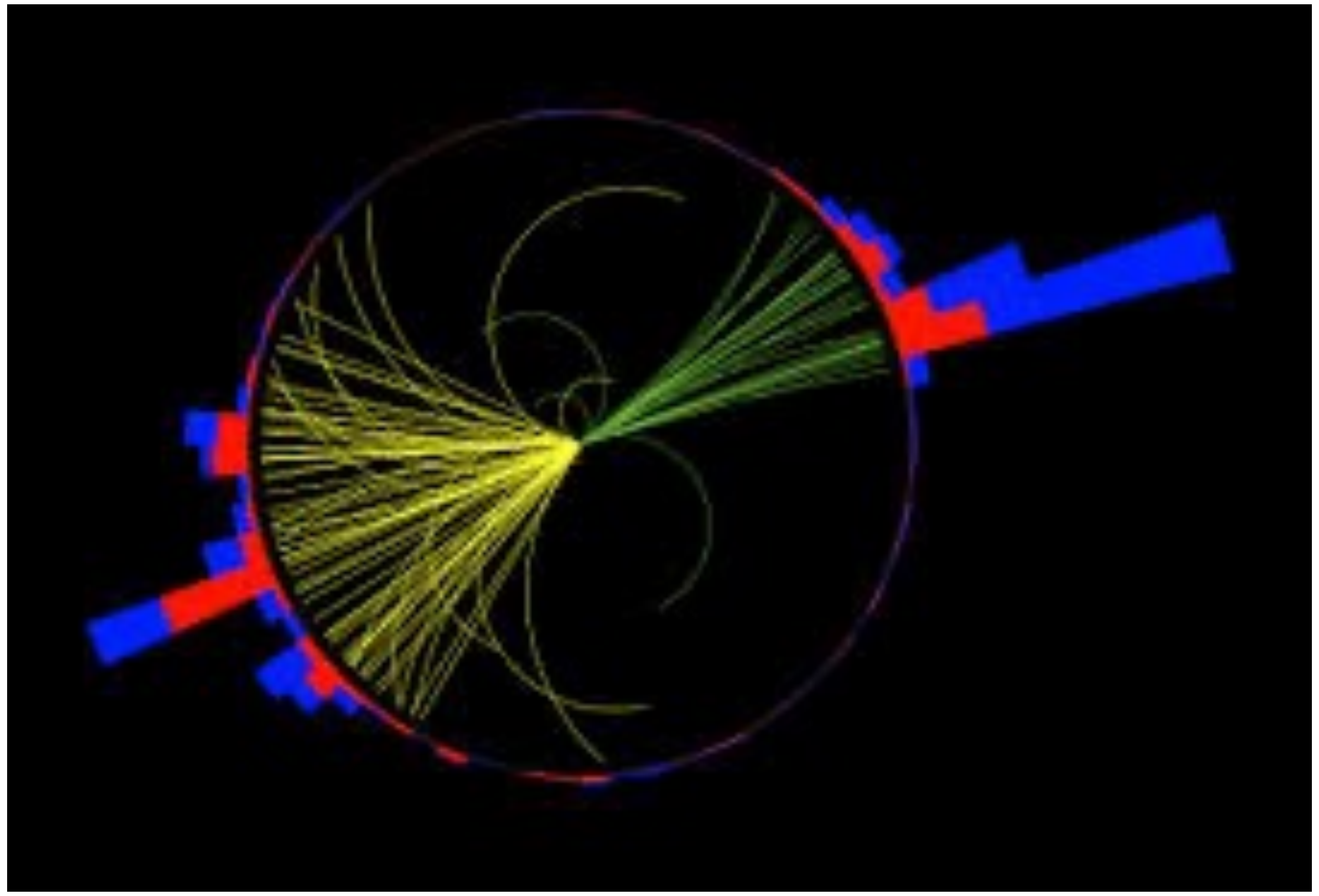
AT-TPC

CLAS 12

CMS

EXPERIMENTAL DATA





AT-TPC

CLAS 12

CMS

LECTURE 1 TOPICS

- Computational graphs
- Gradient-descent optimization
- Logistic regression
- Regression neural networks

MICHELLE KUCHERA
DAVIDSON COLLEGE

CPS-FR
MIT
25 AUGUST 2023

GOALS

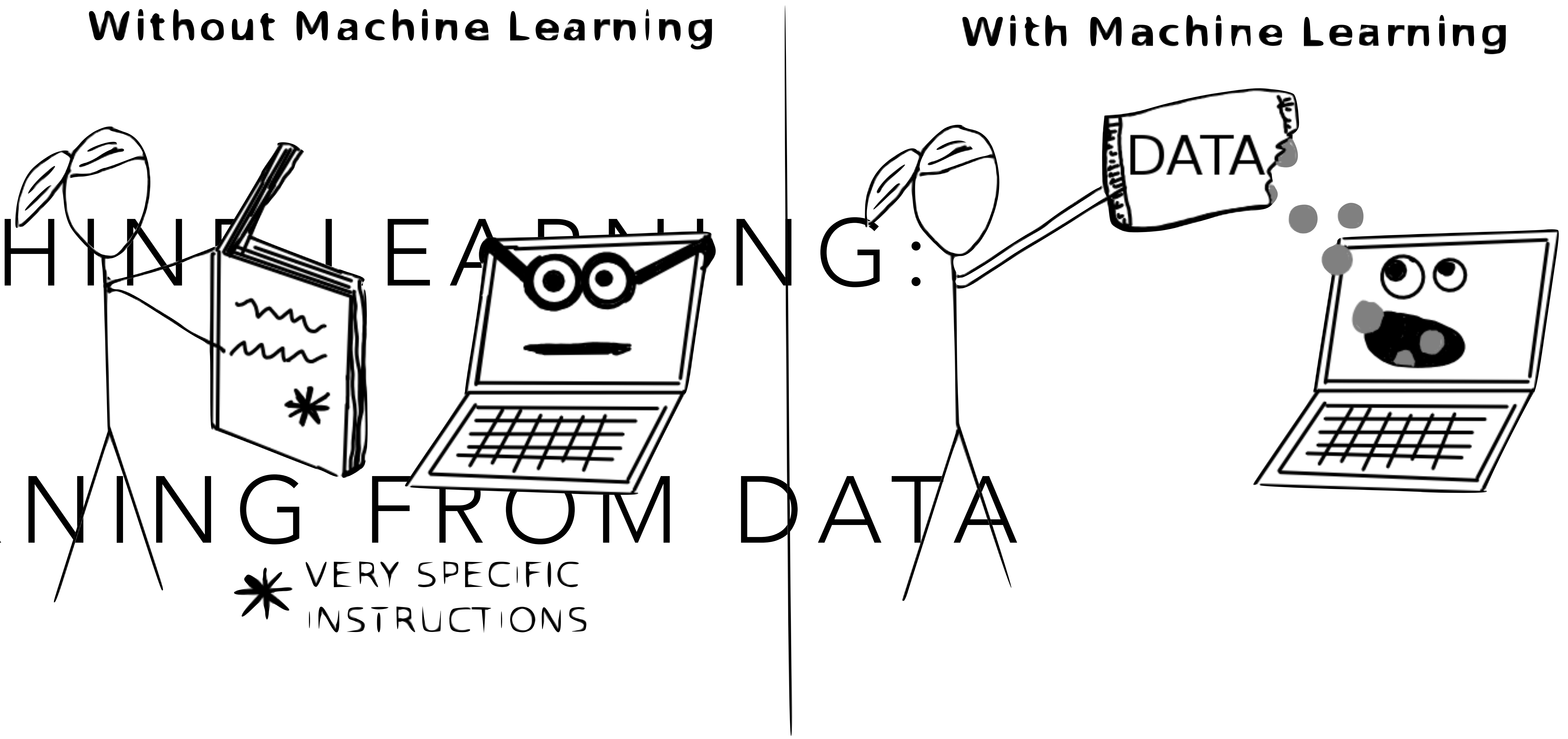
- Each of us learns something today
- Stop me with any questions

MICHELLE KUCHERA
DAVIDSON COLLEGE

COMMUNITY

- Each of you arrived here with your own backgrounds, specialty, and path in life
- Your experience and expertise are valuable here, no matter what it is
- If the activity is within your background, help others!
- If you are totally (or a little) lost, ask for help!
- It is our shared goal to have **each** of us leave with some new skill/knowledge/understanding

MACHINE LEARNING: LEARNING FROM DATA

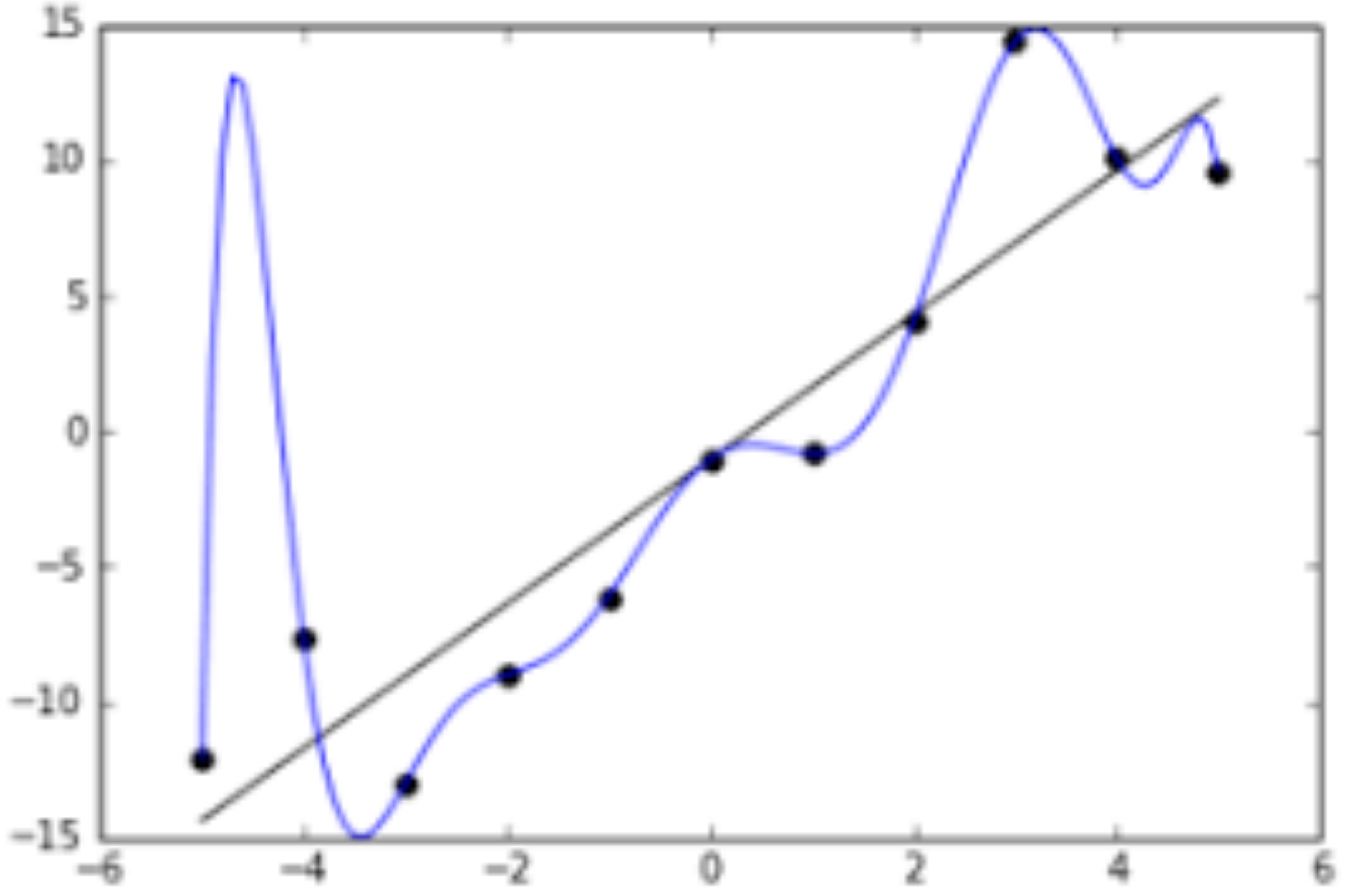


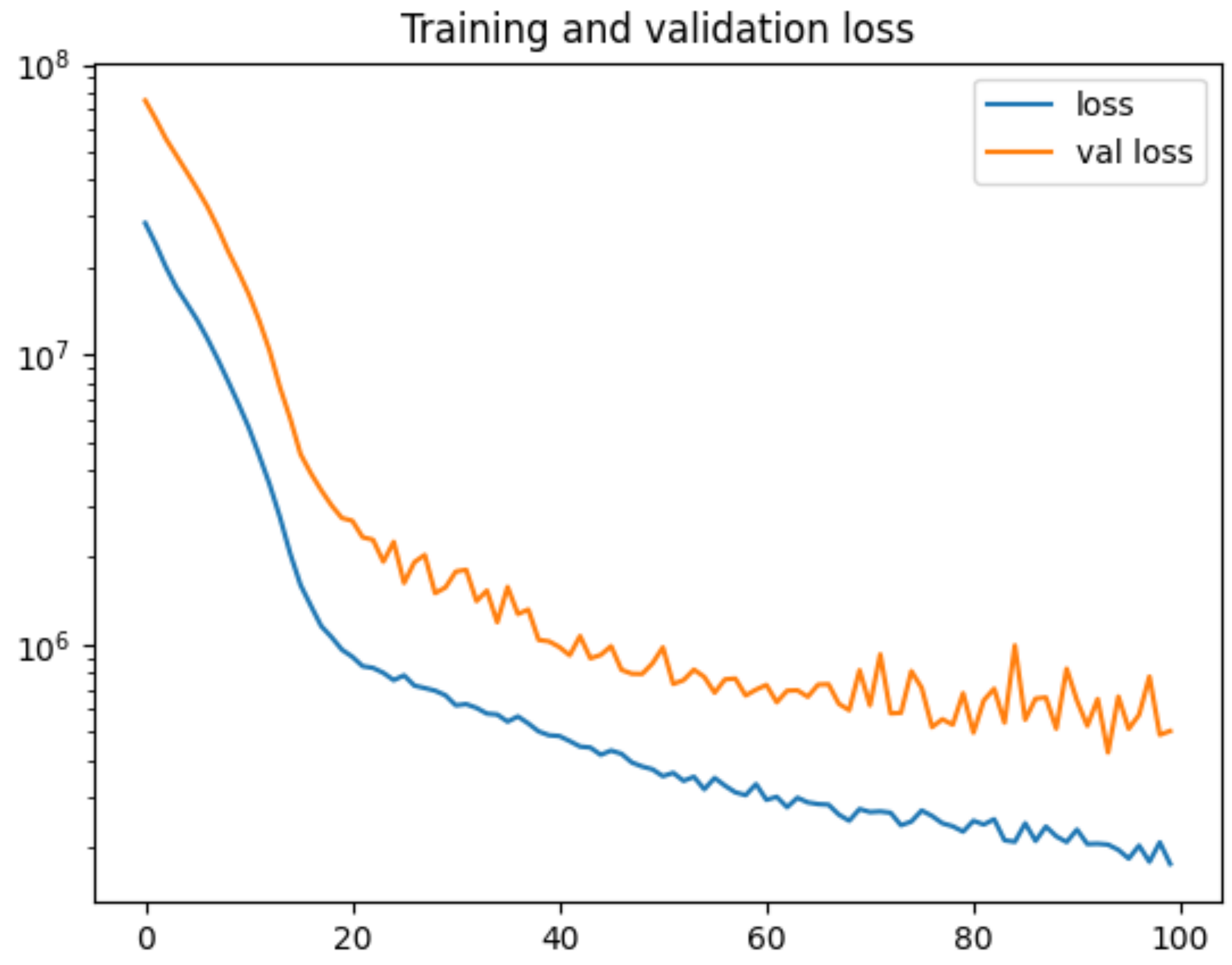
Without Machine Learning

With Machine Learning

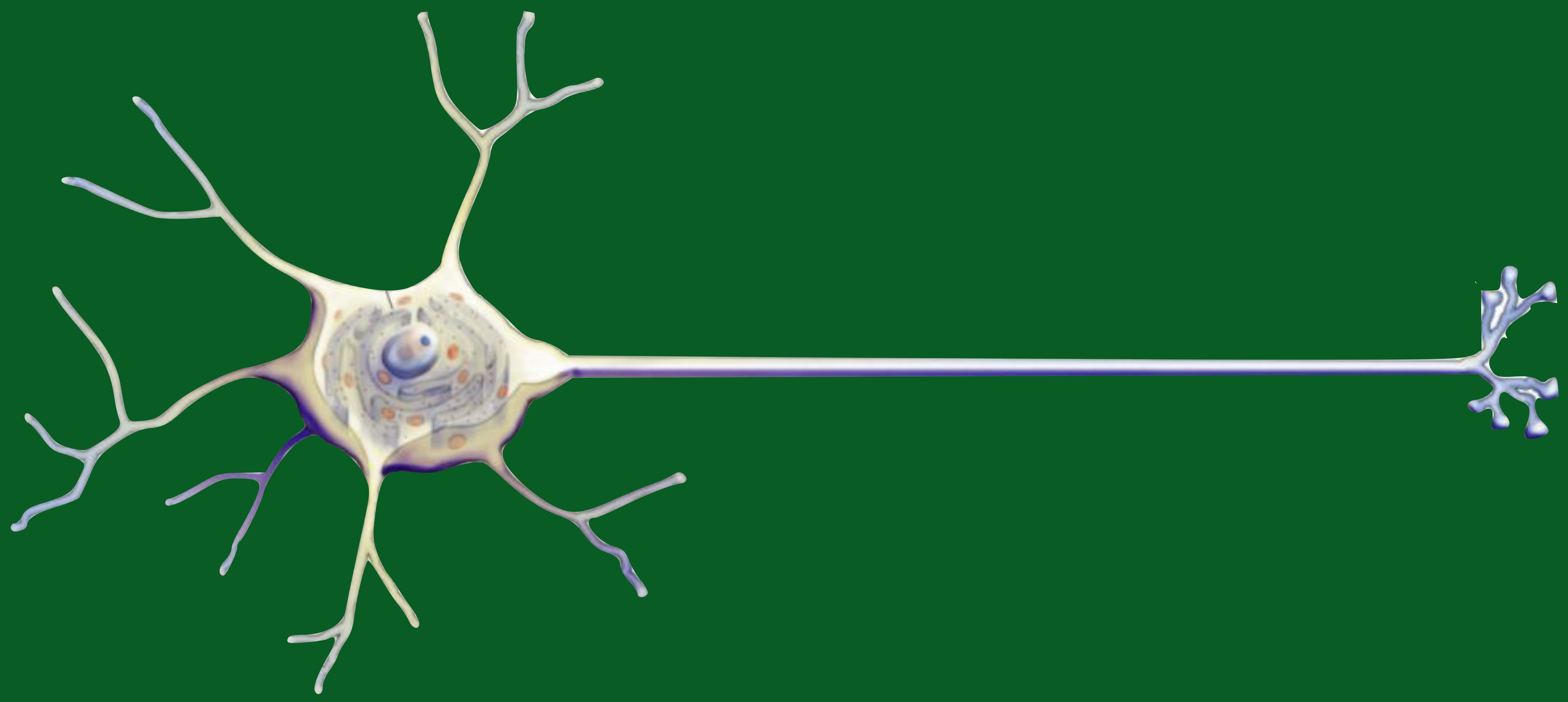
Learning from data is a **paradigm shift** in thinking about predictive models

VERY SPECIFIC
INSTRUCTIONS

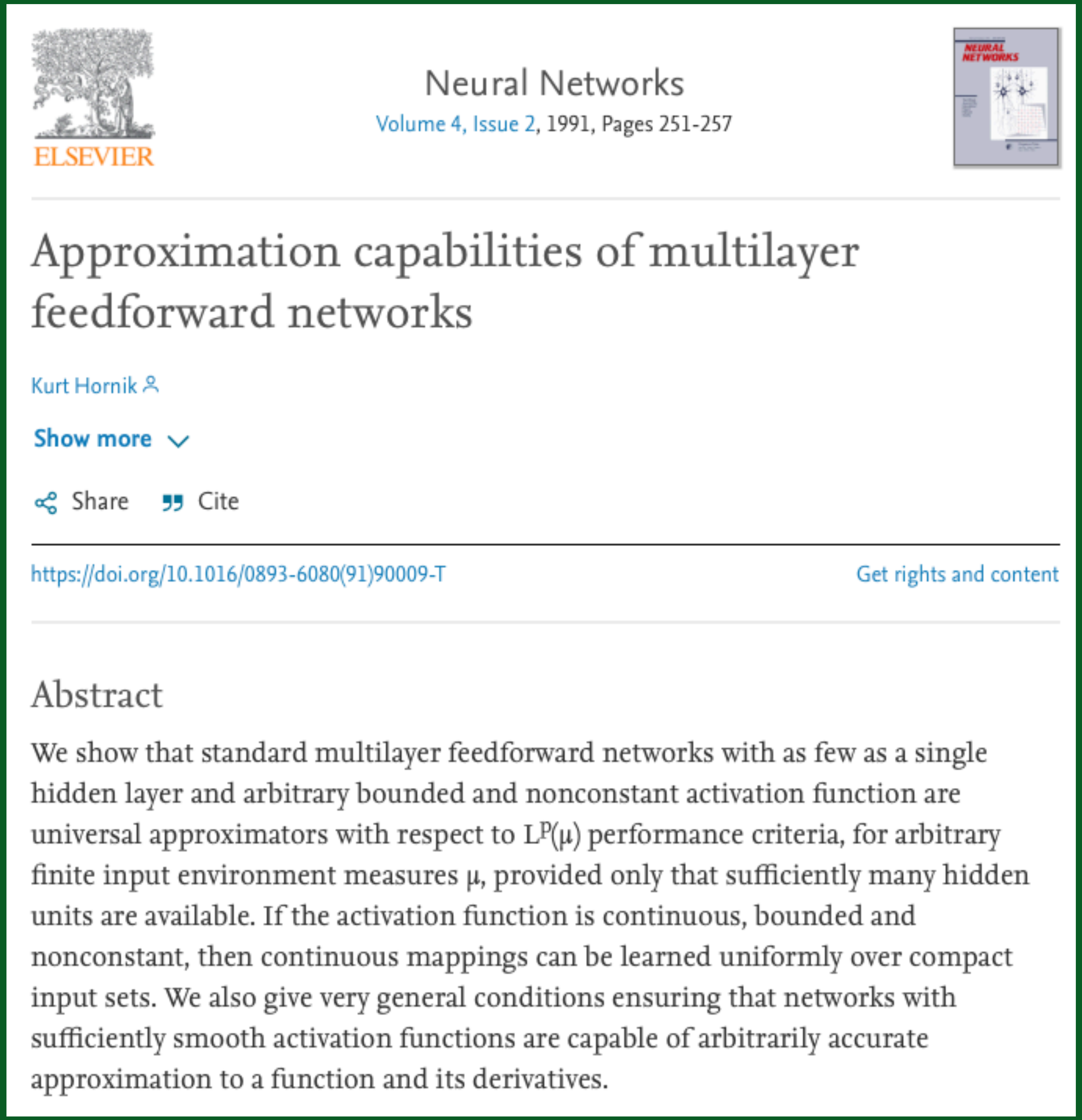




NEURON



MATHEMATICS



ELSEVIER

Neural Networks
Volume 4, Issue 2, 1991, Pages 251-257

Approximation capabilities of multilayer feedforward networks

Kurt Hornik

Show more

Share Cite

https://doi.org/10.1016/0893-6080(91)90009-T Get rights and content

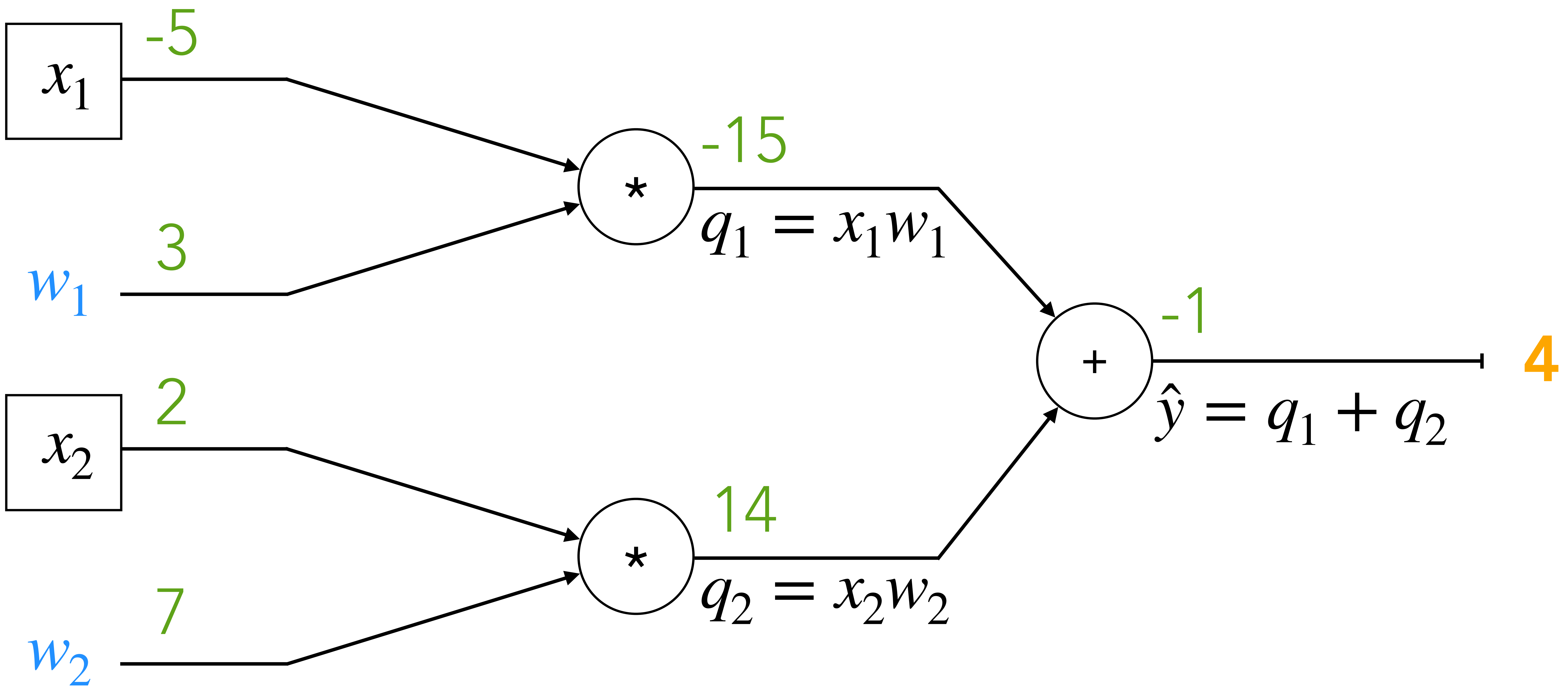
Abstract

We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures μ, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.

MATHEMATICS

COMPUTATIONAL GRAPH

$$\hat{y} = x_1 w_1 + x_2 w_2$$

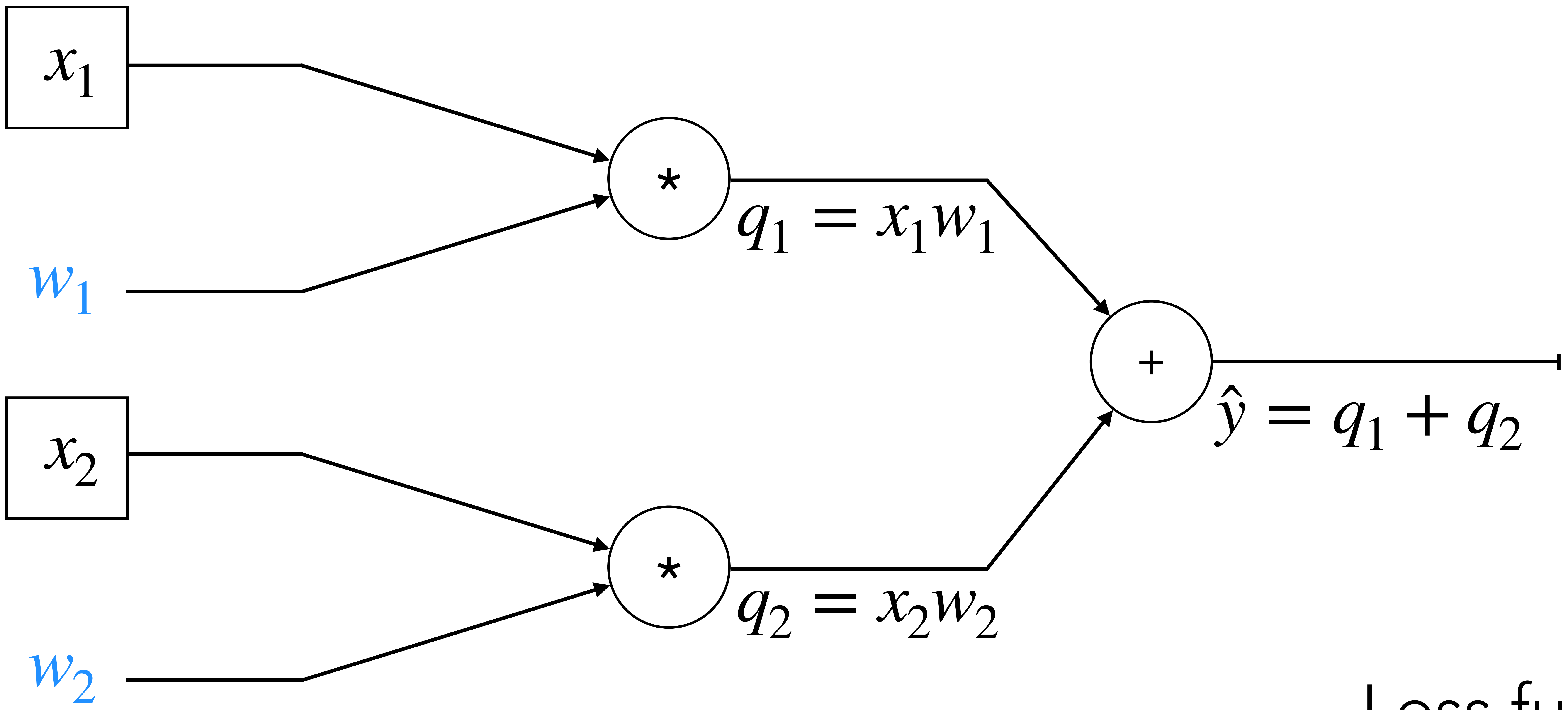


MACHINE LEARNING

SUPERVISED LEARNING

REGRESSION

$$\hat{y} = x_1 w_1 + x_2 w_2$$

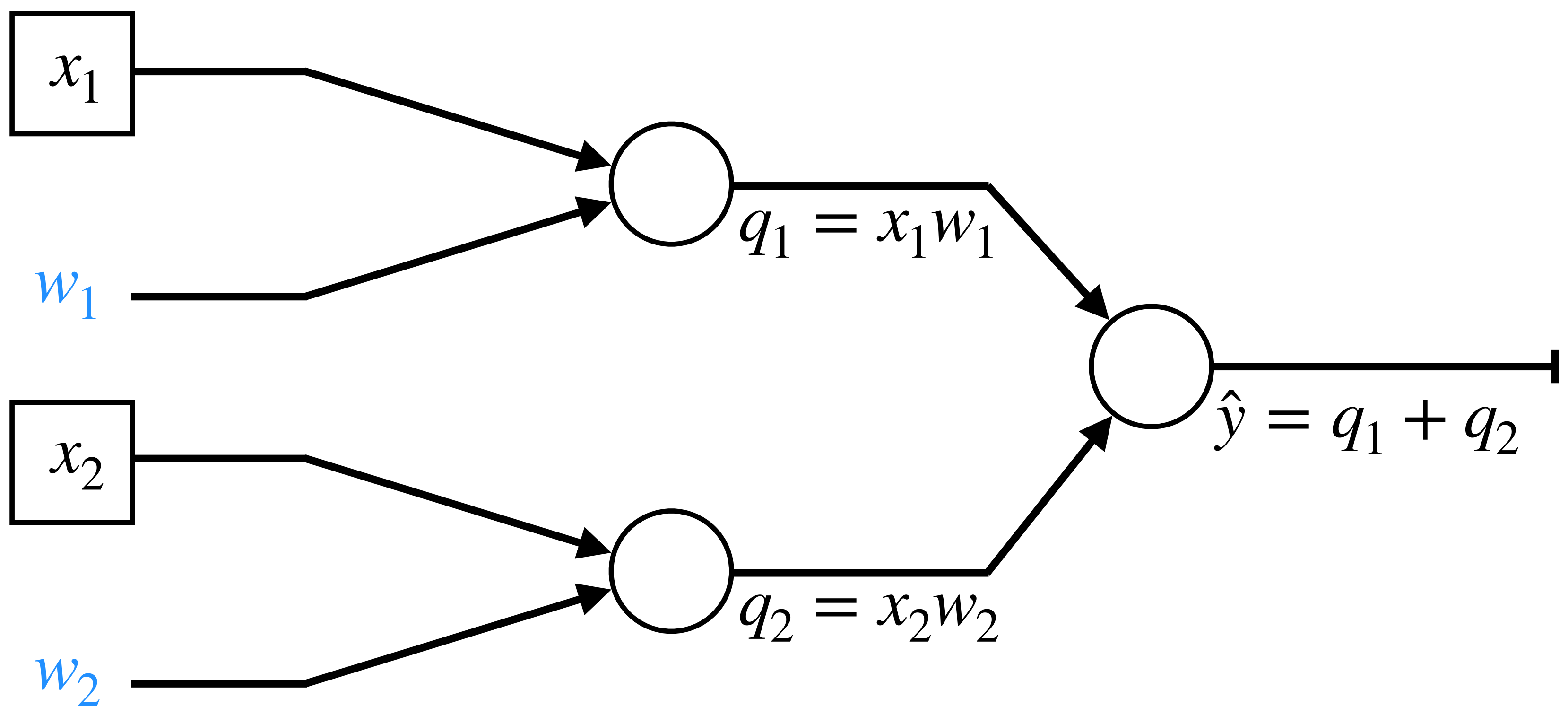


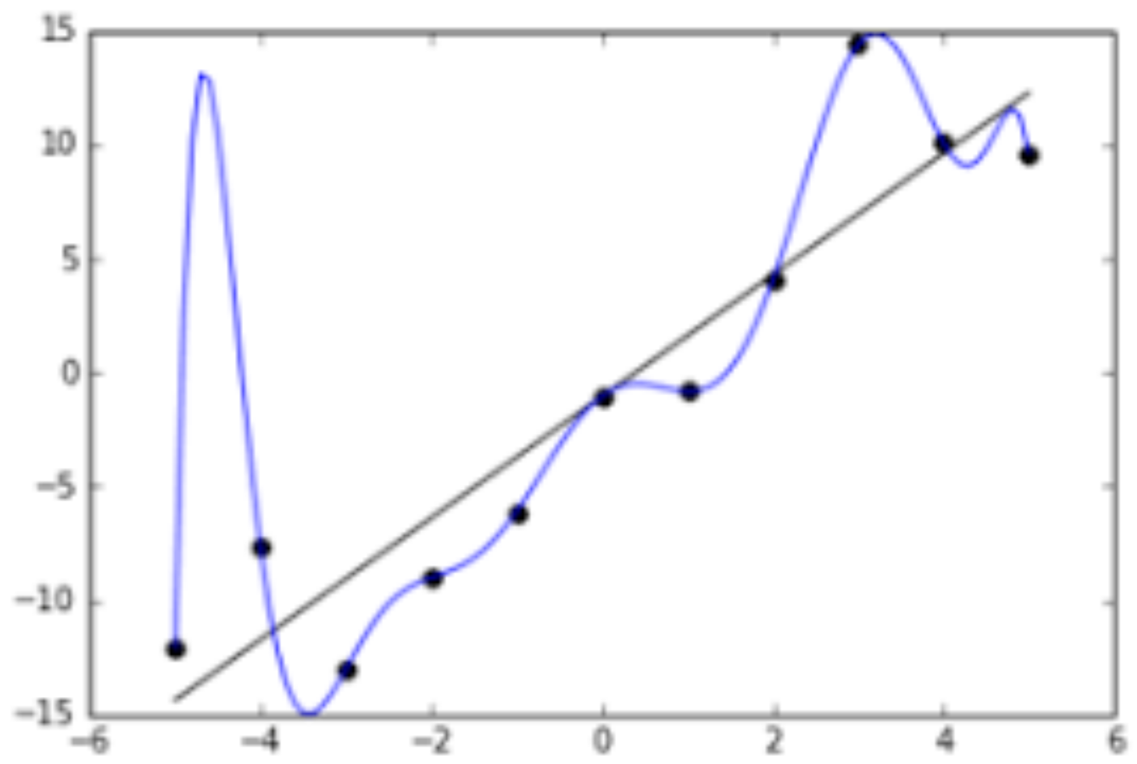
Loss function

$$J(w) = \hat{y} - y$$

SUPERVISED LEARNING

$$\hat{y} = x_1 w_1 + x_2 w_2$$





	Feature 1	Feature 2	Feature 3	Target
Example 1				
Example 2				
Example 3				
Example 4				

Loss function

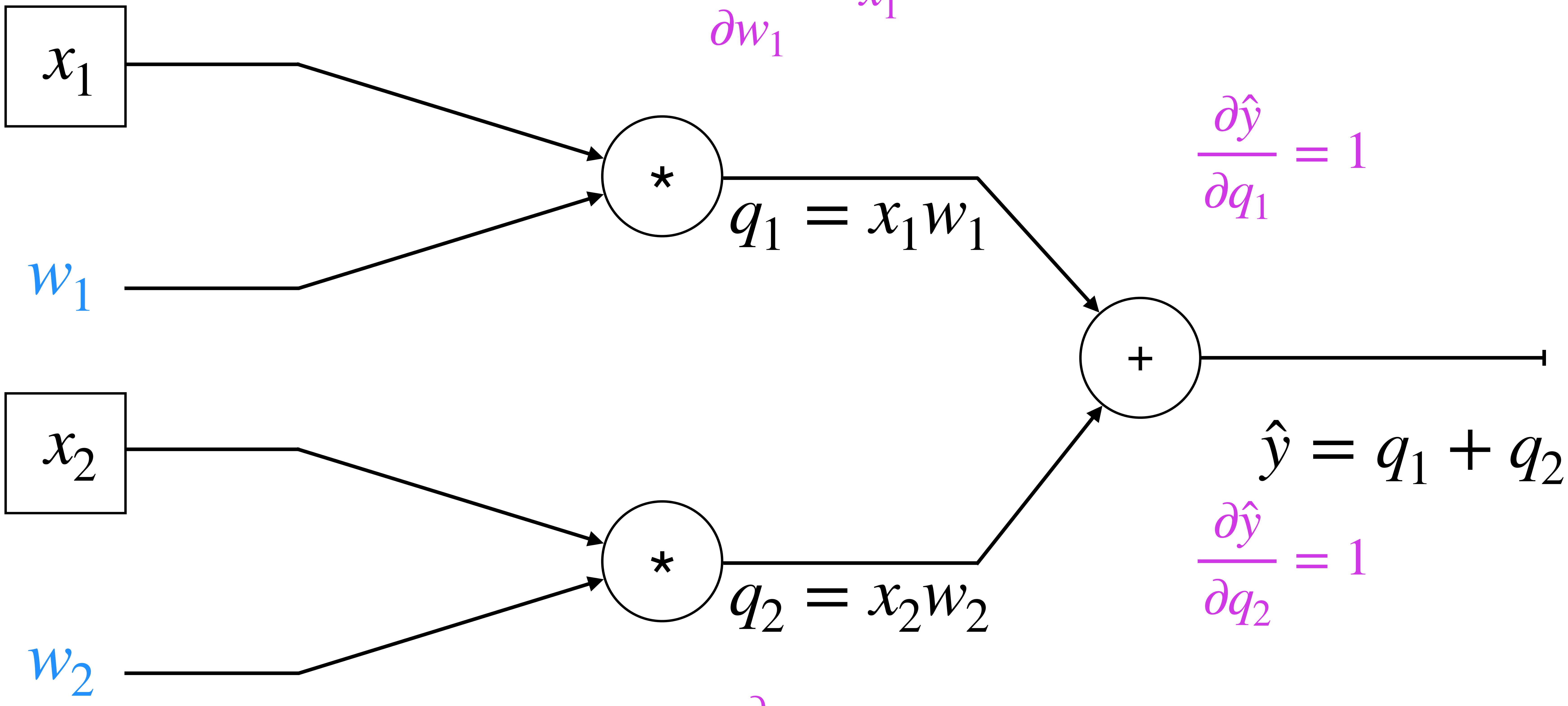
MSE across N examples

$$J(w) = \frac{1}{N} \sum_{i=0}^{N} (\hat{y}_i - y_i)^2$$

BACKPROPAGATION

$$w_1 = w_1 - \eta * \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial q_1} \frac{\partial q_1}{\partial w_1}$$

$$\frac{\partial q_1}{\partial w_1} = x_1$$



$$\frac{\partial \hat{y}}{\partial q_1} = 1$$

$$\frac{\partial J}{\partial \hat{y}} = 1$$

$$\frac{\partial \hat{y}}{\partial q_2} = 1$$

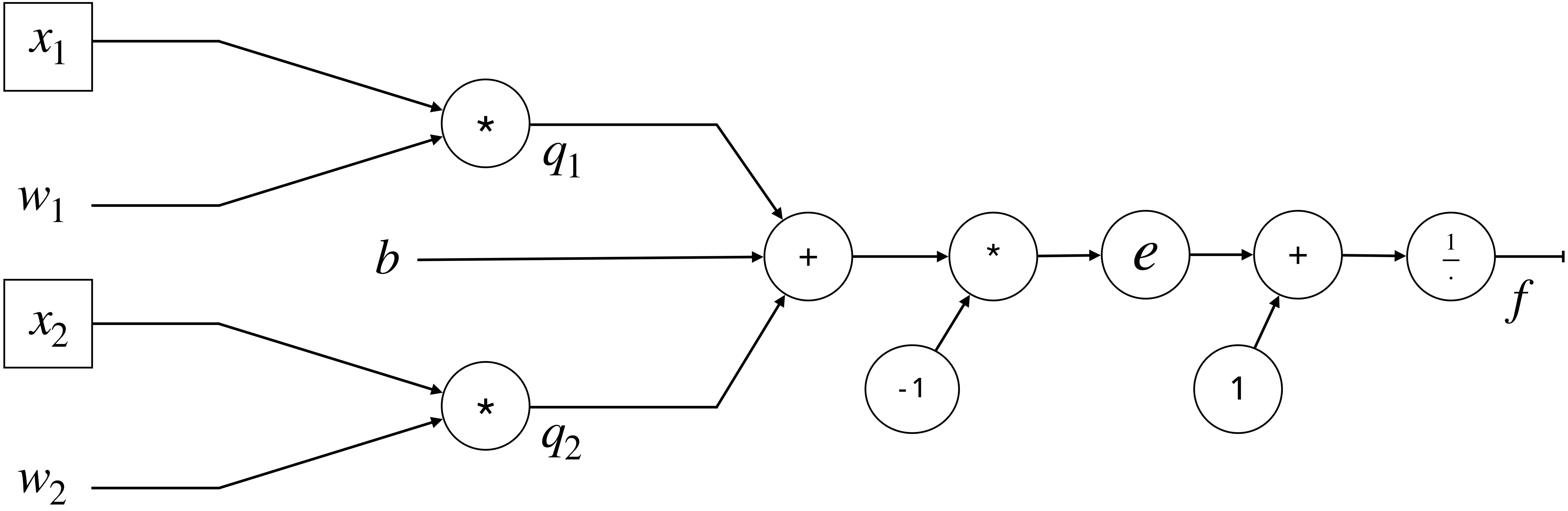
$$\frac{\partial q_2}{\partial w_2} = x_2$$

$$w_2 = w_2 - \eta * \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial q_2} \frac{\partial q_1}{\partial w_2}$$

Loss function

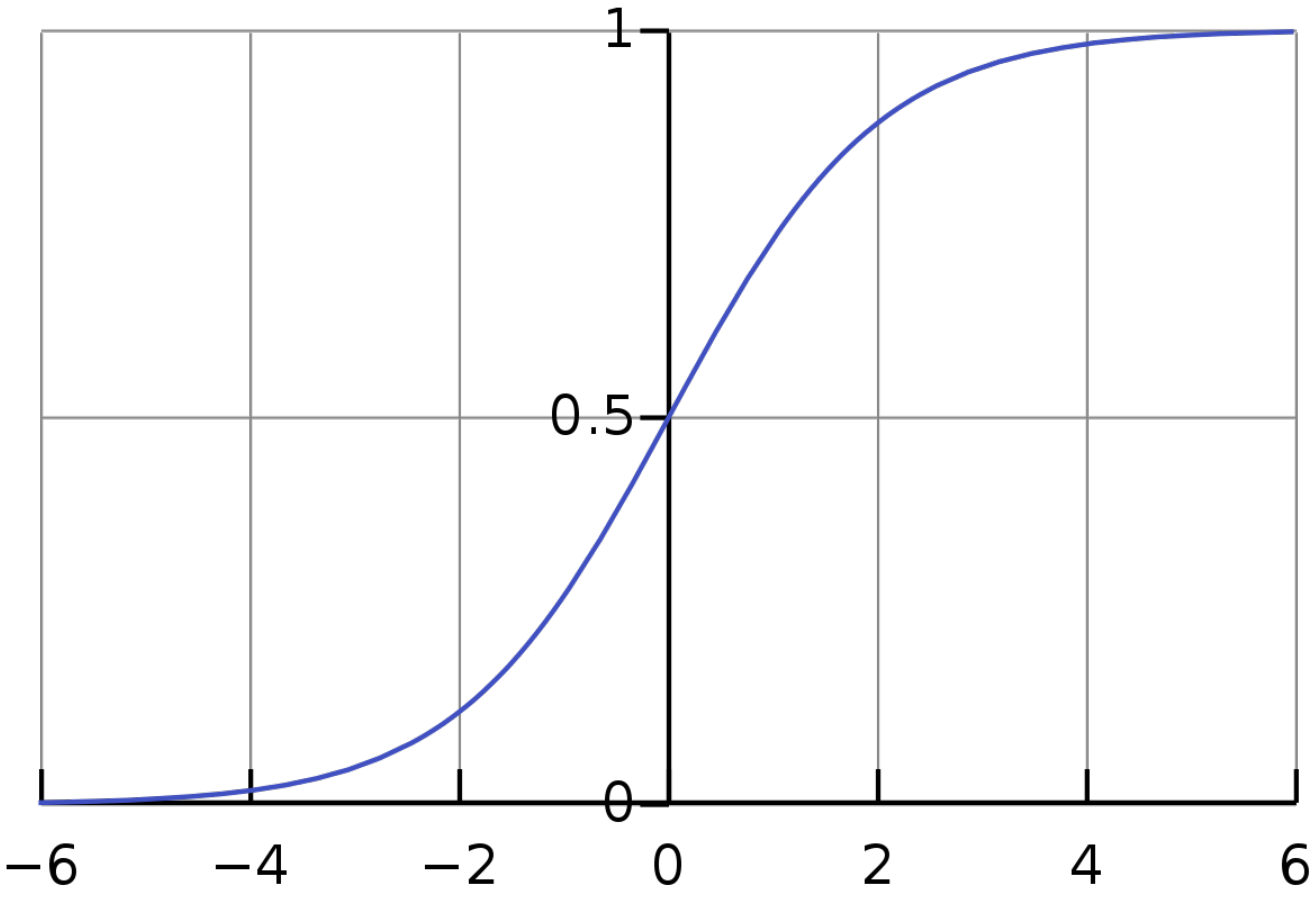
$$J(w) = \hat{y} - y$$

LOGISTIC REGRESSION

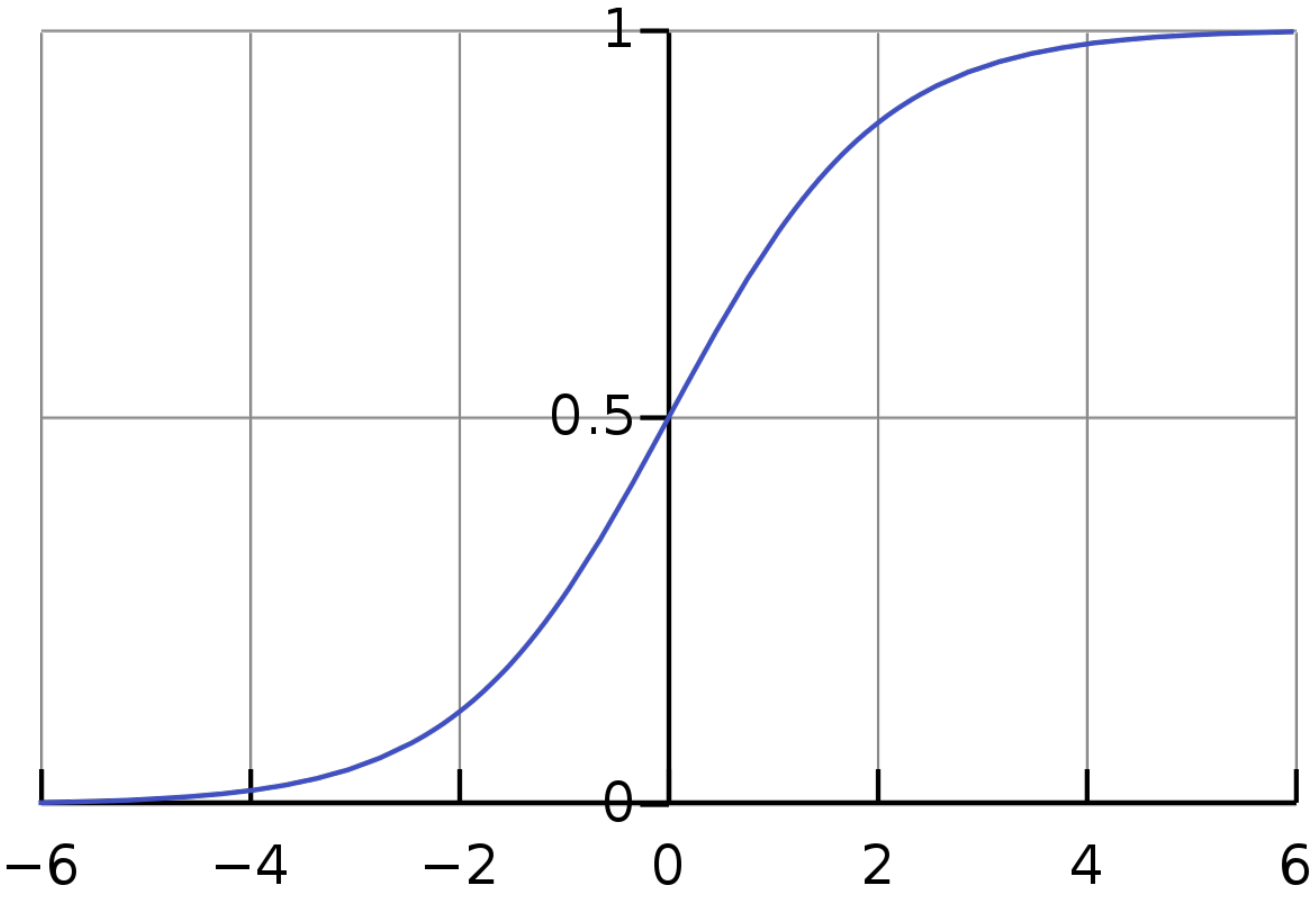


$$f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2 + b)}}$$

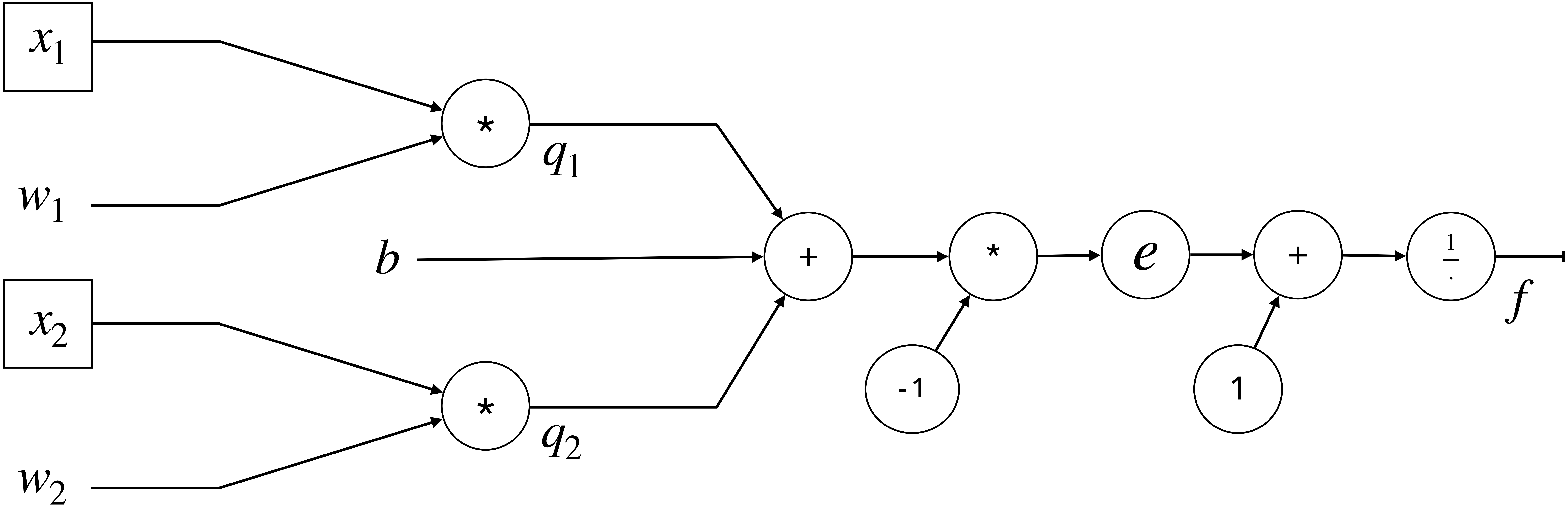
LOGISTIC REGRESSION



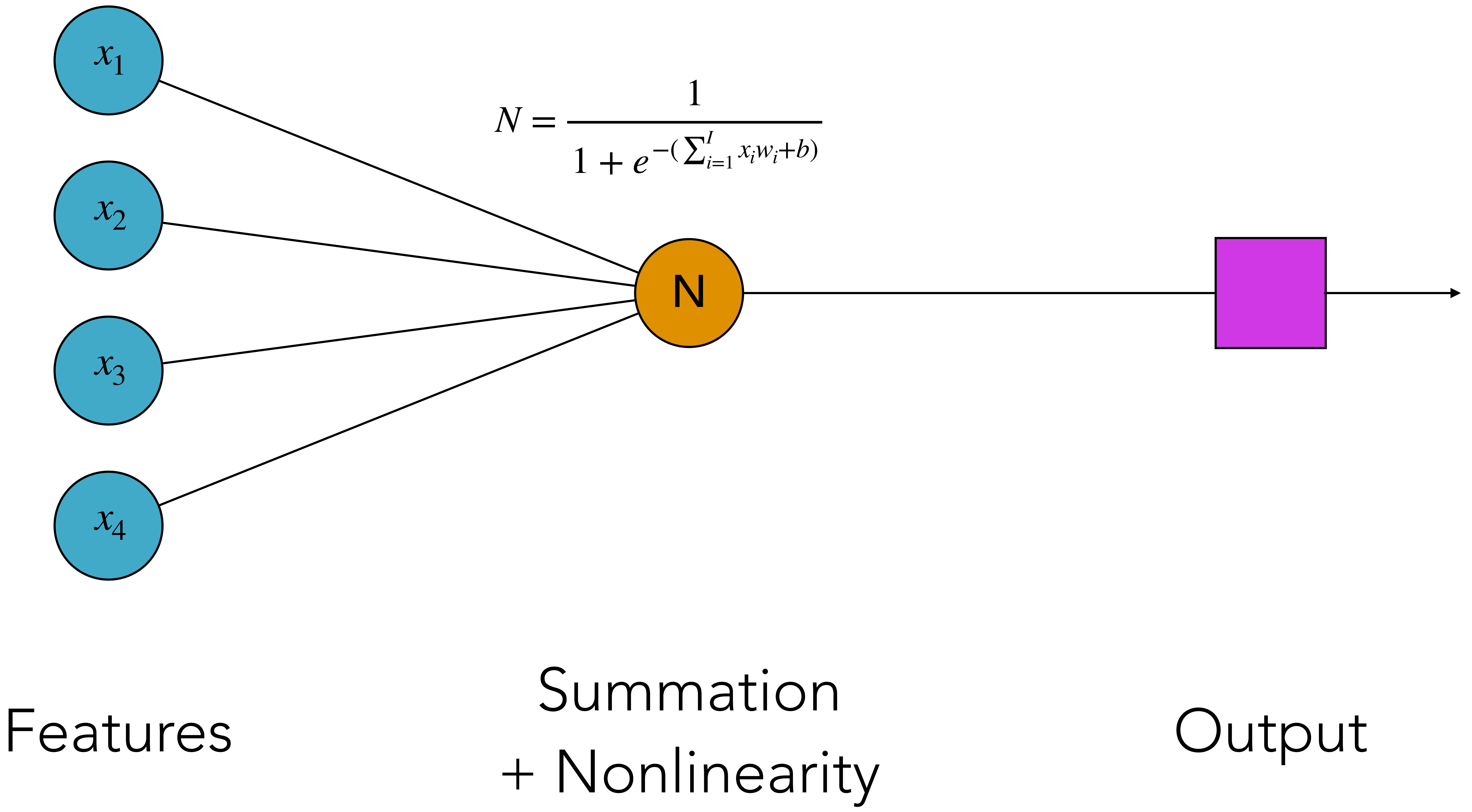
CLASSIFICATION



LOGISTIC REGRESSION



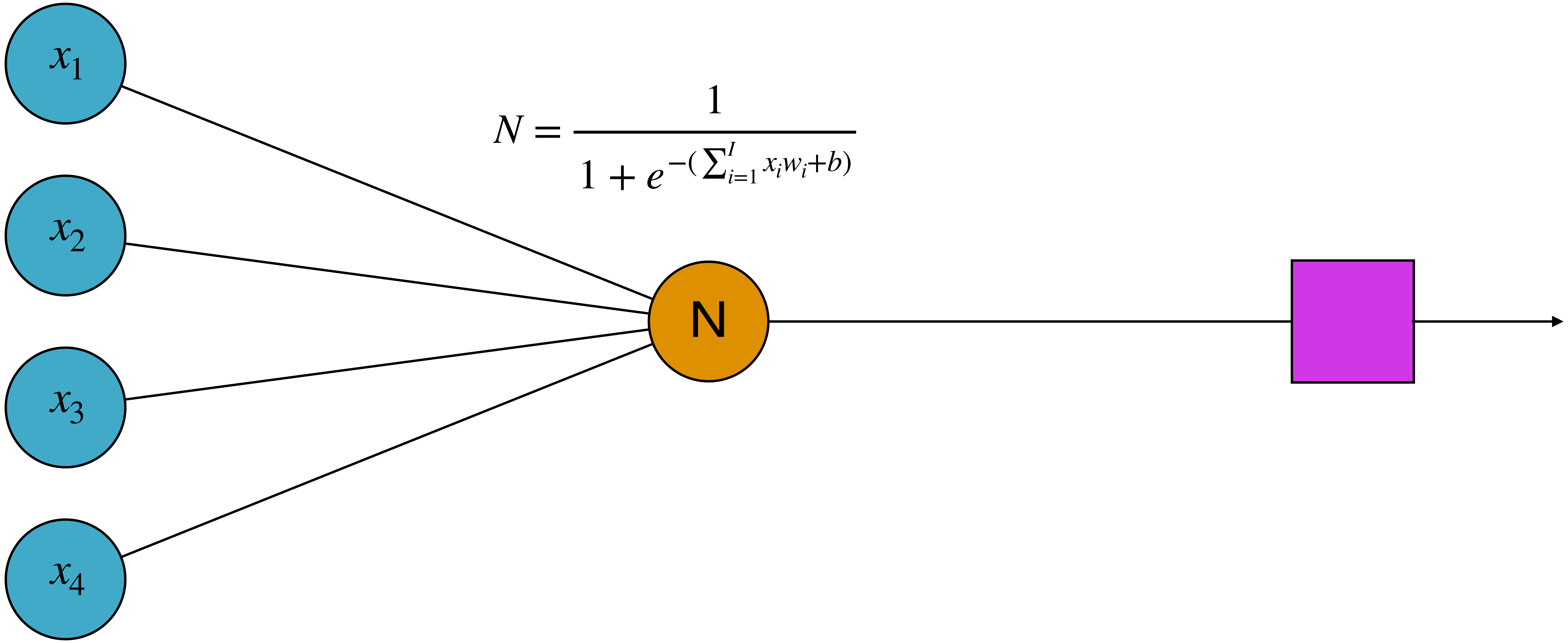
$$f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2 + b)}}$$



x_1
x_2
x_3
x_4
$N = \frac{1}{1 + e^{-(\sum_{i=1}^{I} x_i w_i + b)}}$
N
Features
Summation
+ Nonlinearity
Output

CHECK: HOW MANY TRAINABLE PARAMETERS?

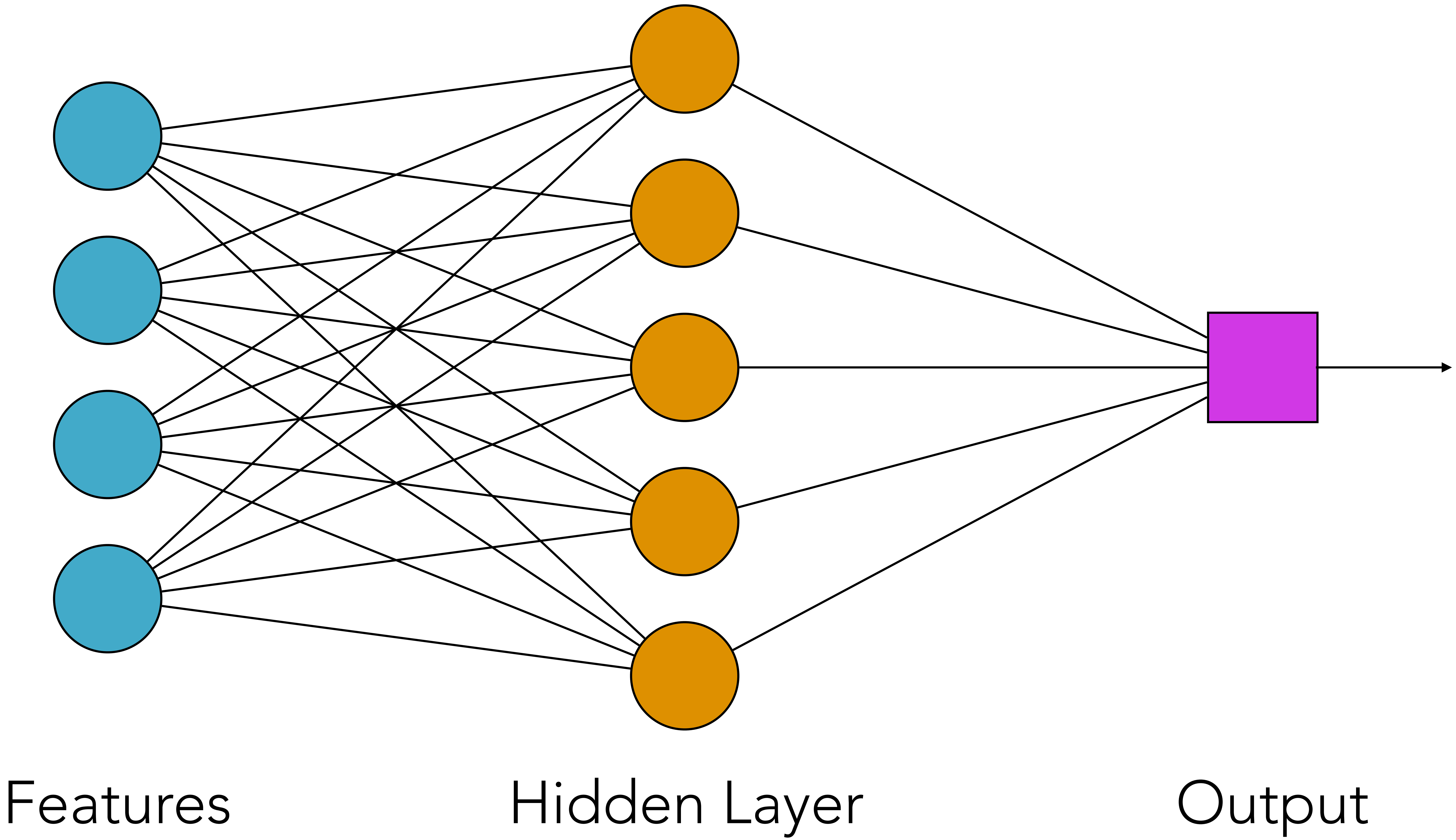
$$N = \frac{1}{1 + e^{-(\sum_{i=1}^{I} x_i w_i + b)}}$$



Features

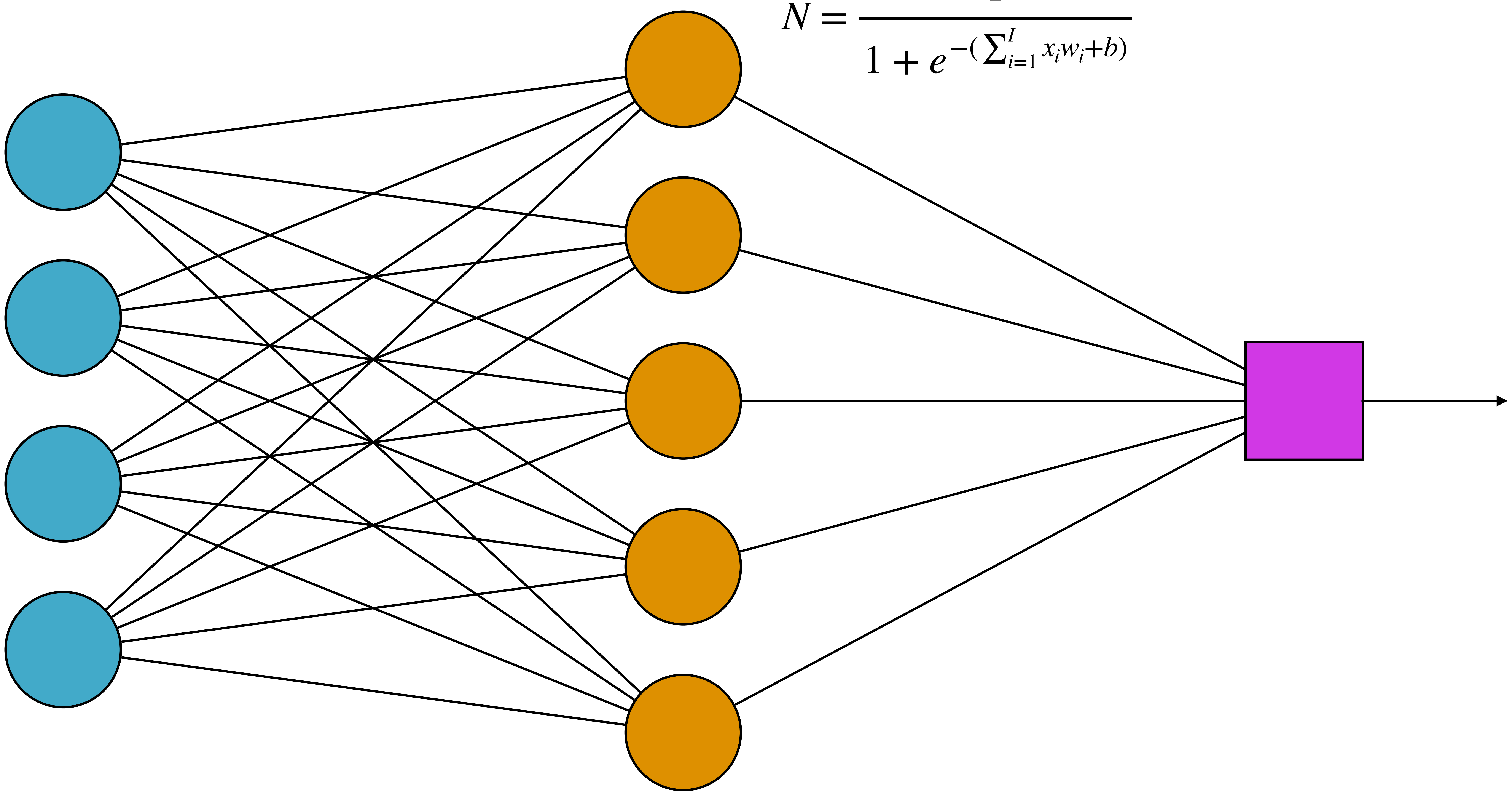
Summation
+ Nonlinearity

Output


Features
Hidden Layer
Output

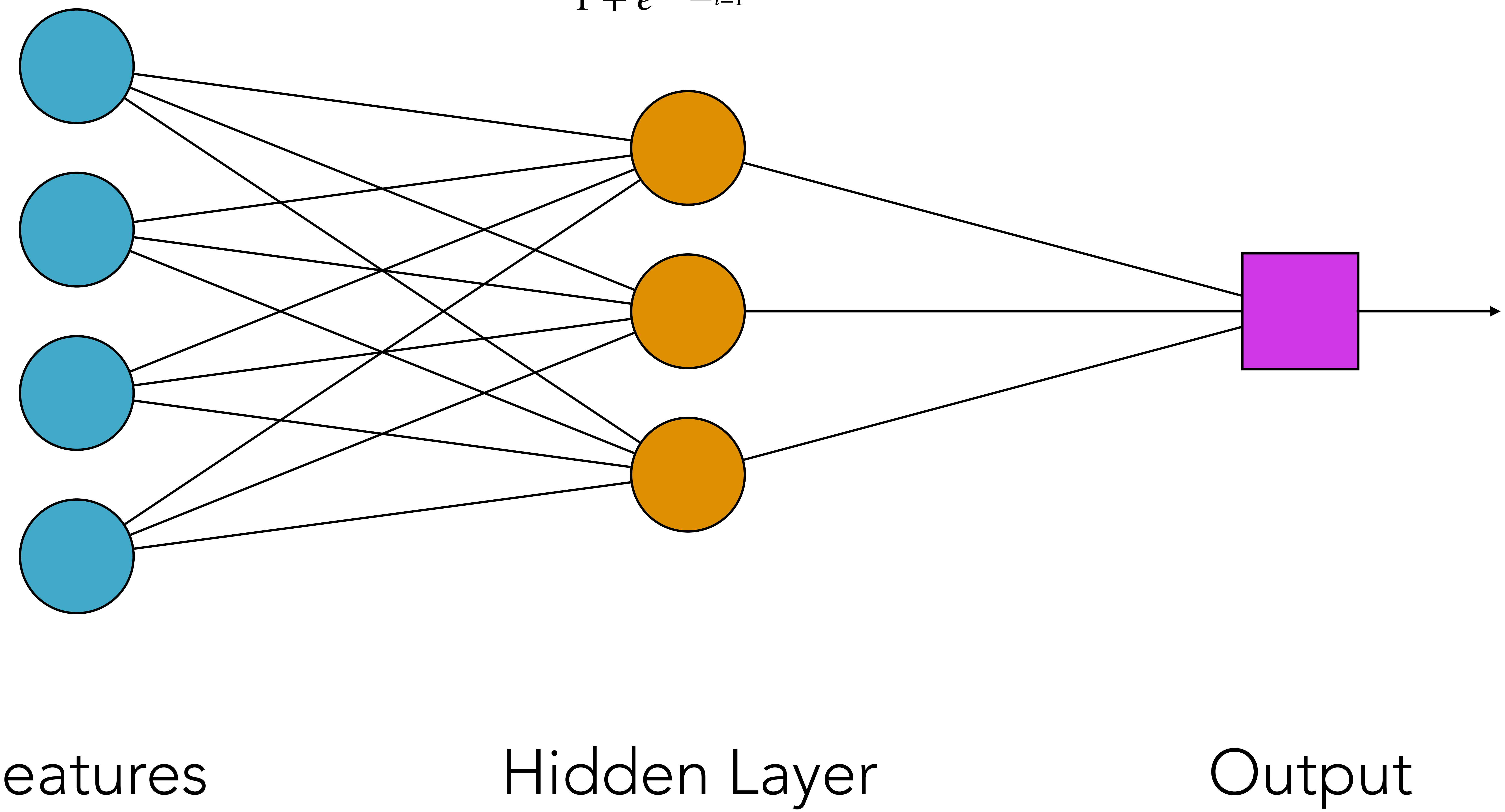
CHECK: HOW MANY TRAINABLE PARAMETERS?

$$N = \frac{1}{1 + e^{-(\sum_{i=1}^{I} x_i w_i + b)}}$$



CHECK: HOW MANY TRAINABLE PARAMETERS?

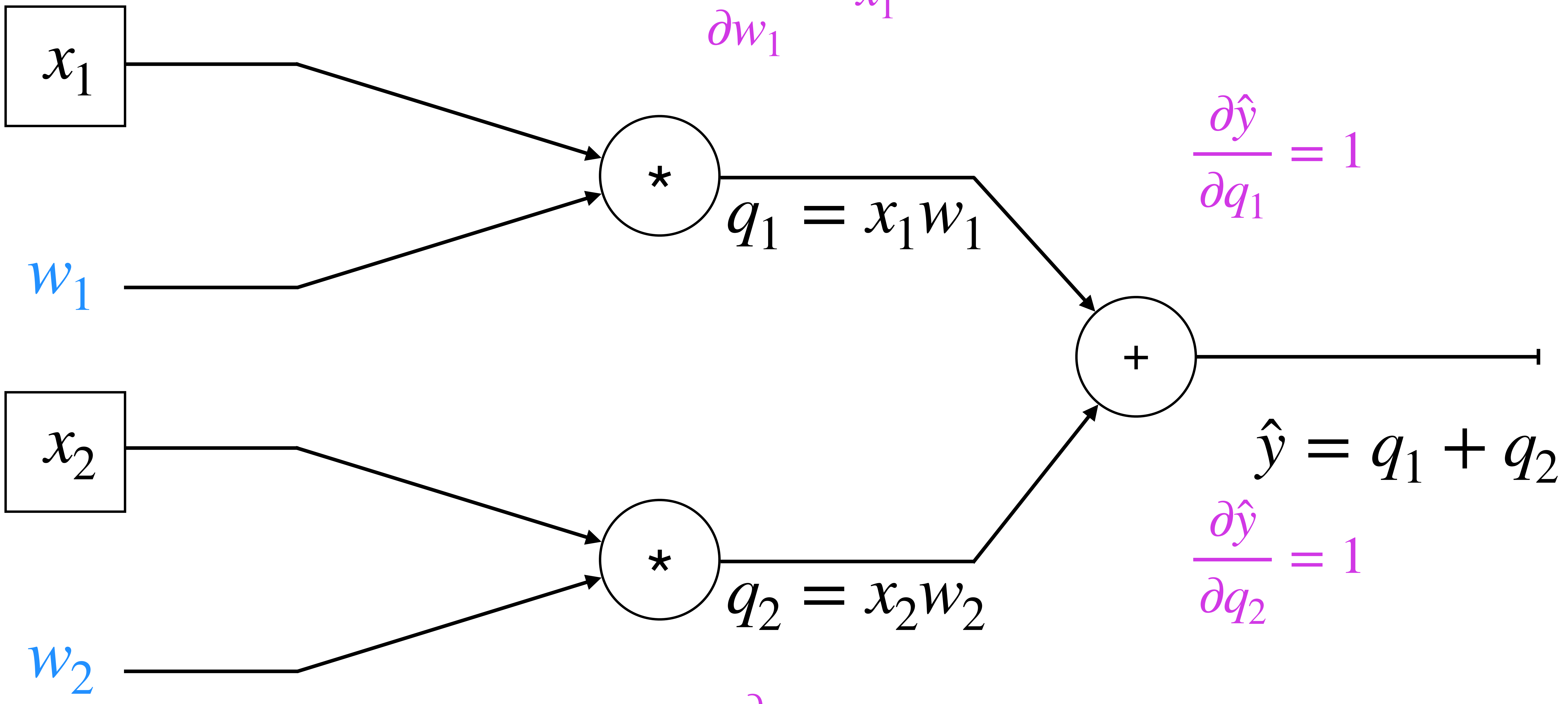
$$N = \frac{1}{1 + e^{-(\sum_{i=1}^{I} x_i w_i + b)}}$$



BACKPROPAGATION

$$w_1 = w_1 - \eta * \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial q_1} \frac{\partial q_1}{\partial w_1}$$

$$\frac{\partial q_1}{\partial w_1} = x_1$$



$$\frac{\partial \hat{y}}{\partial q_1} = 1$$

$$\frac{\partial J}{\partial \hat{y}} = 1$$

$$\frac{\partial \hat{y}}{\partial q_2} = 1$$

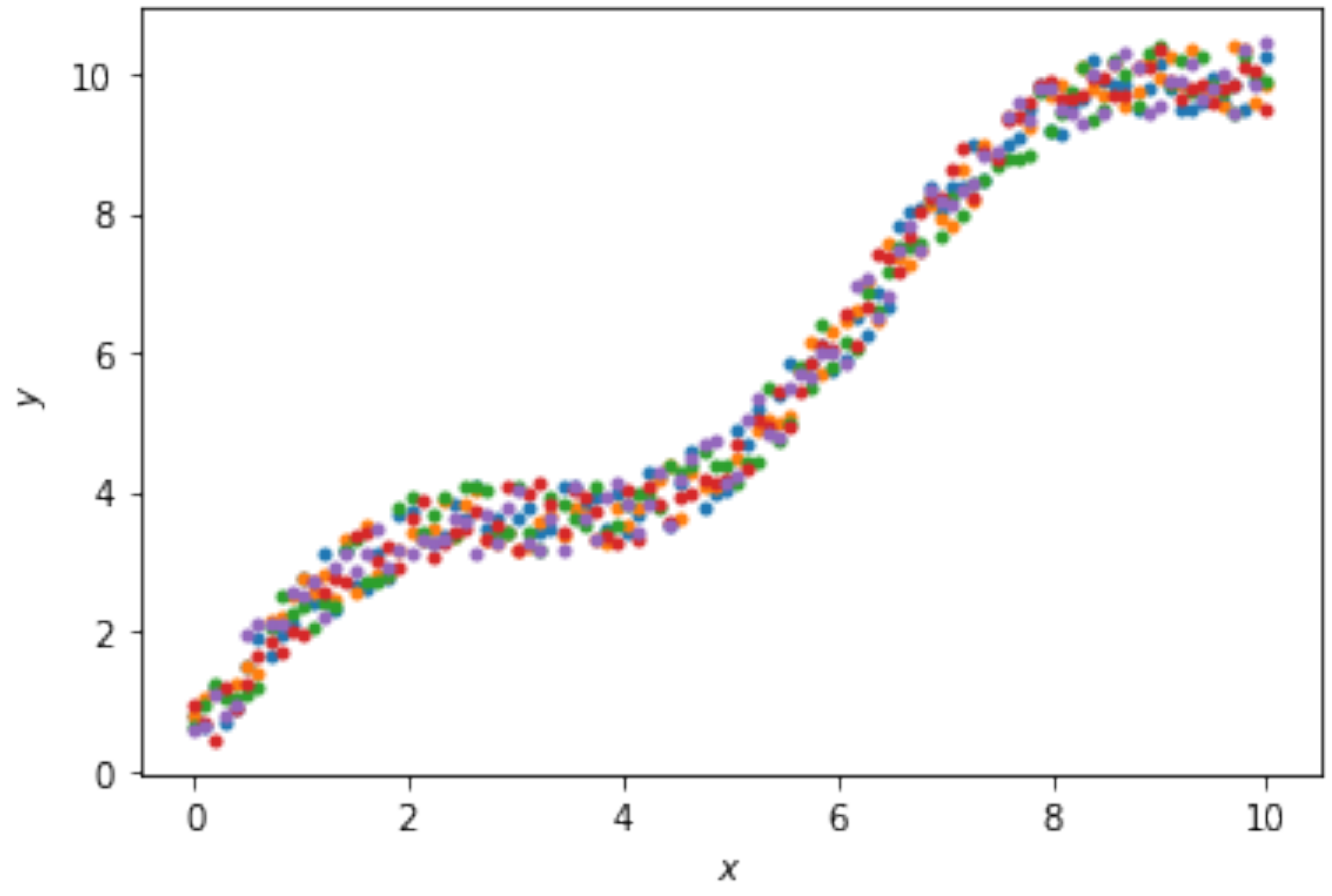
$$\frac{\partial q_2}{\partial w_2} = x_2$$

$$w_2 = w_2 - \eta * \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial q_2} \frac{\partial q_1}{\partial w_2}$$

Loss function

$$J(w) = \hat{y} - y$$

LOSS FUNCTIONS



Loss function

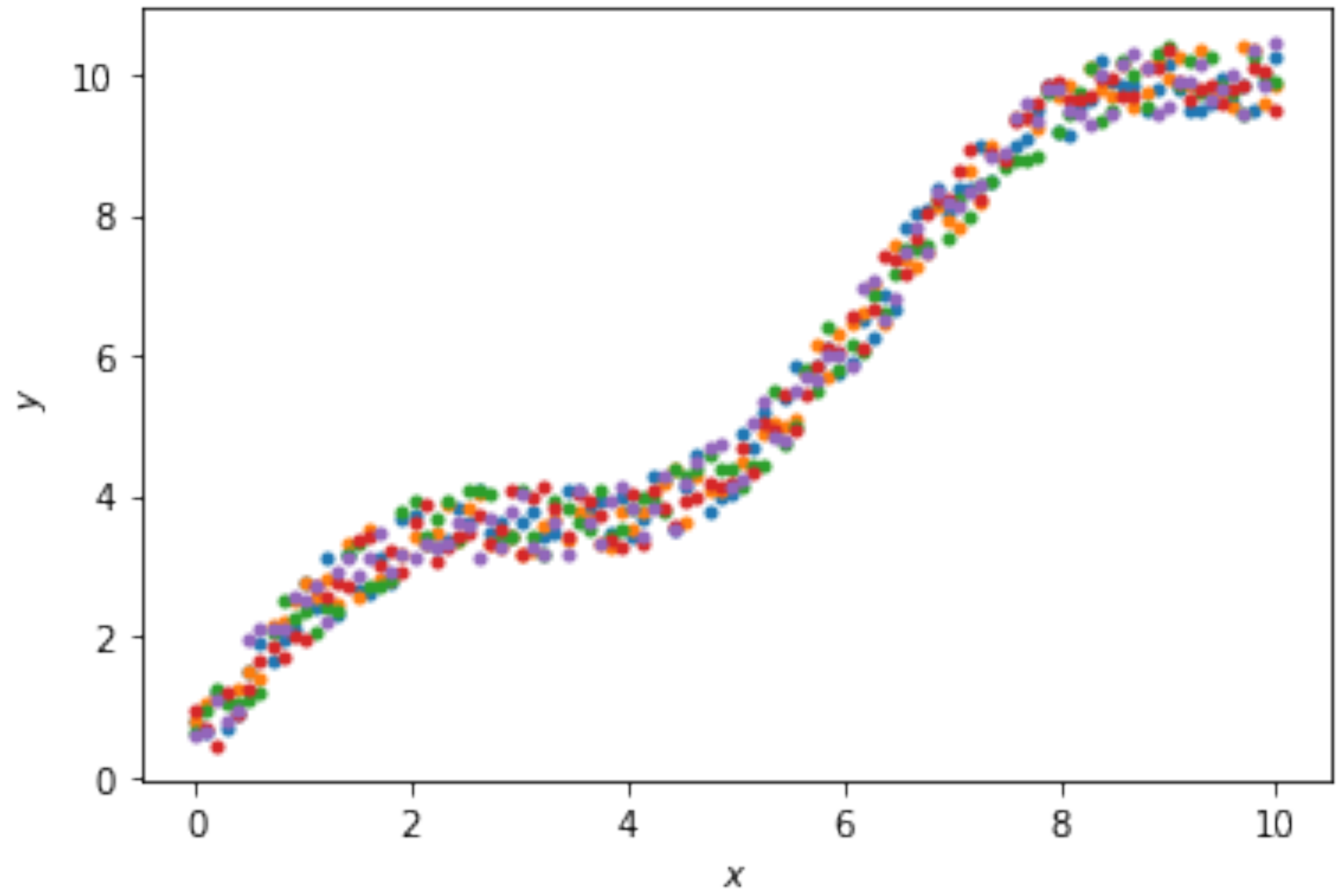
Mean squared error (MSE)

$$J(w) = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2$$

Mean absolute error (MAE)

$$J(w) = \frac{1}{N}\sum_{i=0}^{N}|\hat{y}_i - y_i|$$

LOSS FUNCTIONS



Loss function

Mean squared error

mean

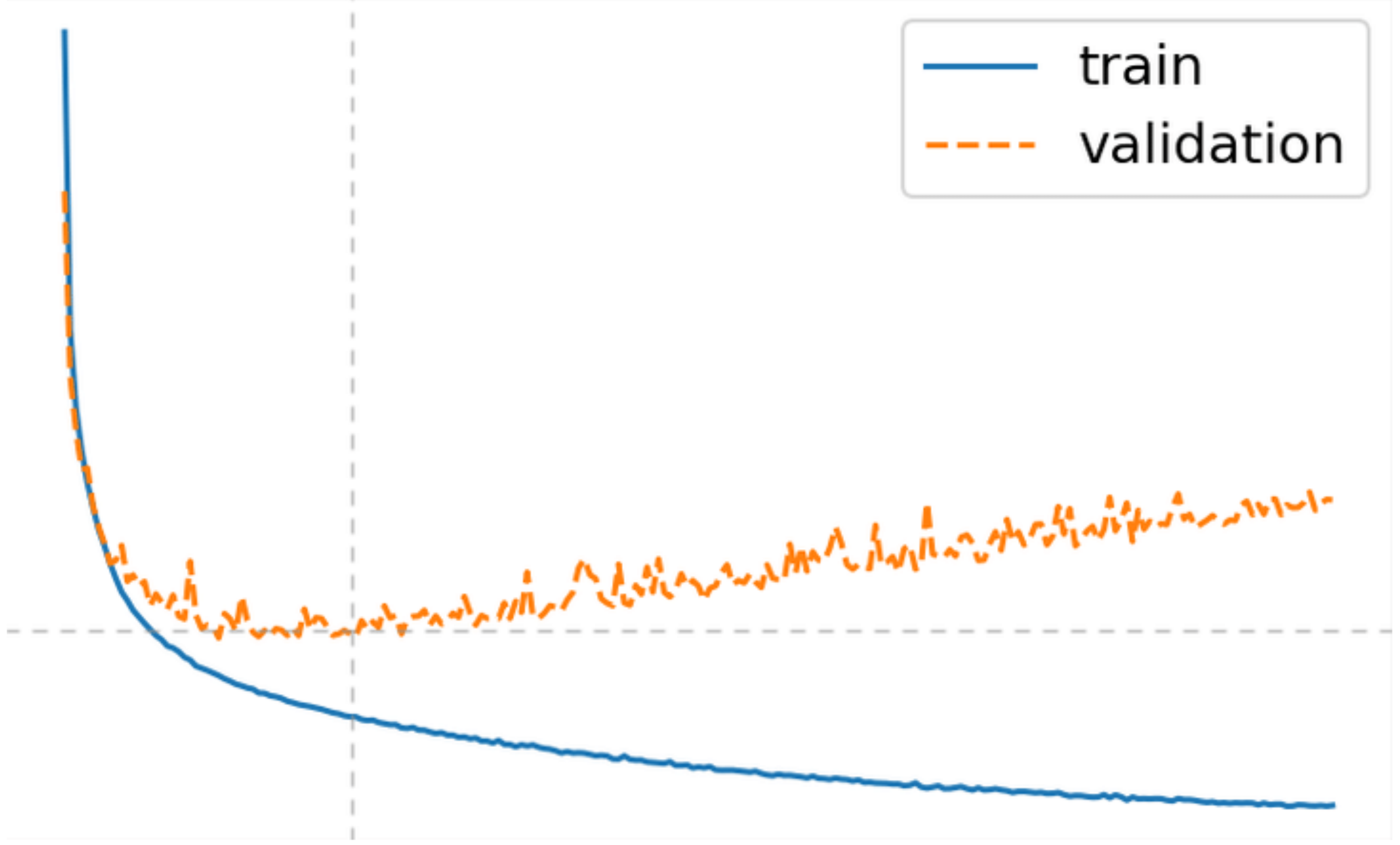
$$J(w) = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2$$

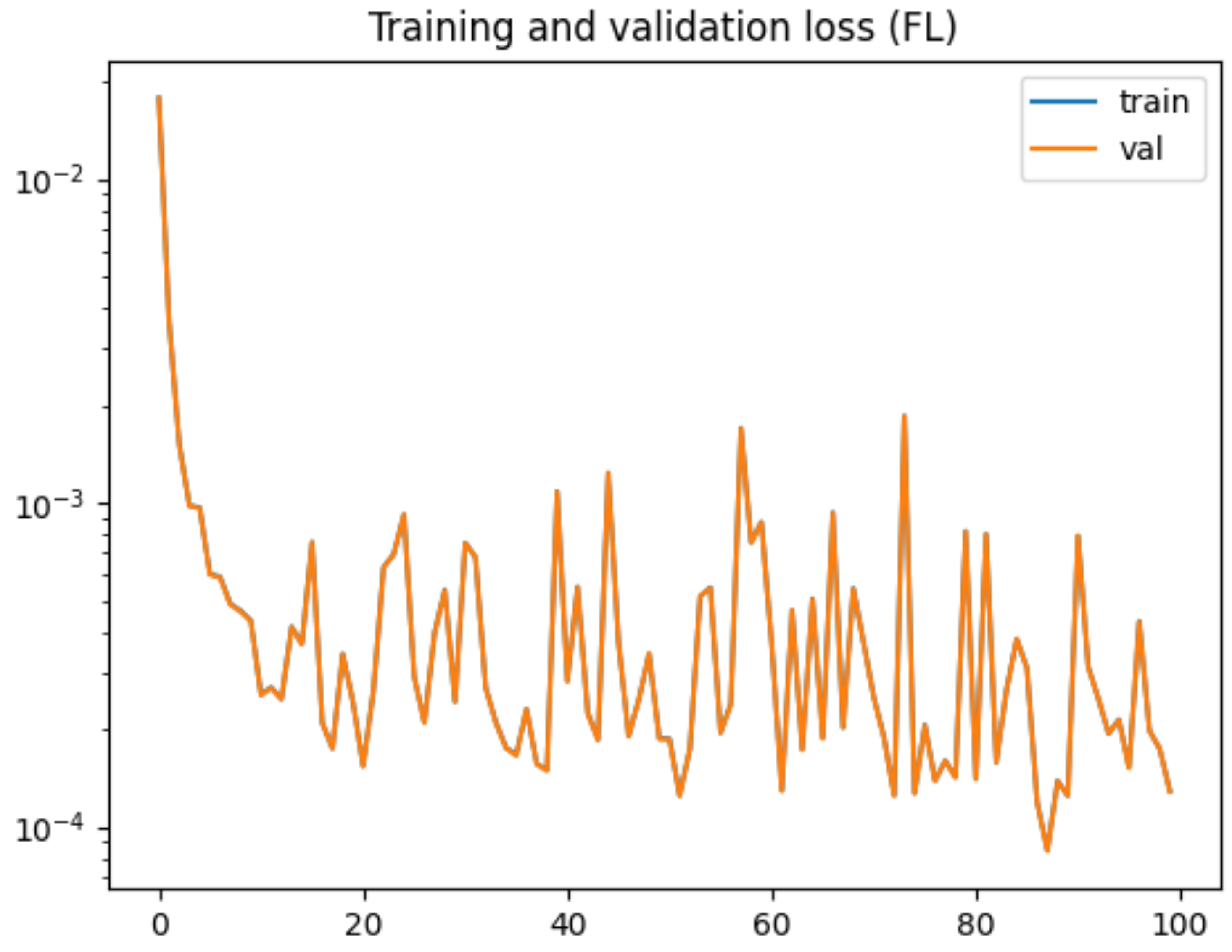
Mean absolute error

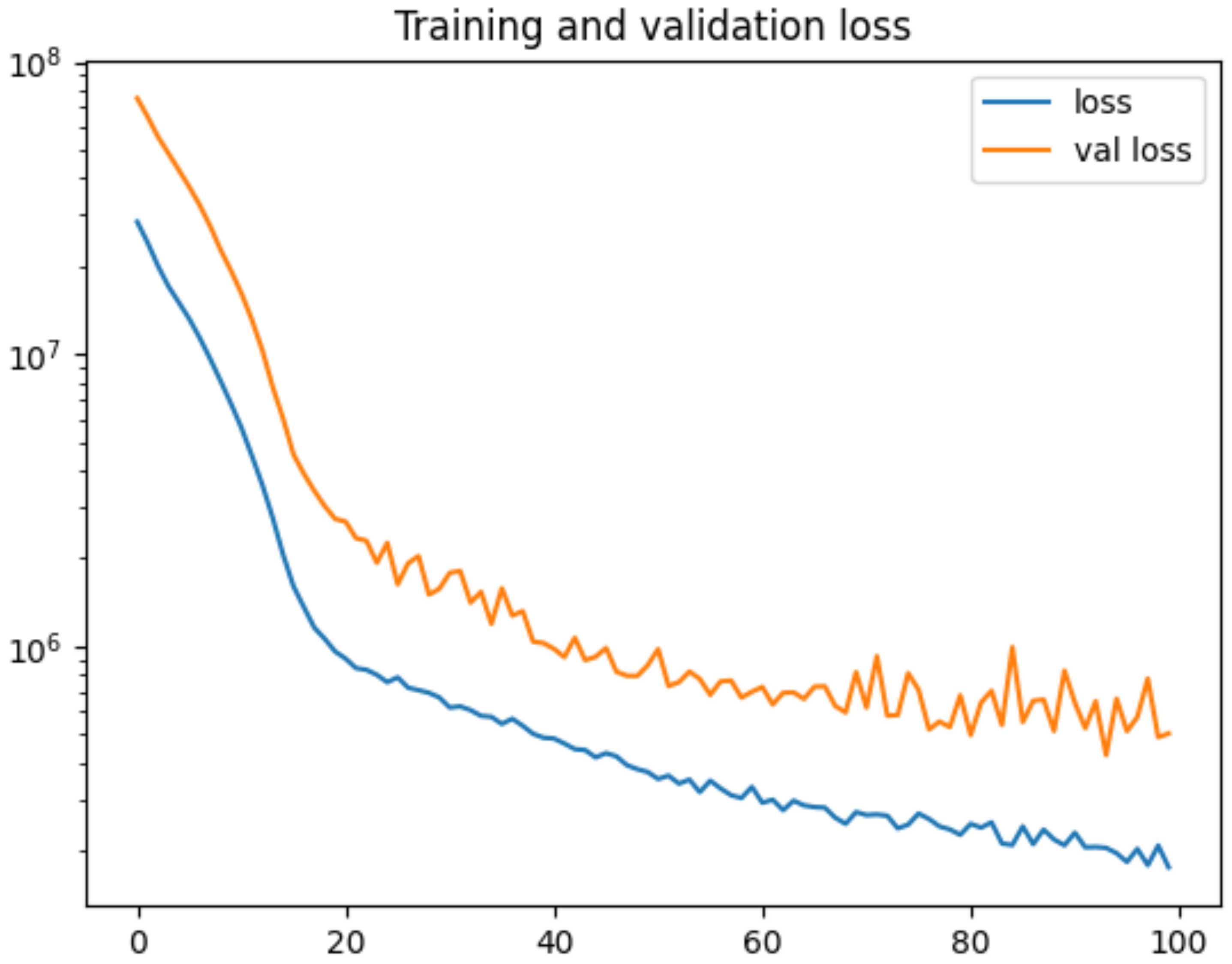
median

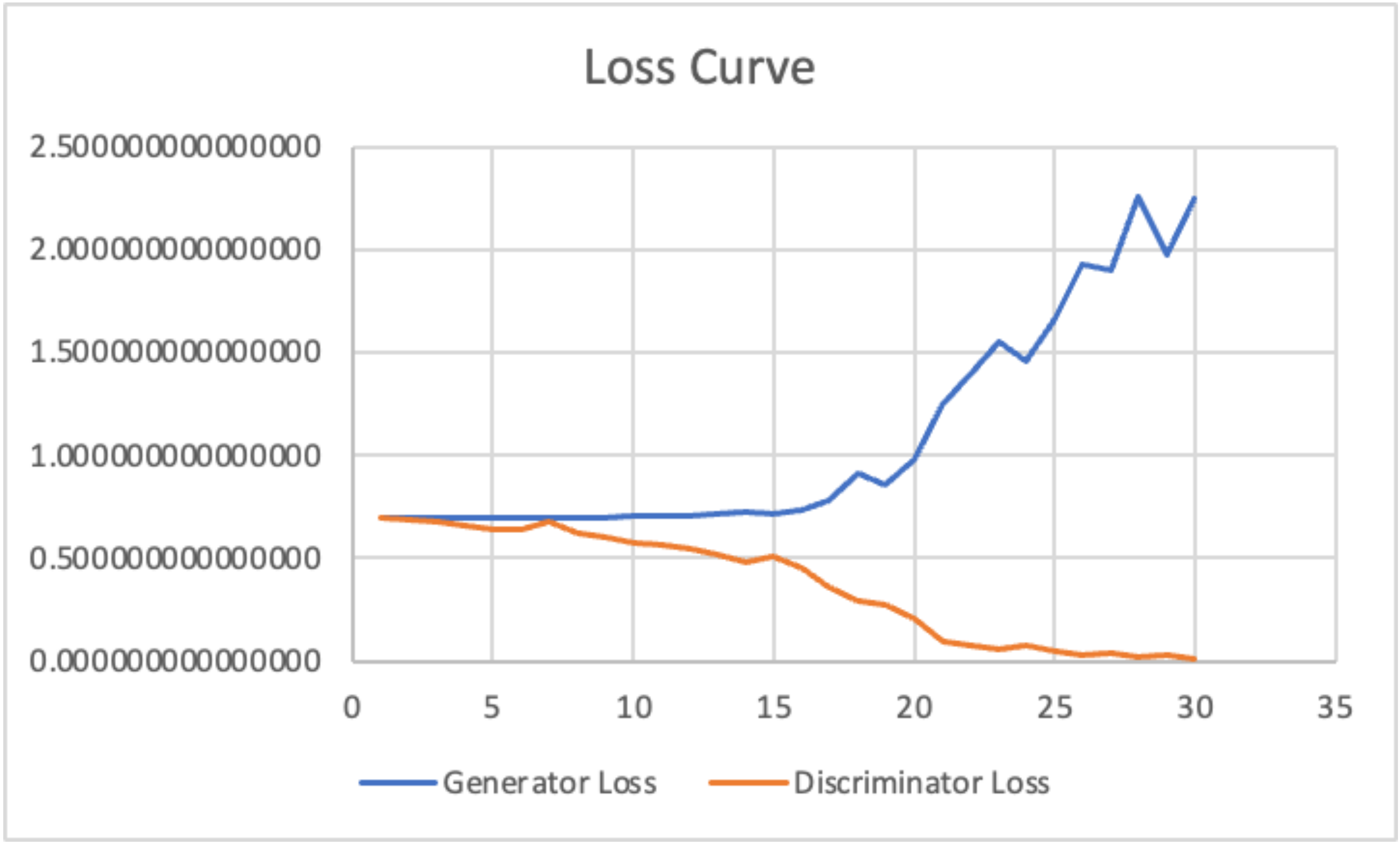
$$J(w) = \frac{1}{N}\sum_{i=0}^{N}|\hat{y}_i - y_i|$$

Learning (loss) curves





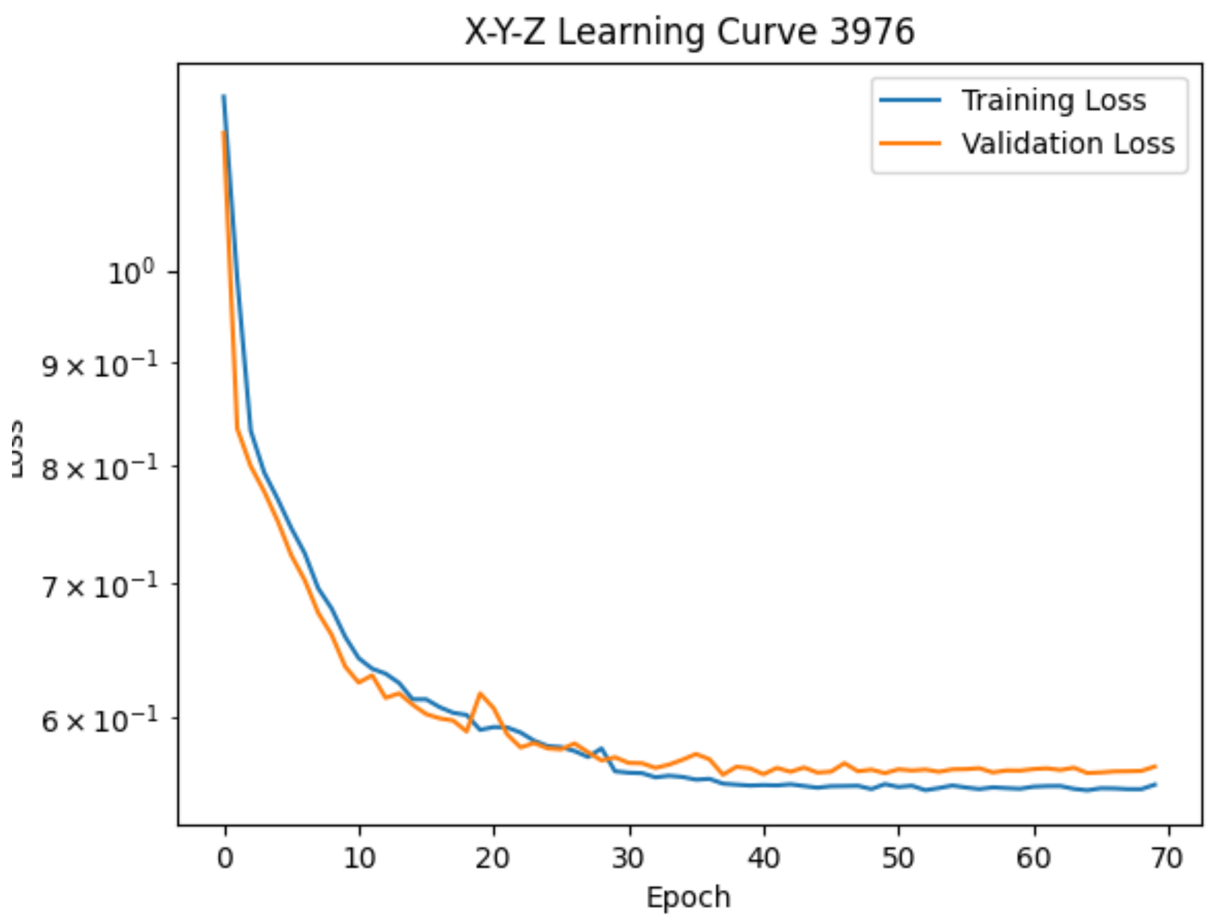


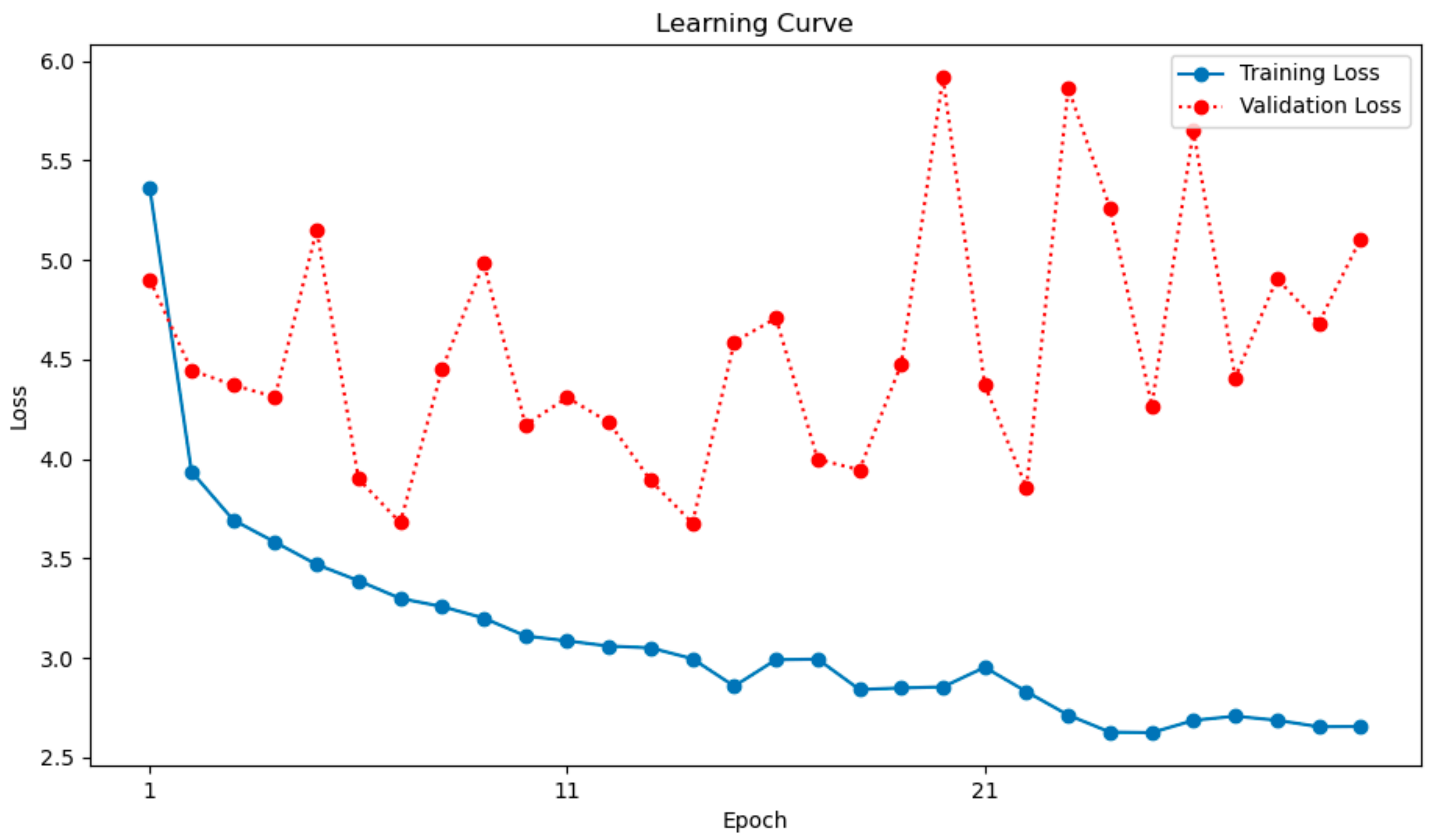


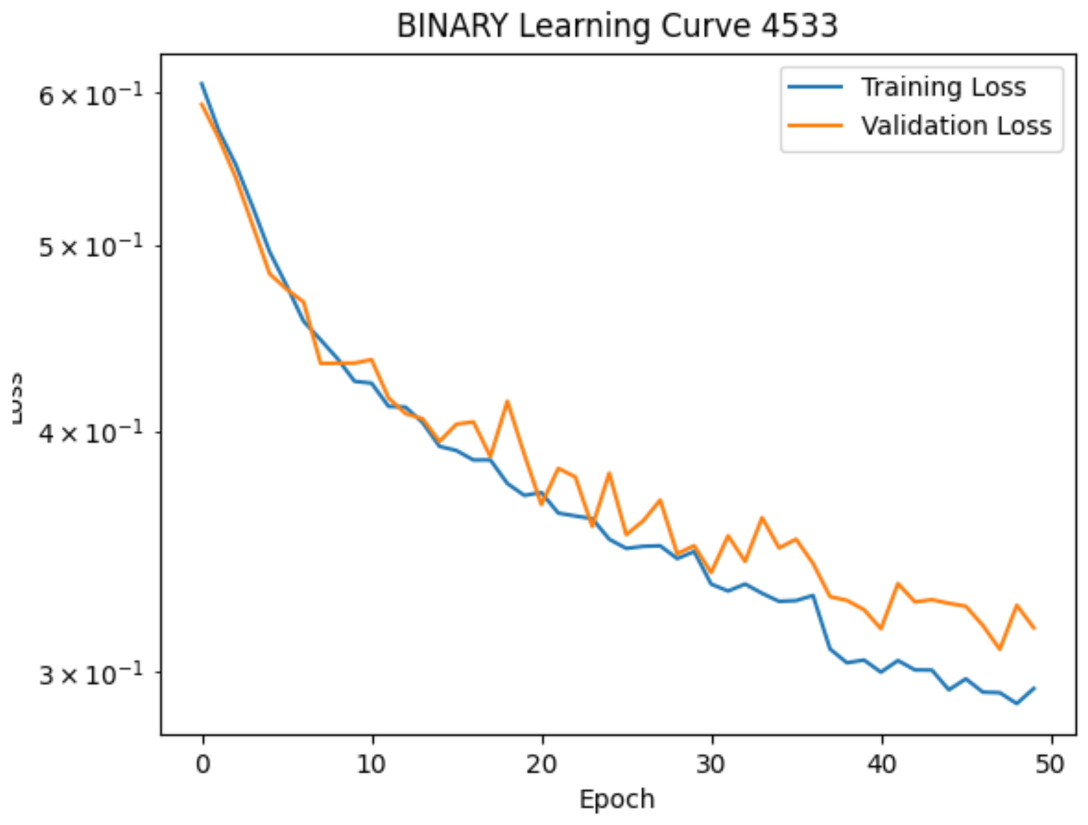
TRAINING

Remember that our goal is NOT to minimize loss on training data!

Learning curves







AUTOMATIC DIFFERENTIATION

TensorFlow

Keras

PyTorch

MODERN NEURAL NETWORK ARCHITECTURES

CHRISTIAN SZEGEDY ET. AL. GOING DEEPER WITH CONVOLUTIONS.

"GoogLeNet network with all the bells and whistles"

MODERN NEURAL NETWORK ARCHITECTURES

CHRISTIAN SZEGEDY ET. AL. GOING DEEPER WITH CONVOLUTIONS.

"GoogLeNet network with all the bells and whistles"

PRACTICAL TIPS FOR TRAINING MODELS

DATA

	Feature 1	Feature 2	Feature 3	Target
Example 1				
Example 2				
Example 3				
Example 4				

NORMALIZATION

- Puts each feature on same scale
- Allows default hyperparamters to be a good starting point
  - learning rate, initialization of weights, etc.
- Options depend on data distribution
  - Standardization: mean: 0 stdev: 1
  - Min-max: [0,1]

DATA

	Feature 1	Feature 2	Feature 3	Target
Example 1				
Example 2				
Example 3				
Example 4				

ENCODING

- Non-numeric data
- Class-based features:
  - One-hot encoding: $2 \rightarrow [0\ 1]$
  - When classes do not have sequential meaning: ✅ cars vs dogs vs plants ❌ months

BUILDING AND TRAINING MODELS

TRAINING

- The most challenging part of machine learning is gaining the experience for tuning models well.
- We will work on this skill!

COMMUNITY

- Each of you arrived here with your own backgrounds, specialty, and path in life
- Your experience and expertise are valuable here, no matter what it is
- If the activity is within your background, help others!
- If you are totally (or a little) lost, ask for help!
- It is our shared goal to have **each** of us leave with some new skill/knowledge/understanding