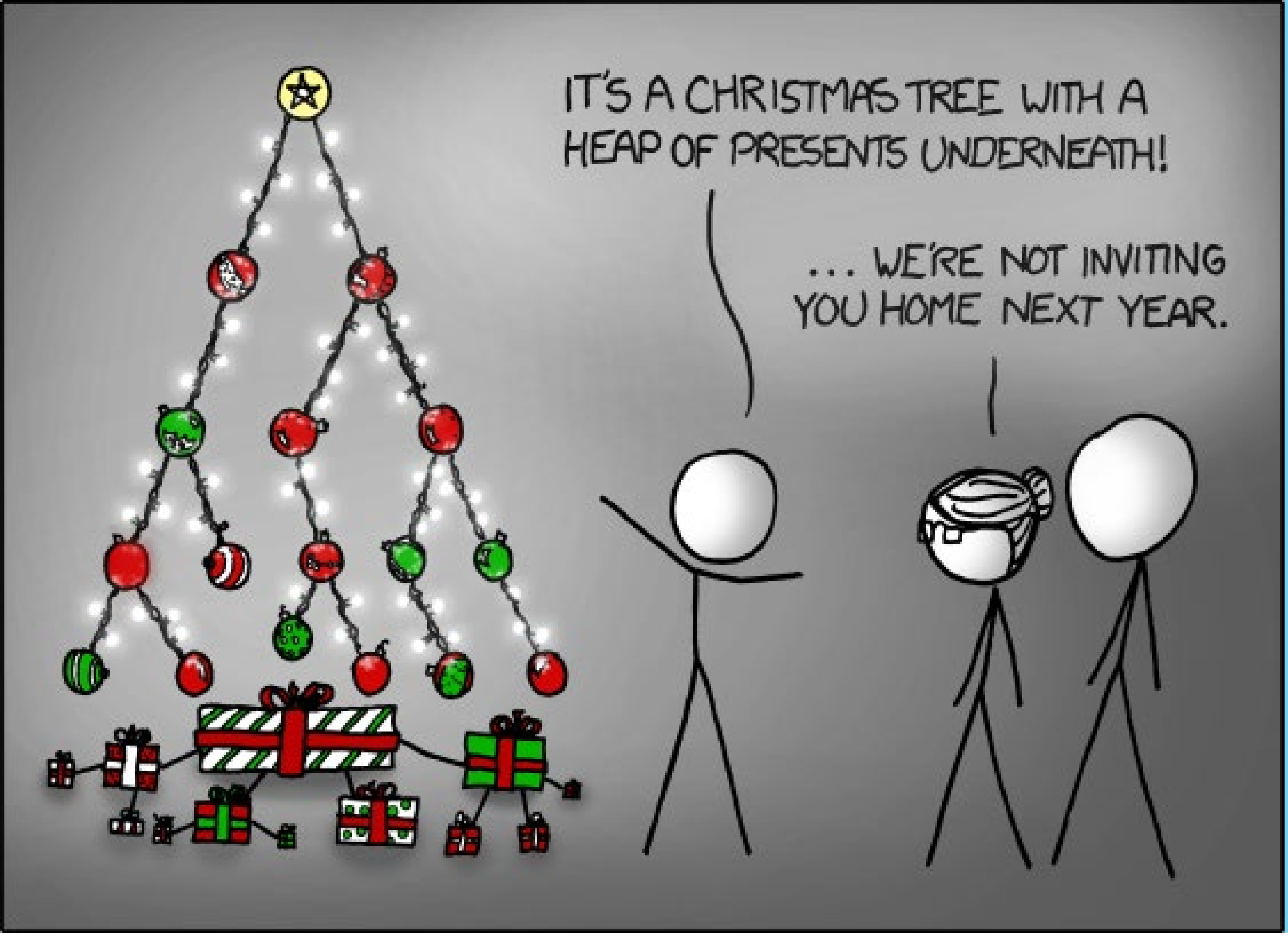
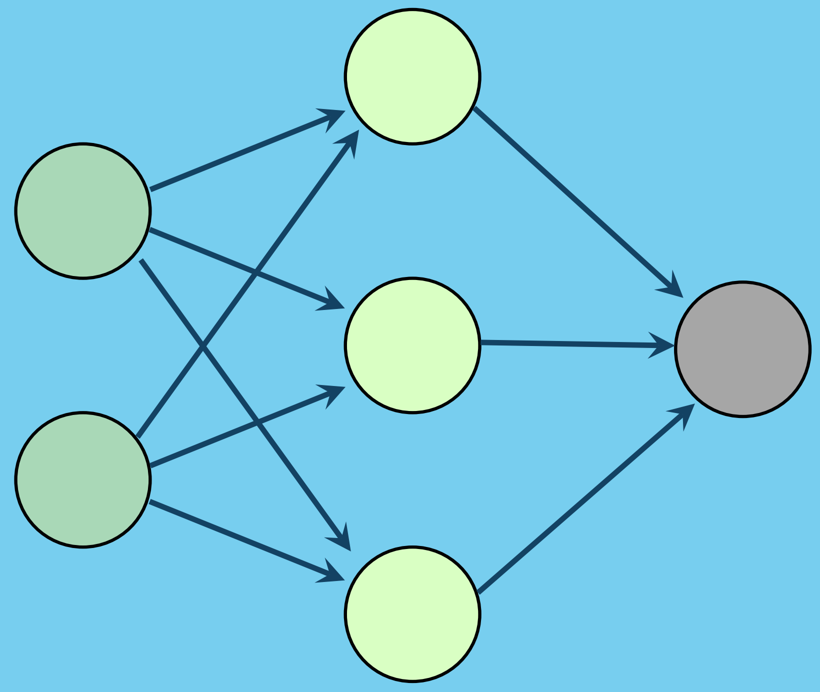




CS5344: CLASSIFICATION II: TREES ENSEMBLES, LOGISTICS REGRESSION AND DEEP LEARNING

Anthony Tung

School of Computing

National University of Singapore

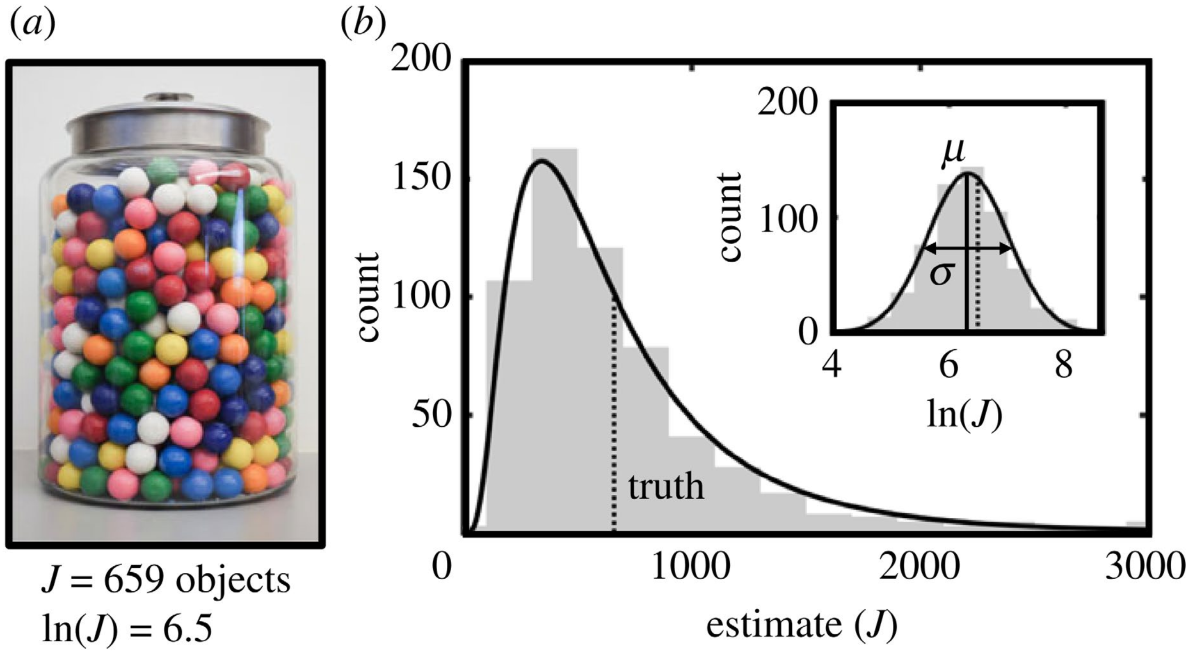
Slide Credit: Bryan Hooi, Wang Wei, Ng See Kiong, Wynne Hsu, Chris von der Weth

CLASSIFICATION OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. **Trees and Ensembles**
   a) Decision Trees
   b) Bagging and Random Forests
   c) Boosting and Gradient Boosting Machines
5. Logistic Regression
6. Deep Learning

BAGGING: MOTIVATION ('WISDOM OF THE CROWDS')

Wisdom of the crowds refers to the idea that large groups of people are collectively smarter than individual experts.



Kao et al., "Counteracting estimation bias and social influence to improve the wisdom of crowds"

BOOTSTRAP SAMPLING

Goal: generate randomly sampled "versions" of the dataset

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

Original Data

Resample same-sized dataset, **with replacement**

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

A Bootstrap Sample

BOOTSTRAP AGGREGATION ("BAGGING")

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

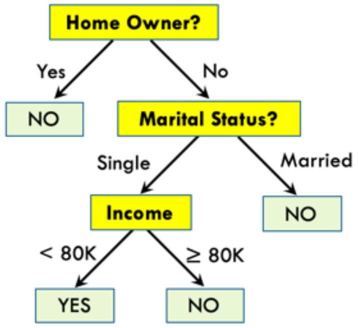
Original Data

Resampling

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

Bootstrap Sample 1

Training



Model 1

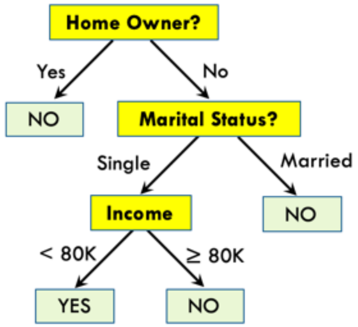
• • •

• • •

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

Bootstrap Sample N

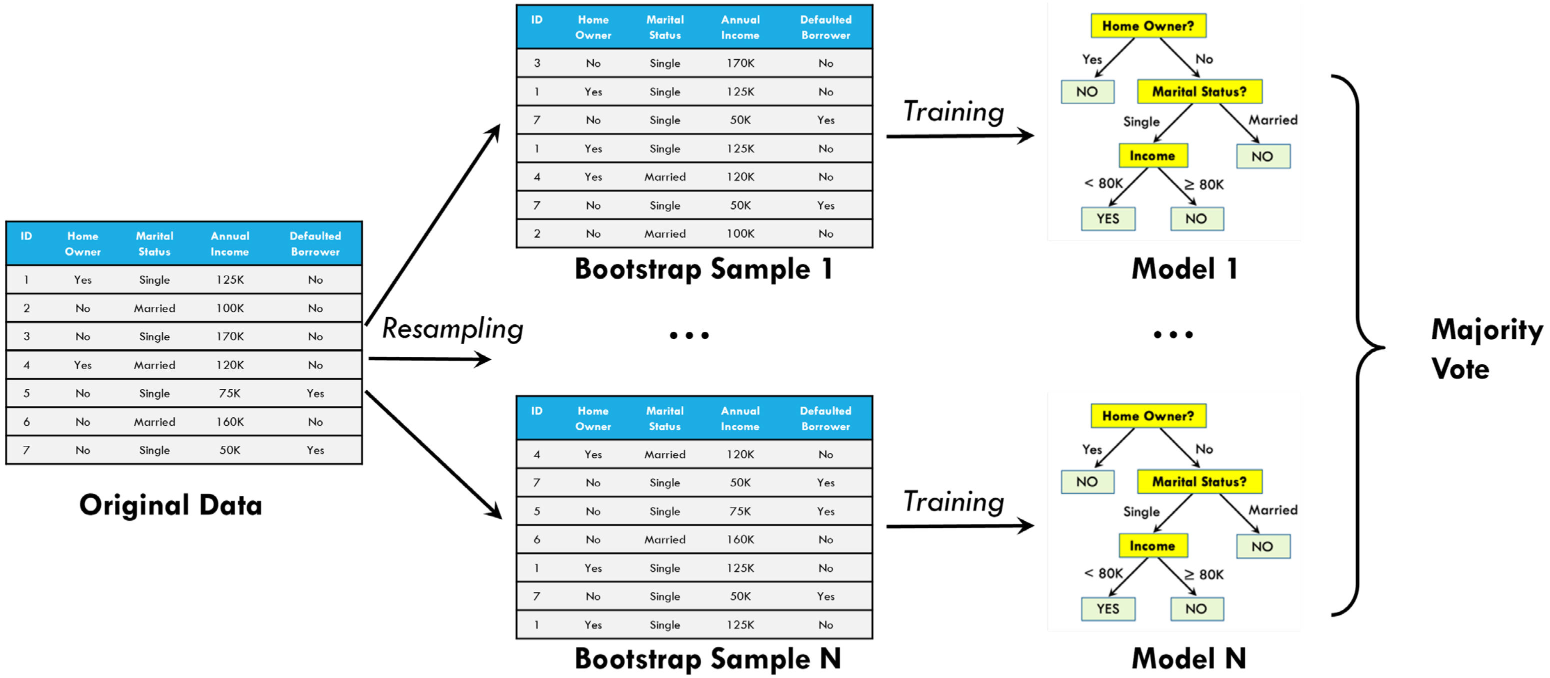
Training



Model N

Majority Vote

RANDOM FORESTS: BOOTSTRAP RESAMPLING ("ROW SAMPLING")

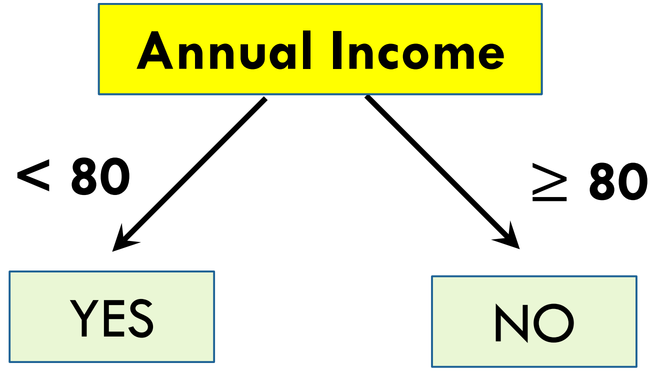


Bootstrap resampling: Random forests train an ensemble of decision trees on N bootstrap samples. (Sometimes, we take bootstrap samples of size smaller than the original dataset size, e.g., based on a user-specified parameter).

RANDOM FORESTS: FEATURE SAMPLING ("COLUMN SAMPLING")

d

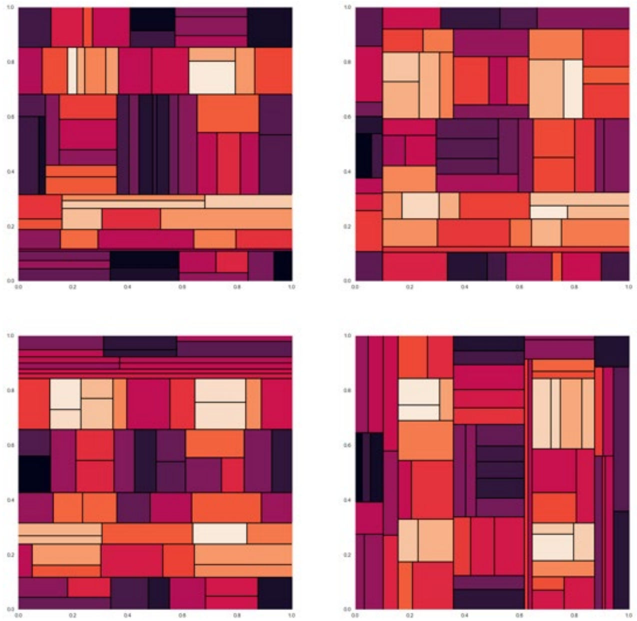
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	170K	No
4	Yes	Married	120K	No
5	No	Single	75K	Yes
6	No	Married	160K	No
7	No	Single	50K	Yes



Feature sampling: During each split, instead of considering **all** d features, we only consider a **random subset** of features, usually of size $\sqrt{d}$.

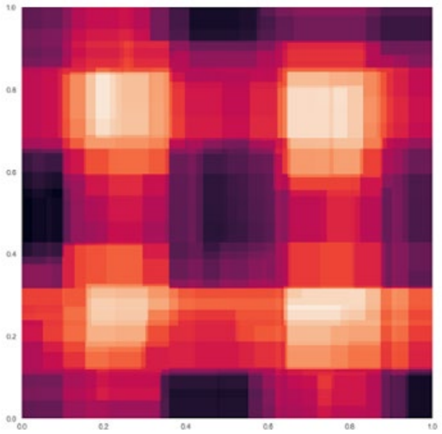
WHY DO RANDOM FORESTS WORK WELL?

Individual decision tree predictions



Averaging

Random Forest predictions



Variance Reduction: individual trees can overfit and give highly variable output. But when averaging them, the predictions are smoother and perform better on test data.

Randomization is important in making the decision trees **decorrelated.**

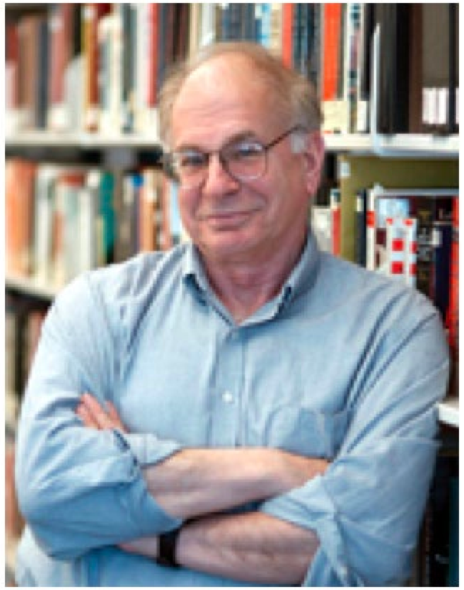
Credit: Tyler White, http://structuringtheunstructured.blogspot.com/2017/11/coloring-with-random-forests.html

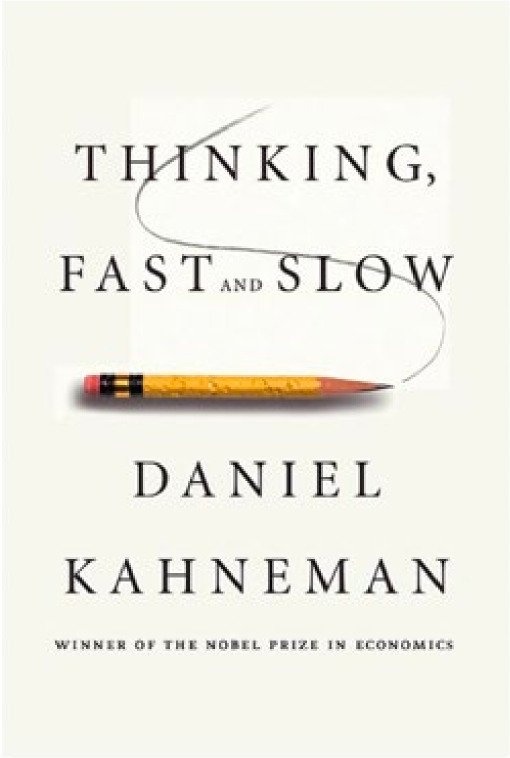
IMPORTANCE OF DECORRELATION

“To derive the most useful information from multiple sources of evidence, you should always try to make these sources independent of each other.”

“A simple rule can help: before an issue is discussed, all members of the committee should be asked to write a very brief summary of their position. This procedure makes good use of the value of the **diversity of knowledge and opinion in the group**. The standard practice of open discussion gives too much weight to the opinions of those who speak early and assertively, causing others to line up behind them.”

Daniel Kahnemann, *Thinking Fast and Slow*





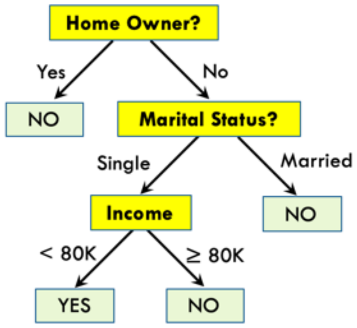
PROS AND CONS (VS. DECISION TREES)

Pros

- *Variance Reduction:* Ensembling many decision trees leads to more stable and accurate predictions
- Accuracy is fairly close to state of the art
- Parallelizable
- Not much tuning required

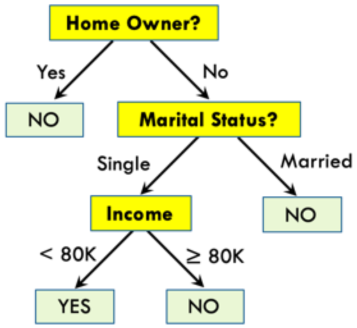
Cons

- Less interpretable than decision trees
- Slower than decision trees



Model 1

…



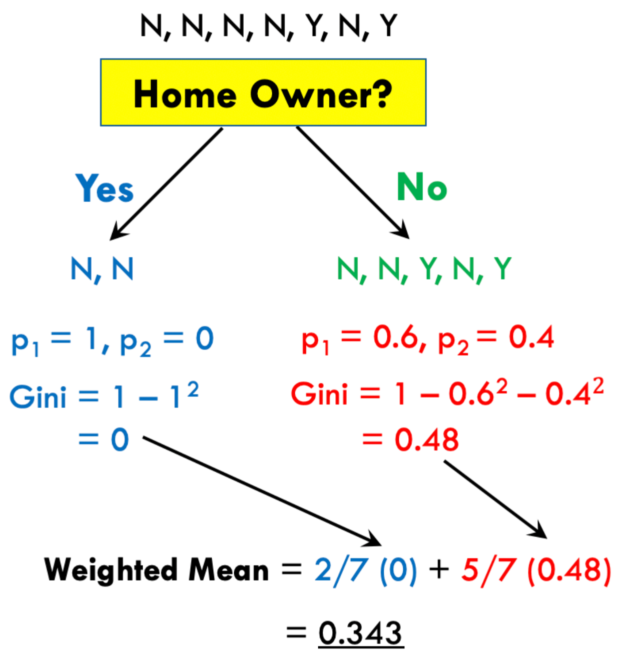
Model N

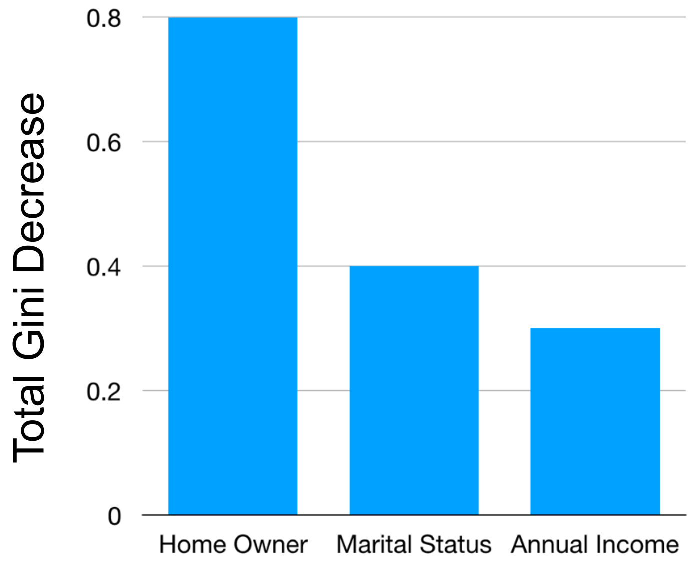
VARIABLE IMPORTANCE PLOTS

The importance of each variable is measured by the **total reduction in Gini index** brought about by that feature.

More important variables result in greater decrease in Gini index on average.

Computing variable importance is useful for **feature selection.**





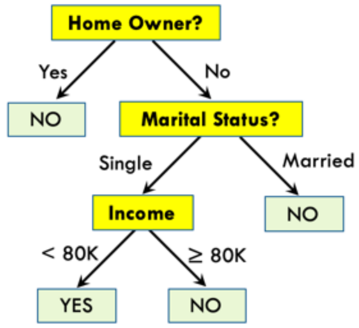
QUIZ: RANDOM FOREST HYPERPARAMETERS

Q: Which of the following tends to reduce overfitting?

1. Increasing number of trees

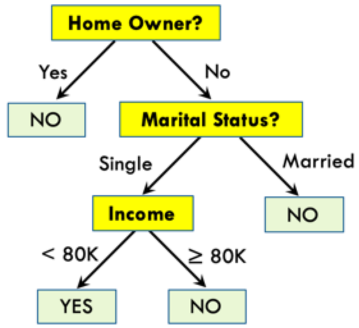
2. Increasing depth of trees

3. Decreasing the number of features considered when selecting each split



Model 1

…

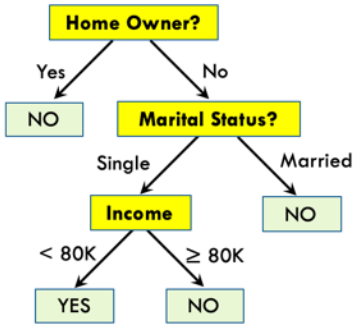


Model N

QUIZ: RANDOM FOREST HYPERPARAMETERS

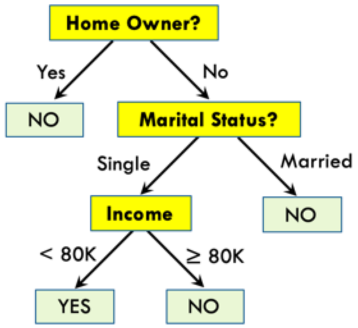
Q: Which of the following tends to reduce overfitting?

1. Increasing number of trees
2. Increasing depth of trees
3. Decreasing the number of features considered when selecting each split



Model 1

...



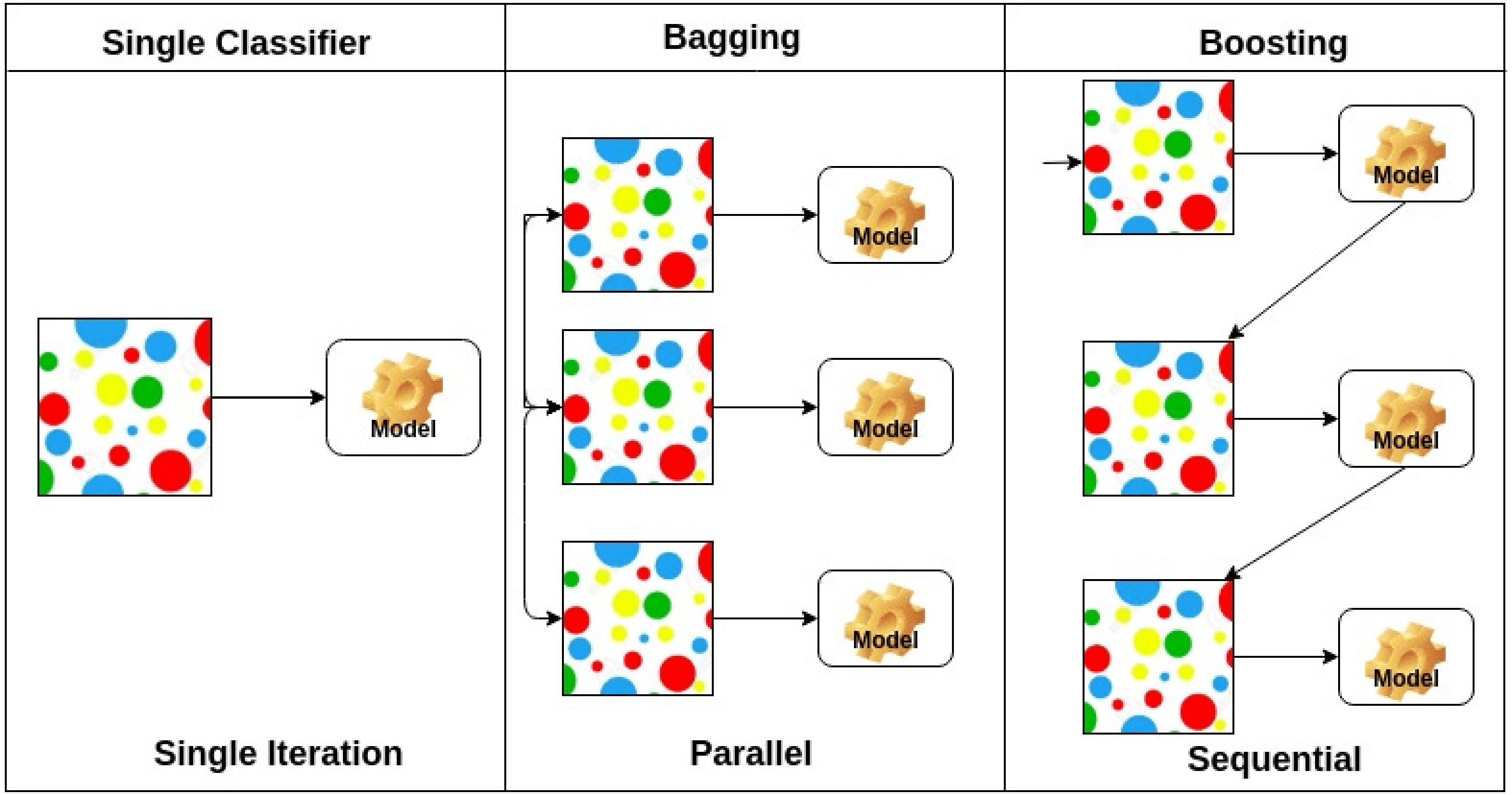
Model N

CLASSIFICATION OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. **Trees and Ensembles**
   a) Decision Trees
   b) Bagging and Random Forests
   c) Boosting and Gradient Boosting Machines
5. Logistic Regression
6. Deep Learning

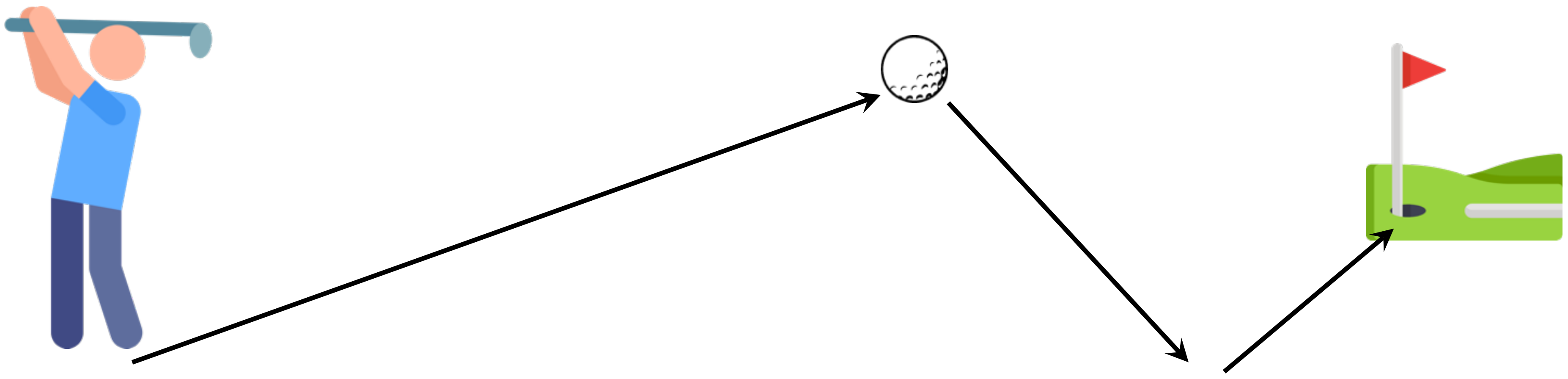
BAGGING VS BOOSTING

Bagging operates on bootstrap samples **independently**, but boosting is **sequential**: each new classifier “corrects for the mistakes” of the previous classifiers.



https://www.datacamp.com/community/tutorials/adaboost-classifier-python

BOOSTING: ITERATIVE PROCESS



Each new classifier is trained to “correct for the mistakes” of the previous set of classifiers, gradually getting closer to the ideal classifier

GRADIENT BOOSTED DECISION TREES

Current Fitted Model

ID	Home Owner	Marital Status	Annual Income	y
1	Yes	Single	125K	0
2	No	Married	100K	0.5
3	No	Single	170K	0.3
4	Yes	Married	120K	1
5	No	Single	75K	0.9
6	No	Married	160K	0.4
7	No	Single	50K	0.3

Original Data (X)

r_0
0
0.5
0.3
1
0.9
0.4
0.3

Residual (Initial)

1. Initial residuals are the original response data.

GRADIENT BOOSTED DECISION TREES

Current Fitted Model

ID	Home Owner	Marital Status	Annual Income	y
1	Yes	Single	125K	0
2	No	Married	100K	0.5
3	No	Single	170K	0.3
4	Yes	Married	120K	1
5	No	Single	75K	0.9
6	No	Married	160K	0.4
7	No	Single	50K	0.3

Original Data (X)

r_0
0
0.5
0.3
1
0.9
0.4
0.3

Residual (Initial)

Fit tree with response r_0 →

f_1
0.2
0.3
0.3
1
1
0.4
0.3

2. Fit a decision tree to predict residuals r_0.

GRADIENT BOOSTED DECISION TREES

Current Fitted Model

Learning rate

$\lambda = 0.5$

ID	Home Owner	Marital Status	Annual Income	y
1	Yes	Single	125K	0
2	No	Married	100K	0.5
3	No	Single	170K	0.3
4	Yes	Married	120K	1
5	No	Single	75K	0.9
6	No	Married	160K	0.4
7	No	Single	50K	0.3

Original Data (X)

r_0
0
0.5
0.3
1
0.9
0.4
0.3

Residual (Initial)

Fit tree with response r_0

f_1
0.2
0.3
0.3
1
1
0.4
0.3

λf_1
0.1
0.15
0.15
0.5
0.5
0.2
0.15

3. Compute 'damped predictions', i.e., λ times decision tree output.

GRADIENT BOOSTED DECISION TREES

Current Fitted Model

Learning rate → $\lambda = 0.5$

ID	Home Owner	Marital Status	Annual Income	y
1	Yes	Single	125K	0
2	No	Married	100K	0.5
3	No	Single	170K	0.3
4	Yes	Married	120K	1
5	No	Single	75K	0.9
6	No	Married	160K	0.4
7	No	Single	50K	0.3

Original Data (X)

r_0
0
0.5
0.3
1
0.9
0.4
0.3

Residual (Initial)

Fit tree with response r_0 →

f_1	λf_1	r_1
0.2	0.1	-0.1
0.3	0.15	0.35
0.3	0.15	0.15
1	0.5	0.5
1	0.5	0.4
0.4	0.2	0.2
0.3	0.15	0.15

subtract

Residual

4. Compute new residual: $r_1 = r_0 - \lambda f_1$.

GRADIENT BOOSTED DECISION TREES

Current Fitted Model

Learning rate → $\lambda = 0.5$  + λ 

ID	Home Owner	Marital Status	Annual Income	y
1	Yes	Single	125K	0
2	No	Married	100K	0.5
3	No	Single	170K	0.3
4	Yes	Married	120K	1
5	No	Single	75K	0.9
6	No	Married	160K	0.4
7	No	Single	50K	0.3

Original Data (X)

r_0	f_1	λf_1	r_1	f_2	λf_2	r_2
0	0.2	0.1	-0.1	-0.2	-0.1	0
0.5	0.3	0.15	0.35	0.6	0.3	0.05
0.3	0.3	0.15	0.15	0.2	0.1	0.05
1	1	0.5	0.5	0.6	0.3	0.2
0.9	1	0.5	0.4	0.4	0.2	0.2
0.4	0.4	0.2	0.2	0.2	0.1	0.1
0.3	0.3	0.15	0.15	0.2	0.1	0.05

Residual (Initial) → *Fit tree with response r_0* → f_1

Residual (r_1) → *Fit tree with response r_1* → f_2

subtract

Residual (r_2)

• • •

5. Repeat the same process (fitting decision tree, etc.) on the new residuals r_1.

GRADIENT BOOSTED DECISION TREES

Current Fitted Model

λ $= 0.5$

$+$ λ

$+$ ...

ID	Home Owner	Marital Status	Annual Income	y
1	Yes	Single	125K	0
2	No	Married	100K	0.5
3	No	Single	170K	0.3
4	Yes	Married	120K	1
5	No	Single	75K	0.9
6	No	Married	160K	0.4
7	No	Single	50K	0.3

Original Data (X)

r_0
0
0.5
0.3
1
0.9
0.4
0.3

Residual (Initial)

Fit tree with response r_0 →

f_1	λf_1	r_1
0.2	0.1	-0.1
0.3	0.15	0.35
0.3	0.15	0.15
1	0.5	0.5
1	0.5	0.4
0.4	0.2	0.2
0.3	0.15	0.15

Residual

Fit tree with response r_1 →

f_2	λf_2	r_2
-0.2	-0.1	0
0.6	0.3	0.05
0.2	0.1	0.05
0.6	0.3	0.2
0.4	0.2	0.2
0.2	0.1	0.1
0.2	0.1	0.05

Residual

...

6. After fitting B trees, the final prediction is the sum of damped trees.

SUMMARY: GRADIENT TREES BOOSTING (REGRESSION CASE)

Model Predictions

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

For b = 1, …, 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	[illegible]	0.3	0.3	0.15	0.15	0.2	0.1	0.05

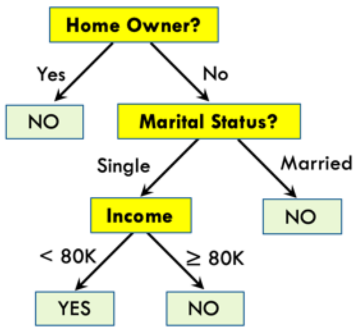
Original Data — Residual (Initial) — *Fit tree with response r_0* → Residual — *Fit tree with response r_1* → Residual …

QUIZ: GRADIENT BOOSTING HYPERPARAMETERS

Q: While tuning a gradient boosting tree model, you decide to decrease the value of λ. How does this affect the number of trees you should use?

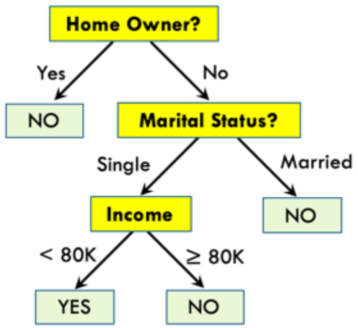
1. Increase

2. Decrease



Model 1

…

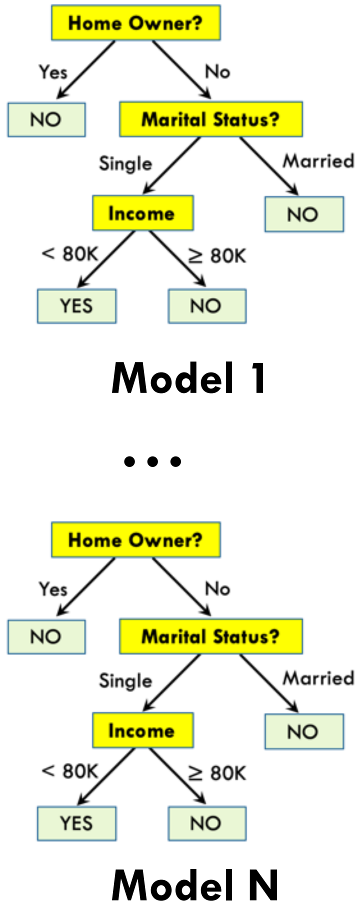


Model N

QUIZ: GRADIENT BOOSTING HYPERPARAMETERS

Q: While tuning a gradient boosting tree model, you decide to decrease the value of λ. How does this affect the number of trees you should use?

1. Increase ✓
2. Decrease



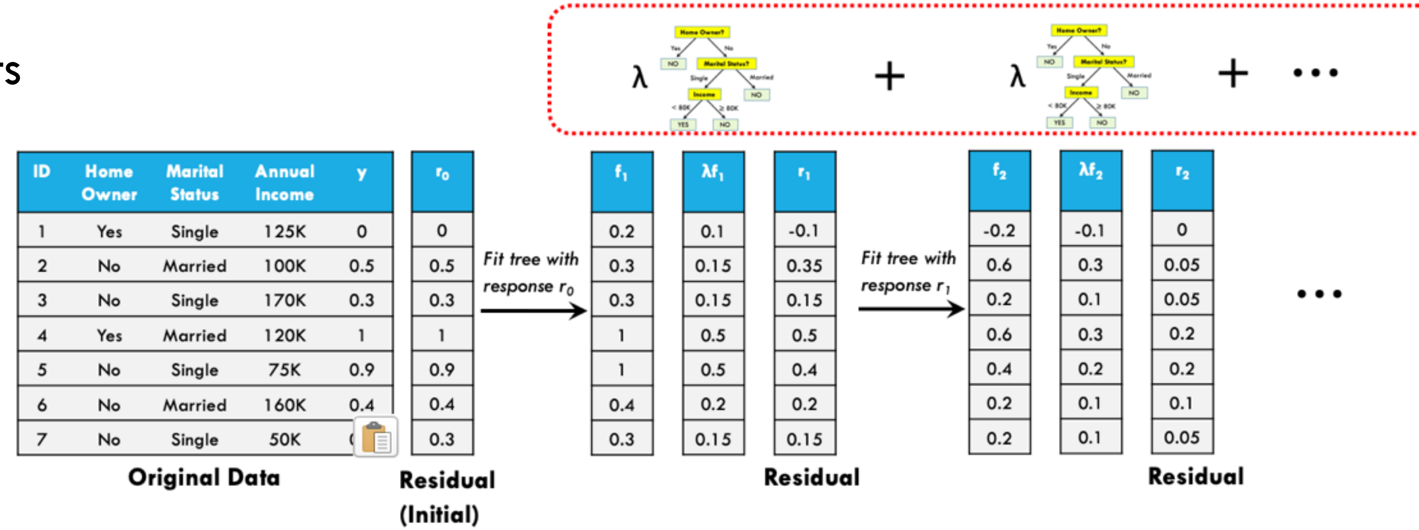
PROS AND CONS

Pros

- *Variance Reduction:* Ensembling many decision trees leads to more accurate predictions
- Generally, more accurate than random forests

Cons

- Less interpretable than decision trees
- Slower than decision trees
- Significant tuning required[1] (for tree depth, learning rate, number of trees)



[1]**General tuning guidelines:** tree depth is usually between 1 to 10, and it is important for controlling overfitting. Tuning it by selecting from {2, 4, 6, 8, 10} based on cross validation is generally fine. Learning rate and no. of trees have to be tuned together: learning rate is usually around 0.01 to 0.1. Smaller learning rate requires more trees (and thus longer training time). If speed is not an issue, a common approach is to set a low learning rate (e.g. 0.01), and train it while selecting the number of trees using early stopping (most gradient boosting tree packages have early stopping already implemented). If this is too slow, it may be advisable to start with a higher learning rate (e.g. 0.1) and later assess whether lowering the learning rate provides improvement in accuracy. See https://machinelearningmastery.com/configure-gradient-boosting-algorithm/ (and other guides that exist online) for more details (including other tuning parameters)

CLASSIFICATION OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. Logistic Regression
6. Deep Learning

CLASSIFICATION OVERVIEW

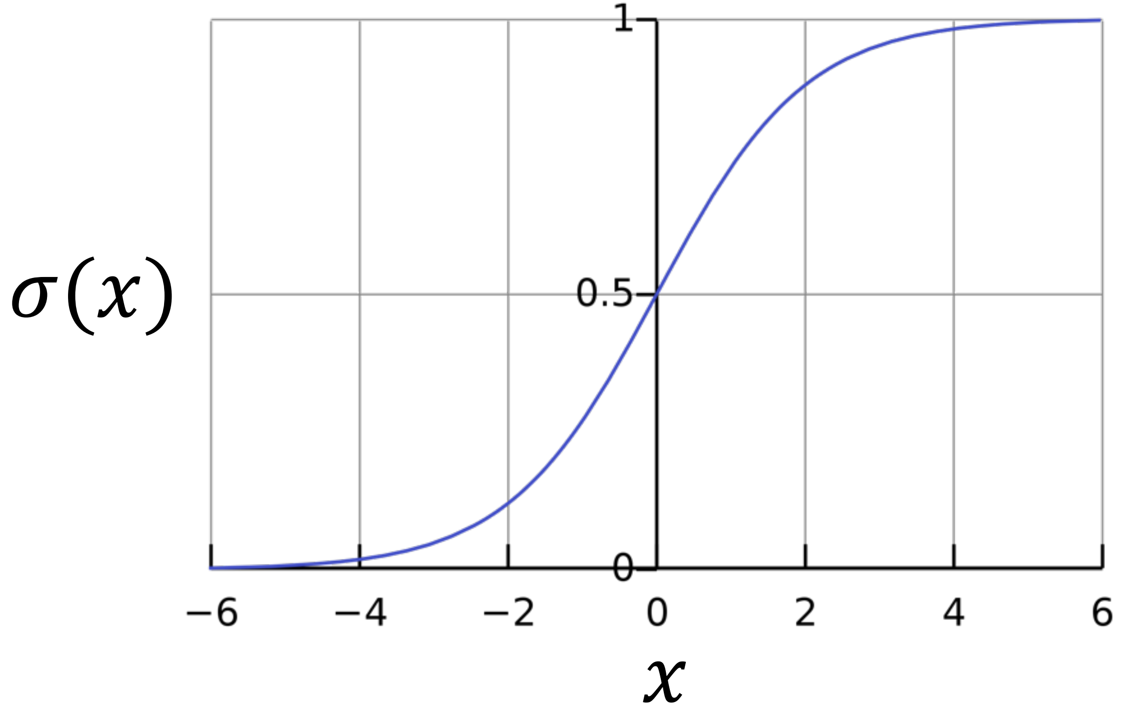
1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. Logistic Regression
6. Deep Learning

SIGMOID FUNCTION

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

It is defined as:

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

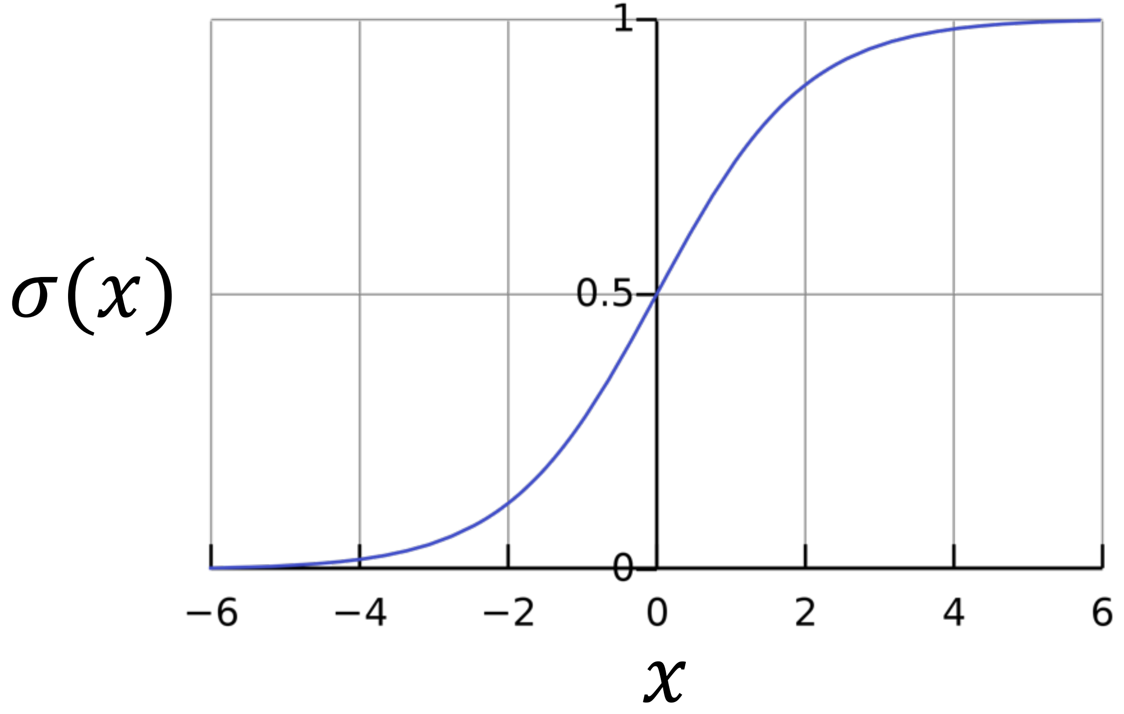


SIGMOID FUNCTION

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

It is defined as:

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



Check:

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

SIGMOID FUNCTION

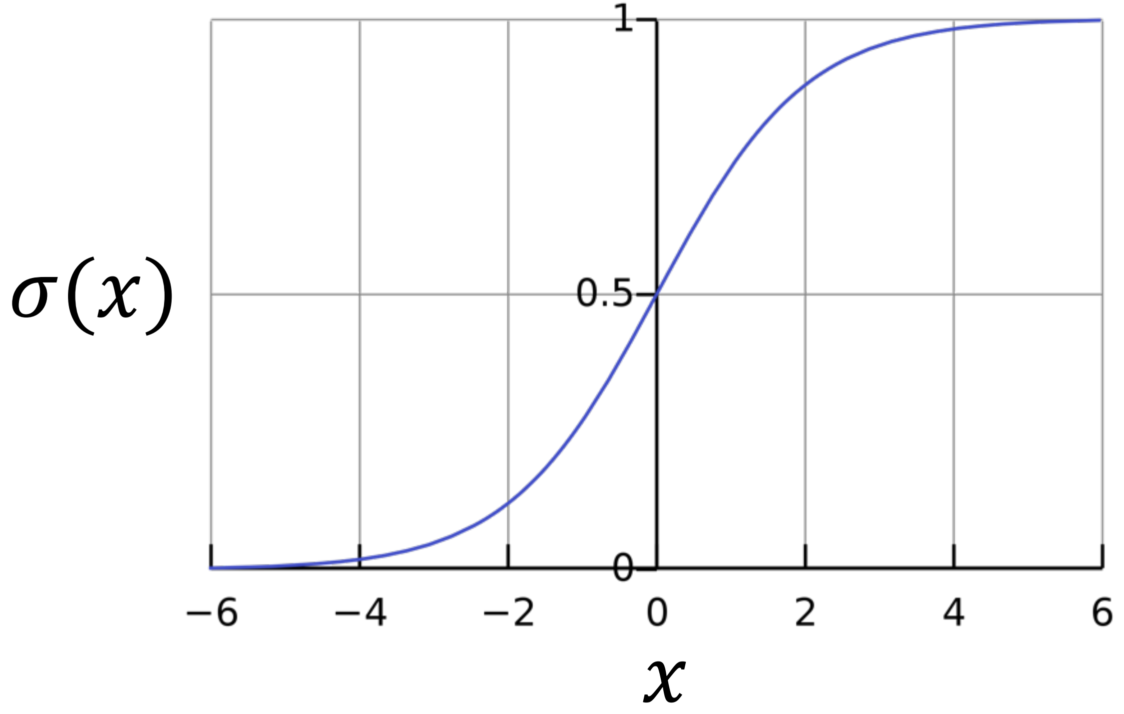
The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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

Check:

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



SIGMOID FUNCTION

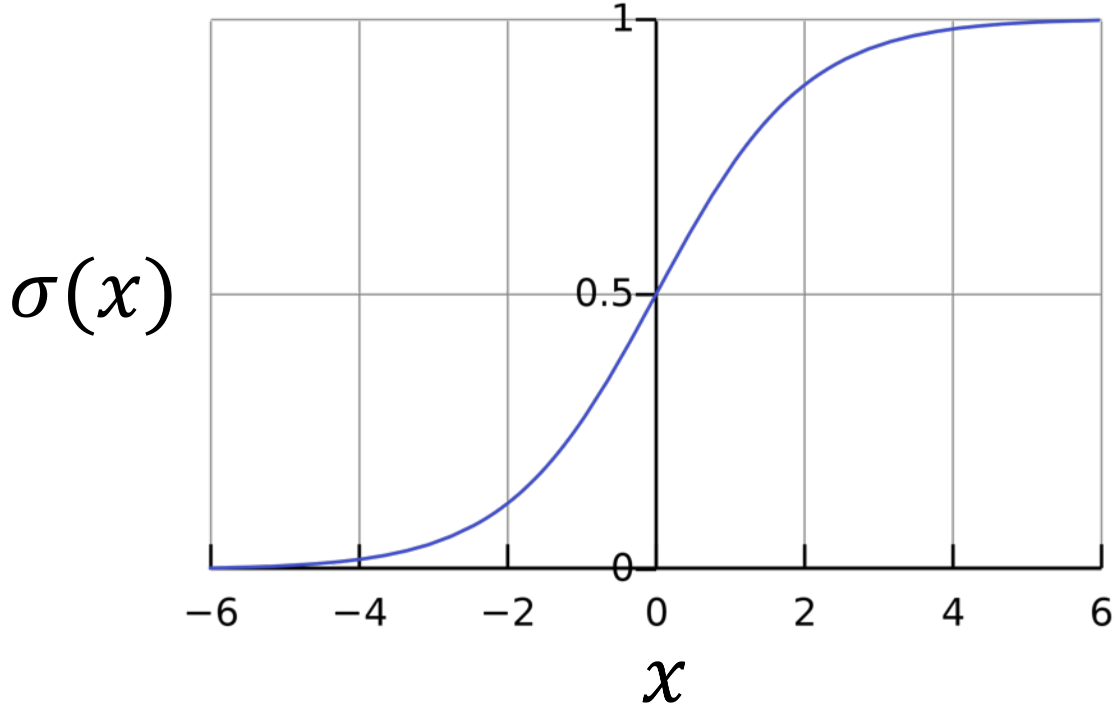
The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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

Check:

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



SIGMOID FUNCTION

Main benefit: can be interpreted as a probability

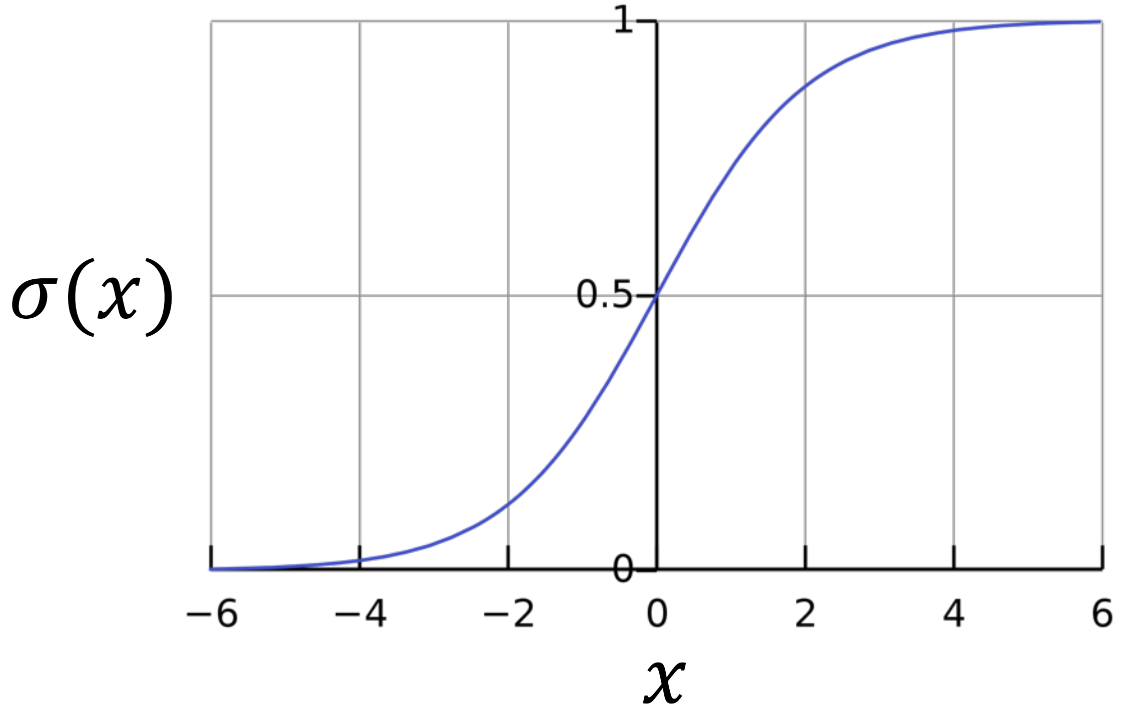
The sigmoid function $\sigma(x)$ maps the real numbers to the range $(0, 1)$

It is defined as:

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

Check:

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



RUNNING EXAMPLE: SPAM CLASSIFICATION

$$\overbrace{\hphantom{xxxxxxxxxxxxxxxx}}^{\text{Features}}$$

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

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

(Predicted probability of the email being spam)

$$\boxed{?}$$

$$\hat{y}$$

PARAMETERS OF LOGISTIC REGRESSION

Features

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

$x = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$ ⇒ **?** $\hat{y}$

(Predicted probability of the email being spam)

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

$x = \binom{5}{2}$ ⇒ ? $\hat{y}$

(Predicted probability of the email being spam)

Weights $w = \binom{1}{2}$ Bias $b = -5$

Parameters:

1	2	-5

Prediction: $\boxed{\hat{y} = \sigma(x \cdot w + b)} = \sigma\left(\binom{5}{2} \cdot \binom{1}{2} - 5\right) = \sigma(9-5) = \frac{1}{1+e^{-4}} = 0.982$

Sigmoid function Dot product

QUIZ: SPAM CLASSIFICATION

Features

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

$x = \binom{1}{0}$

(Predicted probability of the email being spam)

? $\hat{y}$

Weights $w = \binom{1}{2}$

Bias $b = -5$

Parameters:

1	2

-5

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

QUIZ: SPAM CLASSIFICATION

Features

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

$x = \binom{1}{0}$

(Predicted probability of the email being spam)

? $\hat{y}$

Weights $w = \binom{1}{2}$

Bias $b = -5$

Parameters:

1	2

-5

Prediction:

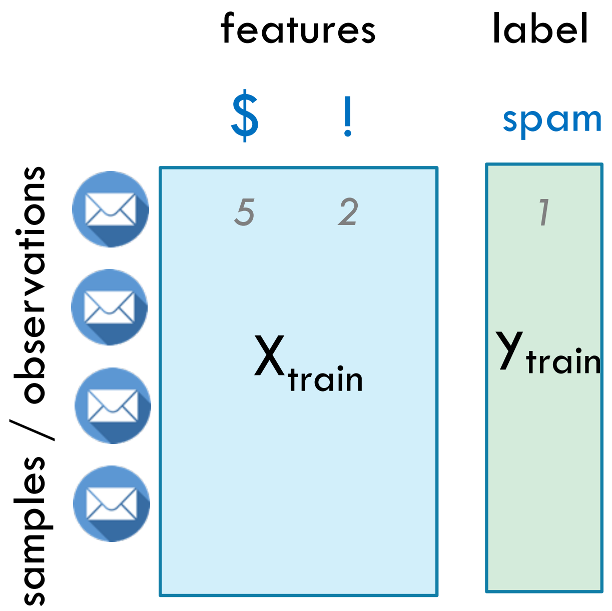
$$\boxed{\hat{y} = \sigma(x \cdot w + b)} = \sigma\left(\binom{1}{0} \cdot \binom{1}{2} - 5\right) = \sigma(1-5) = \frac{1}{1+e^4} = 0.018$$

Sigmoid function

Dot product

TRAINING LOGISTIC REGRESSION

Training Data



Minimize Cost Function $J(w, b)$

Logistic Regression Parameters

Weights $w = \binom{1}{2}$

1	2

Bias $b = -5$

-5

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:

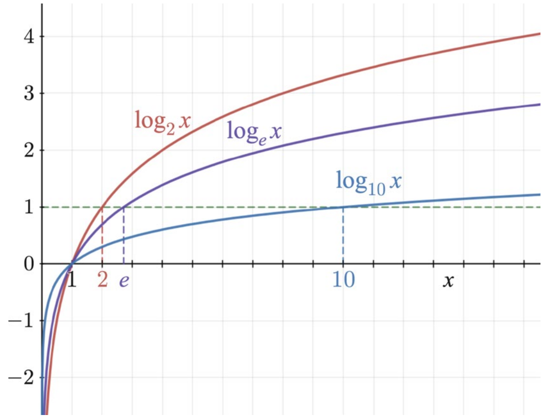
Cross Entropy Loss $L(\hat{\mathbf{y}}, \mathbf{y})$ represents the "**disagreement**" between our predictions $\hat{\mathbf{y}}$ and the labels $\mathbf{y}$.

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).

CROSS-ENTROPY LOSS (FOR A SINGLE DATA POINT)

Predictions   Labels

$\hat{y}$	y
0.2	1



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

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

How to interpret this?

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

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

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 (FOR 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

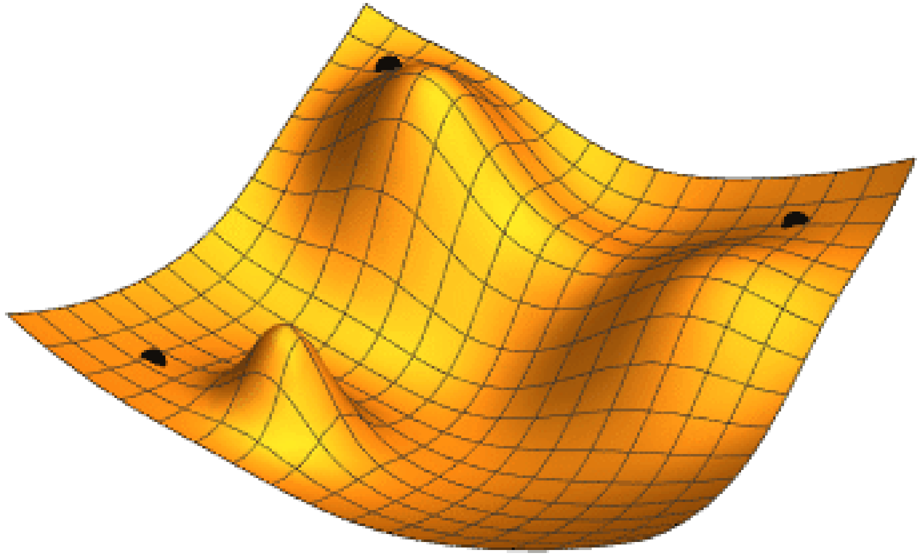
Cost function $J(w,b)$ **for parameters** $\boldsymbol{w}, \boldsymbol{b}$ is defined as the Cross Entropy Loss of the predictions $\hat{y}_i$ obtained from w, b:

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

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 $J(w,b)$

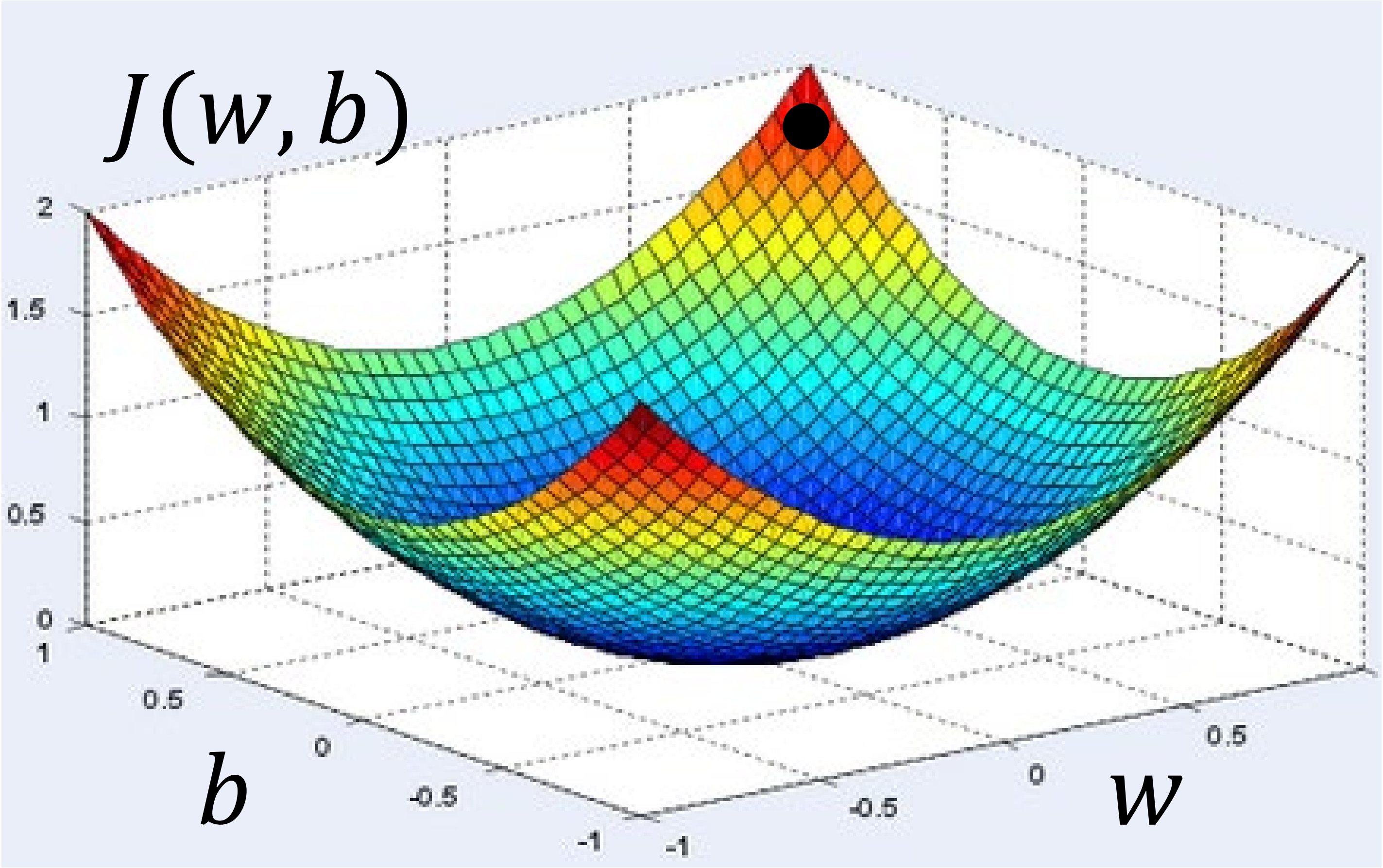
MINIMIZING COST FUNCTIONS: GRADIENT DESCENT



MINIMIZING COST FUNCTIONS: GRADIENT DESCENT

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

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



MINIMIZING COST FUNCTIONS: GRADIENT DESCENT

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

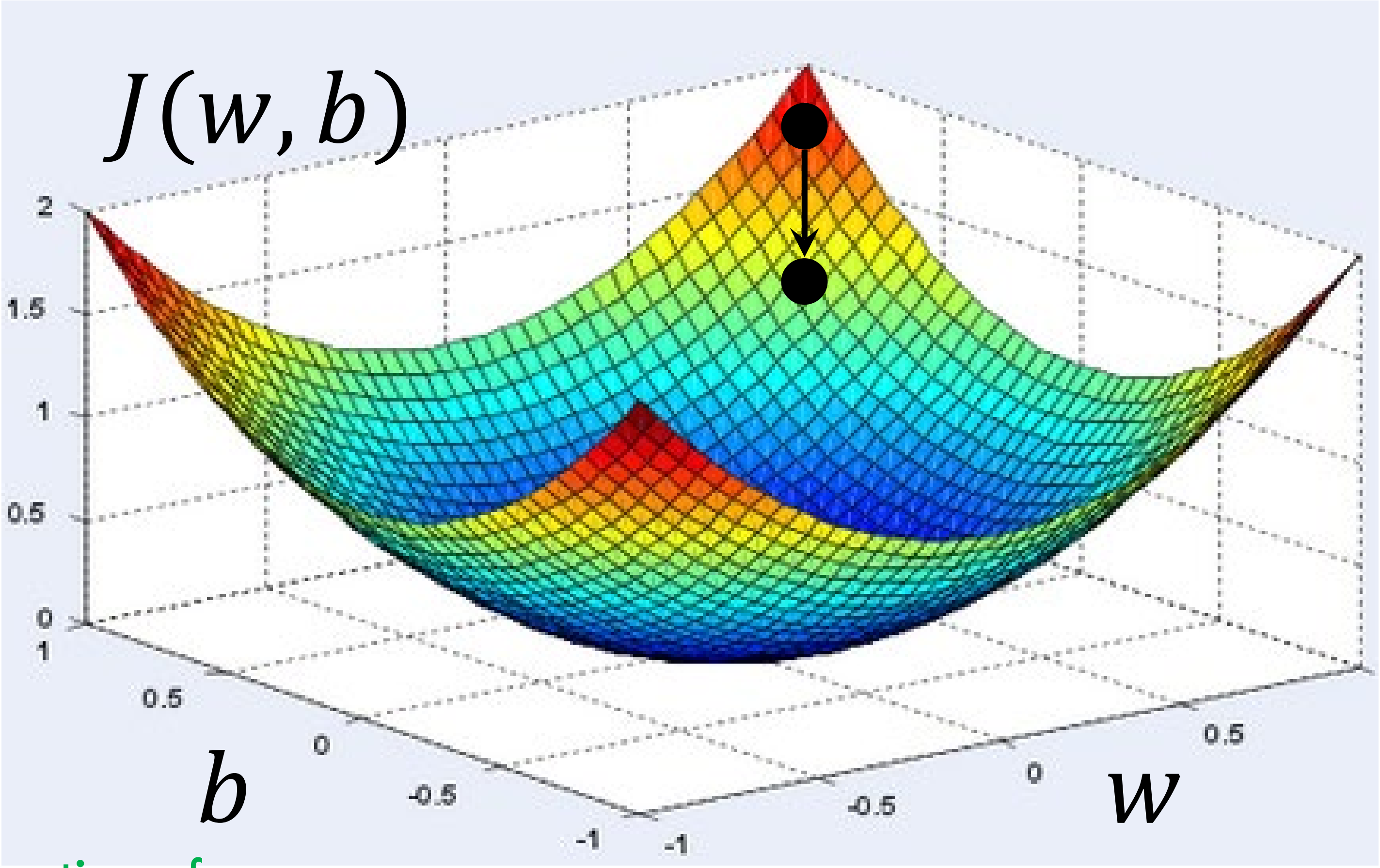
1. Start at an arbitrary point, then move in the negative gradient direction

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

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

"Learning Rate"

"Slope": direction of steepest decrease of J



MINIMIZING COST FUNCTIONS: GRADIENT DESCENT

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

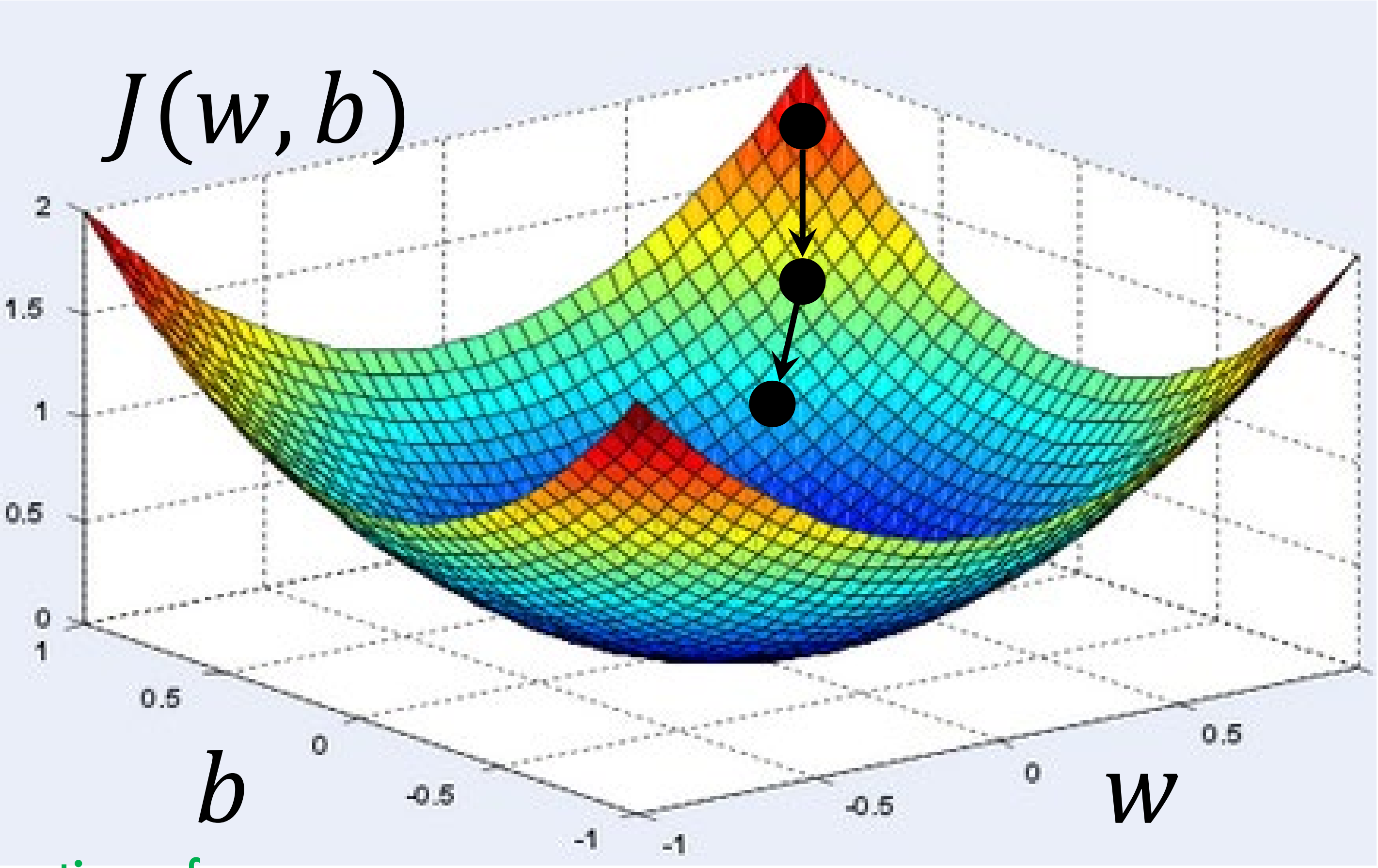
1. Start at an arbitrary point, then move in the negative gradient direction

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

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

"Learning Rate"

"Slope": direction of steepest decrease of J



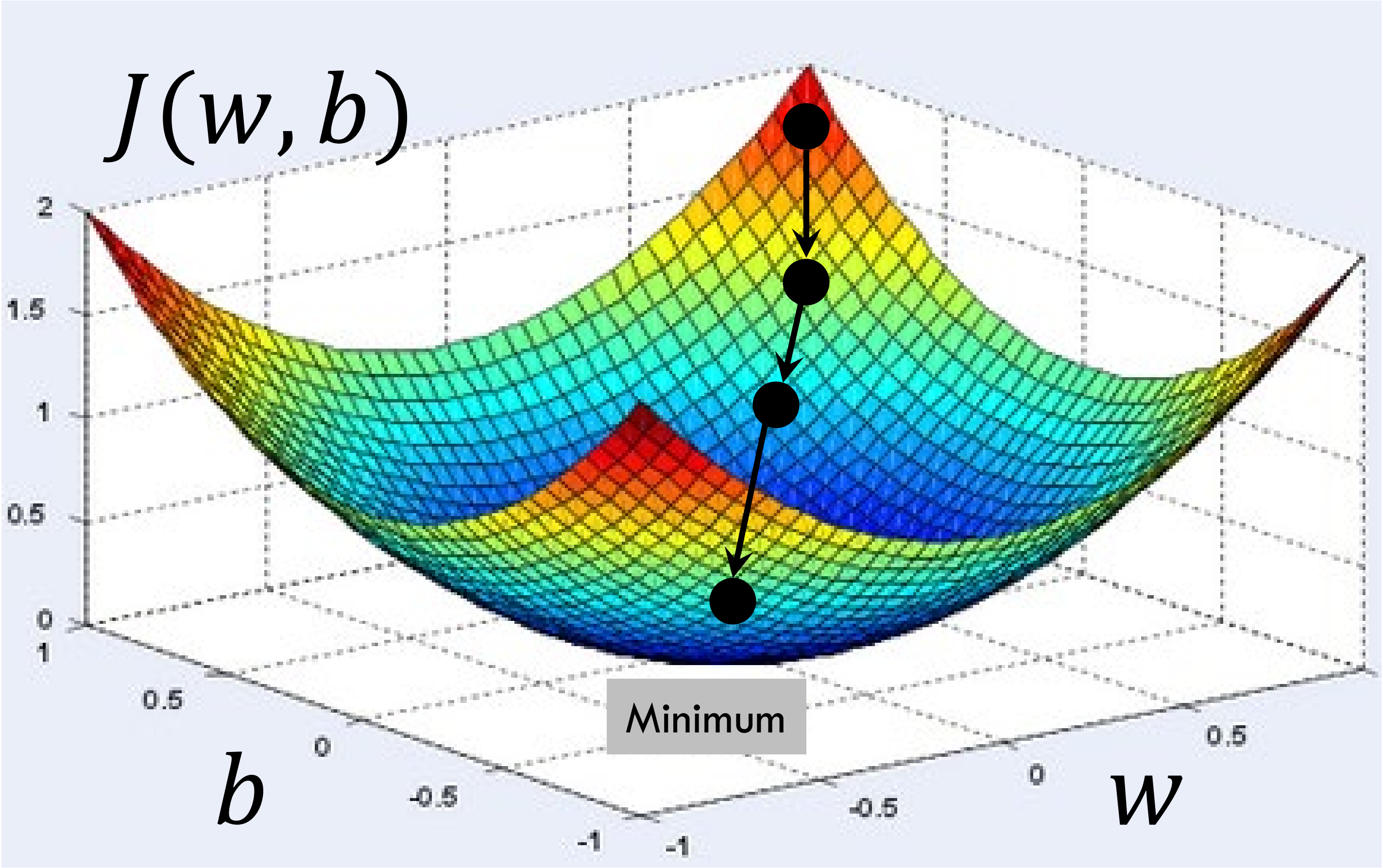
MINIMIZING COST FUNCTIONS: GRADIENT DESCENT

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

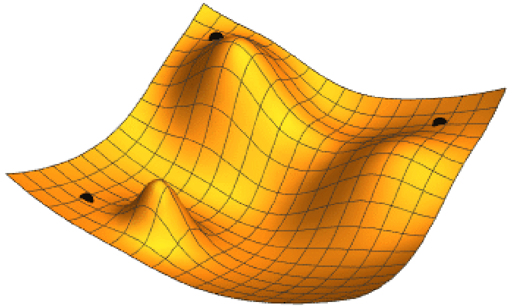
1. Start at an arbitrary point, then move in the negative gradient direction

2. Continue until convergence

- Stop when improvement in J is below a fixed threshold



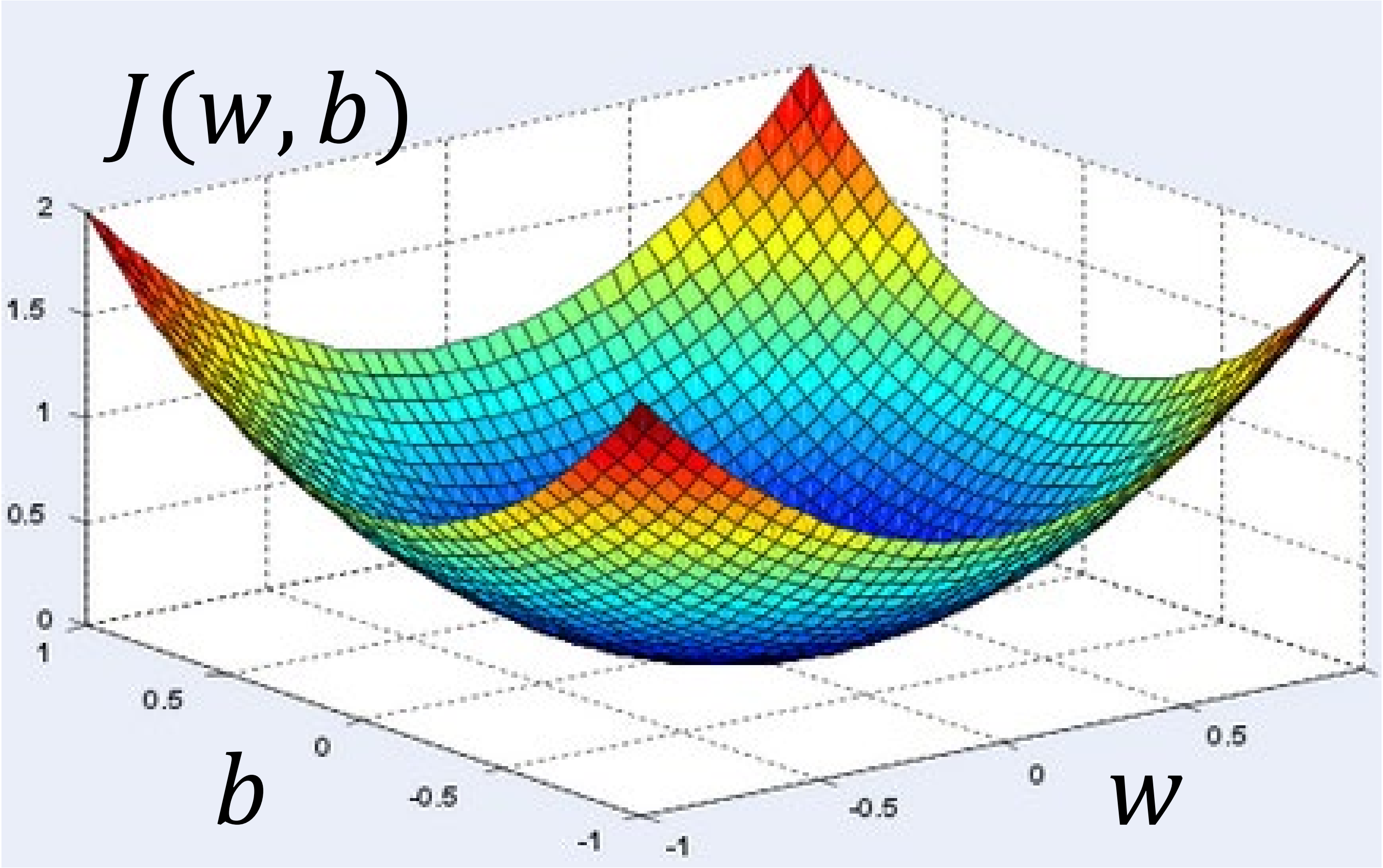
GLOBAL MINIMUM VS LOCAL MINIMA



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

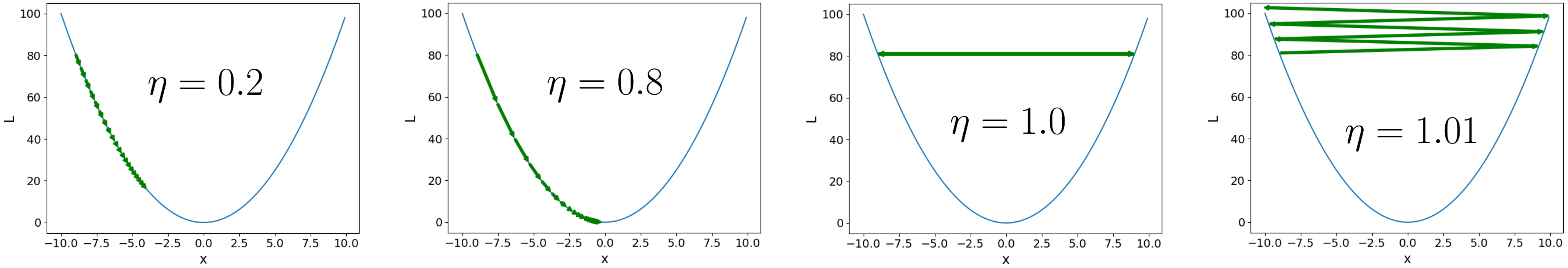
while true :

  Calculate gradient $\nabla_\theta L$

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

In practice: stop loop when loss converges

EFFECTS OF LEARNING RATE



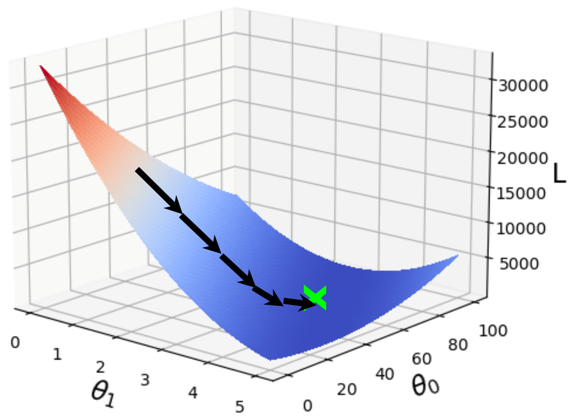
$$L = x^2, \ \frac{\partial L}{\partial x} = 2x, \ \text{20 steps}$$

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

GRADIENT DESCENT: VARIATIONS

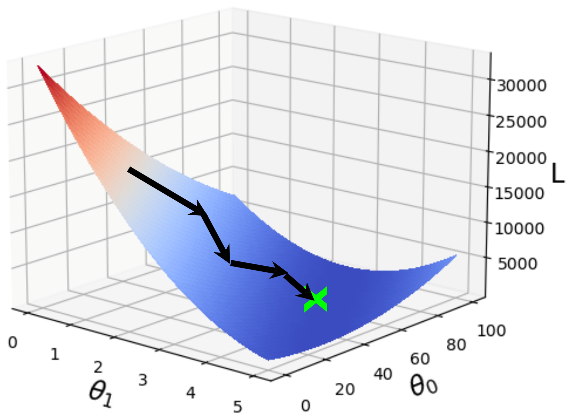
Full-Batch Gradient Descent

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



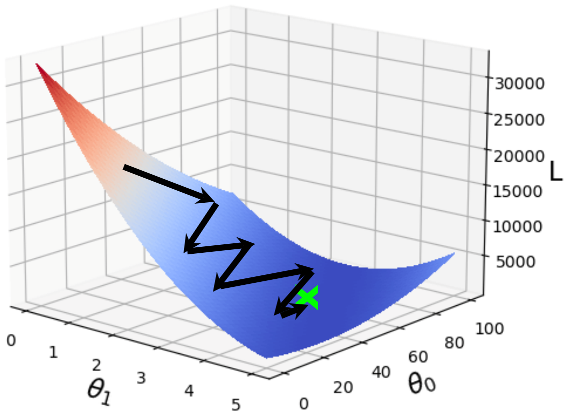
Mini-batch Gradient Descent

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



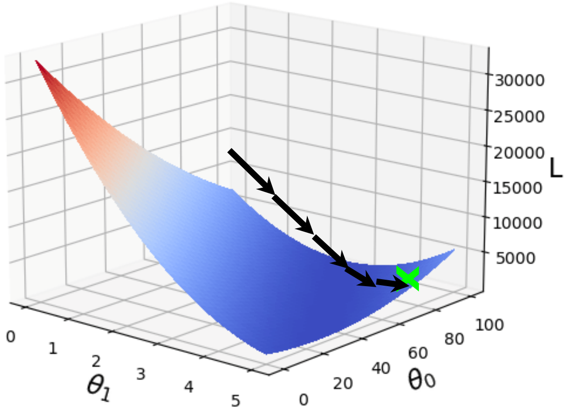
Stochastic Gradient Descent (SGD)

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



GRADIENT DESCENT: VARIATIONS

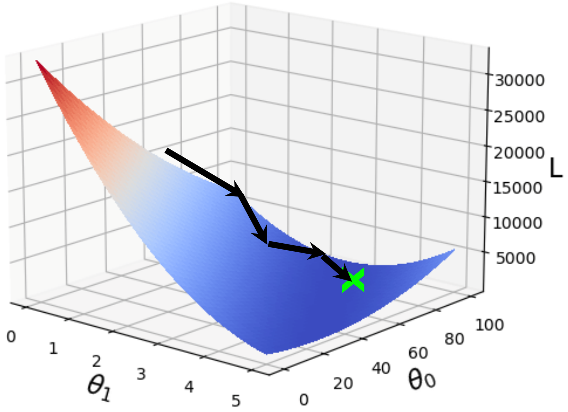
Full-Batch Gradient Descent



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

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

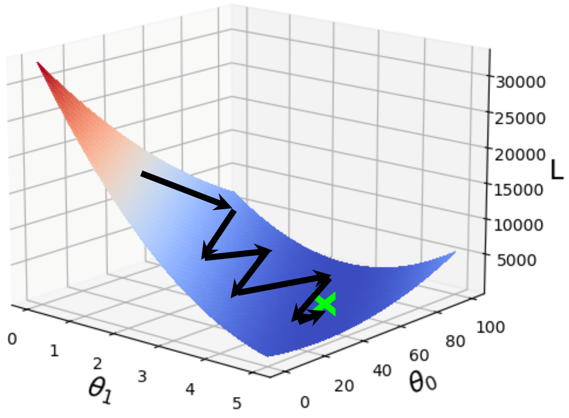
Mini-Batch Gradient Descent



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

Trade-off for gradients and update steps

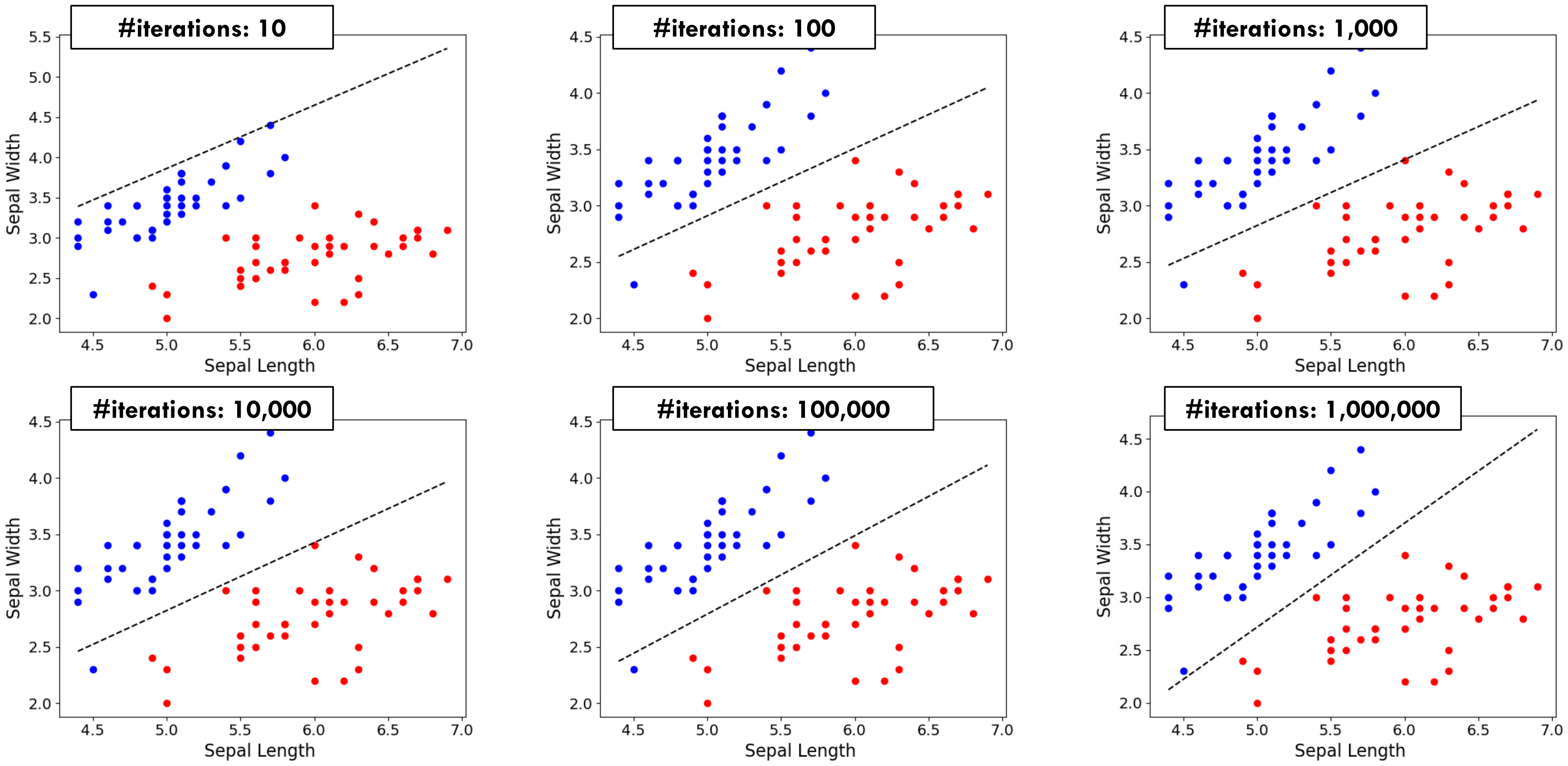
Stochastic Gradient Descent



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 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.

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: $ $

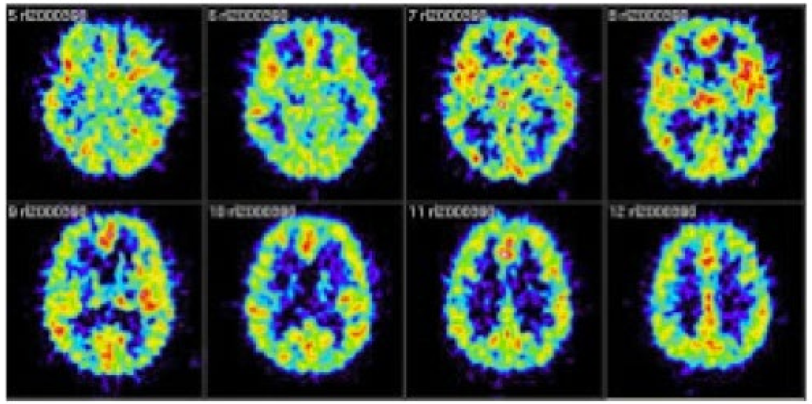
Email B: ! ! !

A: Email A

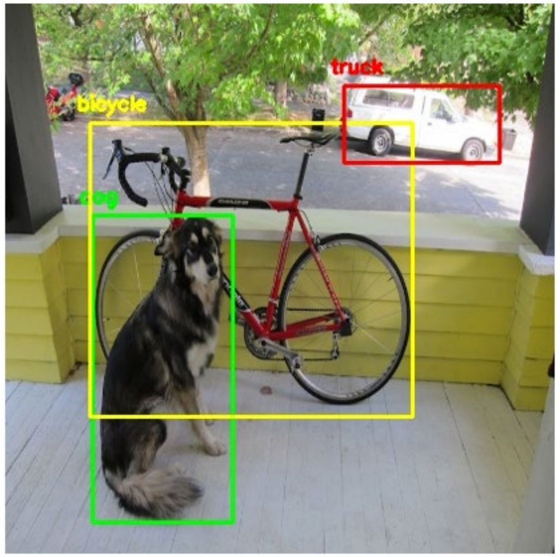
CLASSIFICATION OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Trees and Ensembles
5. Logistic Regression
6. Deep Learning

APPLICATIONS OF DEEP LEARNING



Medical Imaging



Object Detection

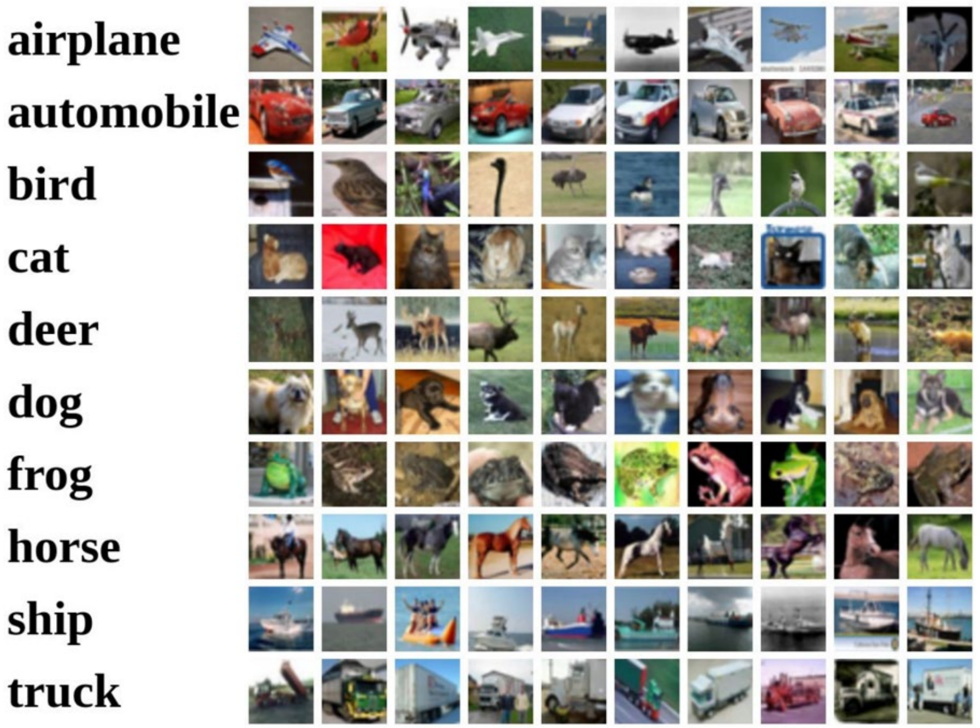
Voice Recognition

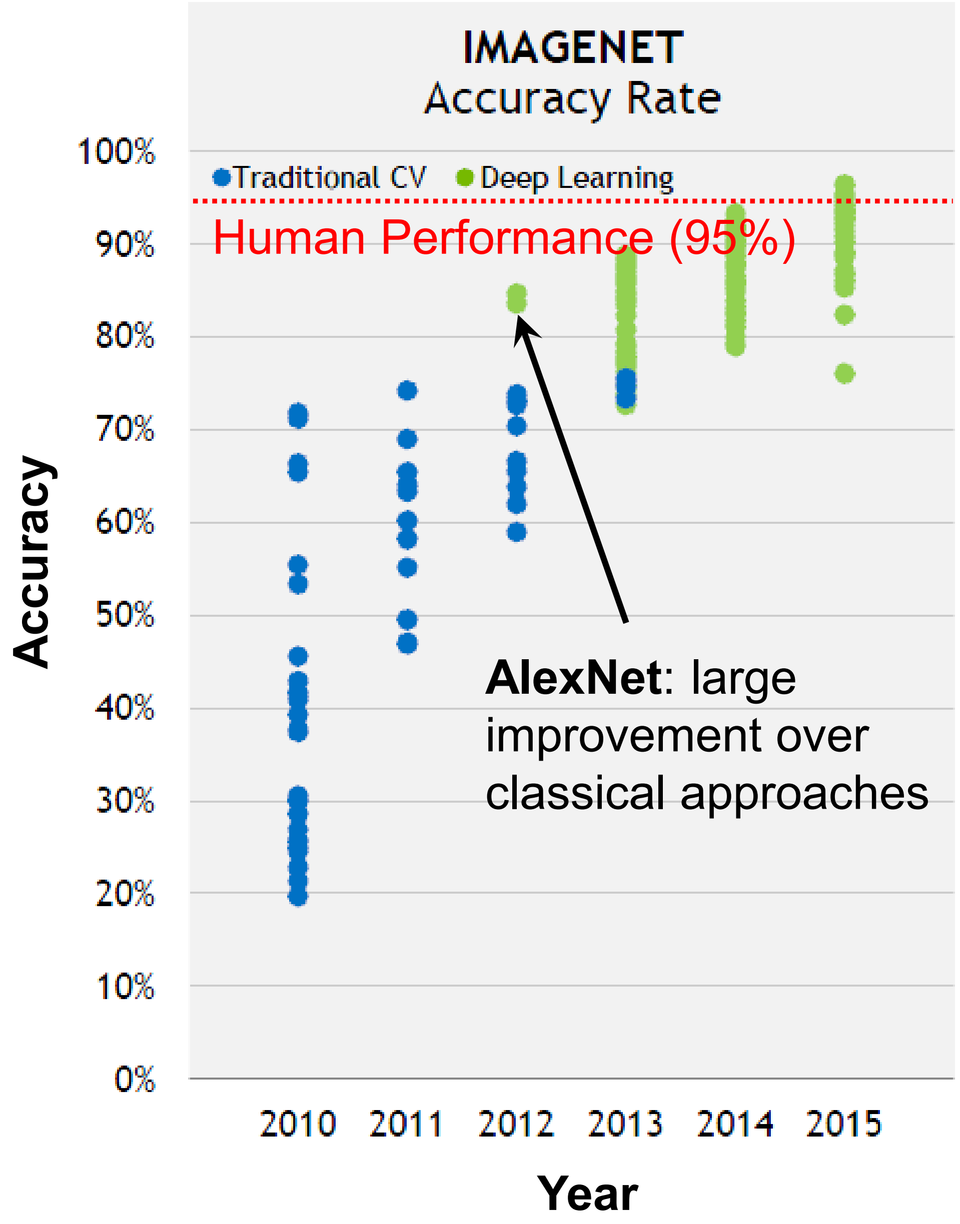
Translation

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*.

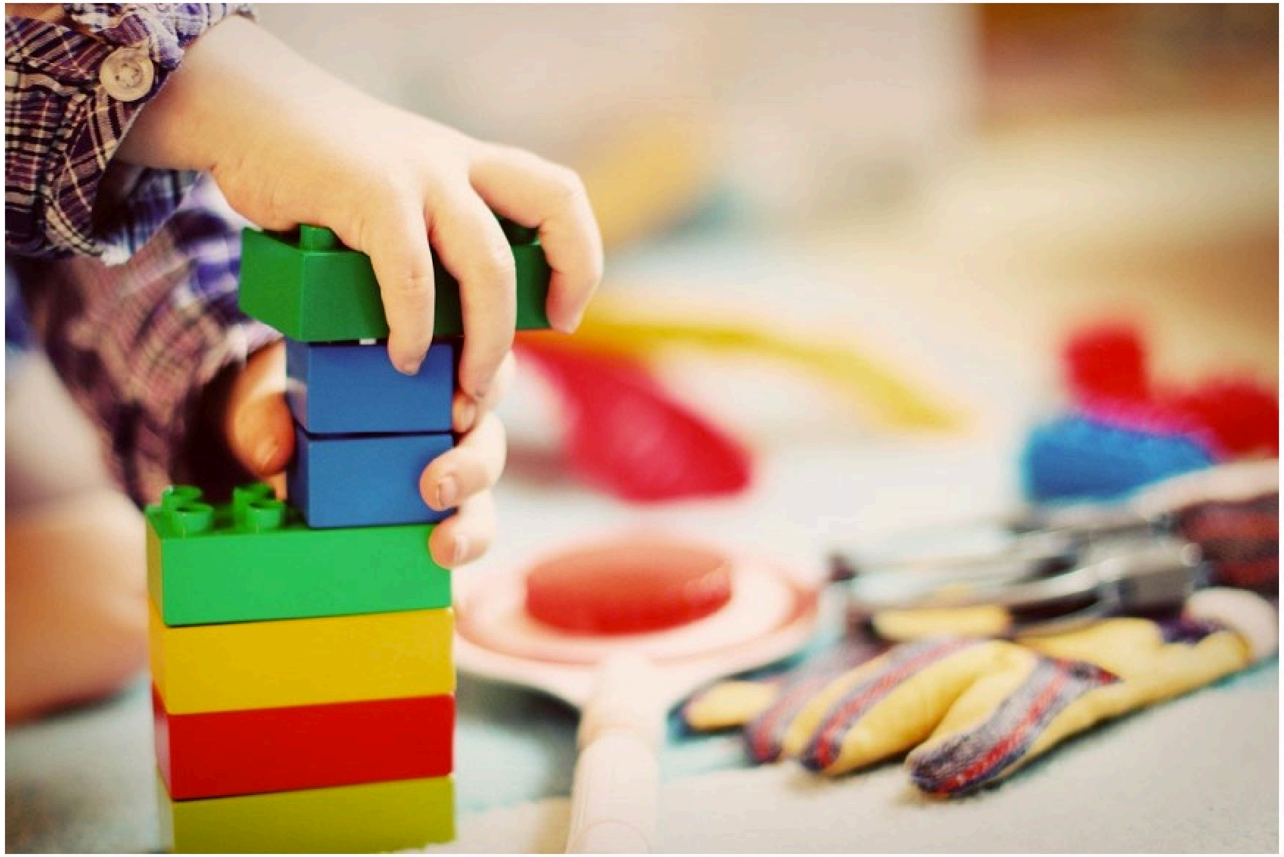
DEEP LEARNING AND IMAGE CLASSIFICATION



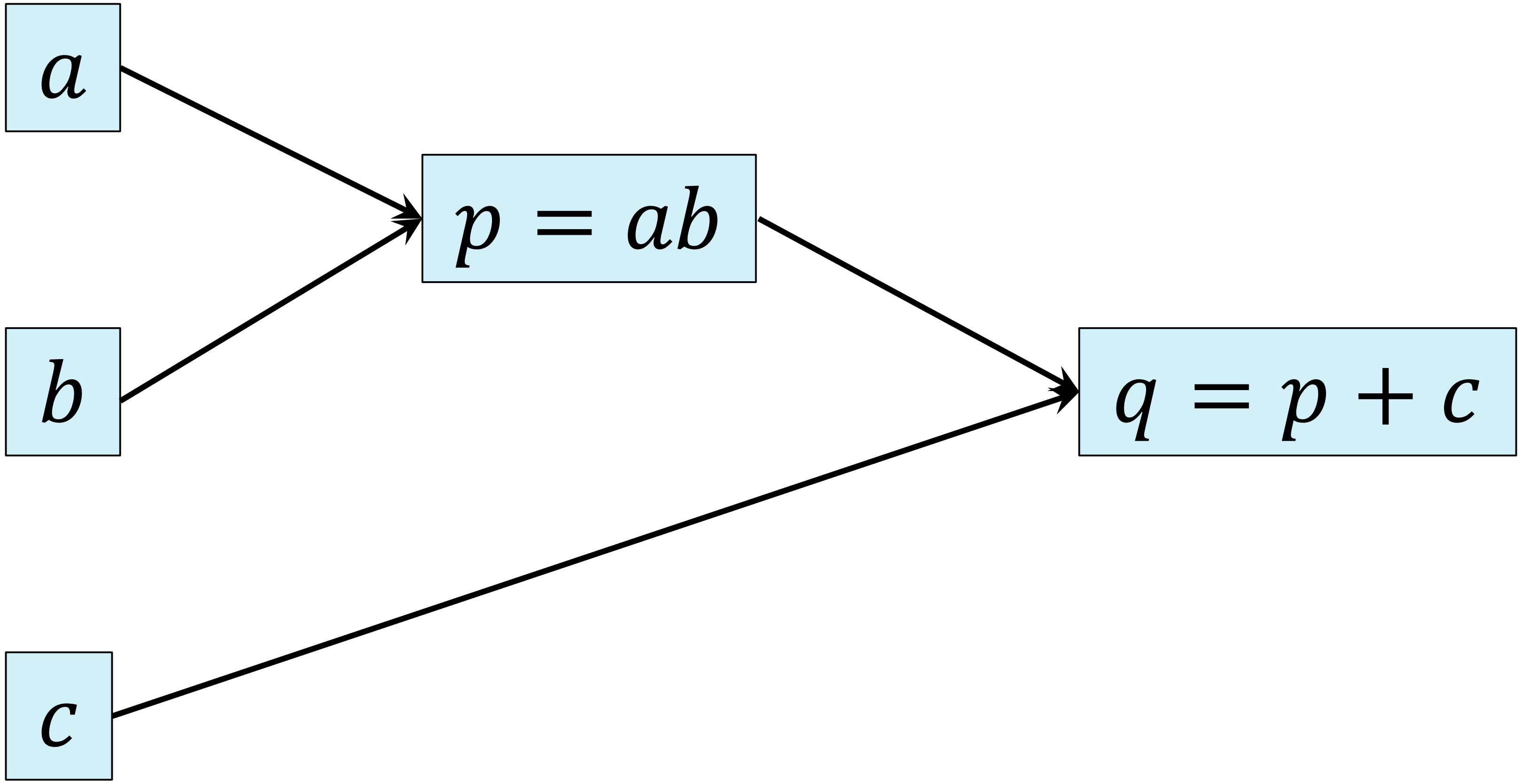


Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", 2009.

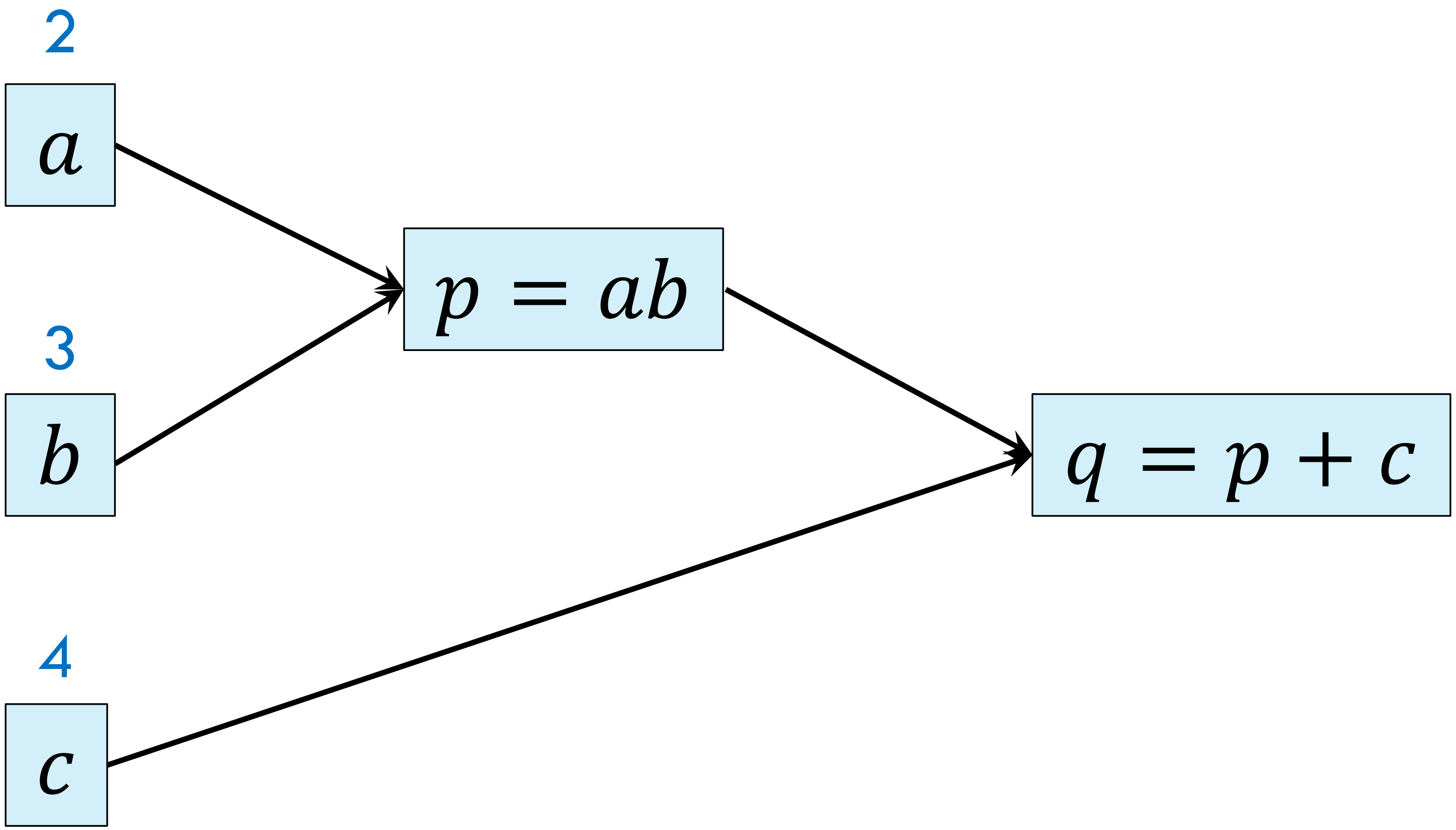
BUILDING COMPLEX CLASSIFIERS FROM SIMPLE PARTS



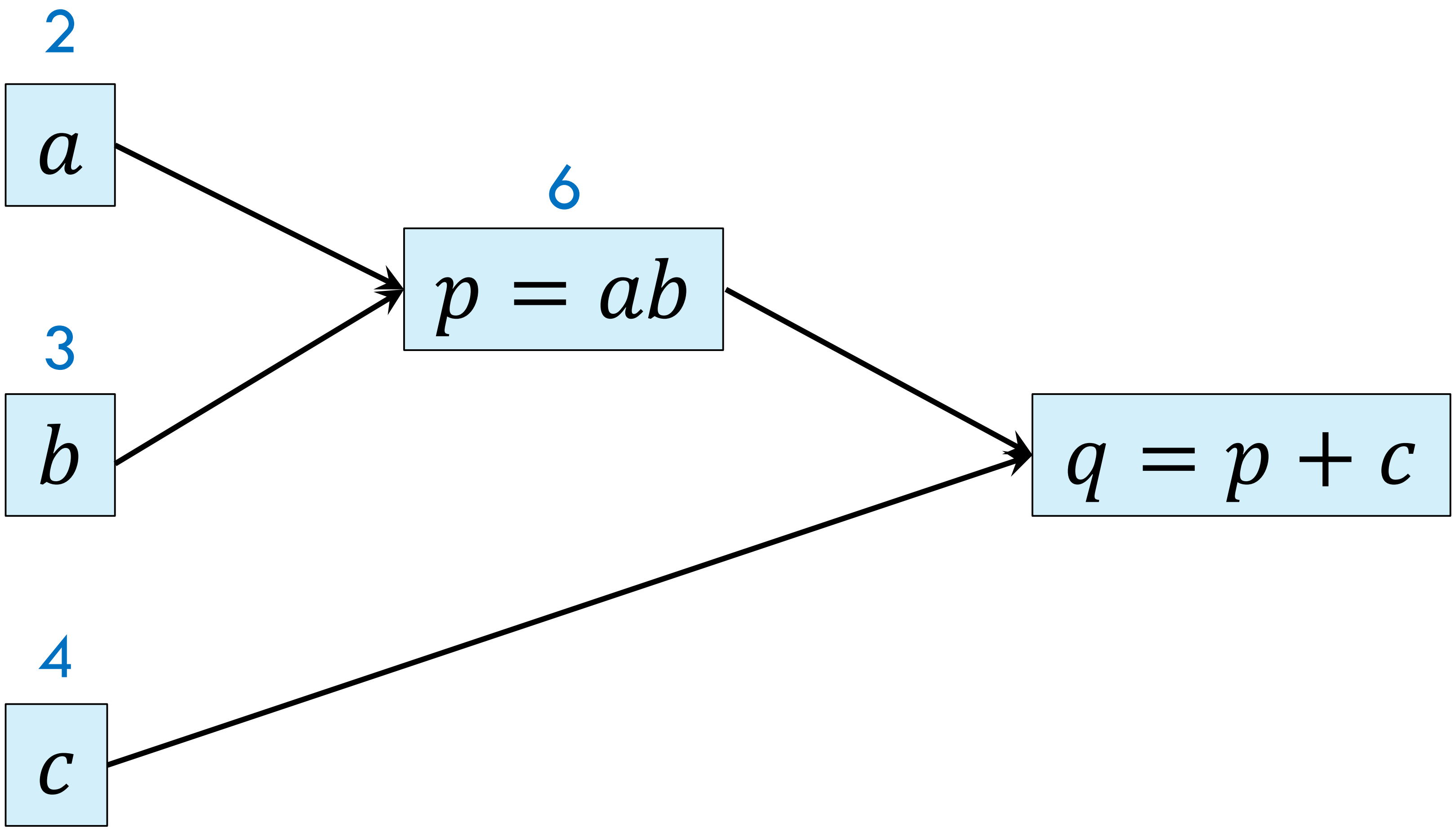
COMPUTATIONAL GRAPH



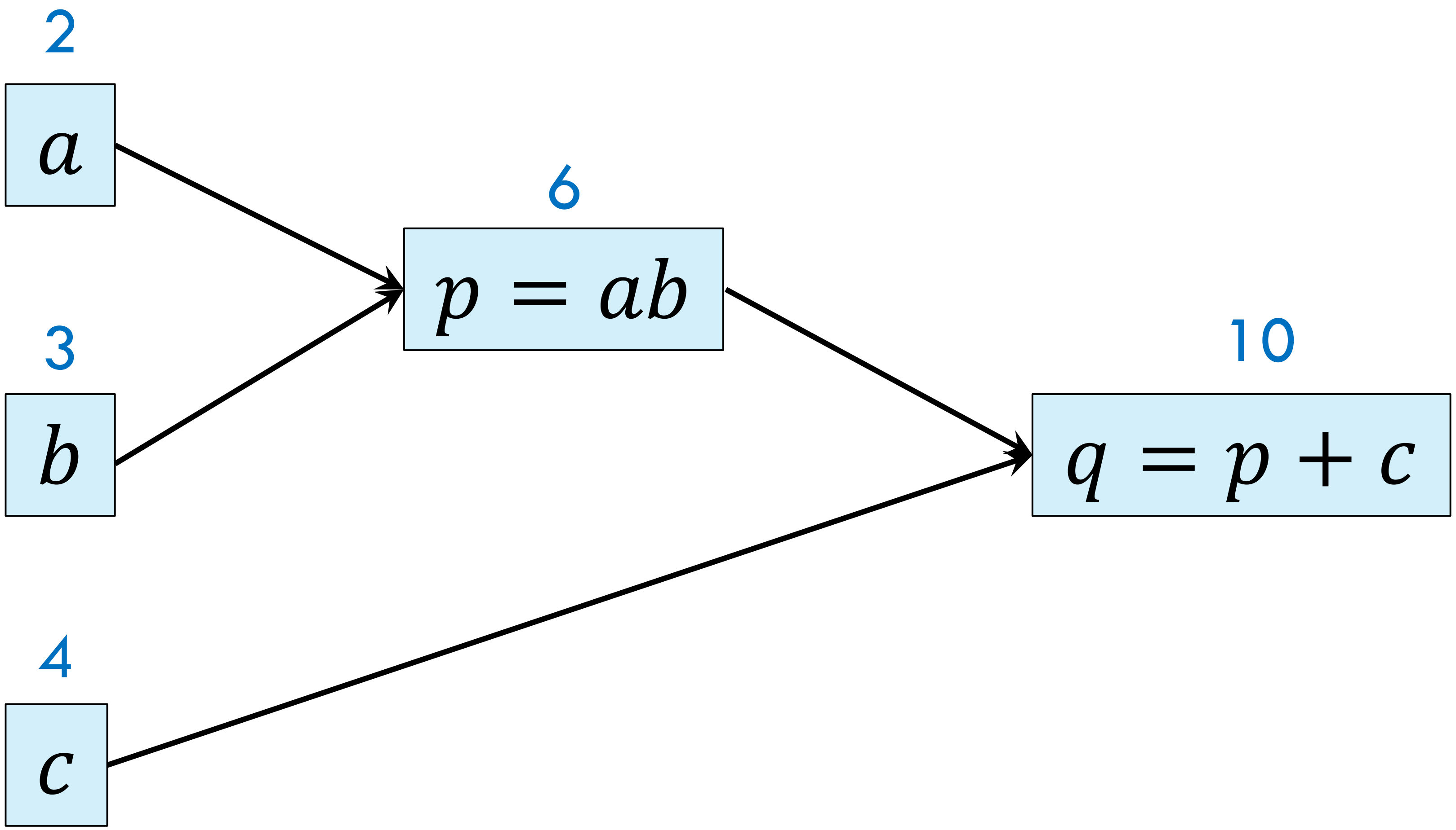
COMPUTATIONAL GRAPH: FORWARD PASS



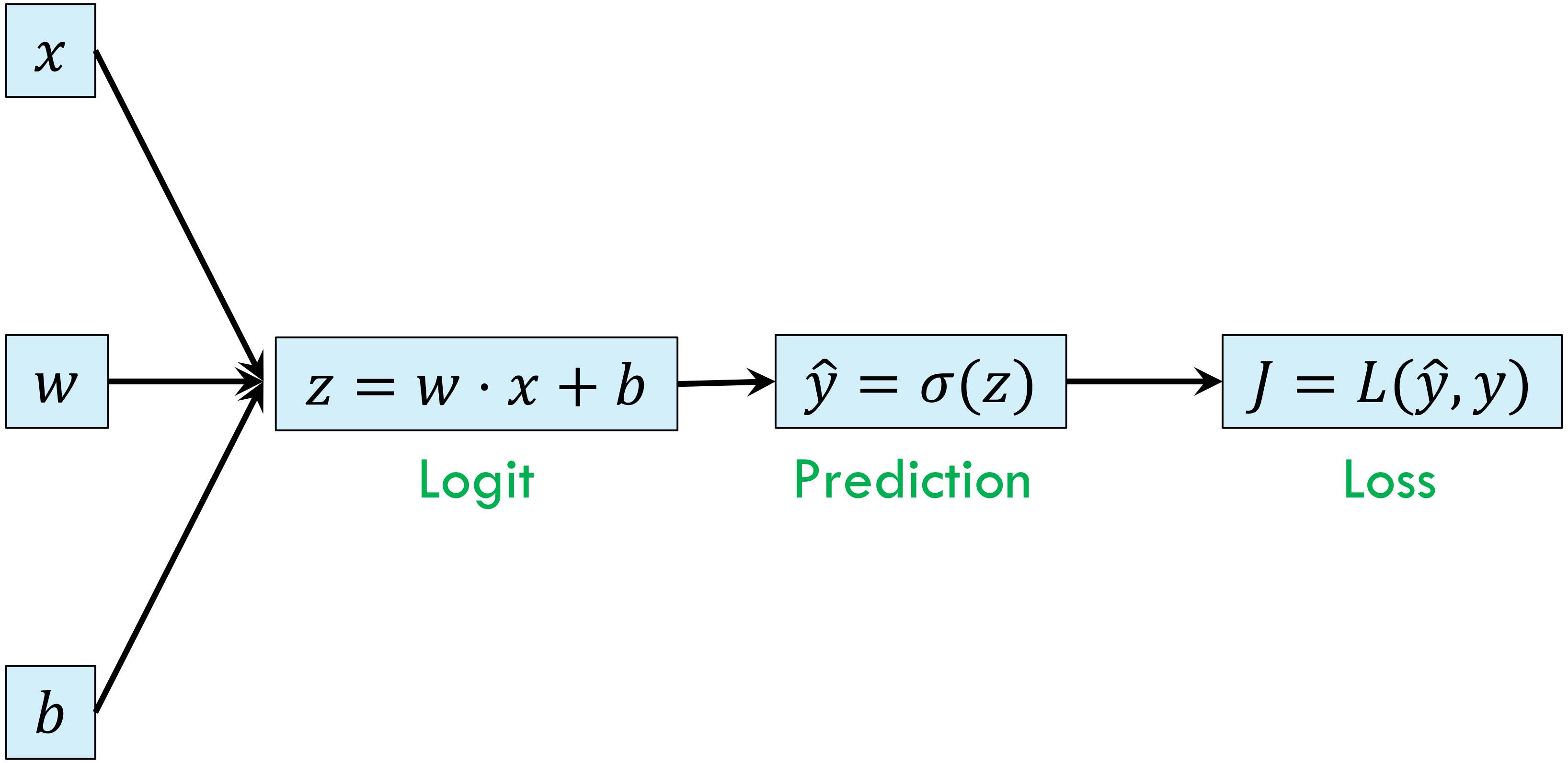
COMPUTATIONAL GRAPH: FORWARD PASS



COMPUTATIONAL GRAPH: FORWARD PASS



LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH



COMPUTATIONAL GRAPH: BACKWARD PASS

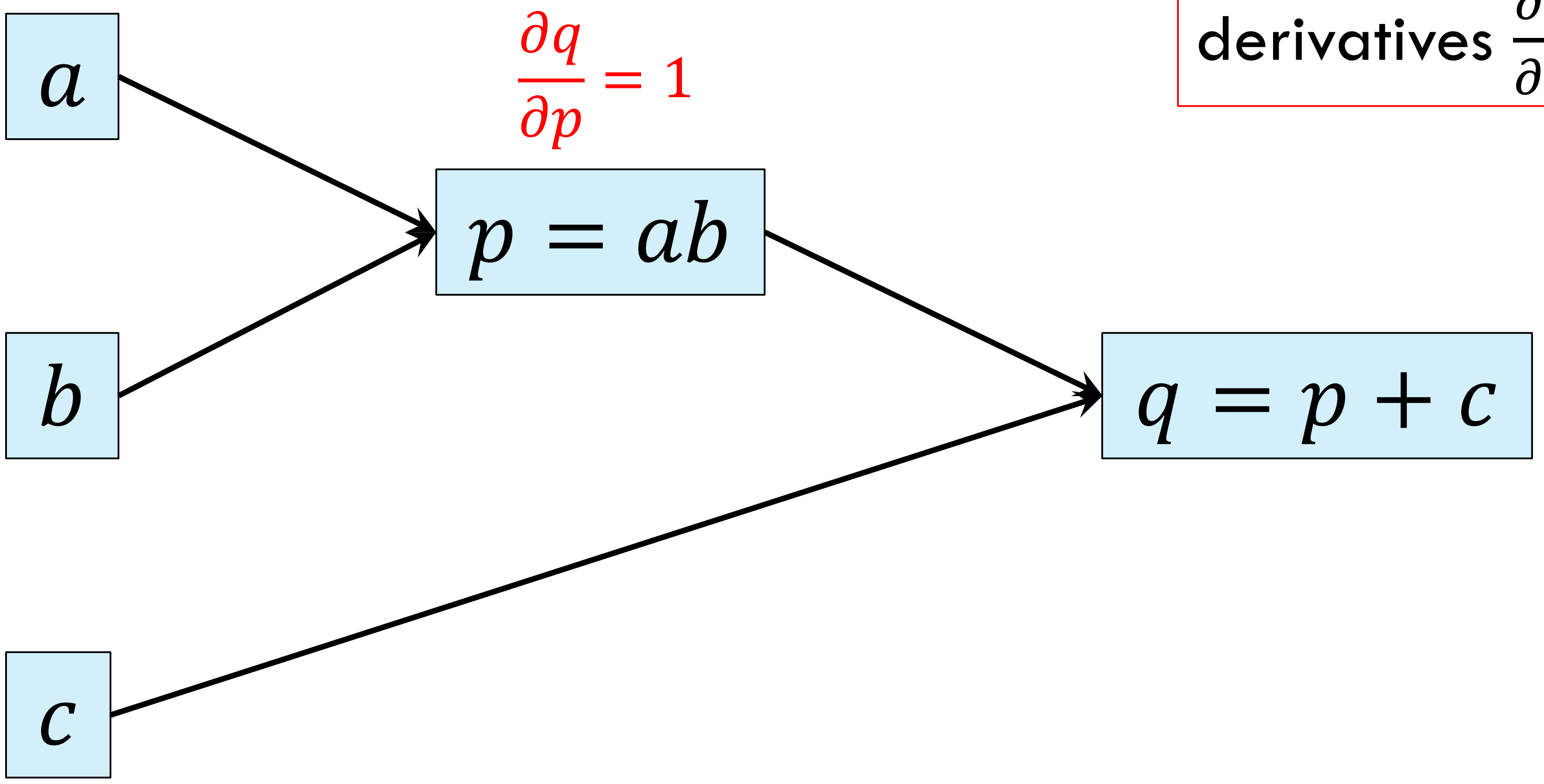
How to compute the partial derivatives $\frac{\partial q}{\partial a}, \frac{\partial q}{\partial b},$ etc.?

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

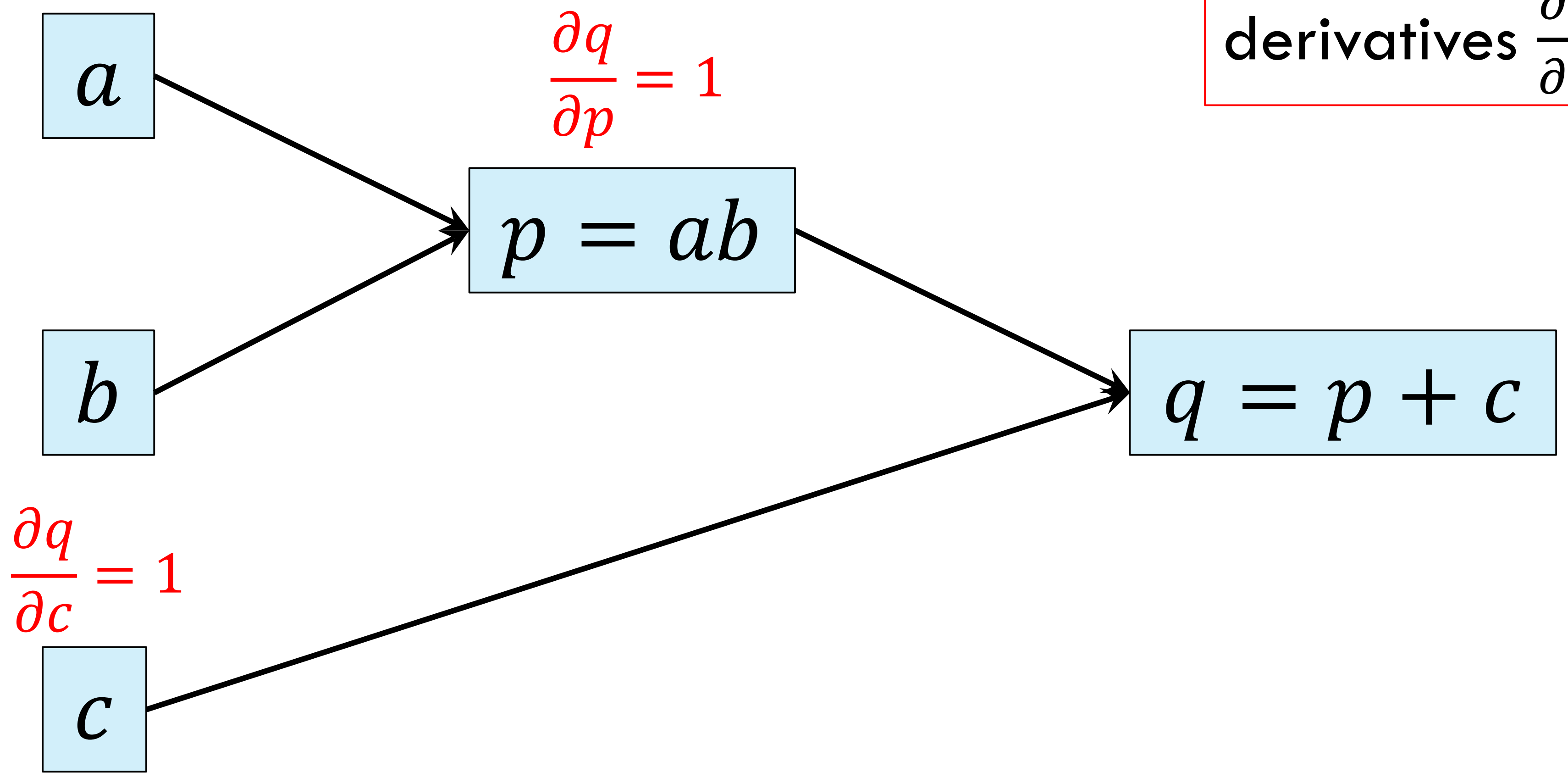
$$\begin{tikzcd}
\boxed{a} \arrow[dr] & & \\
& \boxed{p = ab} \arrow[dr] & \\
\boxed{b} \arrow[ur] & & \boxed{q = p + c} \\
\boxed{c} \arrow[urr] & &
\end{tikzcd}$$

COMPUTATIONAL GRAPH: BACKWARD PASS

How to compute the partial derivatives $\frac{\partial q}{\partial a}, \frac{\partial q}{\partial b}$, etc.?



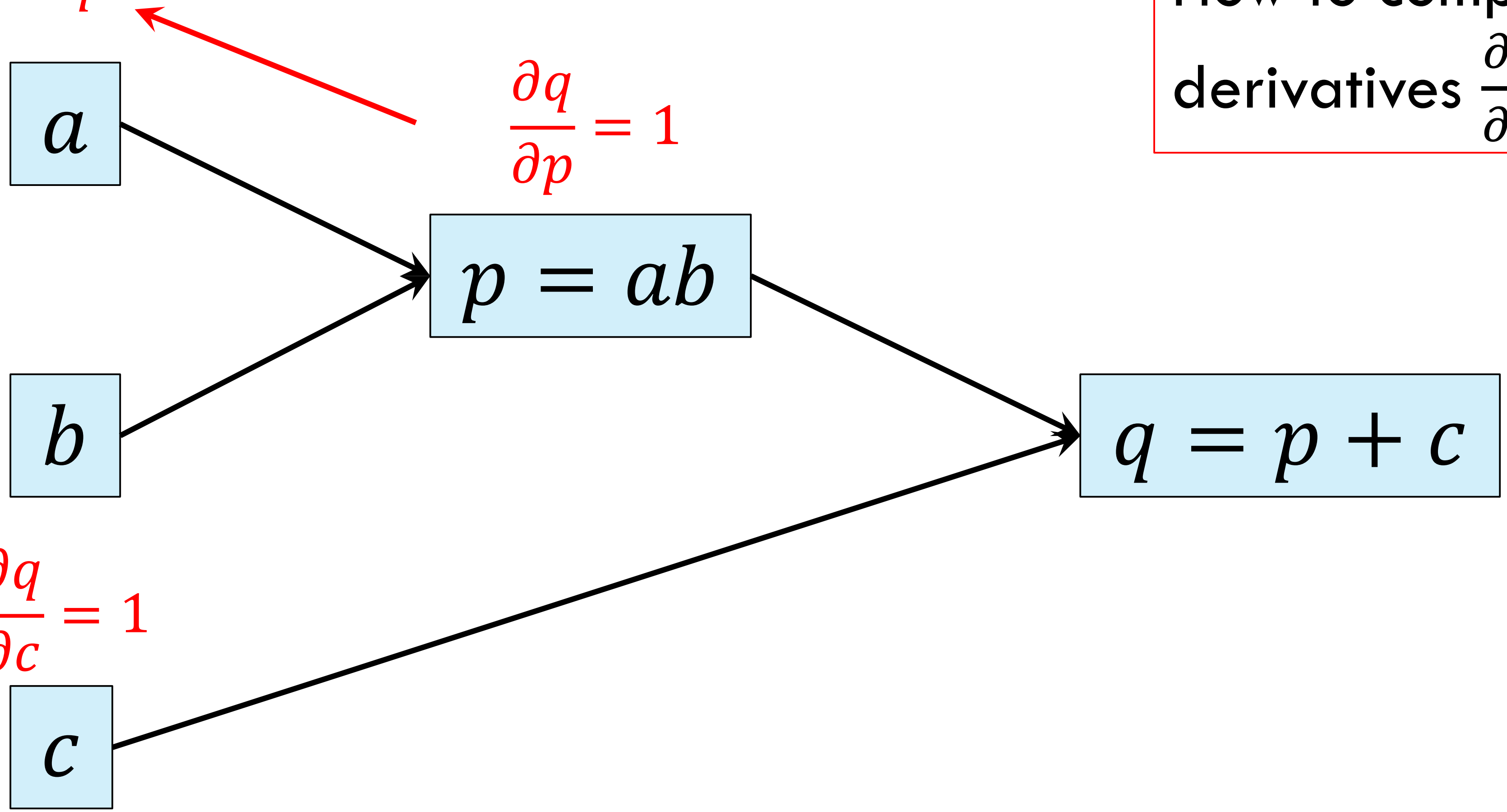
COMPUTATIONAL GRAPH: BACKWARD PASS



How to compute the partial derivatives $\frac{\partial q}{\partial a}, \frac{\partial q}{\partial b}$, etc.?

COMPUTATIONAL GRAPH: BACKWARD PASS

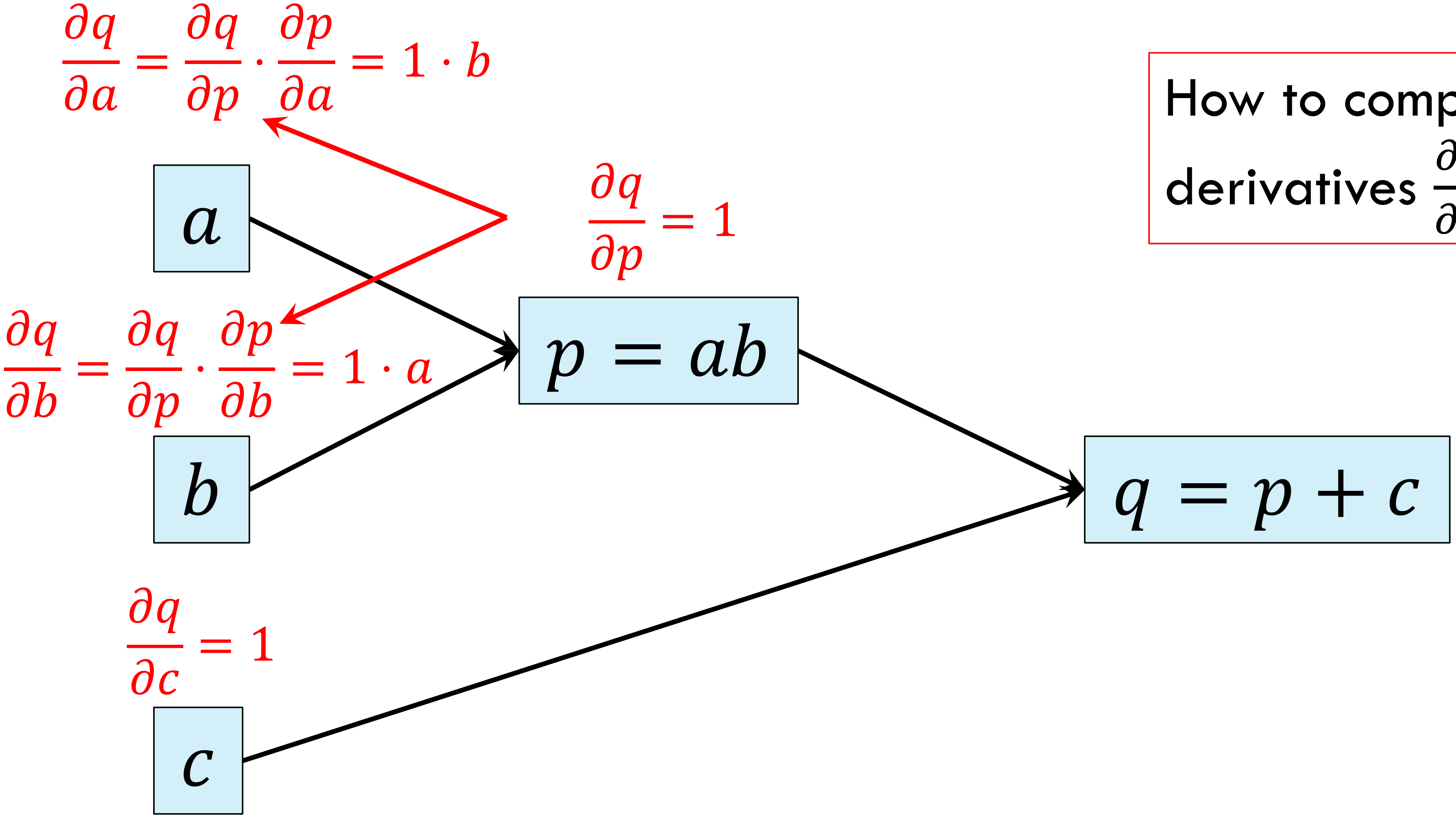
$$\frac{\partial q}{\partial a} = \frac{\partial q}{\partial p} \cdot \frac{\partial p}{\partial a} = 1 \cdot b$$



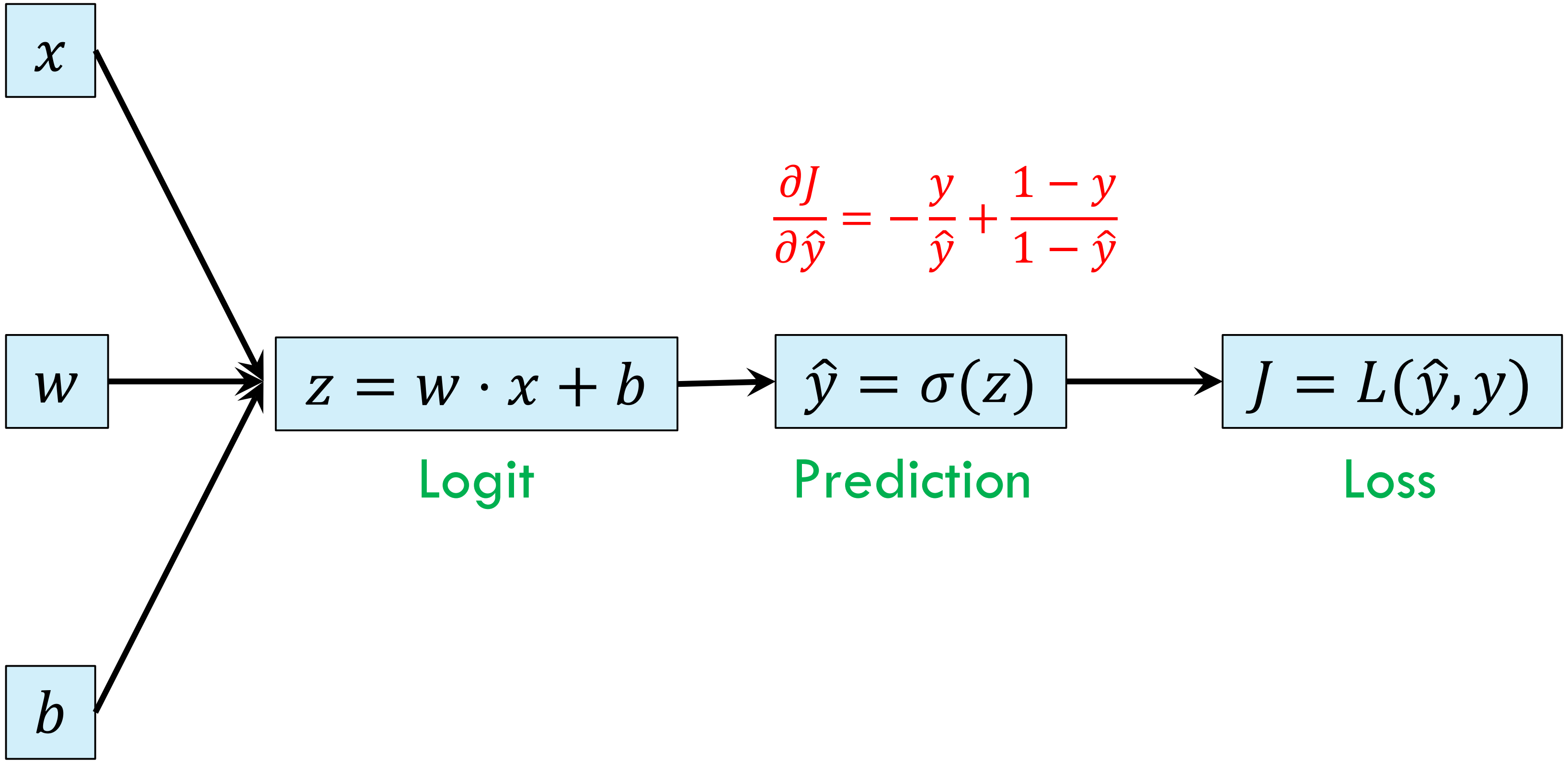
How to compute the partial derivatives $\frac{\partial q}{\partial a}, \frac{\partial q}{\partial b}$, etc.?

COMPUTATIONAL GRAPH: BACKWARD PASS

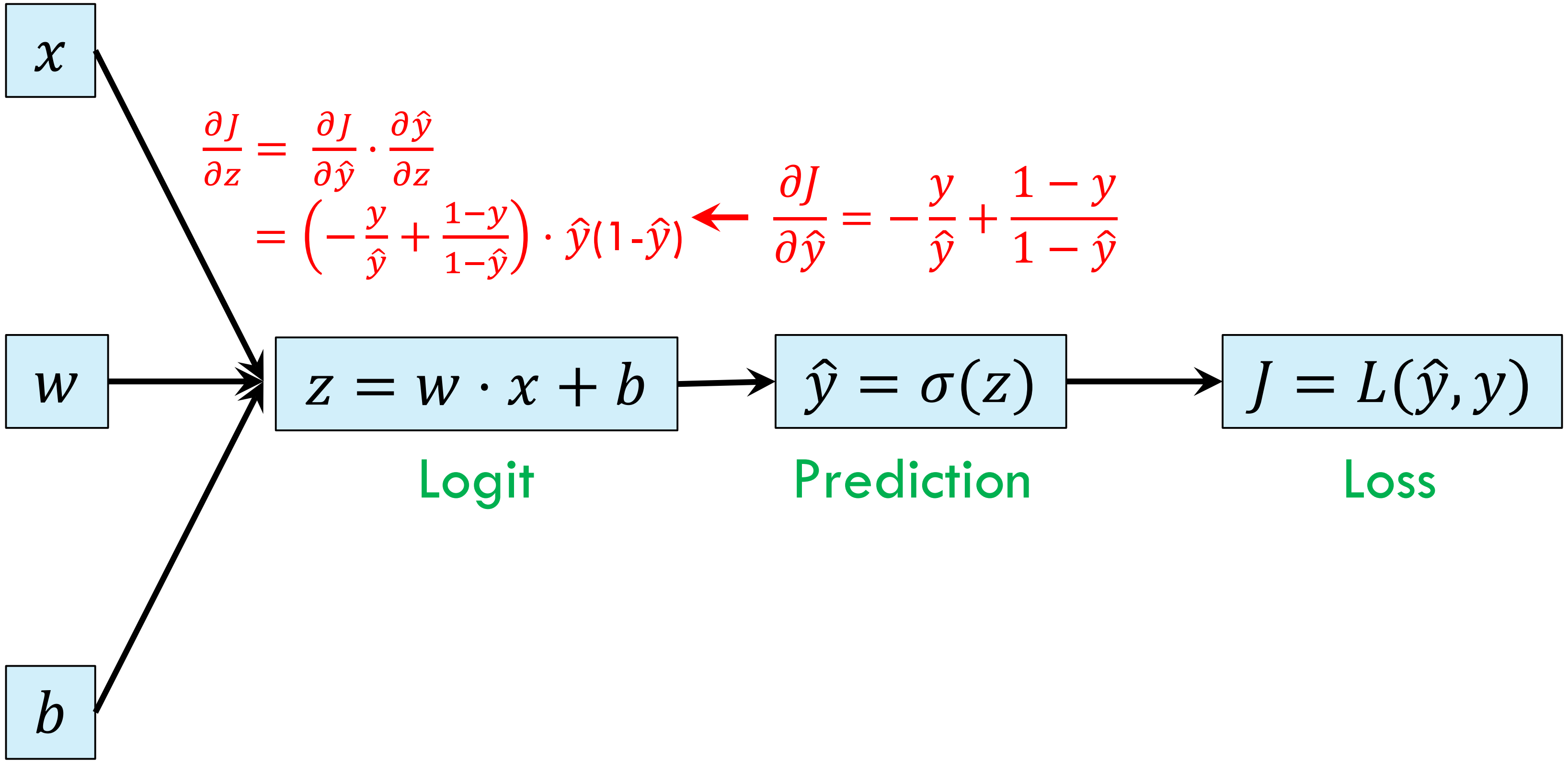
How to compute the partial derivatives $\frac{\partial q}{\partial a}, \frac{\partial q}{\partial b},$ etc.?



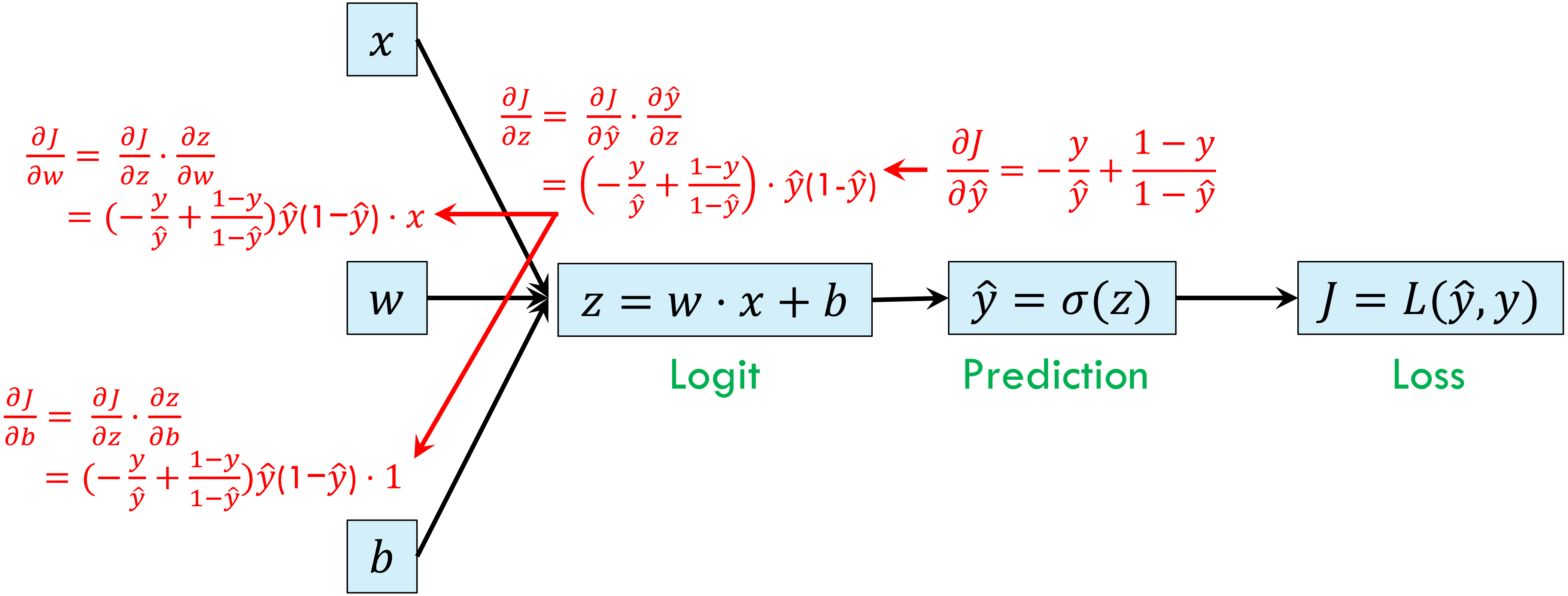
LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH



LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH



LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH

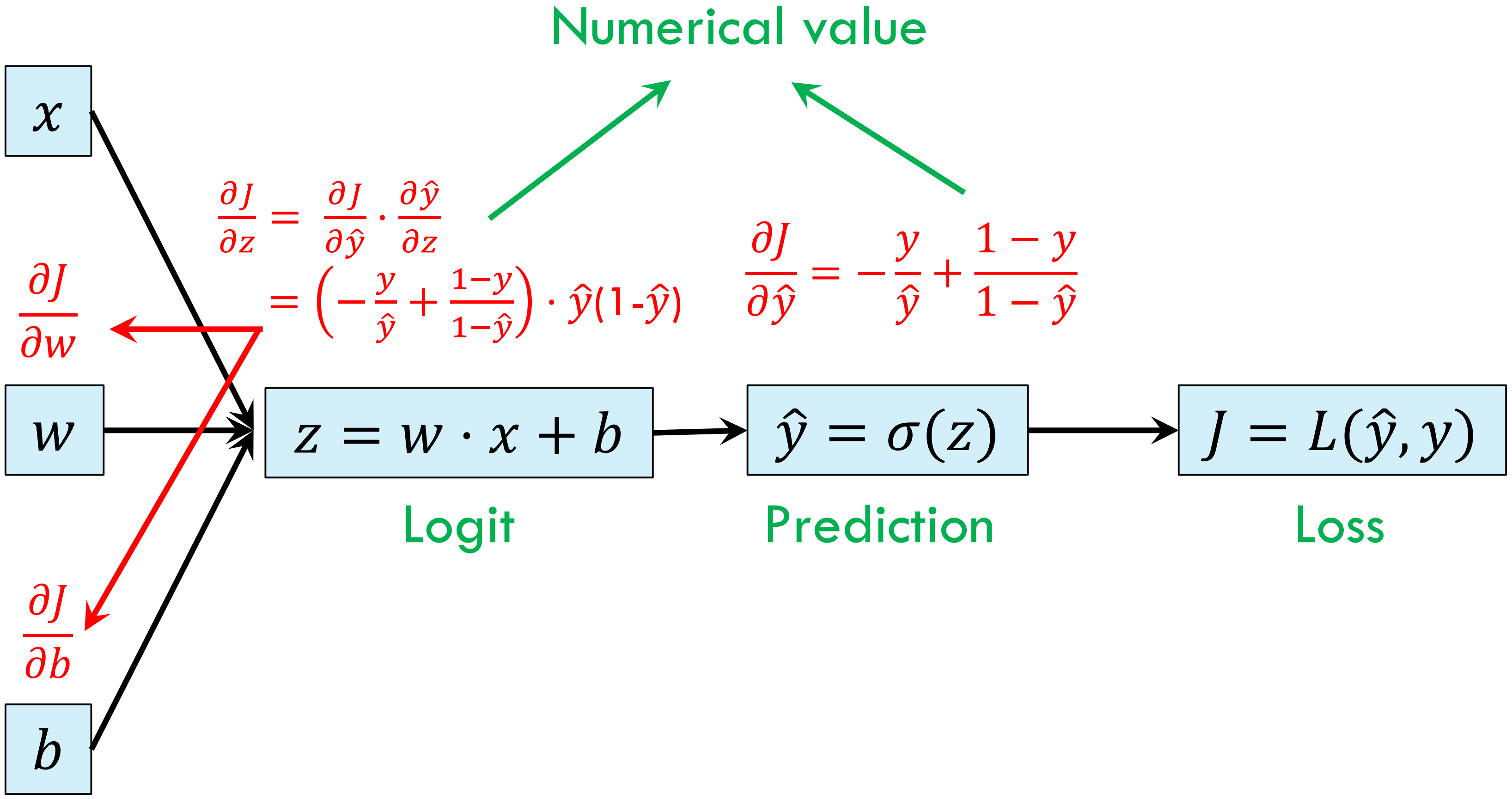


LOGISTIC REGRESSION AS A COMPUTATIONAL GRAPH

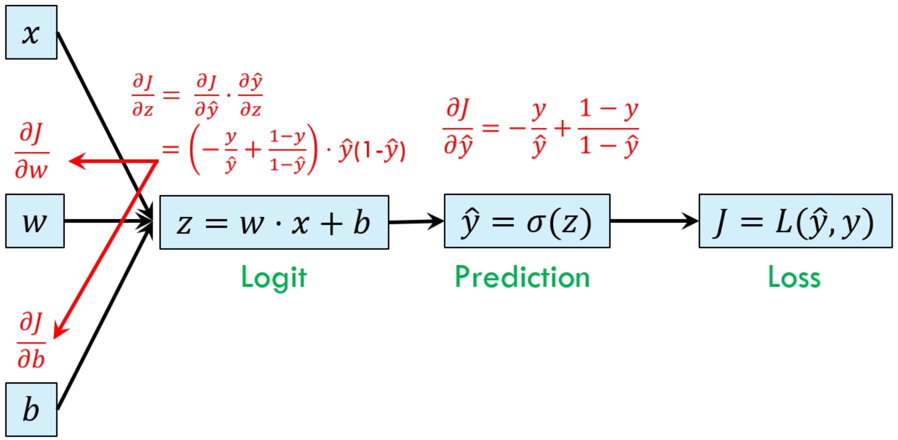
Gradient Descent Steps

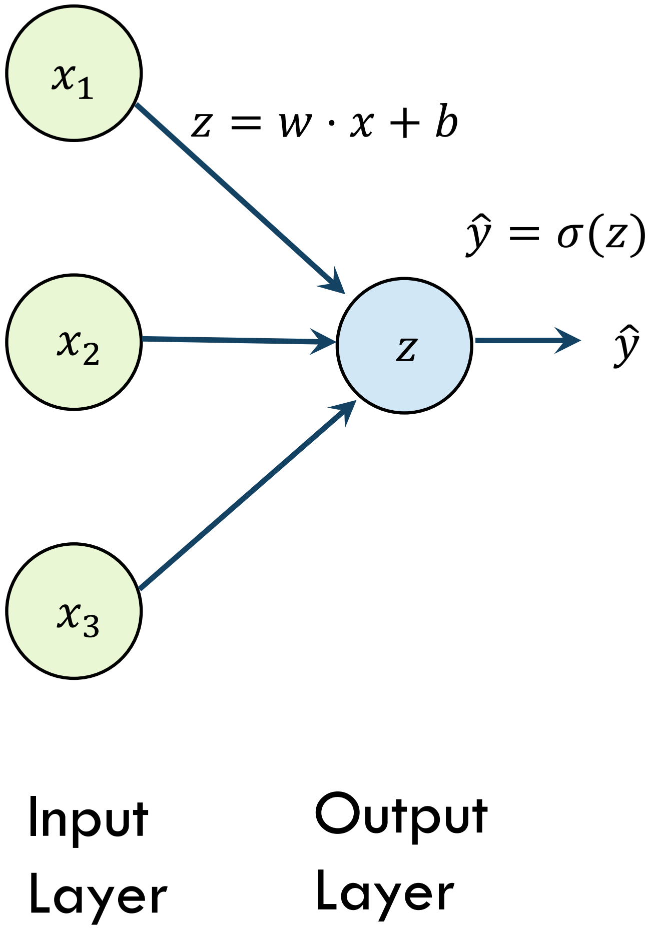
$$w \leftarrow w - \eta \frac{\partial J(w,b)}{\partial w}$$

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



LOGISTIC REGRESSION AS A NEURAL NETWORK

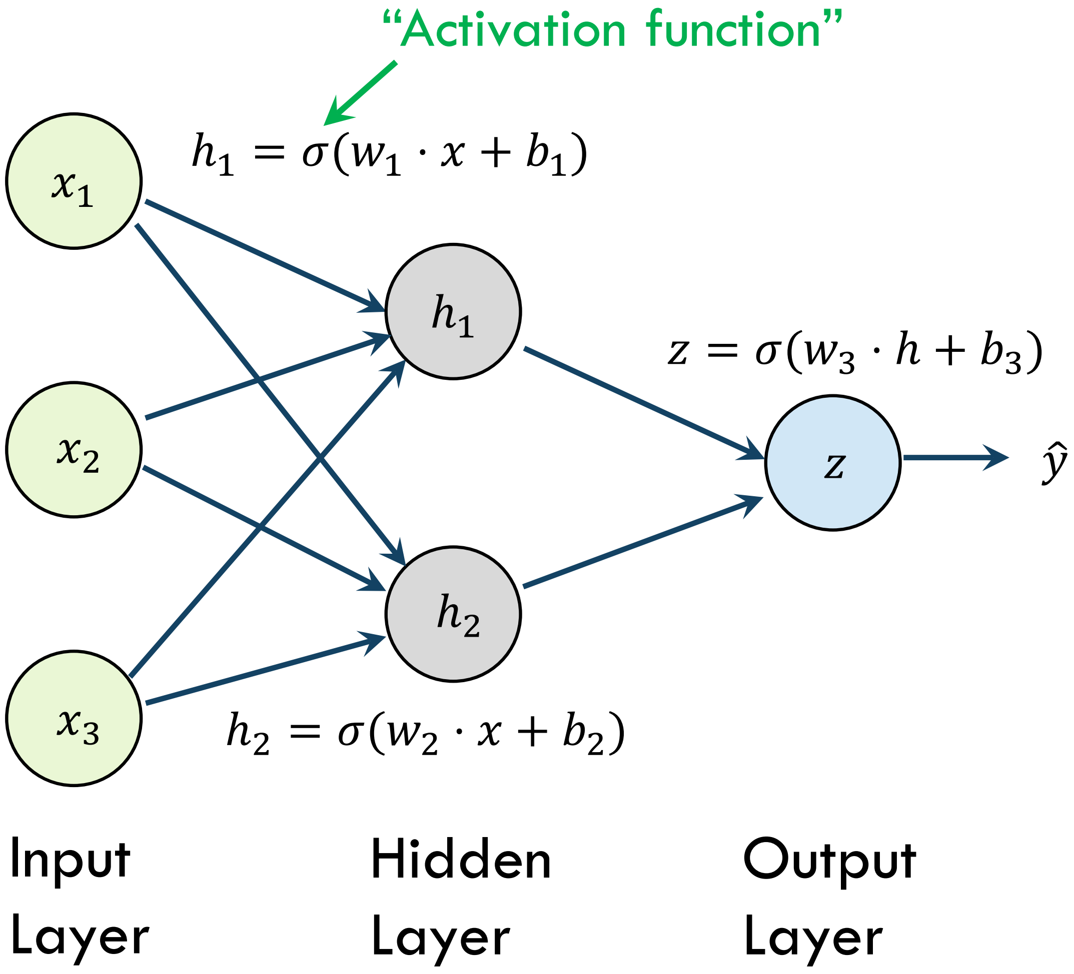




Layers of neurons: each layer computes a function of the previous layer

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

DEEPER NEURAL NETWORKS



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

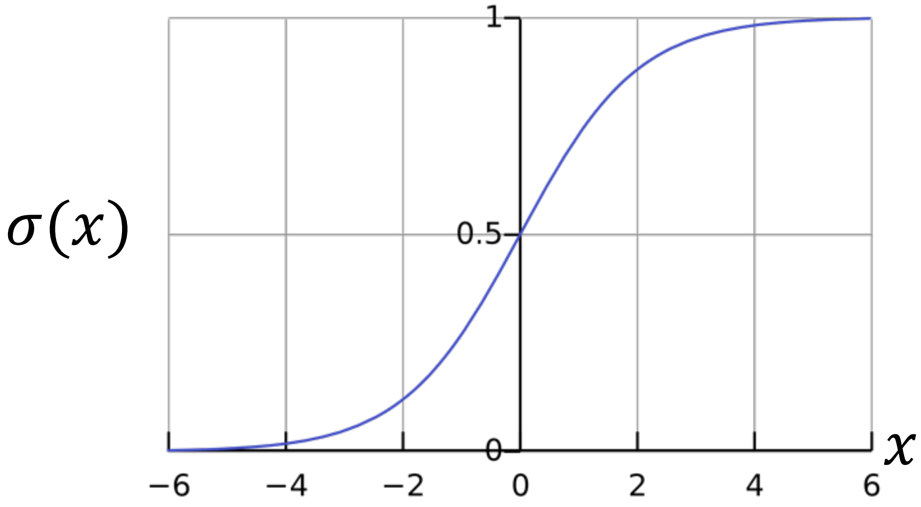
These parameters are used to compute a linear function of the previous layer

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

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

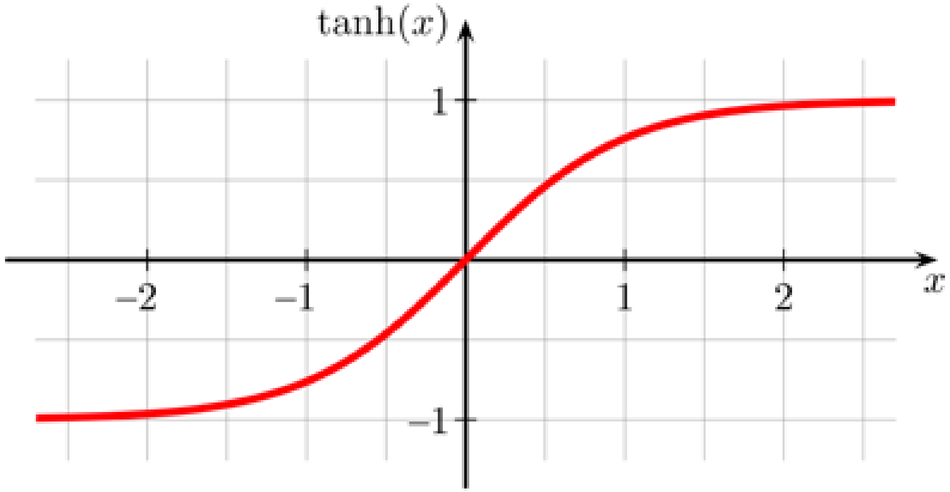
ACTIVATION FUNCTIONS

Sigmoid



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

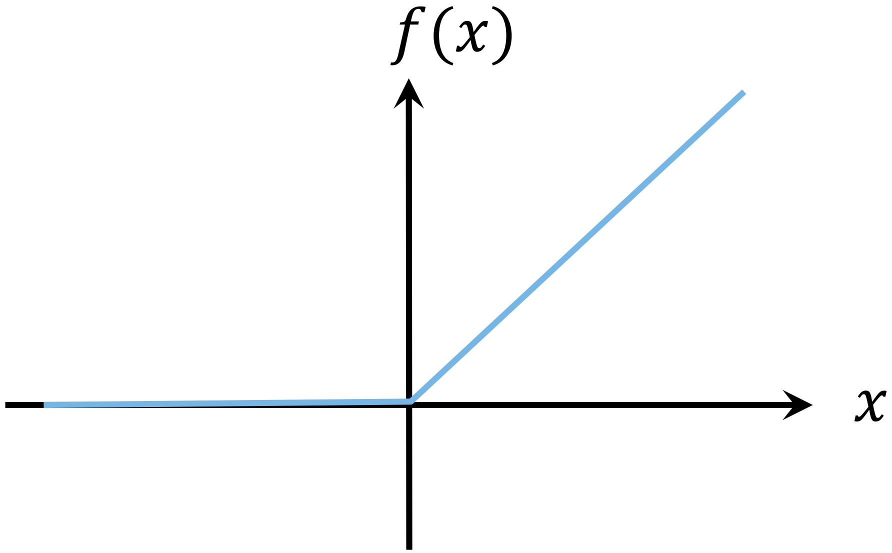
Tanh



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

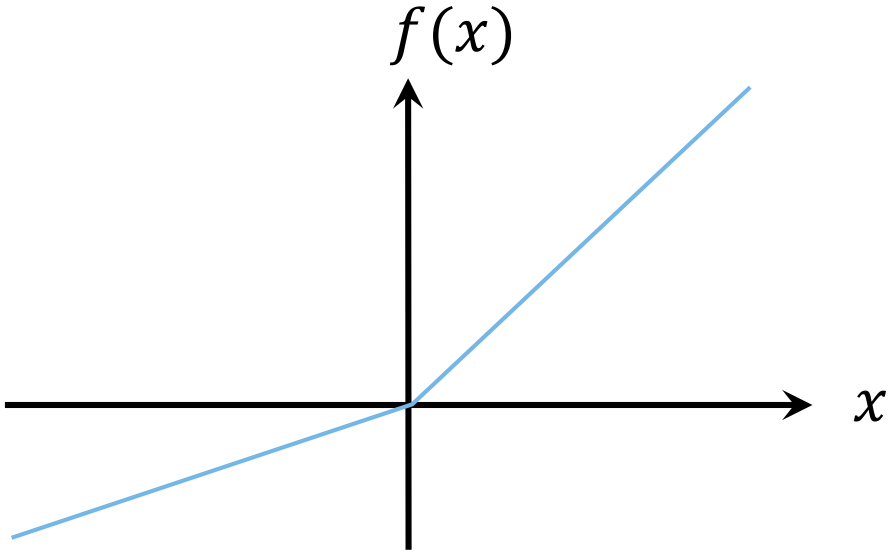
ACTIVATION FUNCTIONS

Rectified Linear Unit (ReLU)



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

Parametric ReLU



$$f(x) = \max(ax, x) \quad (0 < a < 1)$$

MANY TRICKS

Initialization

- Random, Weight initialization, Xavier initialization, …

Adaptive learning rates

- Decay, Momentum, RMSProp, Adam, …

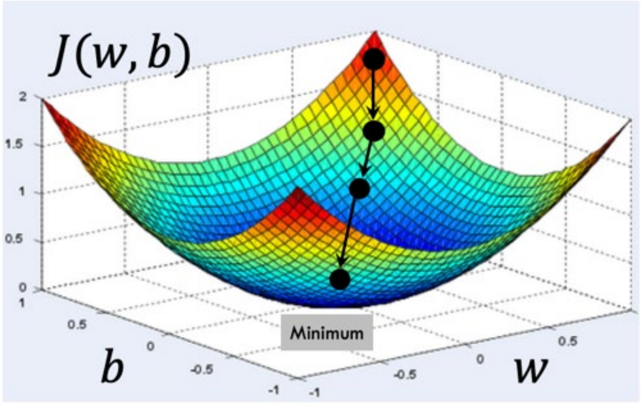
Batch normalization

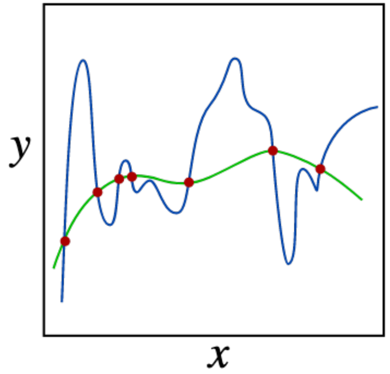
Regularization

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

Speed up learning

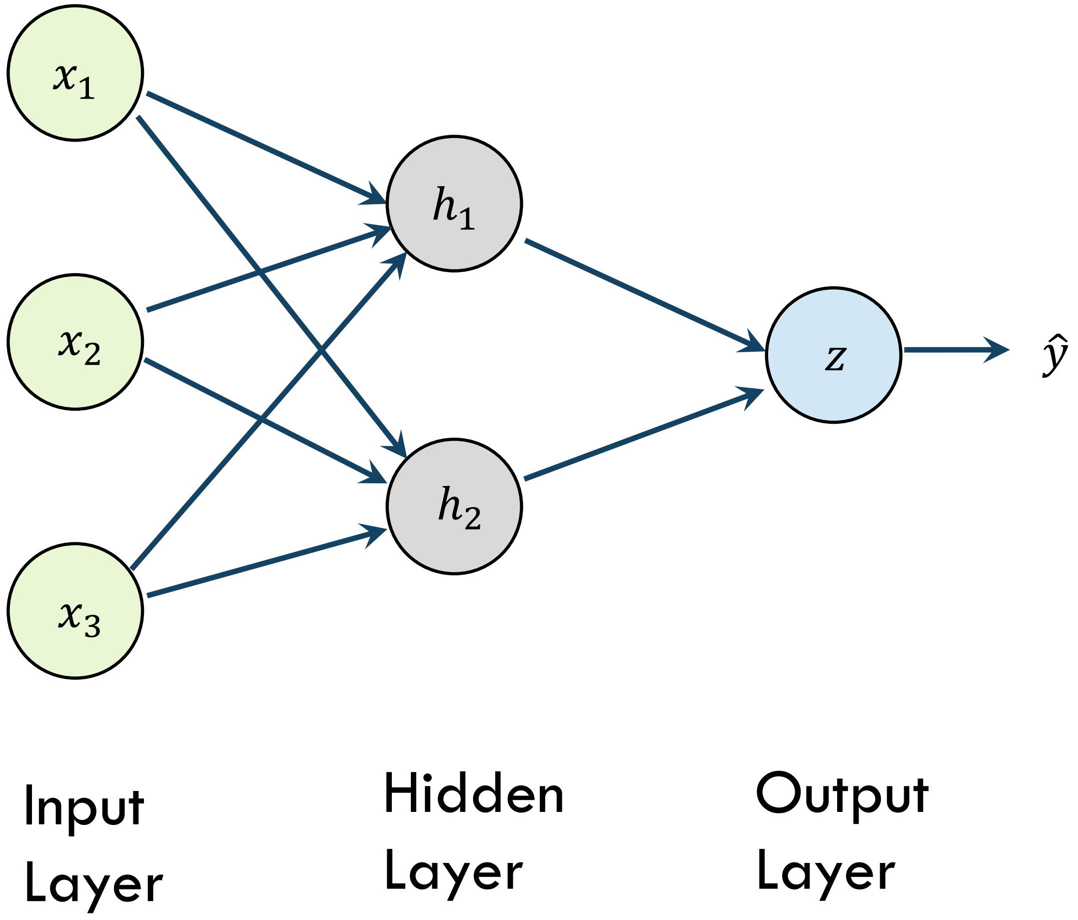
Reduce overfitting





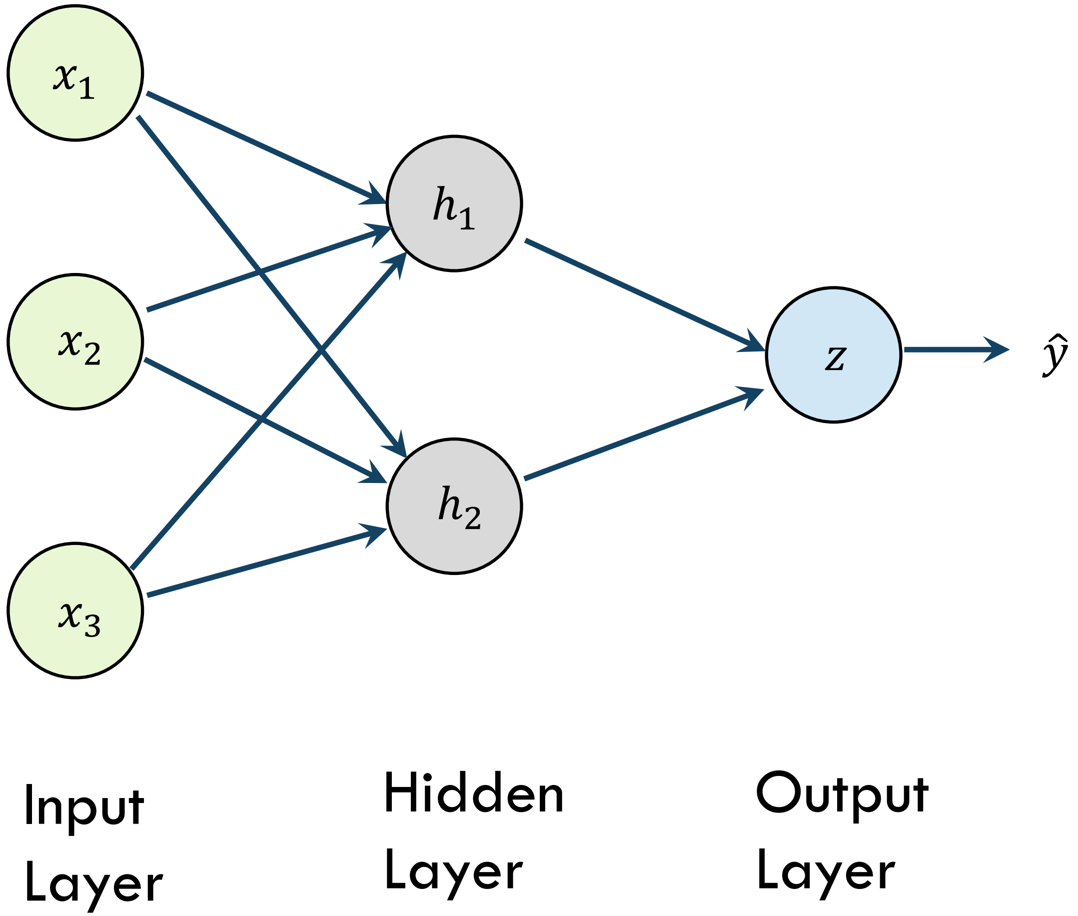
Bjorck N, Gomes CP, Selman B, Weinberger KQ. Understanding batch normalization. NeurIPS 2018

QUIZ: HOW MANY PARAMETERS?



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

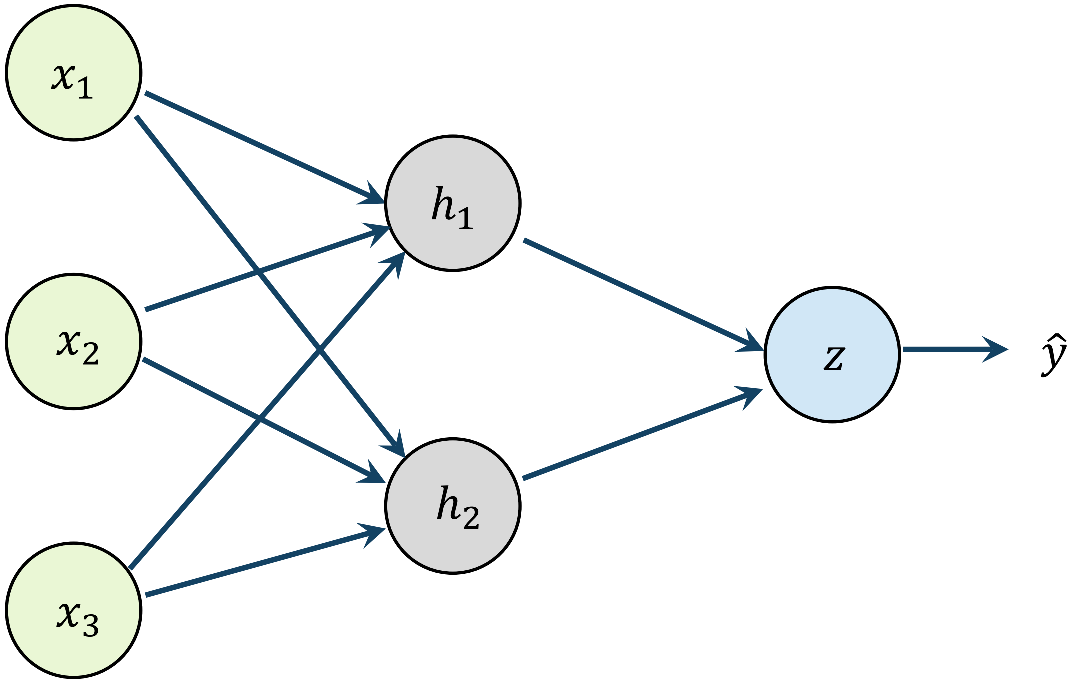
QUIZ: HOW MANY PARAMETERS?



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?



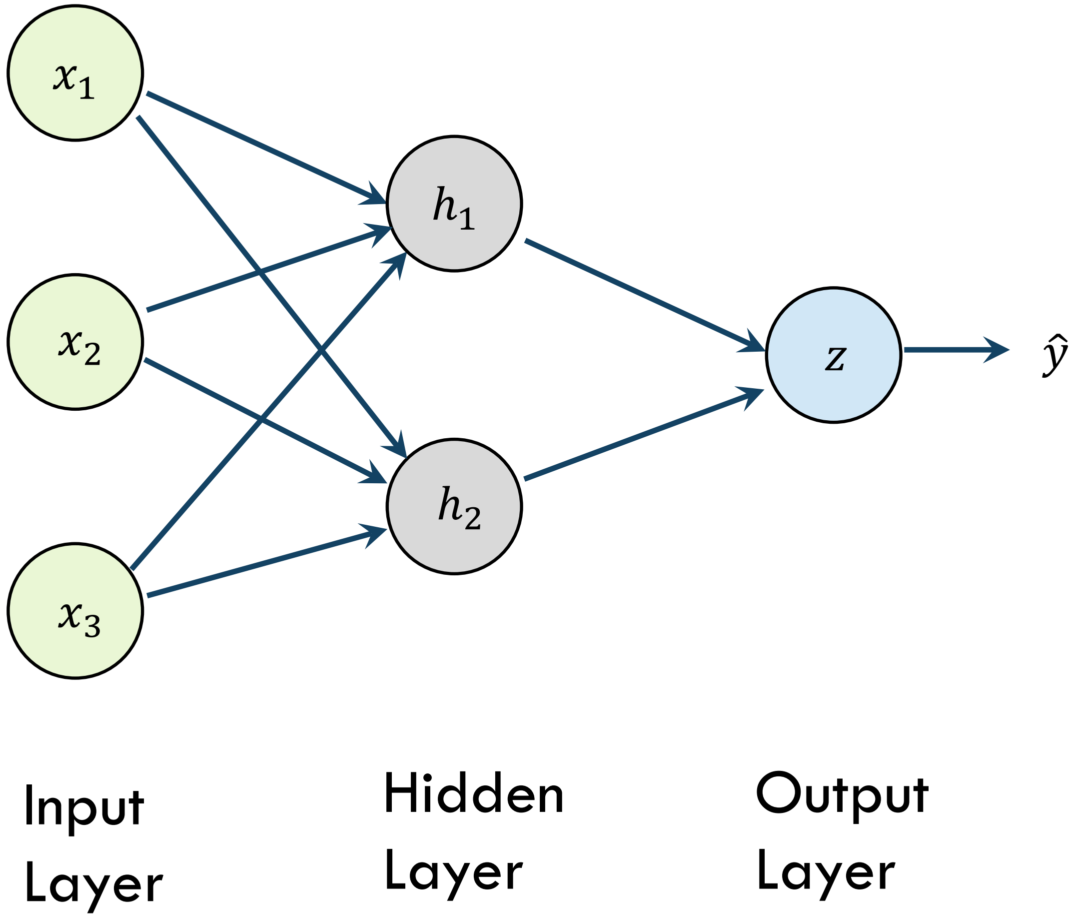
Input Layer

Hidden Layer

Output Layer

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

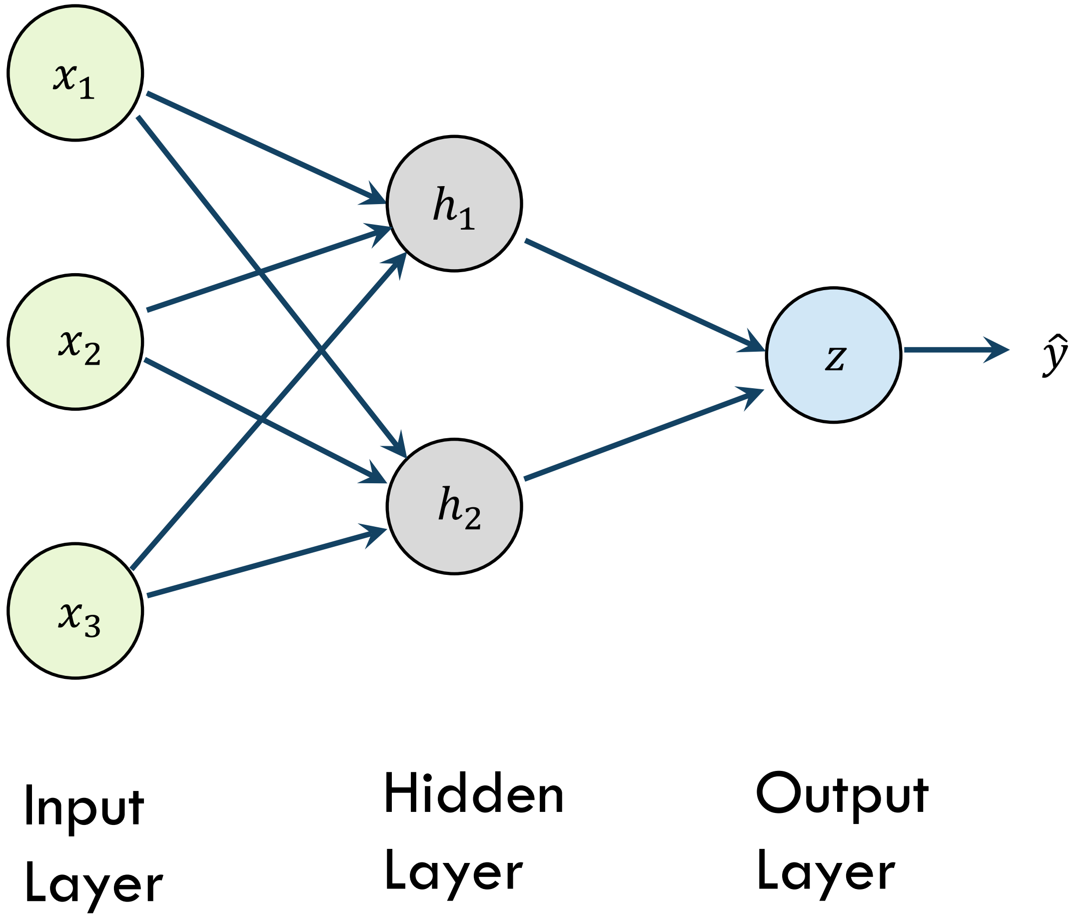
QUIZ: HOW MANY PARAMETERS?



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?



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