

Intro to Deep Learning

Boris Zubarev



@bobazooba

Course Goals

Grading

Result = 0.4 * (mean(homeworks)) + 0.4 * project + 0.2 * exam

Homeworks

Result = $0.4 * \text{mean}(\text{homeworks}) + 0.4 * \text{project} + 0.2 * \text{exam}$

- numpy n-layer NN
- PyTorch word2vec
- Classification
- GRU implementation
- Language modeling
- Attention
- Transfer learning

Homeworks

Result = $0.4 * \text{mean}(\text{homeworks}) + 0.4 * \text{project} + 0.2 * \text{exam}$

- numpy n-layer NN
- PyTorch word2vec

HW1

- Classification
- GRU implementation

HW2

- Language modeling
- Attention
- Transfer learning

HW3

Project

Result = $0.4 * \text{mean}(\text{homeworks}) + 0.4 * \text{project} + 0.2 * \text{exam}$

- Must include DL
- List of projects
- Your own
- Paper implementation
- Competition
- Max team size = 2 (3)
- Choose wisely

Project

Result = $0.4 * \text{mean}(\text{homeworks}) + 0.4 * \text{project} + 0.2 * \text{exam}$

- Must include DL
- List of projects
- Your own
- Paper implementation
- Competition
- Max team size = 2 (3)
- Choose wisely

Exam

Result = 0.4 * (mean(homeworks)) + 0.4 * project + 0.2 * exam

- Test
- ~15 questions from course lectures

Syllabus

- Deep Learning
- Word embeddings
- Convolution NN
- Tricks in DL
- Engineering in DL
- Recurrent NN
- Sequence tagging

Syllabus

- Deep Learning
- Word embeddings
- Convolution NN
- Tricks in DL
- Engineering in DL
- Recurrent NN
- Sequence tagging
- Language models
- Ranking models
- Sequence2sequence
- Attention
- Transformer
- Transfer learning
- Dialogue systems

Framework

Simple

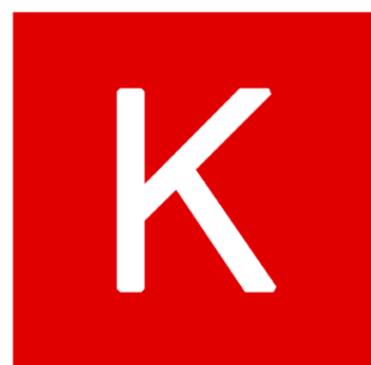
Hard



Framework

Simple

Hard



Keras



Framework

Simple

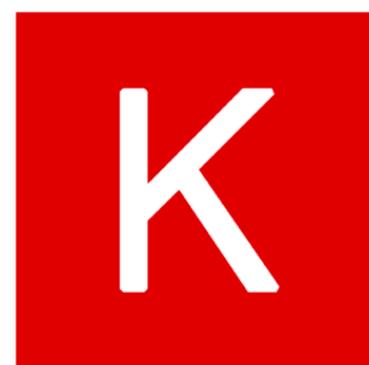
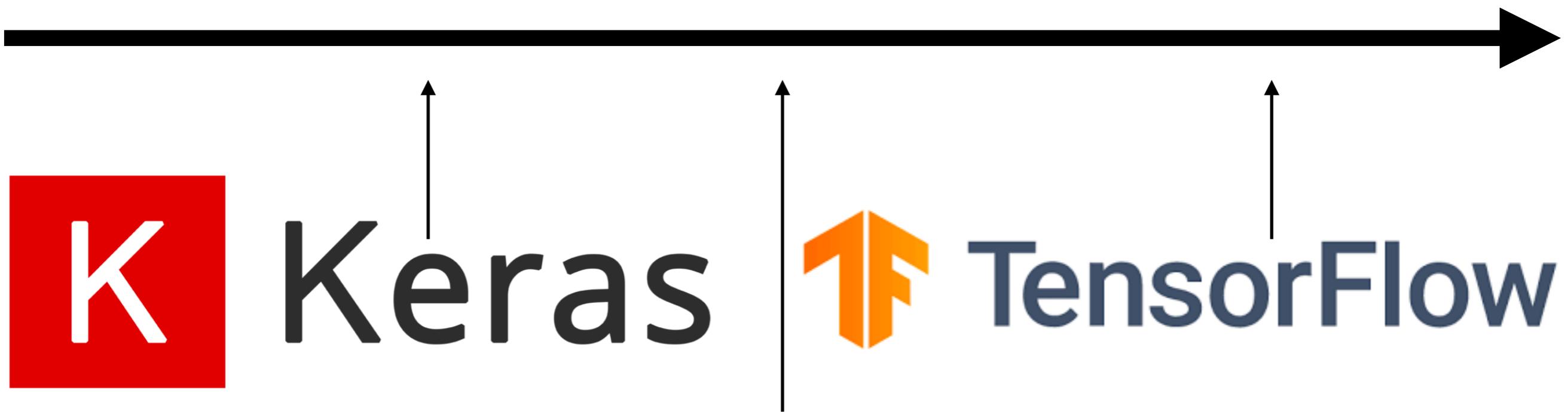
Hard



Framework

Simple

Hard



Keras



TensorFlow

PyTorch

Framework

Simple

Hard



Deep Learning

Classic Machine Learning

Deep Learning

Classic Machine Learning

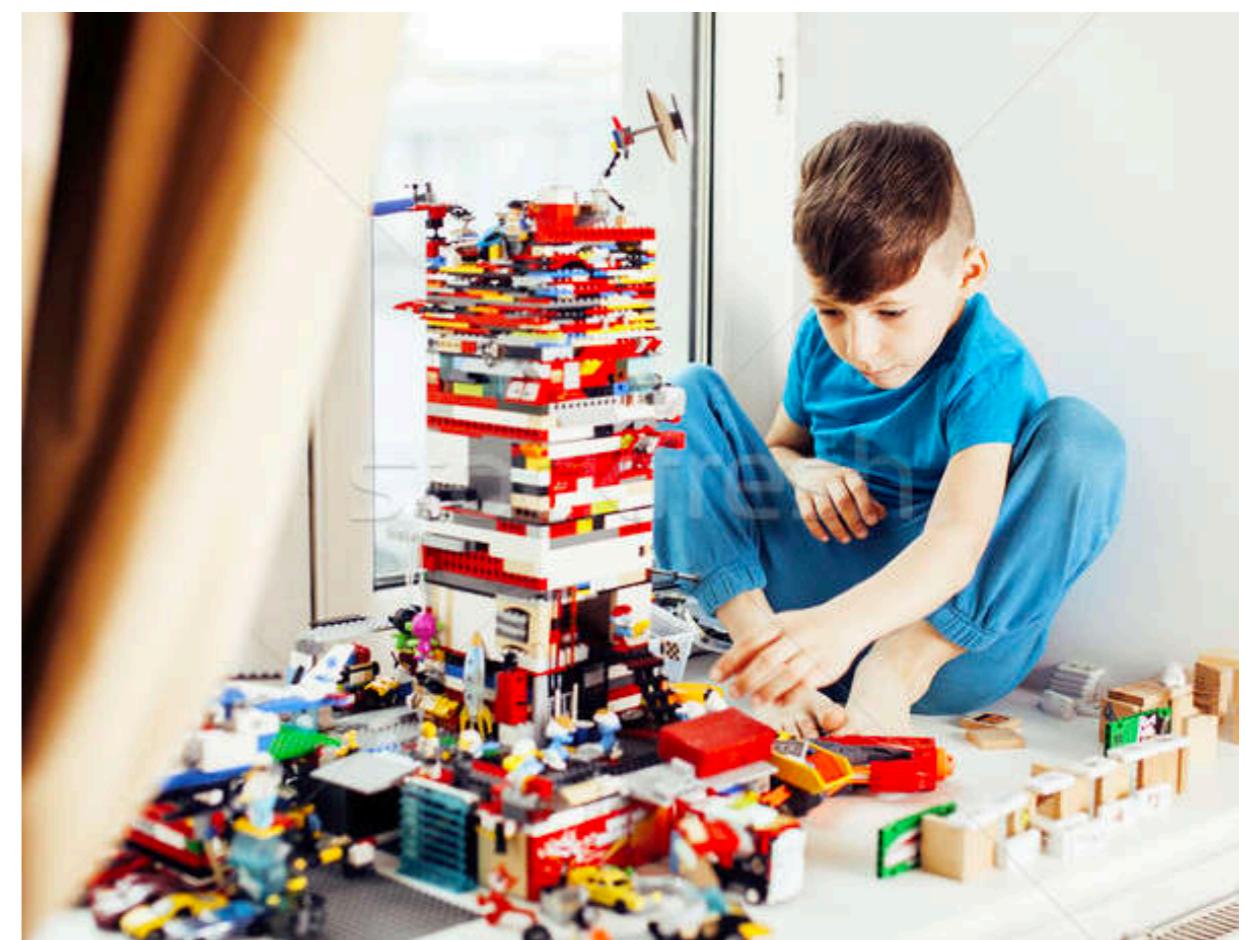
Deep Learning



Classic Machine Learning



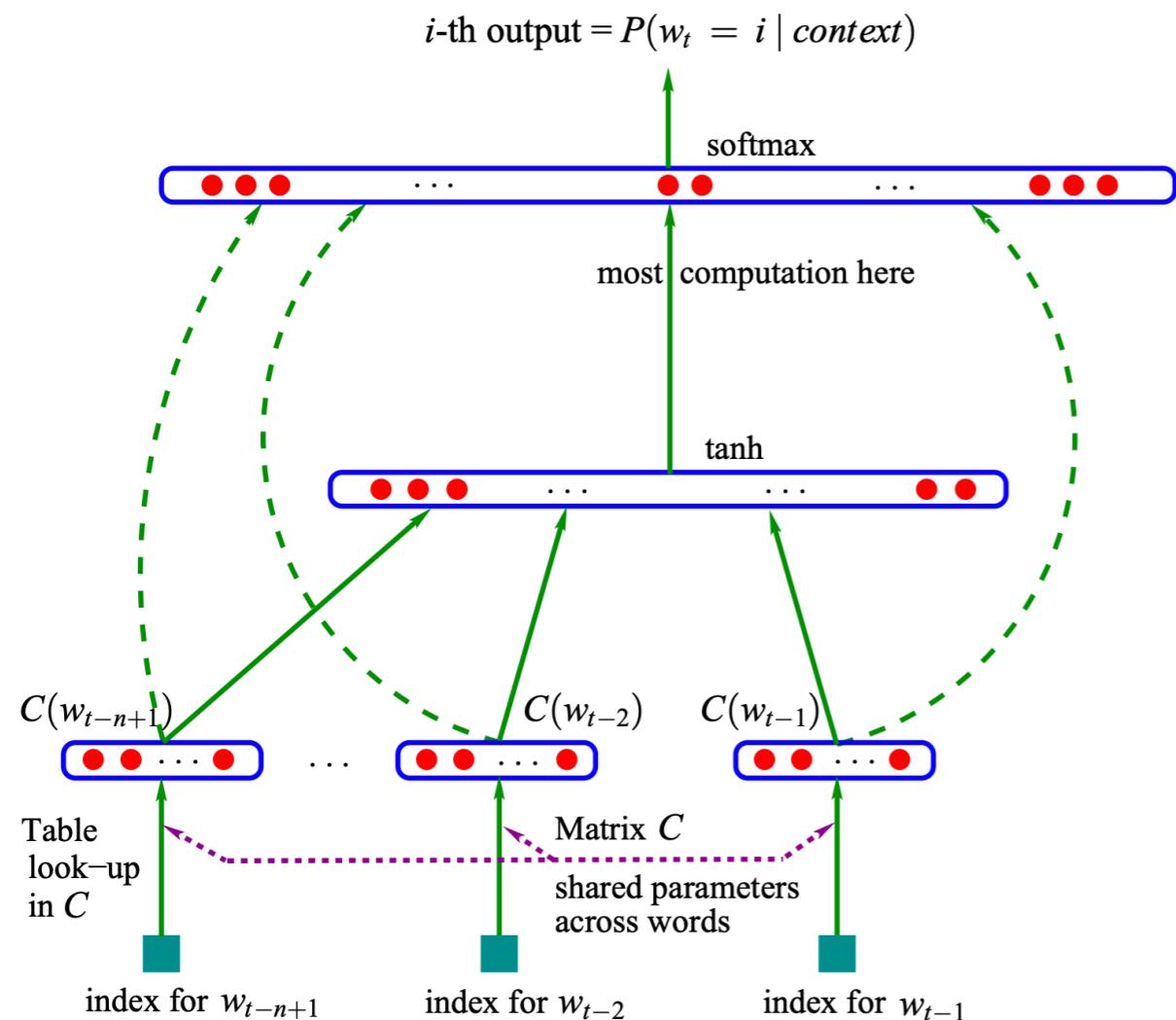
Deep Learning



Deep Learning

Neural Probabilistic Language Model

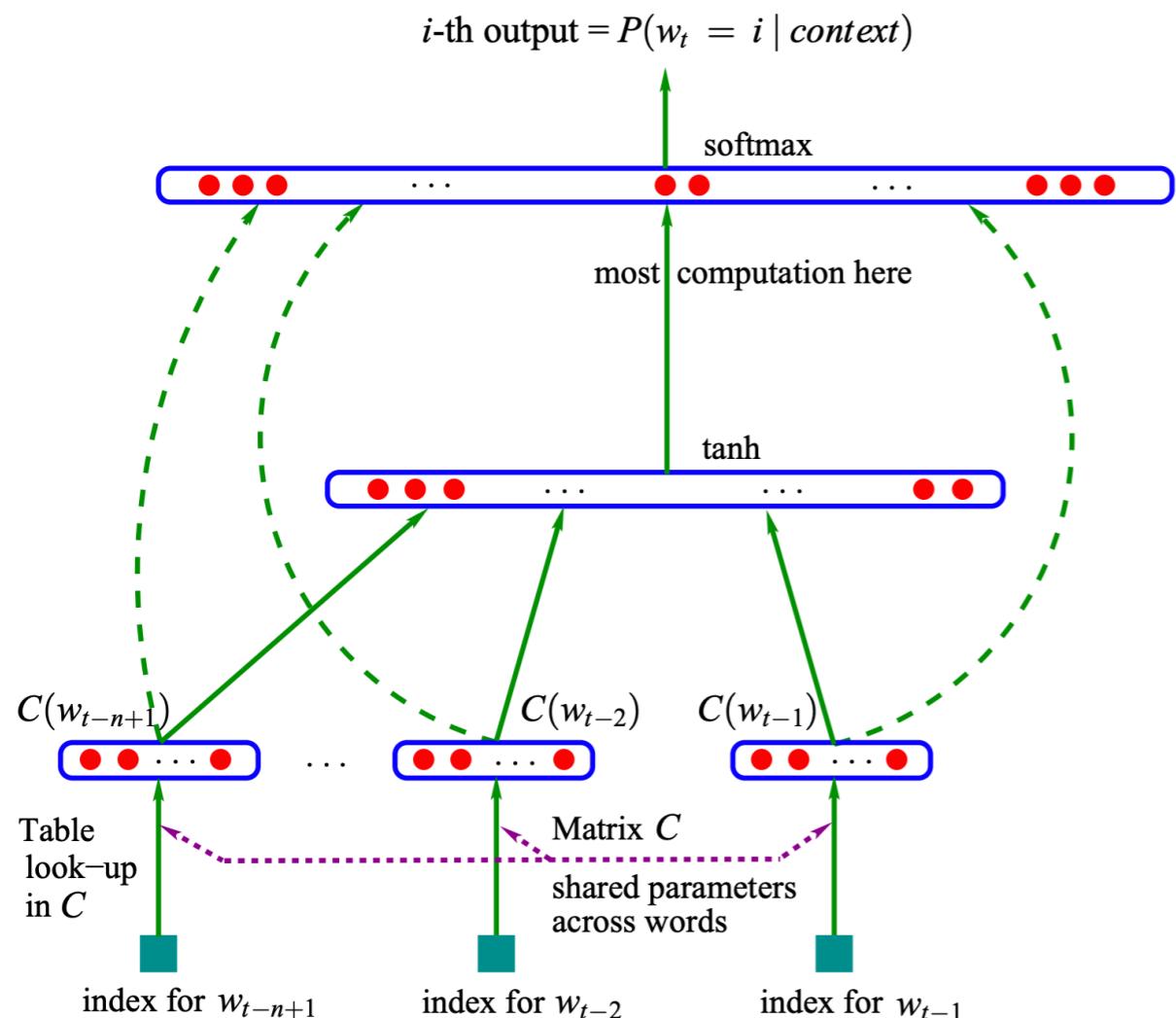
2003



Deep Learning

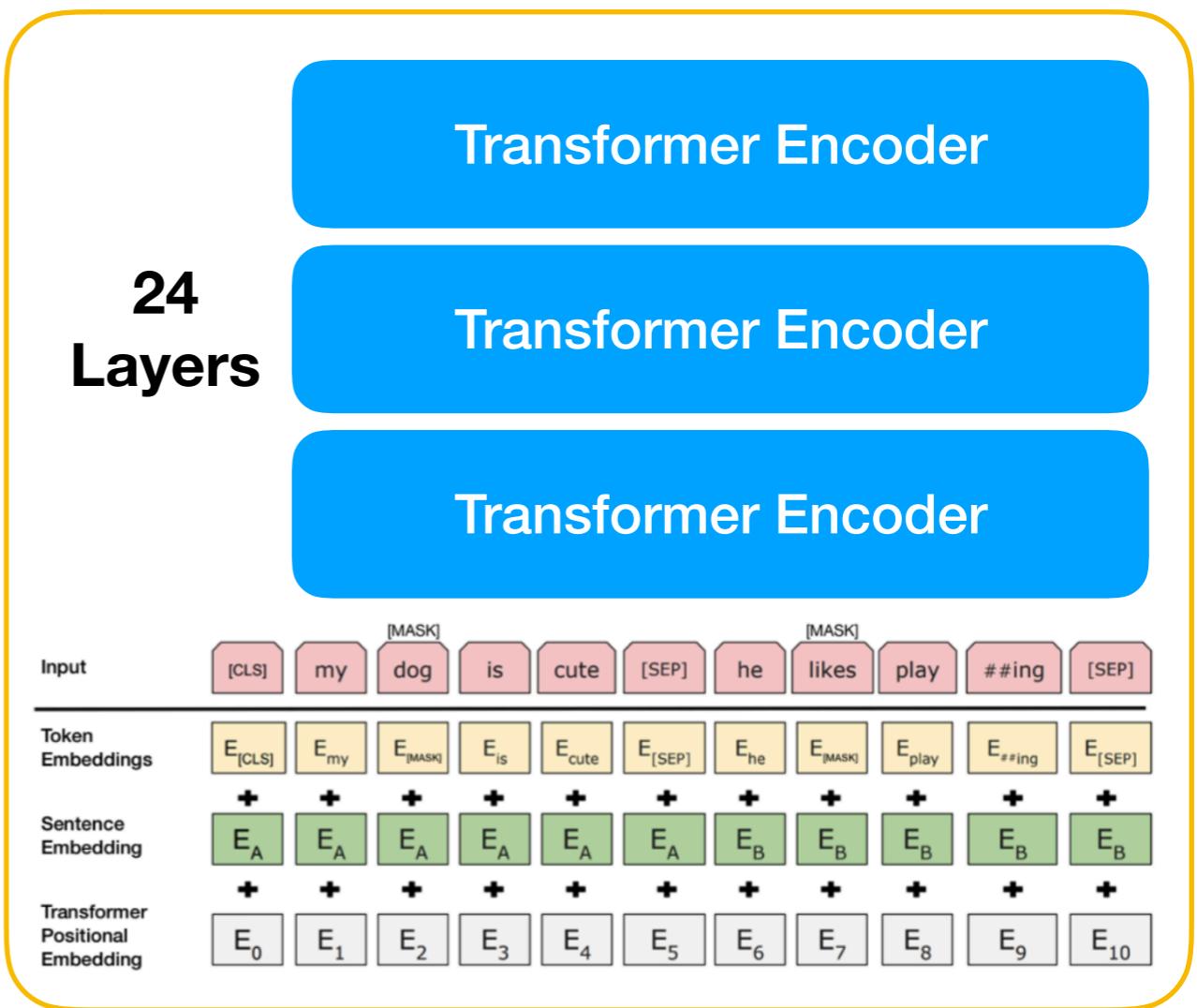
Neural Probabilistic Language Model

2003



BERT

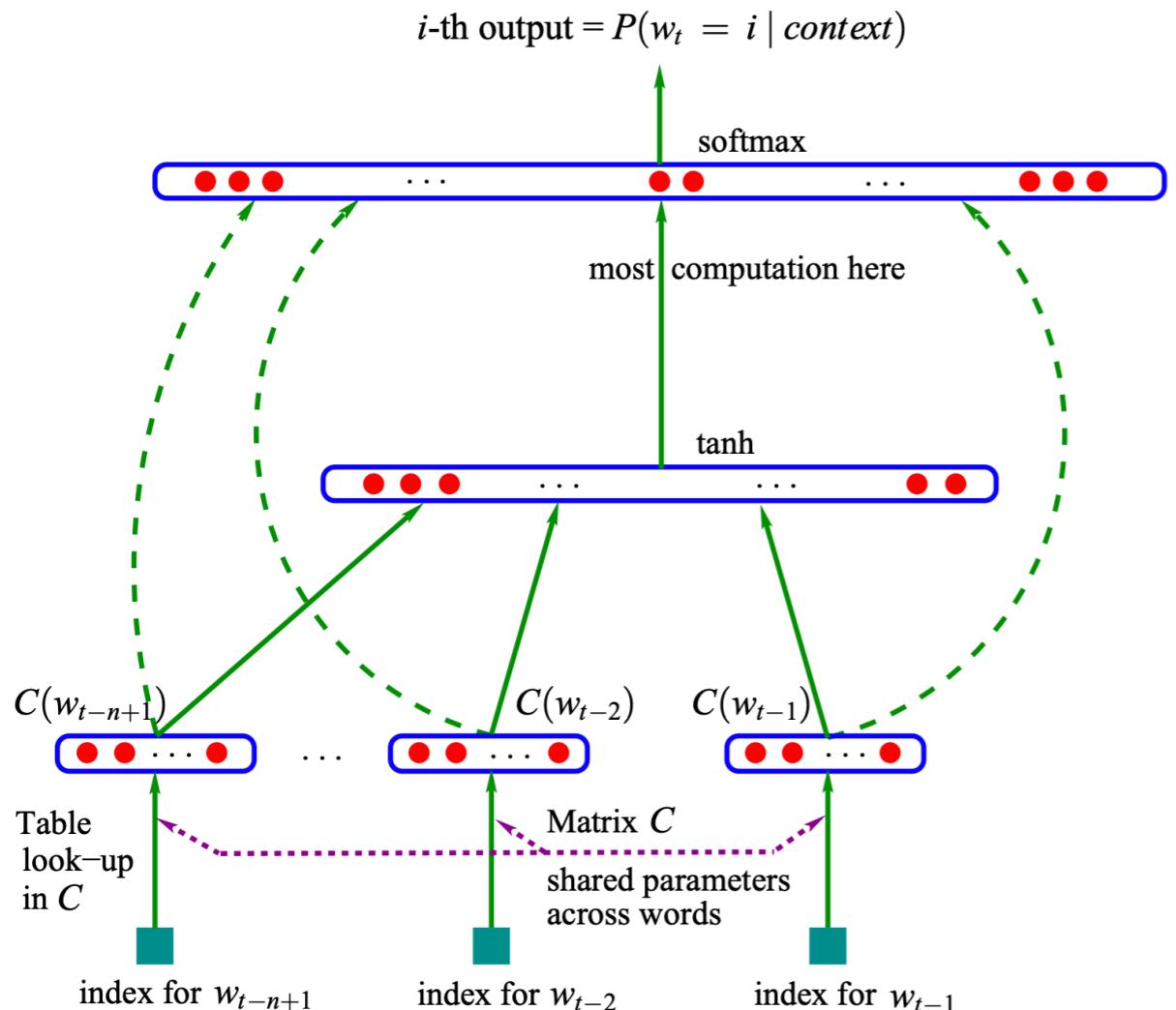
Q42018



Deep Learning

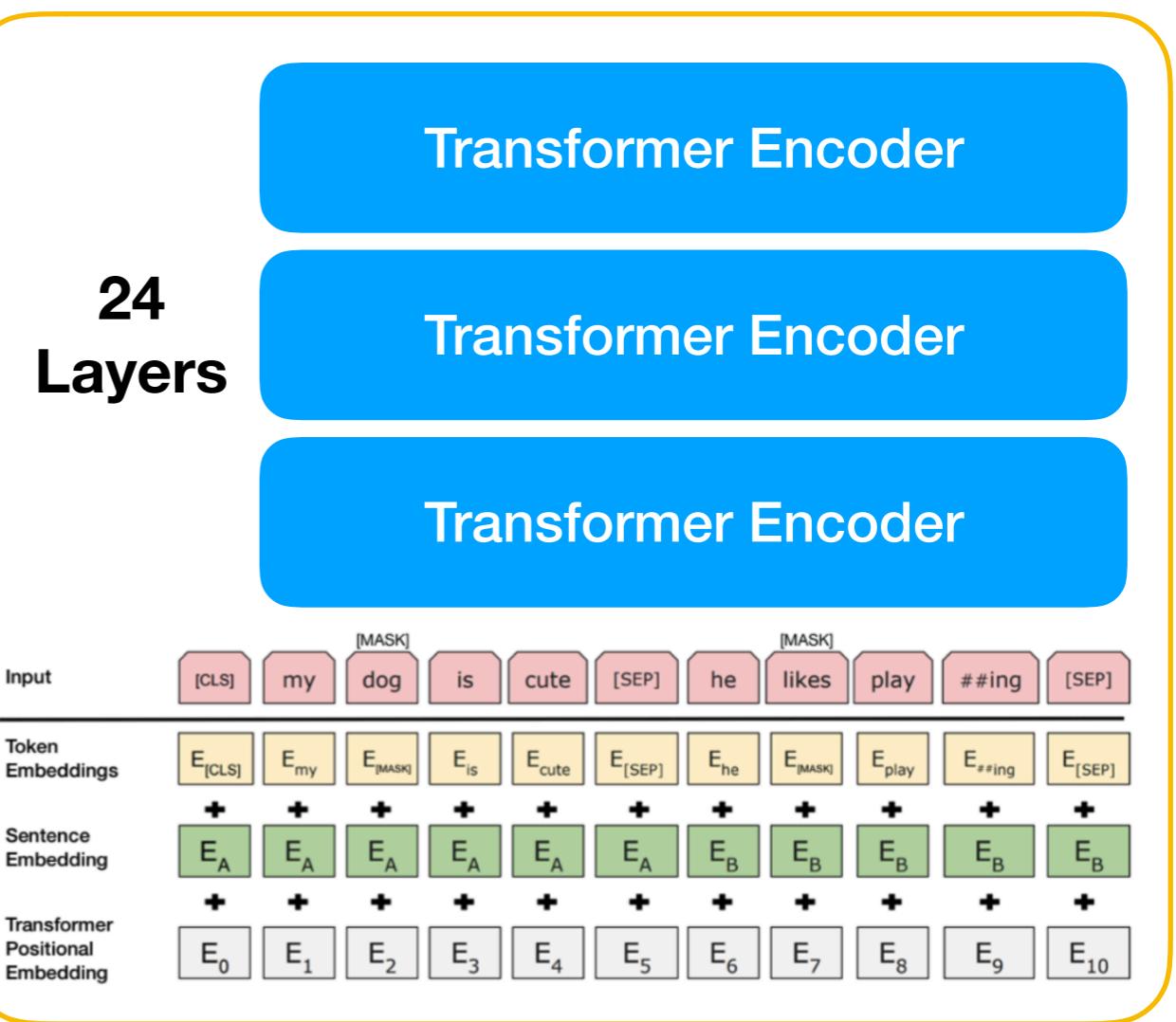
Neural Probabilistic Language Model

2003

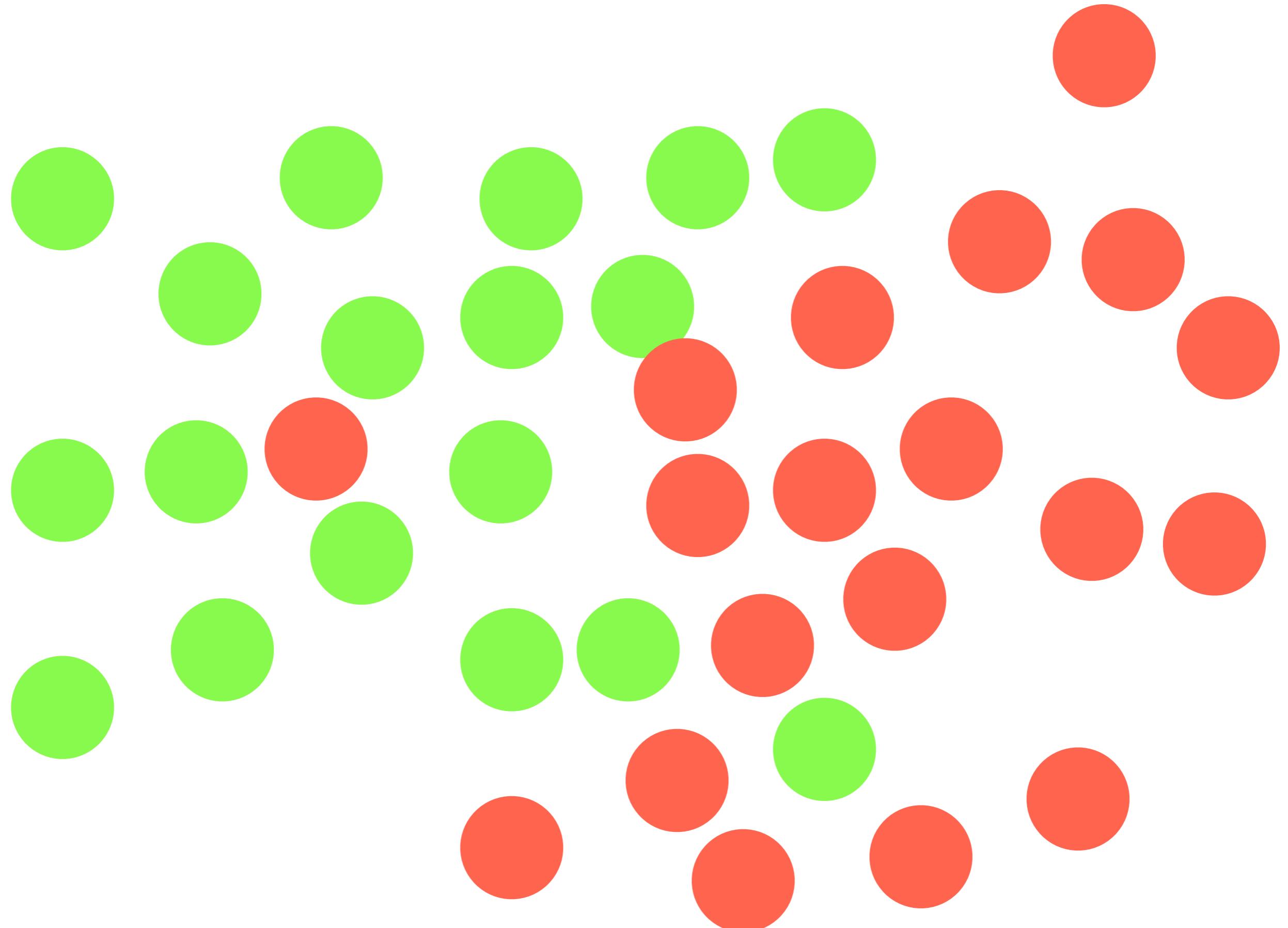


BERT

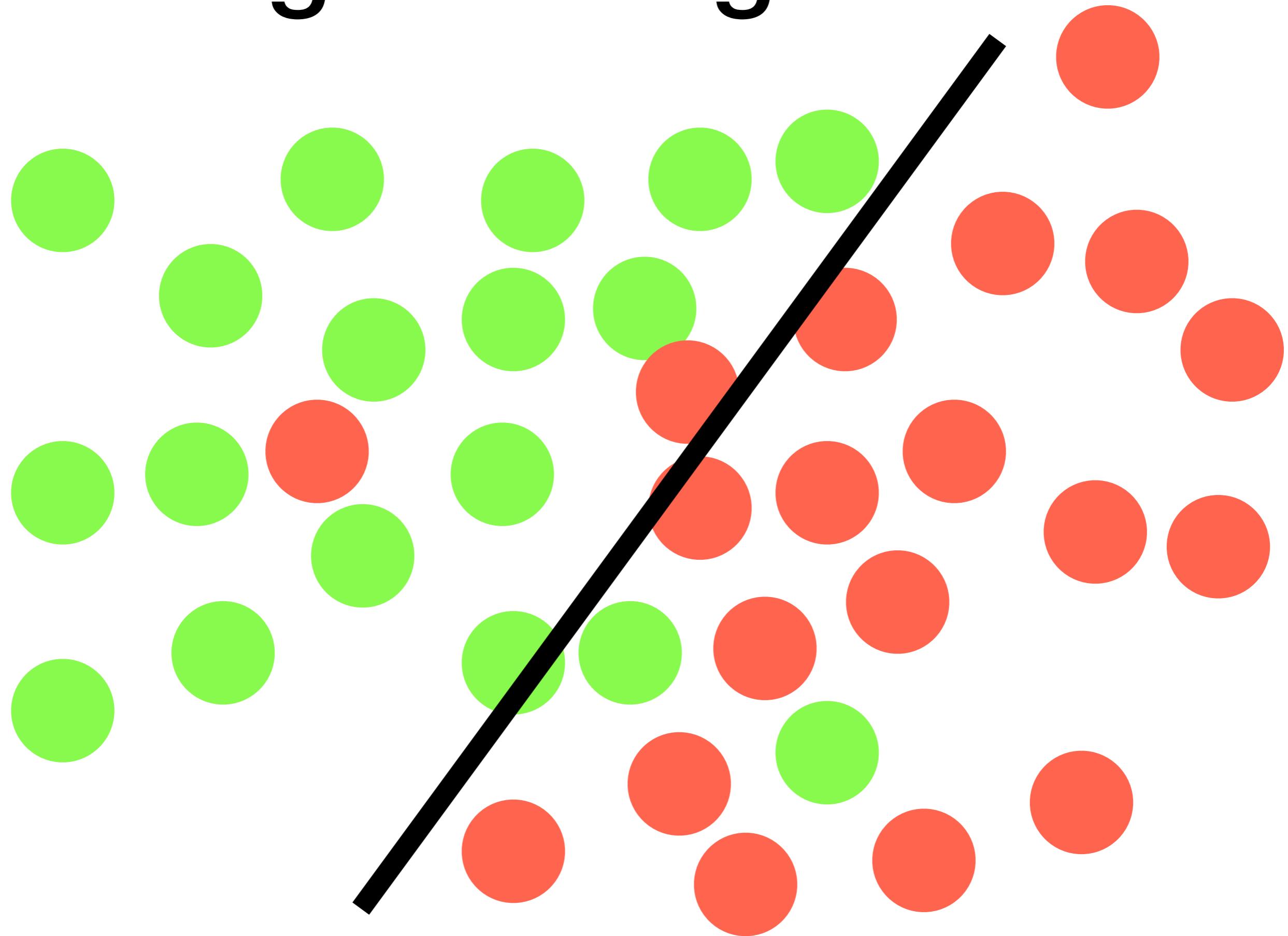
Q42018



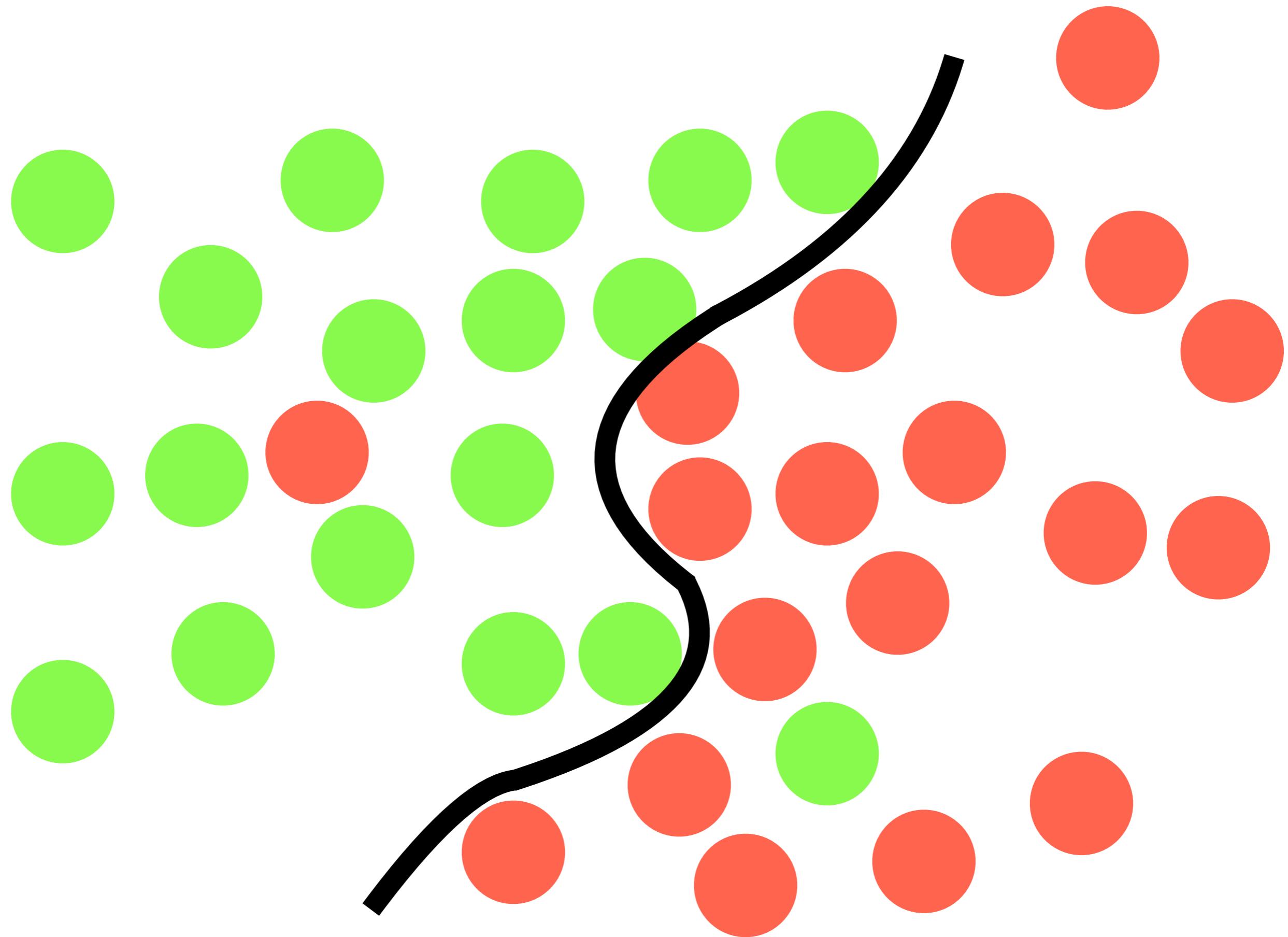
ML Recap



Logistic Regression



Neural Networks



Logistic Regression

Inference

(len(data), 3)

Logistic Regression

Inference

```
array([[0.61417946, 0.14738907, 0.74942813],  
       [0.54257148, 0.7955858 , 0.08140681],  
       [0.31717539, 0.74551499, 0.11464107],  
       [0.90179514, 0.17979926, 0.52133968],  
       [0.63032386, 0.25433298, 0.17785144],  
       [0.45850347, 0.92489706, 0.29492176],  
       [0.38685861, 0.99961577, 0.89145302],  
       [0.68854165, 0.87301941, 0.58840101],  
       [0.39667669, 0.7582801 , 0.46032164],  
       [0.4046151 , 0.53386007, 0.49388535]])  
(len(data), 3)
```

Logistic Regression

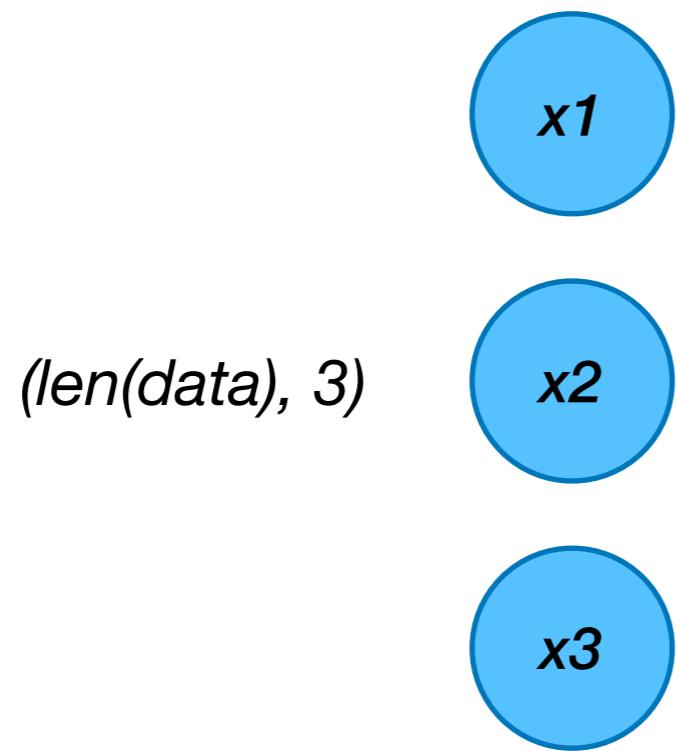
Inference

	0	1	2
0	0.614179	0.147389	0.749428
1	0.542571	0.795586	0.081407
2	0.317175	0.745515	0.114641
3	0.901795	0.179799	0.521340
4	0.630324	0.254333	0.177851
5	0.458503	0.924897	0.294922
6	0.386859	0.999616	0.891453
7	0.688542	0.873019	0.588401
8	0.396677	0.758280	0.460322
9	0.404615	0.533860	0.493885

(len(data), 3)

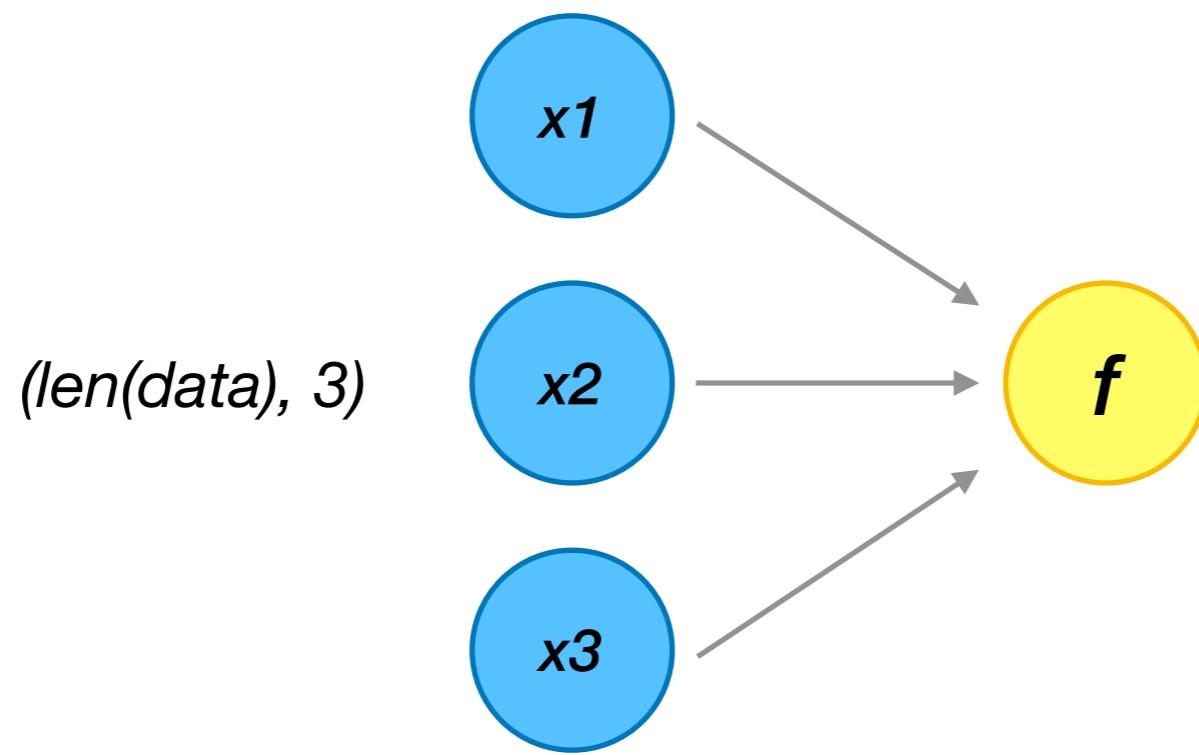
Logistic Regression

Inference



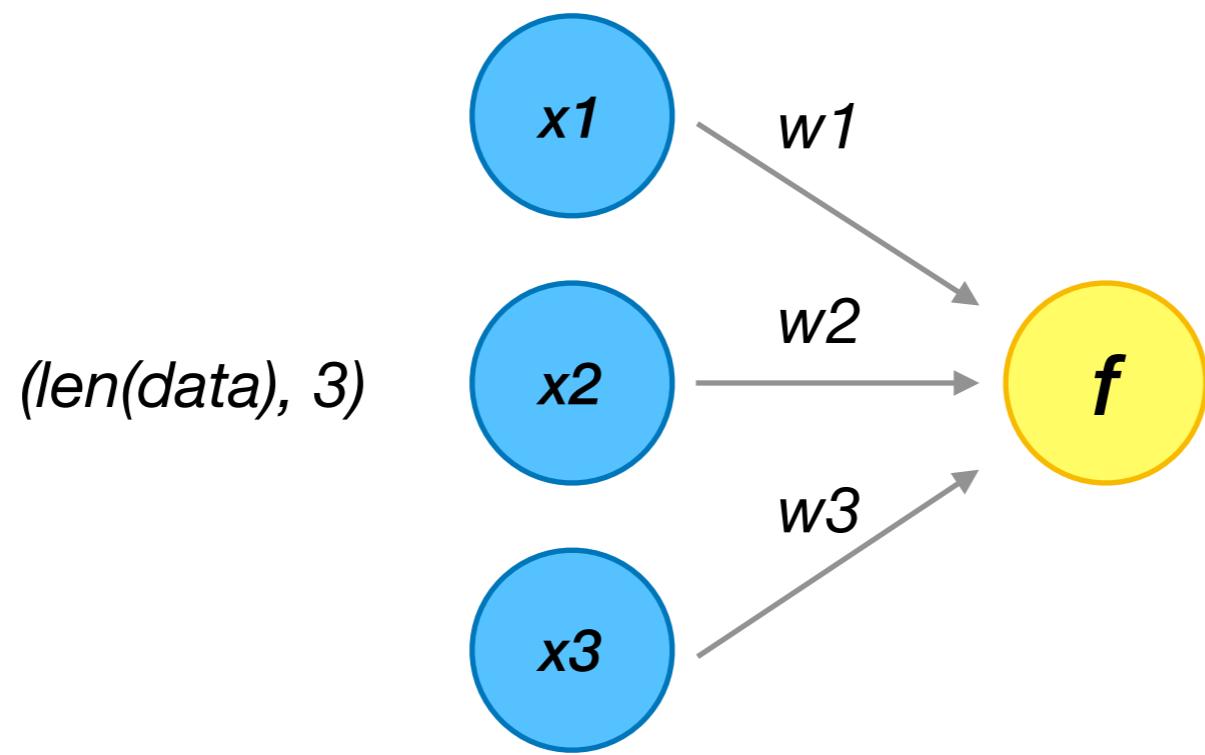
Logistic Regression

Inference



Logistic Regression

Inference



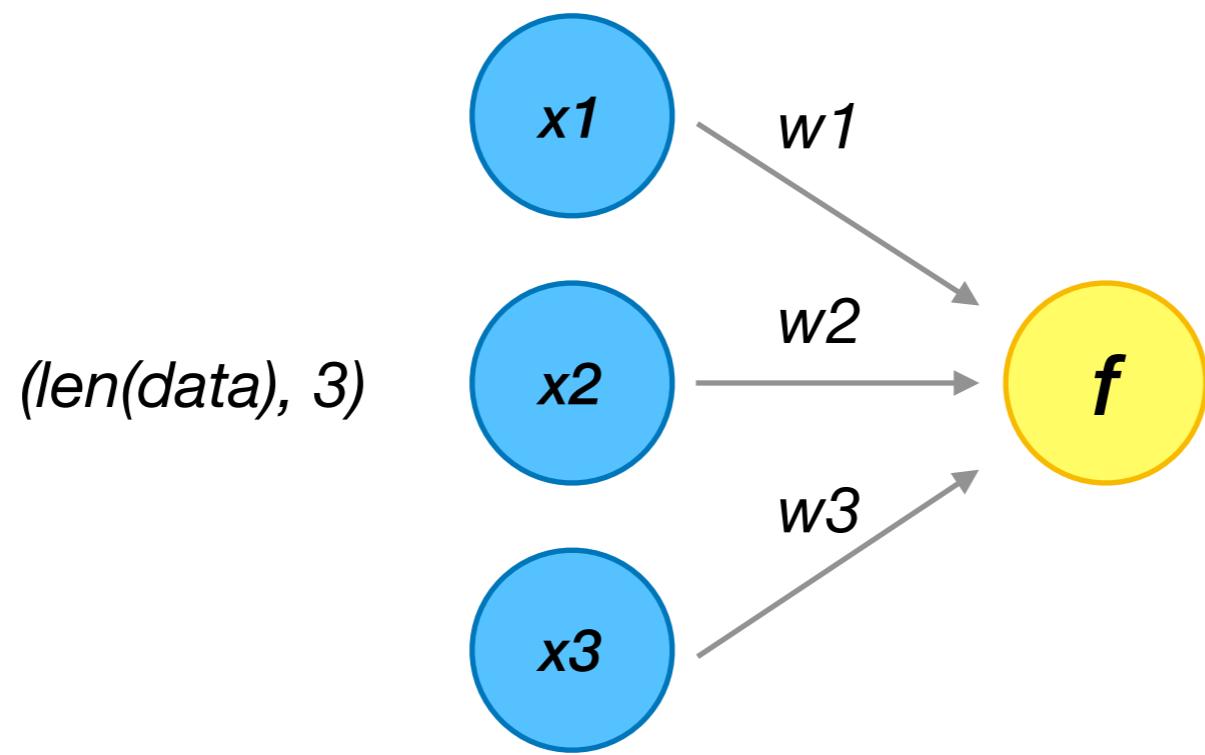
Logistic Regression

Inference

$$f = \mathbf{x}^* \mathbf{w} + b$$

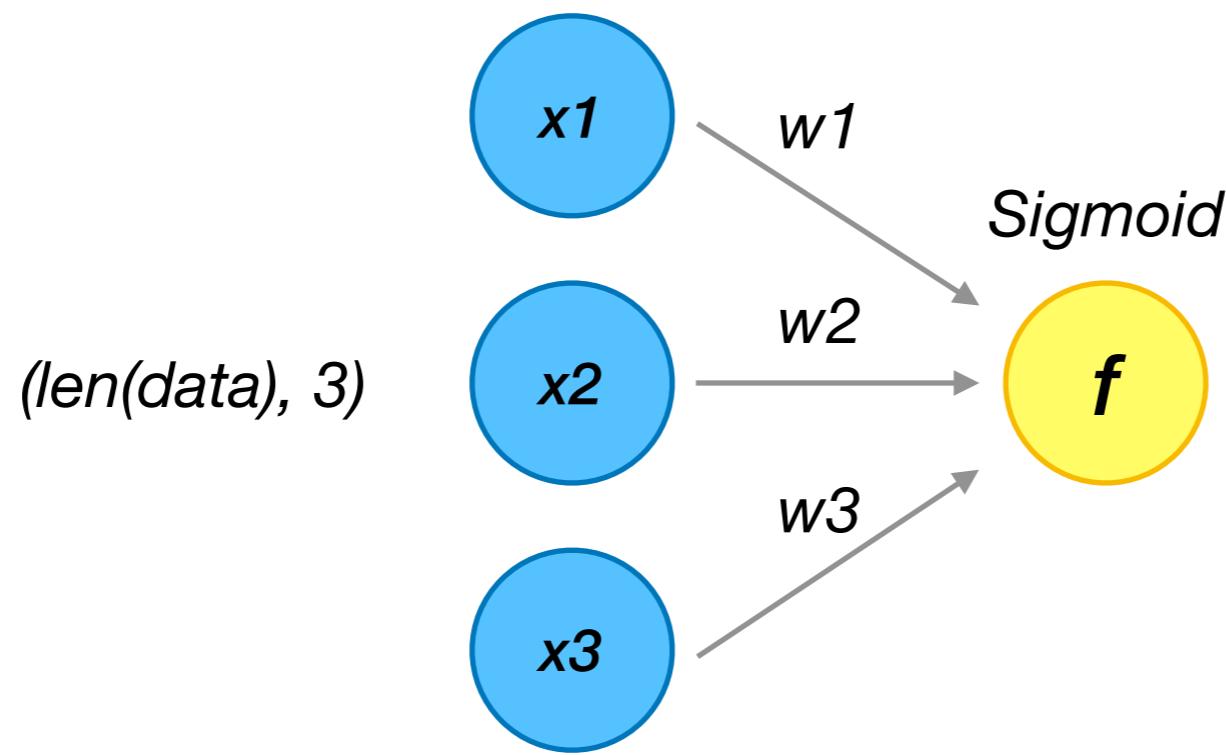
Logistic Regression

Inference



Logistic Regression

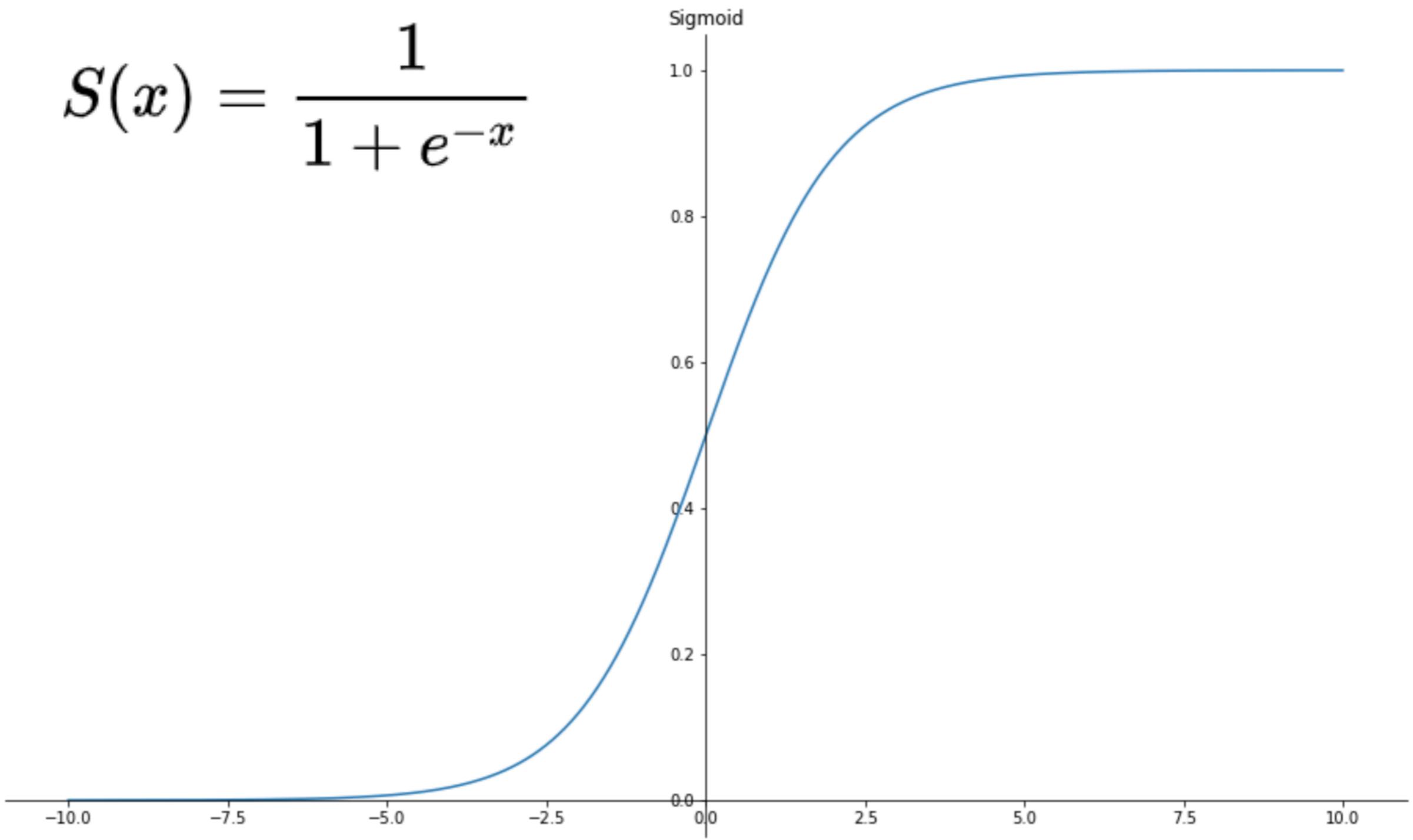
Inference



Logistic Regression

Sigmoid

$$S(x) = \frac{1}{1 + e^{-x}}$$



Logistic Regression

Inference

$$f = 1 / (1 + \exp(-(\mathbf{x} * \mathbf{w} + b)))$$

Logistic Regression

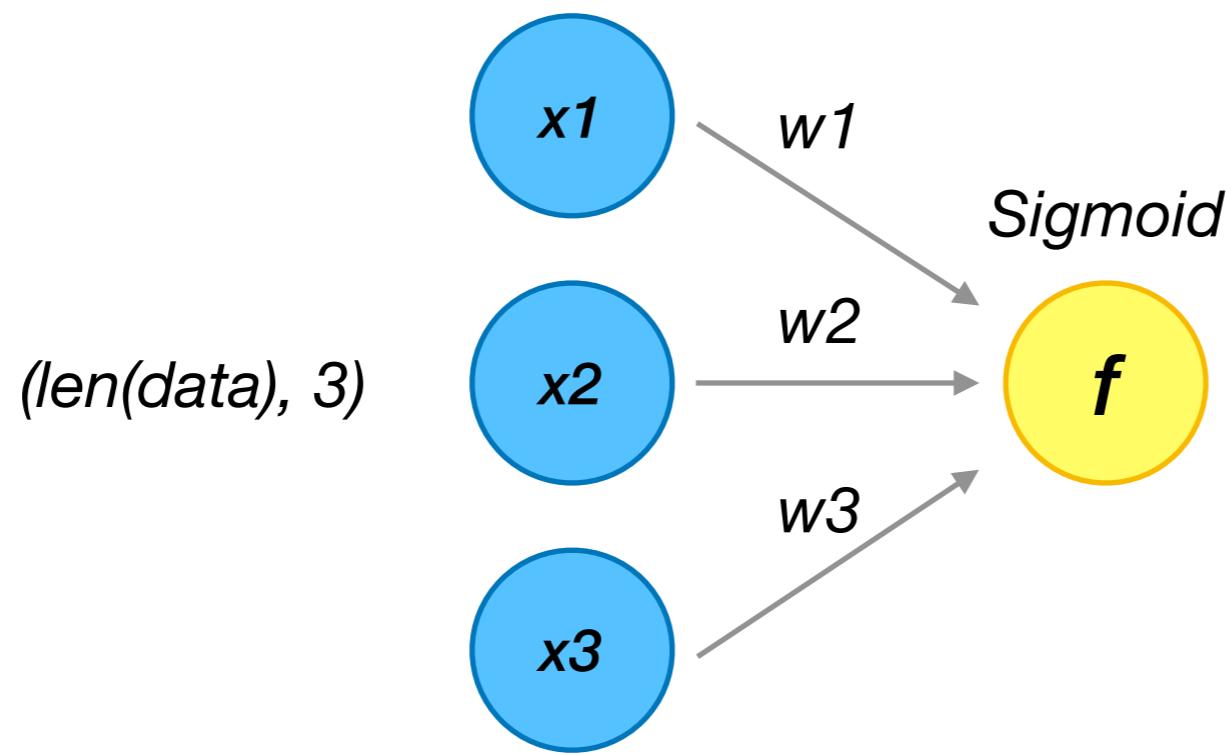
Inference

$$S(x) = \frac{1}{1 + e^{-x}}$$


$$f = 1 / (1 + \exp(-(\mathbf{x} * \mathbf{w} + b)))$$

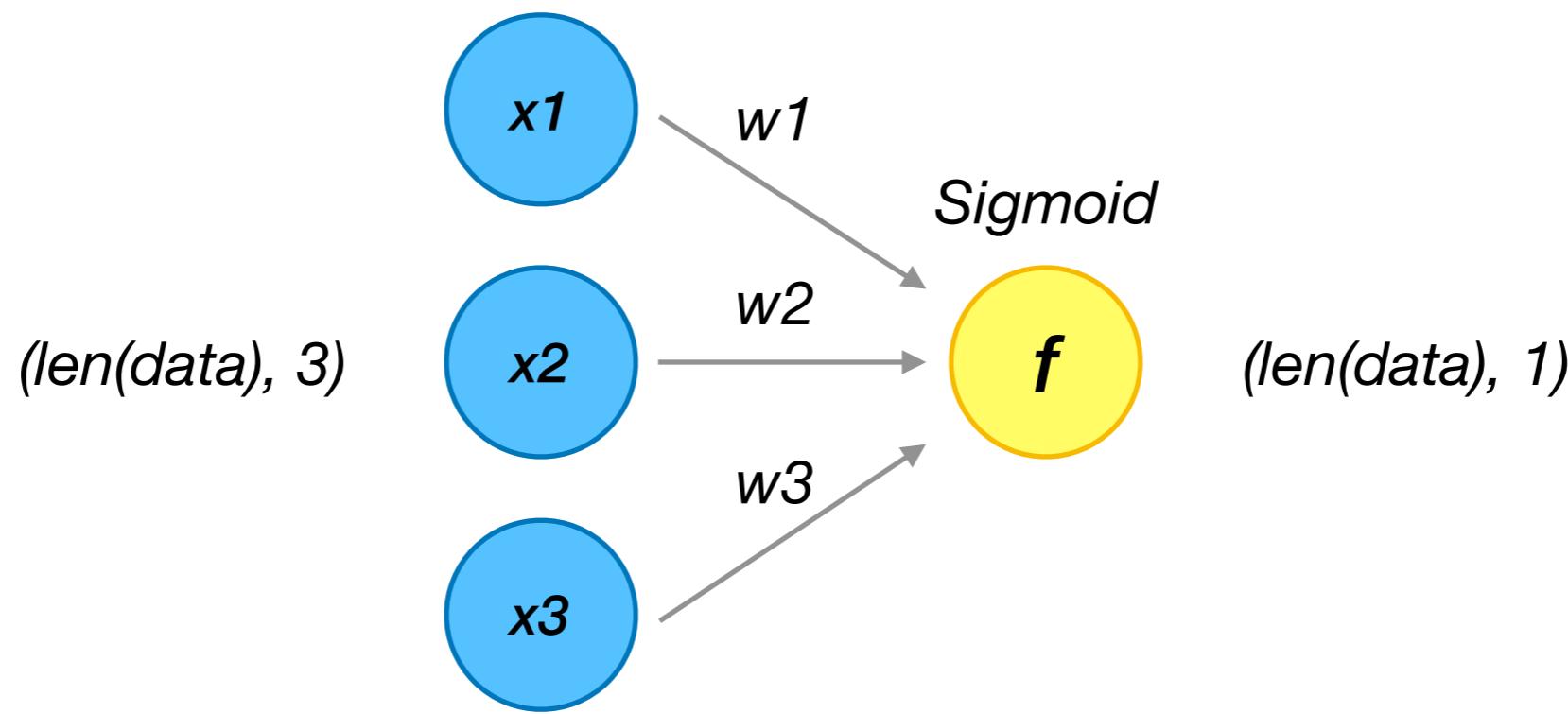
Logistic Regression

Inference



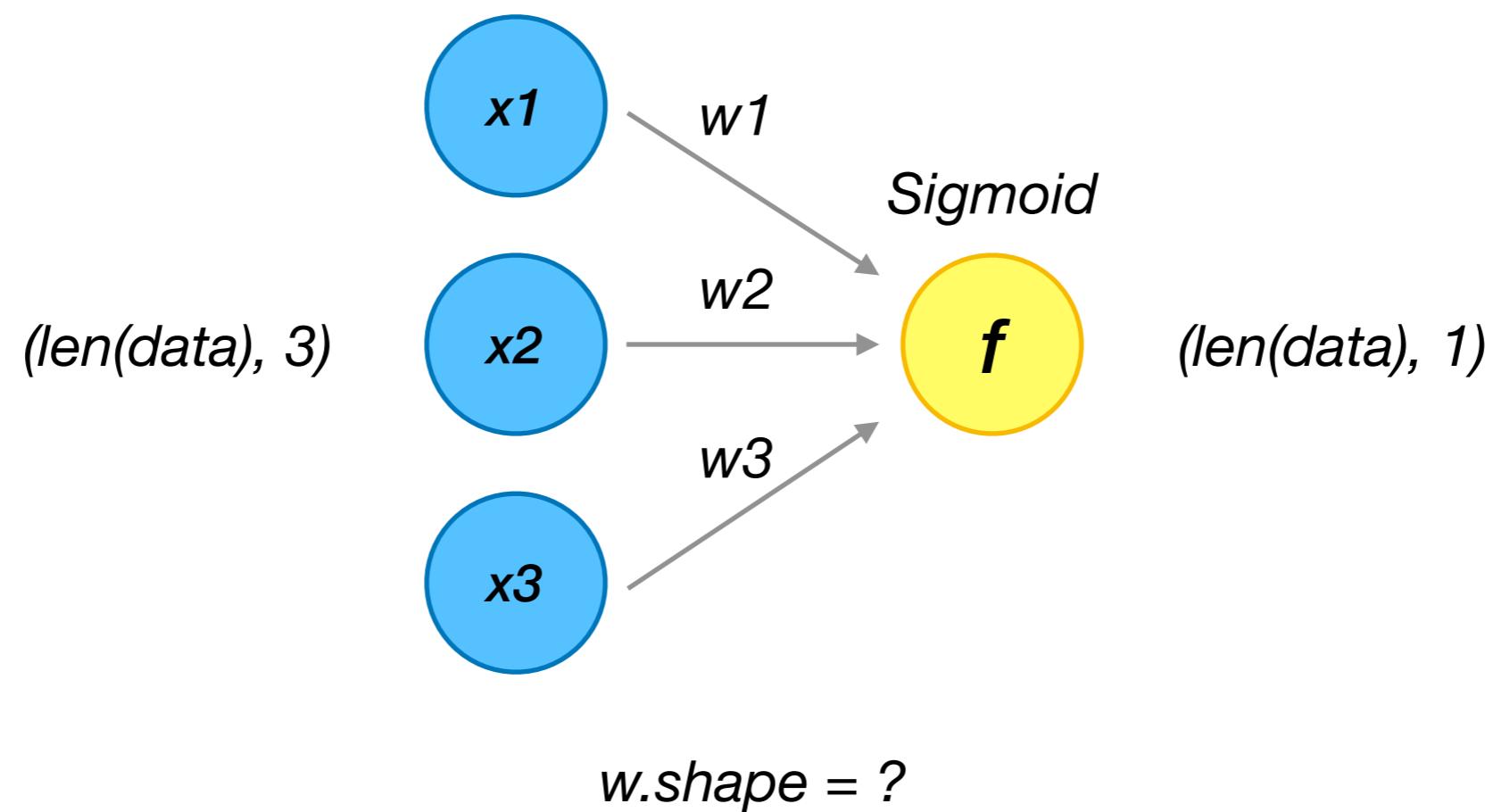
Logistic Regression

Inference



Logistic Regression

Inference



Logistic Regression

Dot Product

$$f = \text{np.dot}(x, w) + b$$

Logistic Regression

Dot Product

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$



$$f = \mathbf{np.dot(x, w)} + \mathbf{b}$$

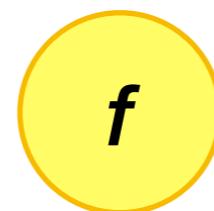
$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

Logistic Regression

Dot Product

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj} = \sum_{s=1}^n a_{sn}b_{sj}$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$



$$\mathbf{f} = \mathbf{np.dot(x, w)} + \mathbf{b}$$

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

Logistic Regression

Dot Product

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj} = \sum_{s=1}^n a_{sn}b_{sj}$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$



$$f = \mathbf{np.dot(x, w)} + \mathbf{b}$$

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

A.shape = (p, m)

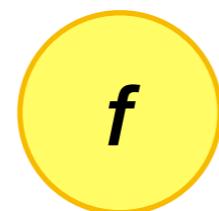
B.shape = (n, k)

Logistic Regression

Dot Product

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj} = \sum_{s=1}^n a_{sn}b_{sj}$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$


$$f = \mathbf{np.dot(x, w)} + \mathbf{b}$$

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

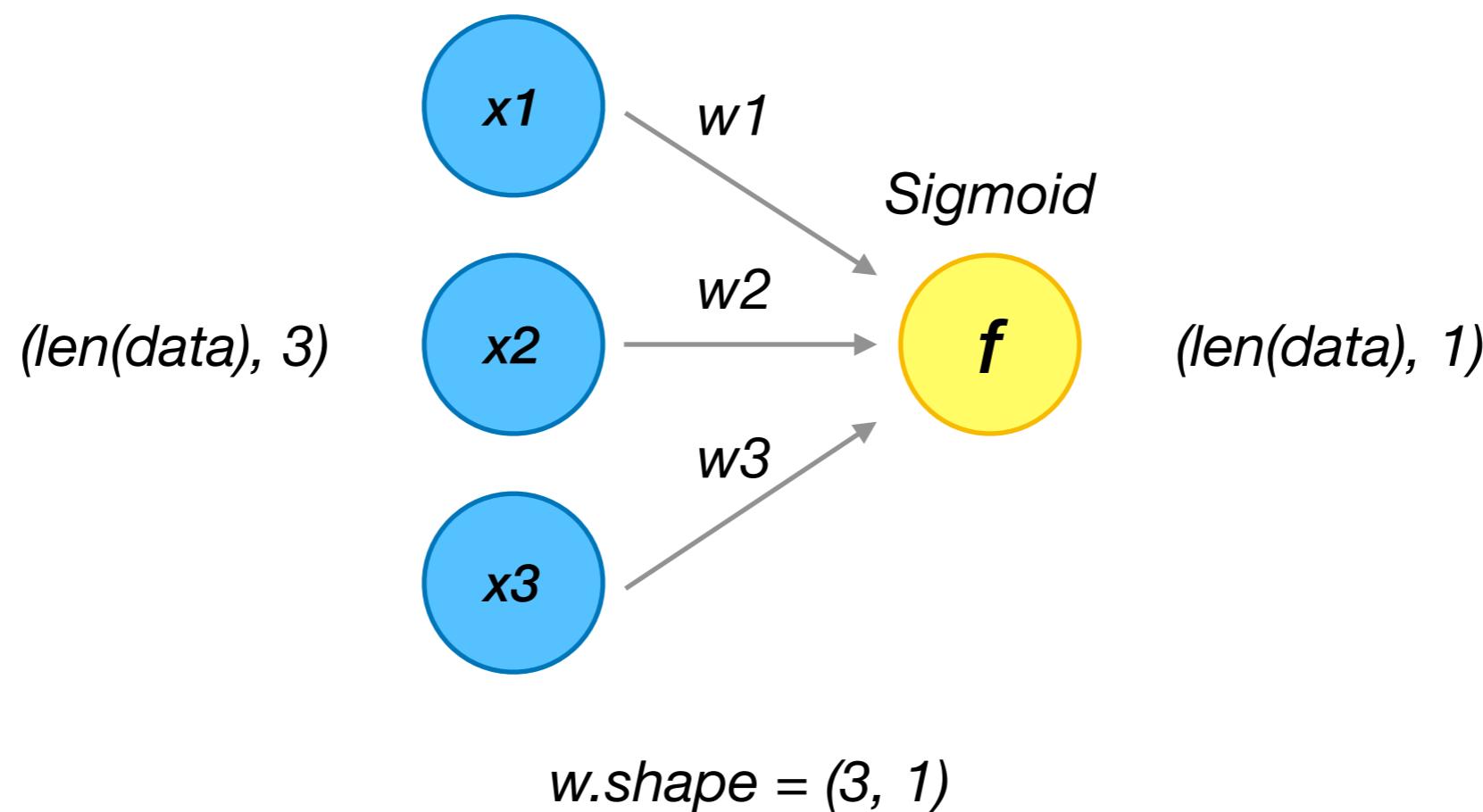
A.shape = (p, m)

B.shape = (n, k)

np.dot(A, B).shape = (p, k) if m == n

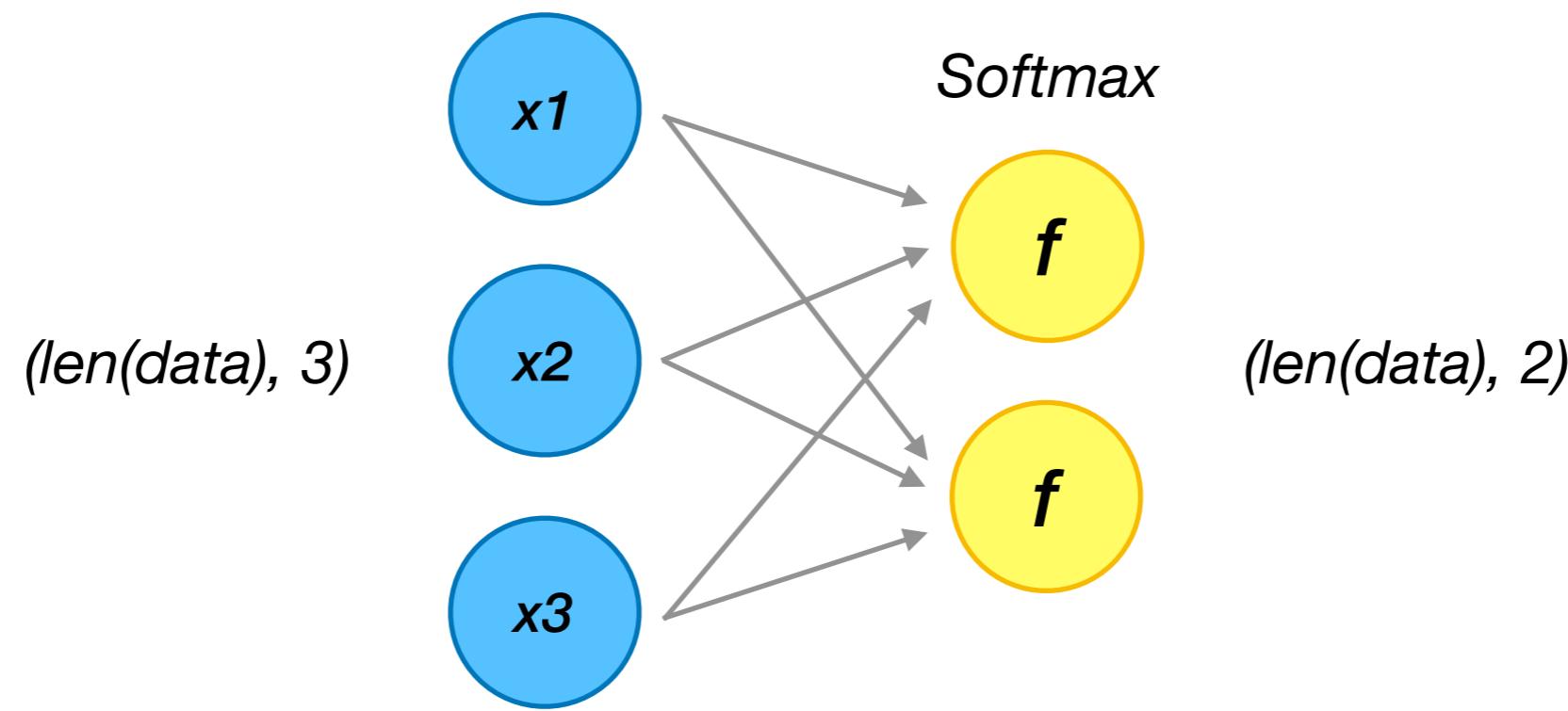
Logistic Regression

Inference



Logistic Regression

Inference



Logistic Regression

Softmax

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

Logistic Regression

Softmax

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

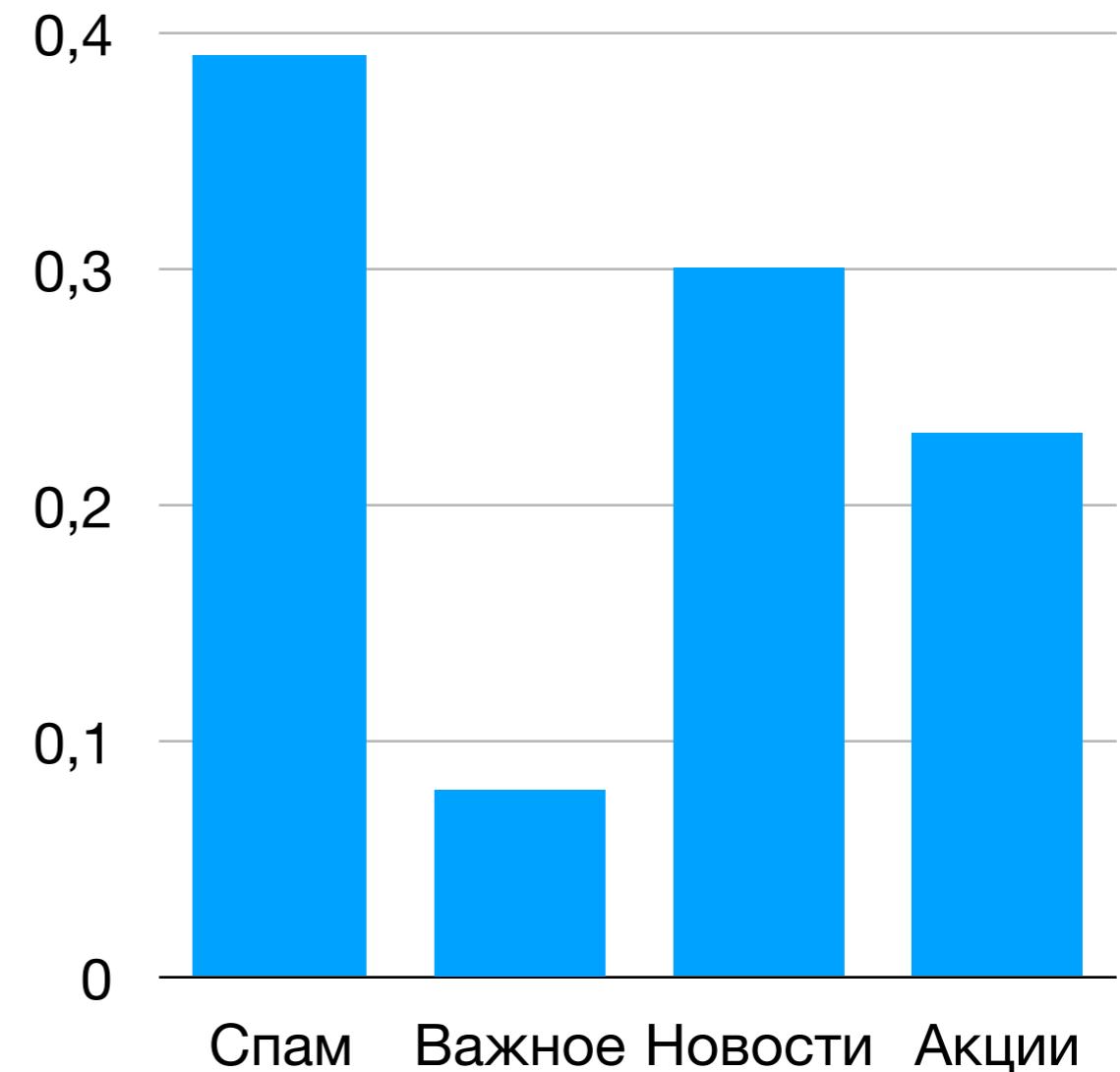
```
exp_scores = np.exp(pred)
```

```
softmax = exp_scores / exp_scores.sum()
```

Logistic Regression

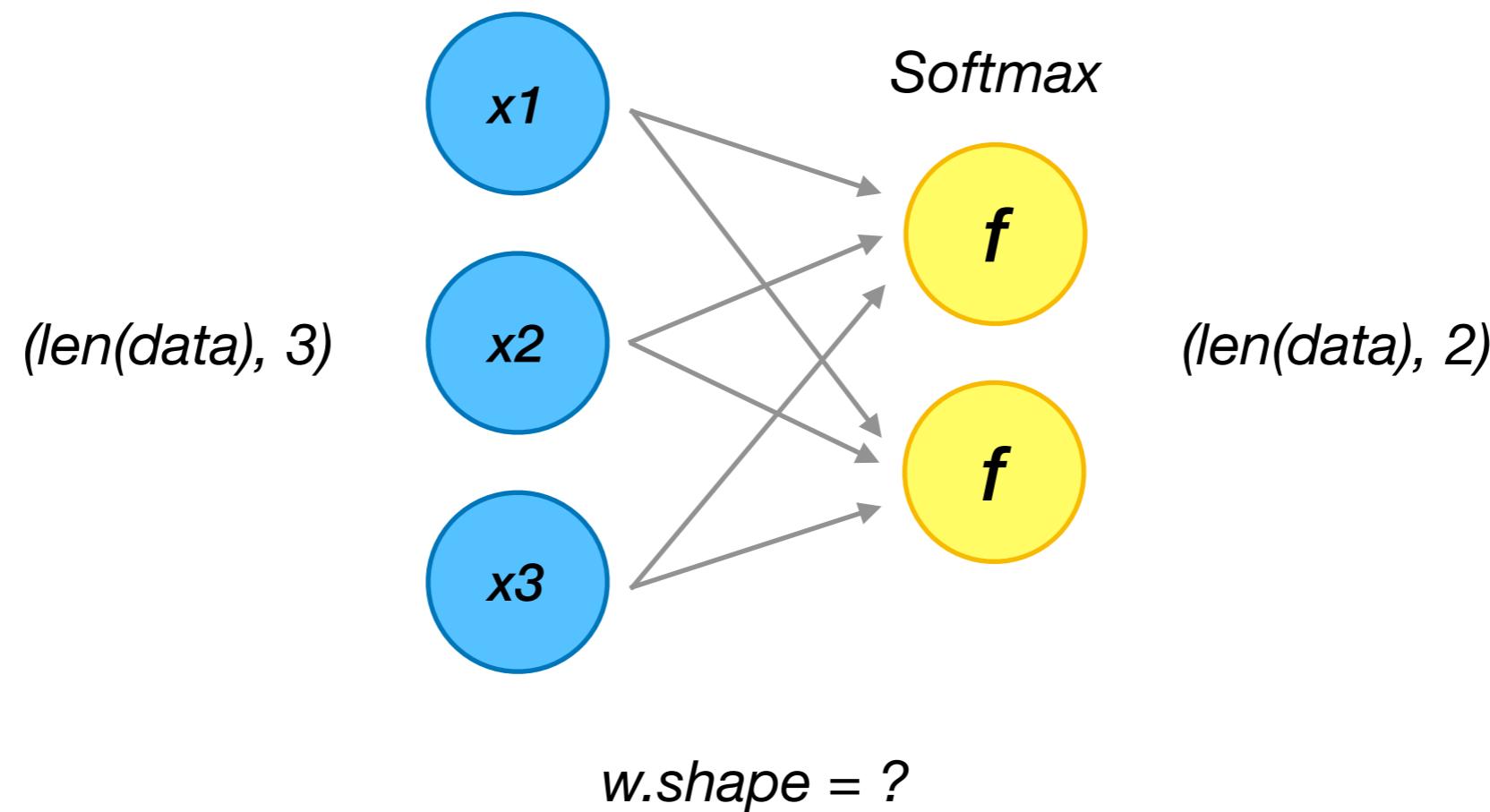
Softmax

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$



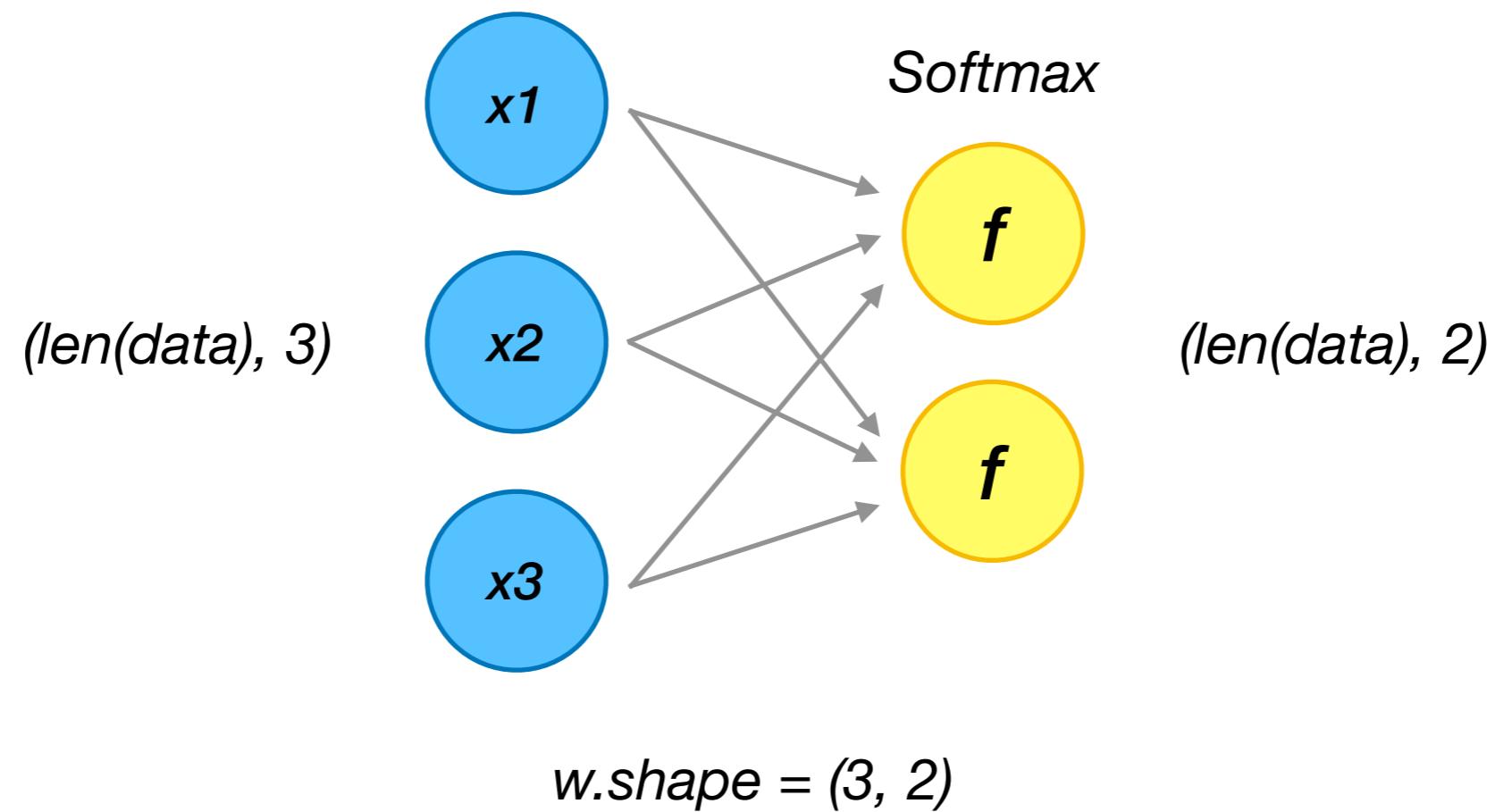
Logistic Regression

Inference



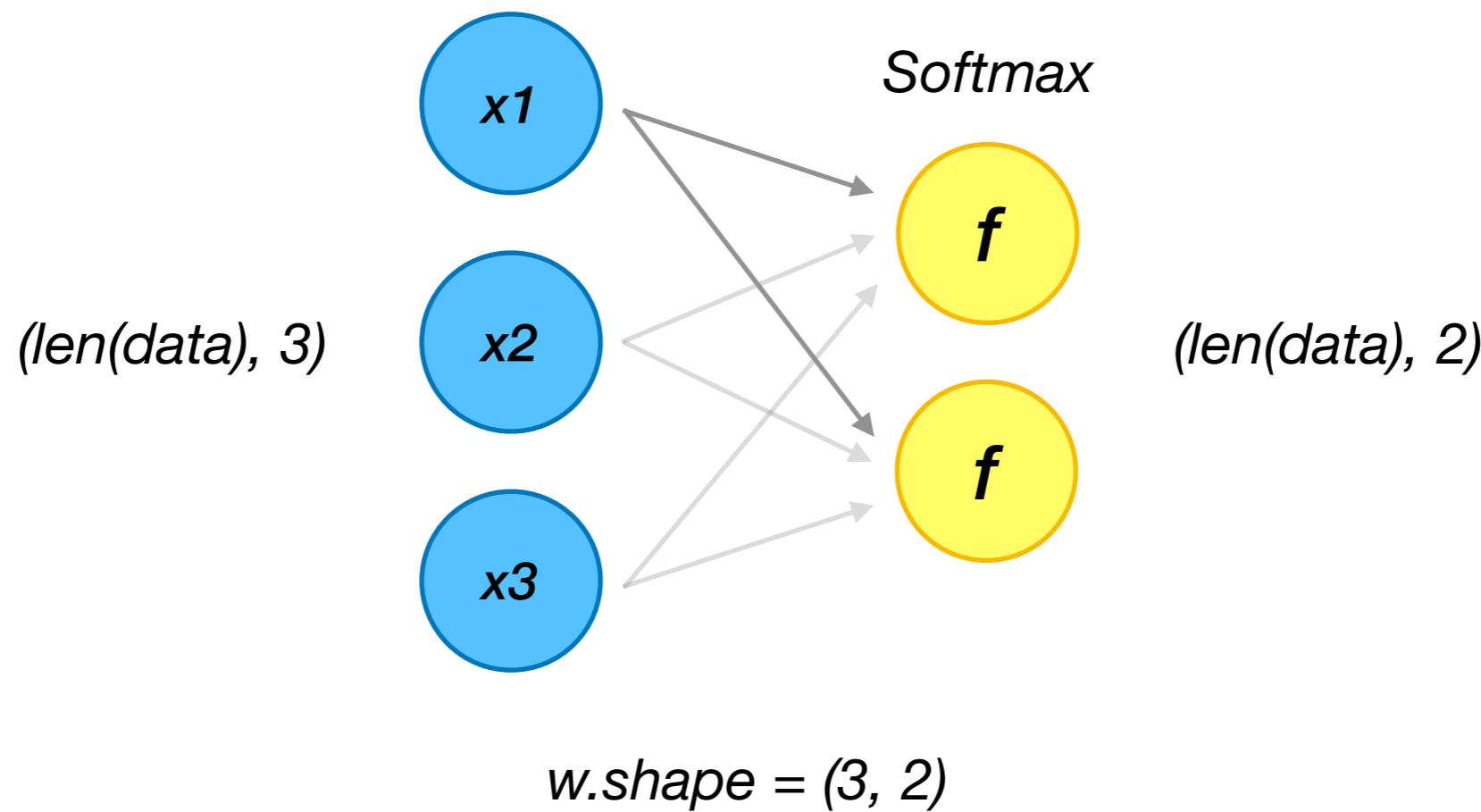
Logistic Regression

Inference



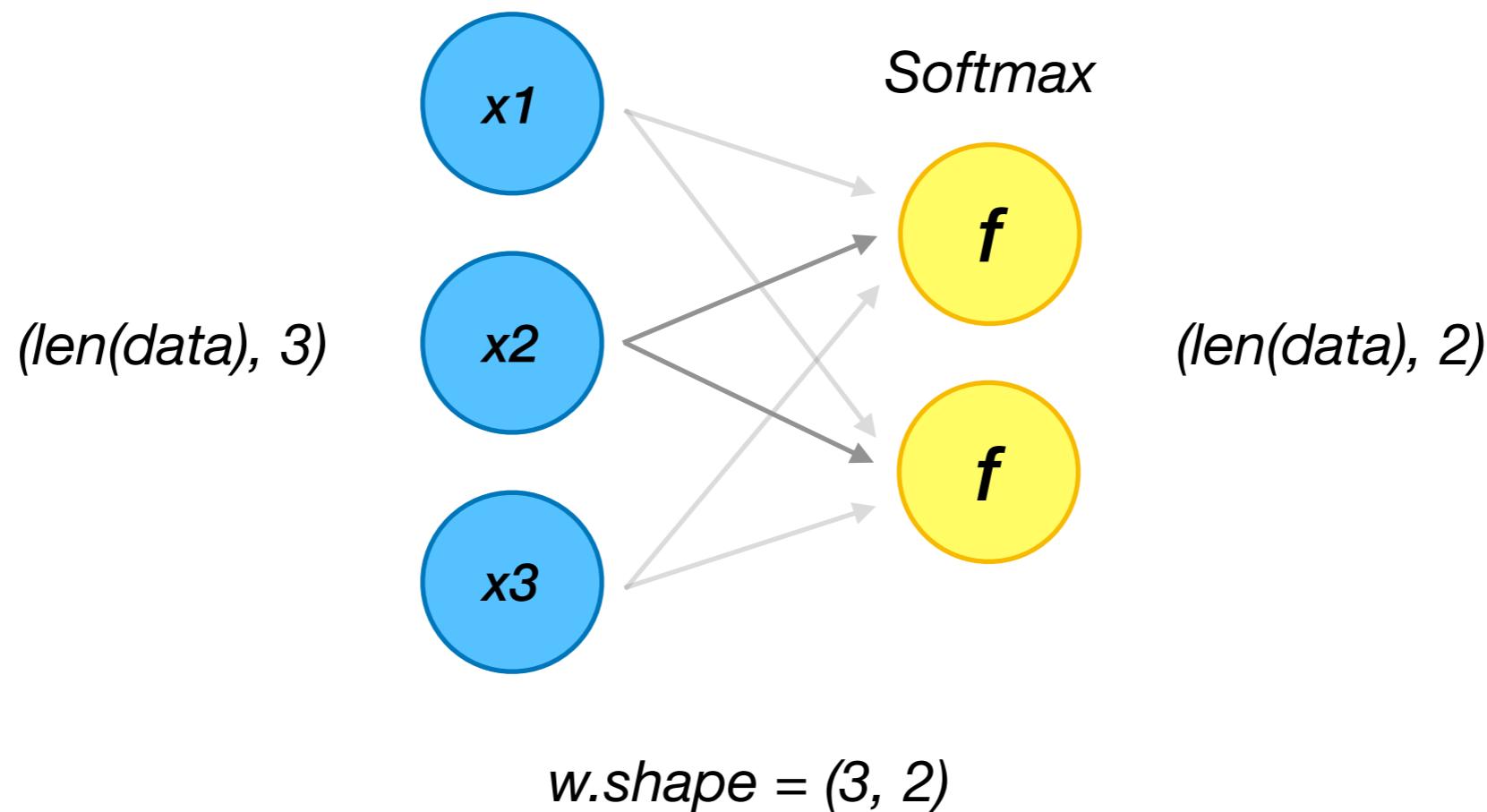
Logistic Regression

Inference



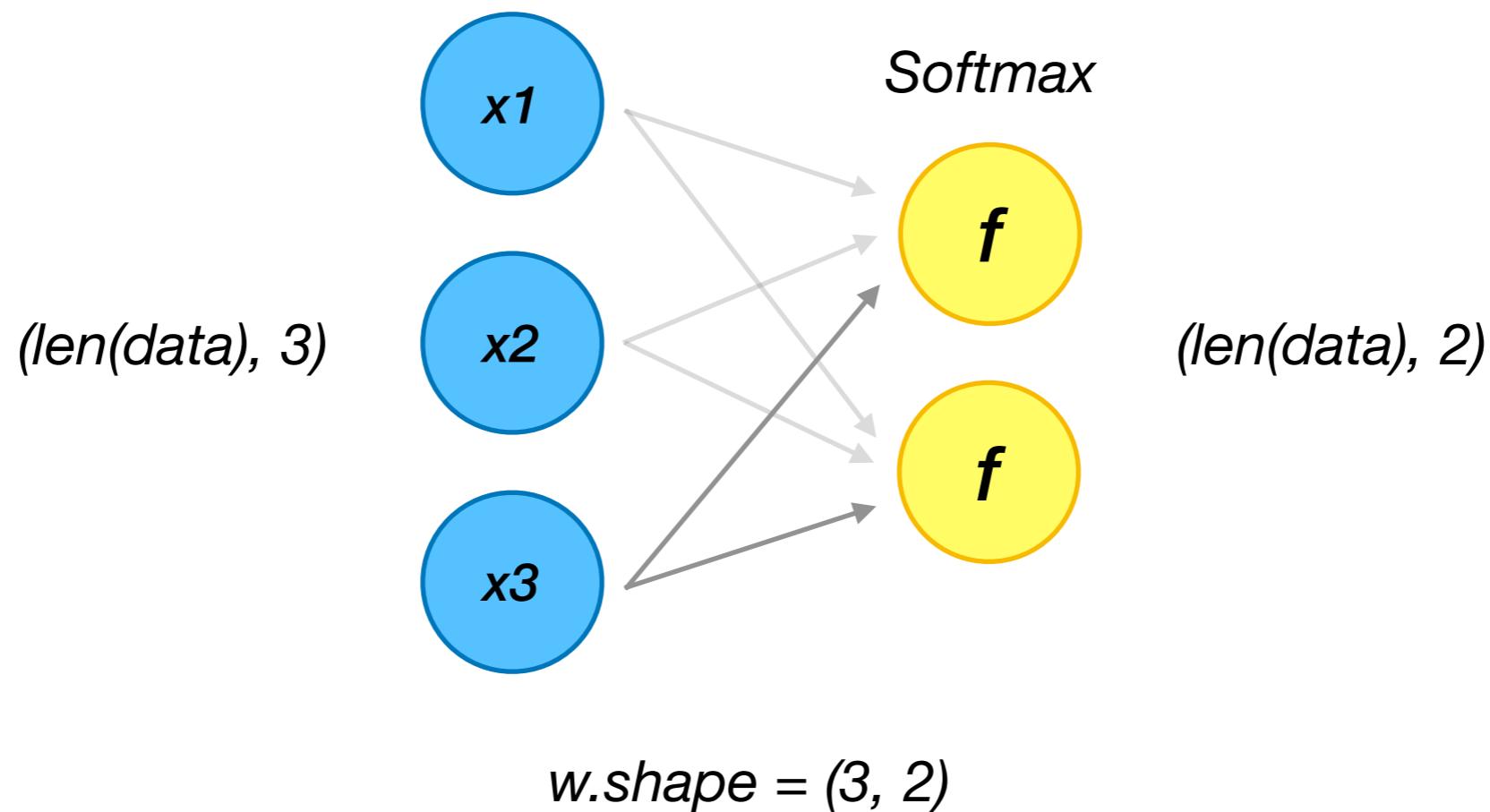
Logistic Regression

Inference



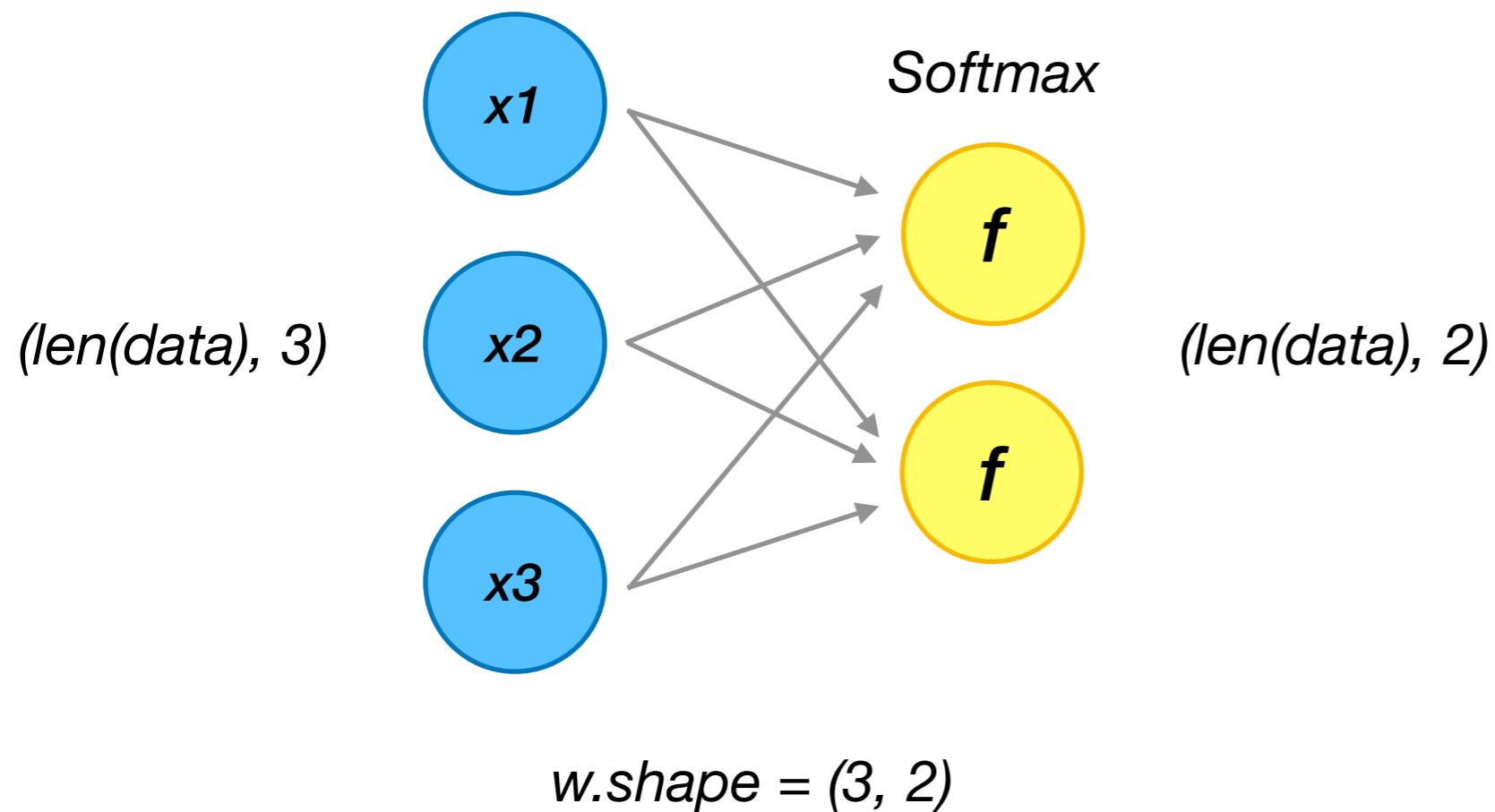
Logistic Regression

Inference



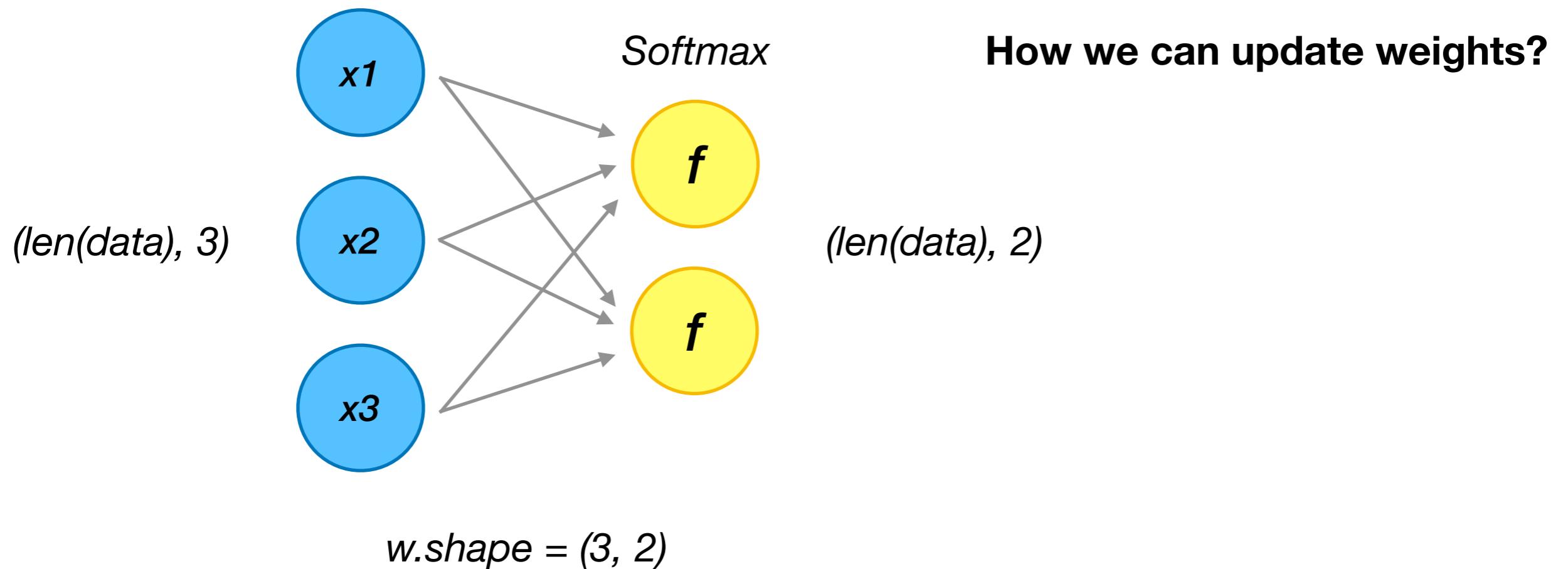
Logistic Regression

Train



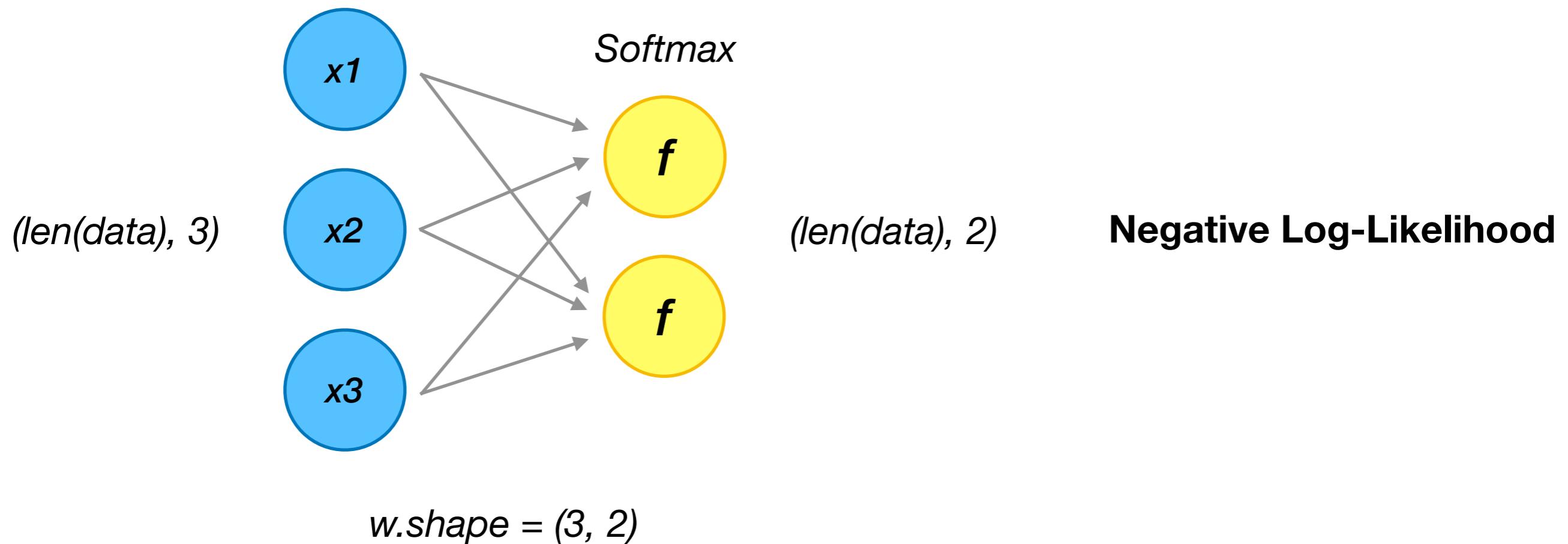
Logistic Regression

Train



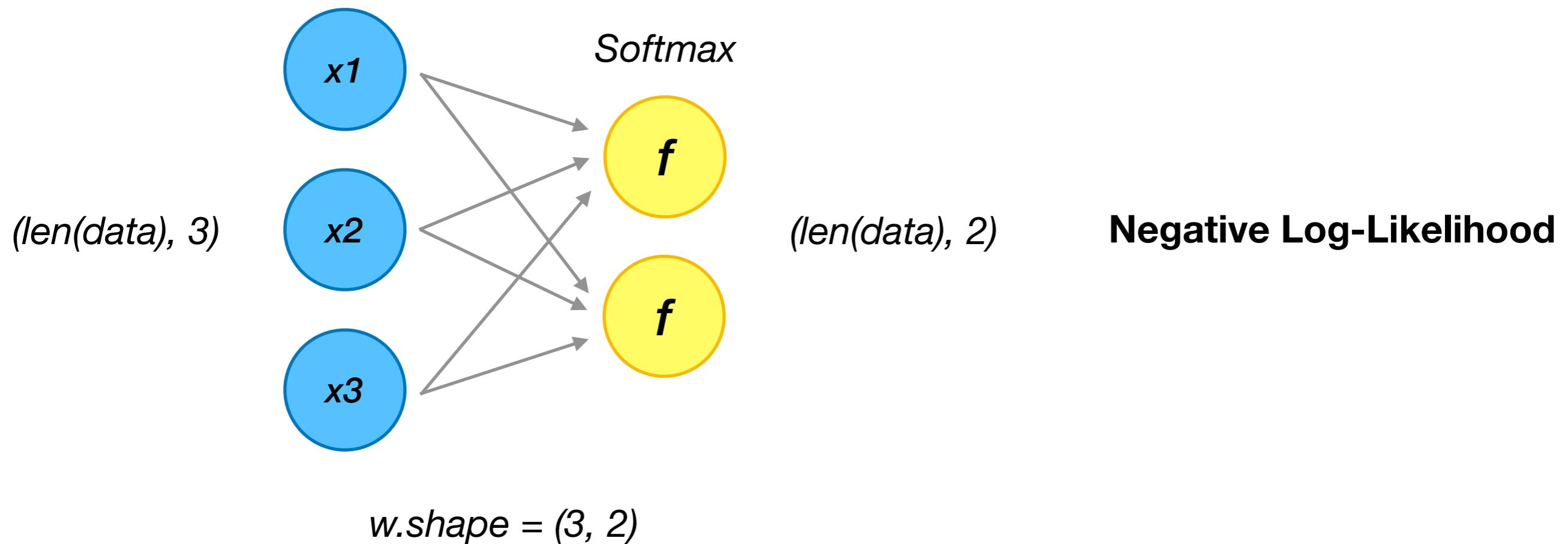
Logistic Regression

Train



Logistic Regression

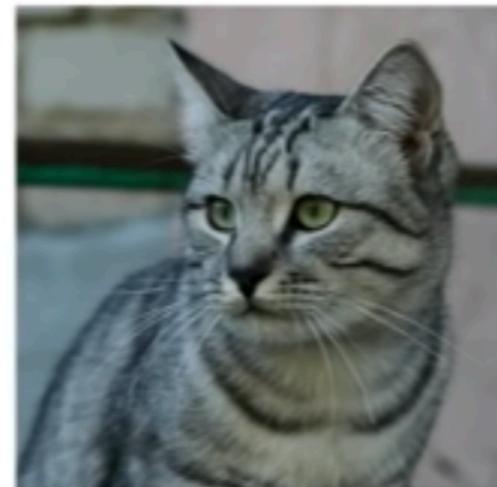
Train



$$H_p(q) = - \sum_{c=1}^C q(y_c) \cdot \log(p(y_c))$$

Logistic Regression

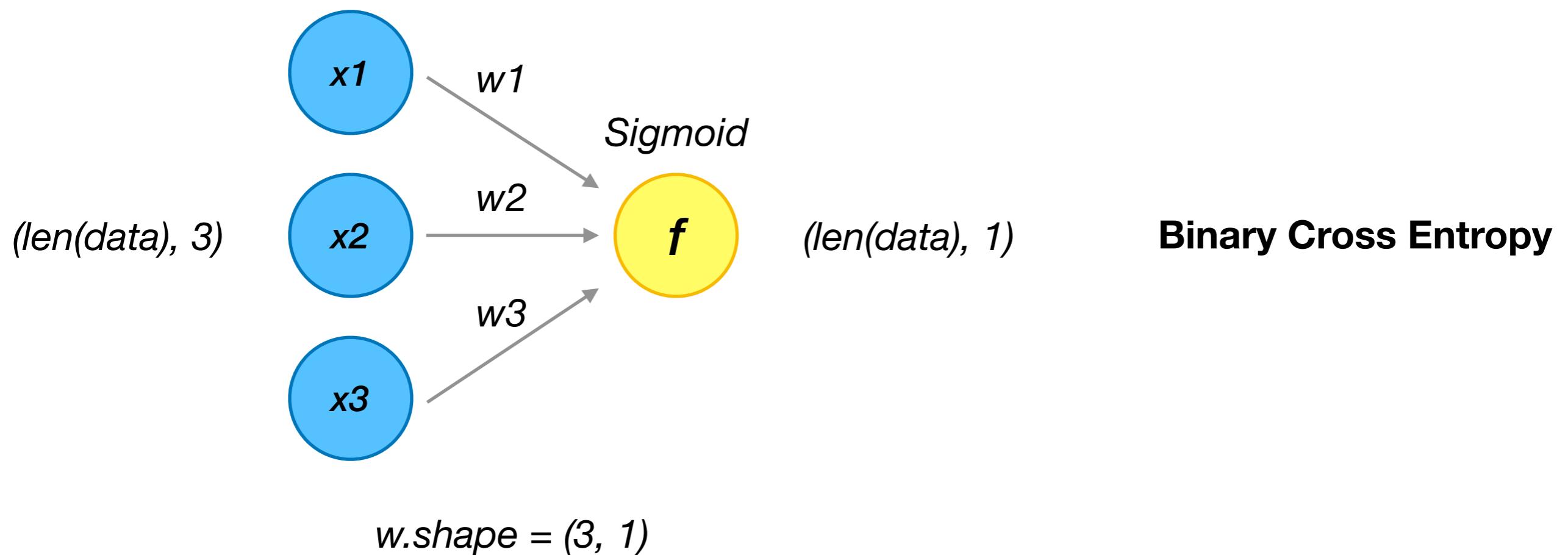
Negative Log-Likelihood



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

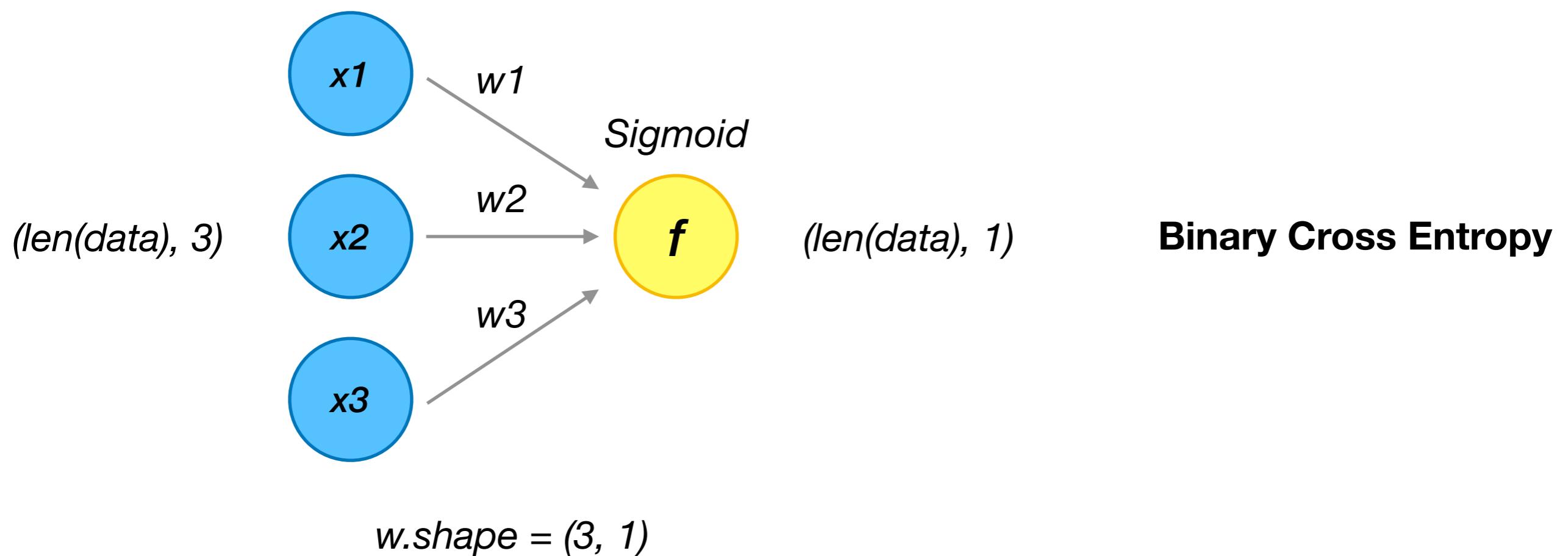
Logistic Regression

Train



Logistic Regression

Train



$$L(y, \hat{y}) = - (y * \log(\hat{y}_i) + (1 - y) * \log(1 - \hat{y}_i))$$

Gradient Descent



Gradient Descent

Numerical derivative

Random Search

Save weight if current loss better than best loss

Gradient Descent

Numerical derivative

~~Random Search~~

Save weight if current loss better than best loss

Gradient Descent

Numerical derivative

$$\frac{\partial L}{\partial w} \approx \frac{L(w + \epsilon) - L(w - \epsilon)}{2\epsilon}$$

Gradient Descent

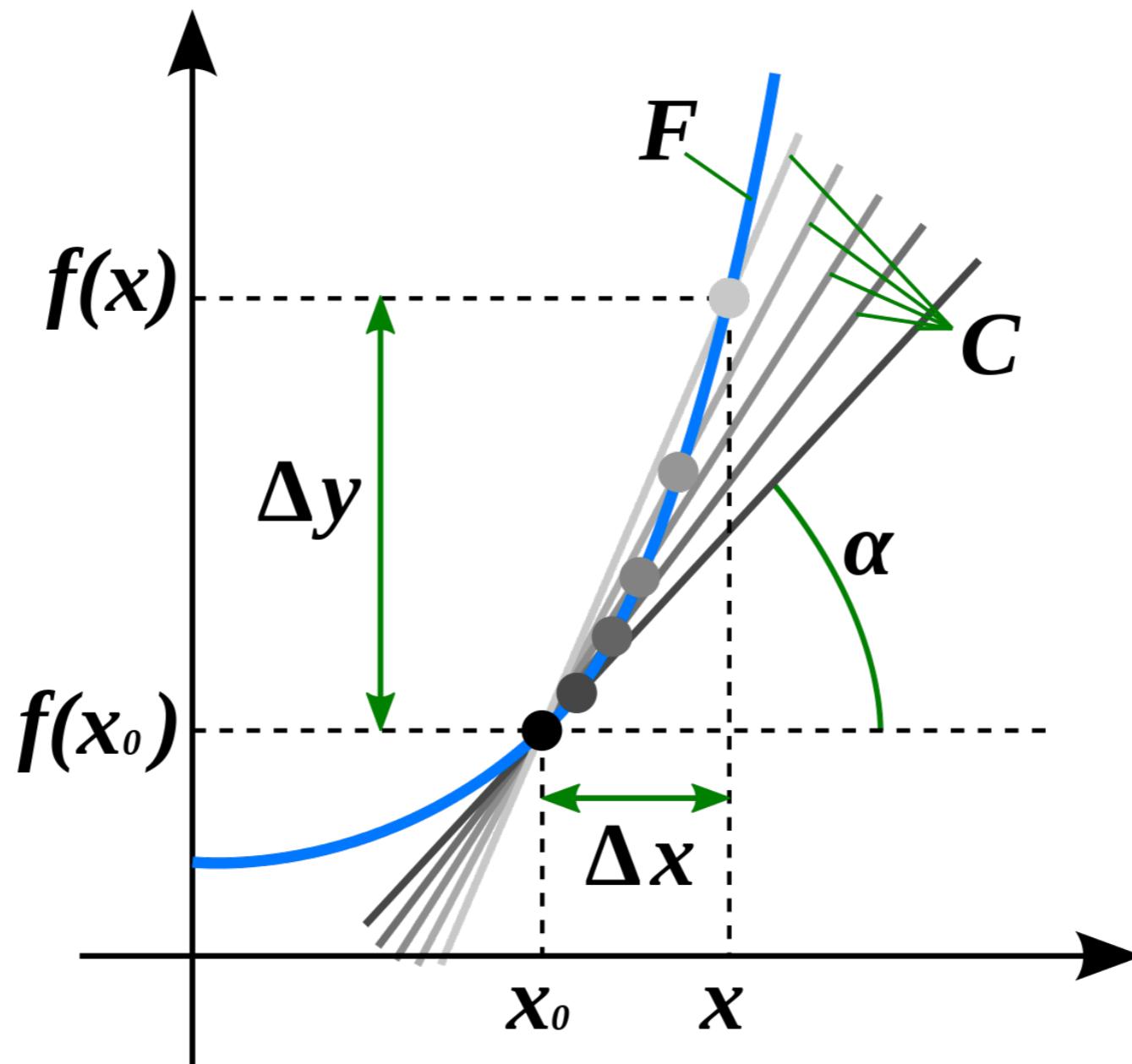
Numerical derivative

$$\frac{\partial L}{\partial w} \approx \frac{L(w + \epsilon) - L(w - \epsilon)}{2\epsilon}$$

But good for gradient checking

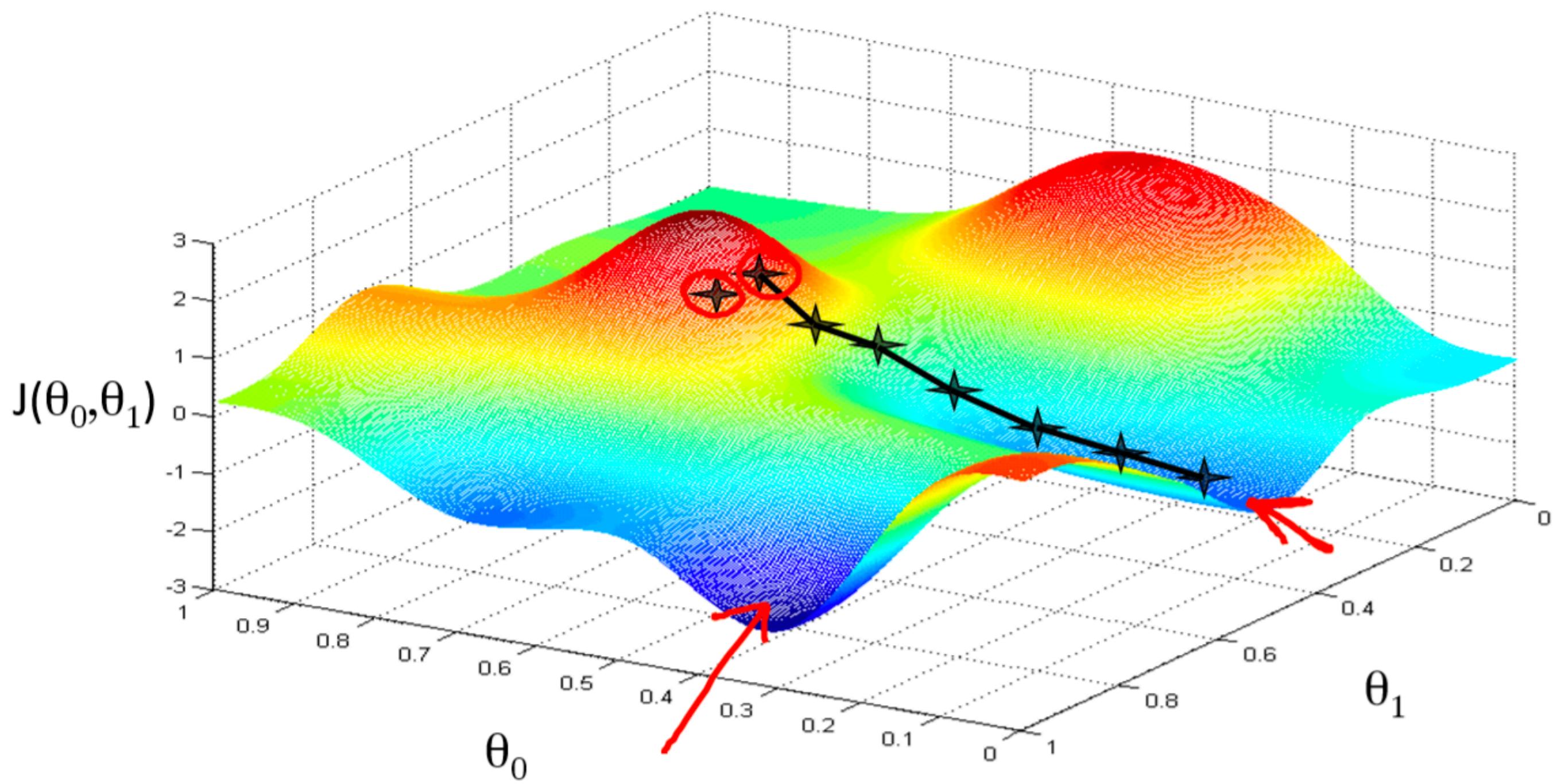
Gradient Descent

Derivative



Gradient Descent

Local minima



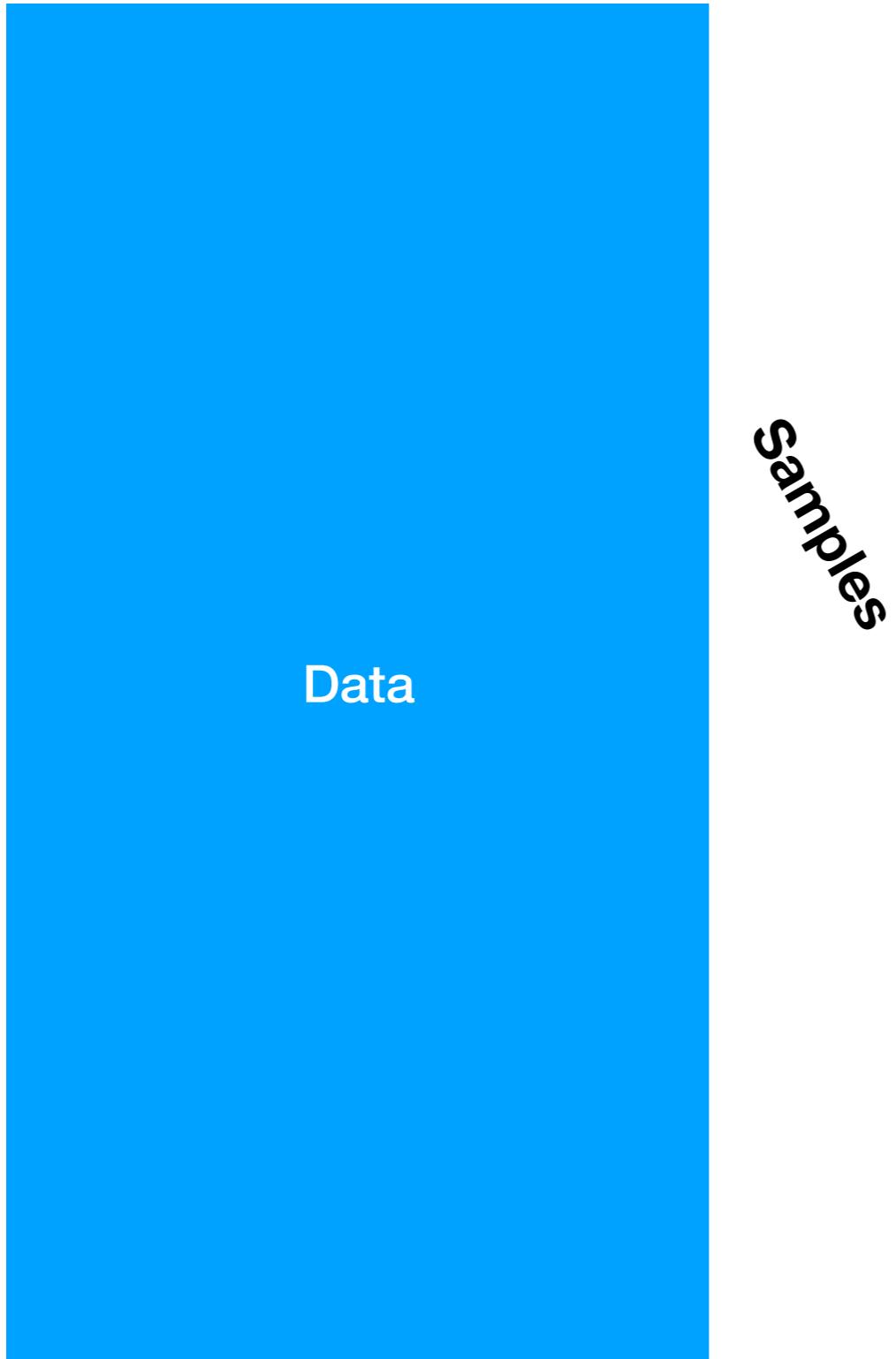
Gradient Descent

```
w = w - learning_rate * dLdW  
b = b - learning_rate * dLdb
```

Gradient Descent

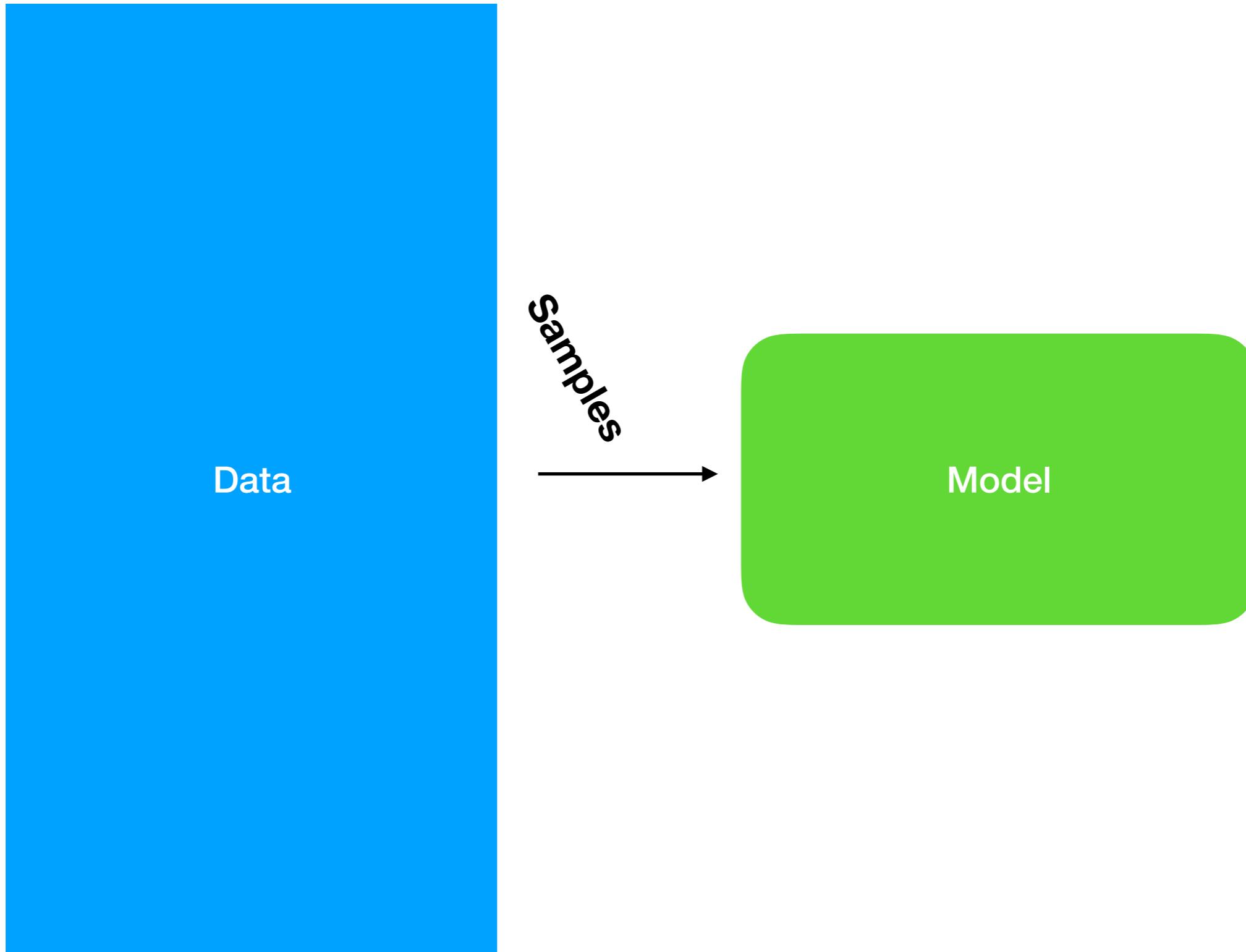
Gradient Descent

Features



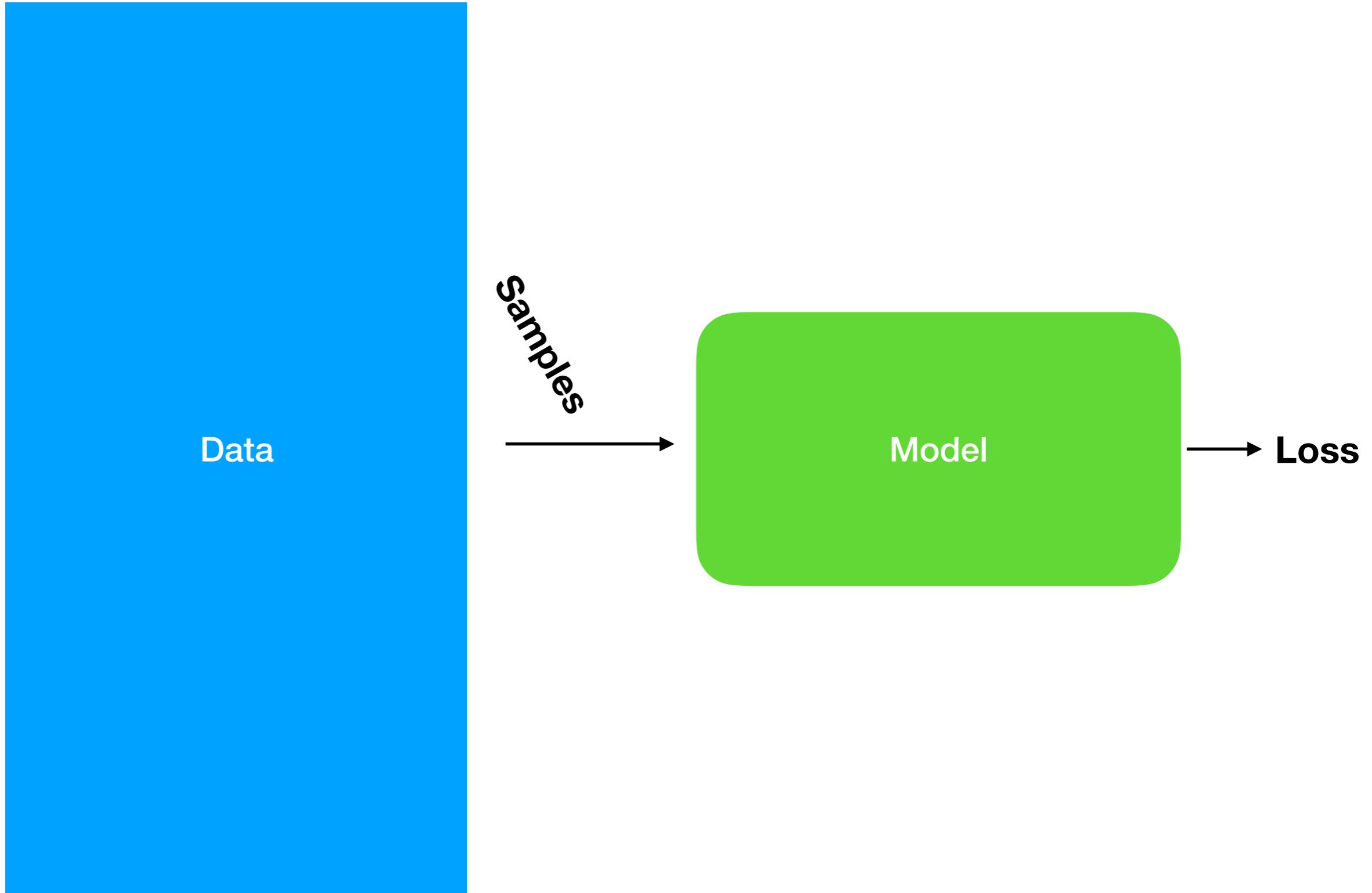
Gradient Descent

Features



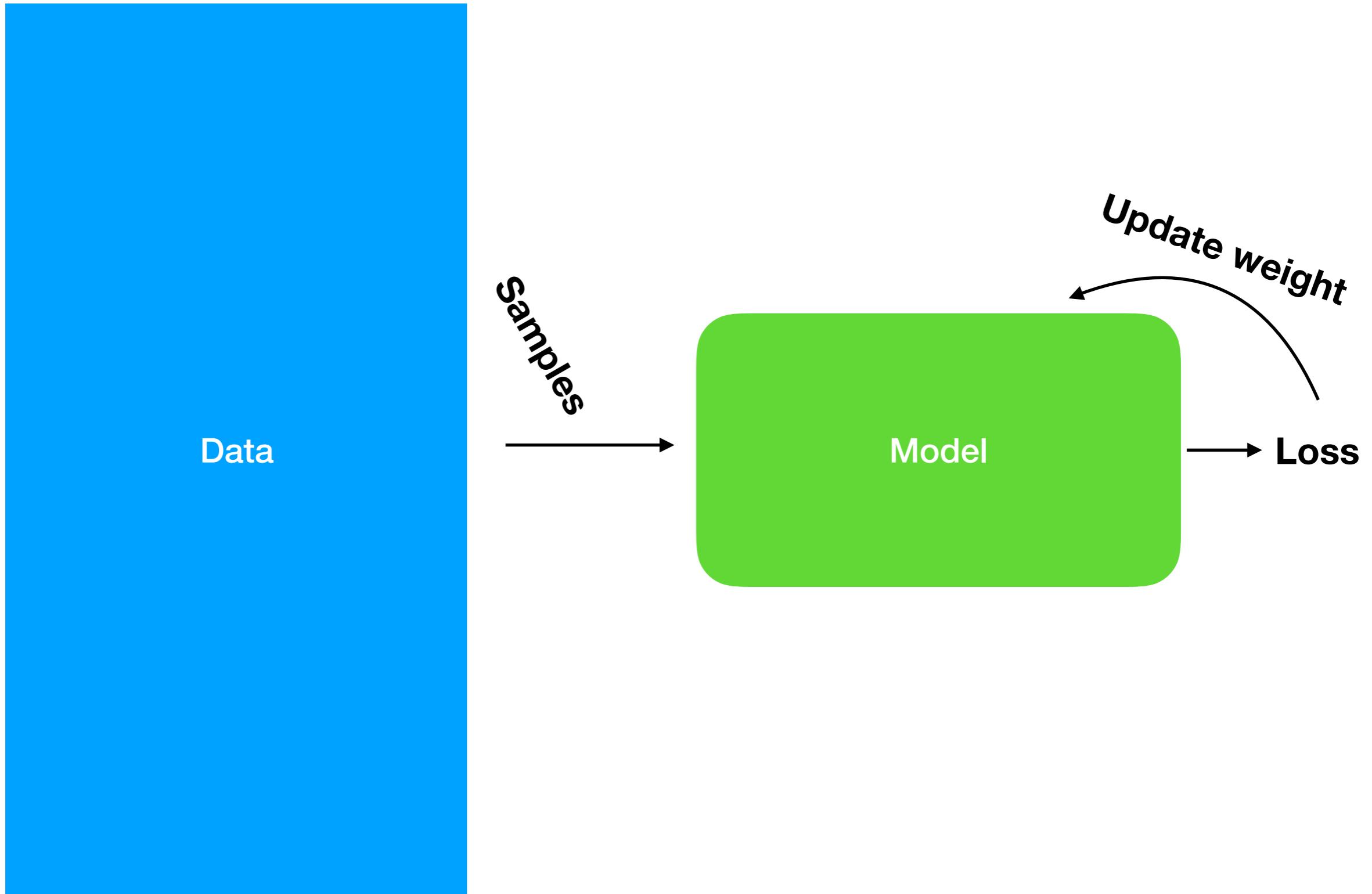
Gradient Descent

Features



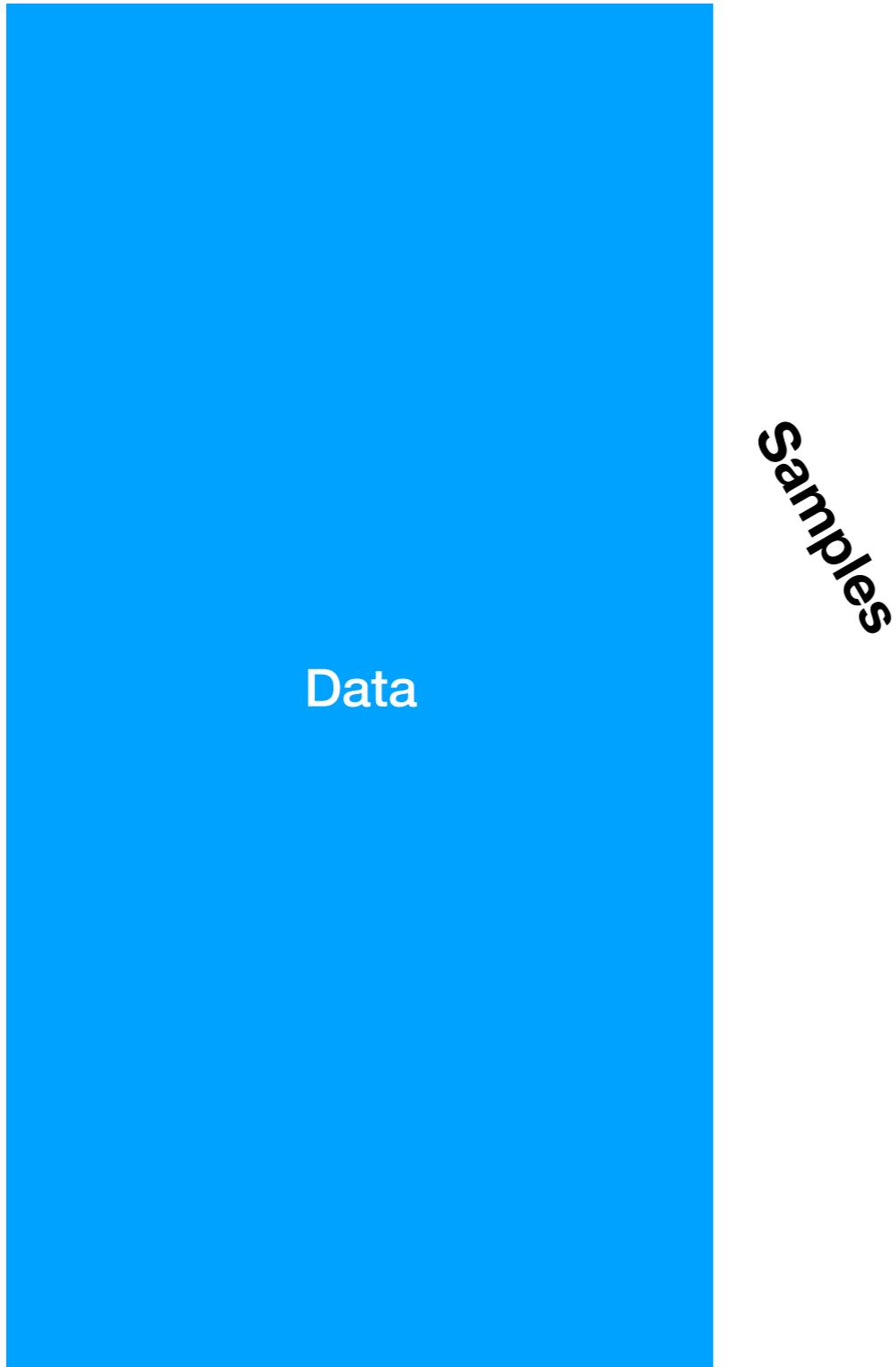
Gradient Descent

Features

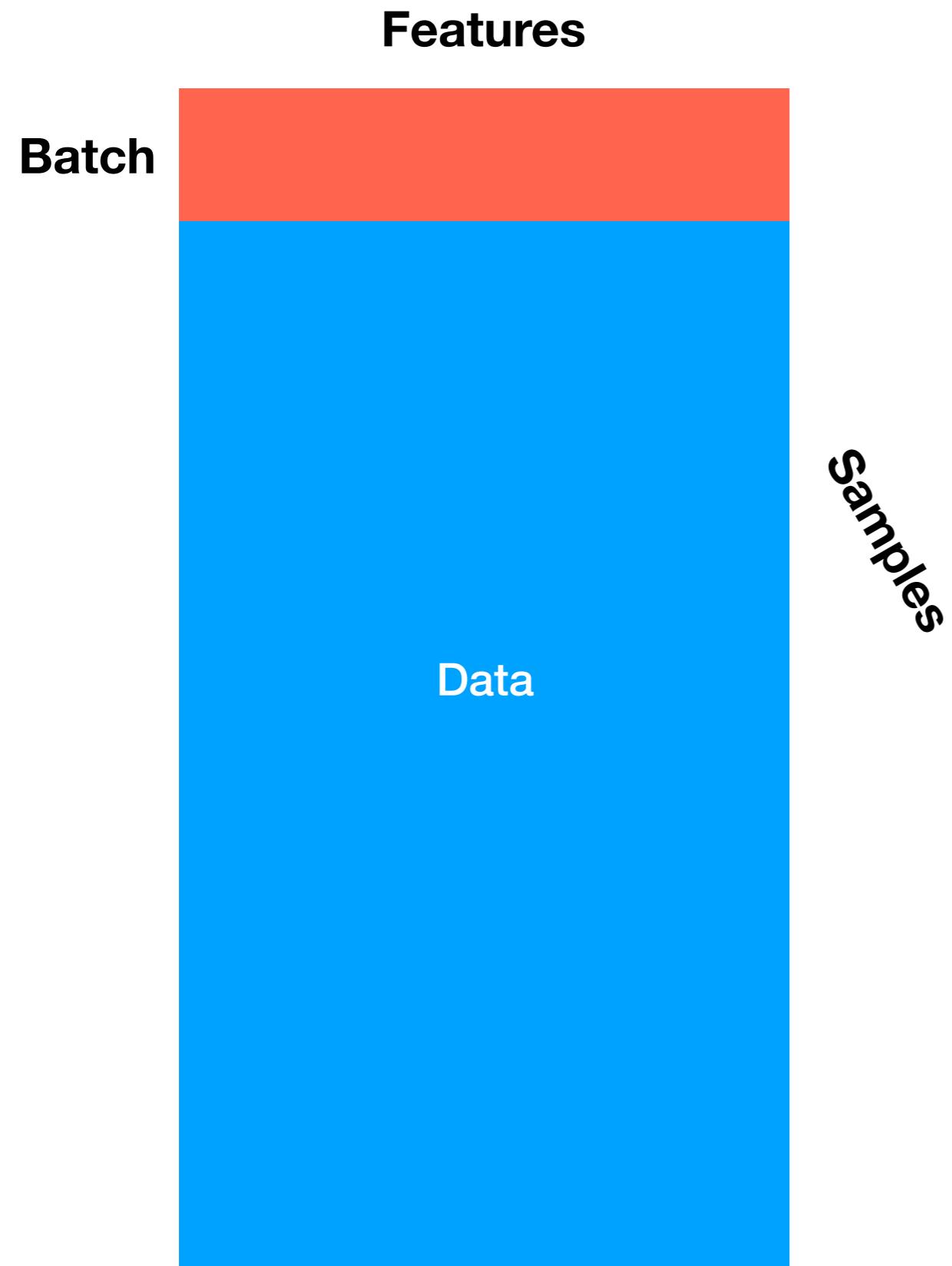


Stochastic Gradient Descent

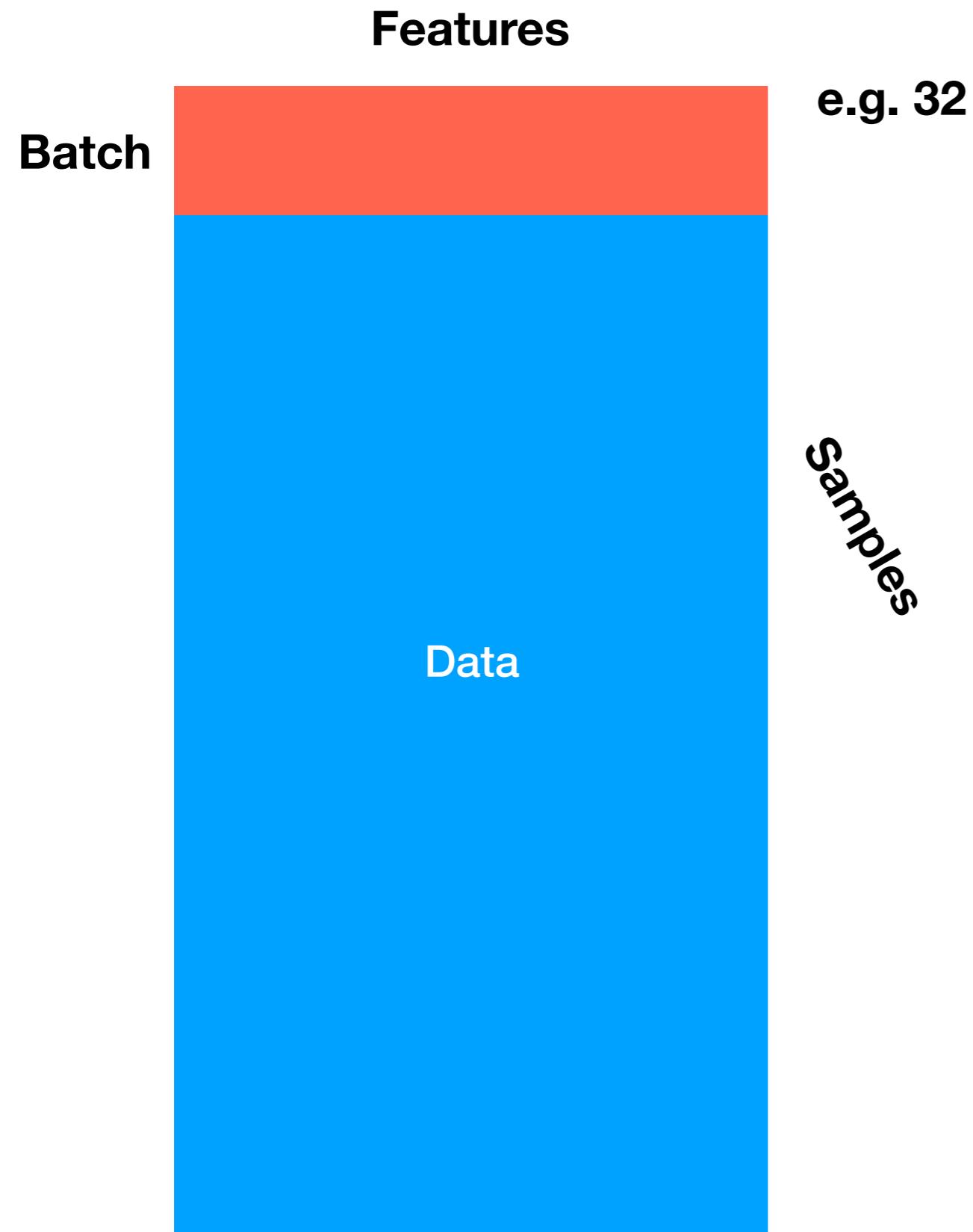
Features



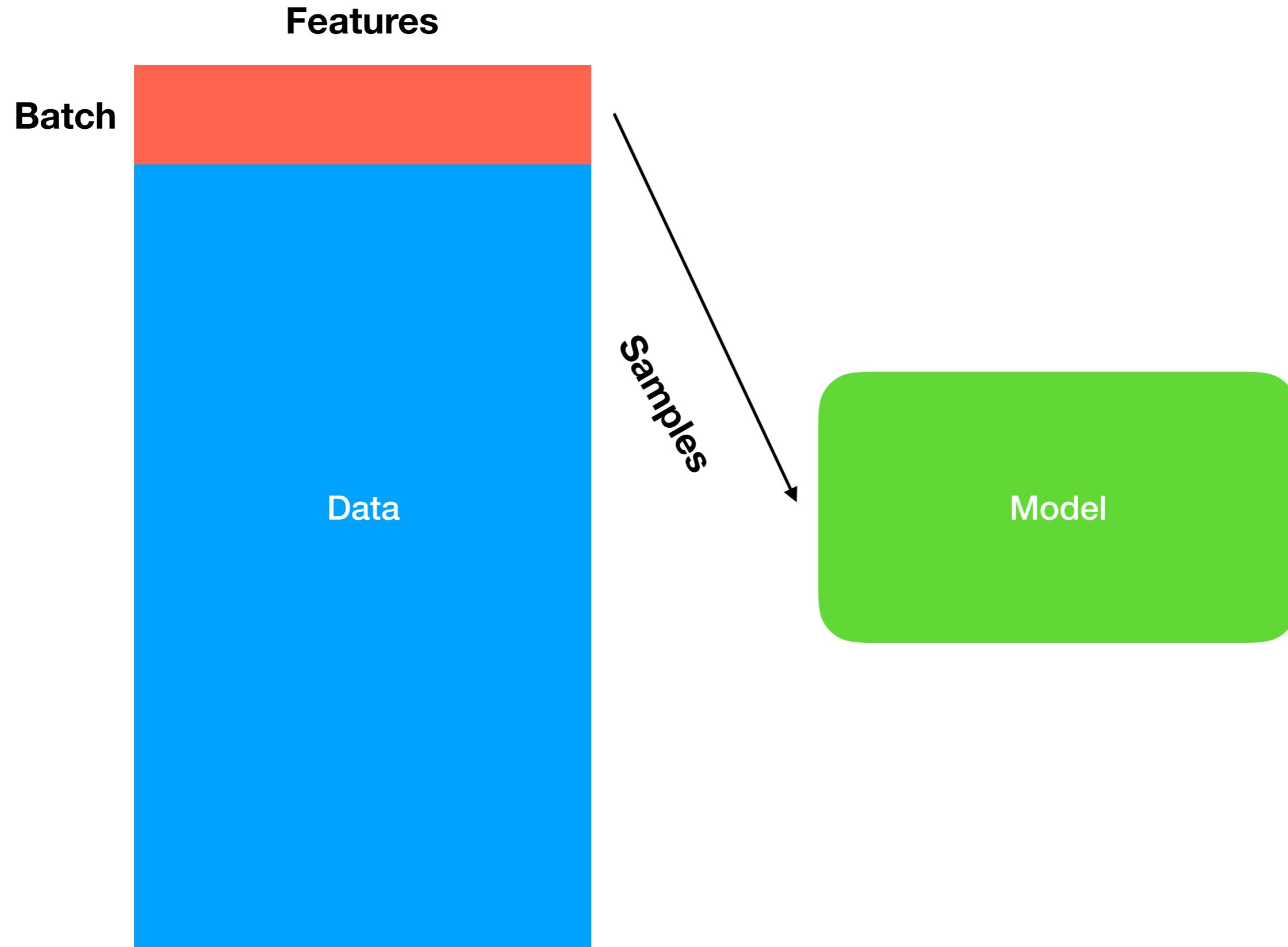
Stochastic Gradient Descent



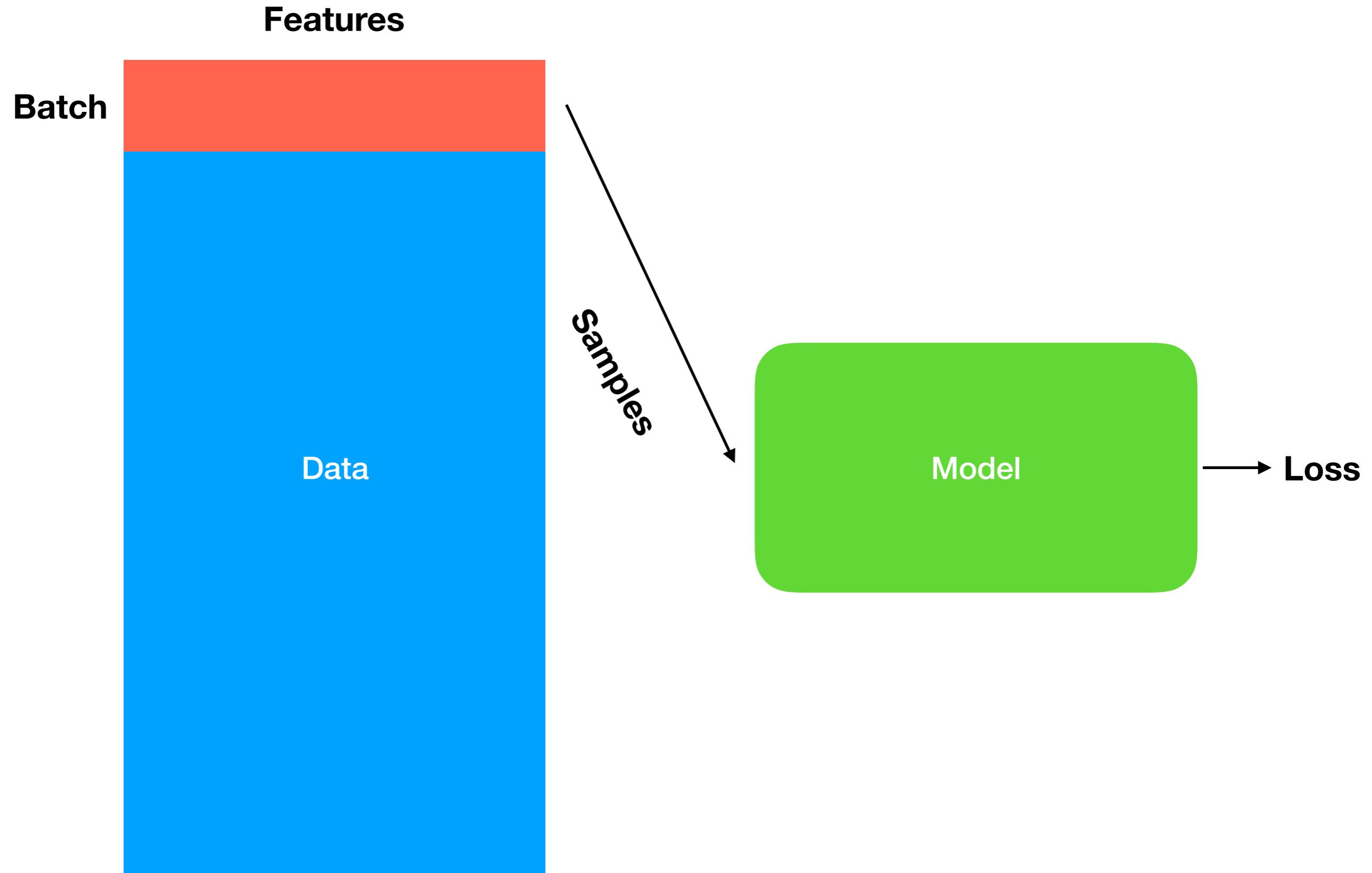
Stochastic Gradient Descent



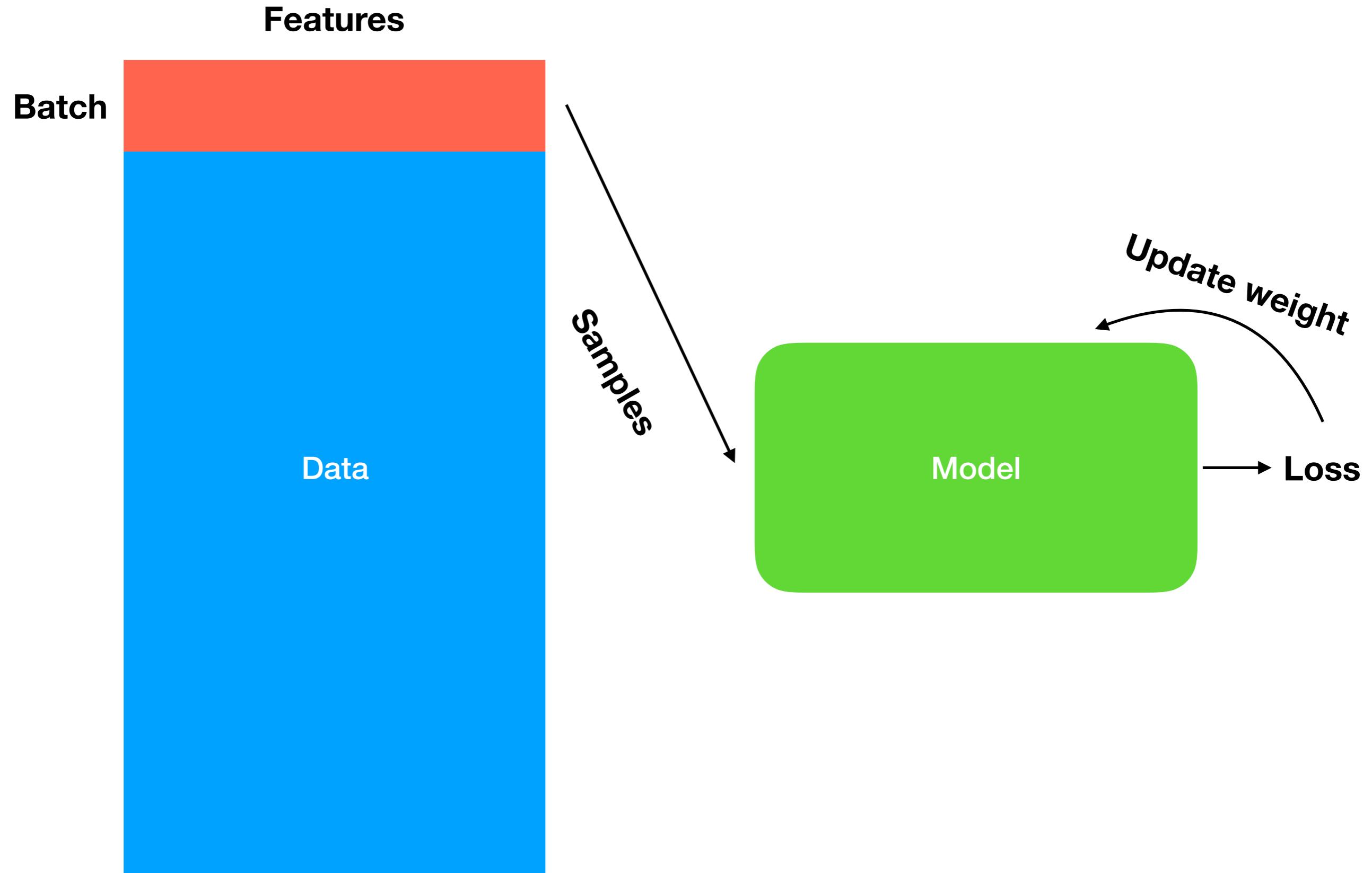
Stochastic Gradient Descent



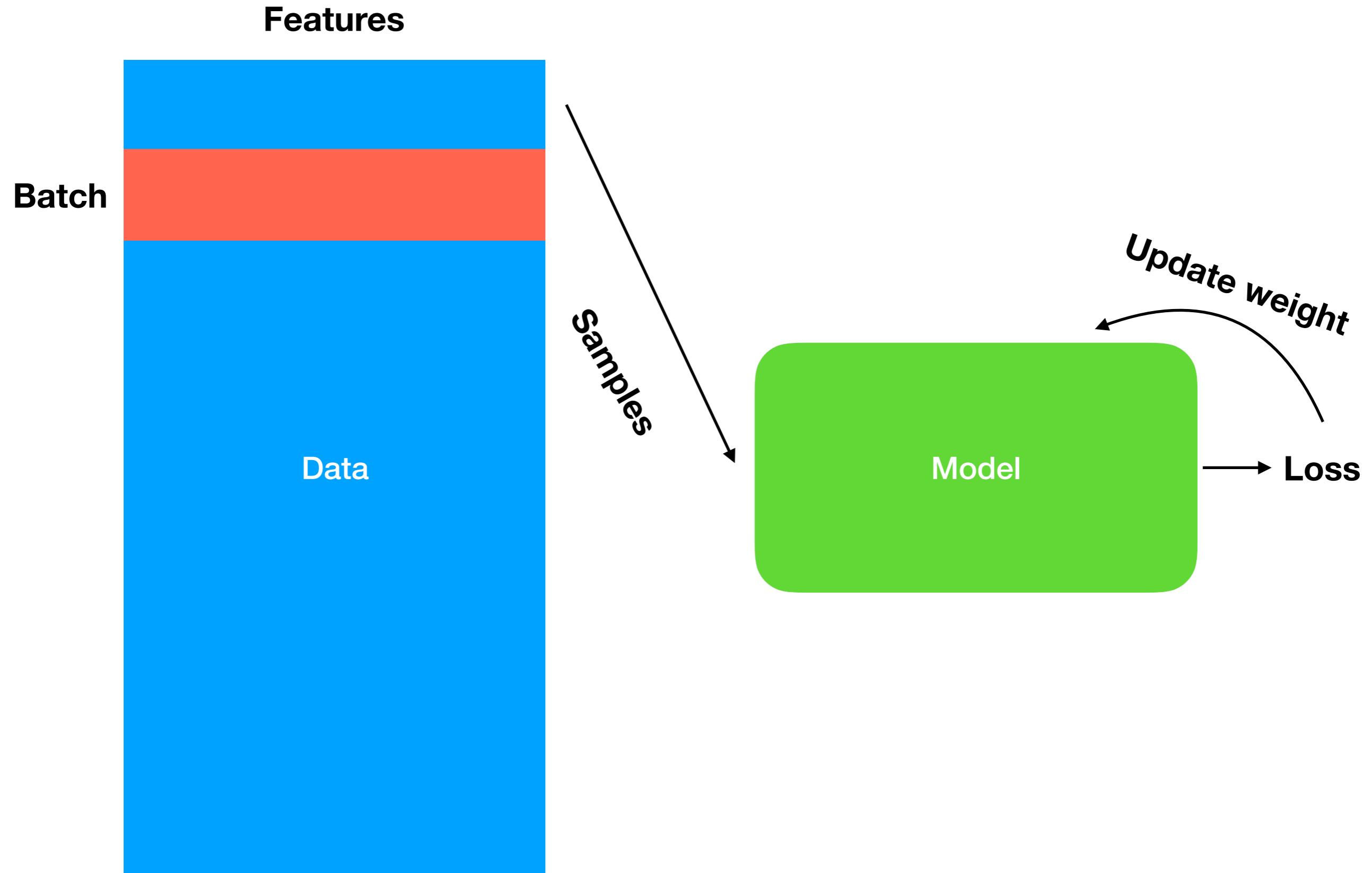
Stochastic Gradient Descent



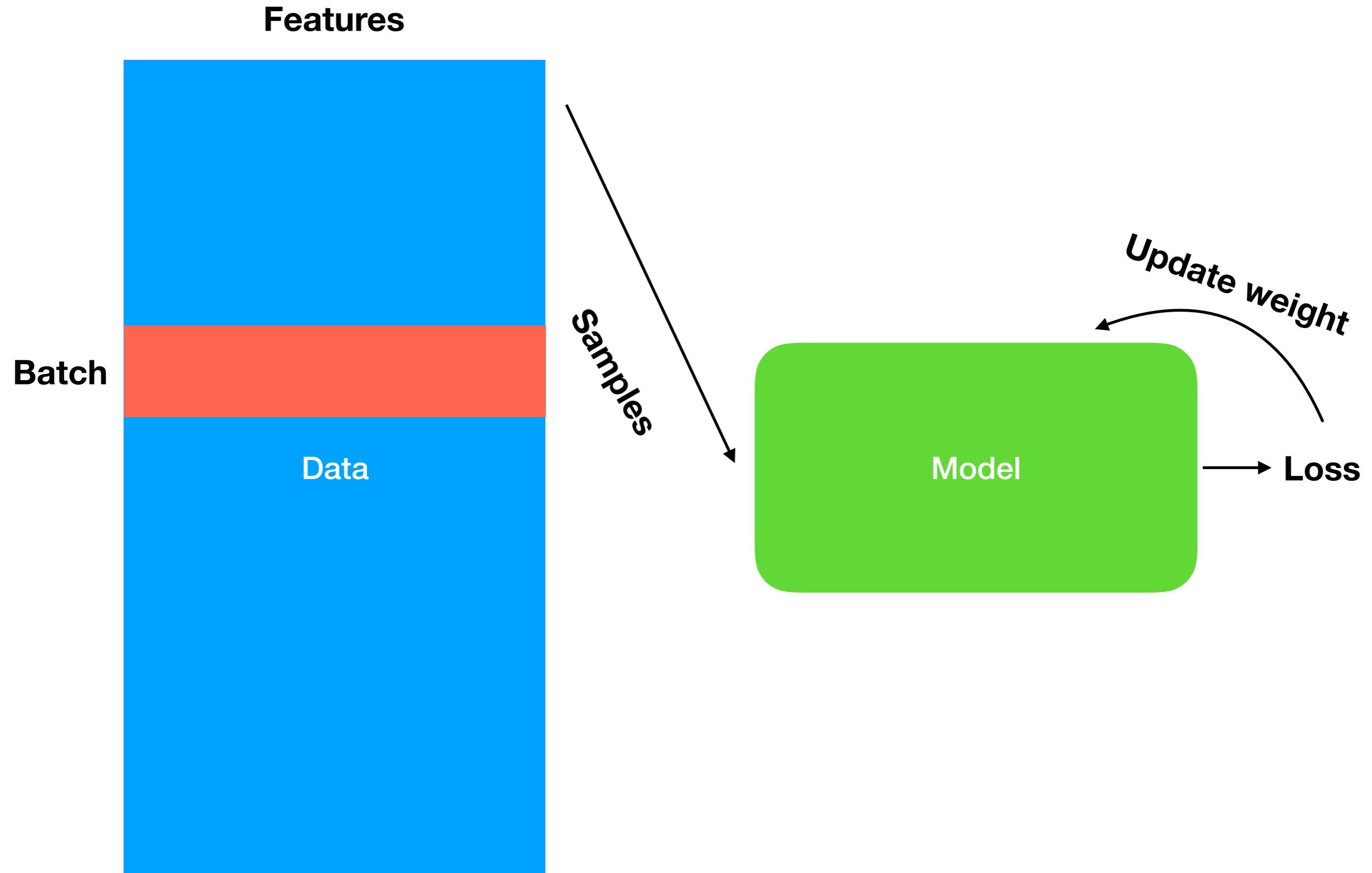
Stochastic Gradient Descent



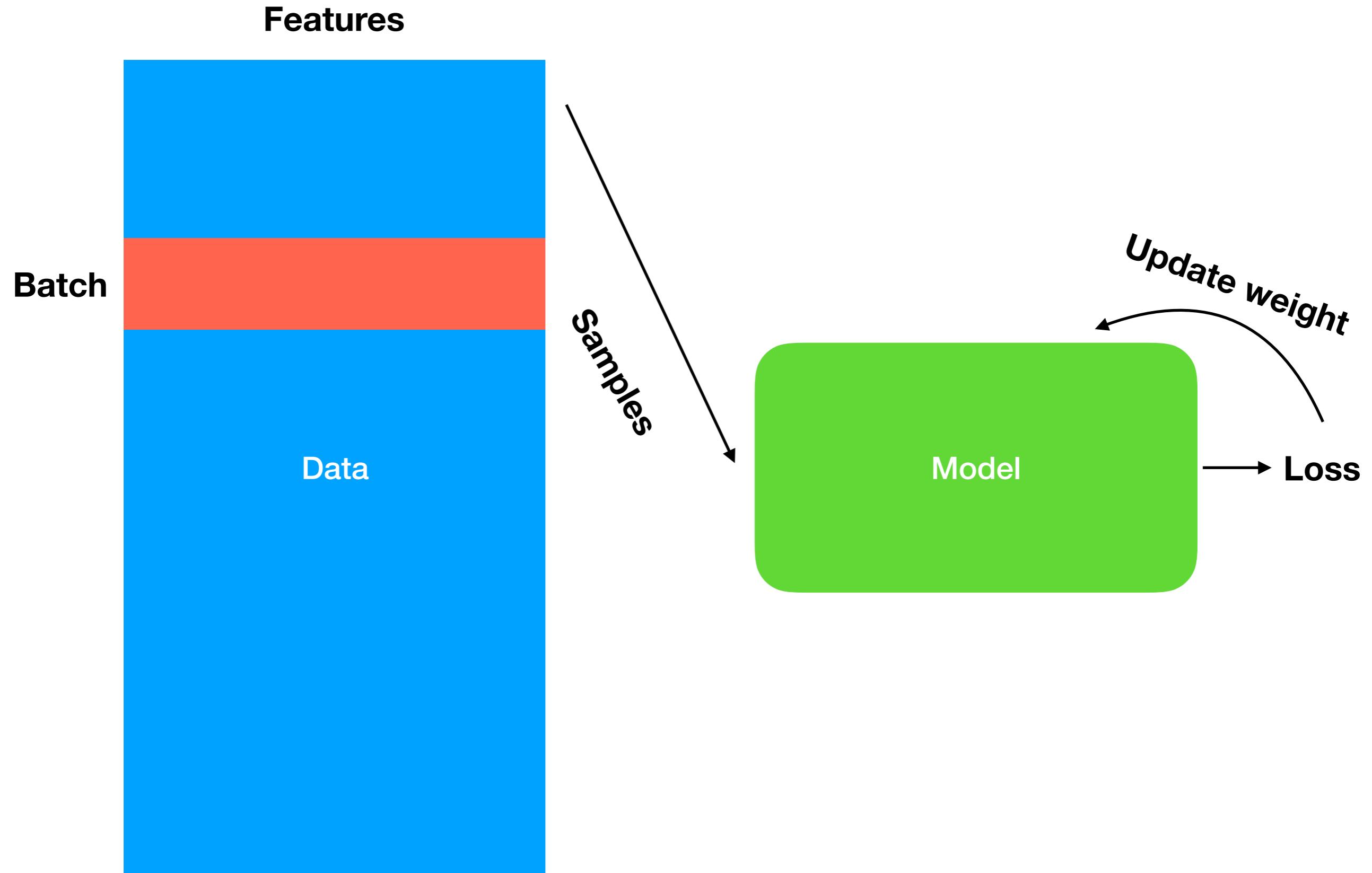
Stochastic Gradient Descent



Stochastic Gradient Descent

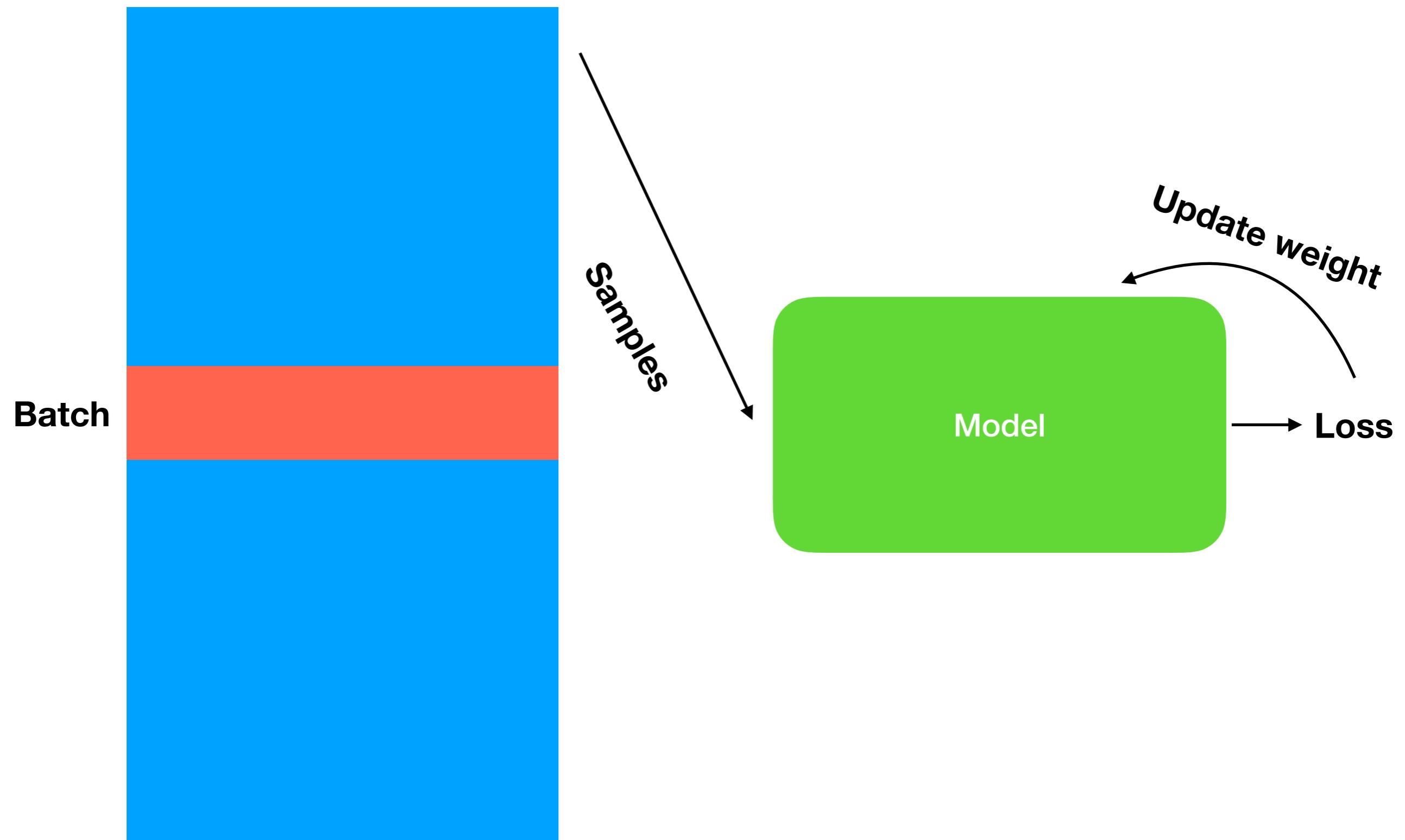


Stochastic Gradient Descent



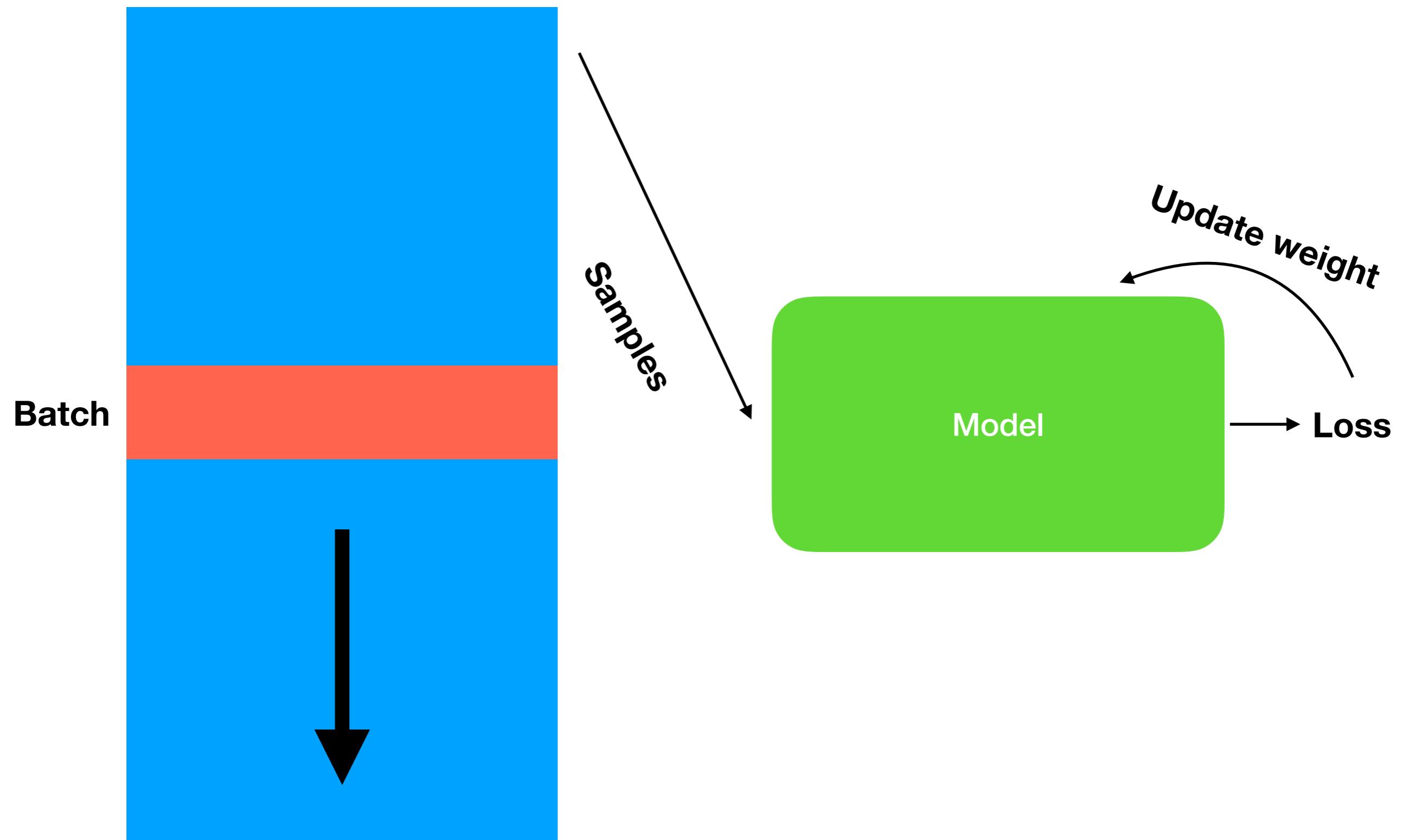
Stochastic Gradient Descent

Features



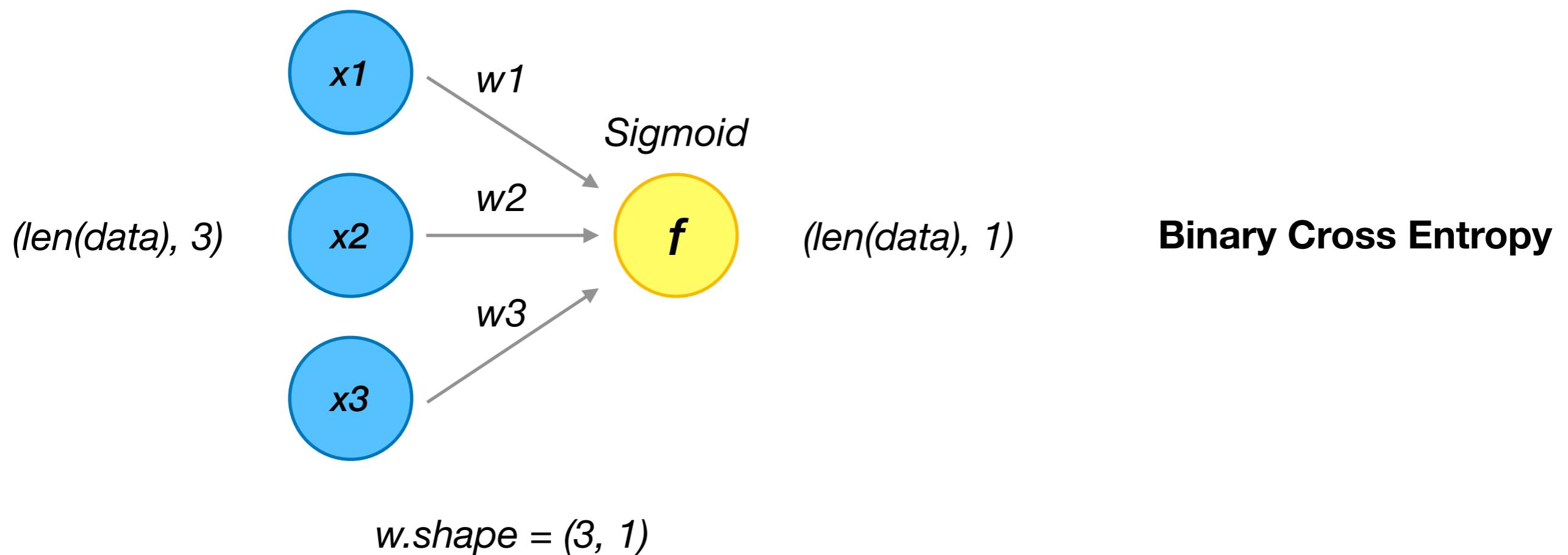
Stochastic Gradient Descent

Features



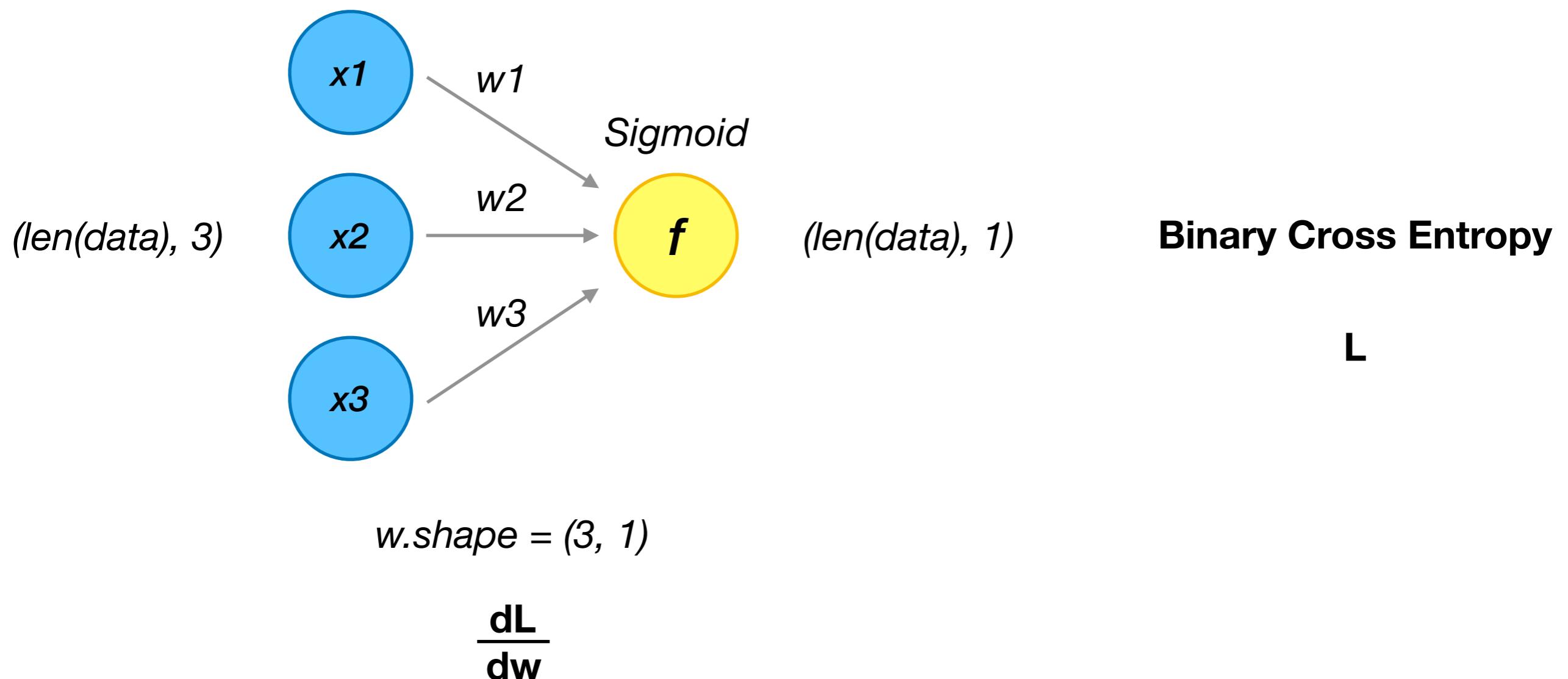
Logistic Regression

Train



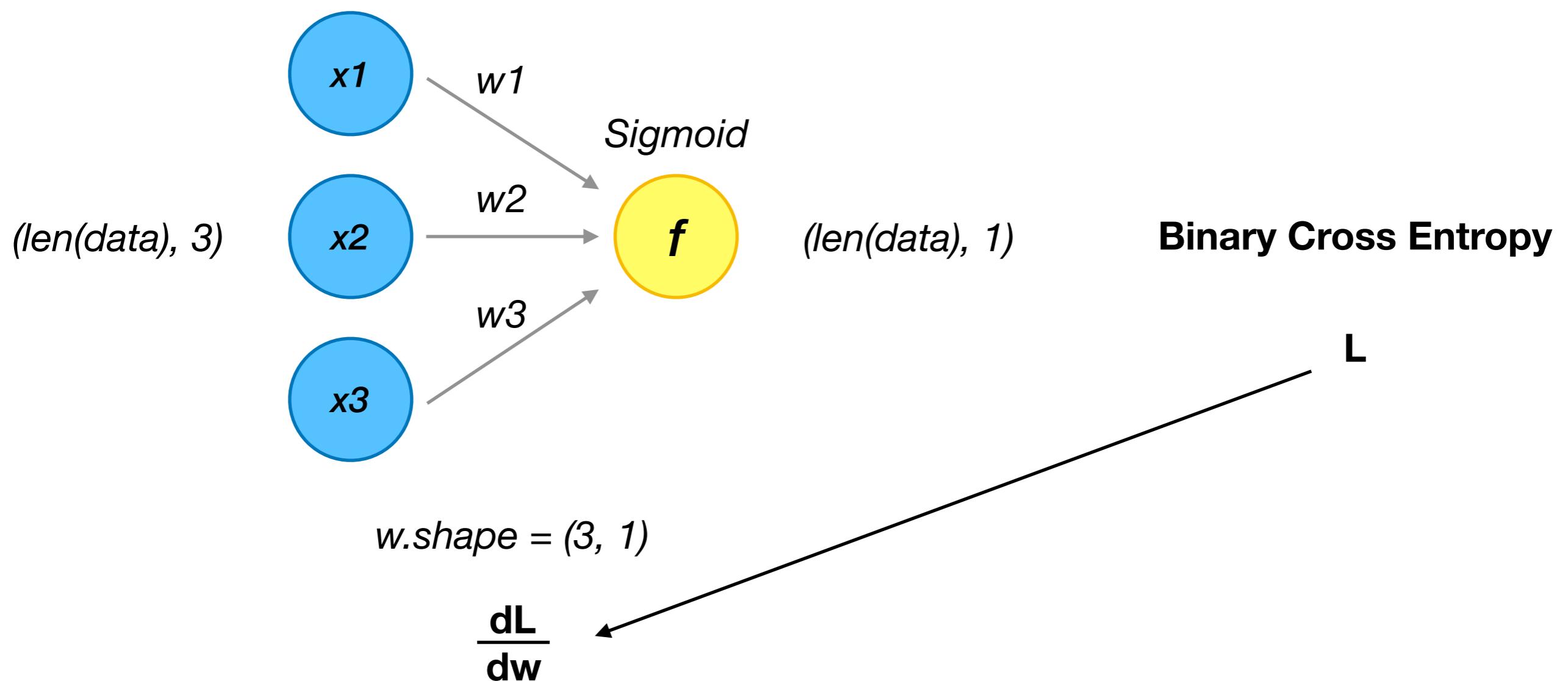
Logistic Regression

Train



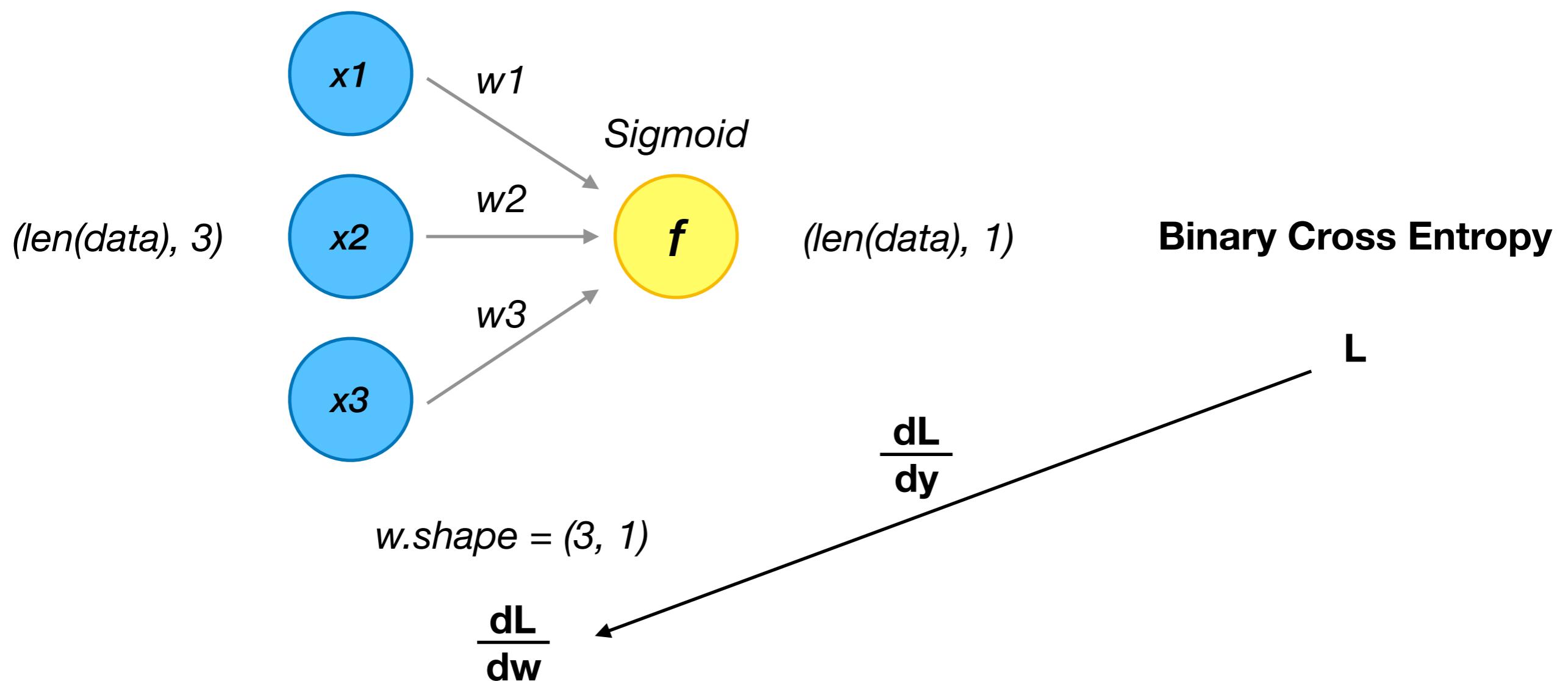
Logistic Regression

Train



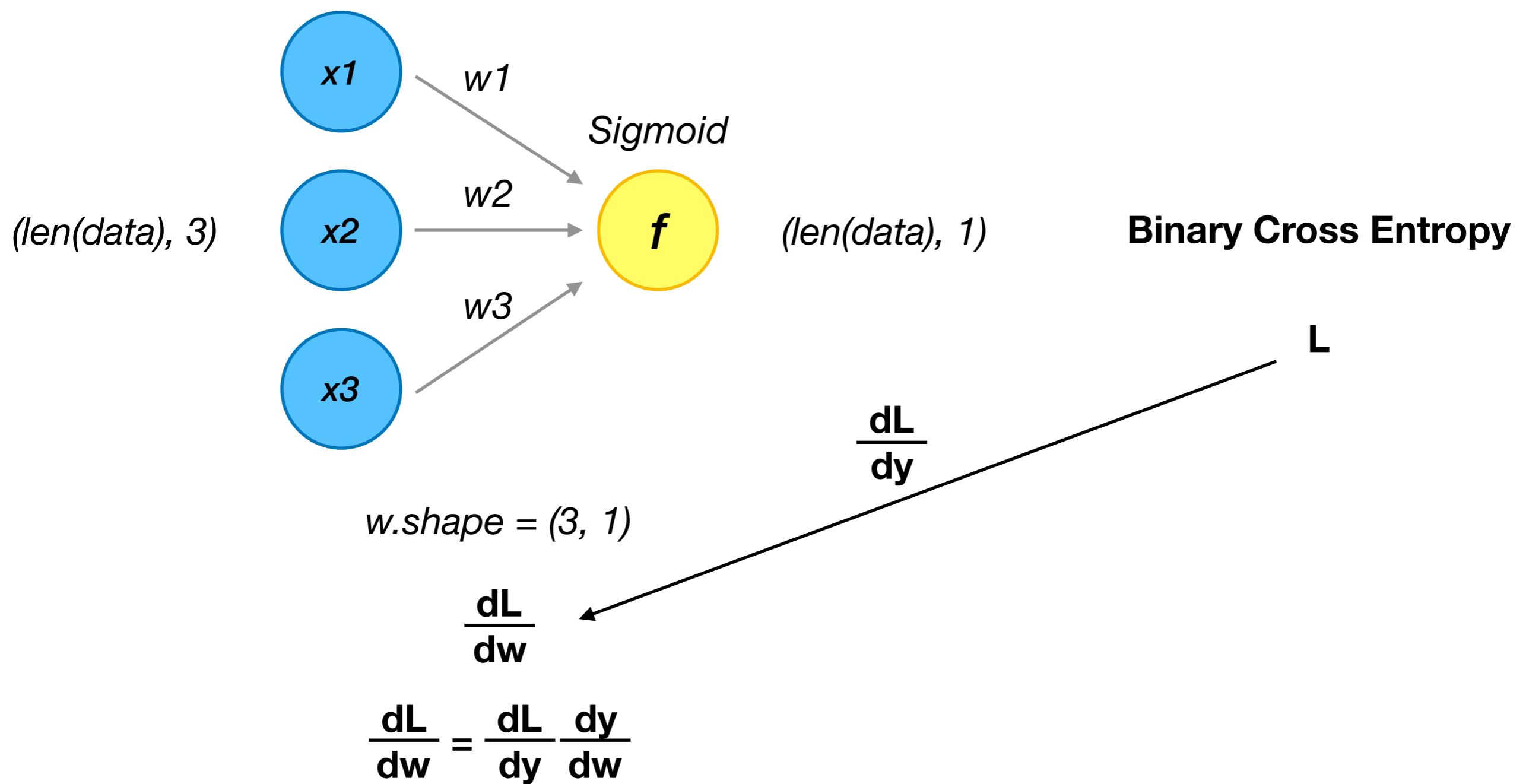
Logistic Regression

Train



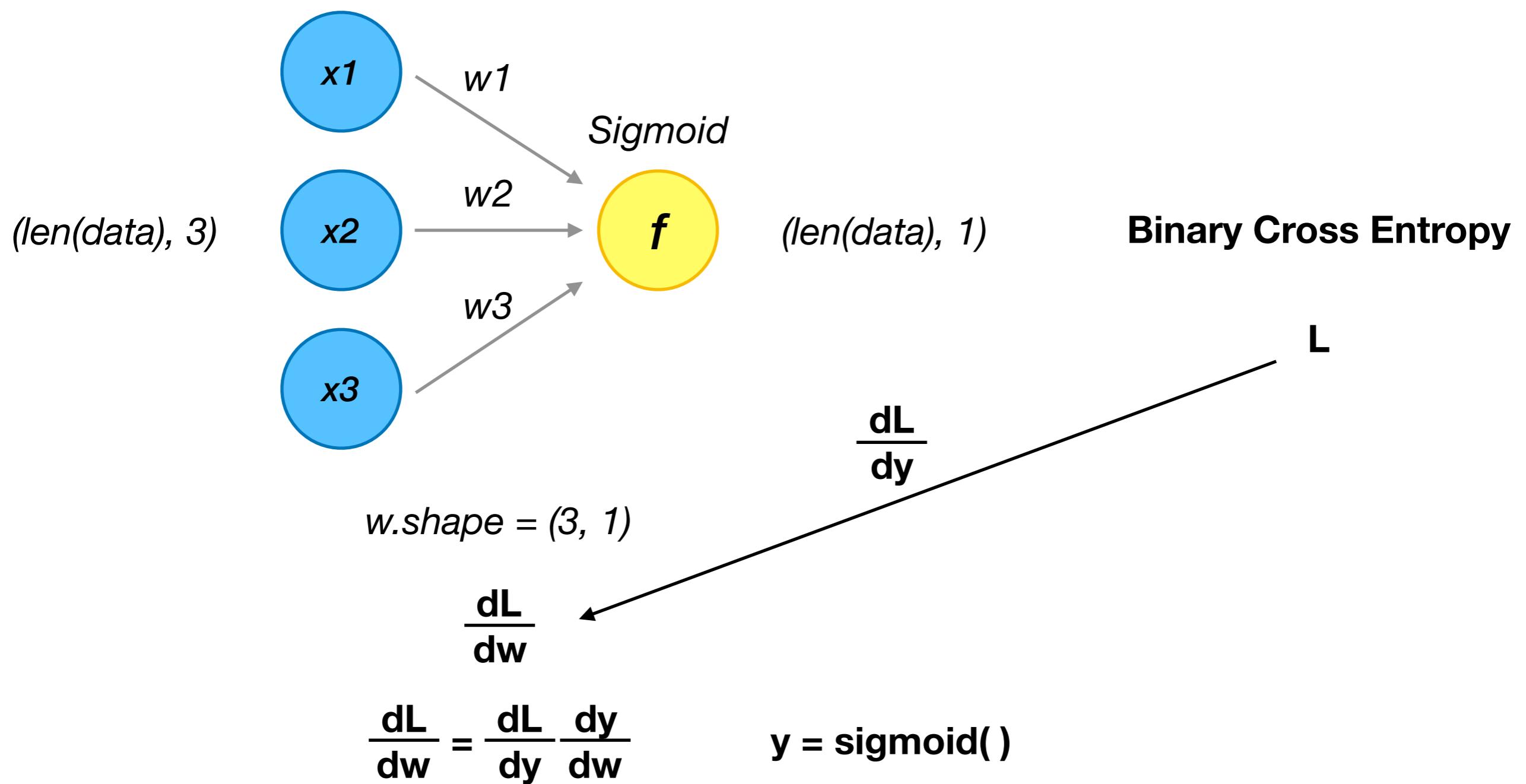
Logistic Regression

Train



Logistic Regression

Train



Regularization

BCELoss = - (y * log(pred) + (1 - y) * log(1 - pred))

Regularization

$$\text{BCELoss} = - (y * \log(\text{pred}) + (1 - y) * \log(1 - \text{pred})) - \text{reg} * R(W)$$

reg - hyperparameter

Regularization

BCELoss = - (y * log(pred) + (1 - y) * log(1 - pred)) - reg * R(W)

reg - hyperparameter

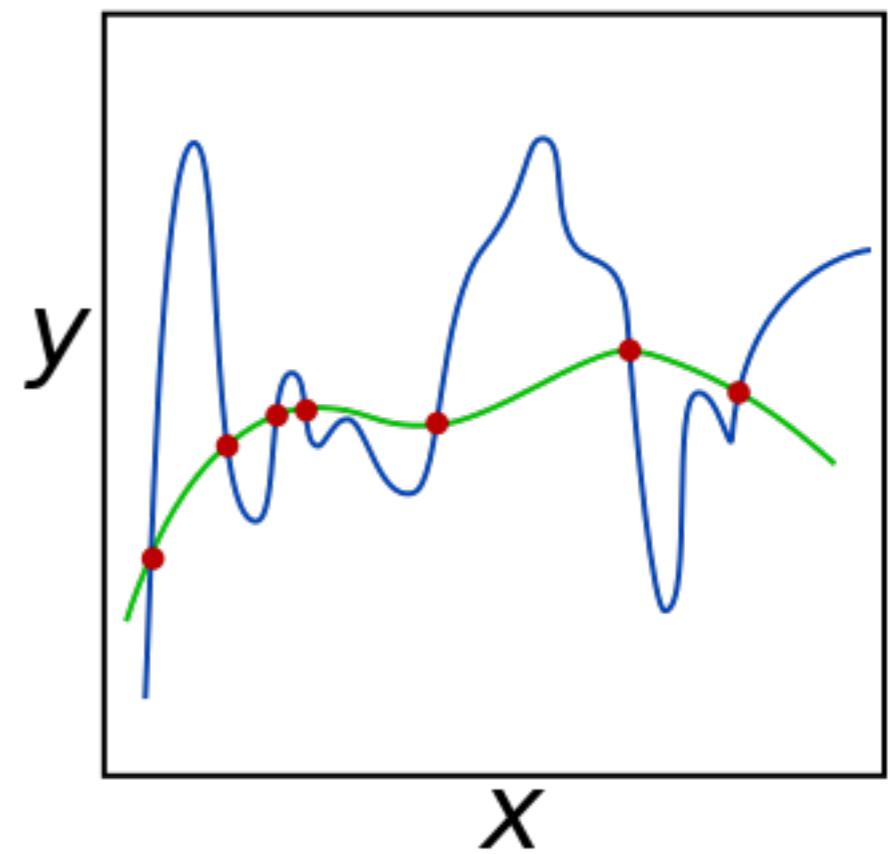
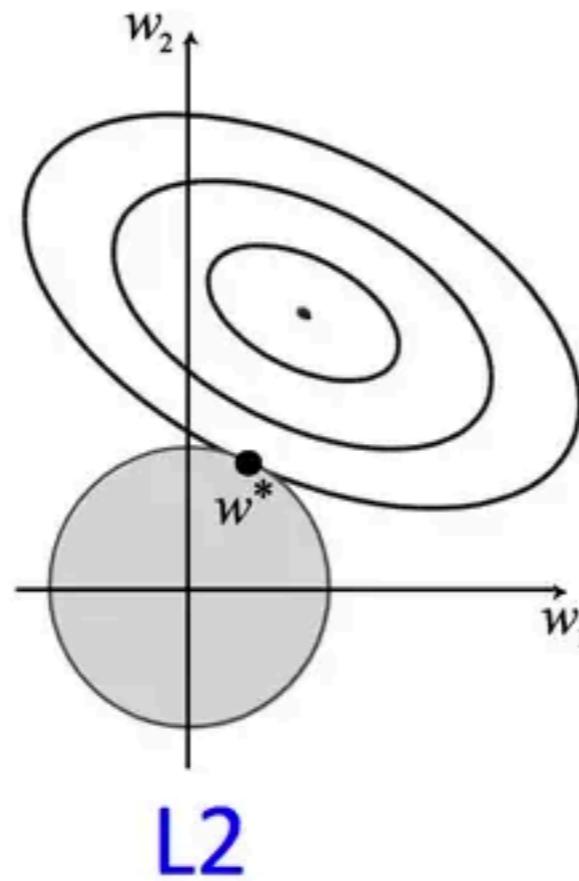
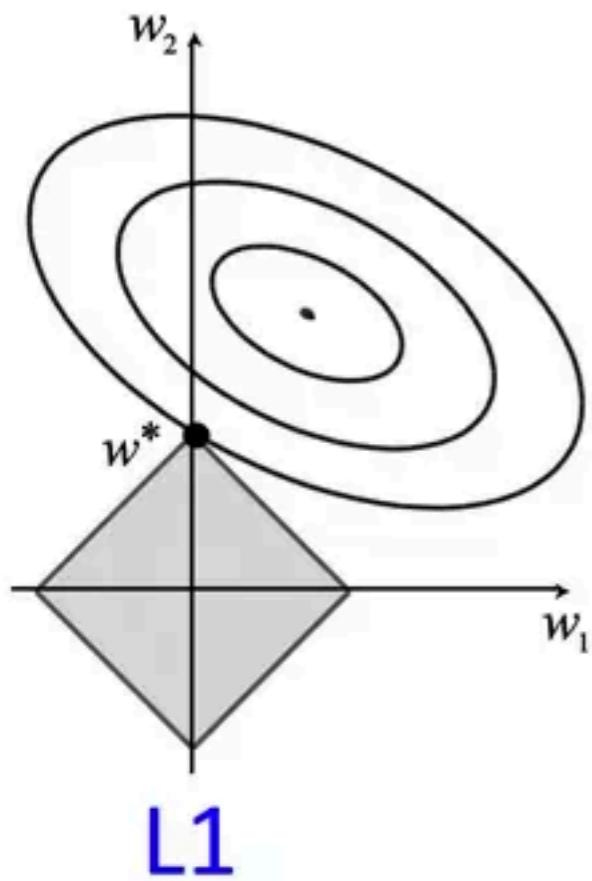
$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

Regularization

$$\text{L2} \quad R(W) = \sum_k \sum_l W_{k,l}^2$$

$$\text{L1} \quad R(W) = \sum_k \sum_l |W_{k,l}|$$

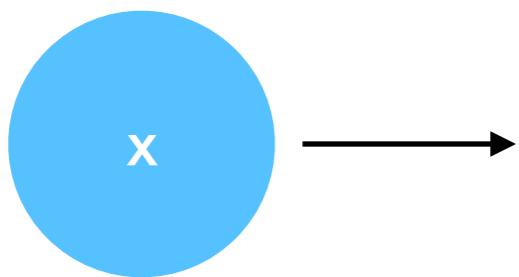


Neural Networks

Forward

Neural Networks

Forward



Neural Networks

Forward



Neural Networks

Forward



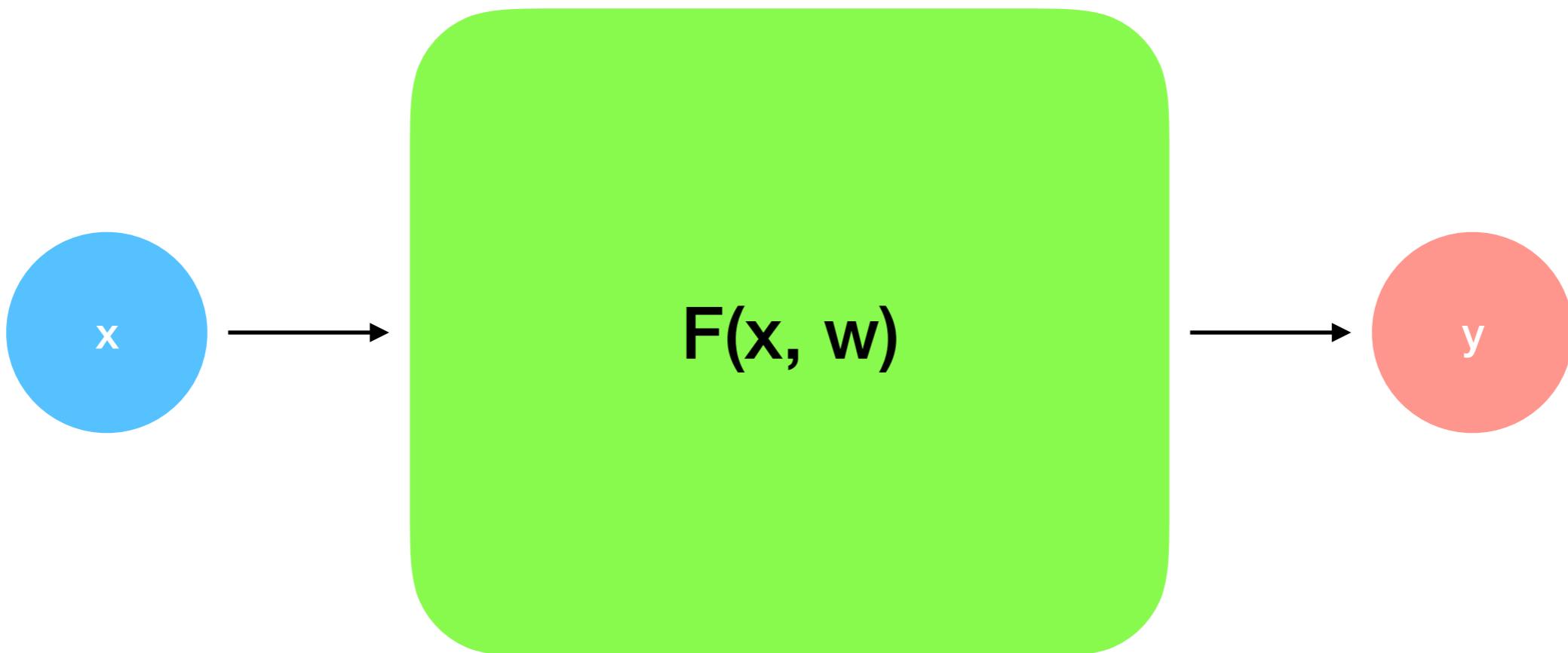
Neural Networks

Forward



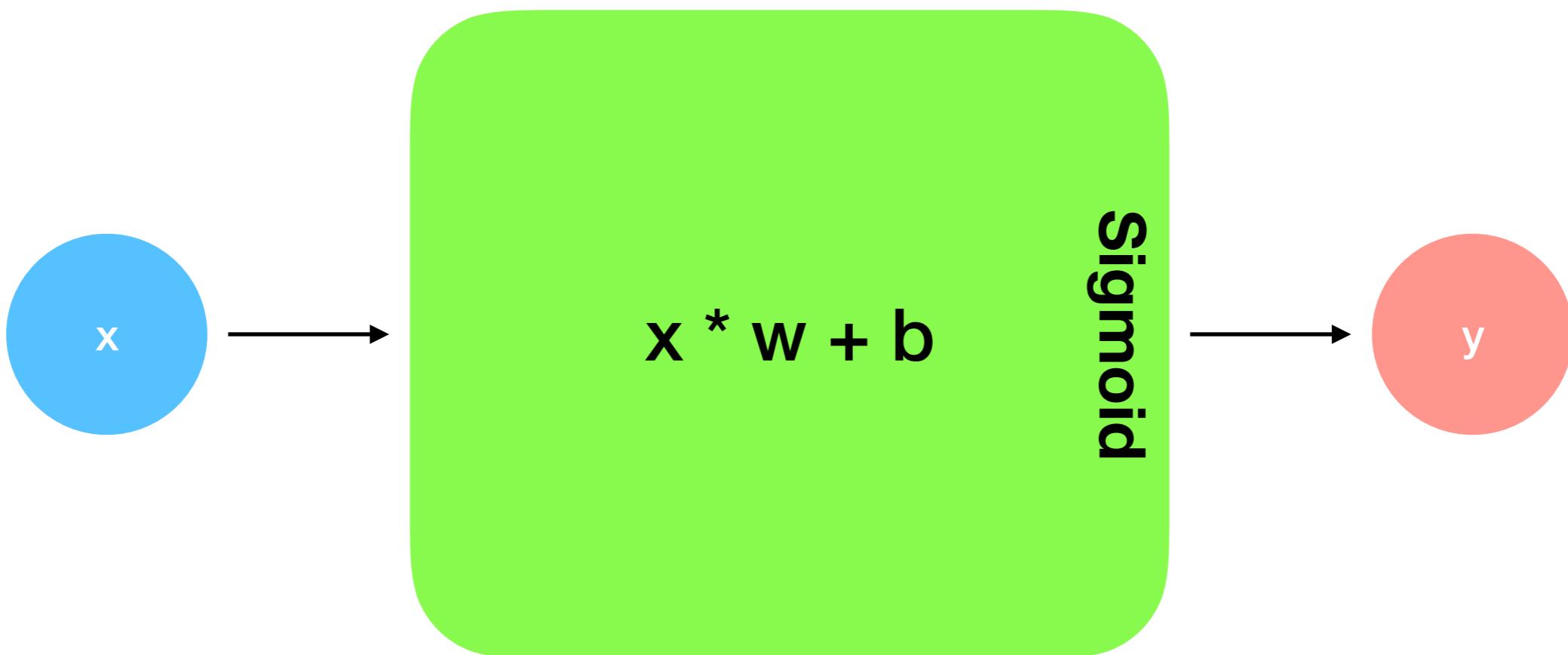
Neural Networks

Forward



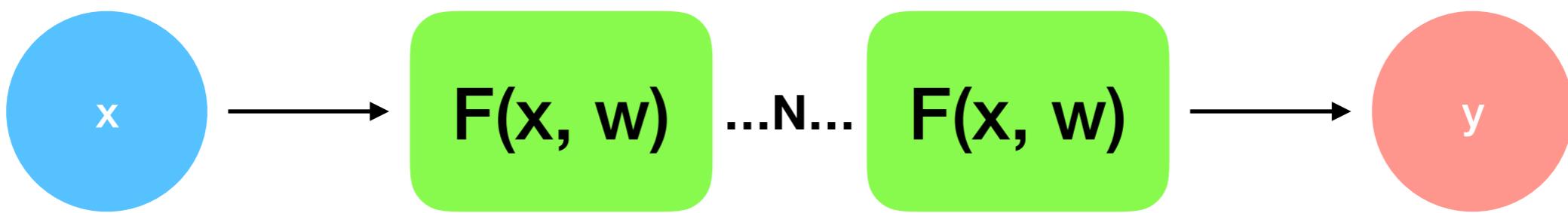
Neural Networks

Forward



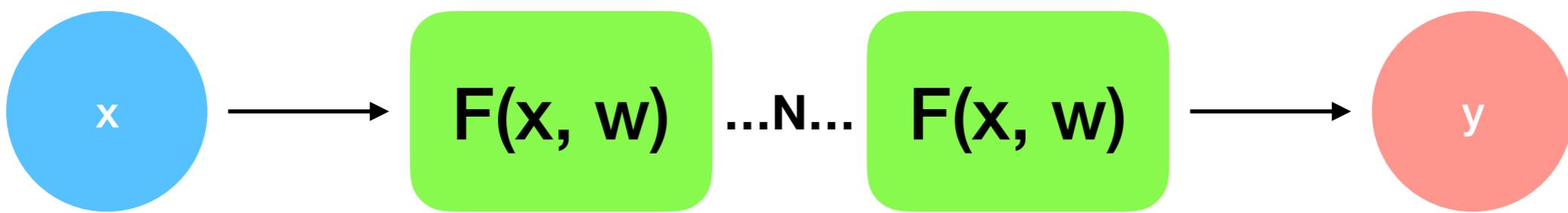
Neural Networks

Forward



Neural Networks

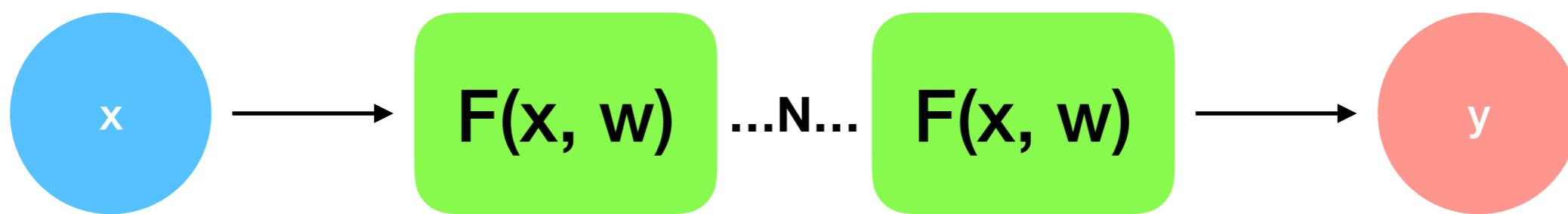
Forward



$$F_n(F_3(F_2(F_1(x, w_1), w_2), w_3), w_n)$$

Neural Networks

Forward

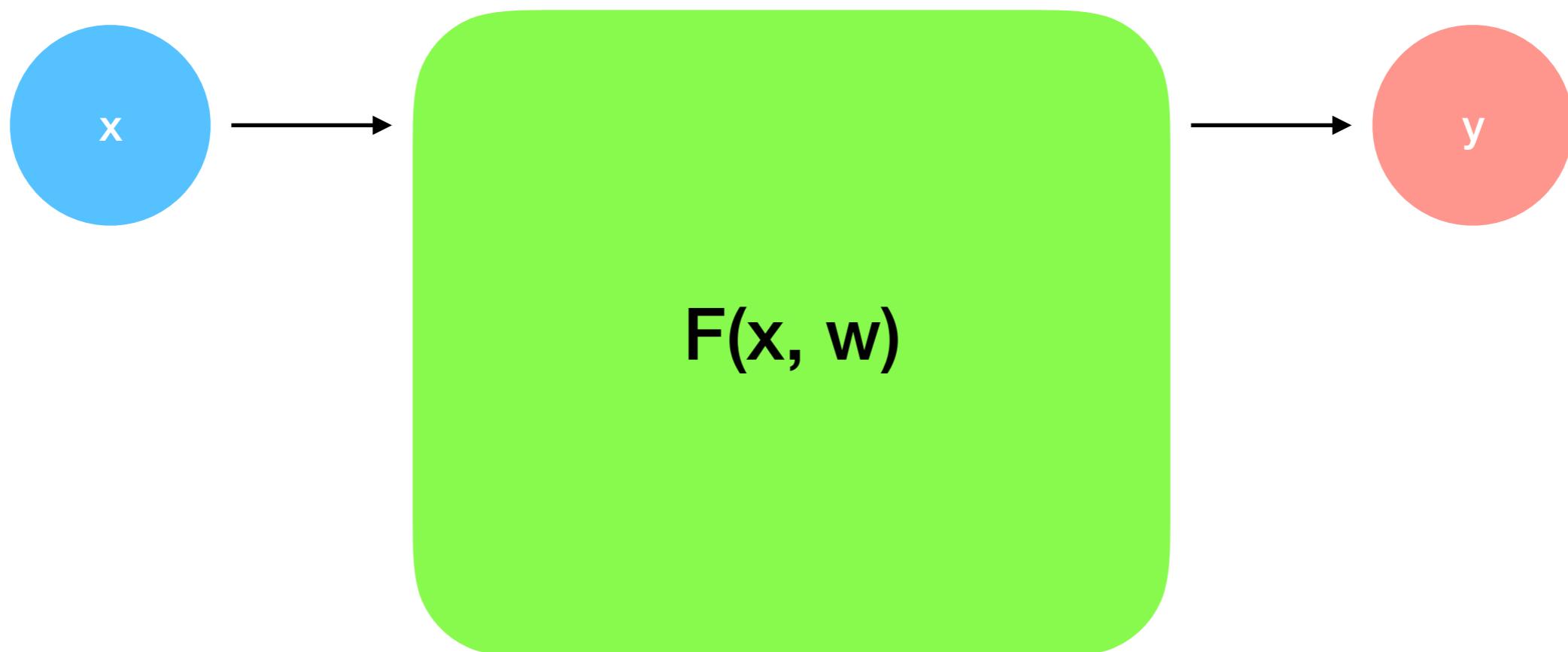


$$F_n(F_3(F_2(F_1(x, w_1), w_2), w_3), w_n)$$

$$F(x, w) = \text{non-linearity}(x * w + b)$$

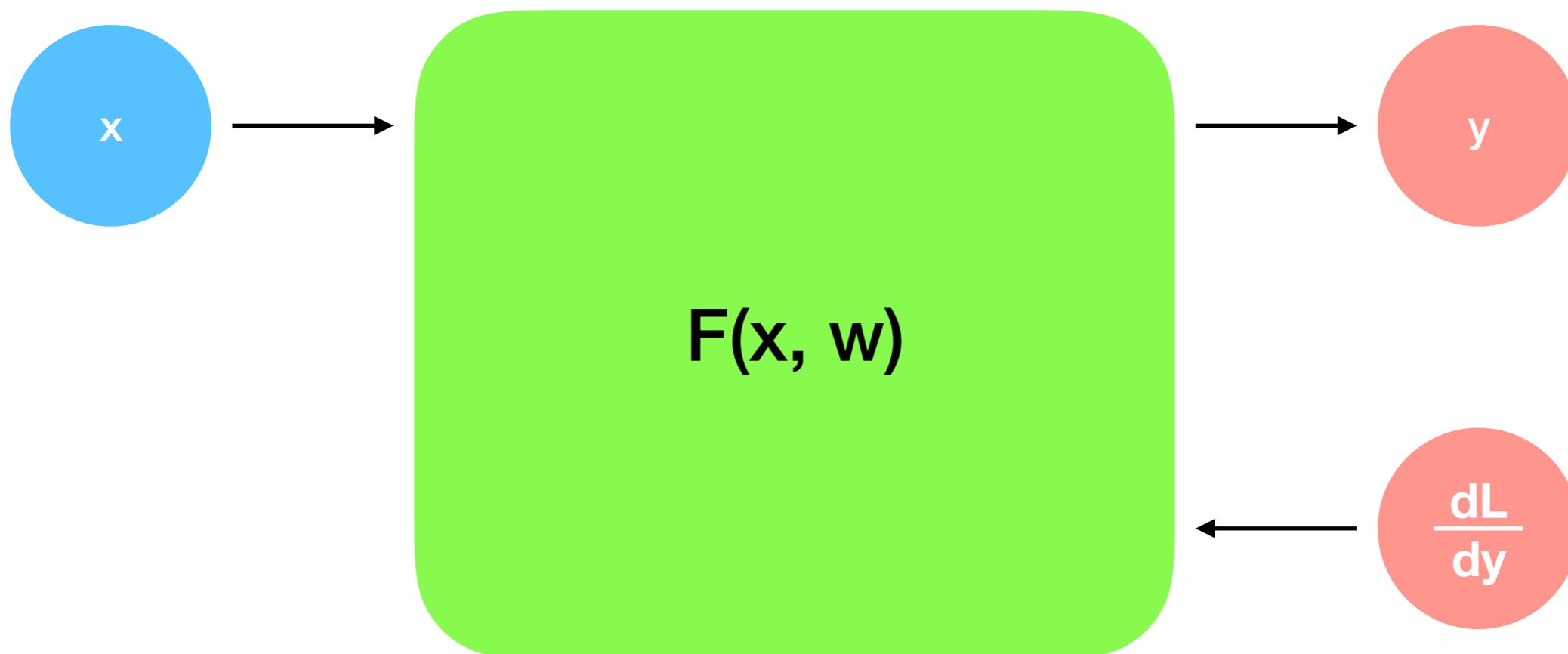
Neural Networks

Backward



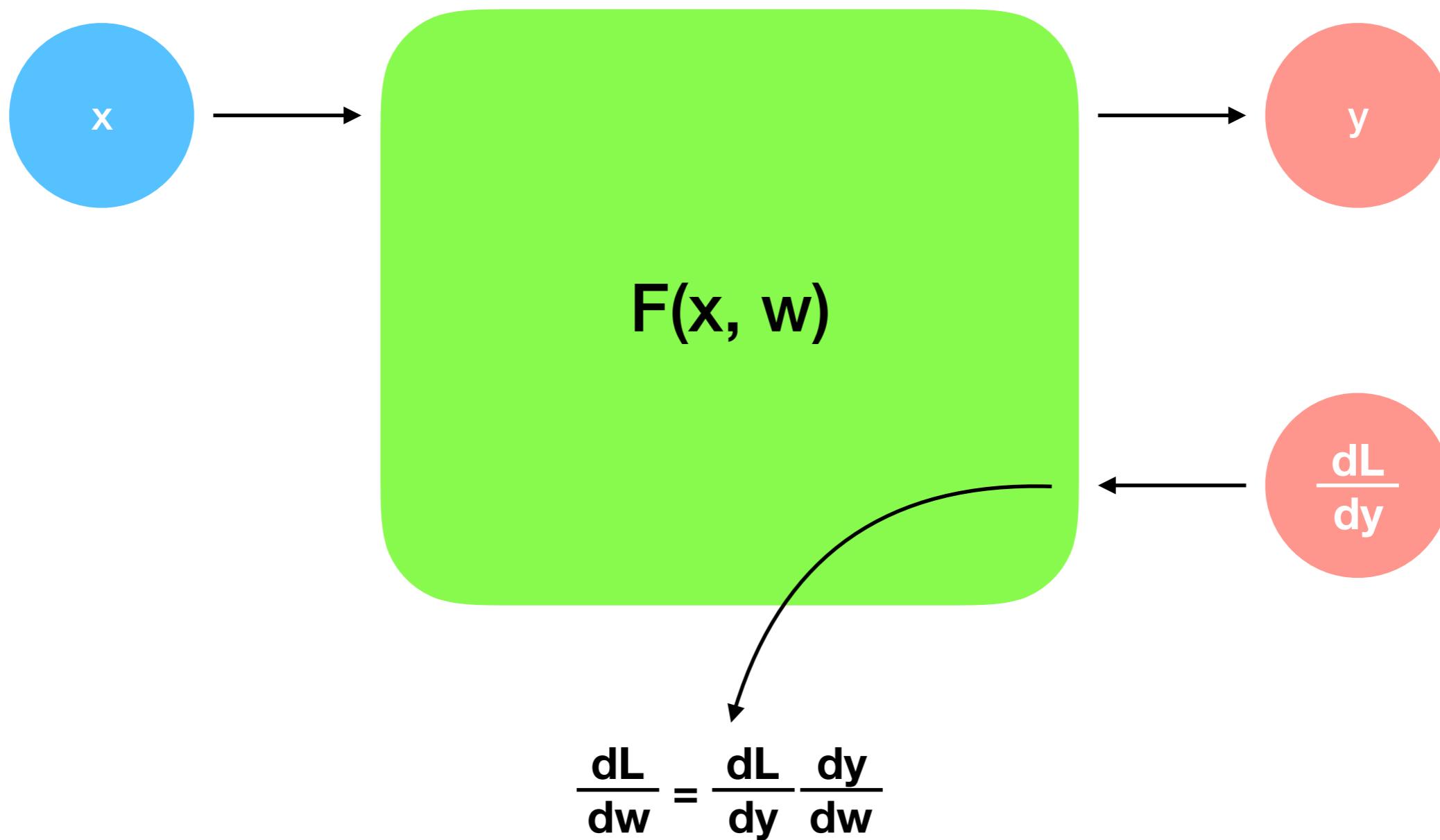
Neural Networks

Backward



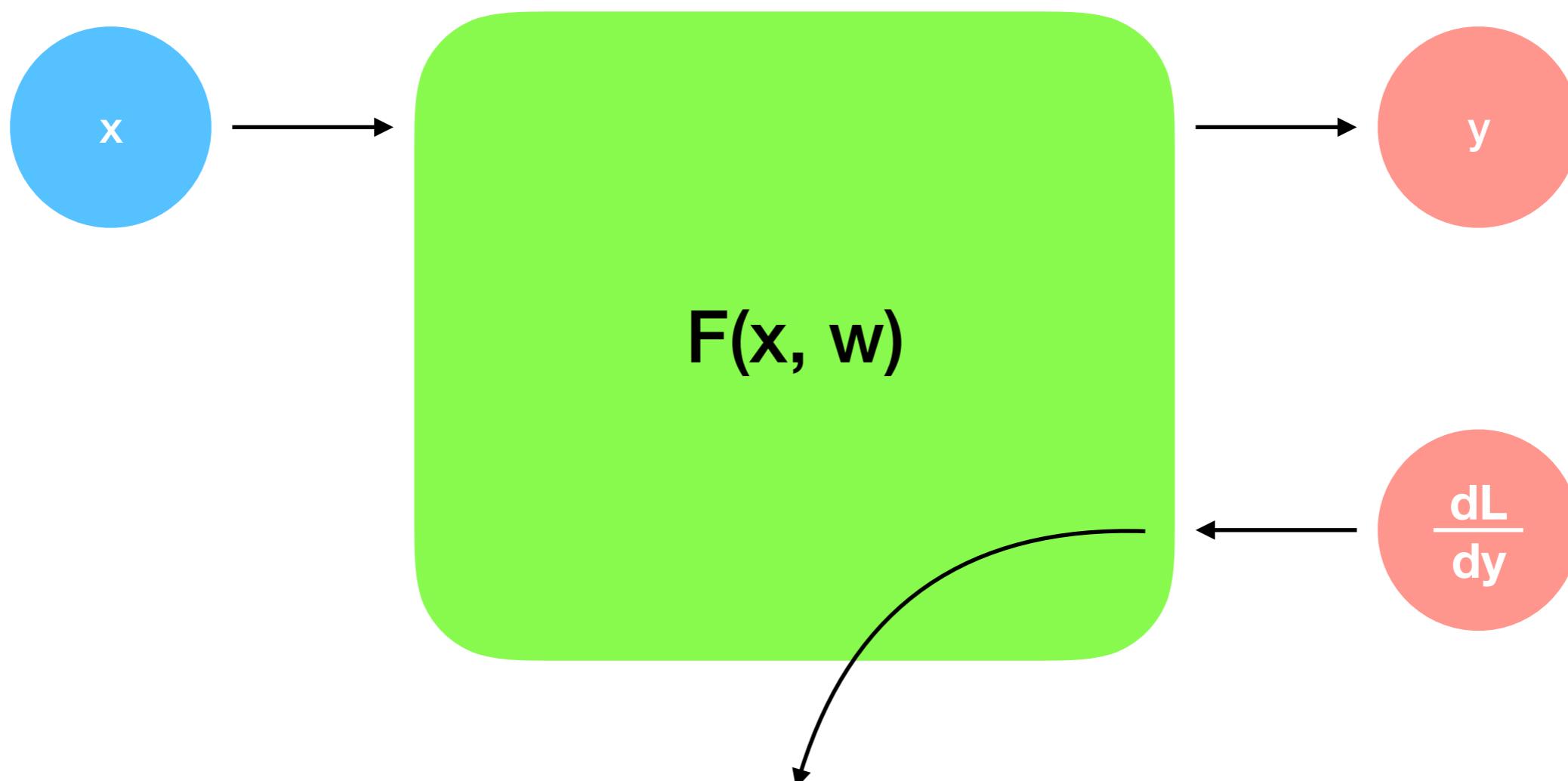
Neural Networks

Backward



Neural Networks

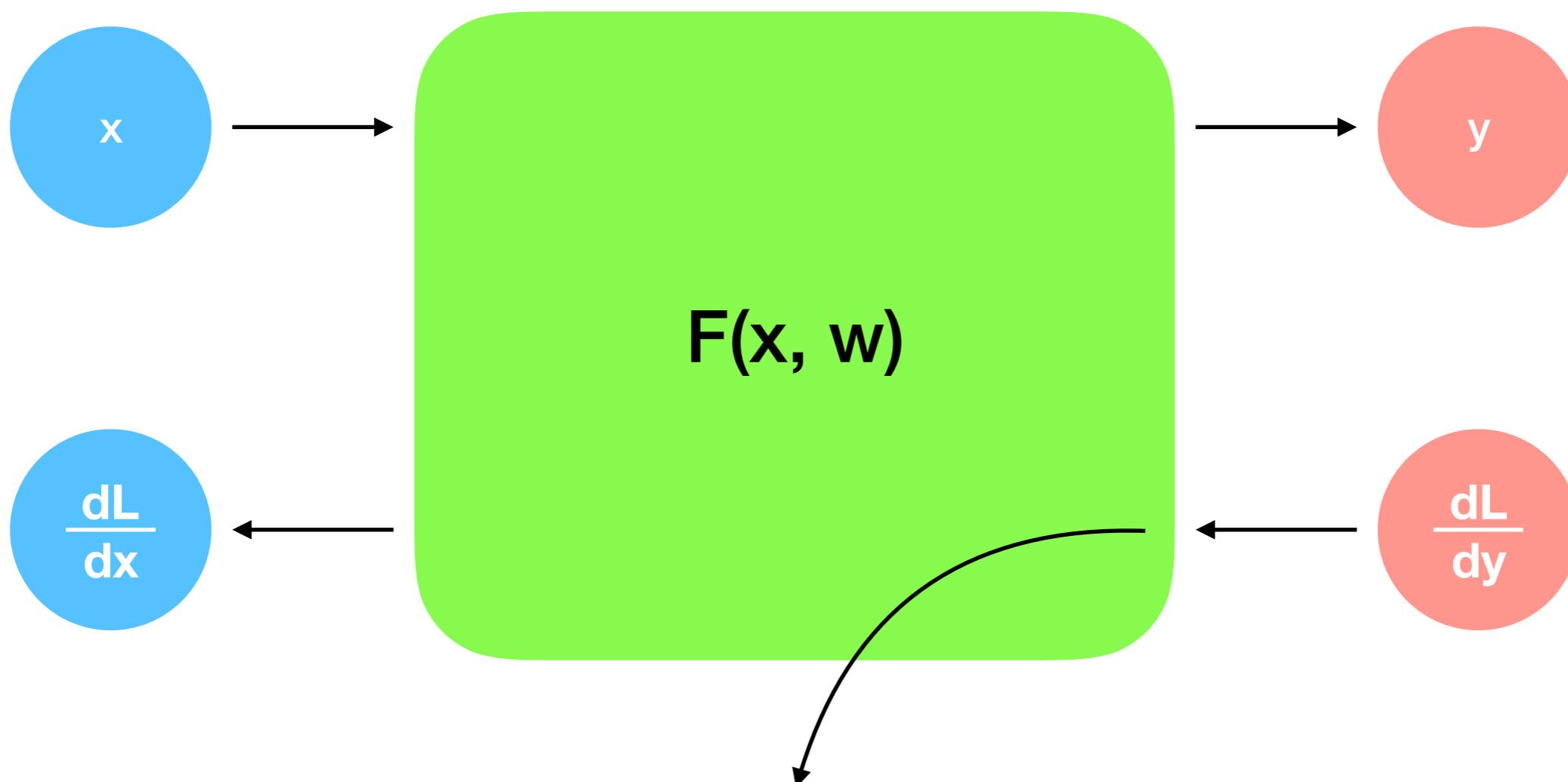
Backward



$$w = w - \text{learning_rate} * \frac{dL}{dw}$$

Neural Networks

Backward

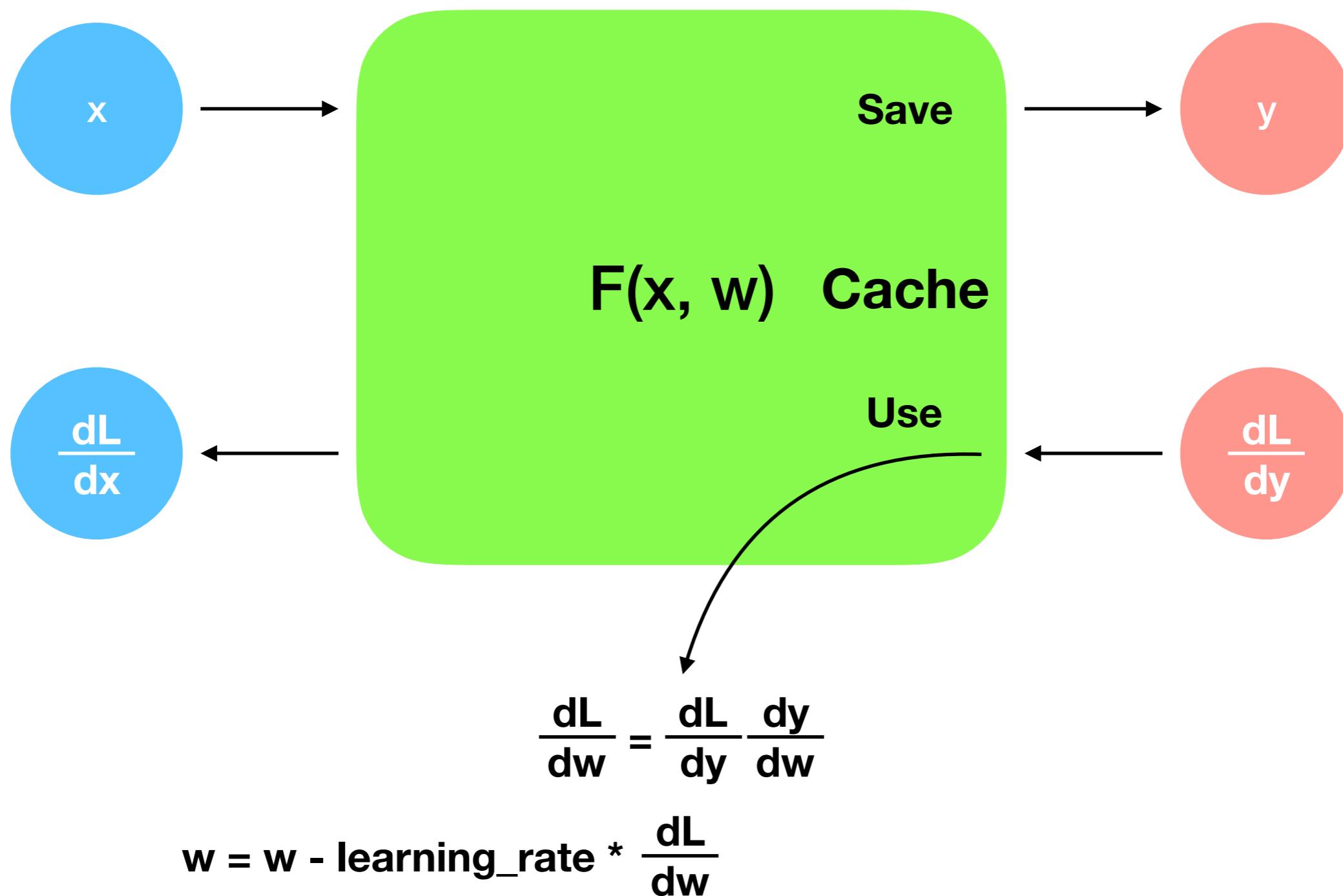


$$\frac{dL}{dw} = \frac{dL}{dy} \frac{dy}{dw}$$

$$w = w - \text{learning_rate} * \frac{dL}{dw}$$

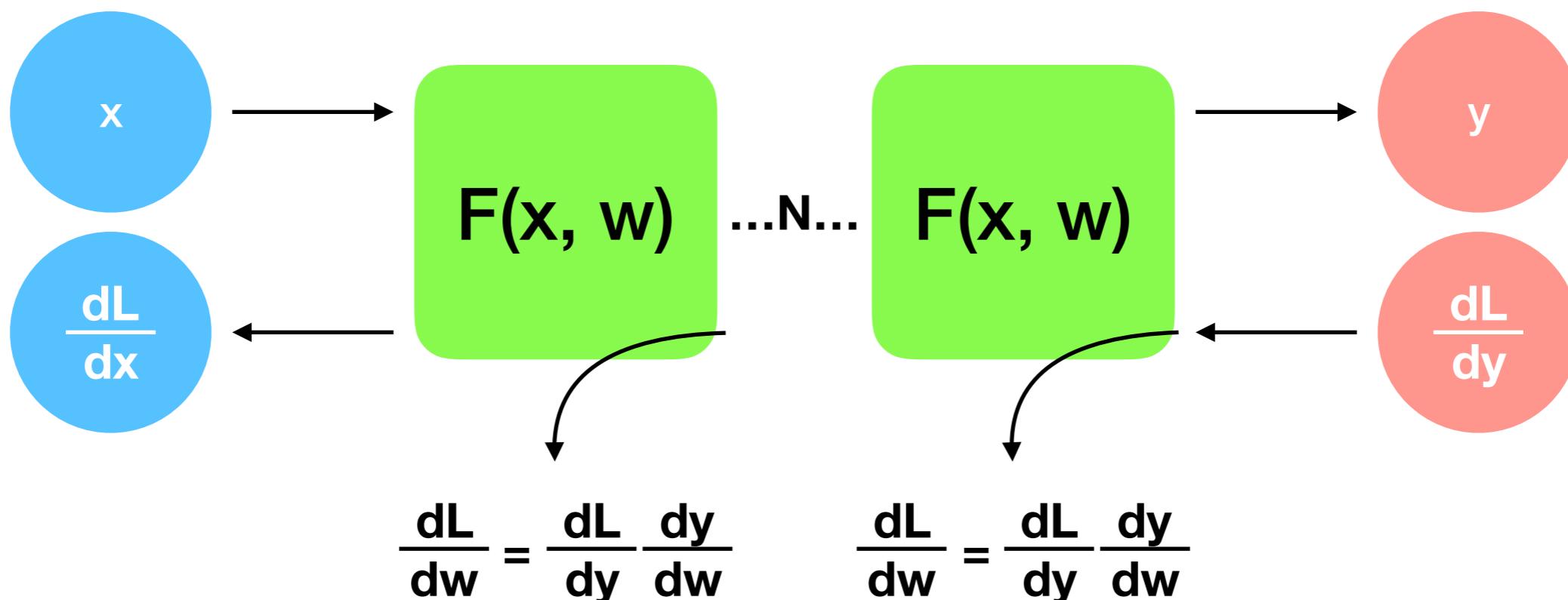
Neural Networks

Backward



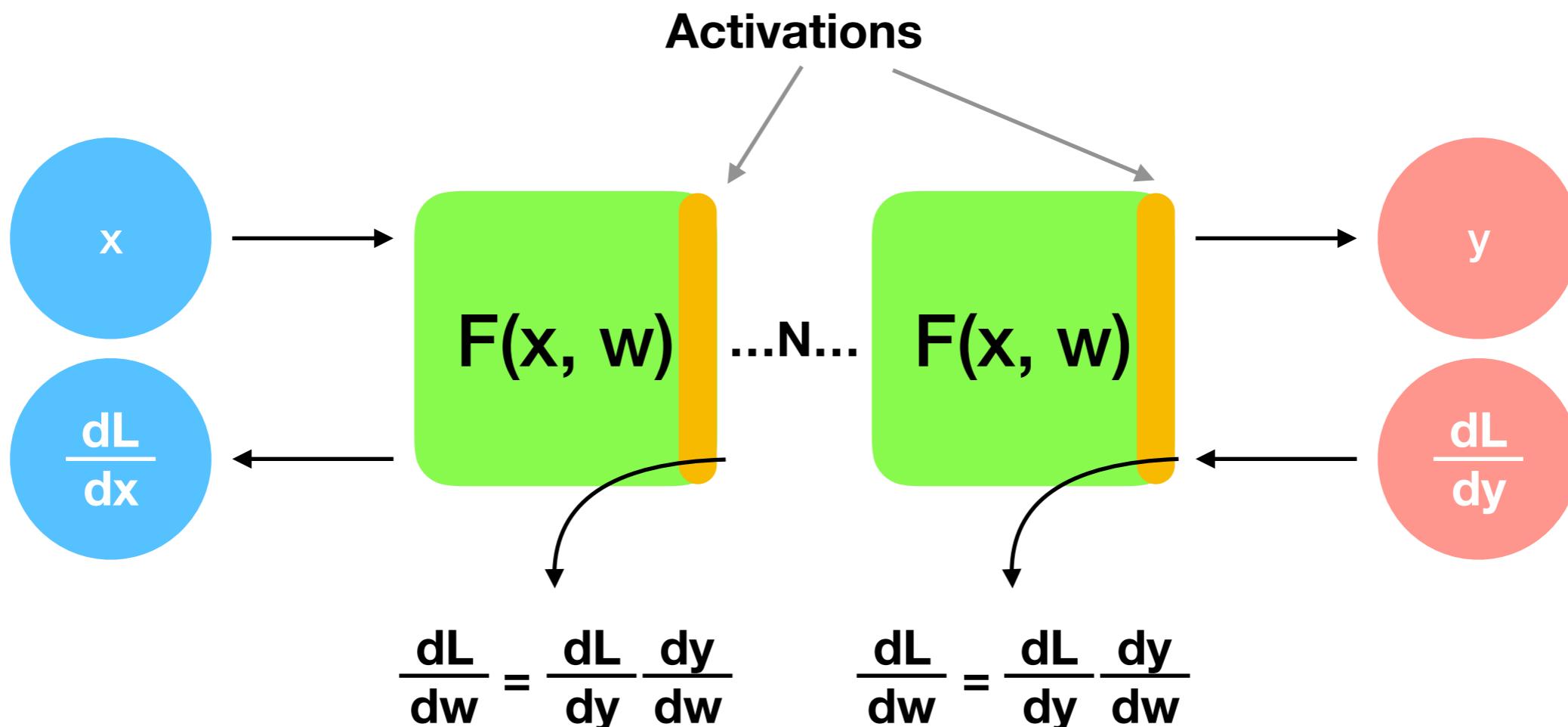
Neural Networks

Backward

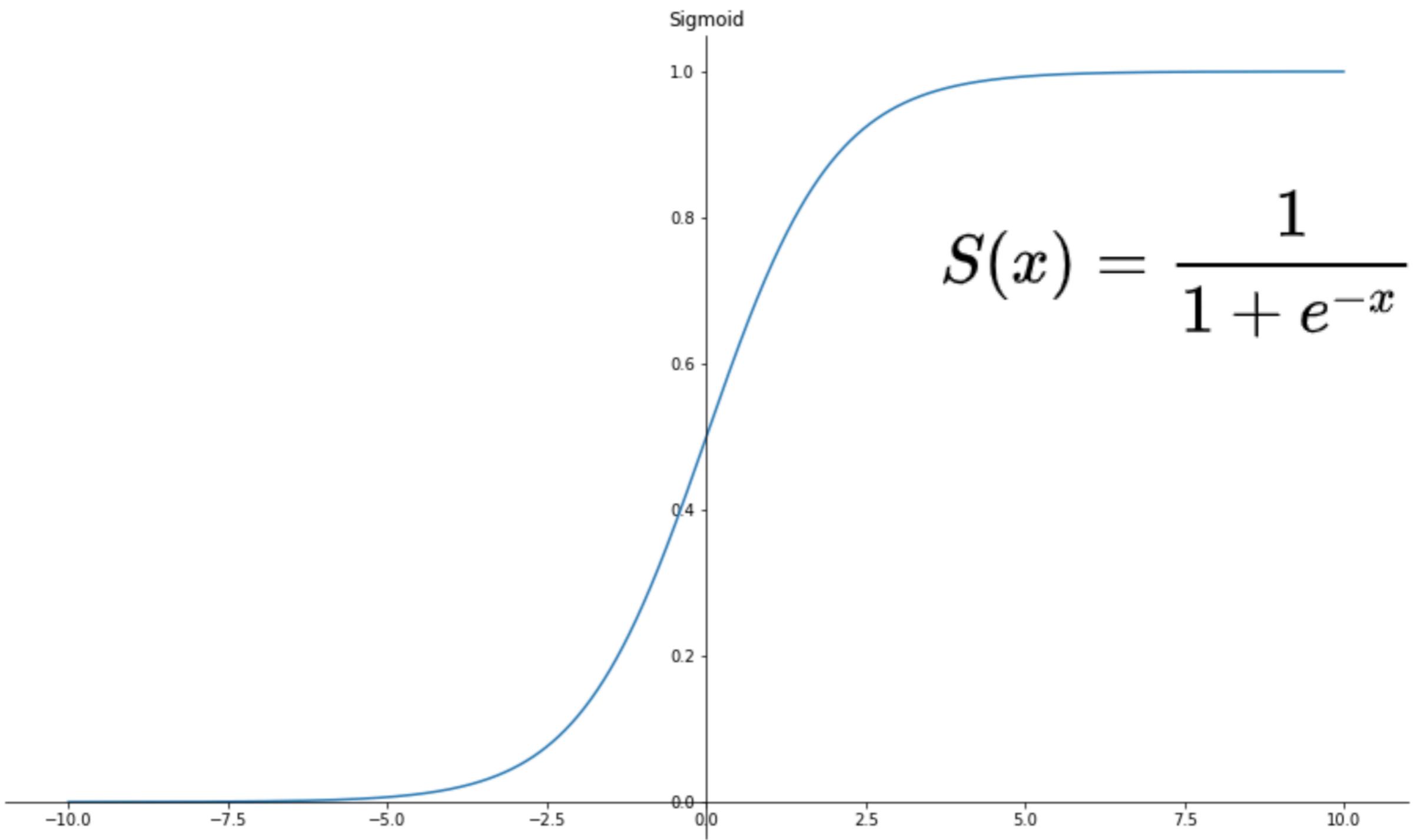


Neural Networks

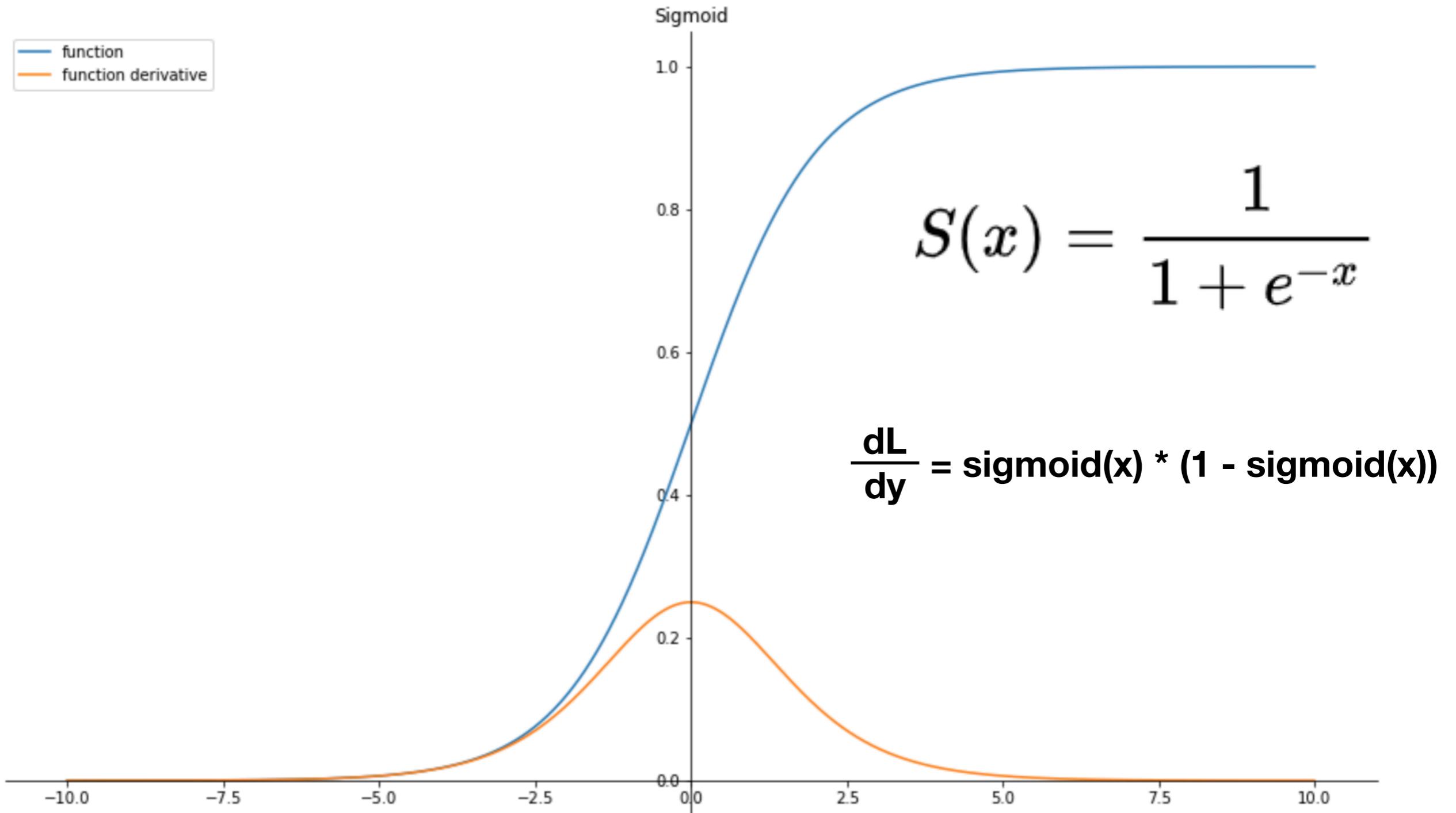
Backward



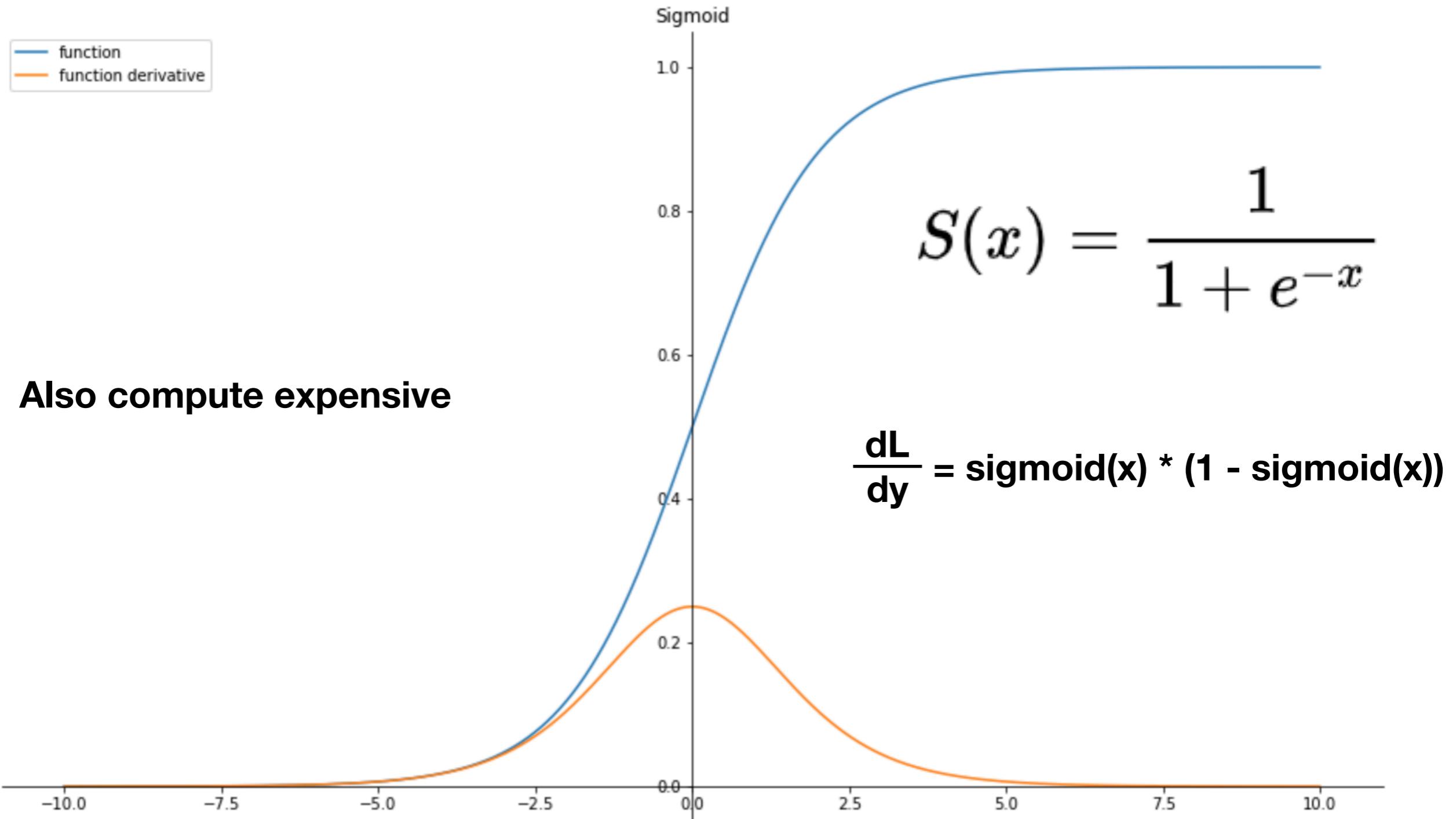
Sigmoid



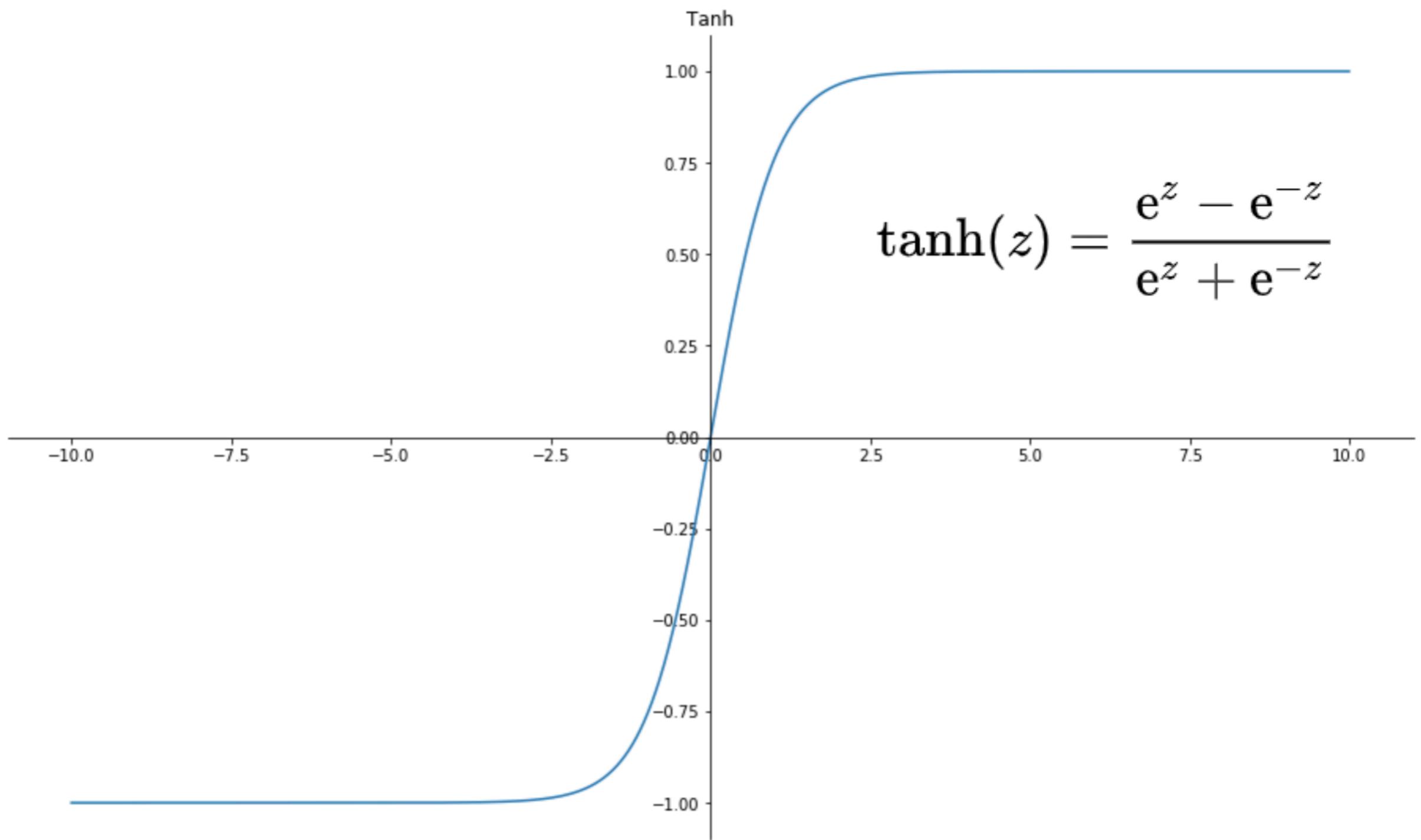
Sigmoid



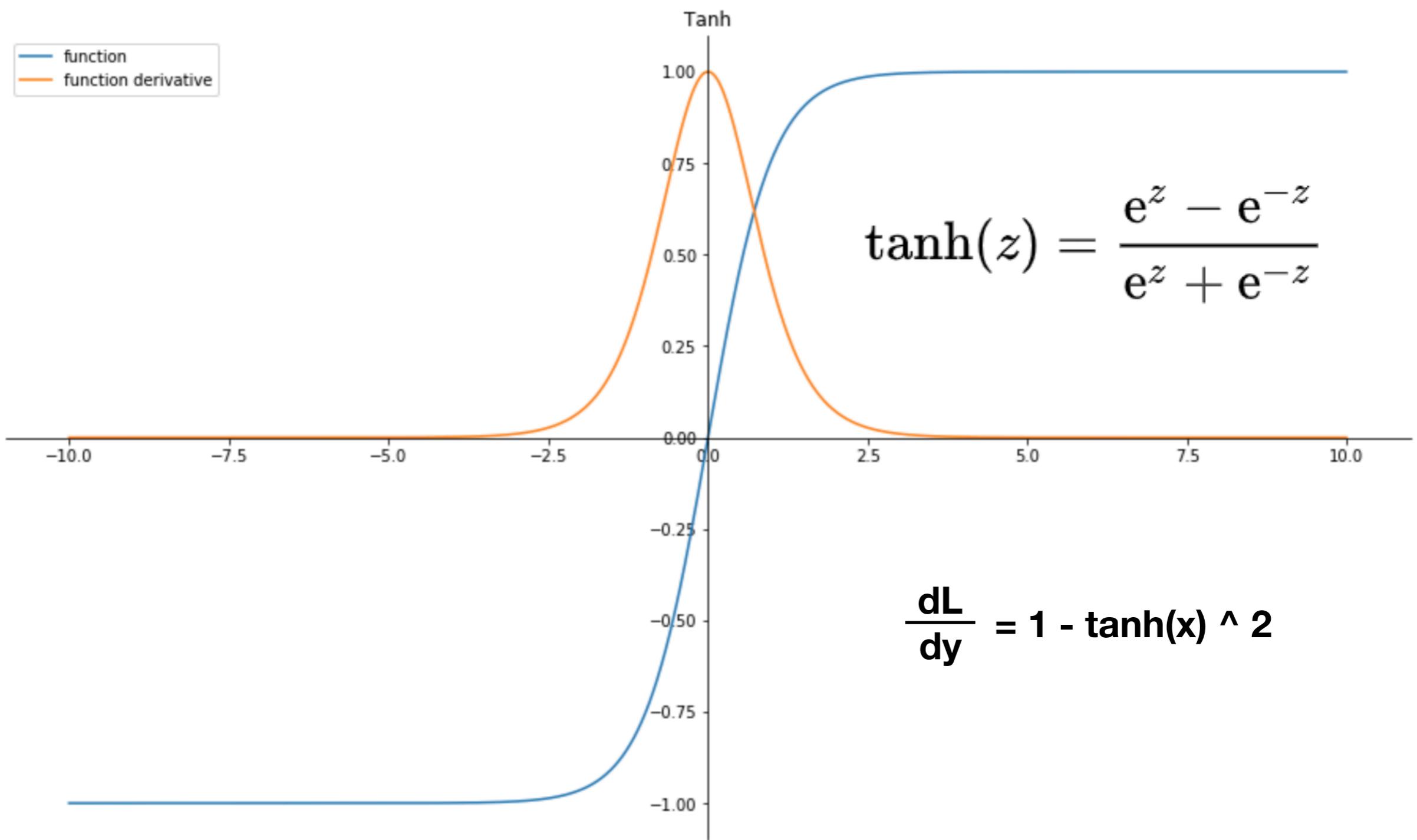
Sigmoid



Tanh

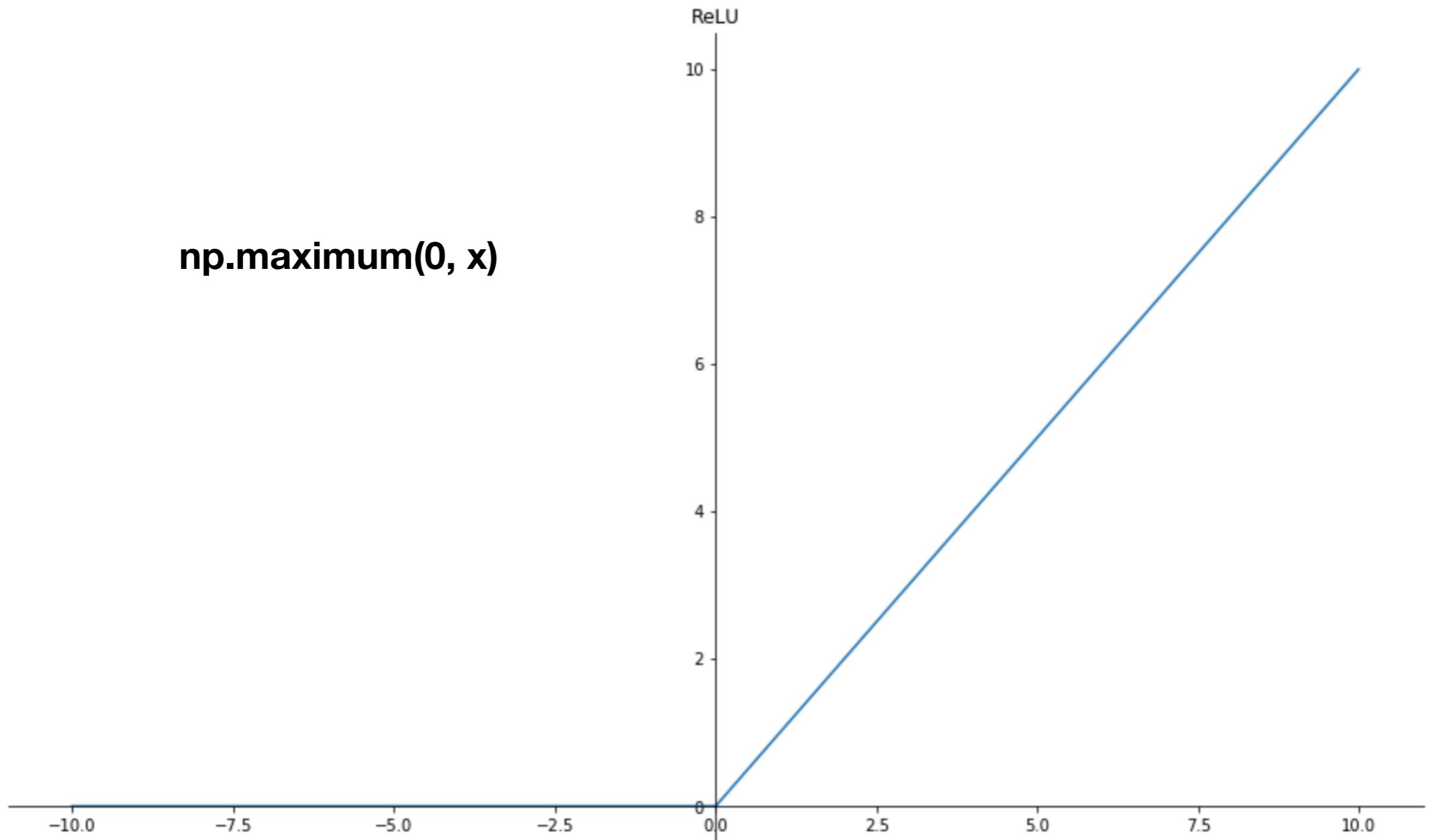


Tanh



ReLU

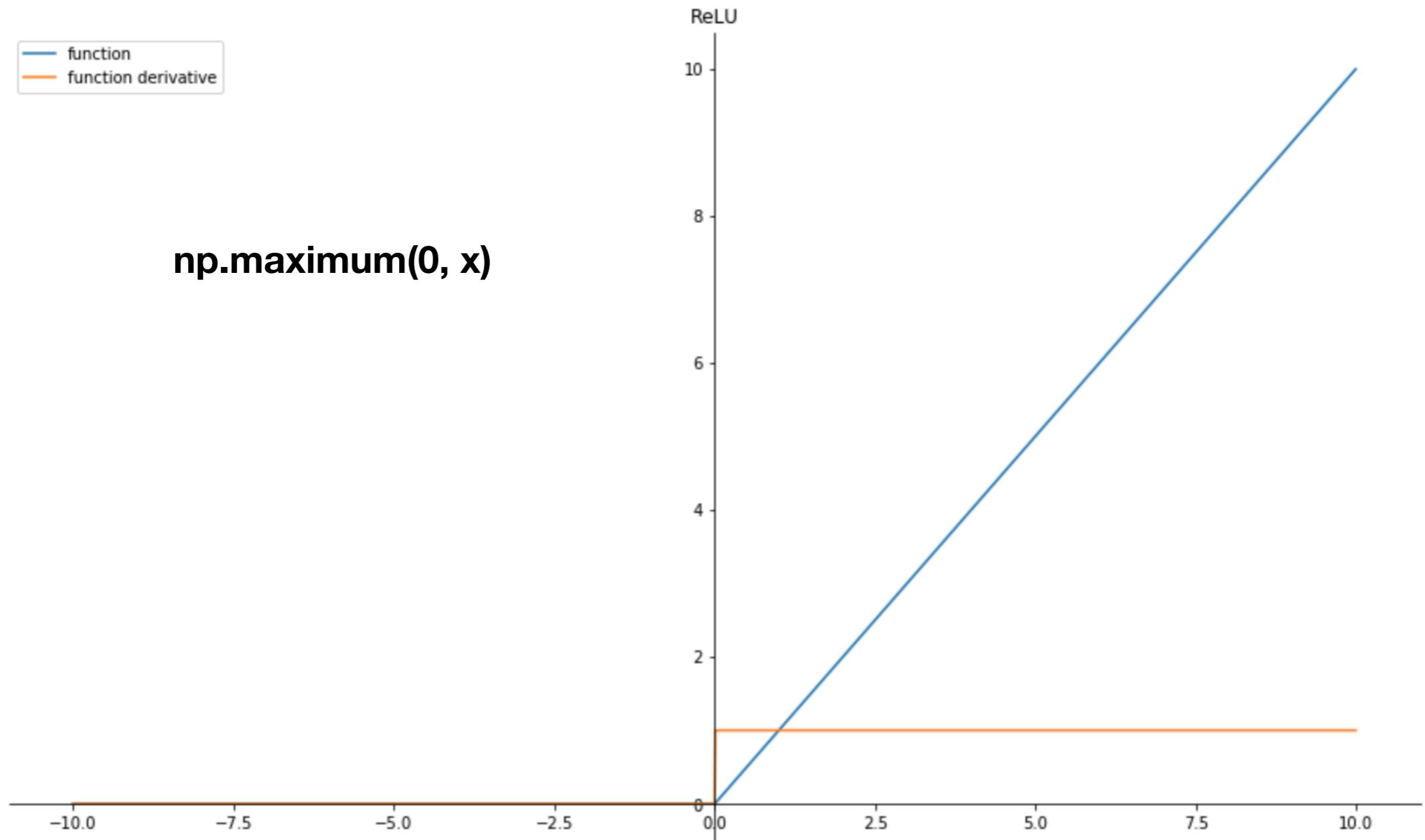
np.maximum(0, x)



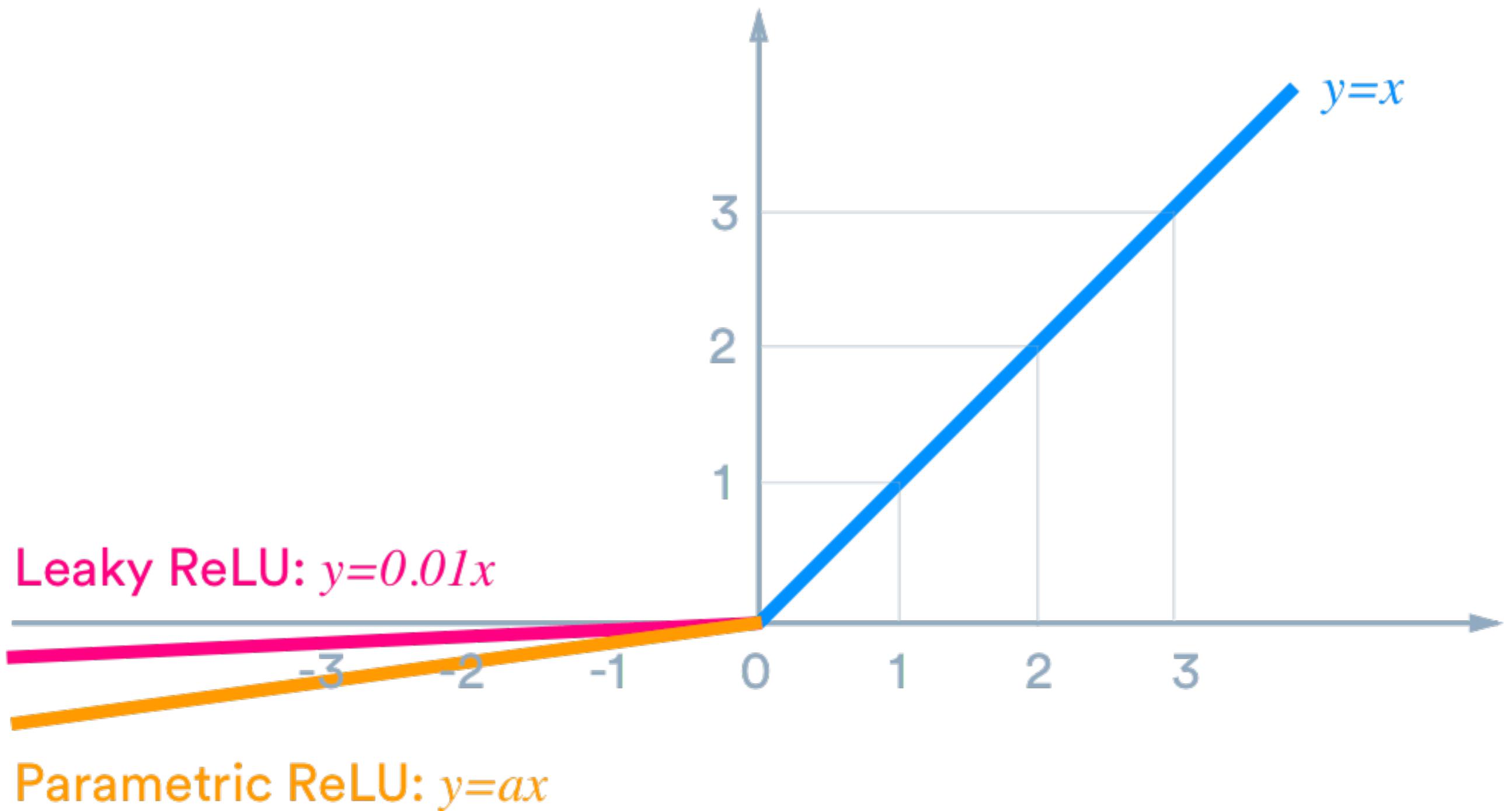
ReLU

function
function derivative

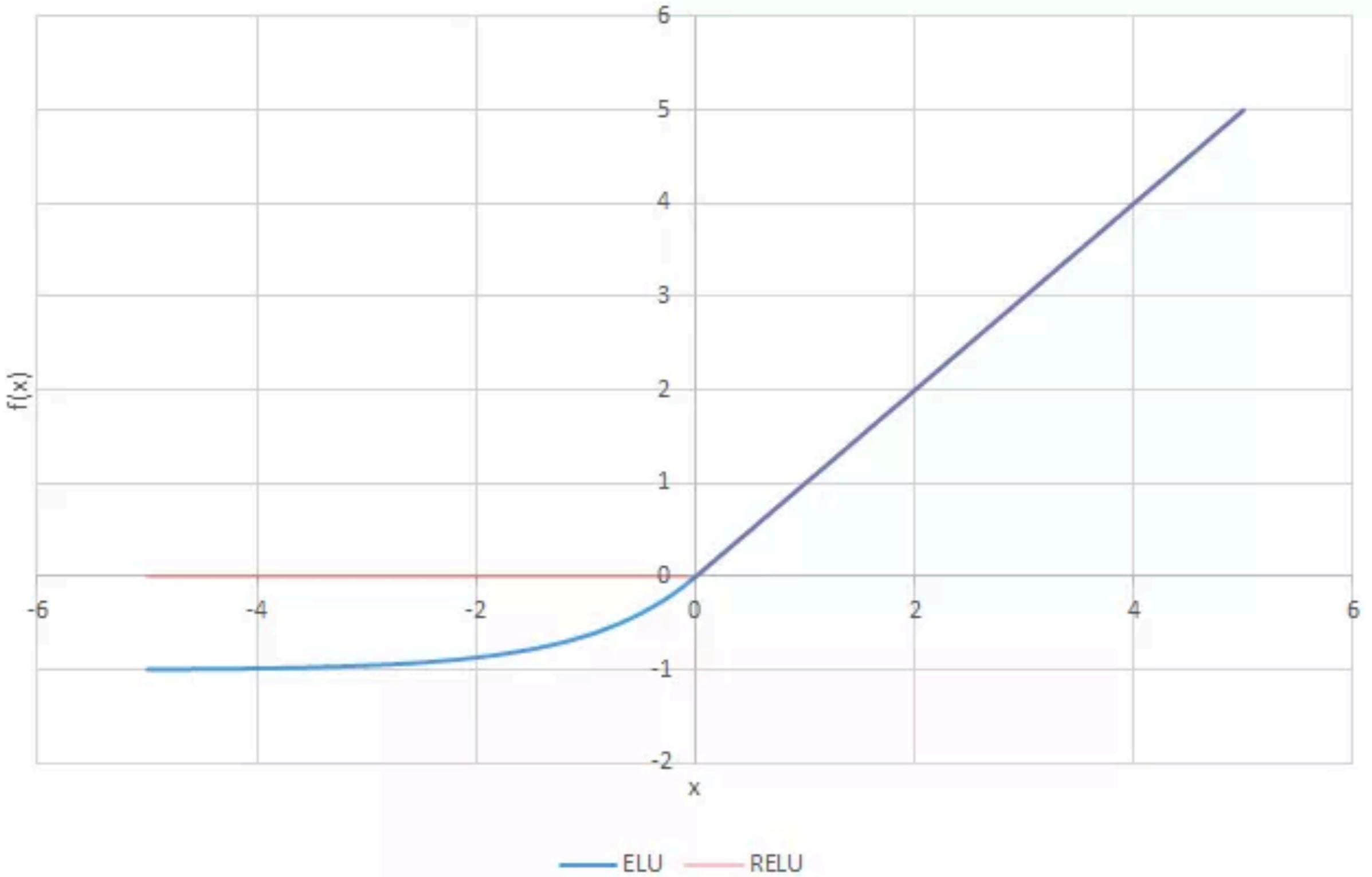
np.maximum(0, x)



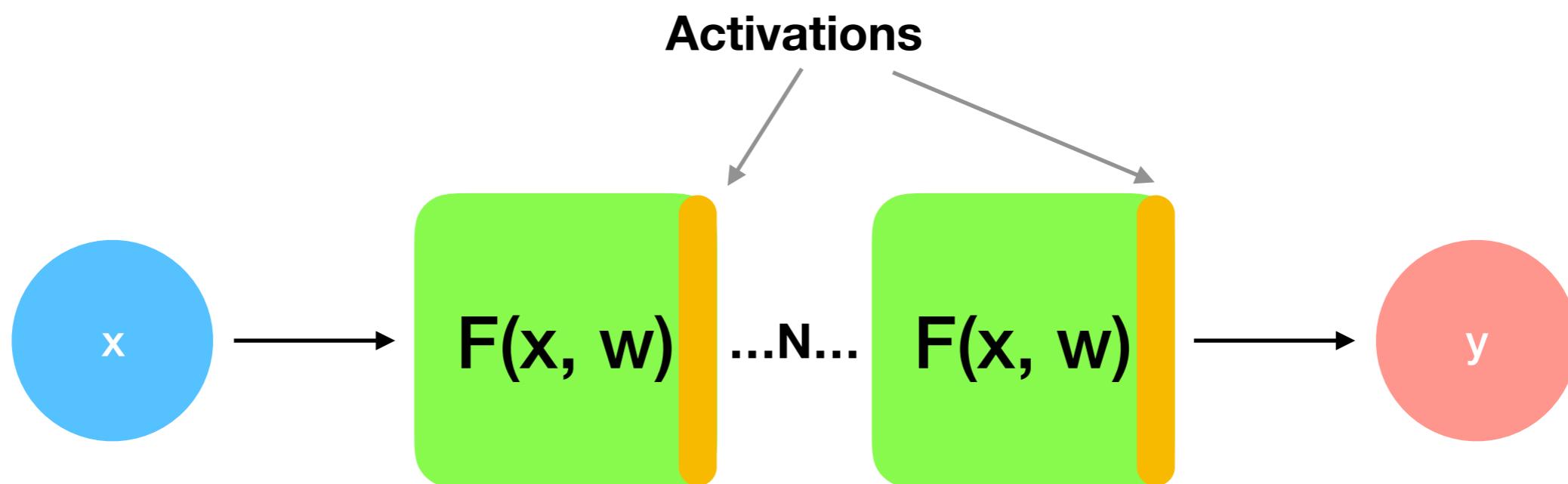
Leaky ReLU



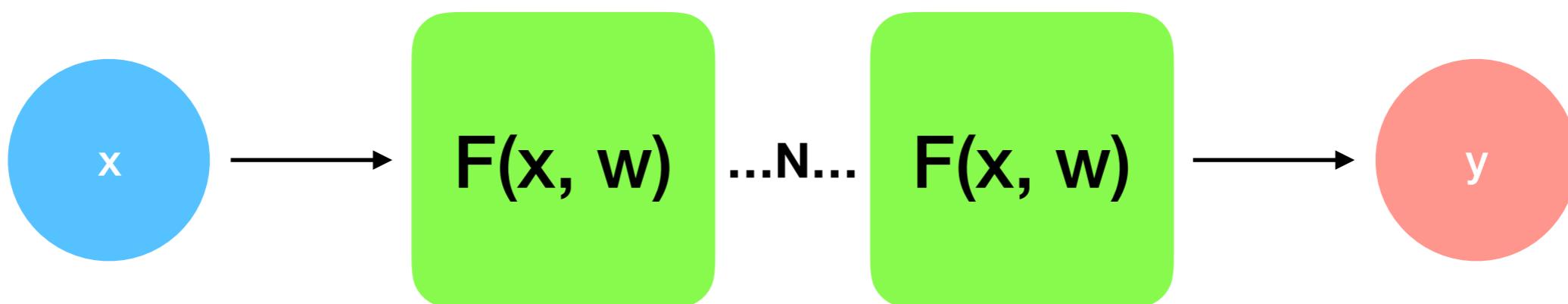
ELU



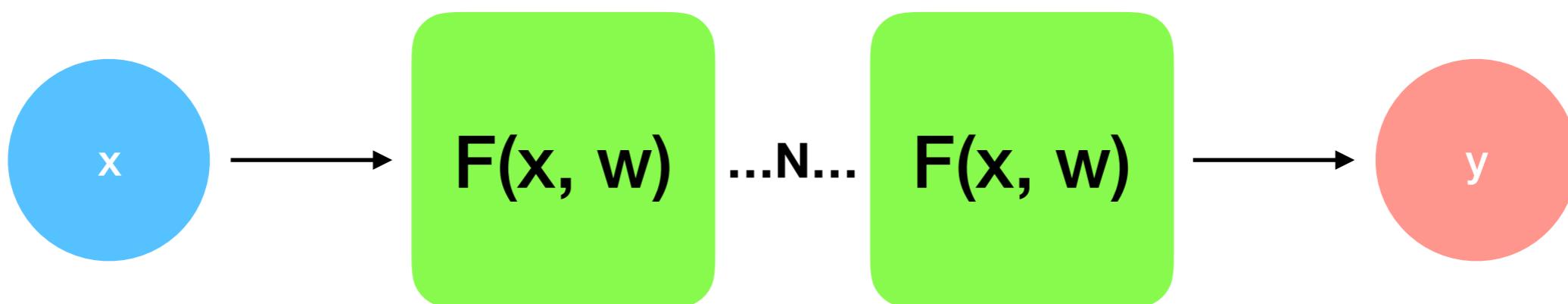
Neural Networks



Neural Networks



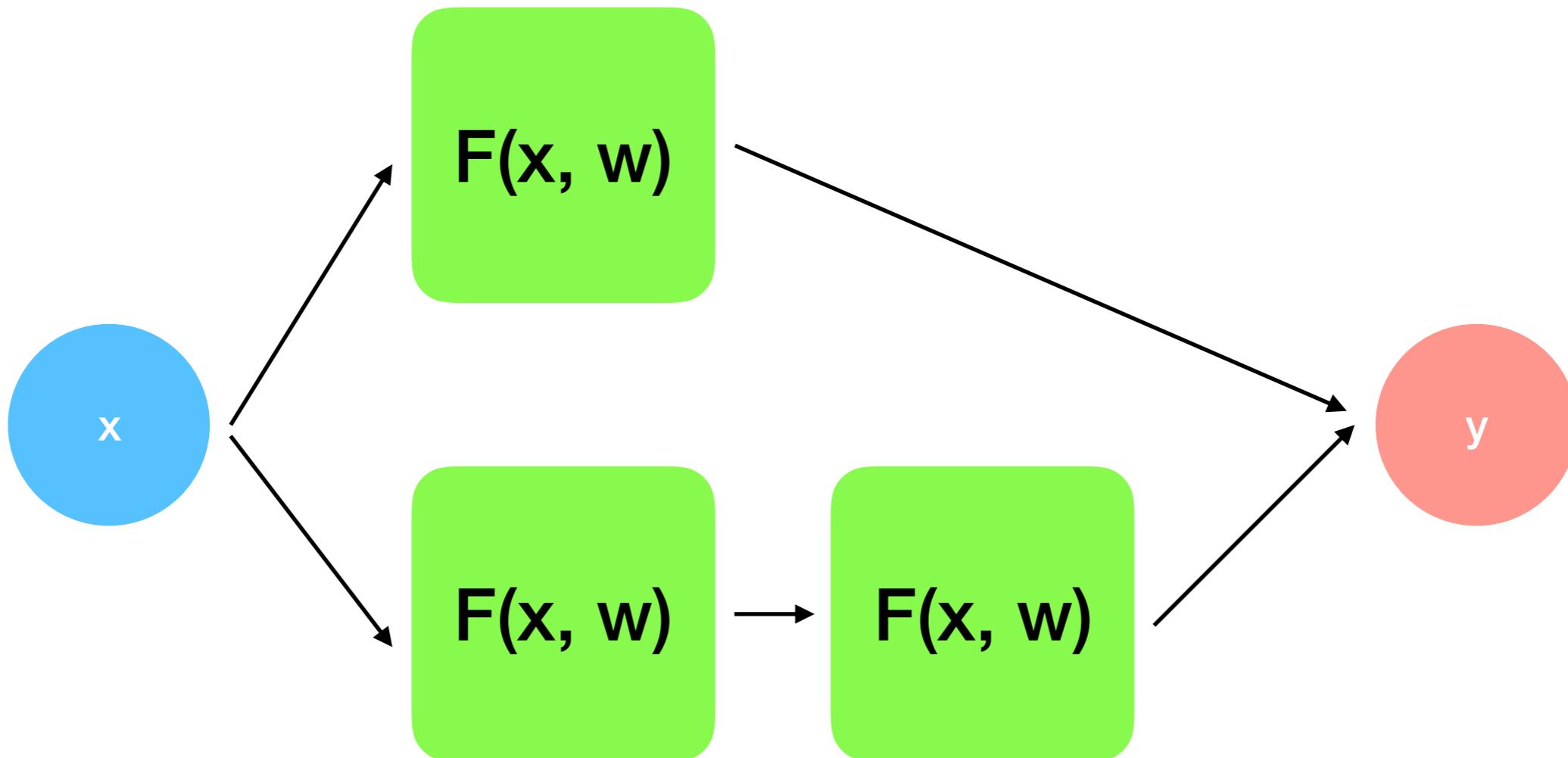
Neural Networks



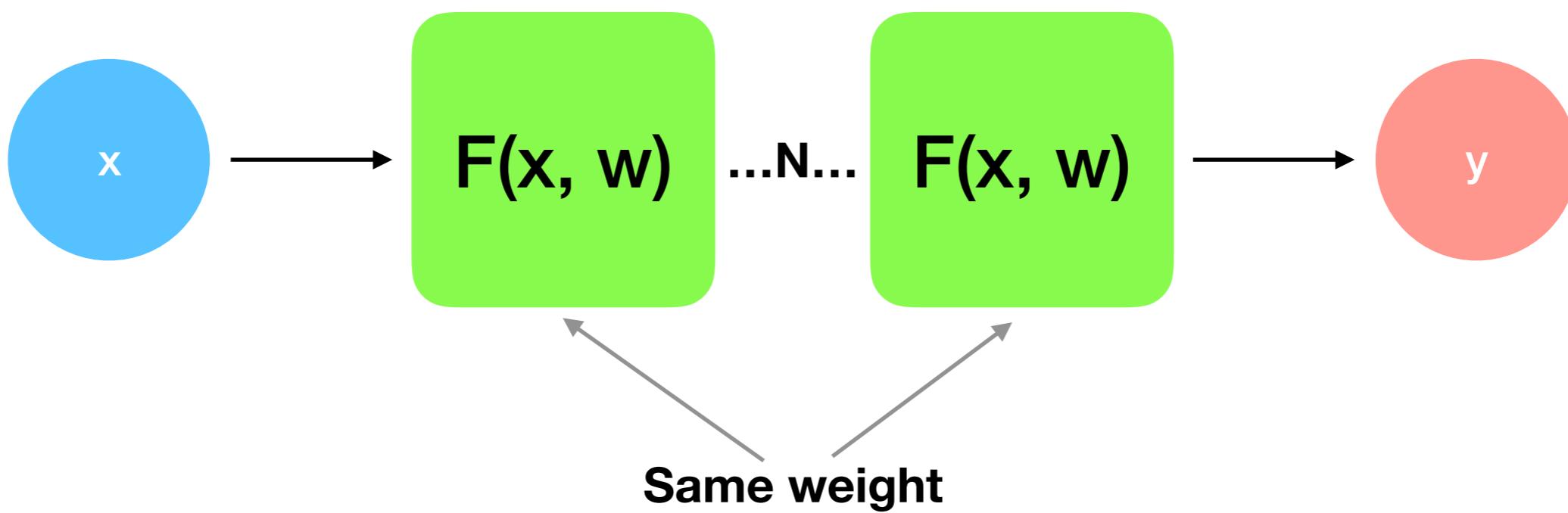
$$W_1 W_2 X = W_3 X$$

Neural Networks

It's a graph!

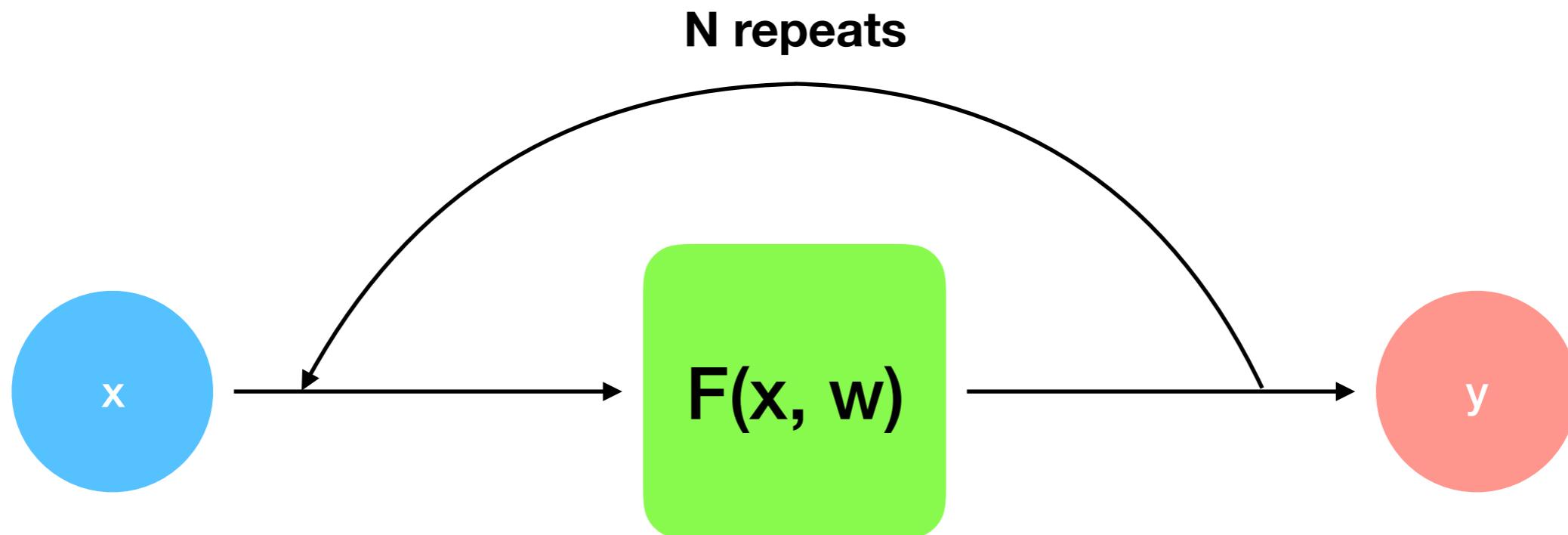


Neural Networks

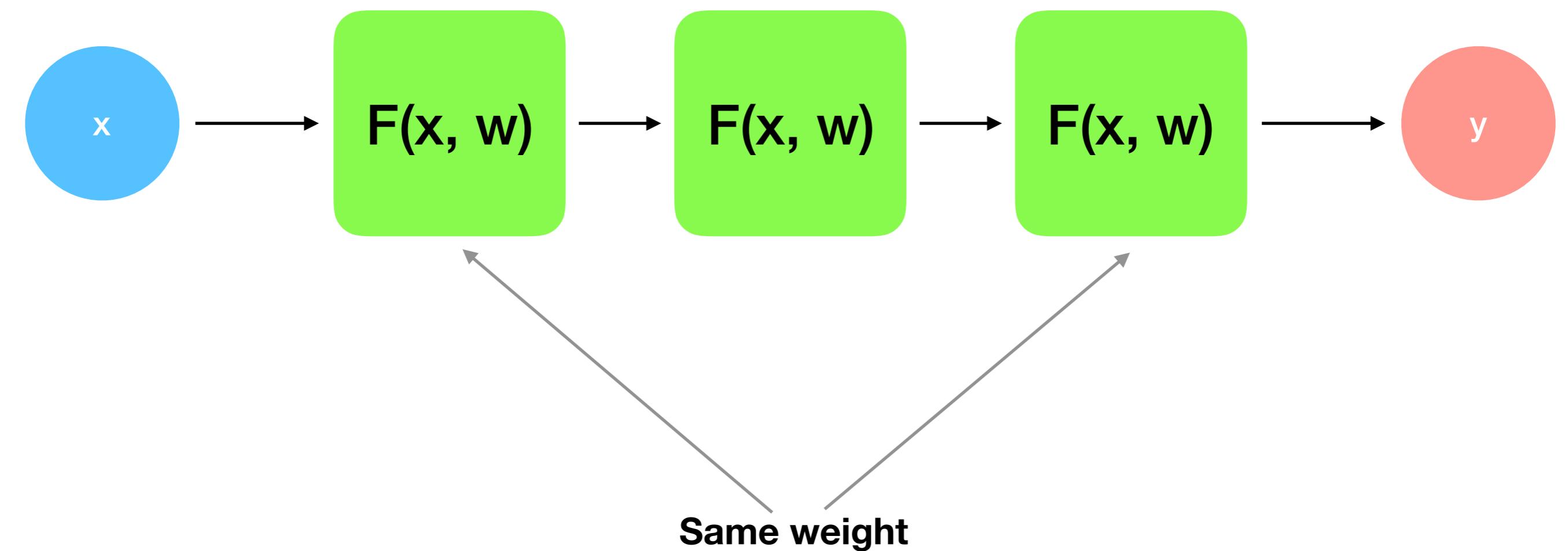


Neural Networks

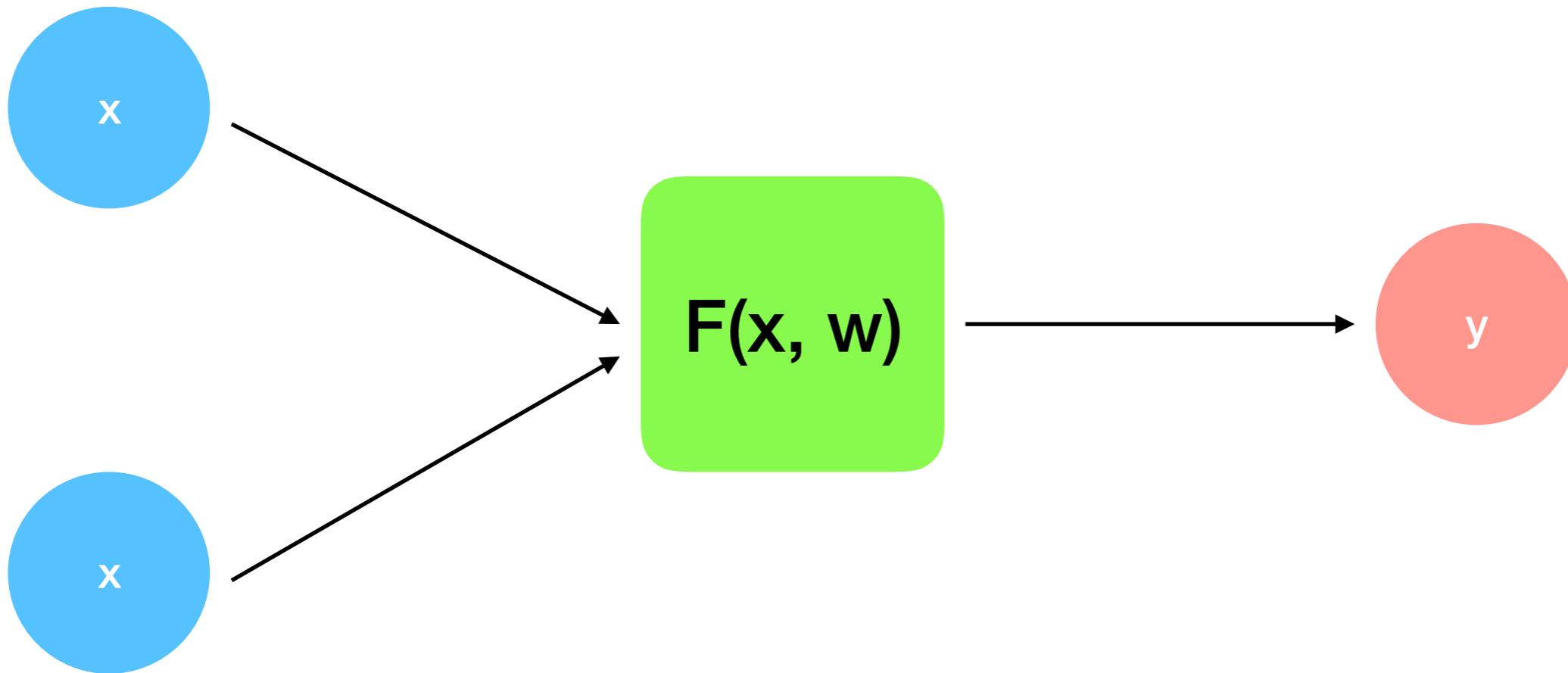
Recurrent



Neural Networks

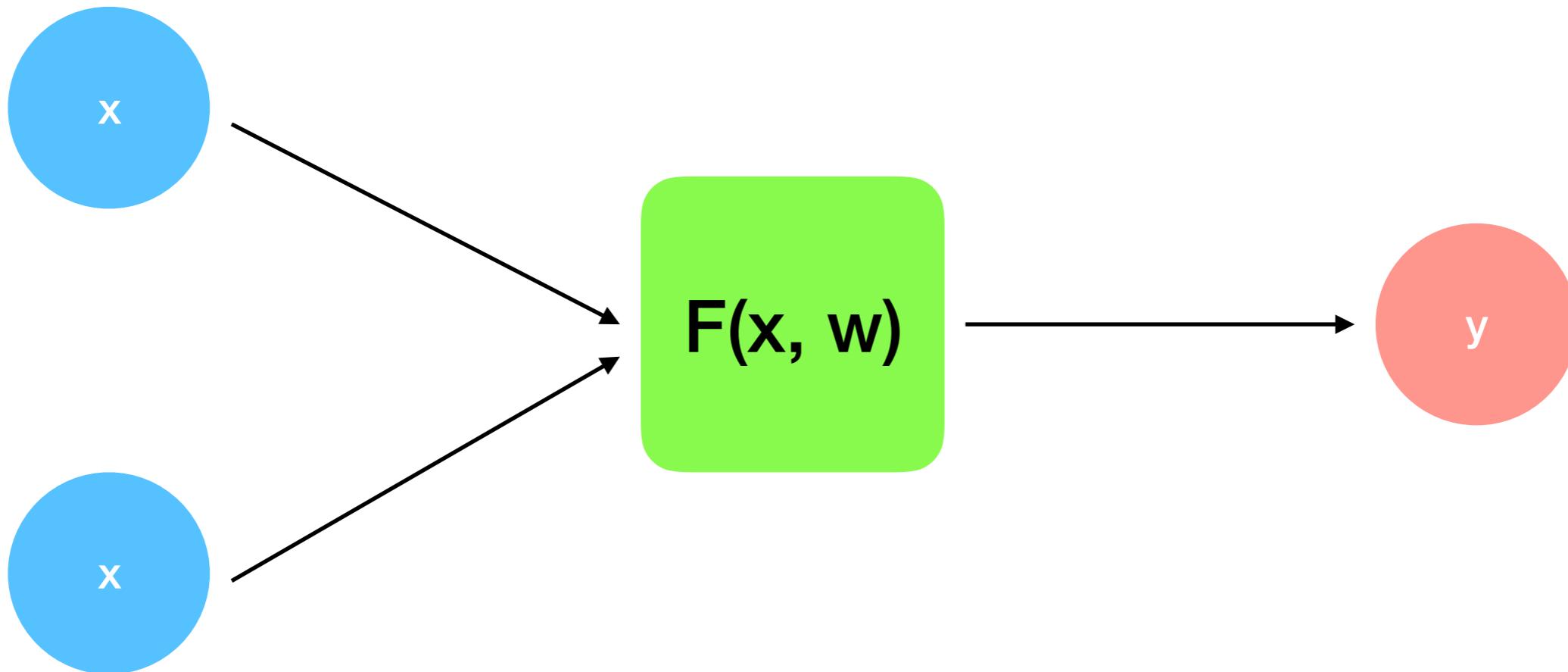


Neural Networks

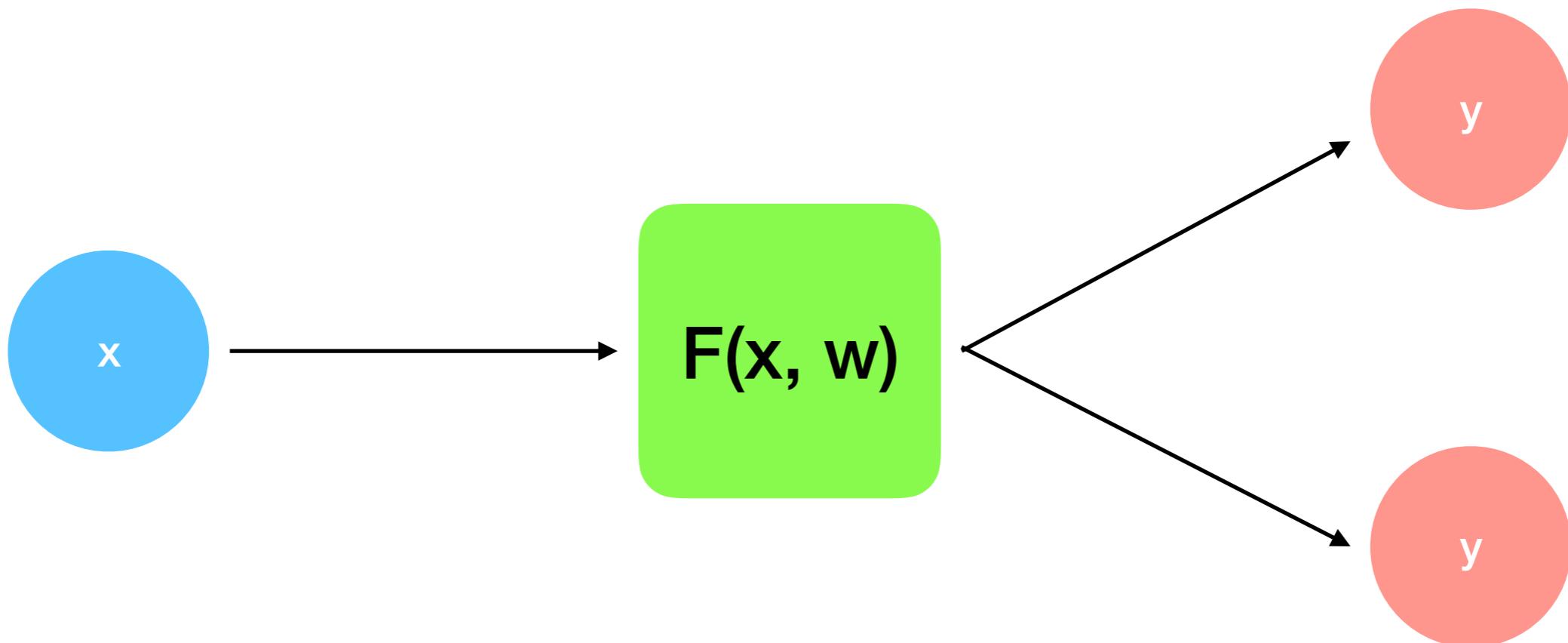


Neural Networks

e.g. summarization

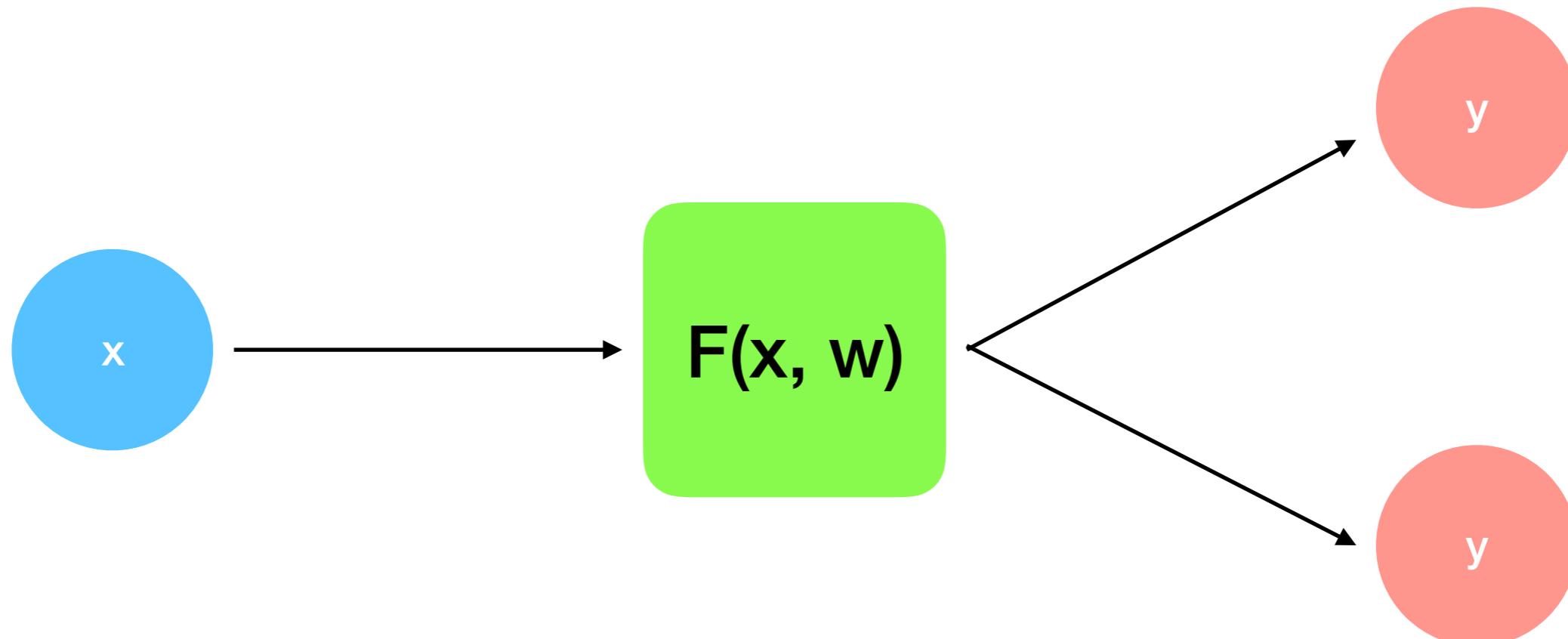


Neural Networks



Neural Networks

e.g. multi-task

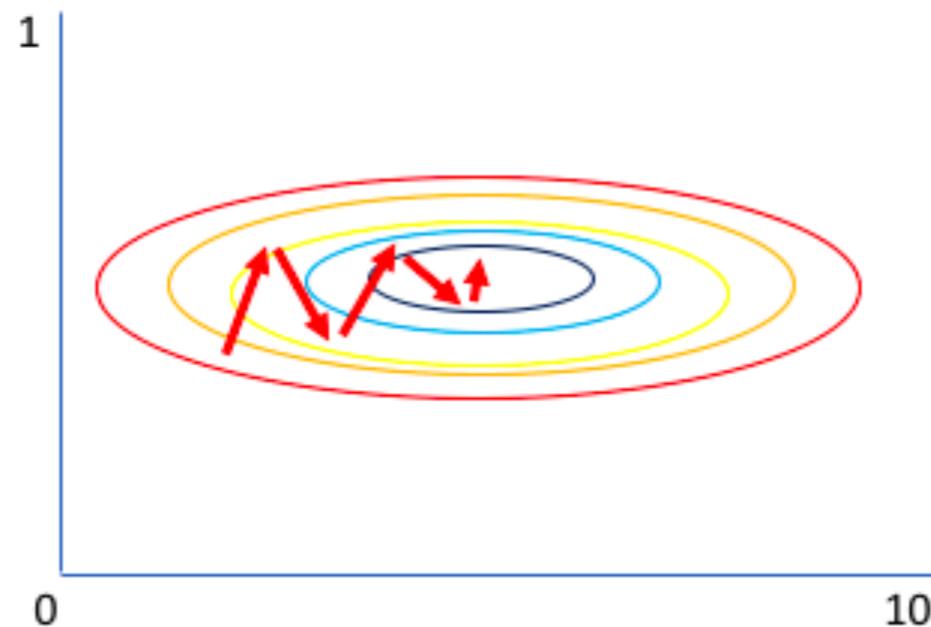


Important things

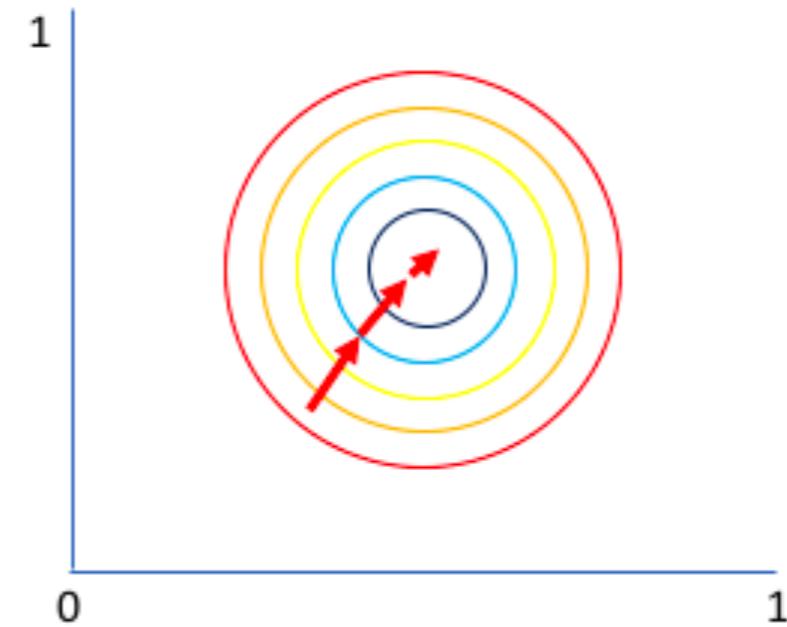
- Don't use sigmoid as inner activation in MLP
- But then we talk about how you can use sigmoid as inner activation
- Normalize your data
- Early stopping

Important things

Why normalize?

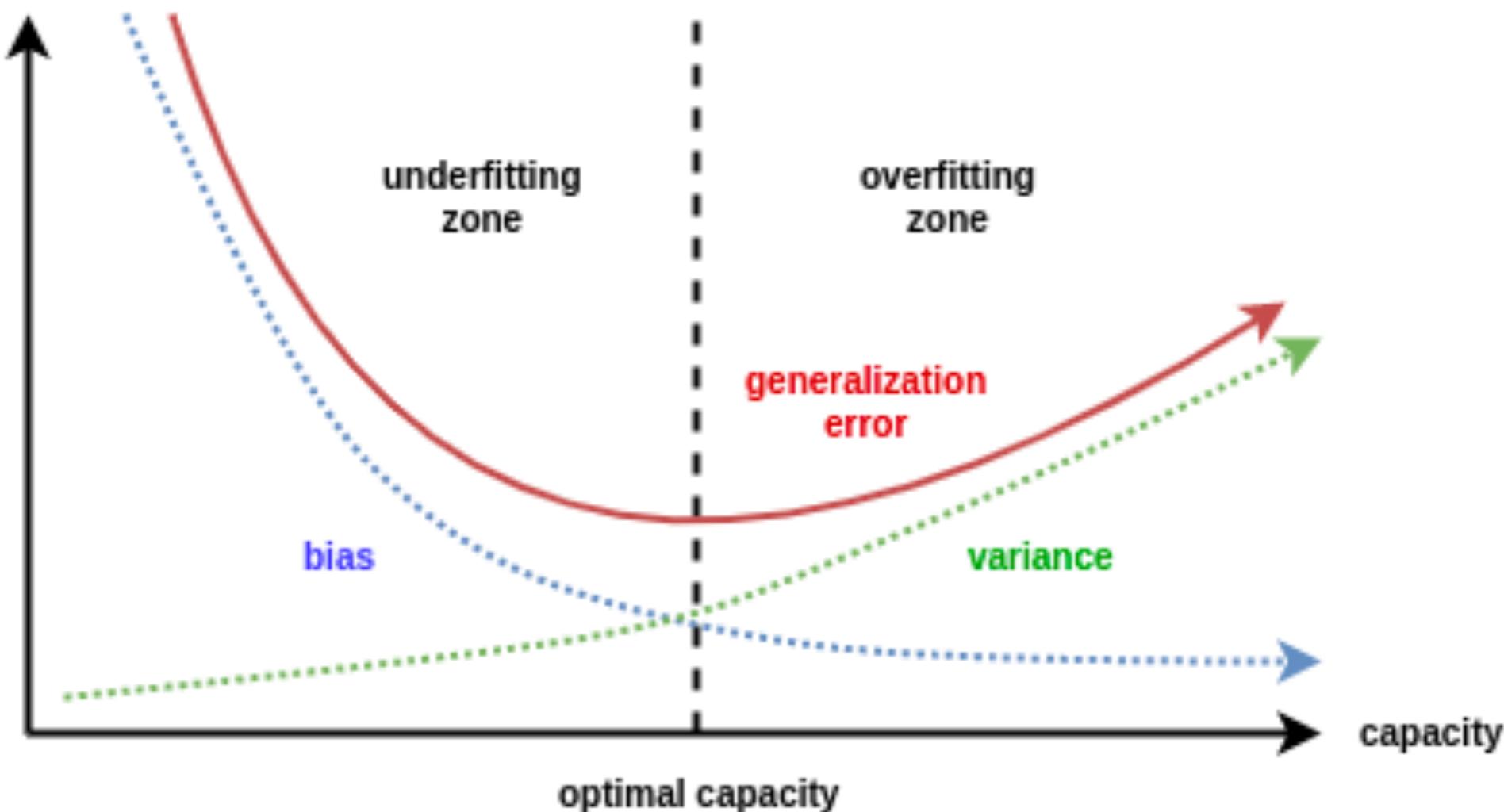


Gradient of larger parameter
dominates the update



Both parameters can be
updated in equal proportions

Early Stopping



Homework

- N-layer neural network

Projects

- Chit-chat chatbot
 - Ranking
 - Generative
- Neural machine translation
- NER or another sequence tagging
- Image captioning
- Kaggle competition

Projects

- Transfer learning from scratch
- Paper implementation
 - ULMFiT
 - GPT
 - BERT
 - XLNet
 - Unsupervised Data Augmentation

Projects

- Question answering
- Low-resource Deep Learning
 - Few-shot learning
- Language modeling
- Transformer
 - Extensions
- Sentiment

Projects

- Long text classification
- Dialogue engines
- Summarization
- Transfer learning from metric learning (sentence embeddings)
- Slot filling
- Your Project

<https://paperswithcode.com/area/natural-language-processing>

Thanks for your Attention!

Boris Zubarev



@bobazooba