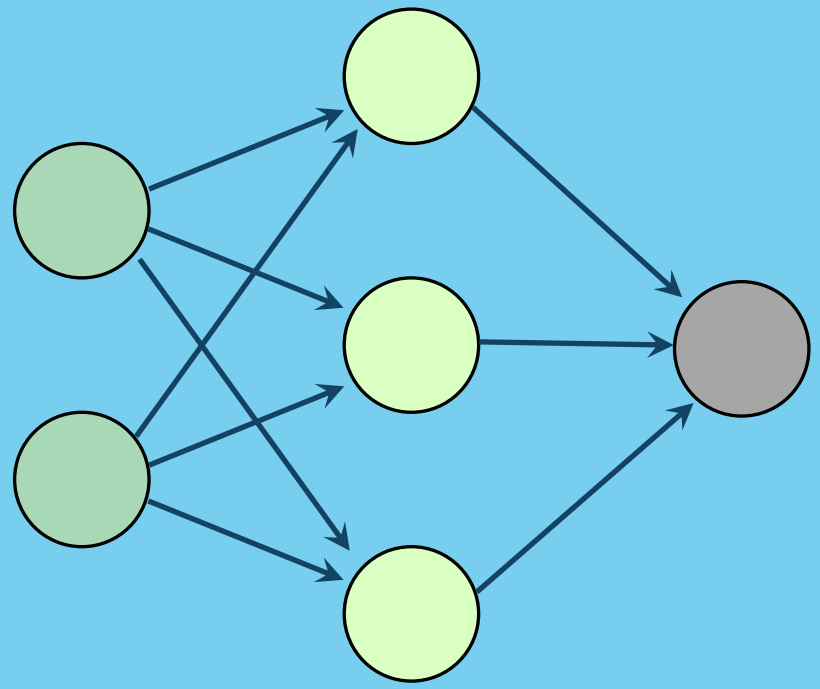


CS5228 LECTURE 7: CLASSIFICATION III: LOGISTIC REGRESSION, DEEP LEARNING

Bryan Hooi

School of Computing

National University of Singapore

Slide Credit: Wang Wei, Ng See Kiong, Wynne Hsu, Coursera deeplearning.ai

ANNOUNCEMENTS

- Assignment 2 has been released (due 2 Apr)

Week	Date	Topics	Tutorials	Important Dates
1	13 Jan	Introduction		
2	20 Jan	No class (public holiday)		
3	27 Jan	Clustering I	Tutorial 1	
4	3 Feb	Clustering II		Release A1 + project
5	10 Feb	Association Rules	Tutorial 2	
6	17 Feb	Regression & Classification I		
Recess		No class		
7	3 Mar	Regression & Classification II	Tutorial 3	A1 due (Sunday 11.59pm), release A2
8	10 Mar	Regression & Classification III		
9	17 Mar	Recommender Systems	Tutorial 4	
10	24 Mar	Graph Mining		
11	31 Mar	Data Stream Mining	Tutorial 5	A2 due (Sunday 11.59pm)
12	7 Apr	No class (public holiday)		
13	14 Apr	Review & Outlook		Project due (Sunday 11.59pm)

ASSIGNMENT 2: GENERAL REMARK

```
# We need to set the seed as the sampling is random, and we want to ensure consistent results
np.random.seed(1)

my_random_forest = MyRandomForestRegressor(n_estimators=100, max_features=1.0).fit(X_toy, y_toy)

print(my_random_forest.predict(np.array([[73, 180], [90, 170]])))
```

```
[ 6.995 10.435]
```

The expected output is as follows:

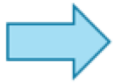
```
[ 6.995 10.435]
```

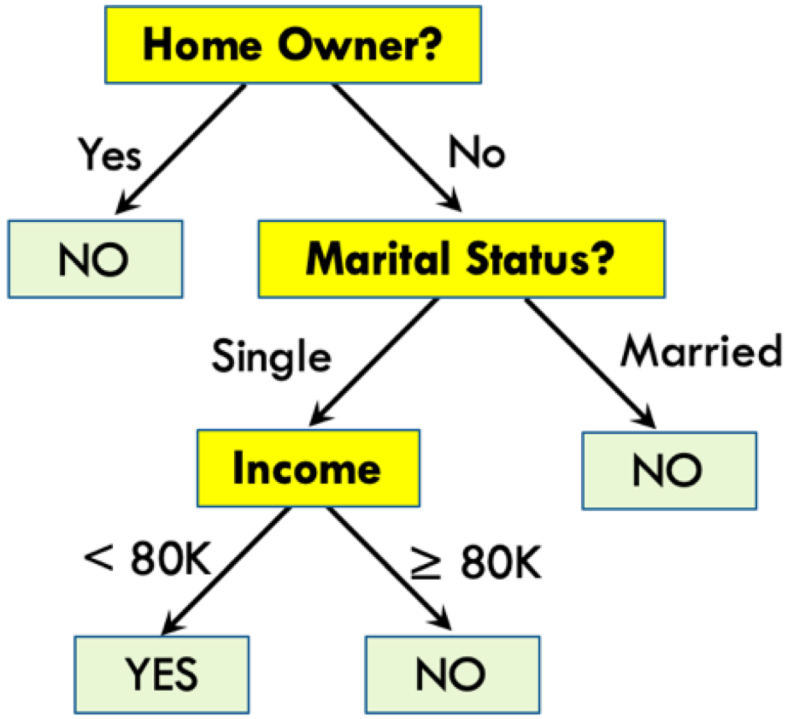
Method involves some randomization → Expected output cannot be guaranteed

- np.random.seed() only ensures the same results in each run
- np.random.seed() does not guarantee that 2 different solutions yield the same result
- Treat the expected output has a guide / ballpark

REVIEW: DECISION TREES

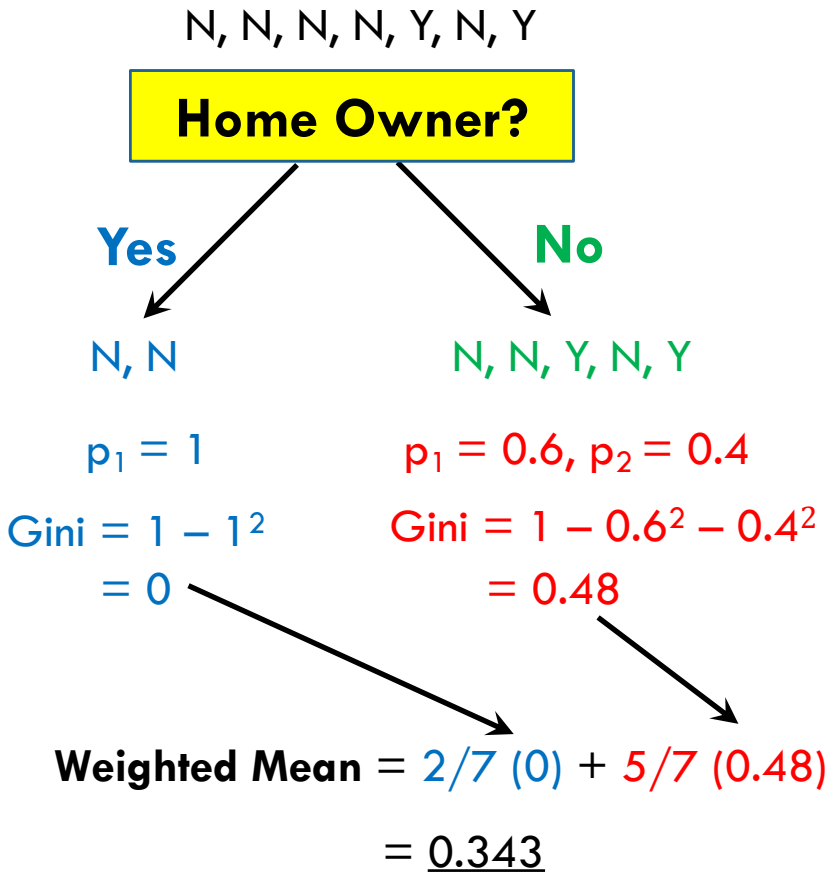
	categorical	categorical	continuous	response
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	170K	No
4	Yes	Married	120K	No
5	No	Single	75K	Yes
6	No	Married	160K	No
7	No	Single	50K	Yes





REVIEW: GINI INDEX FOR EVALUATING SPLITS

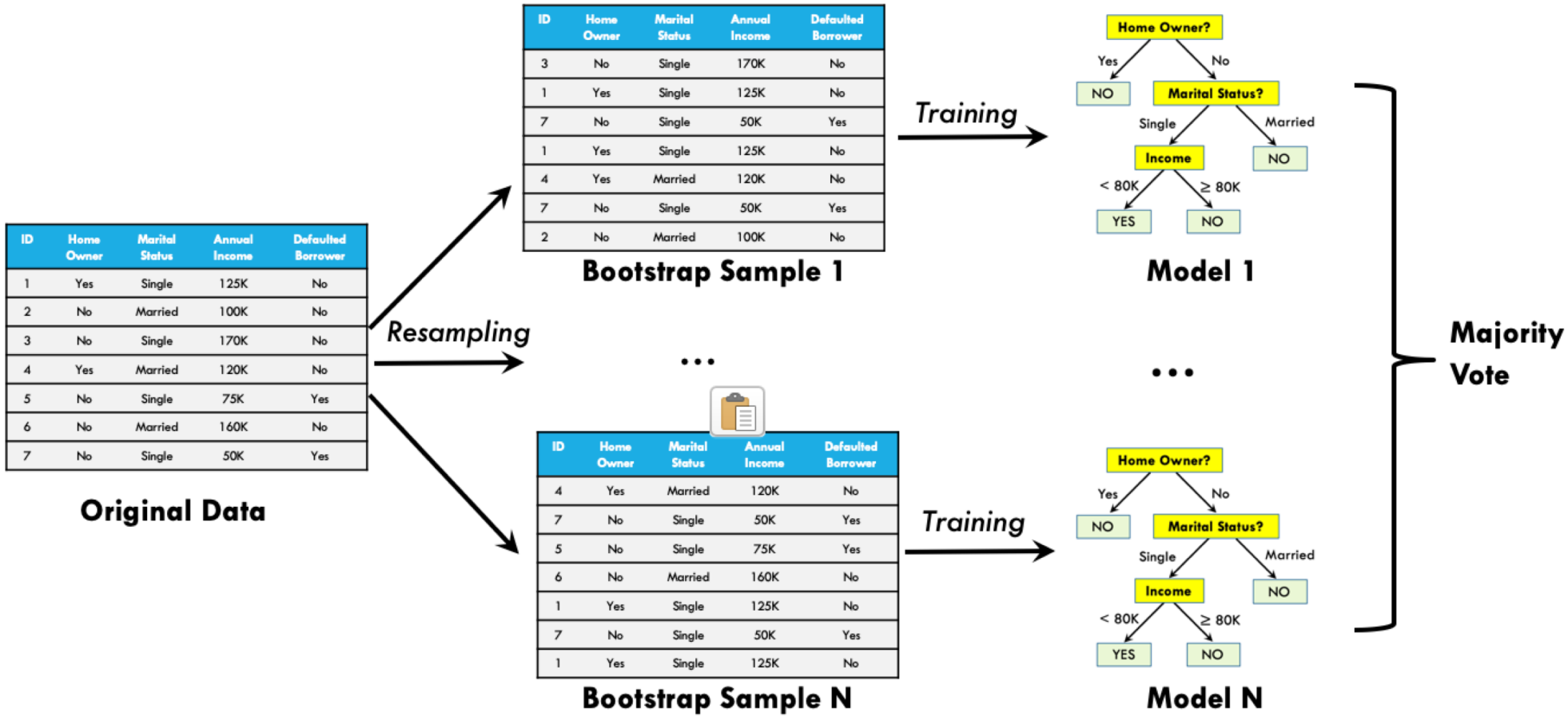
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	170K	No
4	Yes	Married	120K	No
5	No	Single	75K	Yes
6	No	Married	160K	No
7	No	Single	50K	Yes



$$\text{Gini Index} = 1 - \sum_{i=1}^{J} {p_i}^2$$

(Lower Gini = Purer distribution = Better split)

REVIEW: BAGGING AND RANDOM FORESTS



REVIEW: BOOSTING AND GRADIENT BOOSTING MACHINES

Model Predictions

Initialization: $\hat{f}(x) = 0$, and $r_i = y_i$ for all i

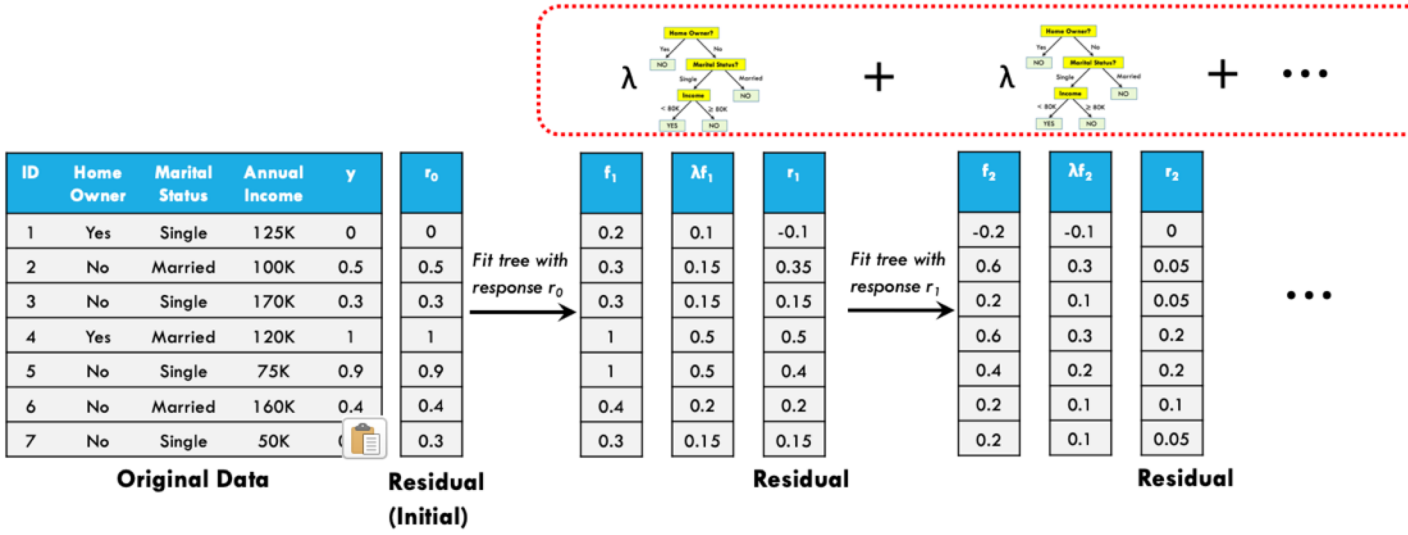
For $b = 1, \ldots, B$:

- **Fit Decision Tree** $\hat{f}^b$ to residuals $r_1, \ldots, r_n$.
- **Update Residual**

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

Output:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$



ID	Home Owner	Marital Status	Annual Income	y	r_0	f_1	λf_1	r_1	f_2	λf_2	r_2
1	Yes	Single	125K	0	0	0.2	0.1	-0.1	-0.2	-0.1	0
2	No	Married	100K	0.5	0.5	0.3	0.15	0.35	0.6	0.3	0.05
3	No	Single	170K	0.3	0.3	0.3	0.15	0.15	0.2	0.1	0.05
4	Yes	Married	120K	1	1	1	0.5	0.5	0.6	0.3	0.2
5	No	Single	75K	0.9	0.9	1	0.5	0.4	0.4	0.2	0.2
6	No	Married	160K	0.4	0.4	0.4	0.2	0.2	0.2	0.1	0.1
7	No	Single	50K	0.9	0.3	0.3	0.15	0.15	0.2	0.1	0.05

CLASSIFICATION OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. **Neural Networks**
   a) Introduction
   b) Logistic Regression
   c) Deep Learning

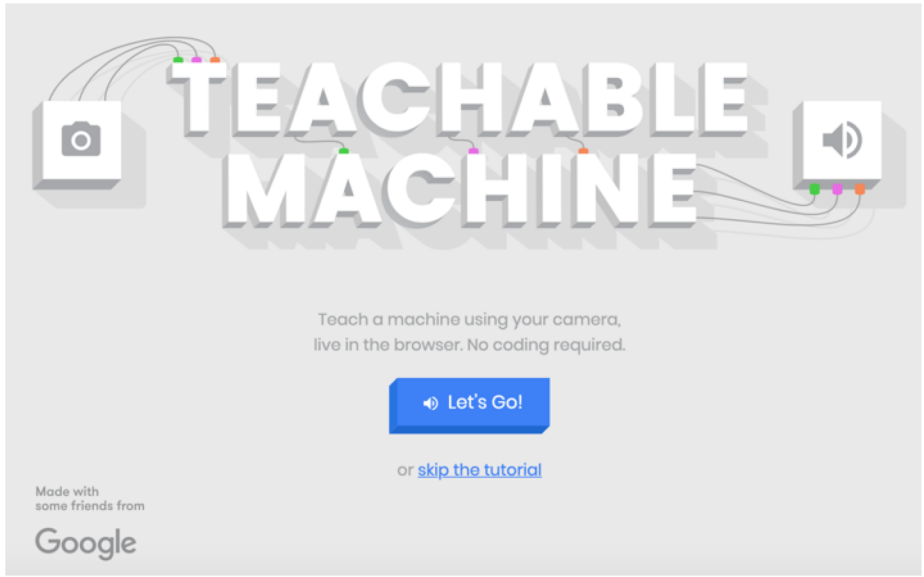
Today's Lecture

CLASSIFICATION OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. **Neural Networks**
   a) Introduction
   b) Logistic Regression
   c) Deep Learning

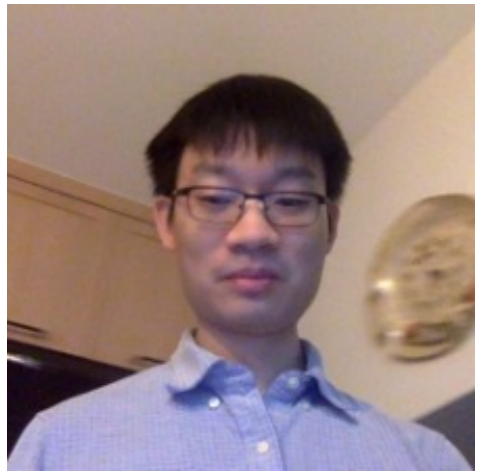
CLASSIFICATION: HUMAN VS DOG

Input Images

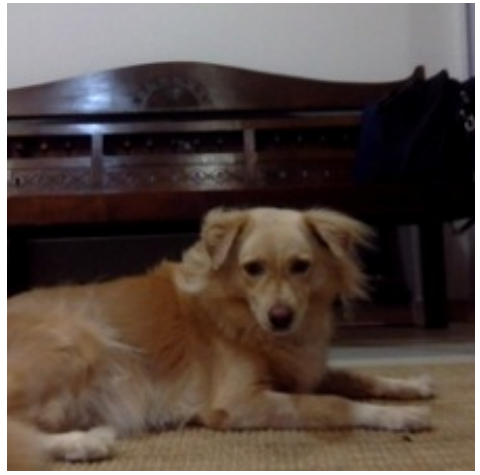


Classification Algorithm

Human



Dog

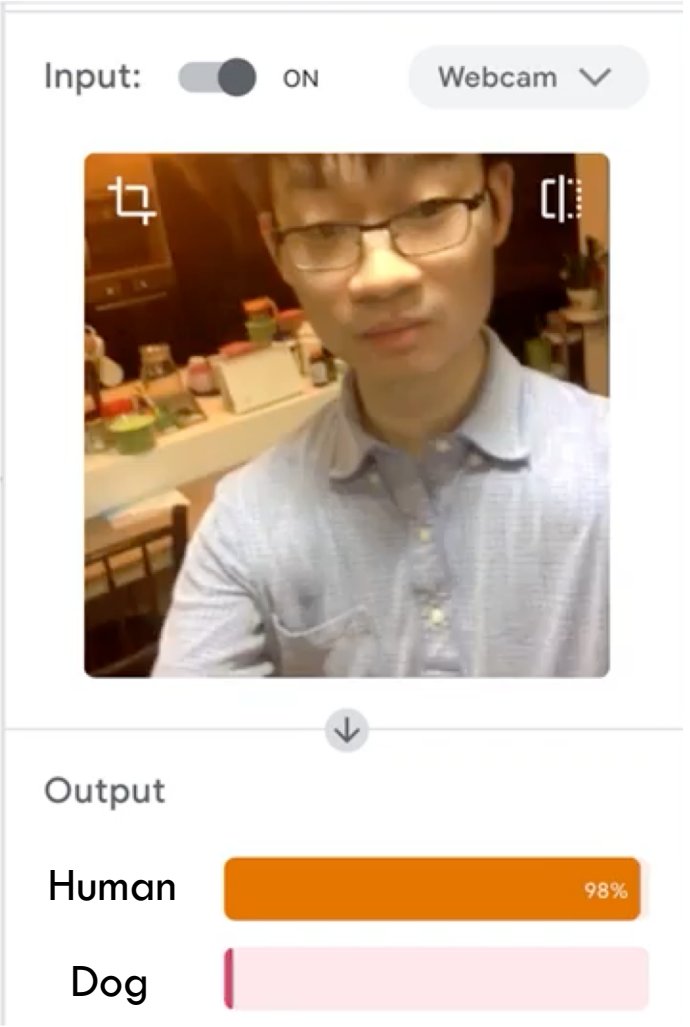


teachablemachine.withgoogle.com

CLASSIFICATION: HUMAN VS DOG

Teachable Machine

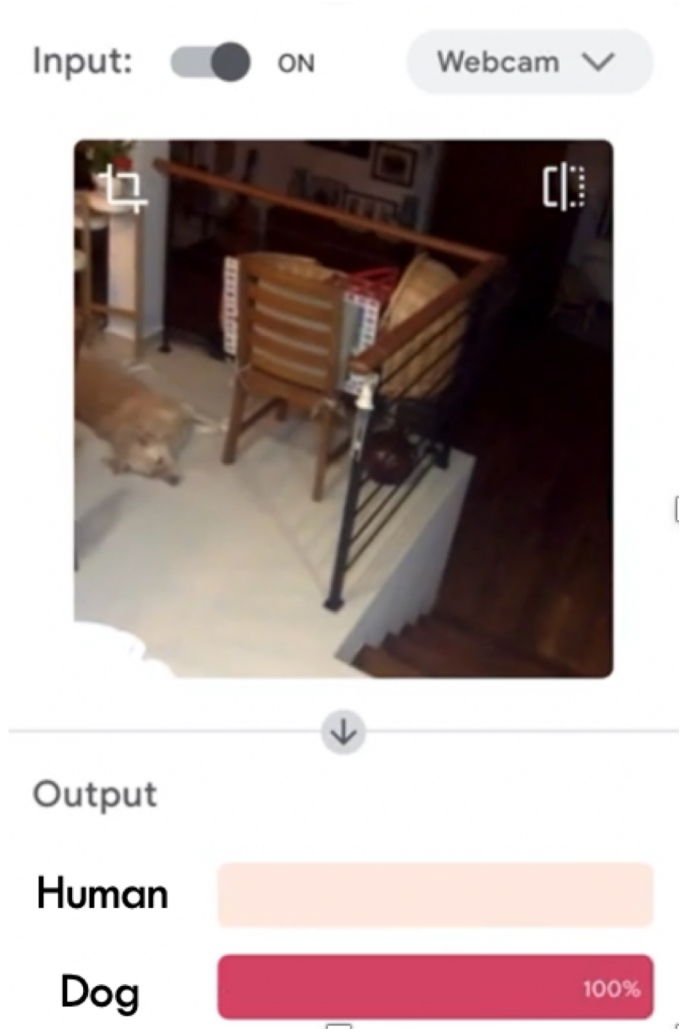
Train a computer to recognize your own images, sounds, & poses.



CLASSIFICATION: HUMAN VS DOG

Teachable Machine

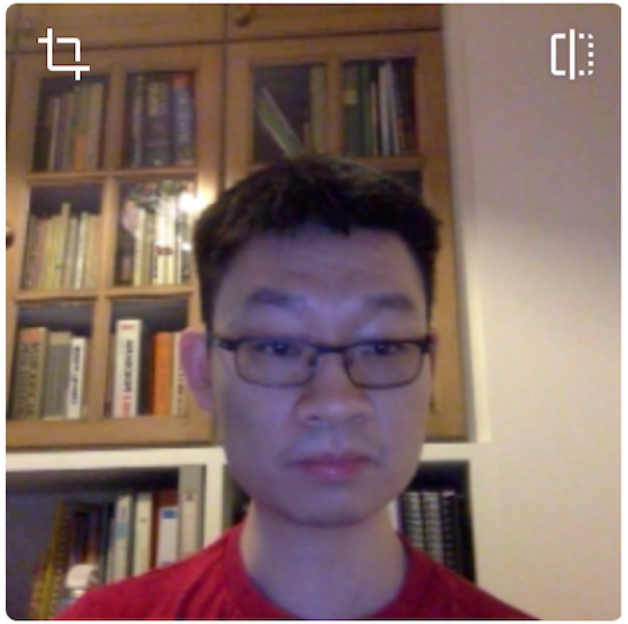
Train a computer to recognize your own images, sounds, & poses.



CLASSIFICATION: HUMAN VS DOG

Teachable Machine

Train a computer to recognize your own images, sounds, & poses.



Output

Human 100%

Dog

teachablemachine.withgoogle.com

CAN DEEP LEARNING HANDLE UNEXPECTED INPUTS?

Teachable Machine

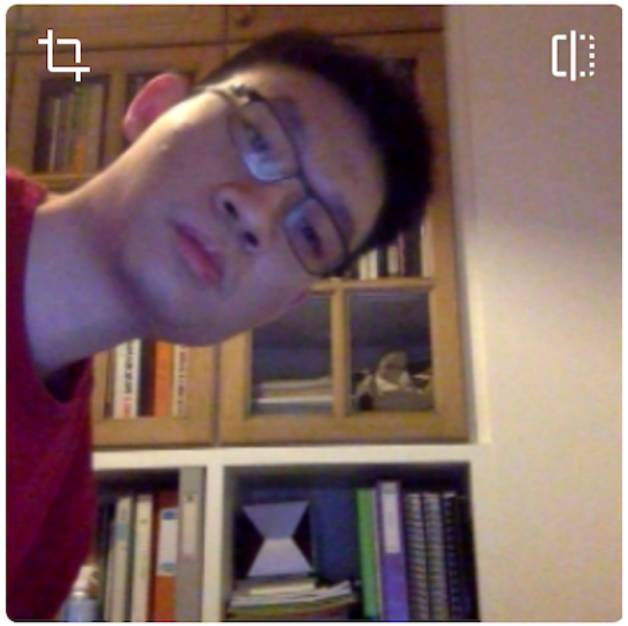
Train a computer to recognize your own images, sounds, & poses.

teachablemachine.withgoogle.com

CAN DEEP LEARNING HANDLE UNEXPECTED INPUTS?

Teachable Machine

Train a computer to recognize your own images, sounds, & poses.



Output

Human

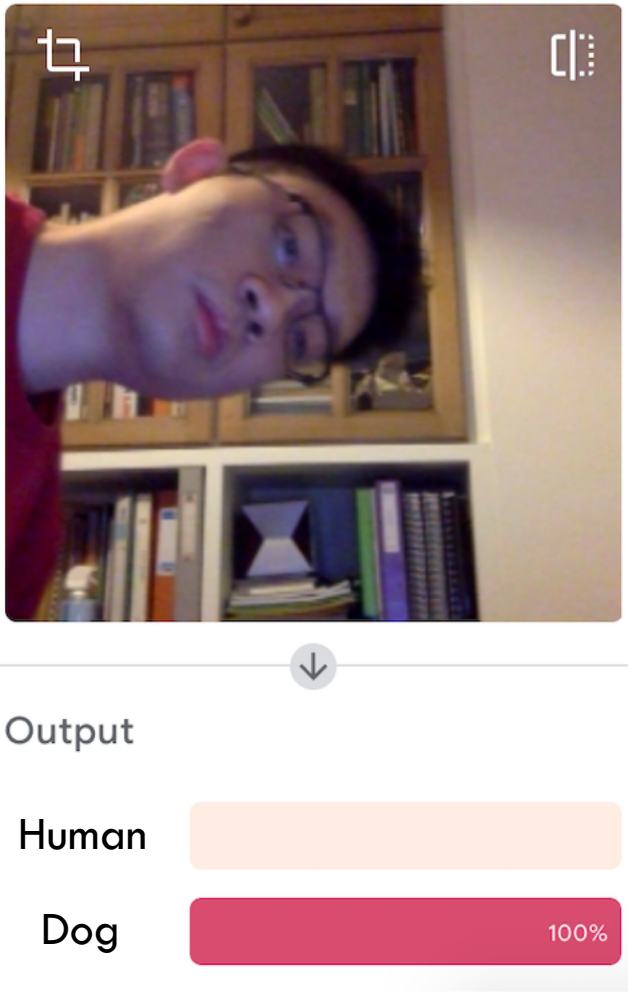
Dog 99%

teachablemachine.withgoogle.com

CAN DEEP LEARNING HANDLE UNEXPECTED INPUTS?

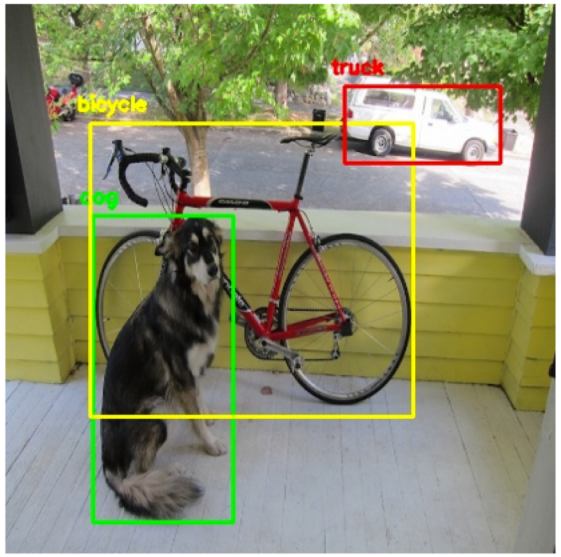
Teachable Machine

Train a computer to recognize your own images, sounds, & poses.



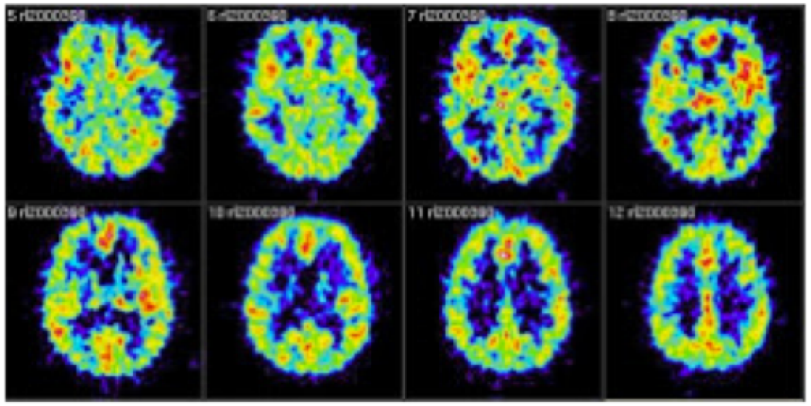
teachablemachine.withgoogle.com

APPLICATIONS OF DEEP LEARNING

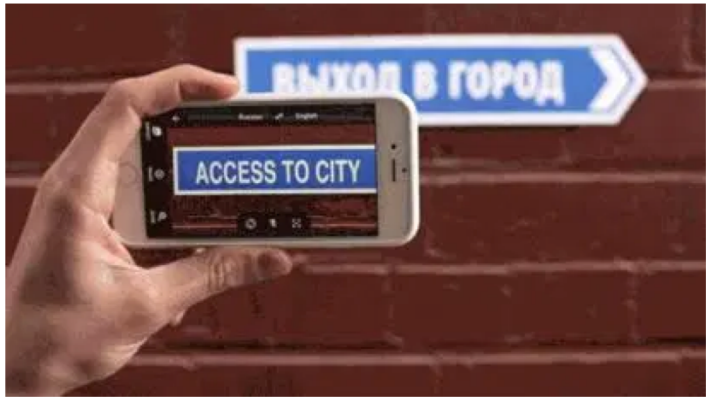


Object Detection

Voice Recognition



Medical Imaging

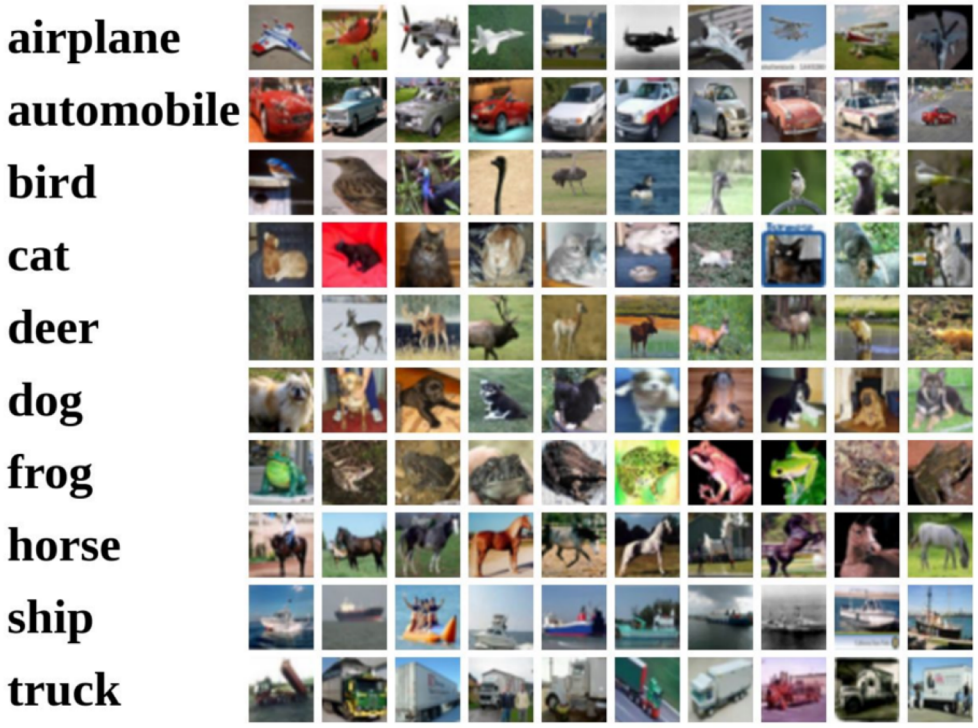


Translation

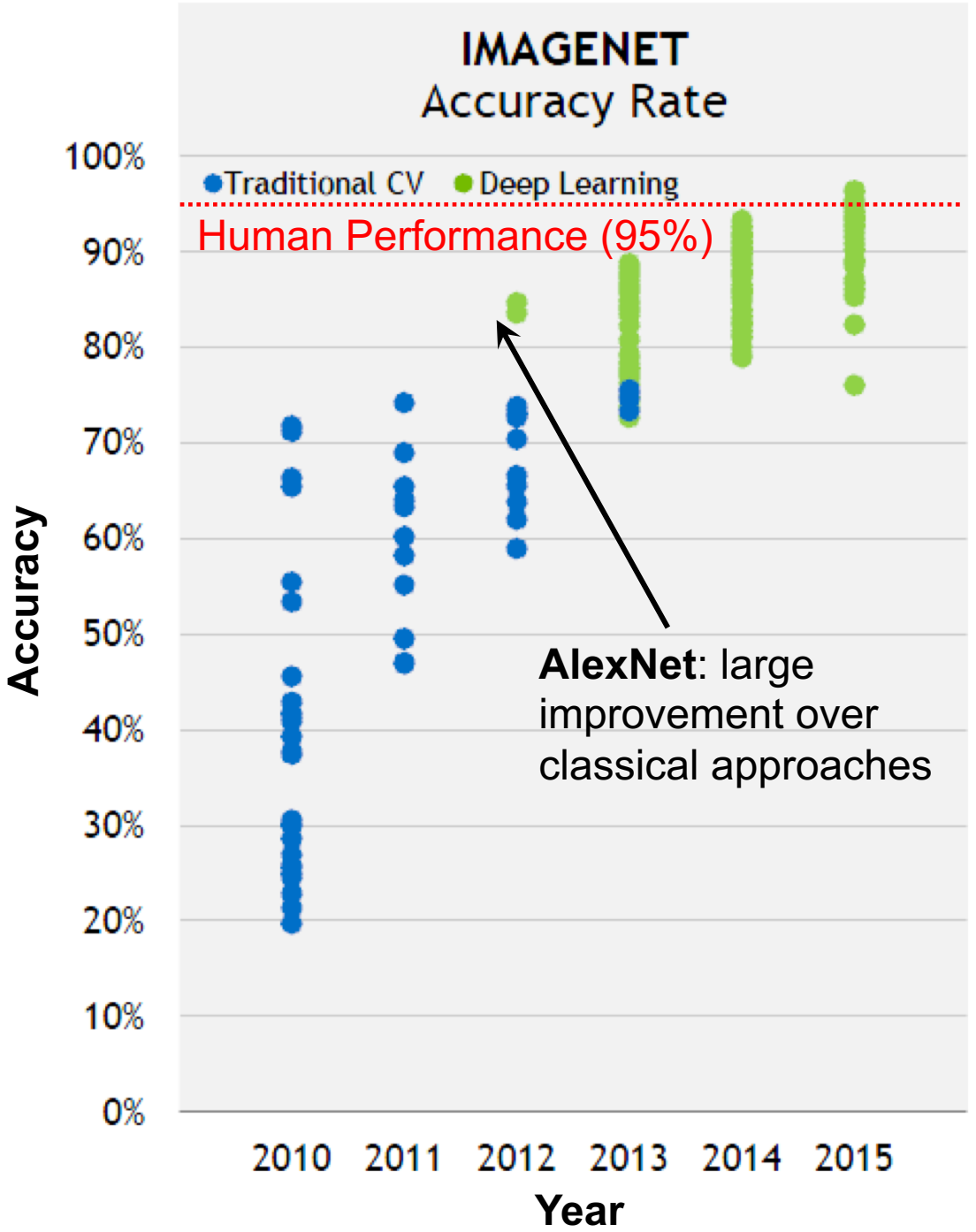
https://blog.google/products/translate/hallo-hola-ola-more-powerful-translate/

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR 2016*.

DEEP LEARNING AND IMAGE CLASSIFICATION



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", 2009.



CLASSIFICATION OVERVIEW

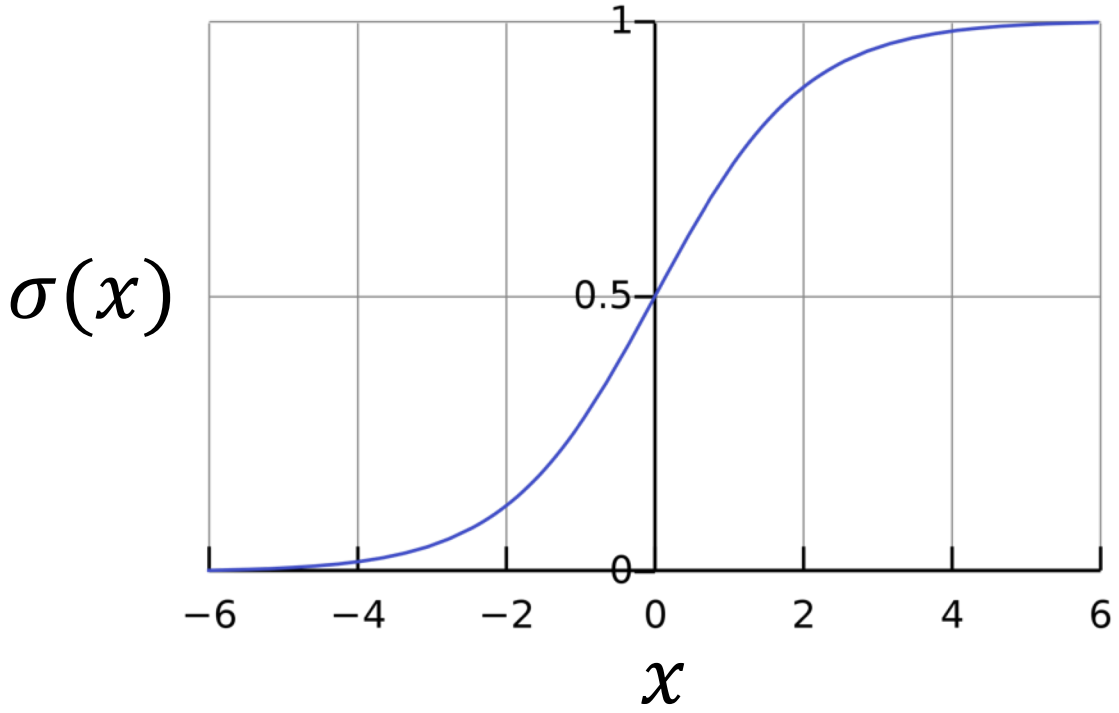
1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. **Neural Networks**
   a) Introduction
   b) Logistic Regression
   c) Deep Learning

SIGMOID FUNCTION

The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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



SIGMOID FUNCTION

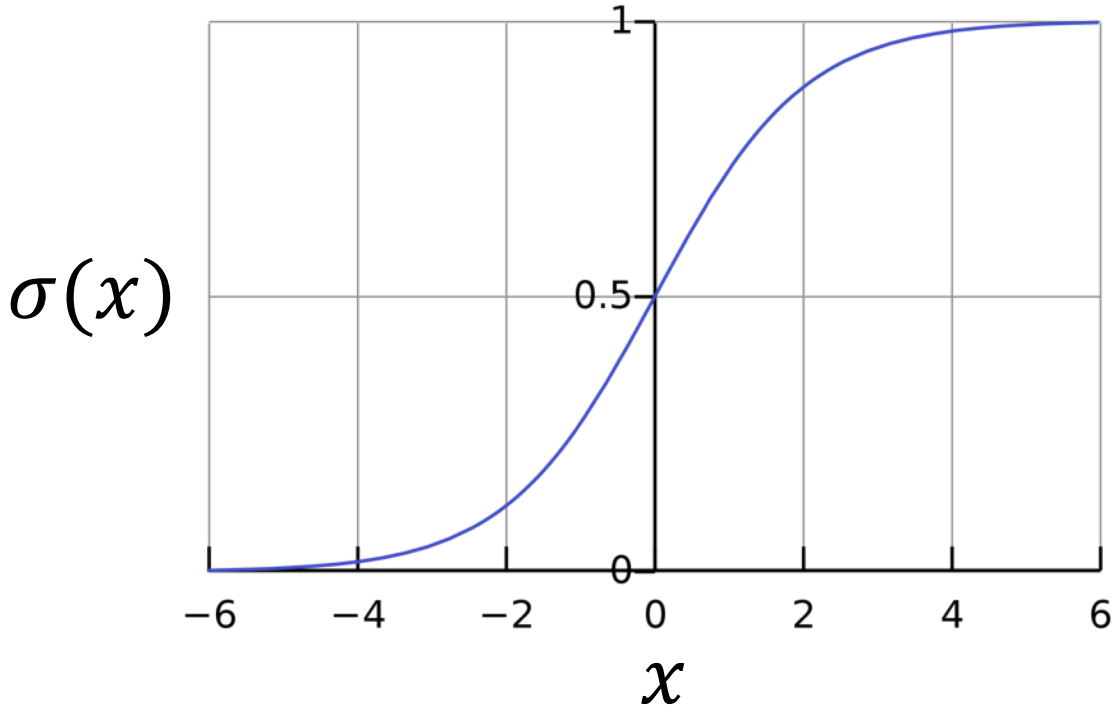
The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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

Check:

As $x \to -\infty, \sigma(x) \to \frac{1}{1+e^{\infty}} = 0$



SIGMOID FUNCTION

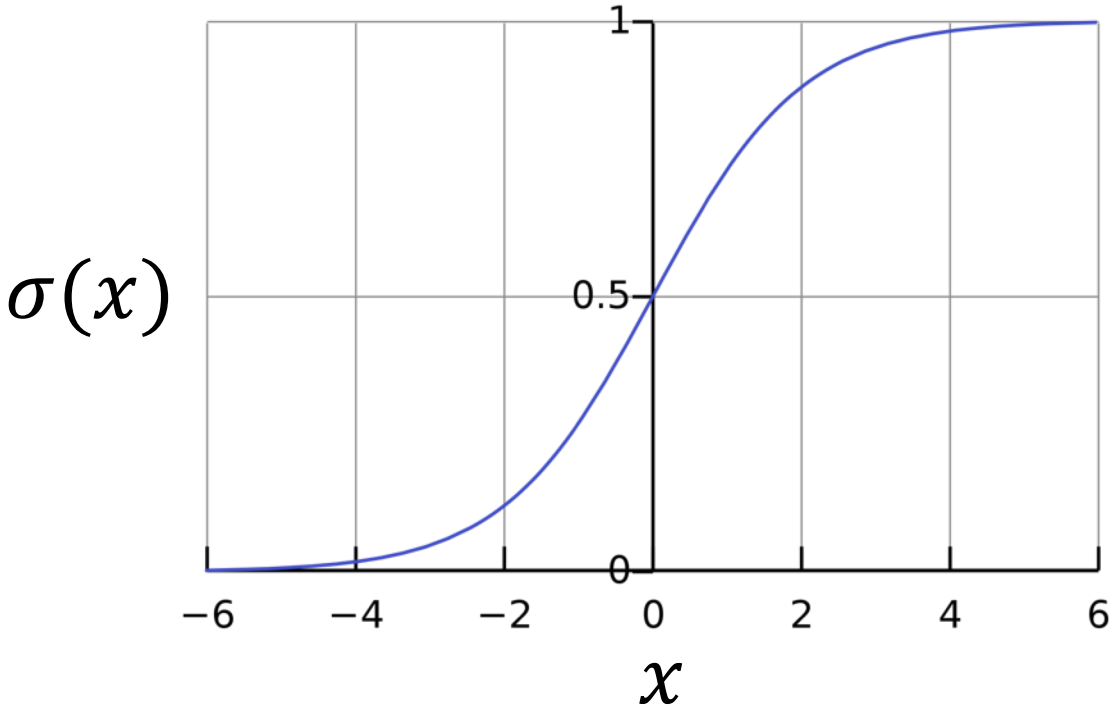
The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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

Check:

$$\sigma(0) = \frac{1}{1+e^{0}} = 0.5$$



SIGMOID FUNCTION

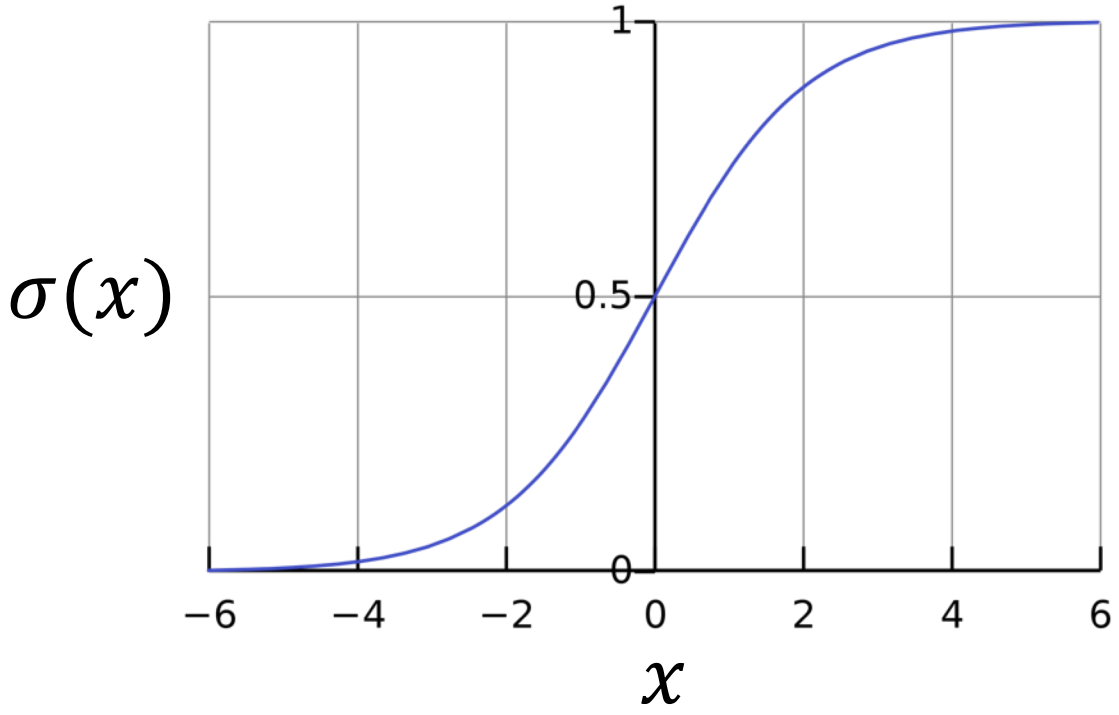
The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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

Check:

As $x \to \infty, \sigma(x) \to \frac{1}{1+e^{-\infty}} = 1$



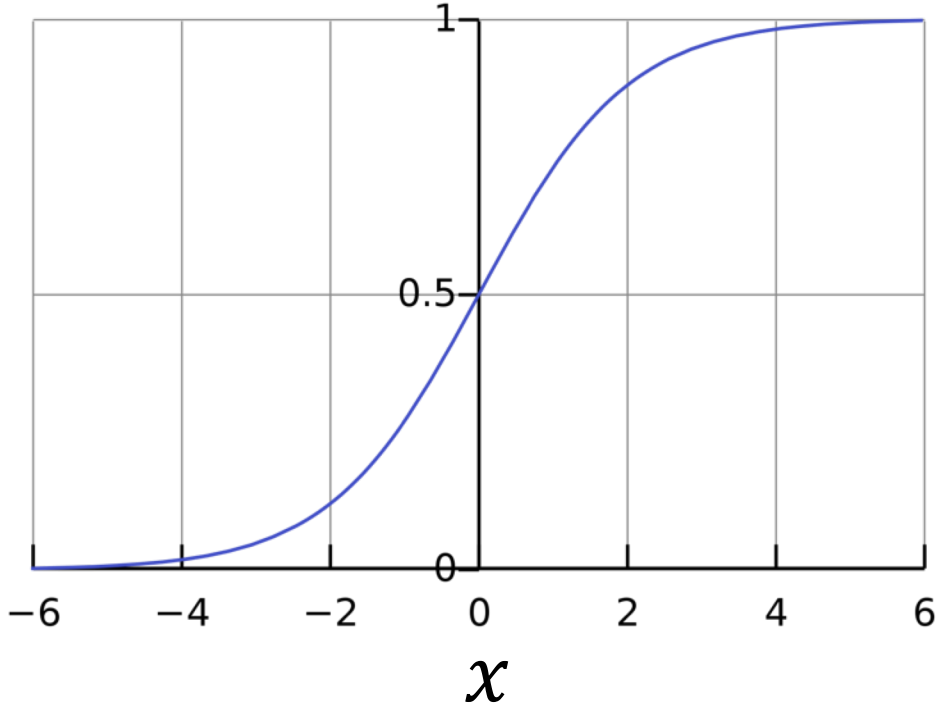
SIGMOID FUNCTION

The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

Main benefit: can be interpreted as a probability

It is defined as:

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



Check:

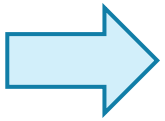
As $x \to \infty, \sigma(x) \to \frac{1}{1+e^{-\infty}} = 1$

RUNNING EXAMPLE: SPAM CLASSIFICATION

Features

Number of '$'	Number of '!'
5	2

$$x = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$



(Predicted probability of the email being spam)

?

$\hat{y}$

PARAMETERS OF LOGISTIC REGRESSION

Features

Number of '$'	Number of '!'
5	2

$x = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$

(Predicted probability of the email being spam)

? $\hat{y}$

Parameters:

Weights $w = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

1	2

Bias $b = -5$

-5

PREDICTION FUNCTION OF LOGISTIC REGRESSION

Features

Number of '$'	Number of '!'
5	2

$x = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$

(Predicted probability of the email being spam)

? $\hat{y}$

Parameters:

Weights $w = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

1	2

Bias $b = -5$

-5

Prediction:

Sigmoid function, Dot product

$$\hat{y} = \sigma(x \cdot w + b) = \sigma\left(\begin{pmatrix} 5 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} - 5\right) = \sigma(9 - 5) = \frac{1}{1 + e^{-4}} = 0.982$$

QUIZ: SPAM CLASSIFICATION

Features

Number of '$'	Number of '!'
1	0

$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

(Predicted probability of the email being spam)

? $\hat{y}$

Parameters:

Weights $w = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

1	2

Bias $b = -5$

-5

Prediction: What is the predicted probability $\hat{y}$ for this new email $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$?

QUIZ: SPAM CLASSIFICATION

Features

Number of '$'	Number of '!'
1	0

$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

(Predicted probability of the email being spam)

? $\hat{y}$

Parameters:

Weights $w = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ | Bias $b = -5$

1	2	-5

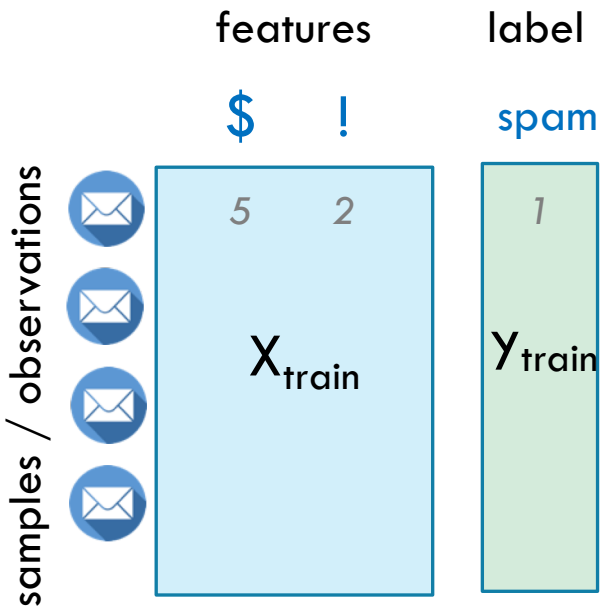
Prediction:

Sigmoid function, Dot product

$$\hat{y} = \sigma(x \cdot w + b) = \sigma\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} - 5\right) = \sigma(1 - 5) = \frac{1}{1 + e^4} = 0.018$$

TRAINING LOGISTIC REGRESSION

Training Data



Logistic Regression Parameters

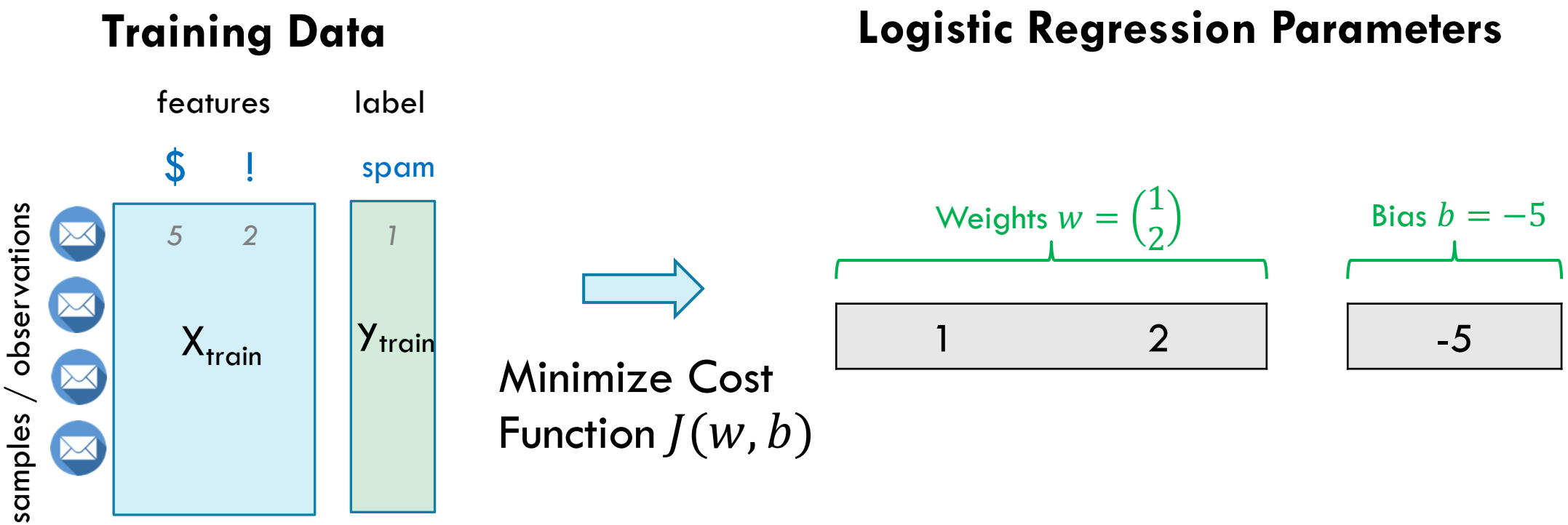
Weights $w = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

Bias $b = -5$

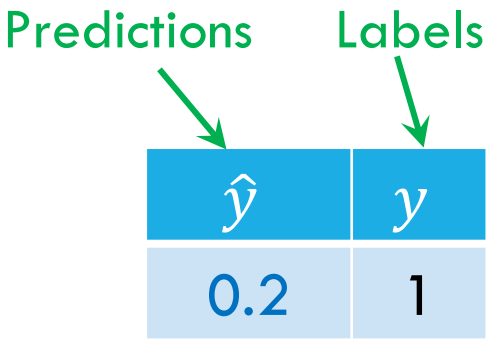
1	2

-5

TRAINING LOGISTIC REGRESSION



CROSS-ENTROPY LOSS (FOR A SINGLE DATA POINT)



Cross Entropy Loss (for a single y and $\hat{y}$):

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

How to interpret this?

The log probability of observing y given $\hat{y}$ is

$$\log \mathrm{P}(\mathrm{y}|\hat{y}) = \begin{cases} \log \hat{y} & \text{if } y = 1 \\ \log(1 - \hat{y}) & \text{if } y = 0 \end{cases}$$

This is exactly the negative of cross entropy loss.

Thus, $L(\hat{y}, y)$ can be interpreted as **negative log-likelihood** of observing y given $\hat{y}$

CROSS-ENTROPY LOSS (FULL DATA)

Now the labels and predictions are vectors, $\mathbf{y}$ and $\hat{\mathbf{y}}$:

Predictions Labels

$\hat{\mathbf{y}}$	$\mathbf{y}$
0.2	1
0.5	0
0.9	1

Cross Entropy Loss (for full data $\mathbf{y}$ and $\hat{\mathbf{y}}$):

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{n} -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$

COST FUNCTION J FOR PARAMETERS

Cost function $J(w, b)$ for parameters $\boldsymbol{w}, \boldsymbol{b}$ is defined as the cross entropy loss of the predictions $\hat{y}_i$ obtained from w, b**:**

$$J(w,b) = L(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{n} -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$

Interpretation: recall that cross-entropy loss $L(\hat{\mathbf{y}}, \mathbf{y})$ represents the "disagreement" between our predictions $\hat{\mathbf{y}}$ and the labels $\mathbf{y}$. Now, we are trying to find the "ideal" values of w and b that minimize this disagreement (note that $\hat{\mathbf{y}}$ is a function of w and b).

COST FUNCTION J FOR PARAMETERS

Cost function $J(w, b)$ **for parameters** $\boldsymbol{w}, \boldsymbol{b}$ is defined as the cross entropy loss of the predictions $\hat{y}_i$ obtained from w, b**:**

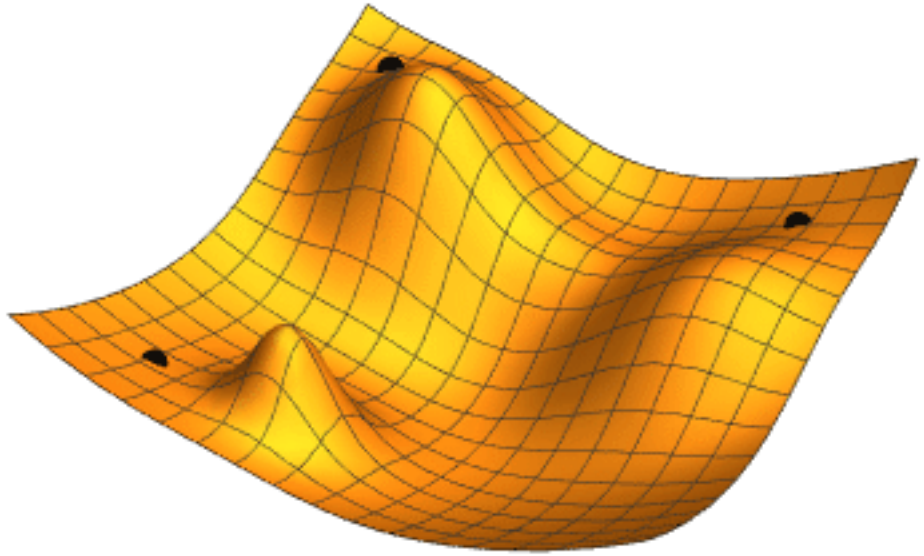
$$J(w, b) = L(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{n} -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$

Goal: find w, b to minimize $J(w, b)$

Approach: gradient descent, an incremental approach that repeatedly makes small changes to w, b to gradually decrease $J(w, b)$

MINIMIZING COST FUNCTIONS: GRADIENT DESCENT

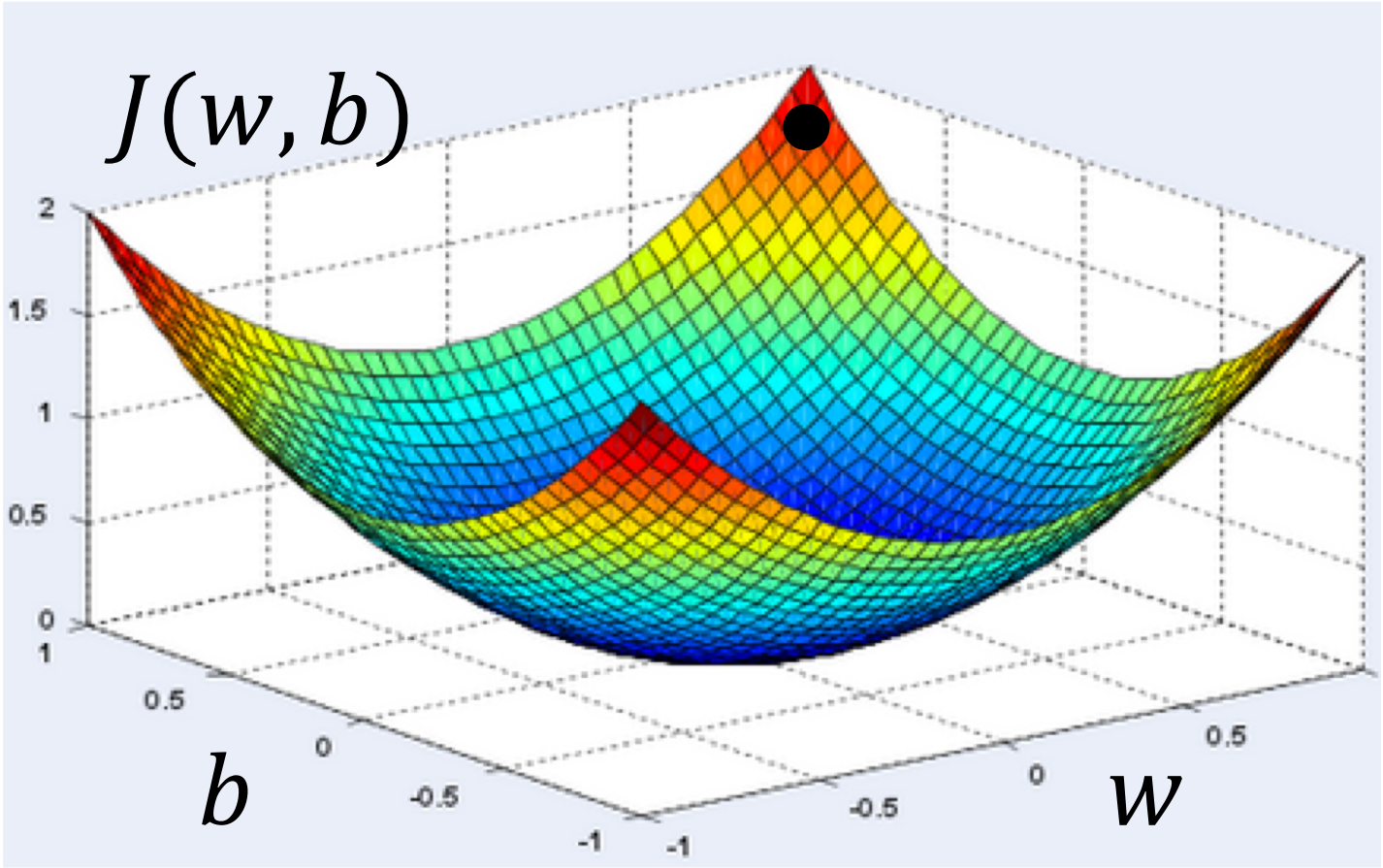
MINIMIZING COST FUNCTIONS: GRADIENT DESCENT



MINIMIZING COST FUNCTION J

We want to find w, b to minimize $J(w, b)$

Start at an arbitrary point, then keep moving in the negative gradient direction until convergence



MINIMIZING COST FUNCTION J

We want to find w, b to minimize $J(w, b)$

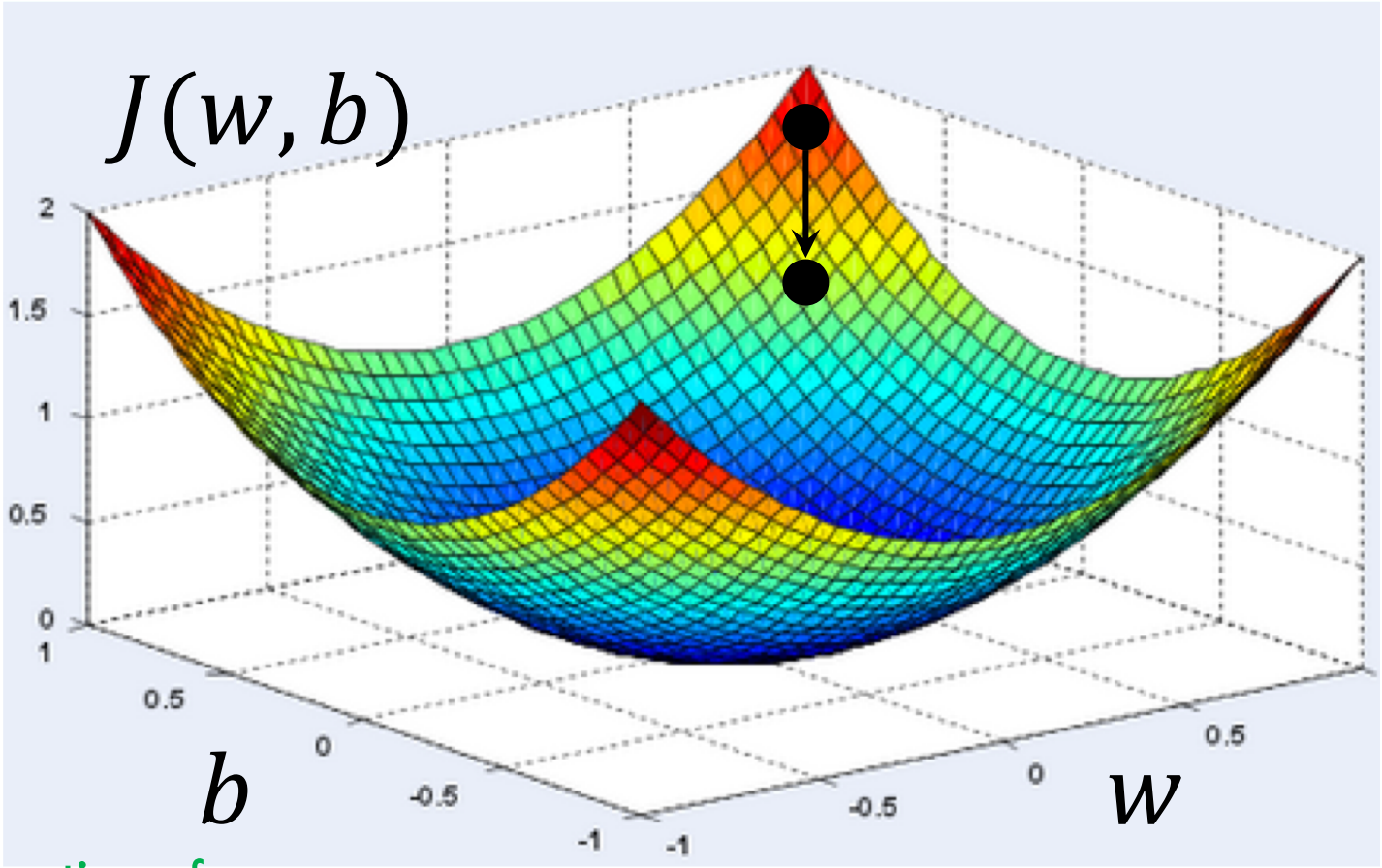
Start at an arbitrary point, then move in the negative gradient direction

$$w \leftarrow w - \eta \frac{\delta J(w,b)}{\delta w}$$

$$b \leftarrow b - \eta \frac{\delta J(w,b)}{\delta b}$$

"Learning Rate"

"Slope": direction of steepest decrease of J



MINIMIZING COST FUNCTION J

We want to find w, b to minimize $J(w, b)$

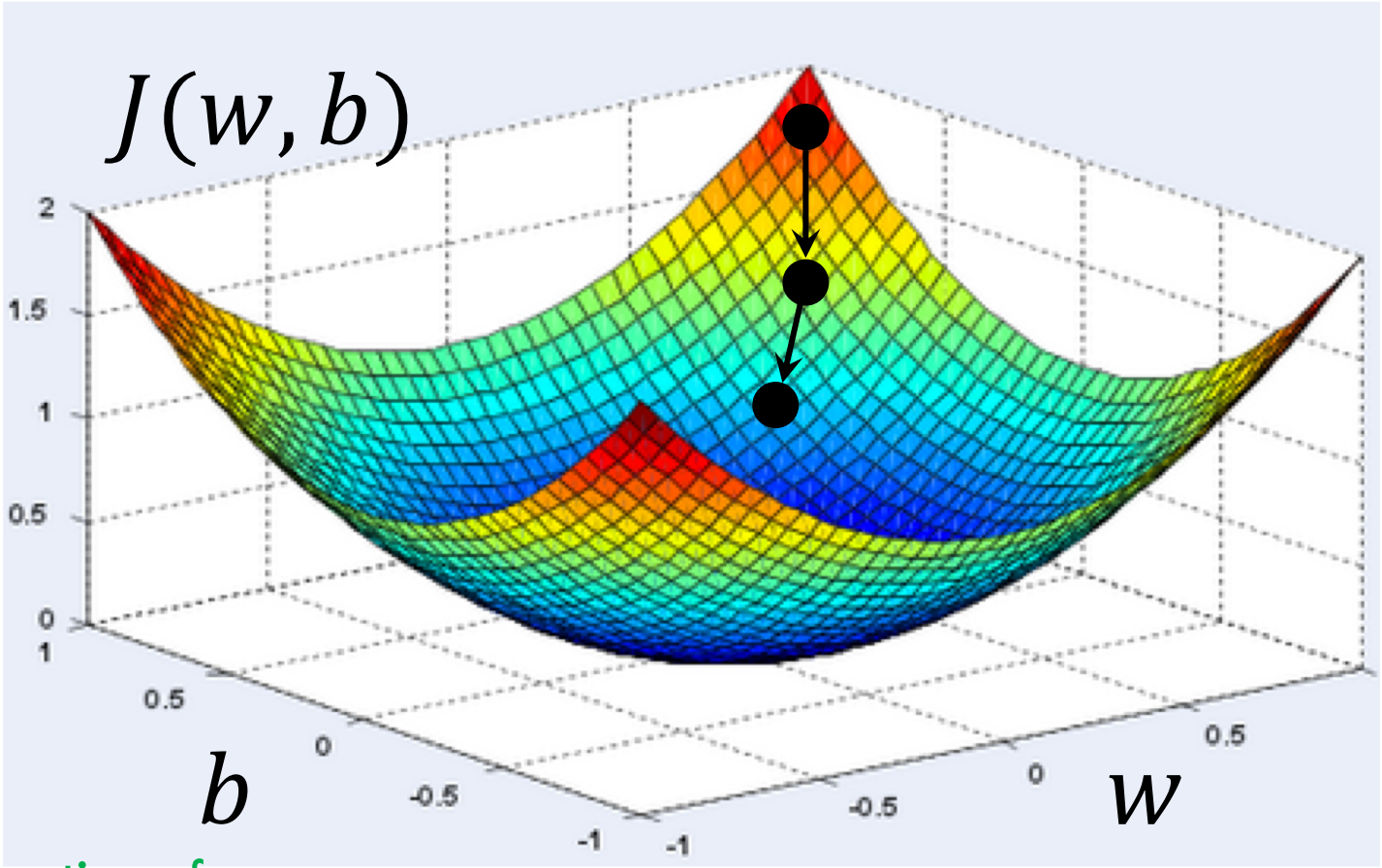
Start at an arbitrary point, then move in the negative gradient direction

$$w \leftarrow w - \eta \frac{\delta J(w,b)}{\delta w}$$

$$b \leftarrow b - \eta \frac{\delta J(w,b)}{\delta b}$$

"Learning Rate"

"Slope": direction of steepest decrease of J



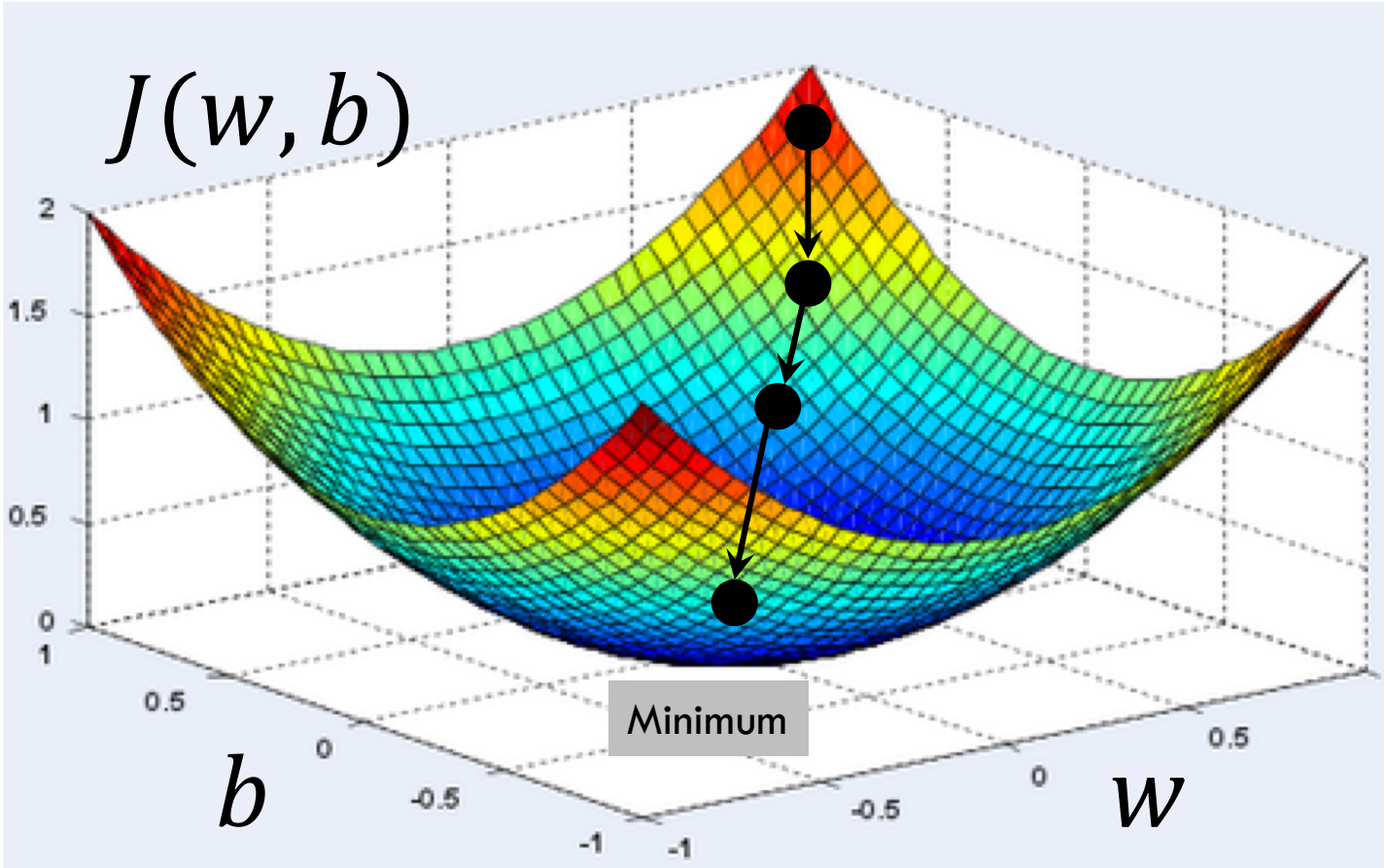
MINIMIZING COST FUNCTION J

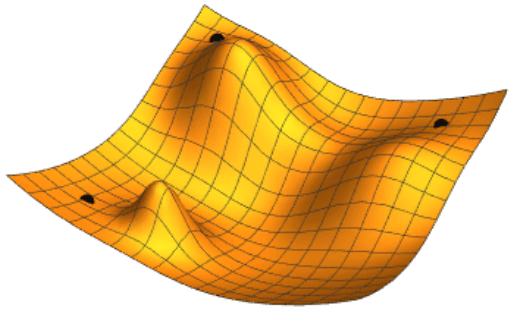
We want to find w, b to minimize $J(w, b)$

Start at an arbitrary point, then move in the negative gradient direction

Continue until convergence

- Stop when improvement in J is below a fixed threshold

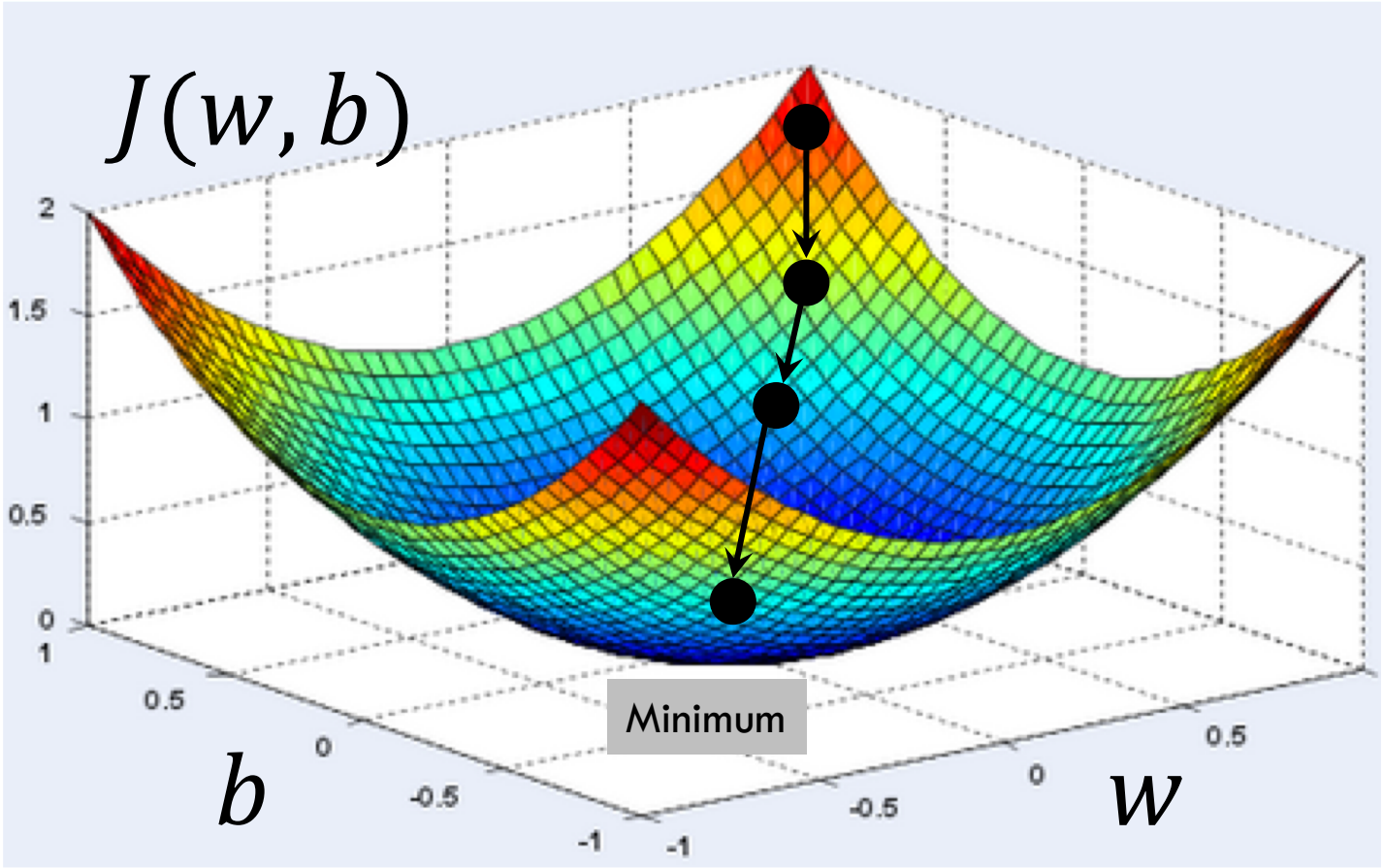




GLOBAL VS LOCAL MINIMUM

This does not always reach the “global minimum”; it may get stuck in a “local minimum”

But for logistic regression, it always reaches the global minimum (due to “convexity” of J)



GRADIENT DESCENT: GENERAL ALGORITHM

Important concept: learning rate η

- Scaling factor for gradient (typical range: 0.01 - 0.0001)

$$\nabla_\theta L = \frac{\partial L}{\partial \theta} = \begin{bmatrix} \frac{\partial L}{\partial \theta_0} \\ \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_d} \end{bmatrix}$$

Input : data (X, y), loss function L, learning rate η

Initialization : Set θ to random values

while true :

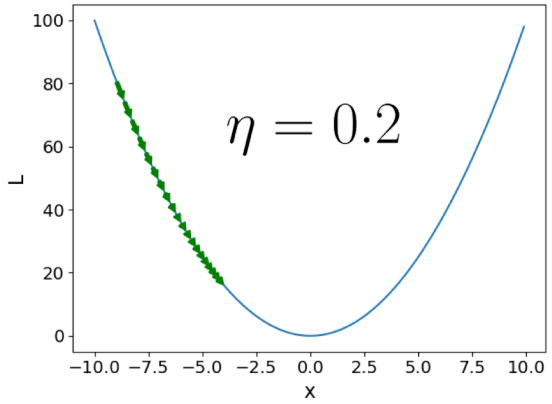
Calculate gradient $\nabla_\theta L$

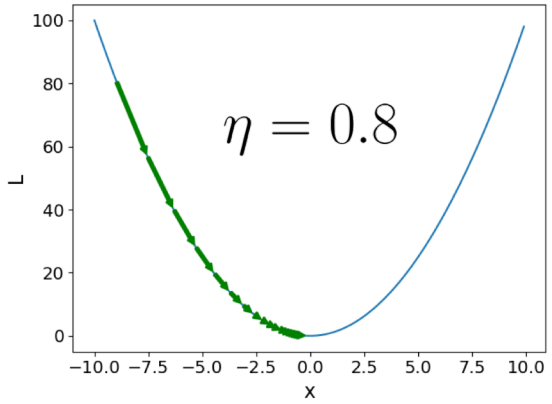
$\theta \leftarrow \theta - (\eta \cdot \nabla_\theta L)$

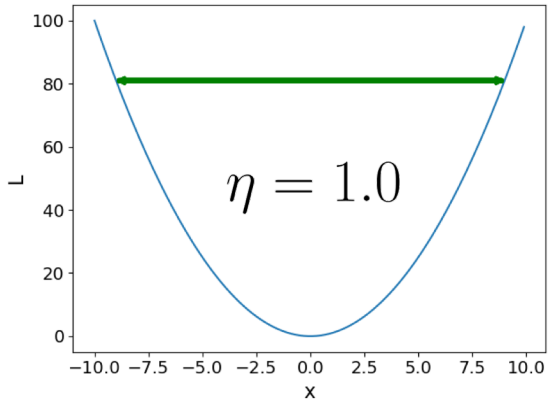
In practice: stop loop when loss converges

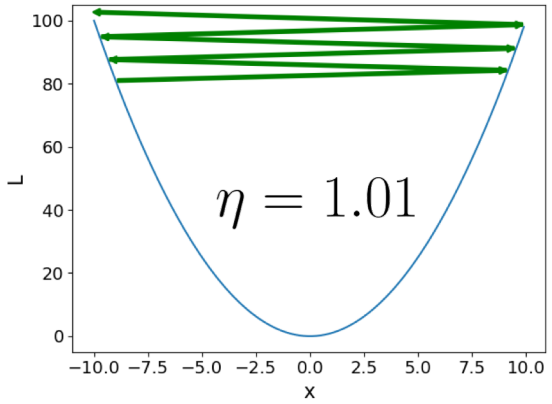
EFFECTS OF LEARNING RATE

$$L = x^2, \ \frac{\partial L}{\partial x} = 2x, \ 20 \text{ steps}$$







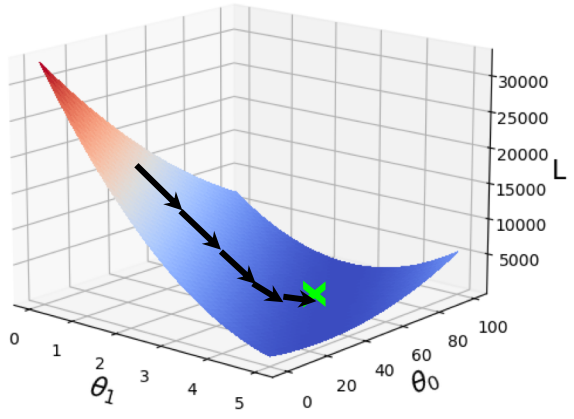


Too low learning rate: slow progress
Too high learning rate: unstable progress

GRADIENT DESCENT: VARIATIONS

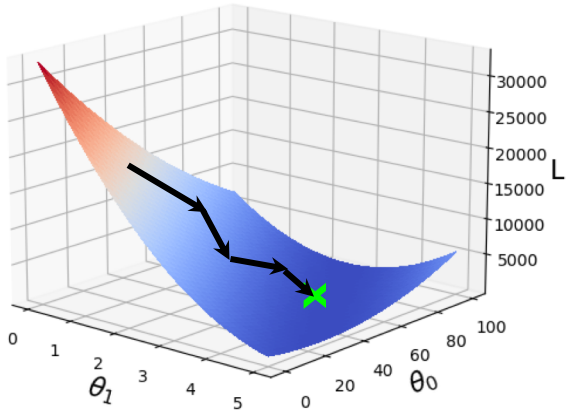
Full-Batch Gradient Descent

- calculate gradient and update θ over **whole training dataset**



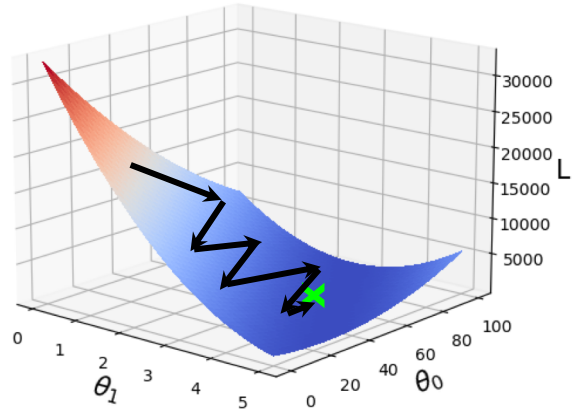
Mini-batch Gradient Descent

- calculate gradient and update θ over **randomly sampled batch of samples**



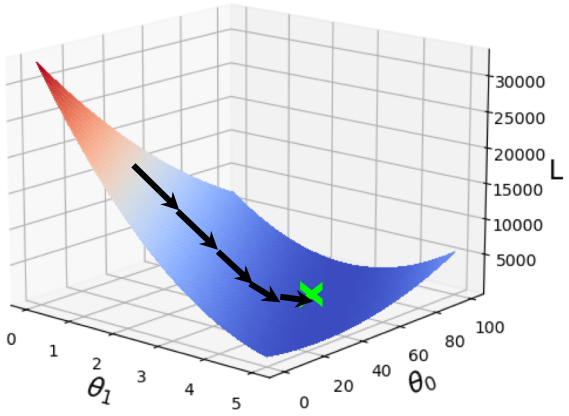
Stochastic Gradient Descent (SGD)

- calculate gradient and update θ over **single training sample**



GRADIENT DESCENT: VARIATIONS

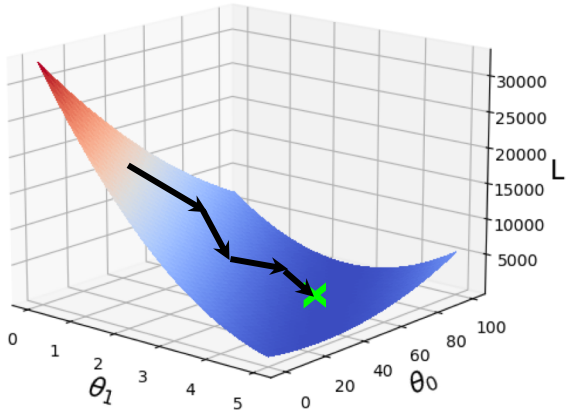
(Full-Batch) Gradient Descent



Gradient averaged over <u>all</u> data items

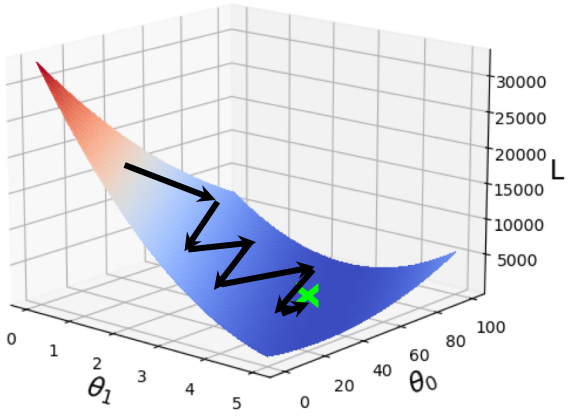
- Smooth descent
- Small(er) gradients
- Small(er) update steps

Mini-Batch Gradient Descent



Gradient averaged over <u>some</u> data items

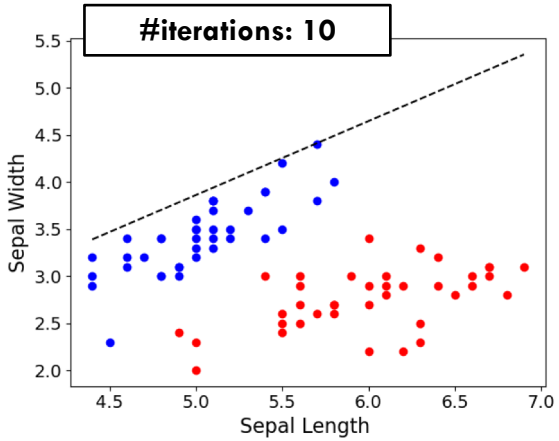
Stochastic Gradient Descent

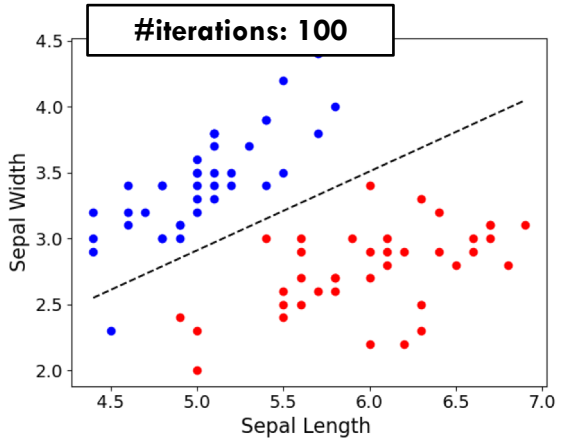


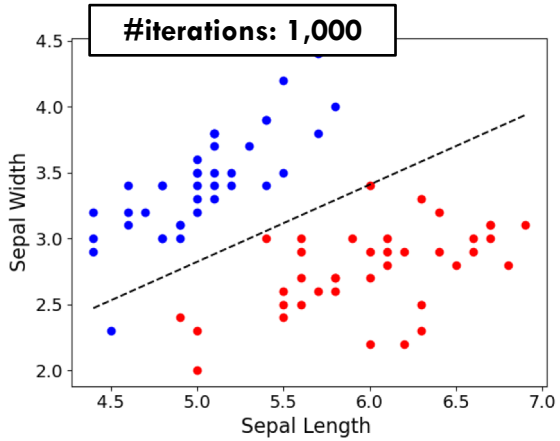
Gradient for each sample

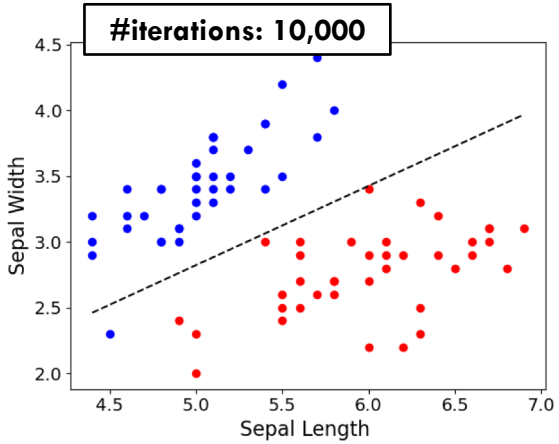
- Choppy descent
- Large(r) gradients
- Large(r) steps

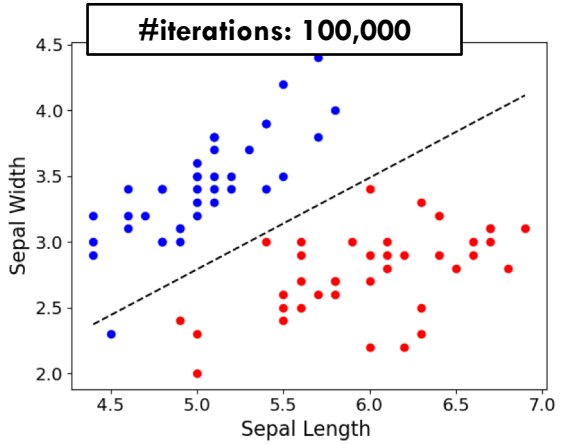
LOGISTIC REGRESSION: 2D EXAMPLE (Full-Batch Gradient Descent)

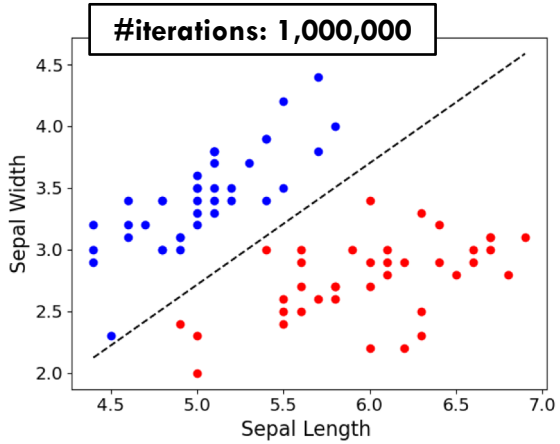












L1 AND L2 REGULARIZATION

L1 and L2 regularization are commonly used to control overfitting in logistic regression.

Given a **regularization parameter** λ, we modify the cost function $J(w, b)$ to:

$J(w, b) + \lambda \|w\|_1 = J(w, b) + \lambda \sum_{i=1}^{p} |w_i|$ *(L1 regularization)*

$J(w, b) + \frac{\lambda}{2} \|w\|_2^2 = J(w, b) + \frac{\lambda}{2} \sum_{i=1}^{p} w_i^2$ *(L2 regularization)*

We can fit these using gradient descent as before.

The penalties discourage complex (i.e. larger w) models, reducing overfitting. The larger λ is, the stronger the effect of the regularization. However, too large regularization can cause underfitting.
L1 regularization induces **sparsity**: i.e. generally, many entries of the fitted w will be exactly 0. However, L2 regularization does not induce sparsity.

DATA NORMALIZATION – YES OR NO?

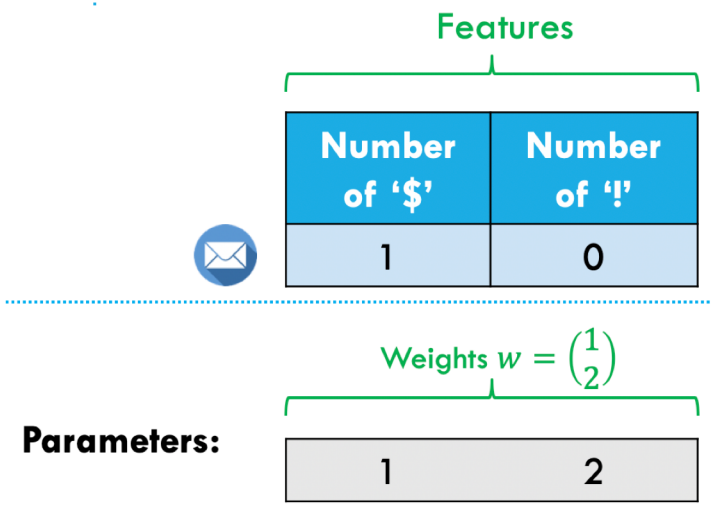
(E.g. standardization, min-max scaling)

Assuming no regularization, data normalization does not affect model performance

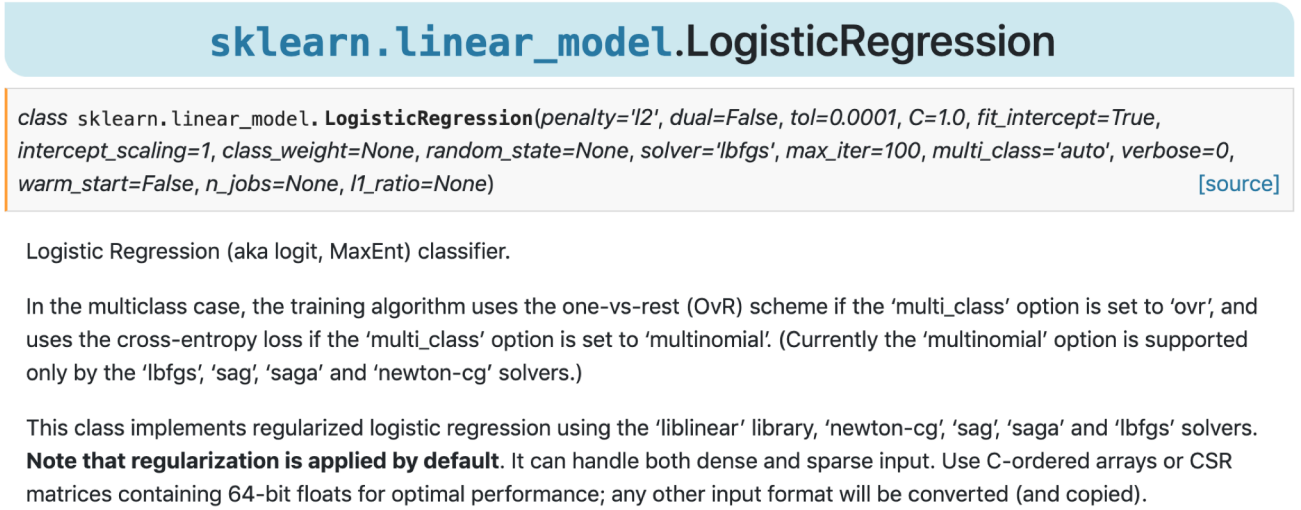
When using regularization: normalization can be seen as helping to regularize all variables "in a more balanced manner"

Ease of interpretation of coefficients w:

- Without normalization: coefficients can be interpreted based on the original data
- With normalization: coefficients can be interpreted as the importance of a feature (relative to other features)



TAKE NOTE: REGULARIZATON IN SCIKIT-LEARN LOGISTIC REGRESSION



sklearn.linear_model.LogisticRegression

class sklearn.linear_model. **LogisticRegression**(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

Scikit-learn logistic regression uses L2 regularization with $C = 1$ by default. $C\ (= \frac{1}{\lambda})$ is an inverse regularization parameter. If your feature values are unnormalized and have very different ranges (e.g. very small or large values), this regularization strength may not be appropriate.

PROS AND CONS OF LOGISTIC REGRESSION

Pros

- Fast and simple
- Loss function is convex
- Interpretable

Cons

- Linear model (up to before sigmoid layer)
- Cannot directly handle categorical features (need one-hot encoding)

QUIZ: INTERPRETABILITY OF COEFFICIENTS

Features

Number of '$'	Number of '!'

Q: In the spam classification example, assume we fit the following weight vector:

Weights $w = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

2	1

These weights can be interpreted as the "strength" of each feature. Which of the following emails will be given a higher probability of being spam?

Email A: $ $

Email B: ! ! !

QUIZ: INTERPRETABILITY OF COEFFICIENTS

Features

Number of '$'	Number of '!'

Q: In the spam classification example, assume we fit the following weight vector:

Weights $w = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

2	1

These weights can be interpreted as the "strength" of each feature. Which of the following emails will be given a higher probability of being spam?

Email A: $ $

Email B: ! ! !

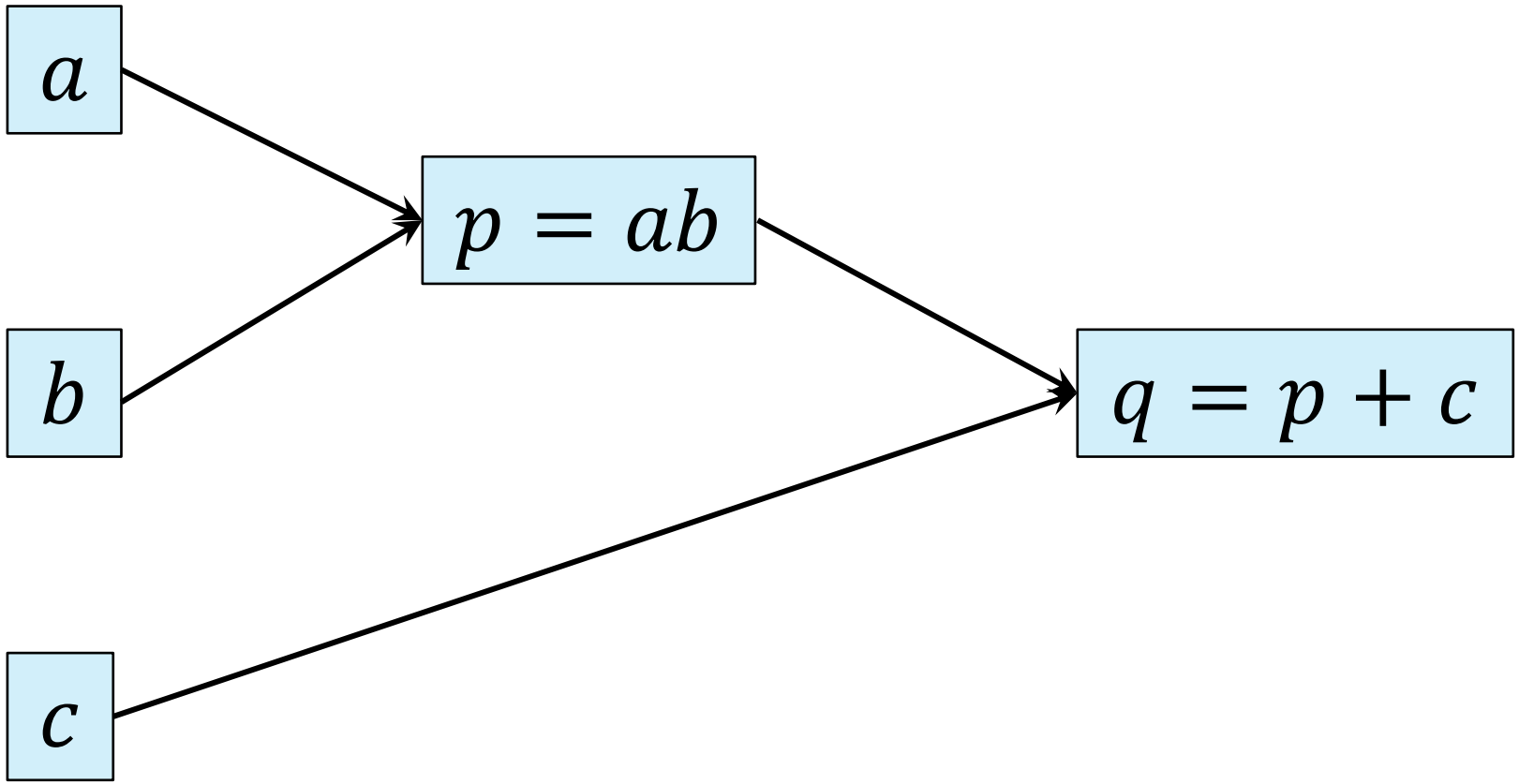
A: Email A

CLASSIFICATION OVERVIEW

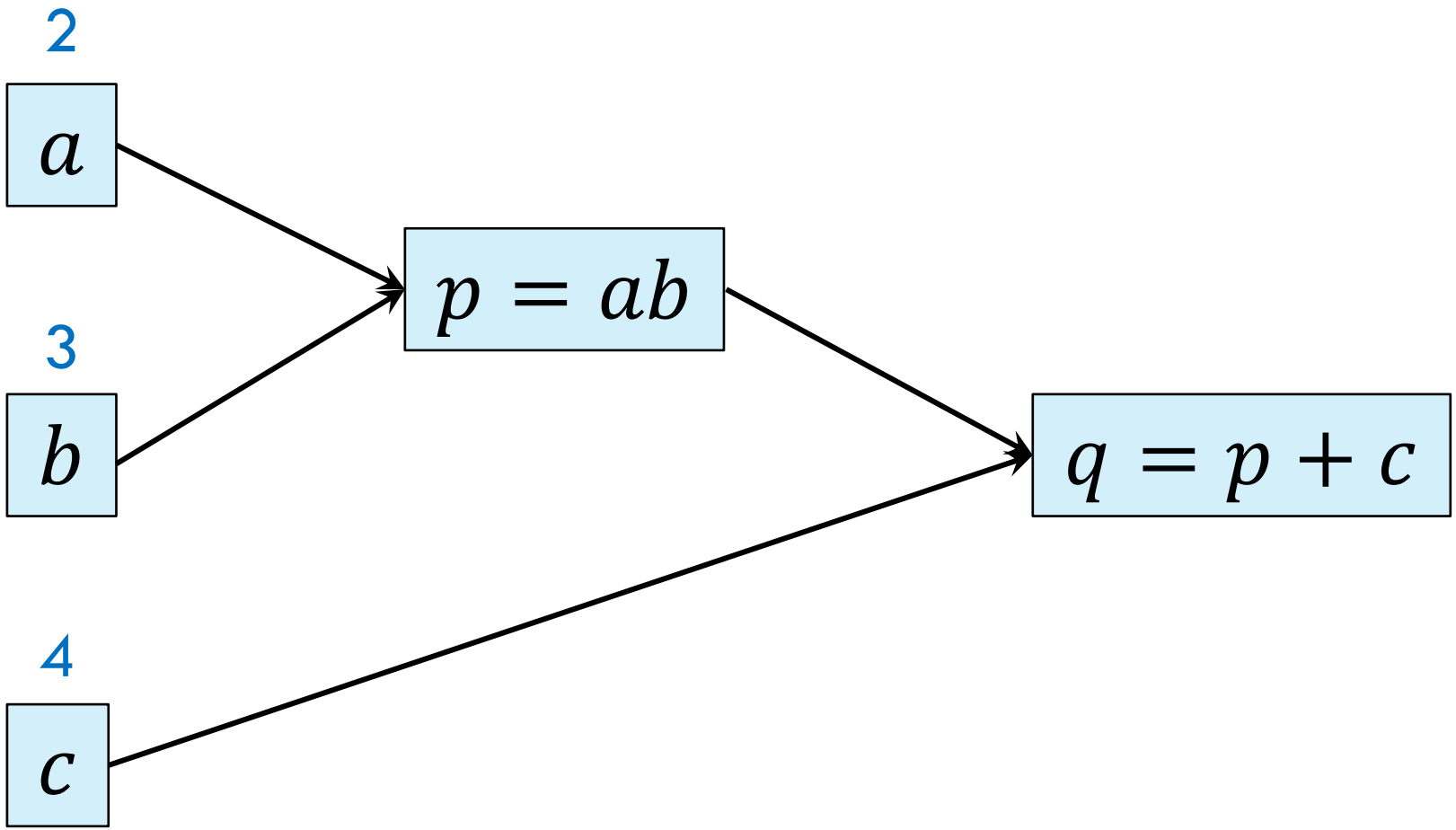
1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. **Neural Networks**
   a) Introduction
   b) Logistic Regression
   c) Deep Learning

BUILDING COMPLEX CLASSIFIERS FROM SIMPLE PARTS

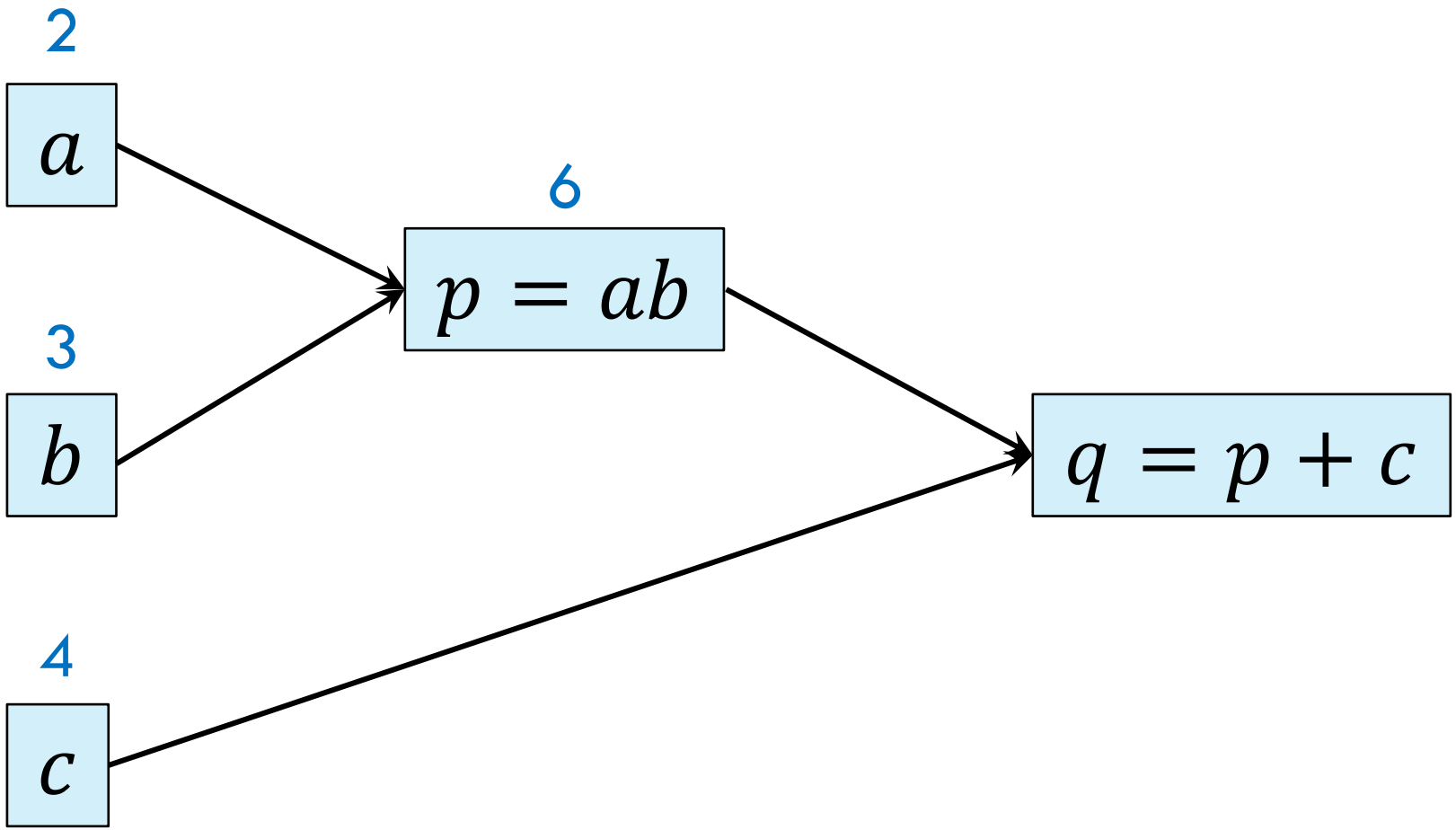
COMPUTATIONAL GRAPH



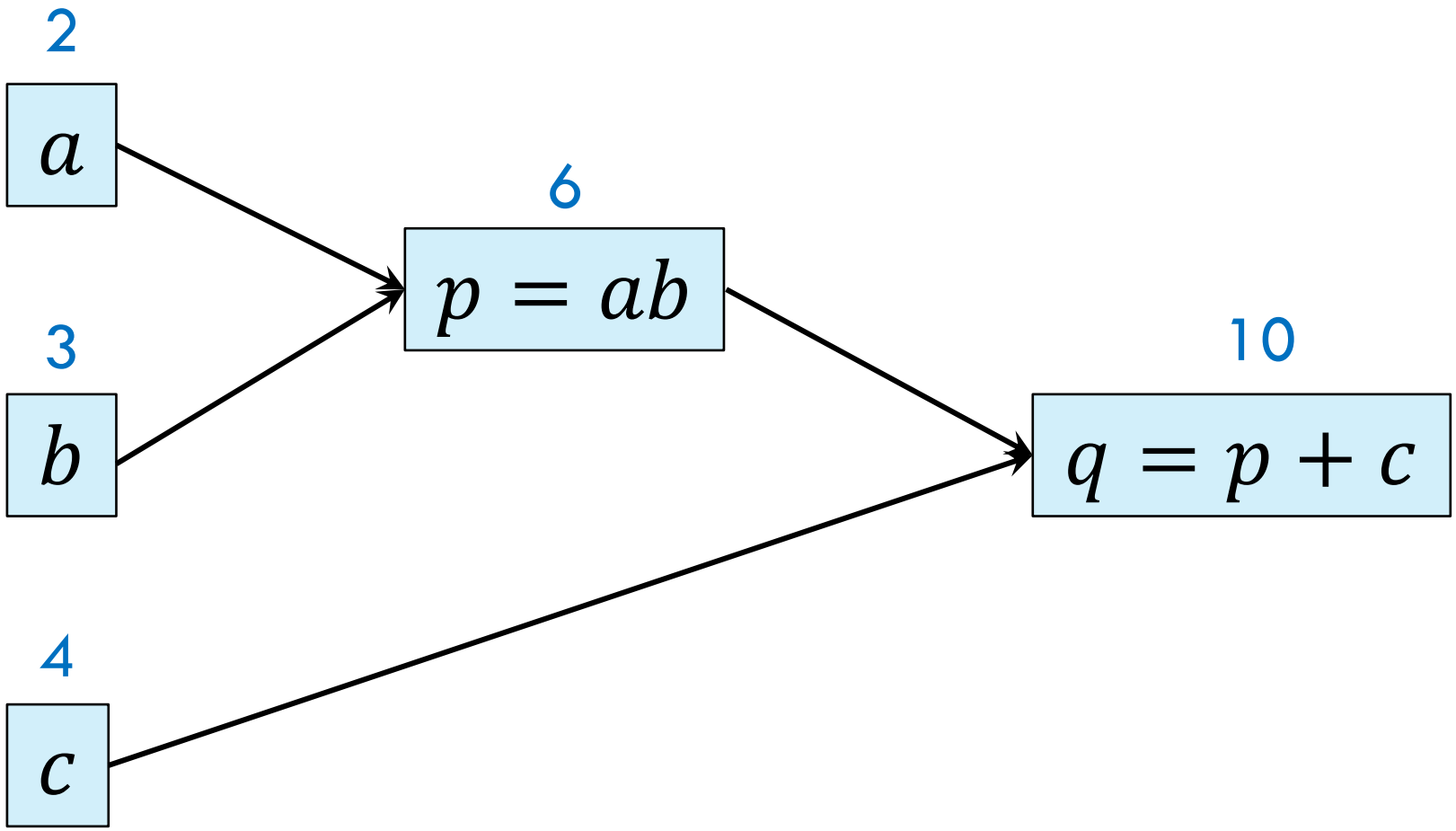
COMPUTATIONAL GRAPH: FORWARD PASS



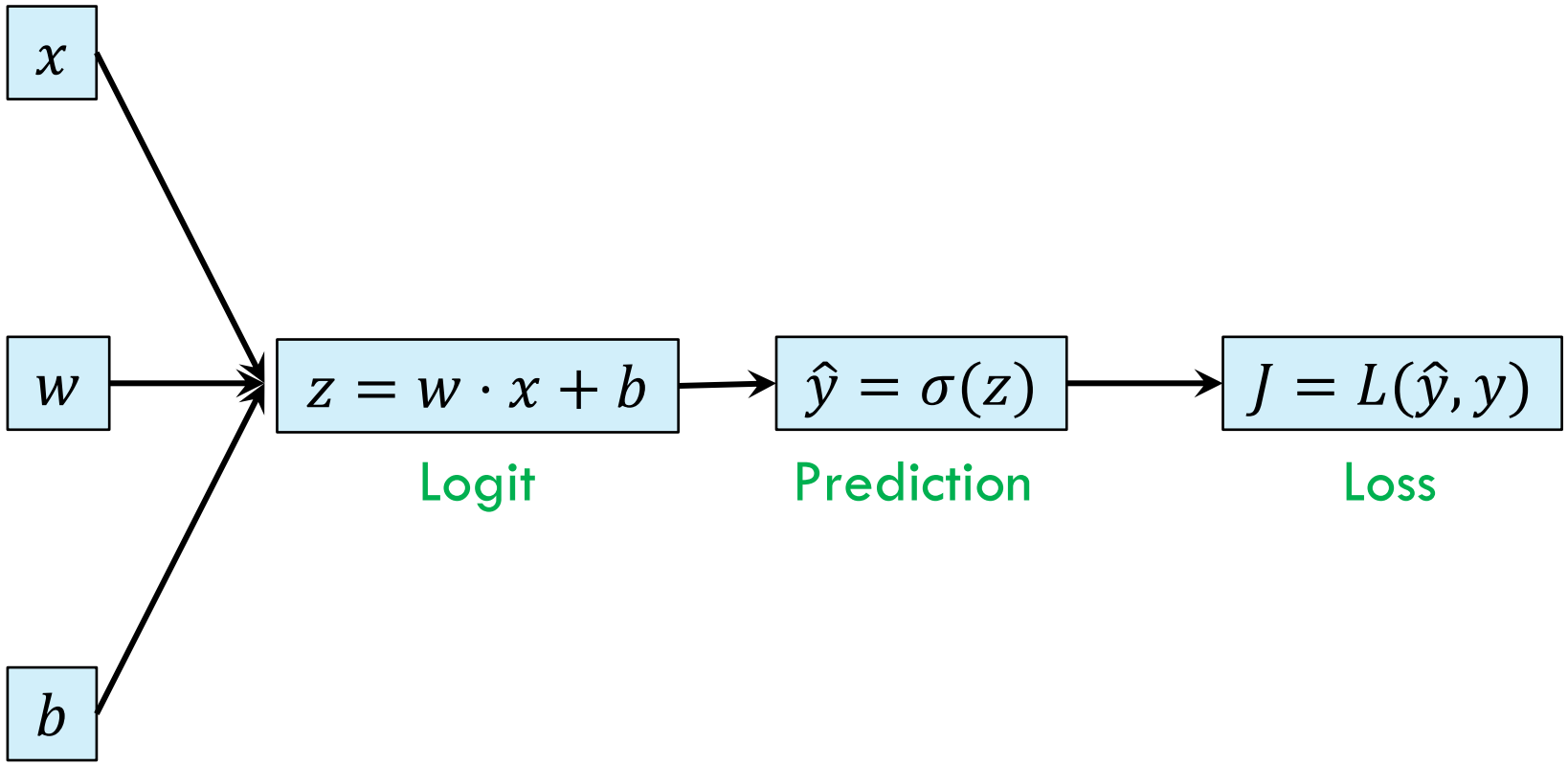
COMPUTATIONAL GRAPH: FORWARD PASS



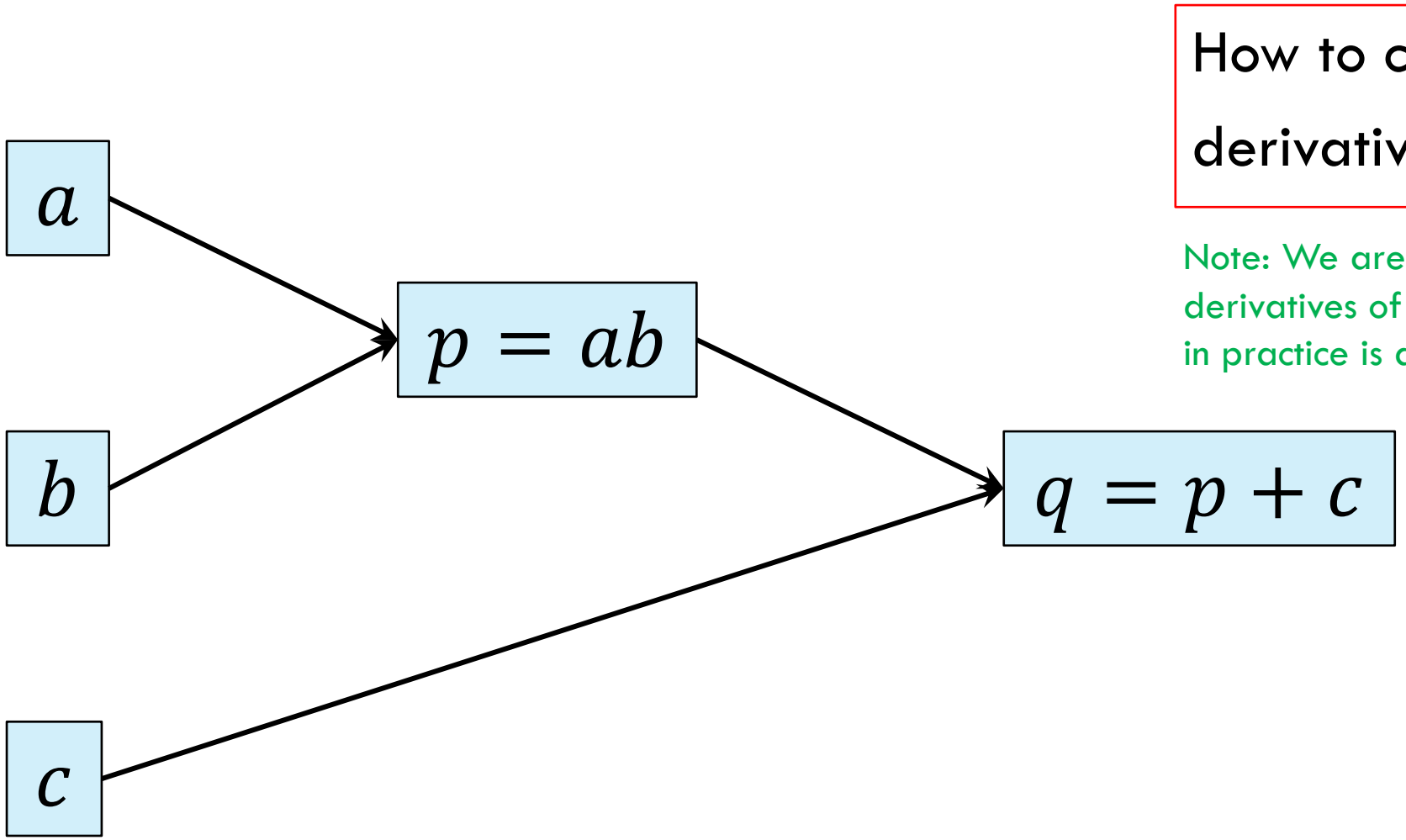
COMPUTATIONAL GRAPH: FORWARD PASS



LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH



COMPUTATIONAL GRAPH: BACKWARD PASS

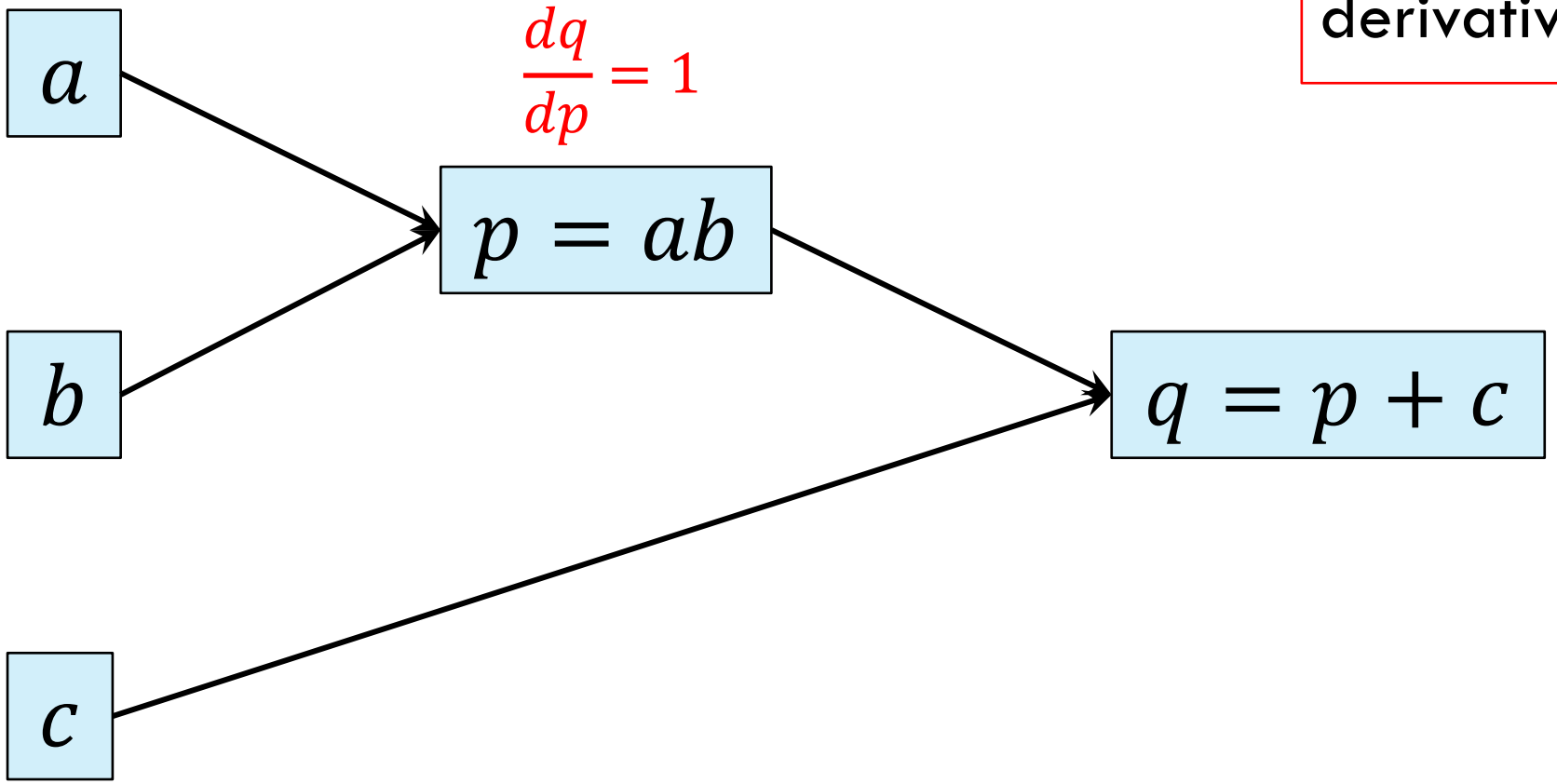


How to compute the derivatives $\frac{dq}{da}$, $\frac{dq}{db}$, etc.?

Note: We are mostly interested in the derivatives of the last variable (which in practice is almost always the loss)

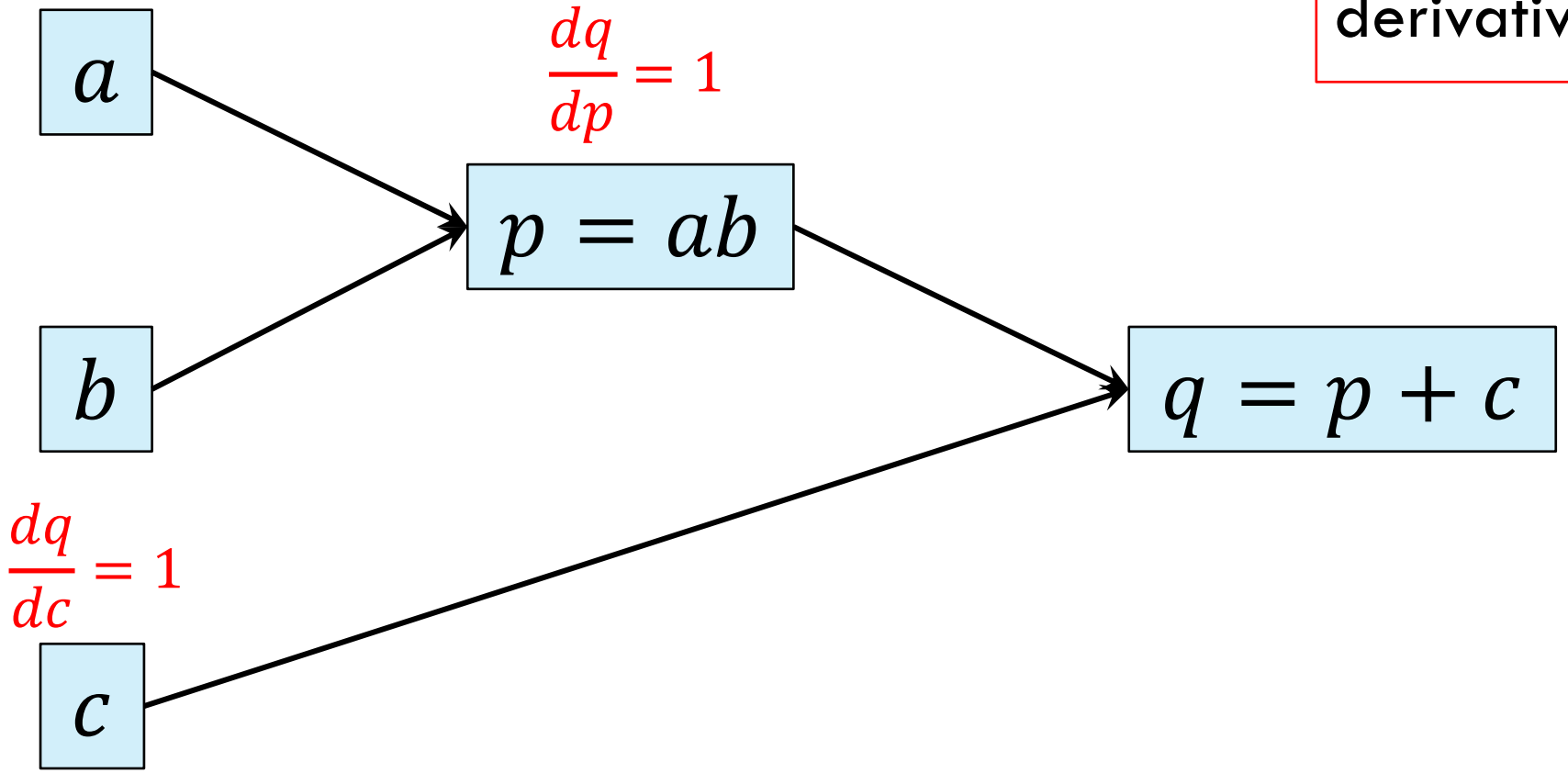
COMPUTATIONAL GRAPH: BACKWARD PASS

How to compute the derivatives $\frac{dq}{da}$, $\frac{dq}{db}$, etc.?

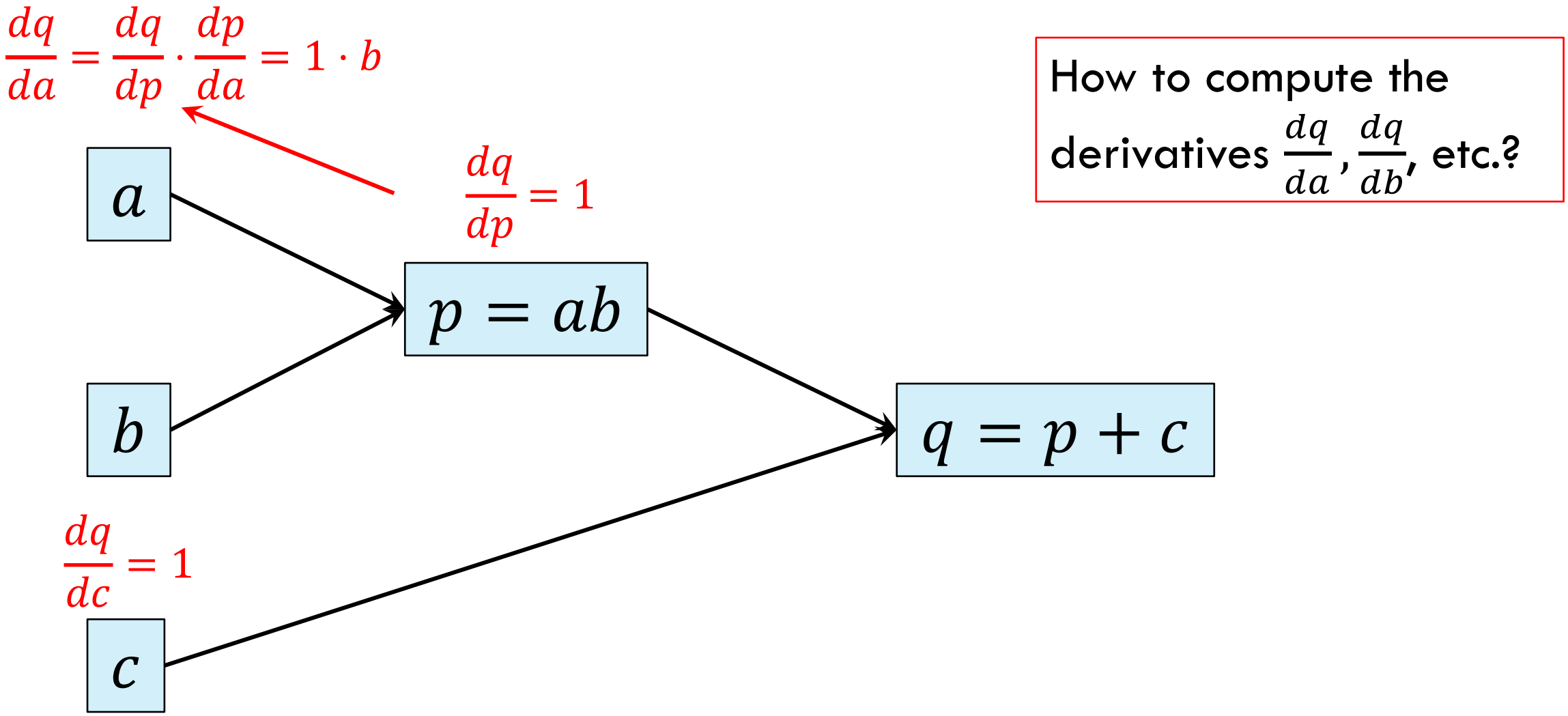


COMPUTATIONAL GRAPH: BACKWARD PASS

How to compute the derivatives $\frac{dq}{da}$, $\frac{dq}{db}$, etc.?

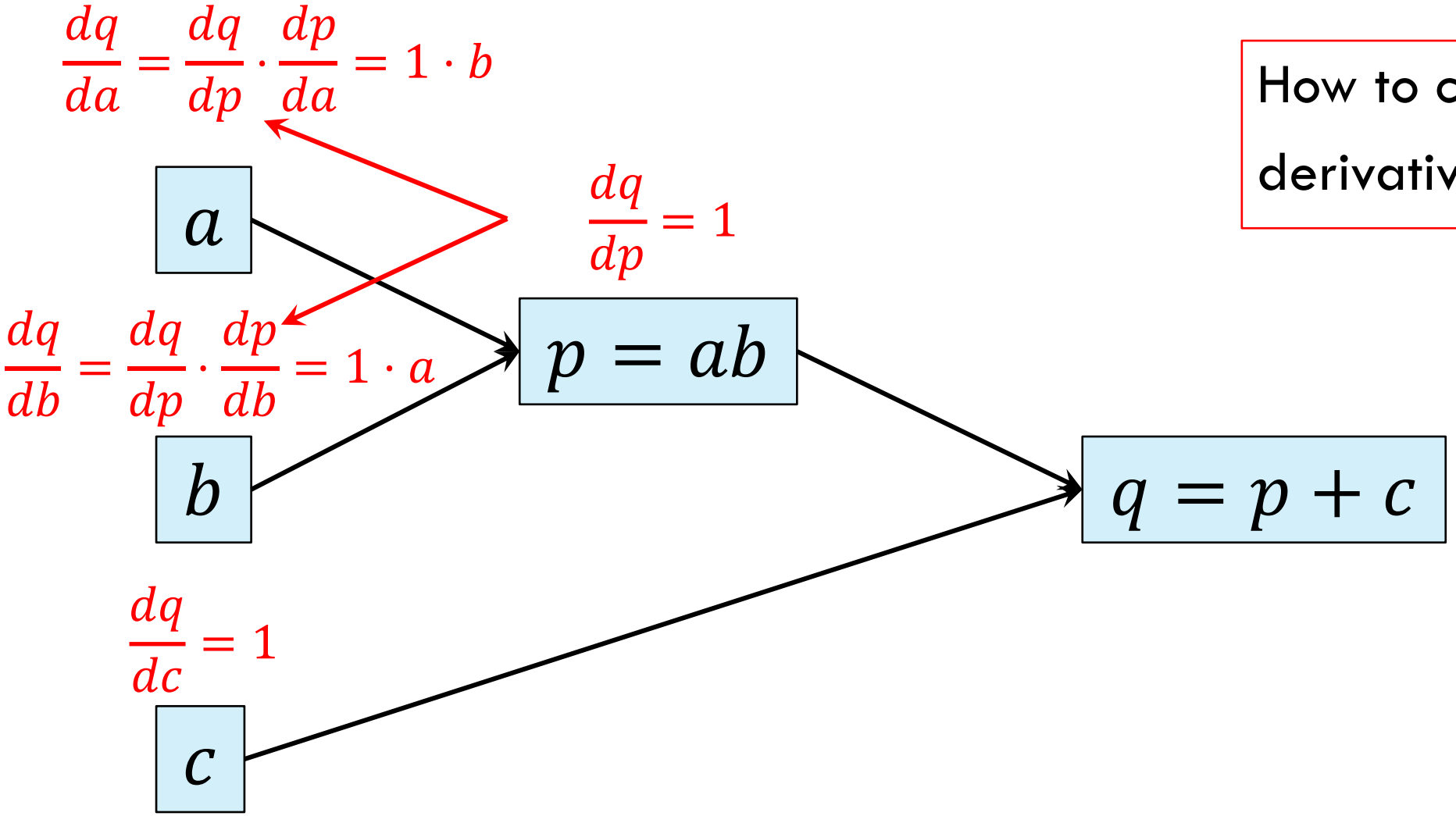


COMPUTATIONAL GRAPH: BACKWARD PASS



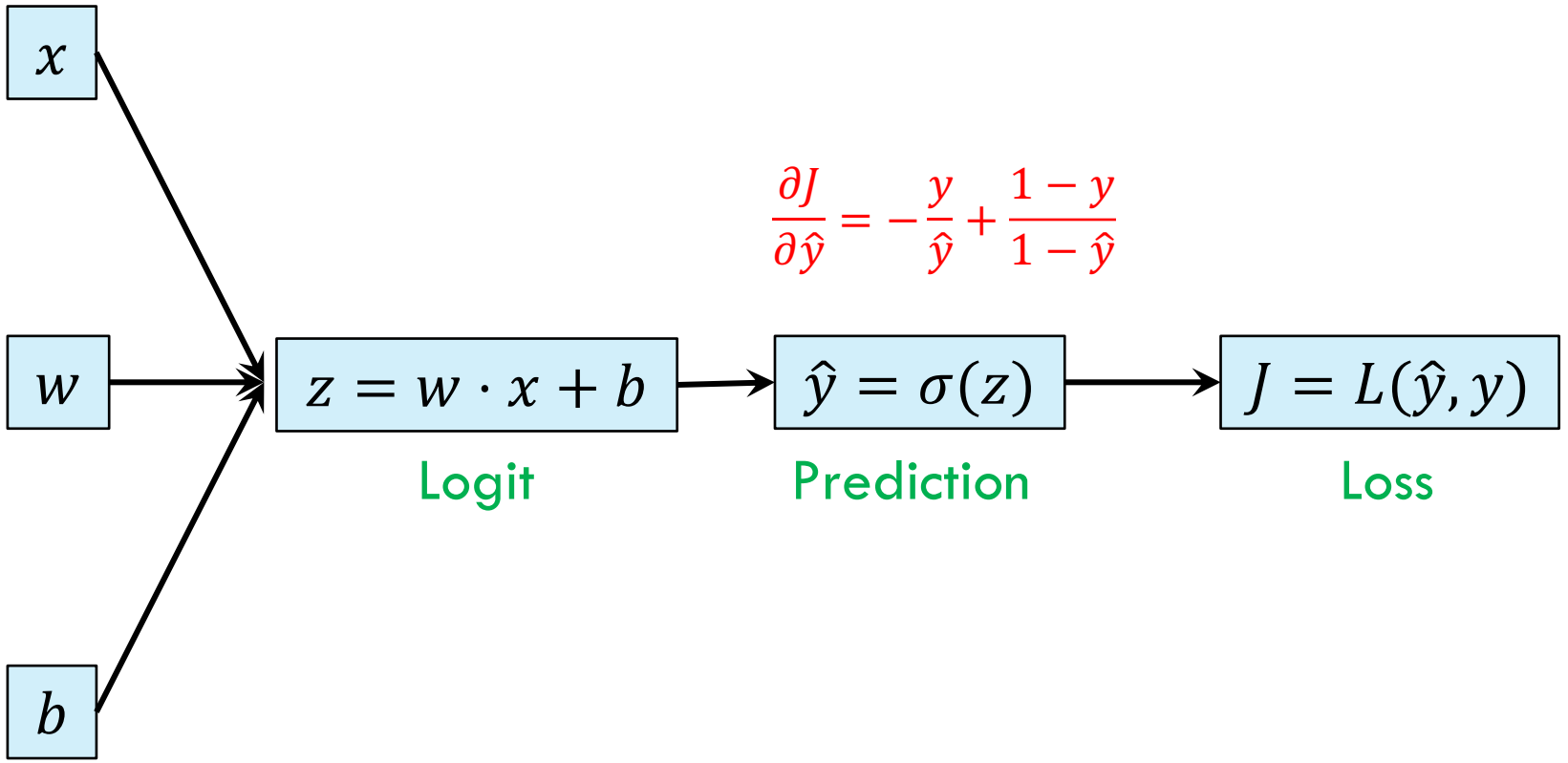
How to compute the derivatives $\frac{dq}{da}$, $\frac{dq}{db}$, etc.?

COMPUTATIONAL GRAPH: BACKWARD PASS

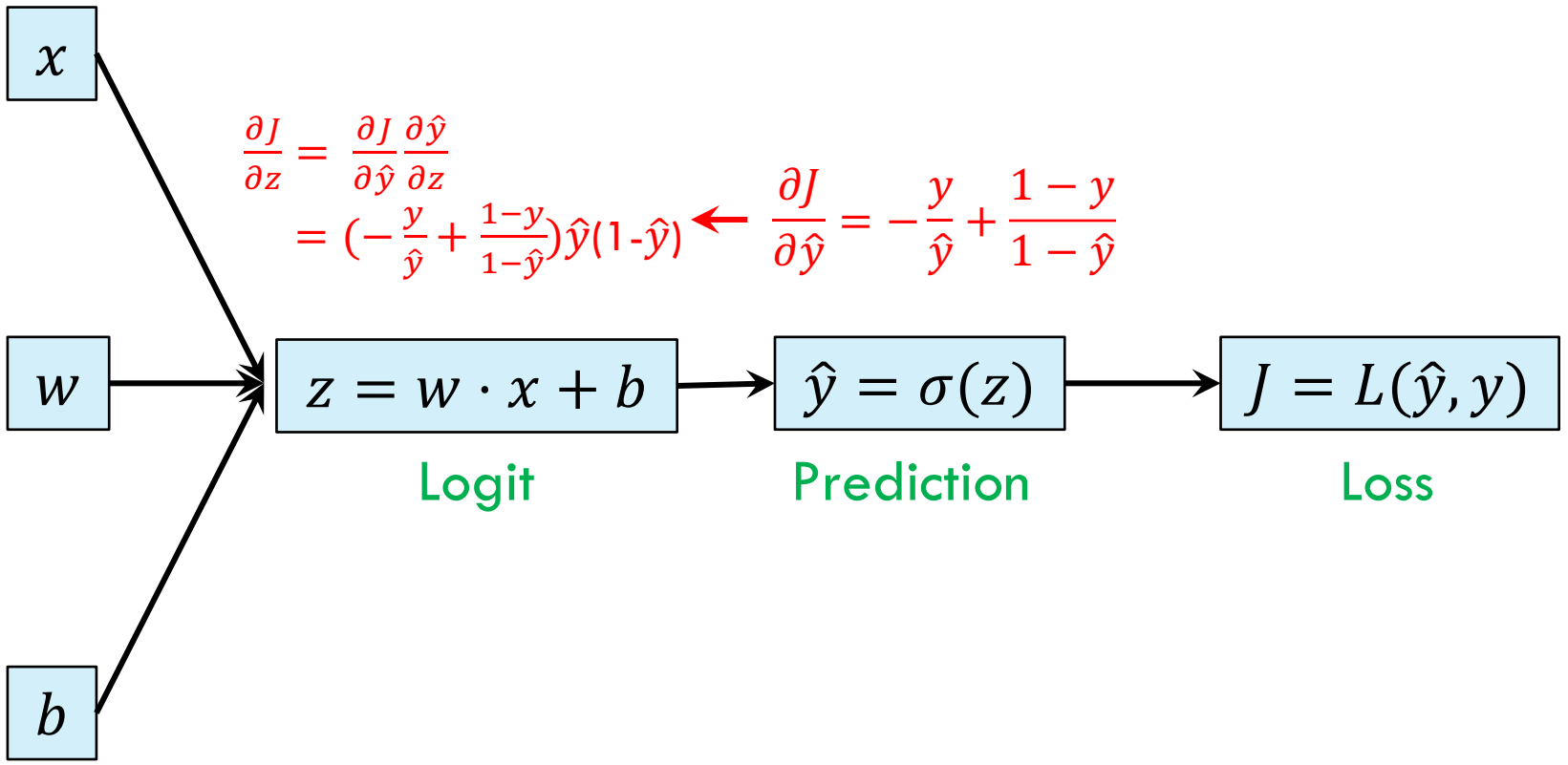


How to compute the derivatives $\frac{dq}{da}$, $\frac{dq}{db}$, etc.?

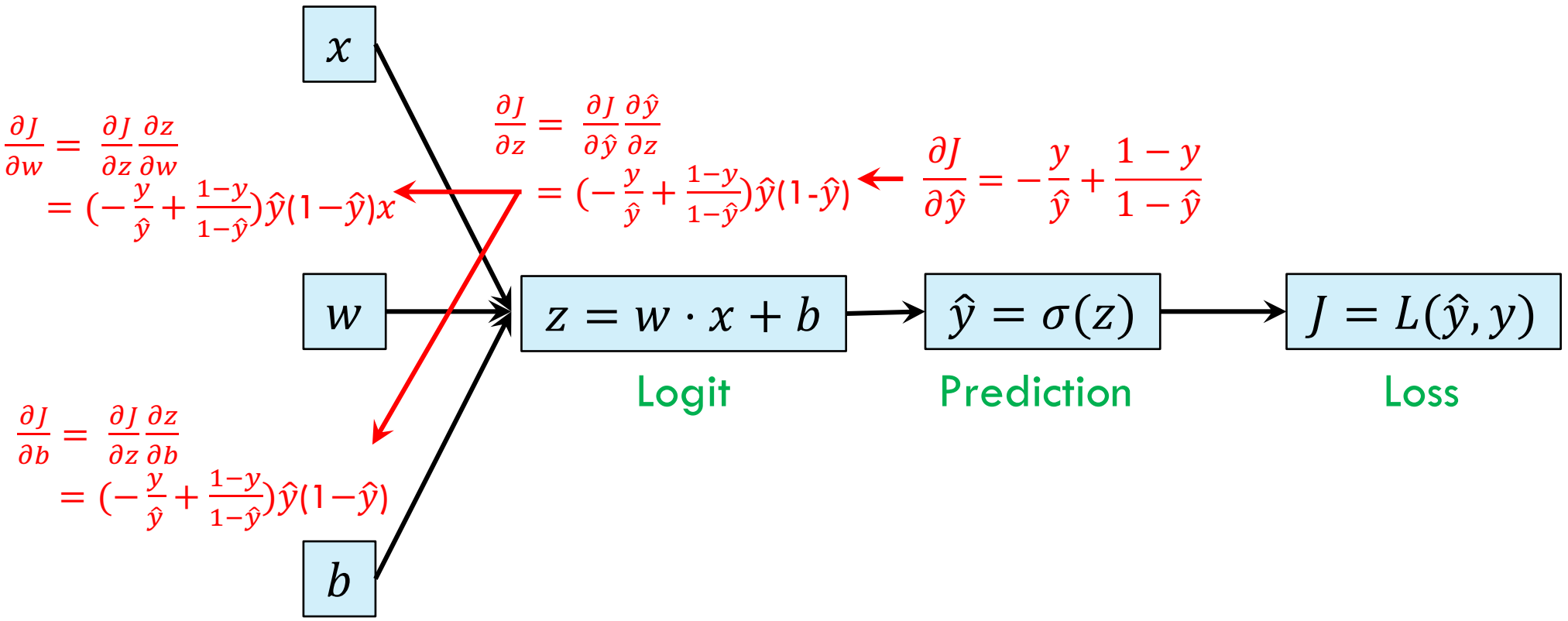
LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH



LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH



LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH

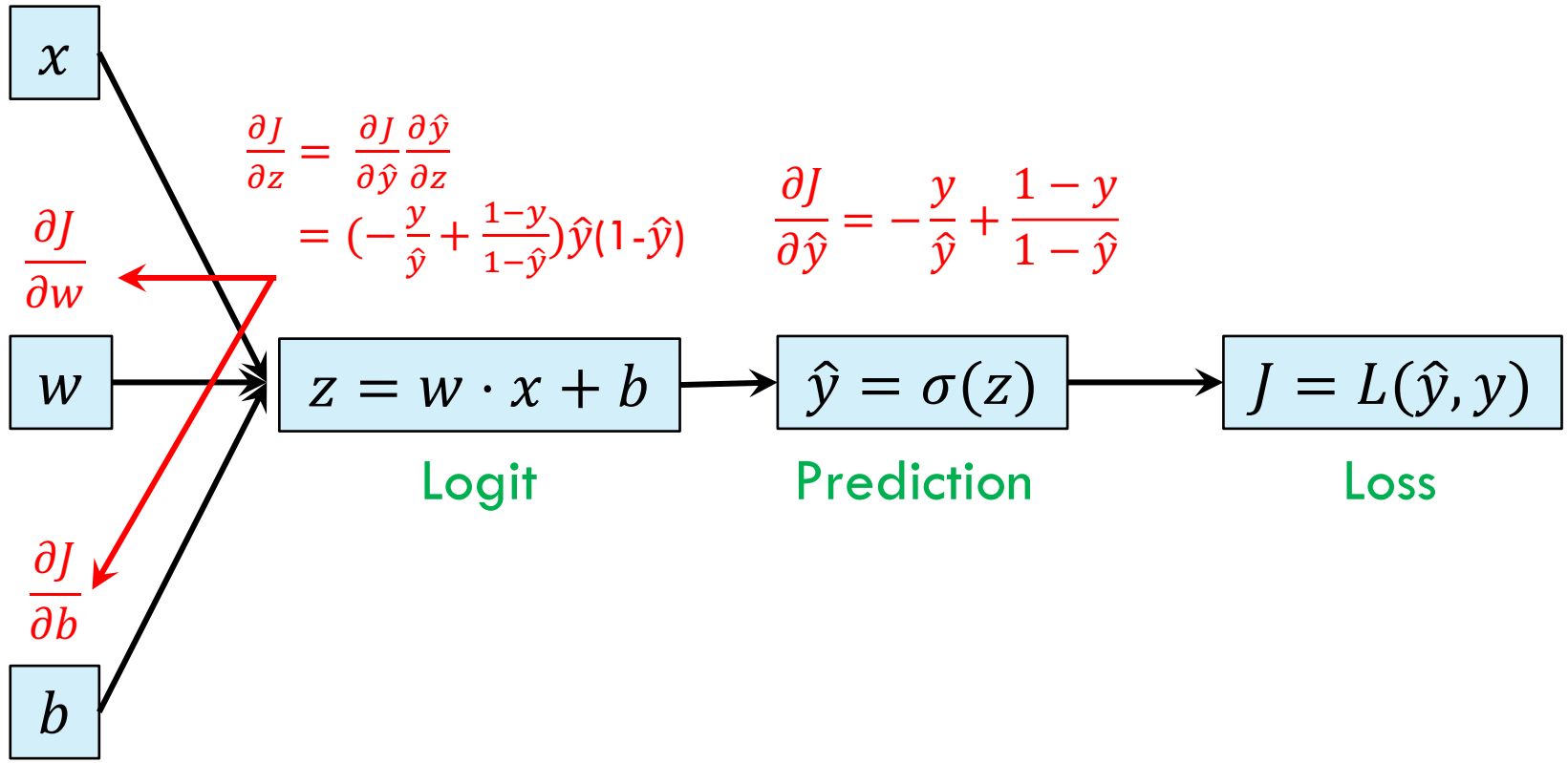


LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH

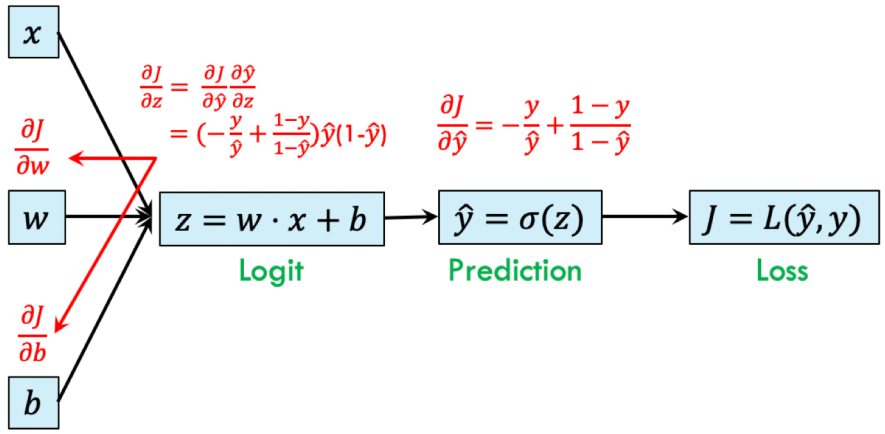
Gradient Descent Steps

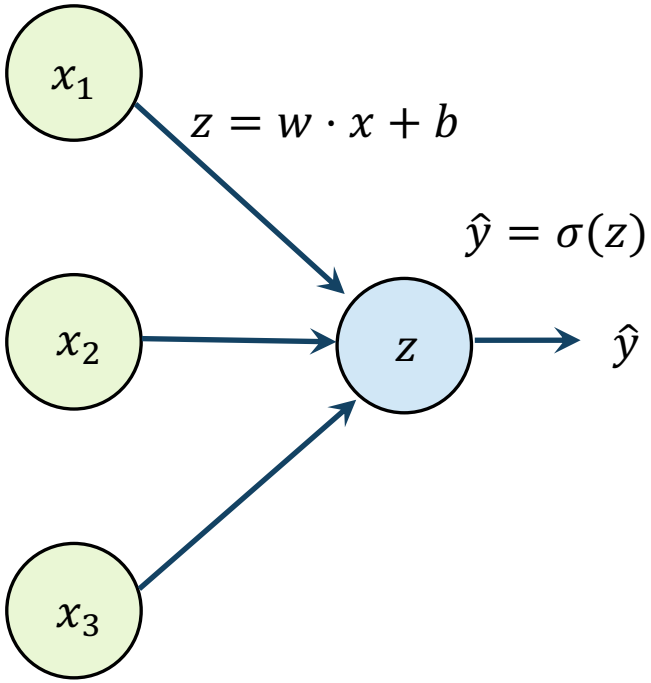
$$w \leftarrow w - \eta \frac{\delta J(w,b)}{\delta w}$$

$$b \leftarrow b - \eta \frac{\delta J(w,b)}{\delta b}$$



LOGISTIC REGRESSION AS A NEURAL NETWORK





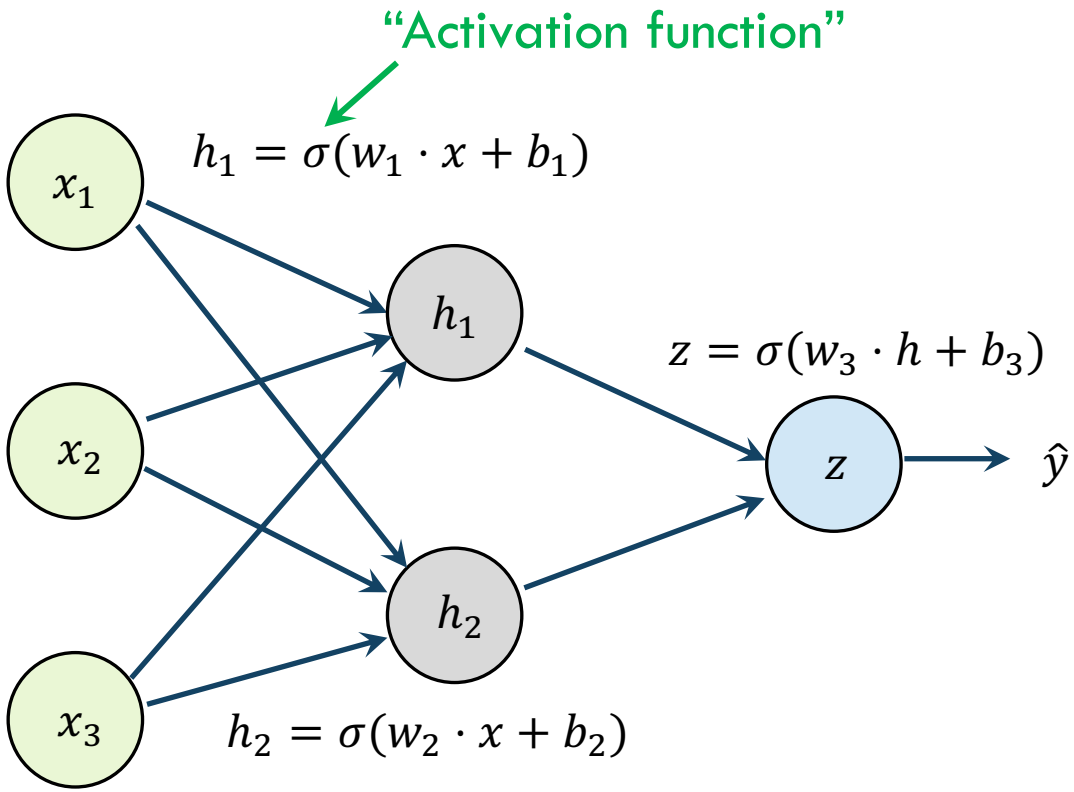
Input Layer

Output Layer

Layers of neurons: each layer computes a function of the previous layer

This is a 1-layer network (input layer is not counted)

DEEPER NEURAL NETWORKS



Each hidden unit ("neuron") has its own weights and biases

These parameters are used to compute a linear function of the previous layer

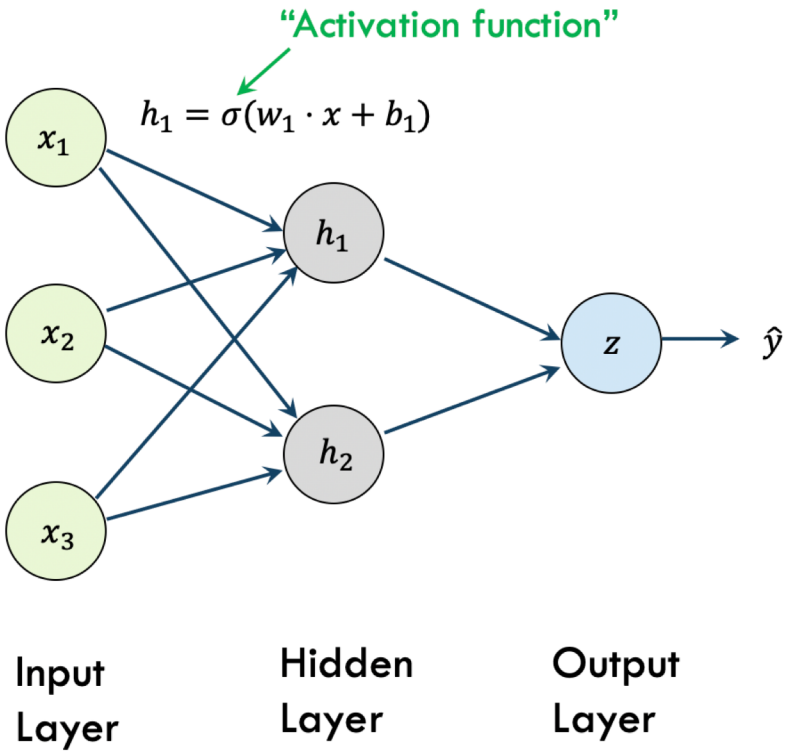
Optionally, they also apply a nonlinear **activation function,** similar to the sigmoid in logistic regression

The resulting models are much **more expressive** than logistic regression / linear models

EXAMPLE CODE FOR SIMPLE NEURAL NETWORK

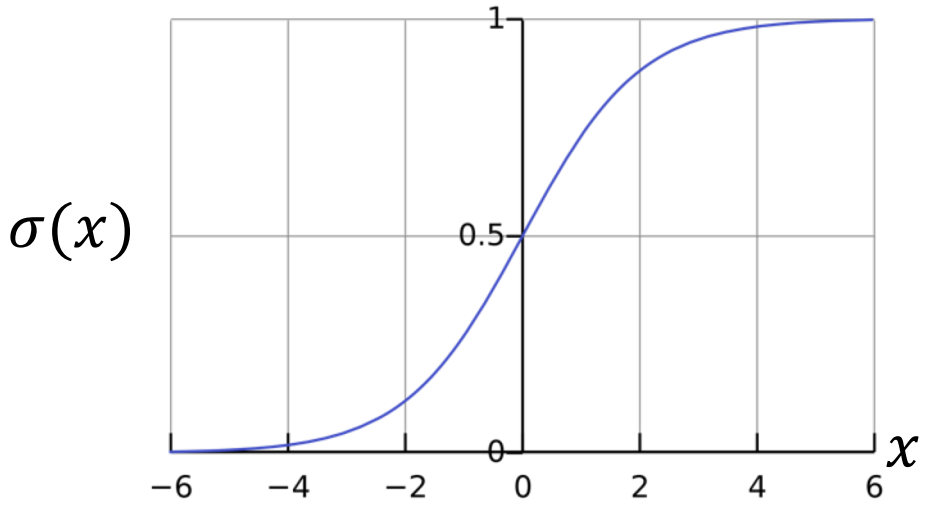
```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(2, input_dim=3))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy')
model.fit(X_train, y_train, epochs=100)

predictions = model.predict(X_test)
```

Output dimension



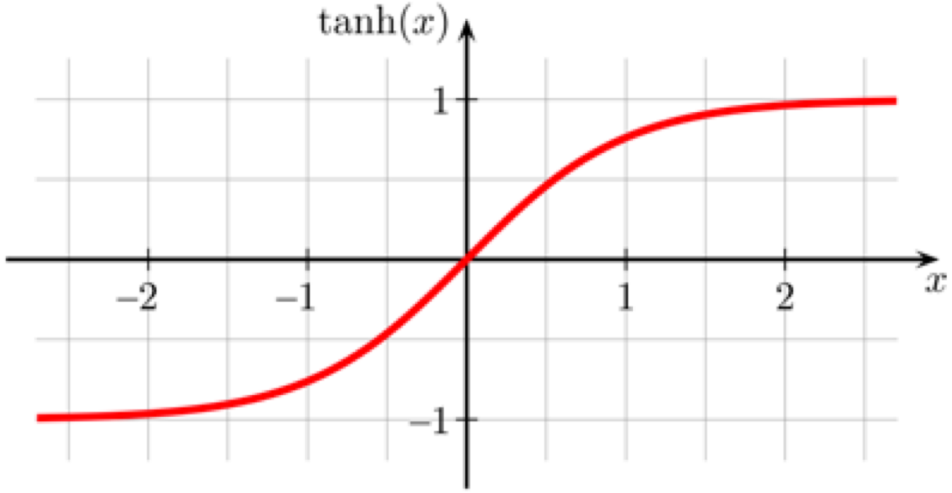
ACTIVATION FUNCTIONS

Sigmoid



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

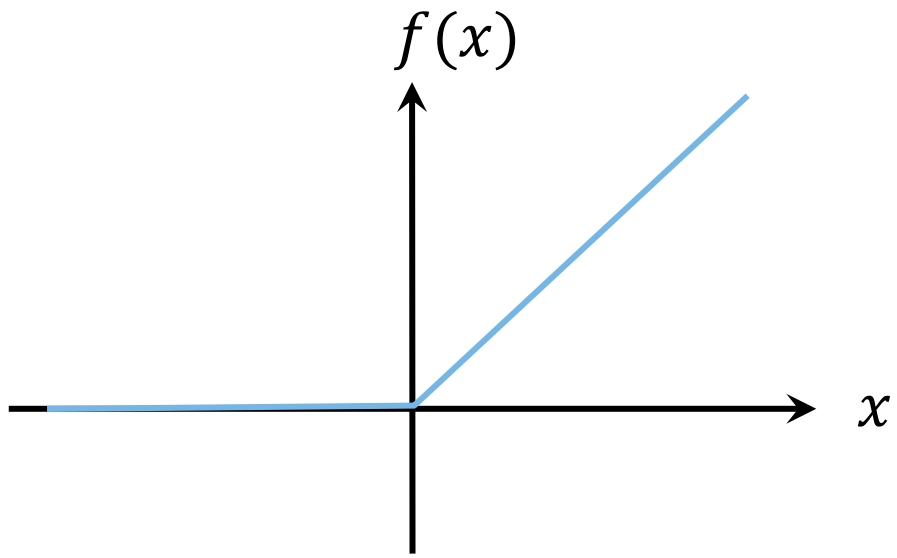
Tanh



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

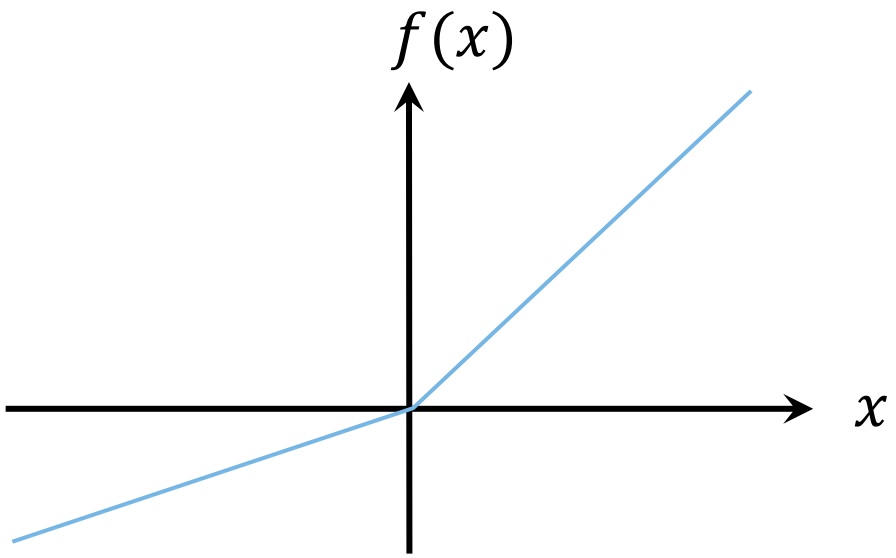
ACTIVATION FUNCTIONS

Rectified Linear Unit (ReLU)



$$f(x) = \max(0, x)$$

Parametric ReLU



$$f(x) = \max(ax, x)$$

MANY TRICKS

Initialization

- Random, Weight initialization, Xavier initialization, …

Adaptive learning rates

- Decay, Momentum, RMSProp, Adam, …

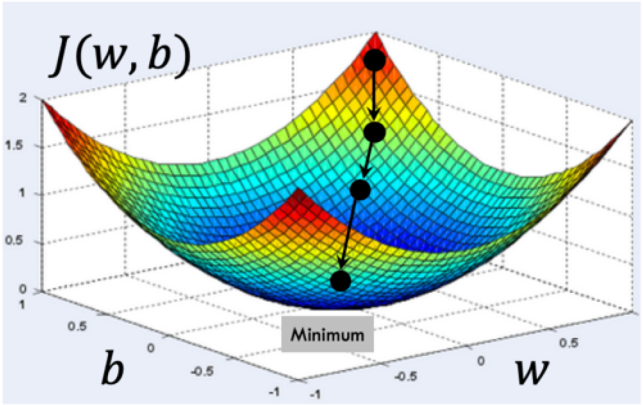
Batch normalization

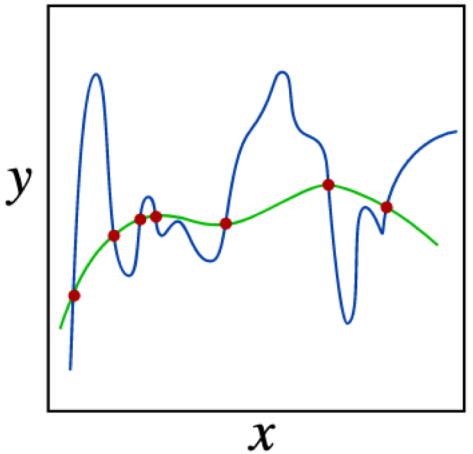
Regularization

- Early stopping, dropout, L1 / L2 regularization, data augmentation, …

Speed up learning

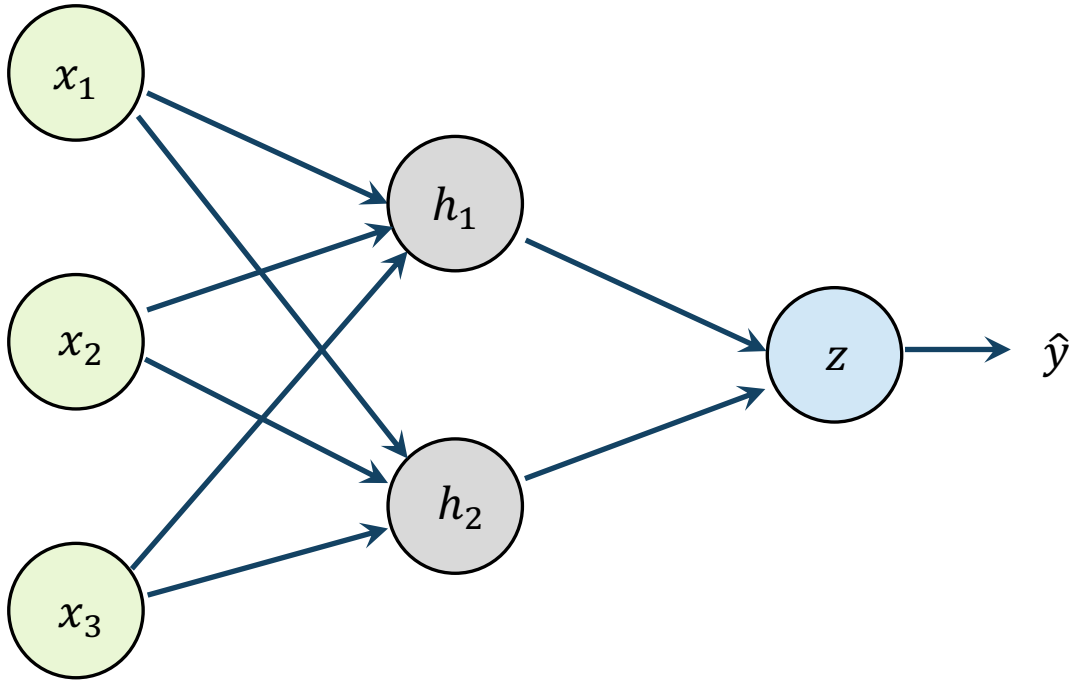
Reduce overfitting





Bjorck N, Gomes CP, Selman B, Weinberger KQ. Understanding batch normalization. NeurIPS 2018

QUIZ: HOW MANY PARAMETERS?



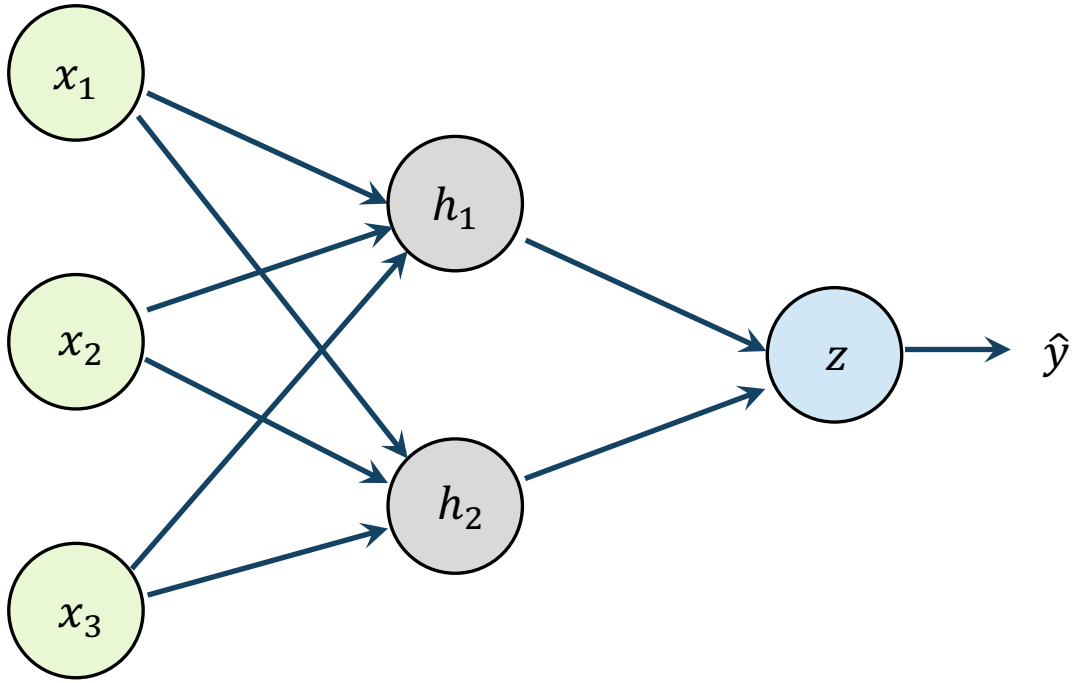
Input Layer

Hidden Layer

Output Layer

Q: In this neural network, how many weight parameters (i.e. not bias parameters) in total?

QUIZ: HOW MANY PARAMETERS?



Input Layer

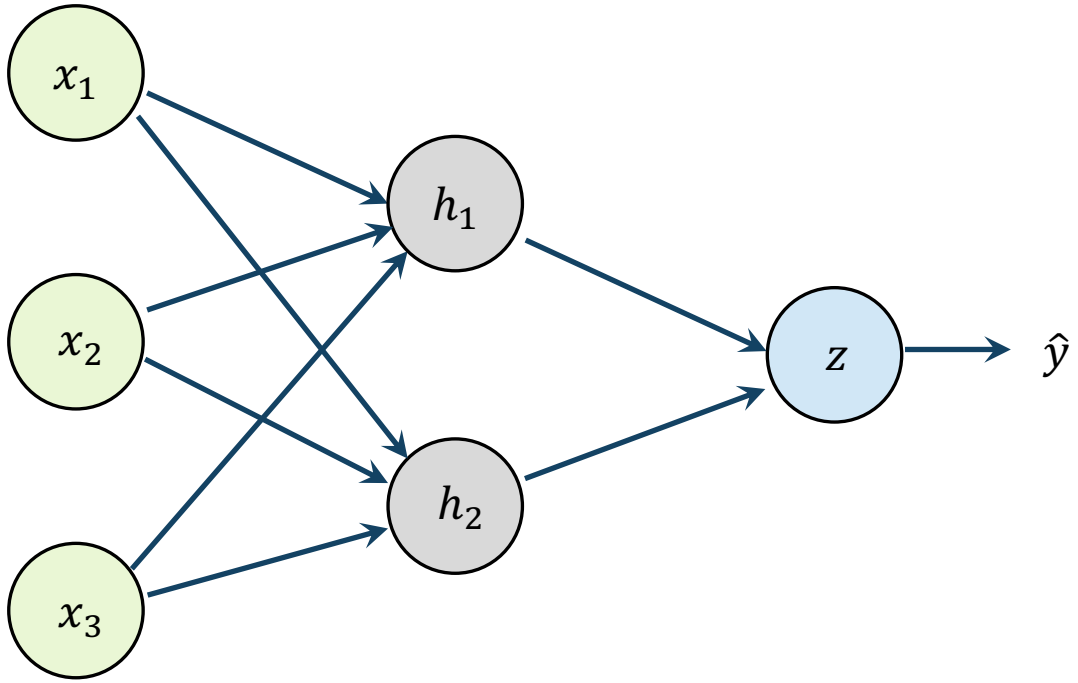
Hidden Layer

Output Layer

Q: In this neural network, how many weight parameters (i.e. not bias parameters) in total?

A: 8 (3×2=6 between the input and hidden layer, and 2 between the hidden and output layer)

QUIZ: HOW MANY PARAMETERS?



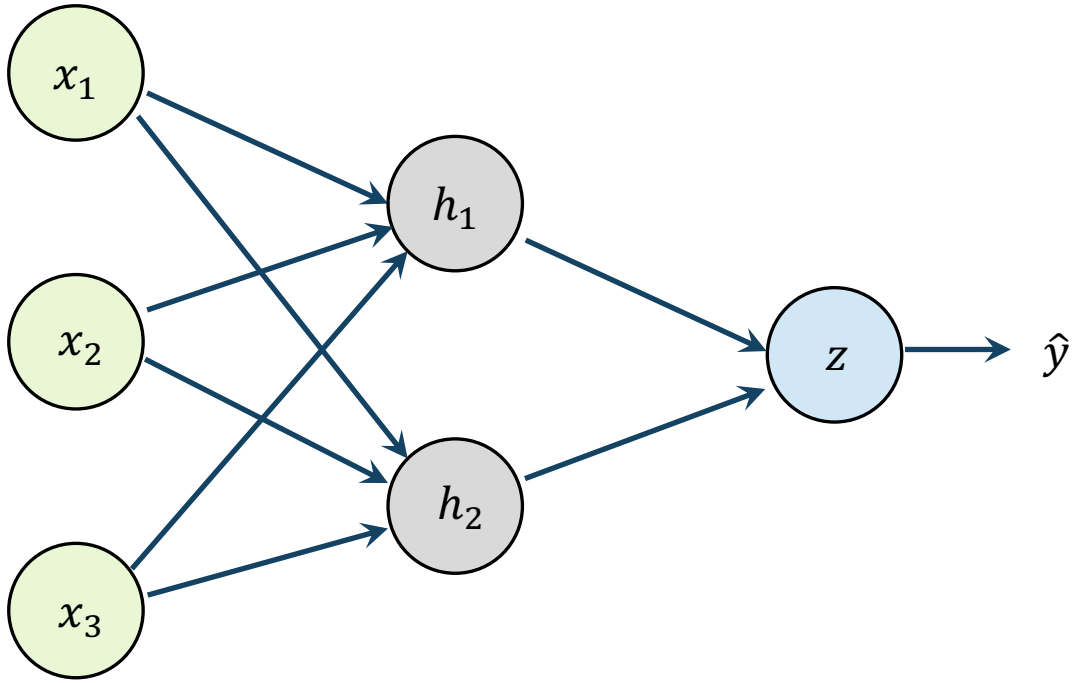
Input Layer

Hidden Layer

Output Layer

Q: In this neural network, how many bias parameters in total are there?

QUIZ: HOW MANY PARAMETERS?



Input Layer

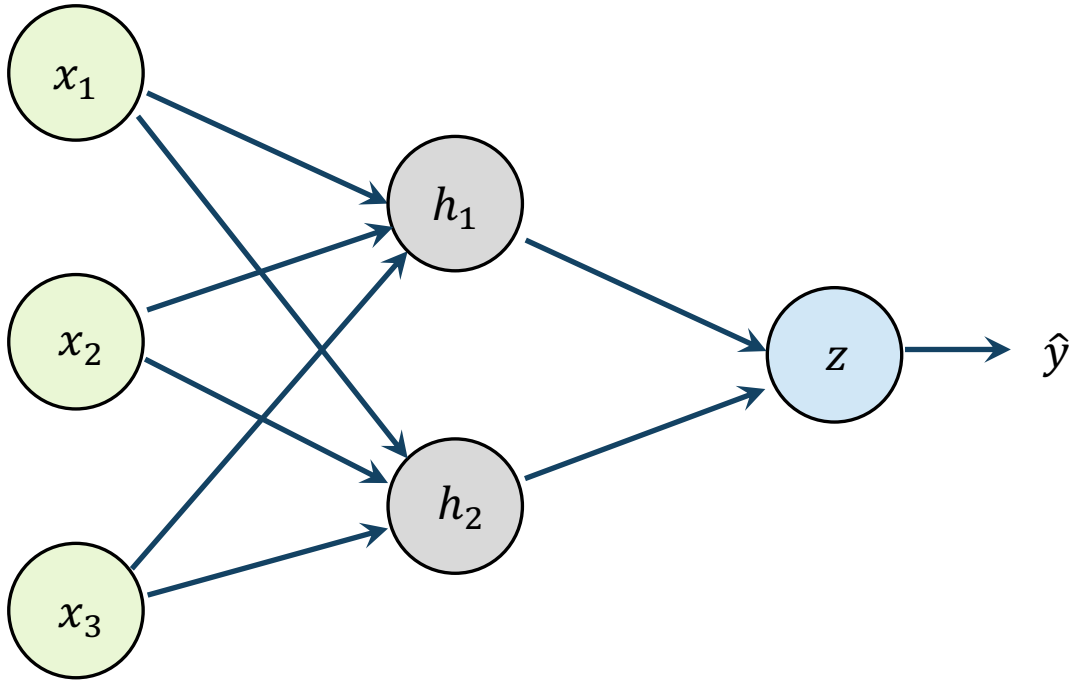
Hidden Layer

Output Layer

Q: In this neural network, how many bias parameters in total are there?

A: 3 (once for each hidden and output layer neuron)

HOW MANY PARAMETERS?



Input Layer

Hidden Layer

Output Layer

Fully connected layers:

- Number of weights: # inputs × #outputs
- Number of bias terms: #outputs

Memory usage:

- 4 bytes × number of parameters
- Should fit in your GPU memory