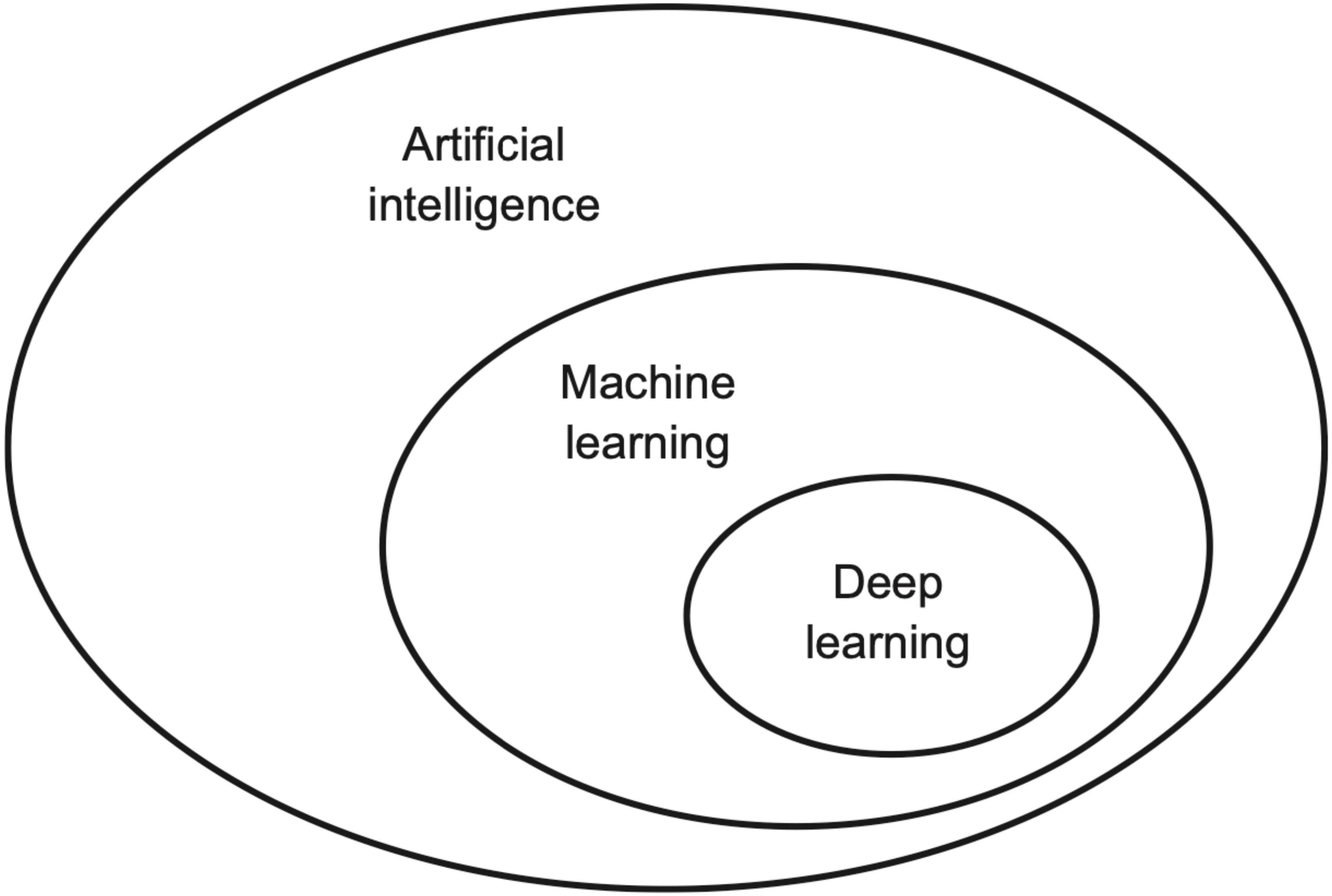
What is deep learning?

Harvey Alférez, Ph.D.

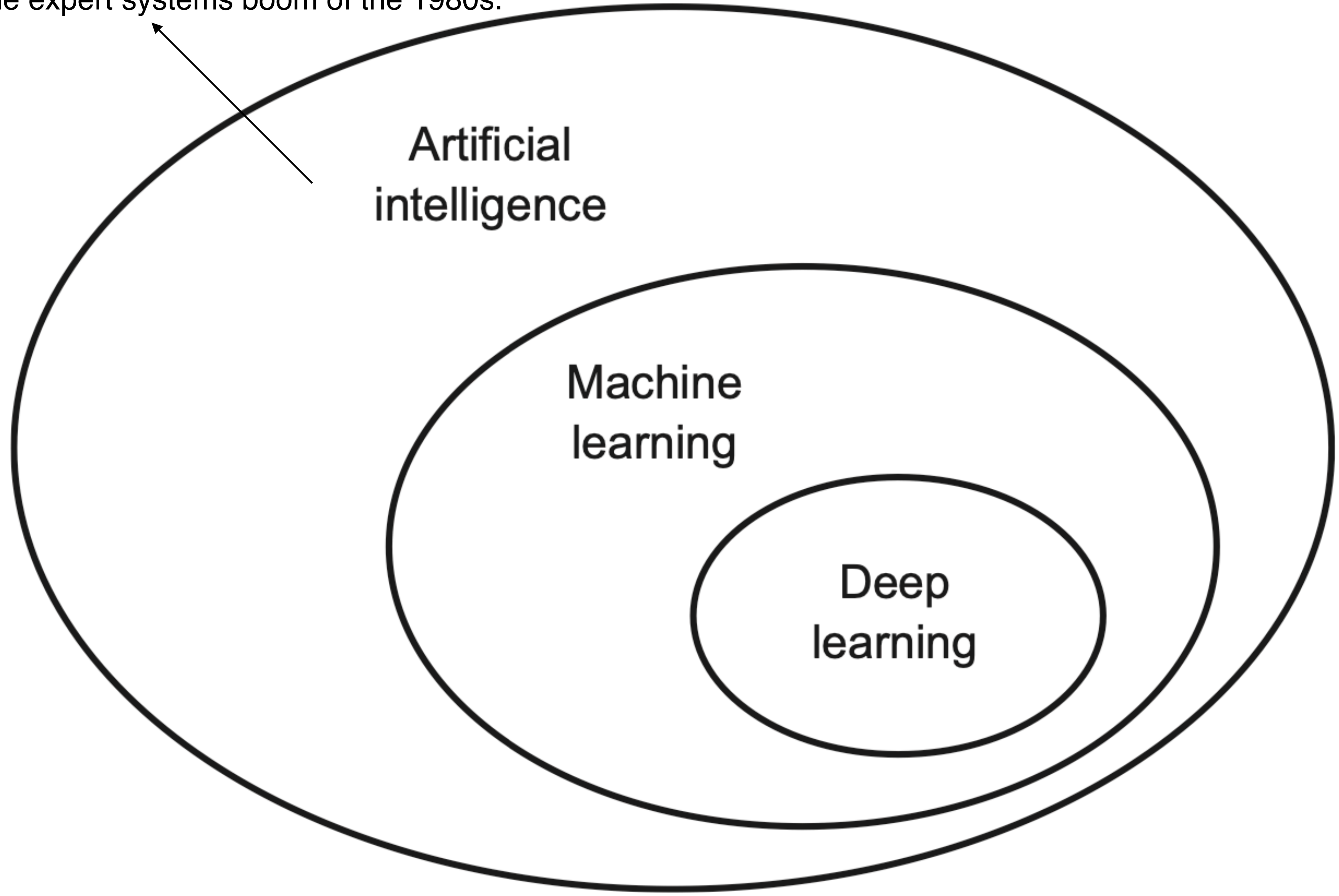
The AI Universe



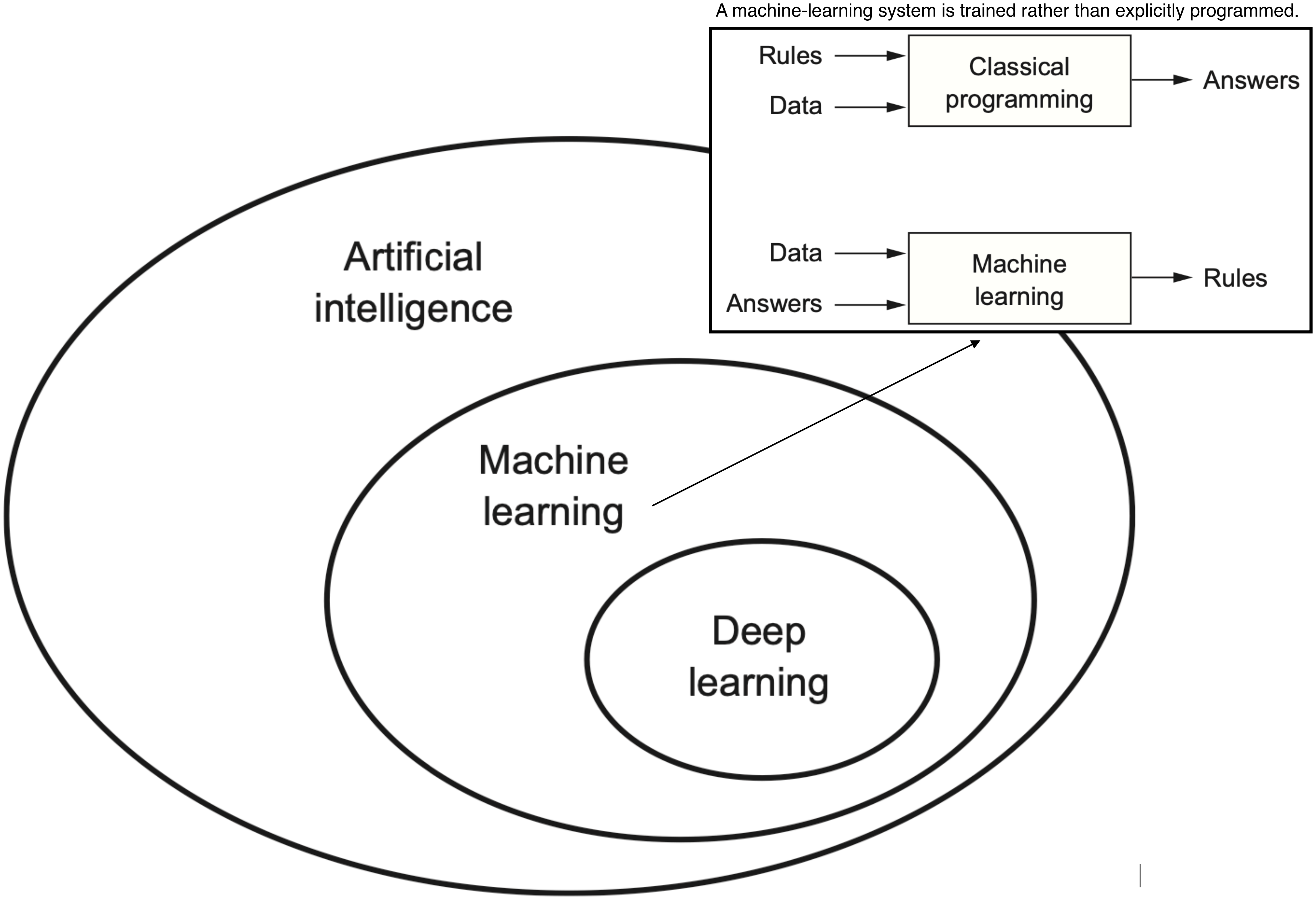
The AI Universe

The effort to automate intellectual tasks normally performed by humans, e.g., early chess programs, for instance, only involved hardcoded rules crafted by programmers, and didn't qualify as machine learning.

This approach is known as symbolic AI. Dominant paradigm in AI from the 1950s to the late 1980s. Peak popularity during the expert systems boom of the 1980s.



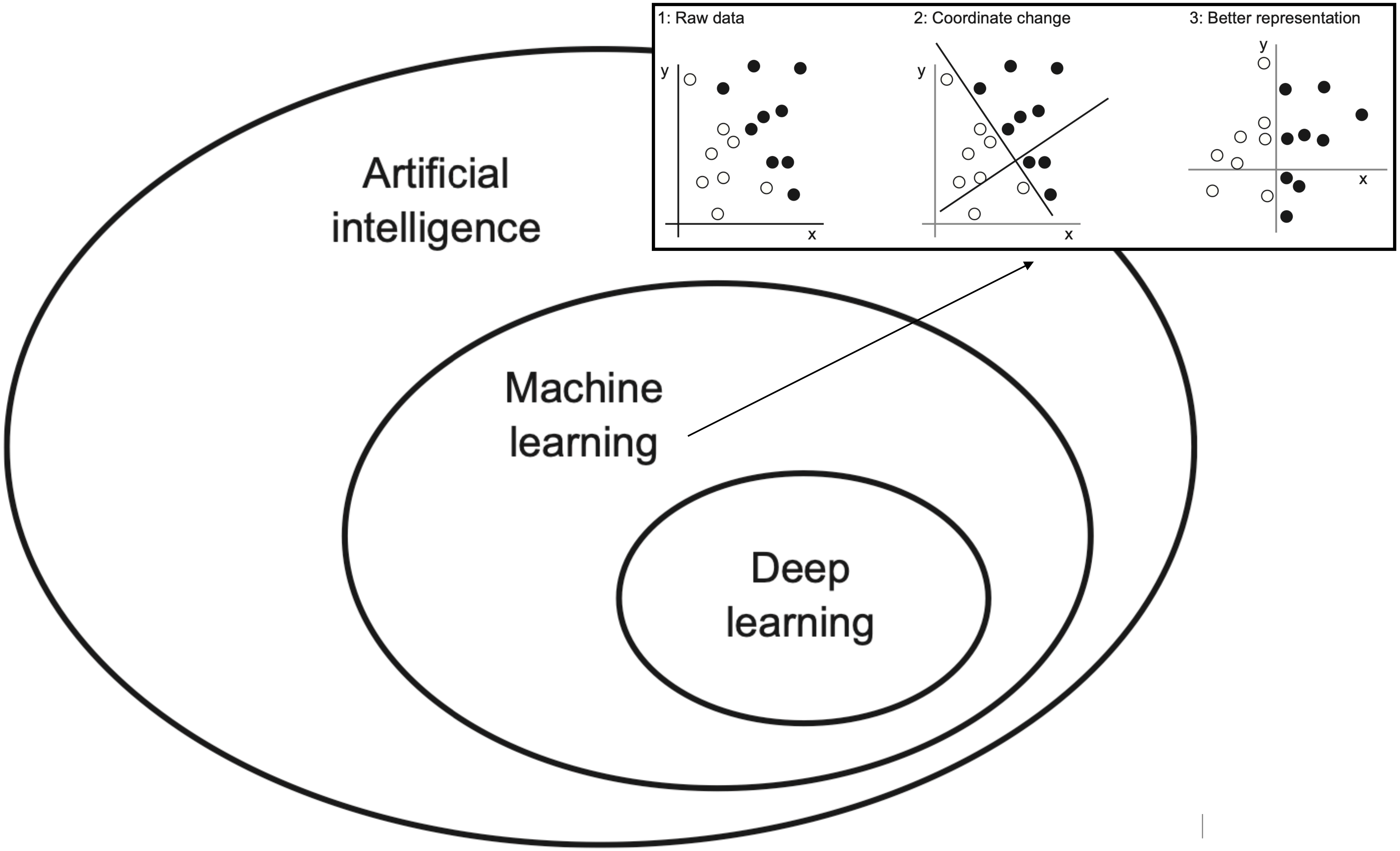
The AI Universe



The AI Universe

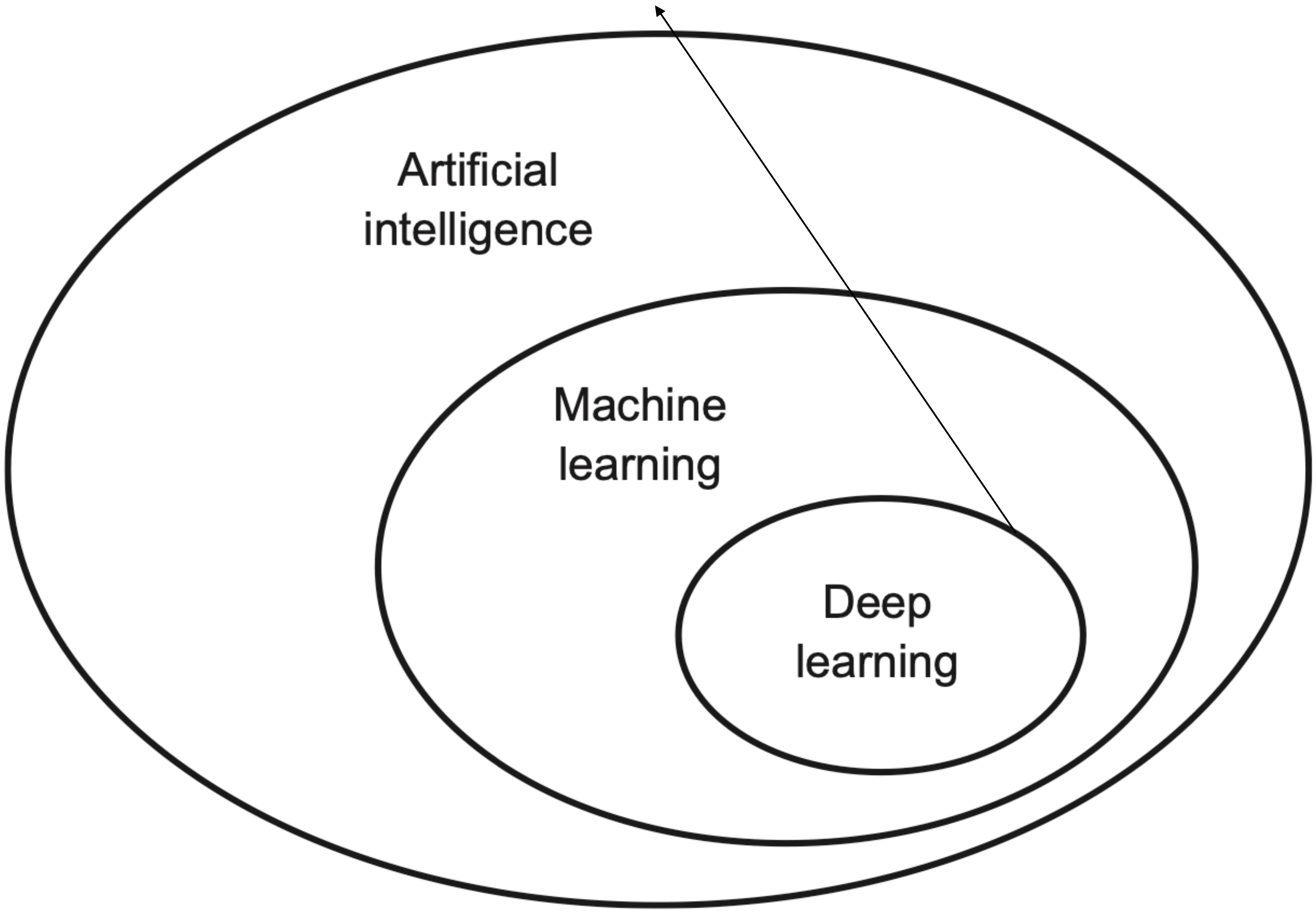
Learning, in the context of machine learning, describes an automatic search process for better representations.

“Black points are such that x > 0,” or “White points are such that x < 0.”

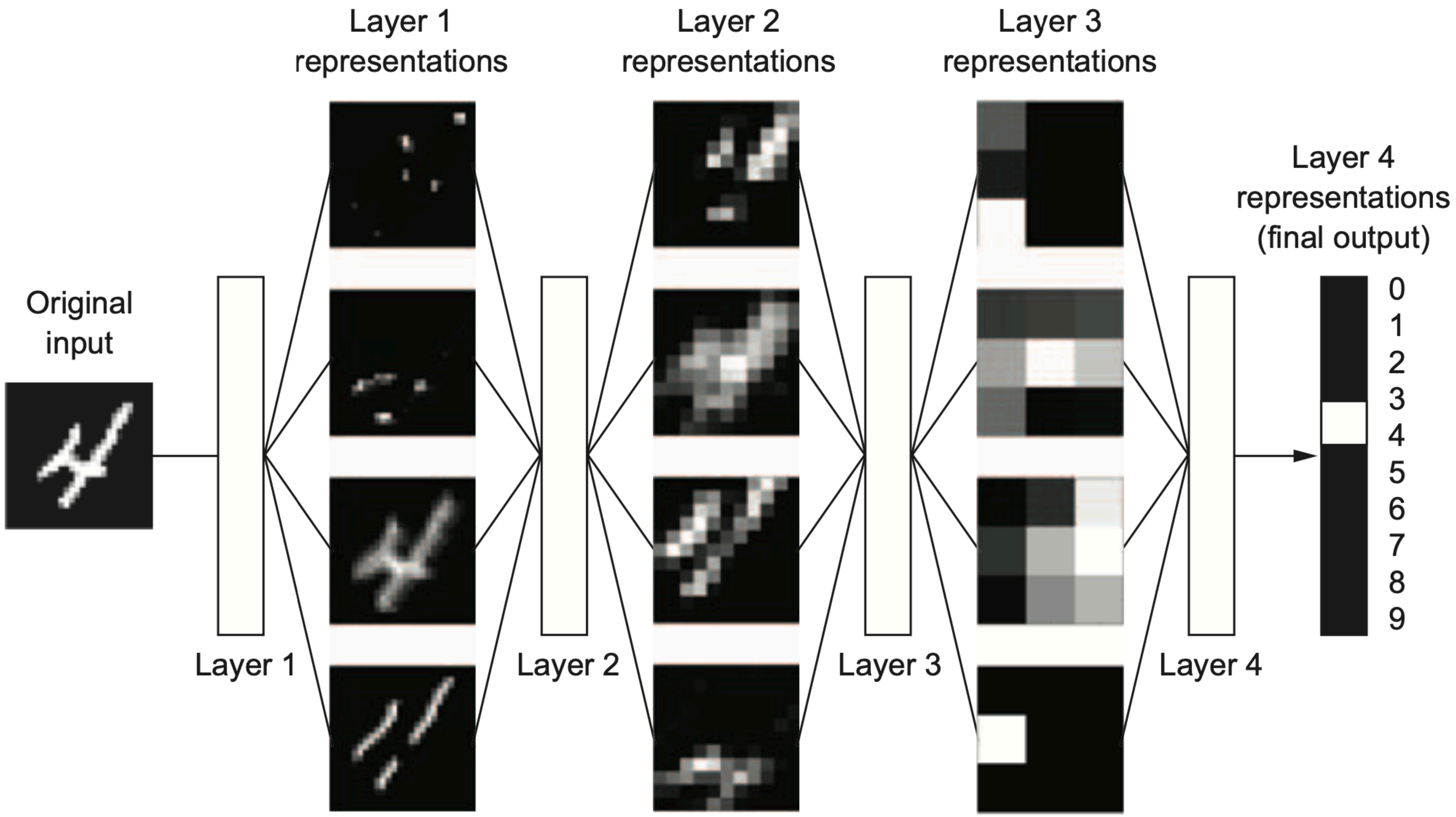


The AI Universe

The deep in deep learning stands for this idea of successive layers of representations.



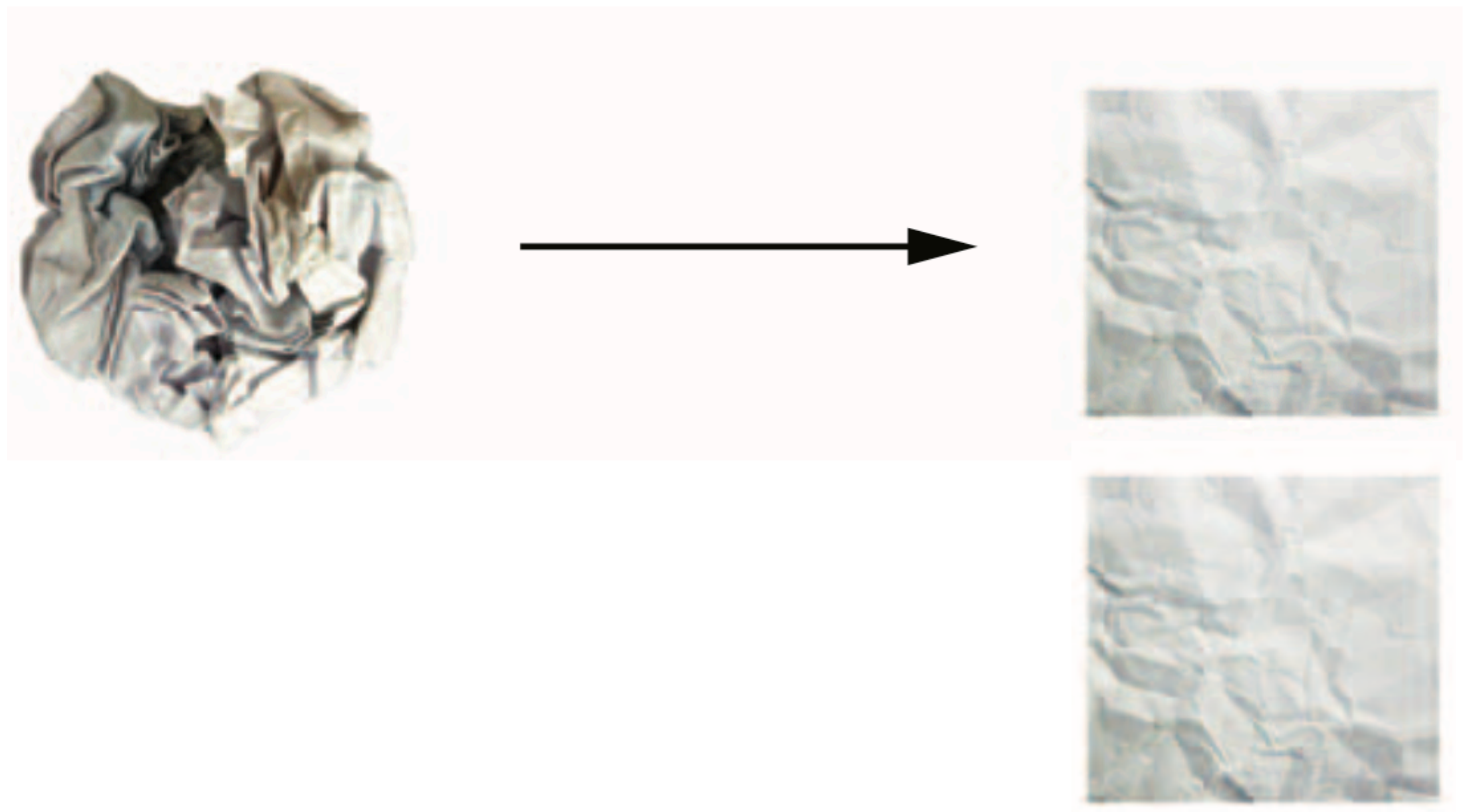
Deep Learning Example



You can think of a deep network as a multistage information-distillation operation, where information goes through successive filters and comes out increasingly purified.

Deep Learning Logic

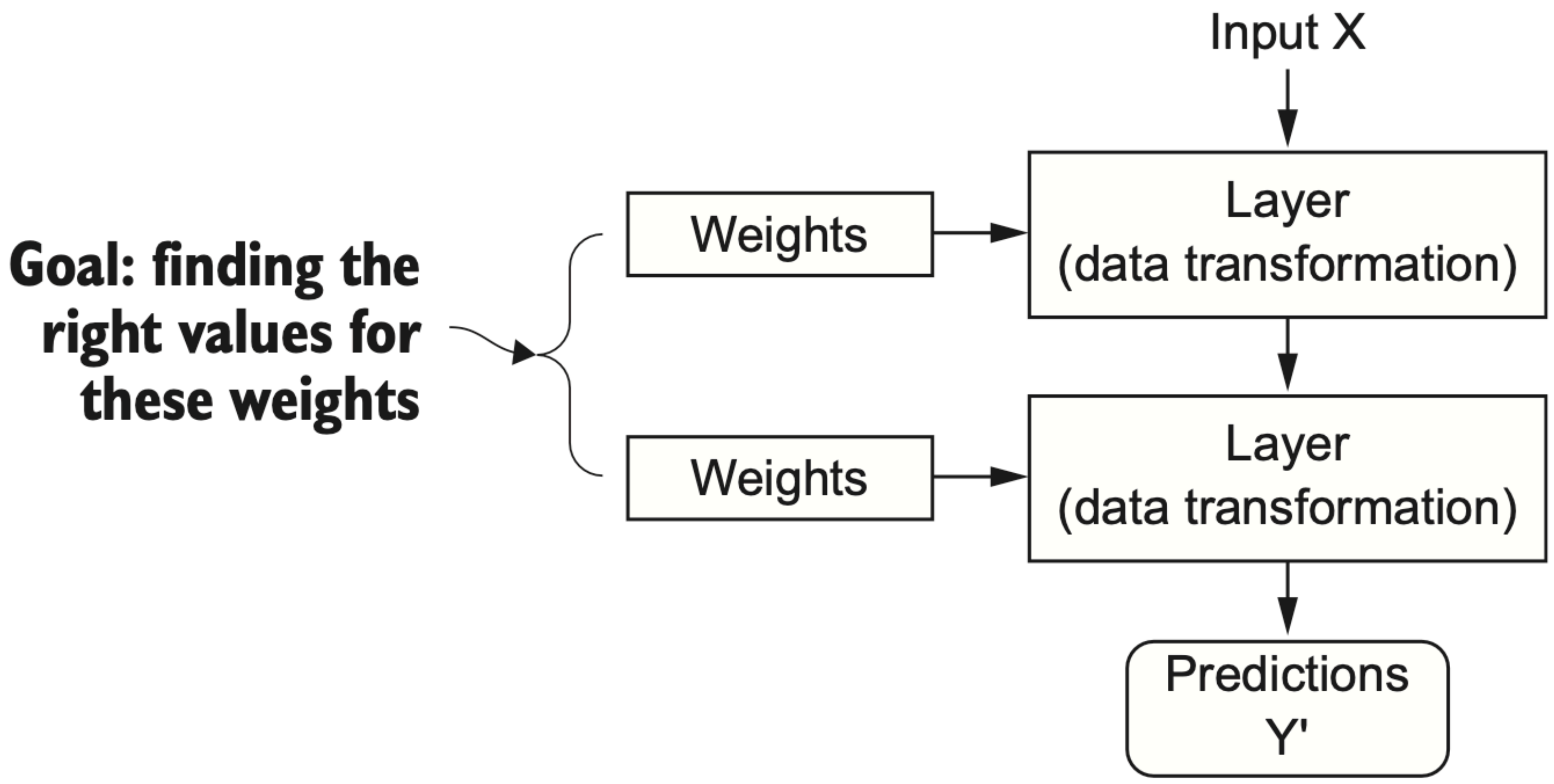
Uncrumpling a complicated manifold of data



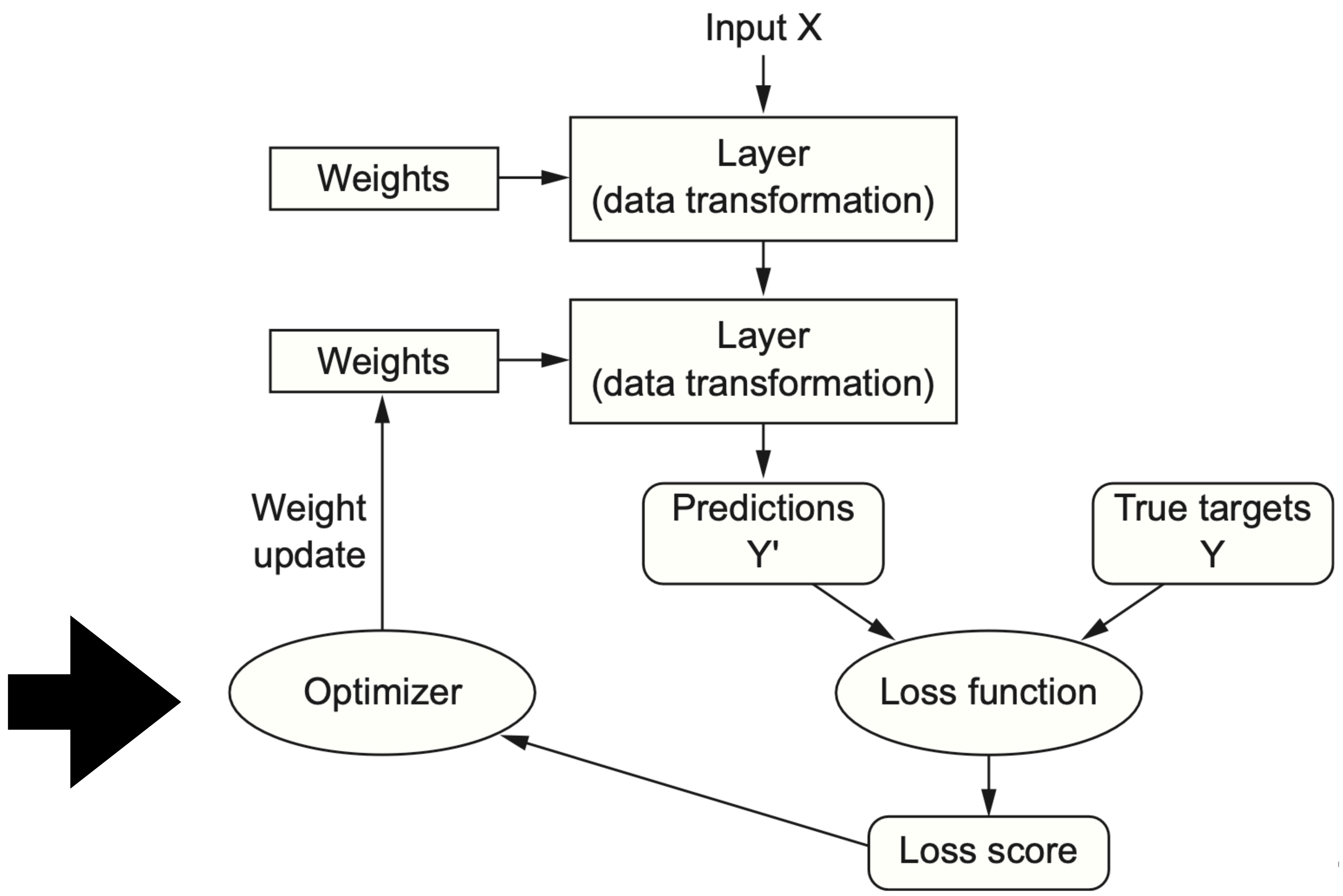
Why Deep Learning? Why Now?

- Hardware
- Datasets and benchmarks
- Algorithmic advances

A Neural Network is Parameterized by Its Weights


Input X
Goal: finding the right values for these weights
Weights
Layer (data transformation)
Weights
Layer (data transformation)
Predictions Y'

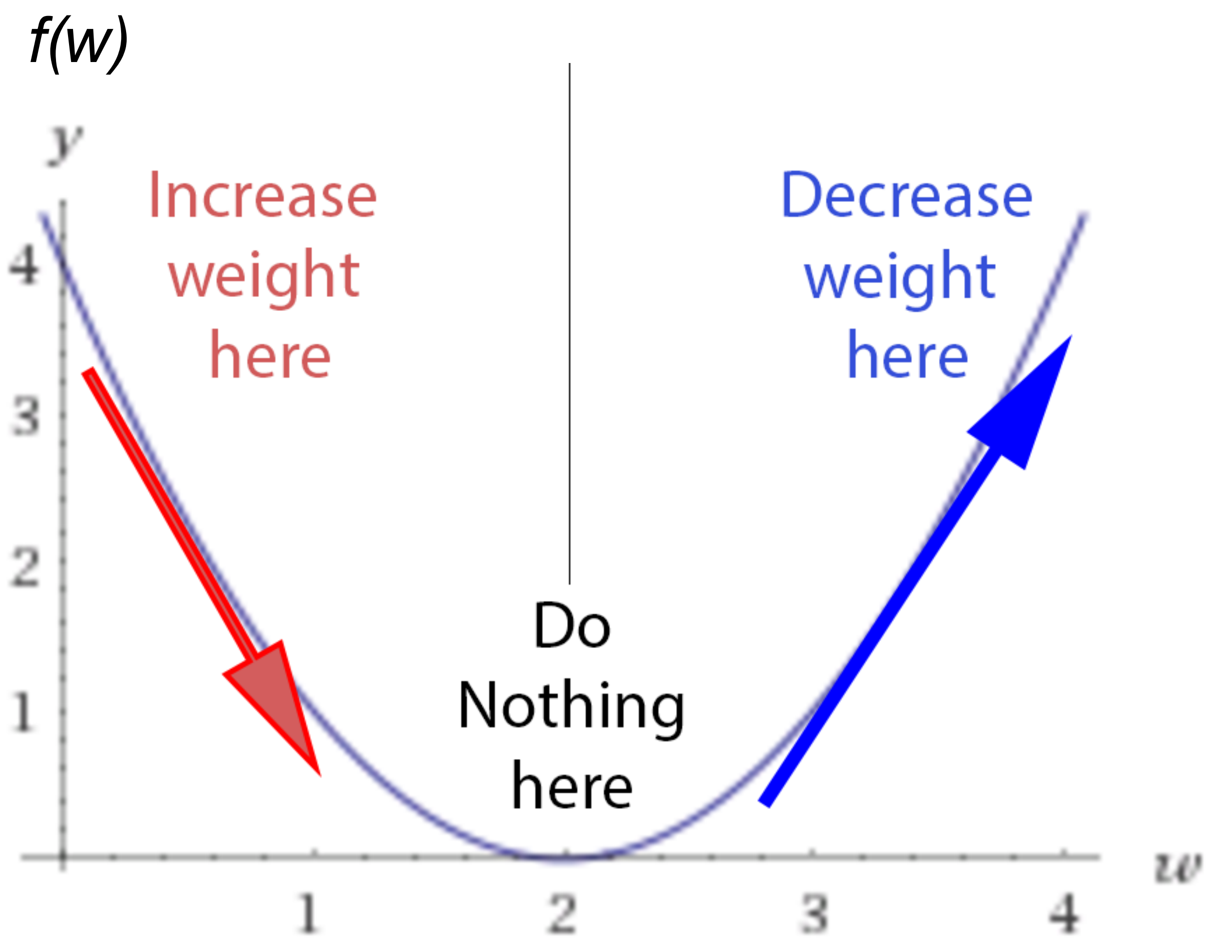
The Loss Scored is Used as a Feedback Signal to Adjust the Weights



Mini-Batch Stochastic Gradient Descent

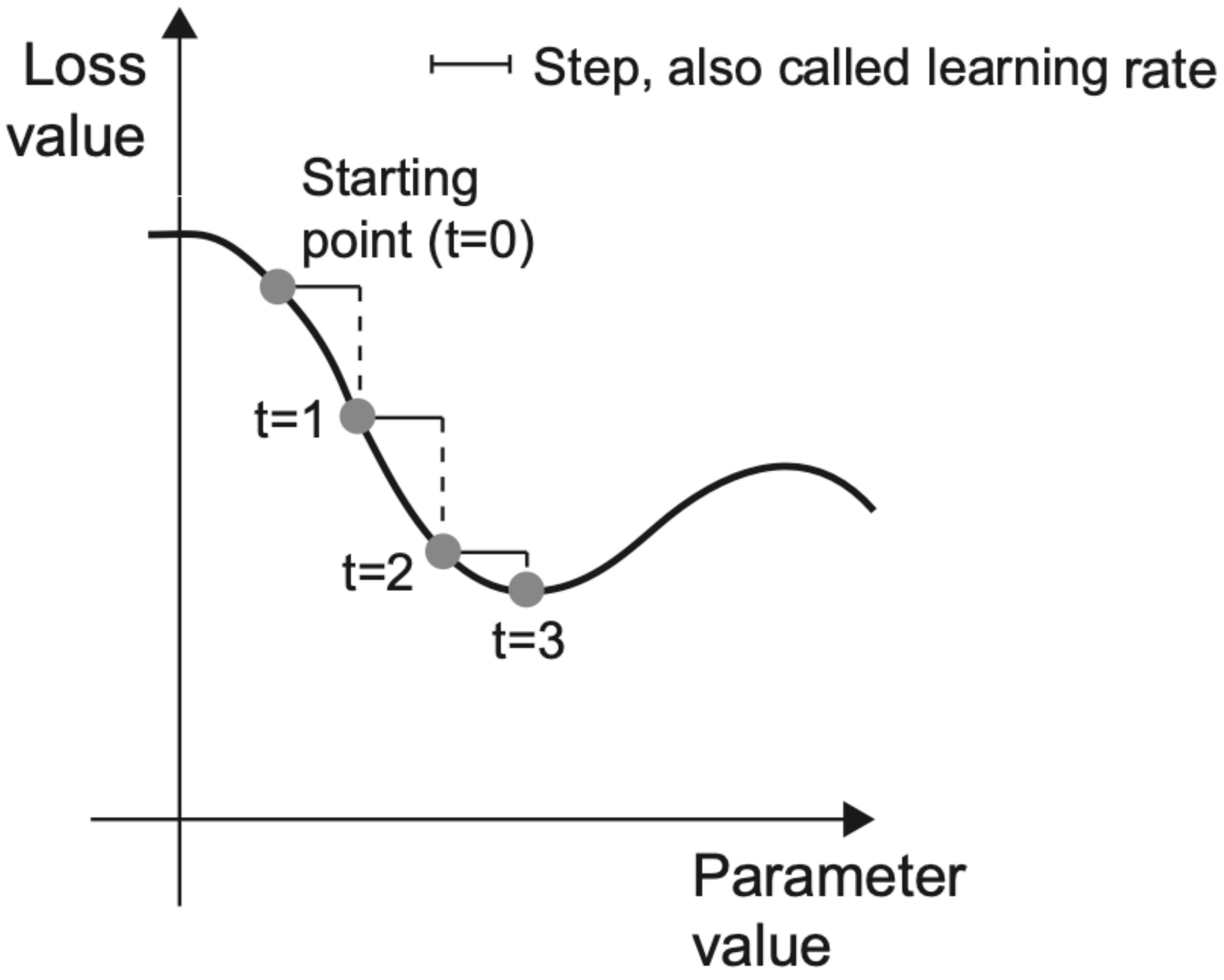
1. Draw a batch of training samples x and corresponding targets y.
2. Run the network on x to obtain predictions y_pred.
3. Compute the loss of the network on the batch, a measure of the mismatch between y_pred and y.
4. Compute the gradient of the loss with regard to the network's parameters (a backward pass).
5. New weight = old weight—Derivative Rate * learning rate

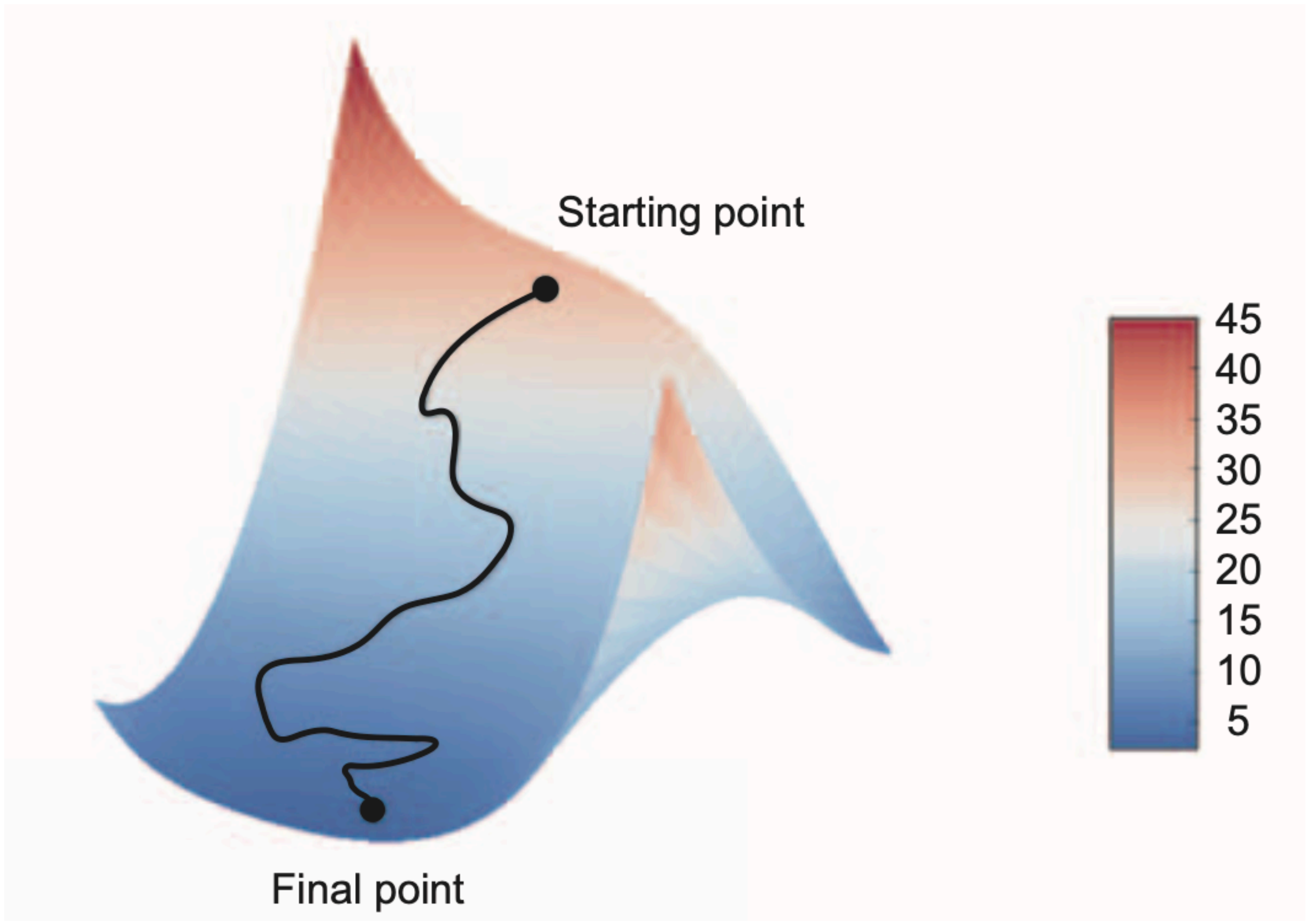
- **New weight = old weight — Derivative Rate * learning rate**
  - If the derivative rate is positive, it means that an increase in weight will increase the error, thus the new weight should be smaller.
  - If the derivative rate is negative, it means that an increase in weight will decrease the error, thus we need to increase the weights.
  - If the derivative is 0, it means that we are in a stable minimum. Thus, no update on the weights is needed -> we reached a stable state. Global cost minimum.



Derivative is negative

Derivative is positive

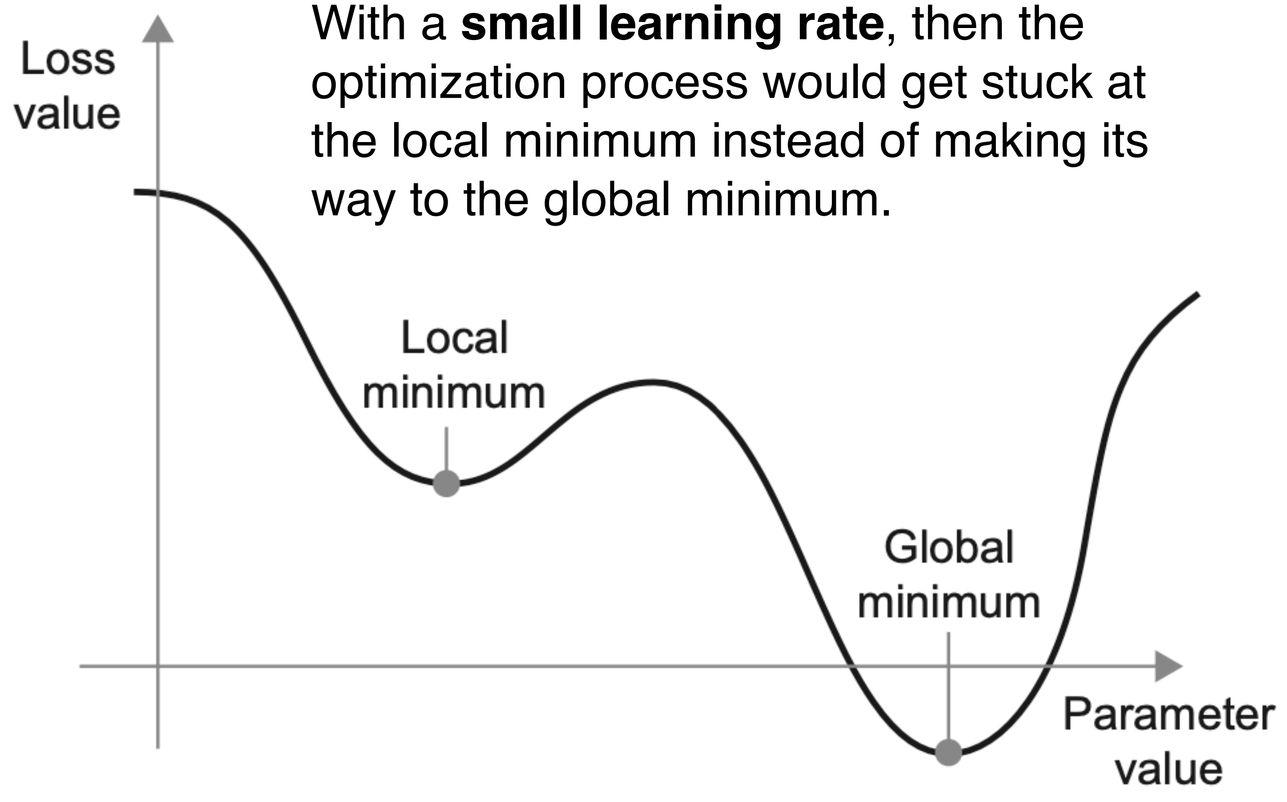

Loss value
Step, also called learning rate
Starting point (t=0)
t=1
t=2
t=3
Parameter value

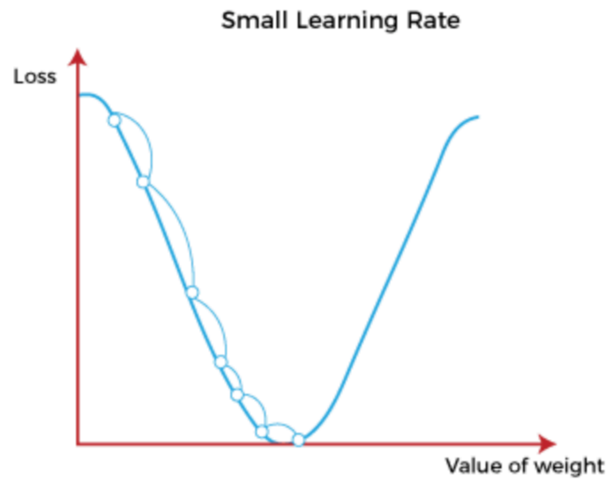


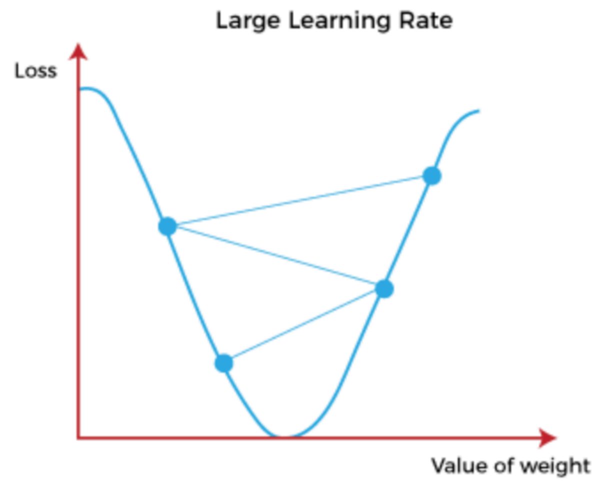
Gradient descent down a 2D loss surface (two learnable parameters)

A Local Minimum and a Global Minimum

With a **small learning rate**, then the optimization process would get stuck at the local minimum instead of making its way to the global minimum.







Momentum is implemented by moving the ball at each step based not only on the **current slope value** (current acceleration) but also on the **current velocity** (resulting from past acceleration). In practice, this means updating the parameter w based not only on the current gradient value but also on the **previous parameter update.**

A Local Minimum and a Global Minimum

```
past_velocity = 0.
momentum = 0.1
while loss > 0.01:
    w, loss, gradient = get_current_parameters()
    velocity = past_velocity * momentum + learning_rate * gradient
    w = w + momentum * velocity - learning_rate * gradient
    past_velocity = velocity
    update_parameter(w)
```

Constant momentum factor

Optimization loop

Chaining derivatives: the Backpropagation algorithm

- In practice, a neural network function consists of many tensor operations chained together, each of which has a simple, known derivative. For instance, this is a network f composed of three tensor operations, a, b, and c, with weight matrices W1, W2, and W3:
    - f(W1, W2, W3) = a(W1, b(W2, c(W3)))
- Calculus tells us that such a chain of functions can be derived using the following identity, called the chain rule: f(g(x)) = f'(g(x)) * g'(x). Applying the chain rule to the computation of the gradient values of a neural network gives rise to an algorithm called Backpropagation

Chaining derivatives: the Backpropagation algorithm

- Backpropagation starts with the final loss value and works backward from the top layers to the bottom layers, applying the chain rule to compute the contribution that each parameter had in the loss value.
- Nowadays, and for years to come, people will implement networks in modern frameworks that are capable of symbolic differentiation, such as TensorFlow.
  - Given a chain of operations with a known derivative, they can compute a gradient function for the chain (by applying the chain rule) that maps network parameter values to gradient values.
  - When you have access to such a function, the backward pass is reduced to a call to this gradient function.
  - Thanks to symbolic differentiation, you’ll never have to implement the Backpropagation algorithm by hand!