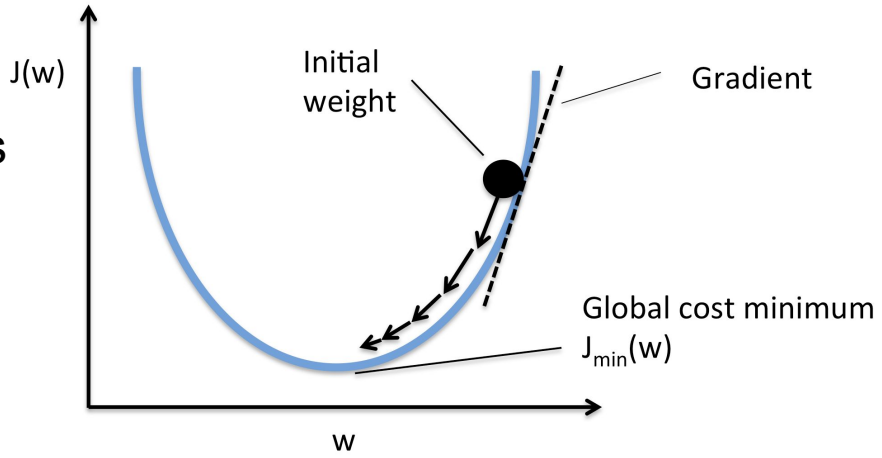
Neural Networks

BOSTON UNIVERSITY
MACHINE INTELLIGENCE
COMMUNITY

Charles Ma, Rex Wang, Justin Chen
Oct. 3, 2017

Brief Recap

- Data-driven learning
- Deep learning models biological neurons
- Learn weights through backpropagation
- Cost/Loss/Objective functions
- Stochastic vs. Batch Gradient Descent
- What's a computation graph?
- What's backpropagation?



https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

A few things neural nets can do

Optical character recognition
LeNet5 1993

Atari DQN
Mnih 2014 (Deep Mind)

+

=

MACHINE INTELLIGENCE COMMUNITY

Current Limitations of Neural Networks

- Supervised learning requires lots of labeled data (10^5, 10^6,...)
- Expensive computation power
- Training time (weeks or months)
- Catastrophic Forgetting
  - Forgetting old parameters when learning new tasks and domains
- Unsupervised learning
- Model-based reinforcement learning
- Interpretability of learned representations

https://www.amazon.com/

MACHINE INTELLIGENCE COMMUNITY

What is Deep Learning?

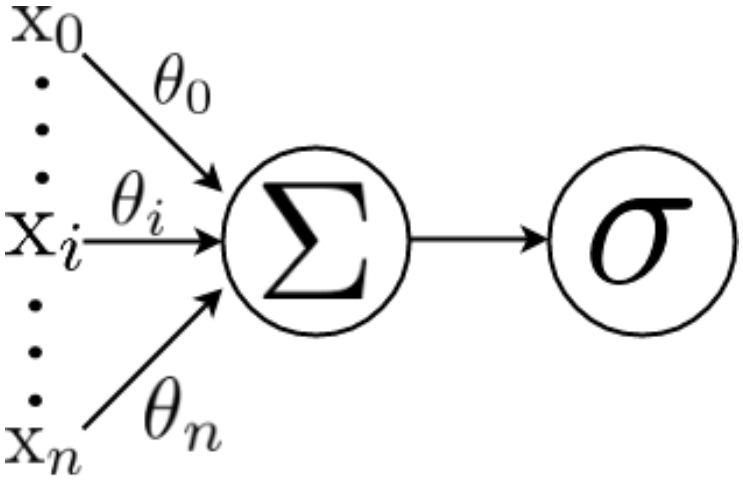
- **Artificial neuron** - single unit of computation
- **Artificial neural network** – graph of connected artificial neurons
- **Deep neural network** – artificial neural network with more than 2 hidden layers
- Deeply nested composite functions for approximating functions from data
- **Goal**: learn an approximate function that can **generalize to new data**

$$f \circ g \circ \ldots$$

MACHINE INTELLIGENCE COMMUNITY

Artificial Neurons

- Used to model nonlinear data
  - Nonlinear data cannot be approximated with simple linear equation (e.g. y = mx + b)
- A neuron is a linear equation composed into a nonlinear one (like sigmoid)



$$\sigma(\theta_0 x_0 + ... + \theta_n x_n)$$

MACHINE INTELLIGENCE COMMUNITY

Artificial Neuron in PYTORCH

Create a linear equation

```
>> import torch.nn as nn
>> an = nn.Linear(4,1)
>> an

Linear (4 -> 1)
```

Weights and bias of neuron

```
>> [x.data for x in an.parameters()]
```

Weights

```
[
 0.0245  0.1270  0.1709 -0.3200
[torch.FloatTensor of size 1x4]
,
```

Bias

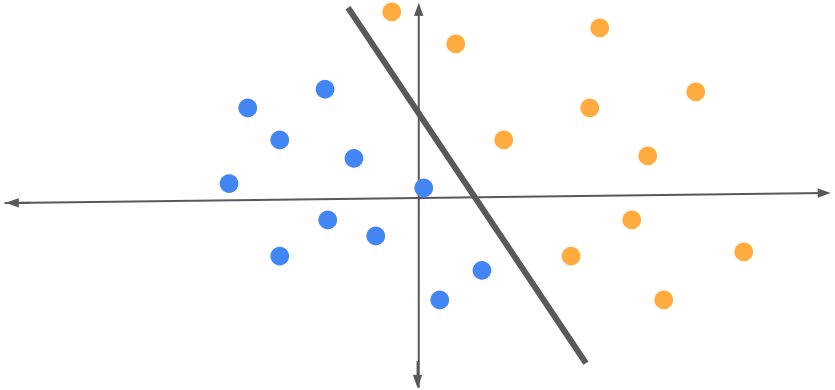
```
1.00000e-02 *
 -8.3333
[torch.FloatTensor of size 1]
]
```

MACHINE INTELLIGENCE COMMUNITY

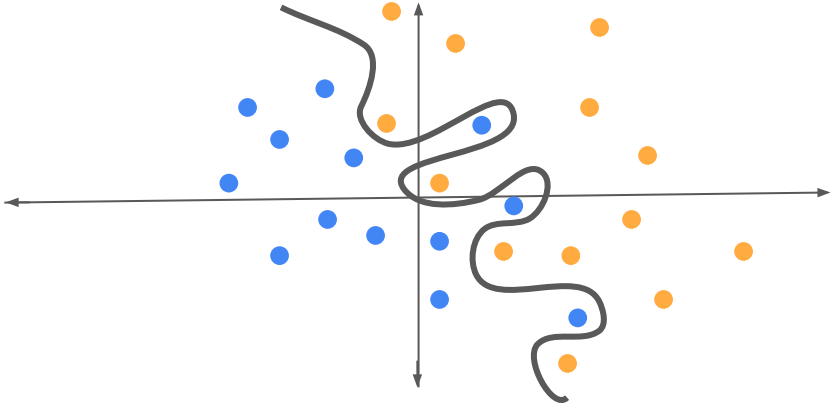
Nonlinearity

- Pass sum of all inputs (x in equations below) into a activation function
- Draw nonlinear **decision boundary**
  - Line in 2D
  - **Hyperplane** in dimensions greater than 2D
- Handle **nonlinear data**

Linear Boundary



Nonlinear Boundary

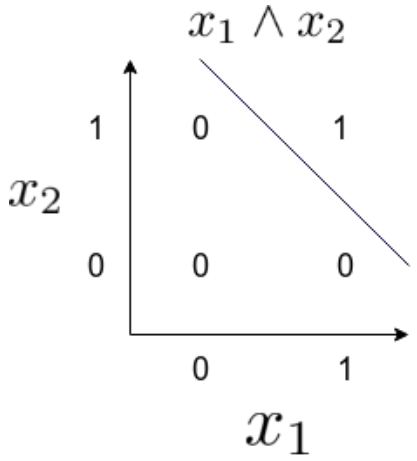


MACHINE INTELLIGENCE COMMUNITY

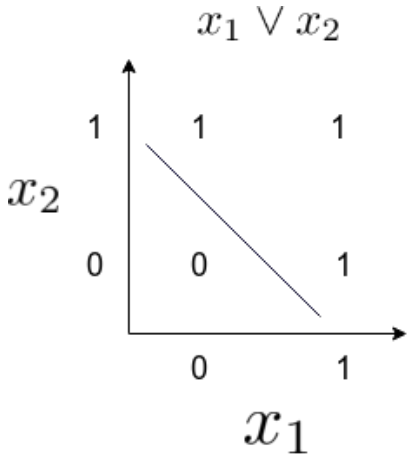
Logical XOR

- Used for simple logical expressions
  - Simple circuits/networks could not compute XOR, leading to AI Winter
- Logical AND and OR are linearly separable
  - You can draw a linear boundary between classes
    - With 2 dimensions, boundary is a line; with >2 dimensions, it is a hyperplane

Logical AND:



Logical OR:

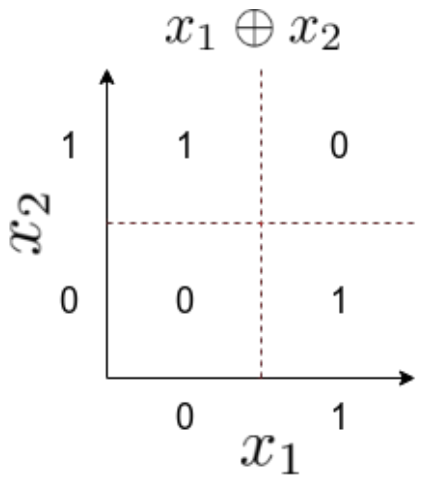


MACHINE INTELLIGENCE COMMUNITY

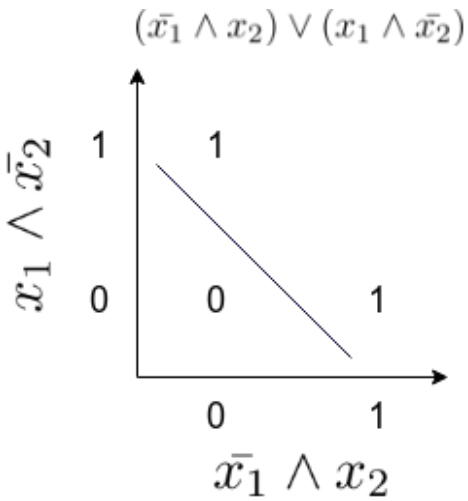
Logical XOR cont.

- XOR is not linearly separable in two dimensions
  - Simple set of linear equations composed into nonlinear function cannot compute XOR
  - Can be separated if transformed through different dimensions

Not linearly separable:



Linearly Separable:

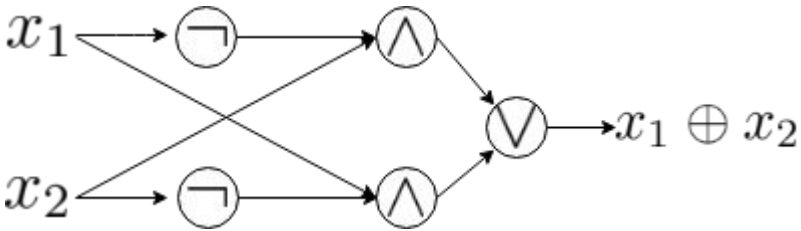


MACHINE INTELLIGENCE COMMUNITY

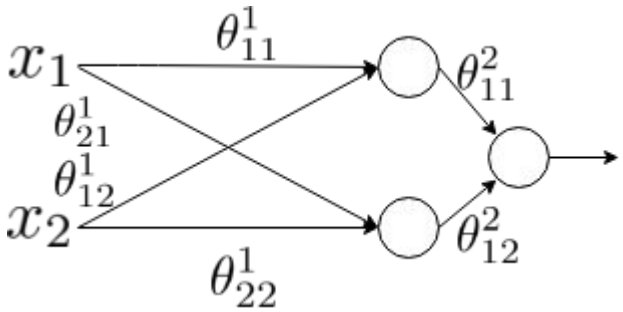
Logical XOR cont.

- Solution: Compose expression that can compute XOR and separate classes
  - Causes zero class to collapse to left of boundary

XOR Logical Circuit



XOR Computation Graph



MACHINE INTELLIGENCE COMMUNITY

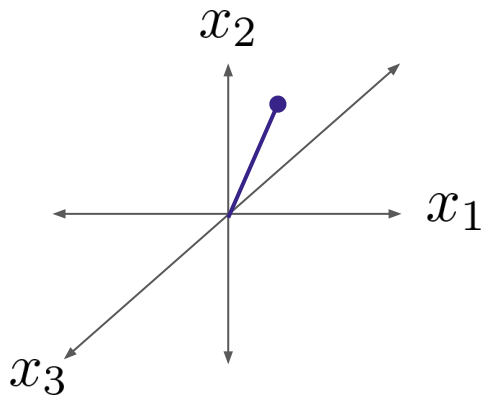
High Dimensional and Nonlinear Data

- Simple linear functions can’t model nonlinear data
- e.g. XOR because it is higher-order function
- e.g. Space of all natural images of doggos

https://www.facebook.com/photosbyjaq/

Dimensionality and Vectorization

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} =$$



1,365

2,048

$$\rightarrow \begin{bmatrix} x_1 \\ \vdots \\ x_{2,795,520} \end{bmatrix} \rightarrow \mathbb{R}^{2,795,520}$$

https://www.facebook.com/photosbyjaq/

MACHINE INTELLIGENCE COMMUNITY

Vectorization in PYTORCH

Create a random 2x2 image

```
>> import torch
>> image = torch.randn(2,2)
>> image

 0.0044 -0.7816
 0.6725  0.0016
[torch.FloatTensor of size 2x2]
```

Vectorize image

```
>> image.view(1,4)

 0.0044 -0.7816  0.6725  0.0016
[torch.FloatTensor of size 1x4]
```

MACHINE INTELLIGENCE COMMUNITY

Activation Functions: Overview

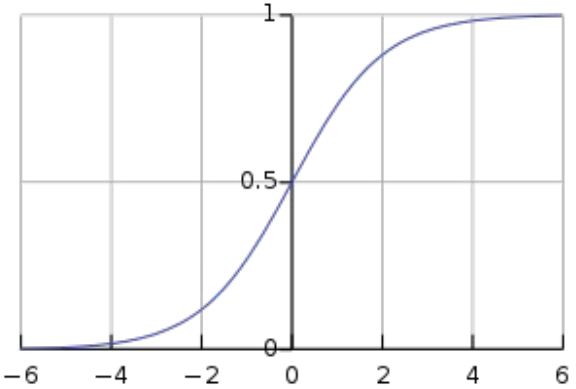
- Used to keep, map, and represent features of the input data that linear models fail to
  - Determine activation functions with rate code interpretation on firing rate of the neurons
- **Nonlinearity**, **differentiability**, and **monotonicity**; they matter for the particular classification task used used for activation function choice
  - Use these to make sure initialization is done correctly, as training depends on the output ranges of the activation function chosen

Activation Functions: Logistic Sigmoid

- Takes a number and “squashes” it into range [0, 1]
  - Large negatives approach 0 and large positives approach 1
- Good for interpreting firing rate of a neuron, but:
  - Saturates and kills gradient
    - Makes you be more careful when initializing weights
  - Output is not zero-centered
    - Affects efficiency of gradient descent

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

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$



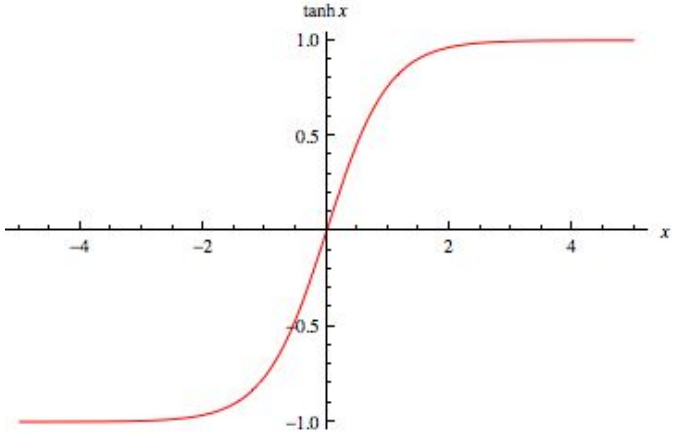
MACHINE INTELLIGENCE COMMUNITY

Activation Functions: Hyperbolic Tangent (TanH)

- Takes a number and “squashes” it into range [-1, 1]
  - Large negatives approach -1, large positives approach 1
- Converges quicker than sigmoid function, and has better accuracy, but:
  - ReLU trains much faster
  - ReLU has same training error
  - Overshadowed by ReLU

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} = 2\sigma(2x) - 1$$

$$tanh'(x) = 1 - tanh^2(x)$$



http://mathworld.wolfram.com/images/interactive/TanhReal.gif

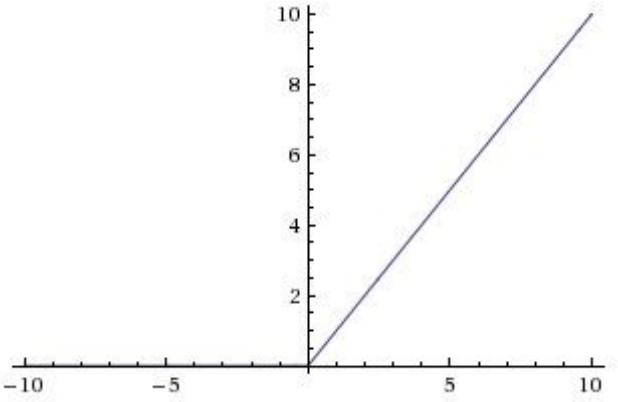
MACHINE INTELLIGENCE COMMUNITY

Activation Function: Rectified Linear Unit (ReLU)

- Takes a number and either floors it at 0 or leaves it be
  - All negatives or 0 become 0, and positives retain their value
- Reduces chance for vanishing gradient problem, and increases data sparsity
  - Sparsity of data allows for information disentangling, easier linear separability, among others
  - Also trains much faster because of simplicity of the function
  - However, can cause network to “die”. Not differentiable at point (0, 0).

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

$$f'(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x > 0 \end{cases}$$



MACHINE INTELLIGENCE COMMUNITY

ReLU Function in PYTORCH

Apply Sigmoid pointwise

```
>> import torch.nn.functional as F
>> activation = F.sigmoid(image)
>> activation

Variable containing:
 0.5011  0.3140
 0.6621  0.5004
[torch.FloatTensor of size 2x2]
```

Apply ReLU pointwise

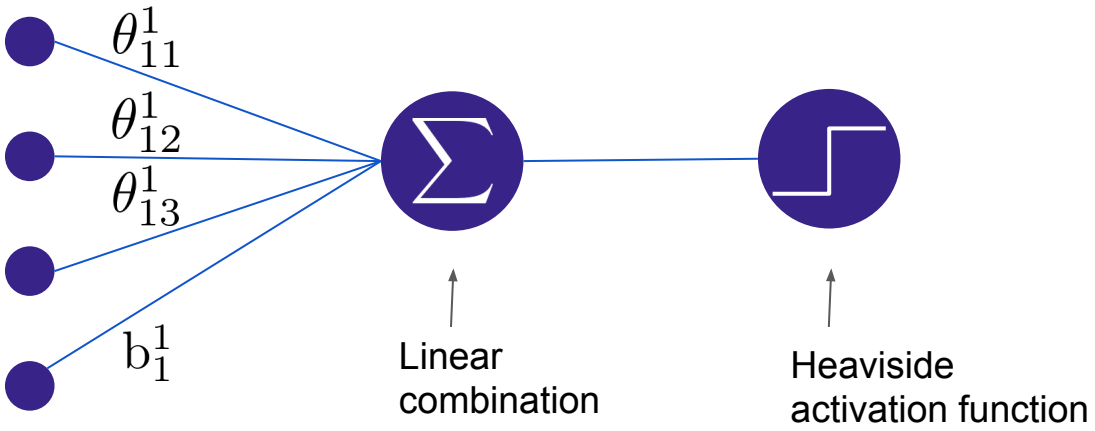
```
>> activation = F.relu(image)
>> activation

Variable containing:
 0.0044  0.0000
 0.6725  0.0016
[torch.FloatTensor of size 2x2]
```

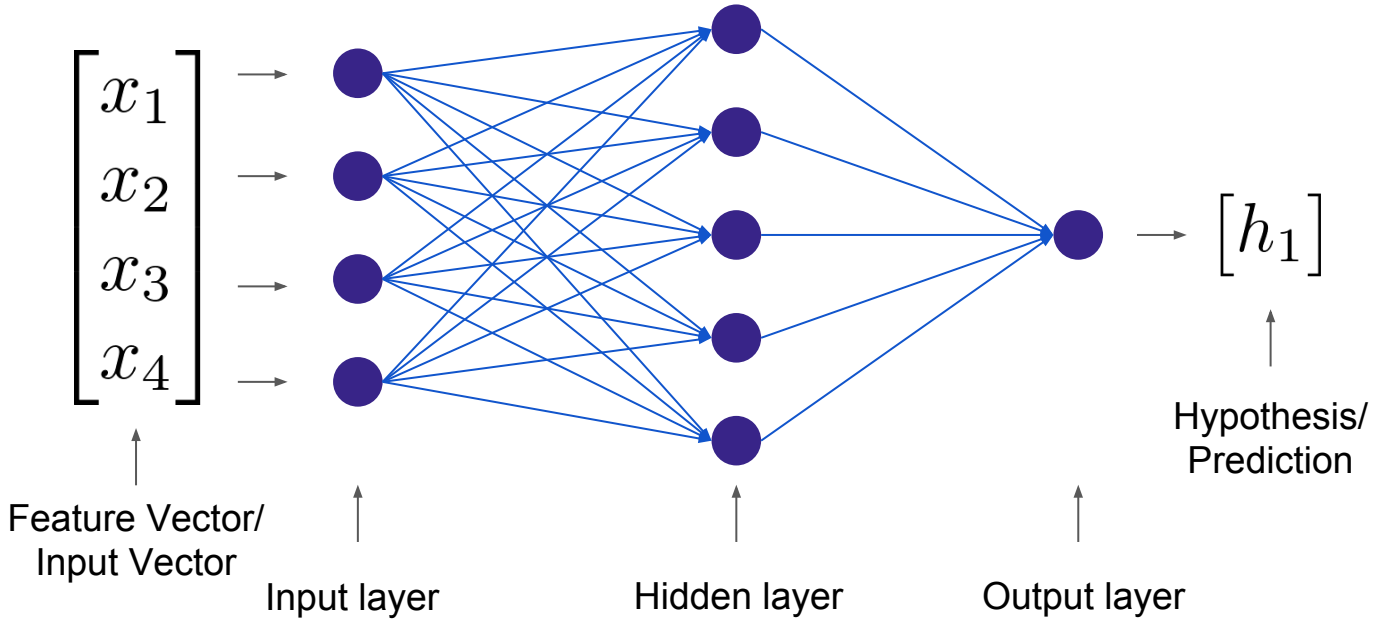
MACHINE INTELLIGENCE COMMUNITY

Perceptron

- An artificial neuron
  - Each neuron has weights associated with its inputs, and an activation function to delinearize linear data produced by its associated algorithm
  - Biases are assigned to each layer, to help fit the data without changing the function's shape



Multilayer Perceptron (MLP)



MACHINE INTELLIGENCE COMMUNITY

MLP in PYTORCH

Define fully-connected network

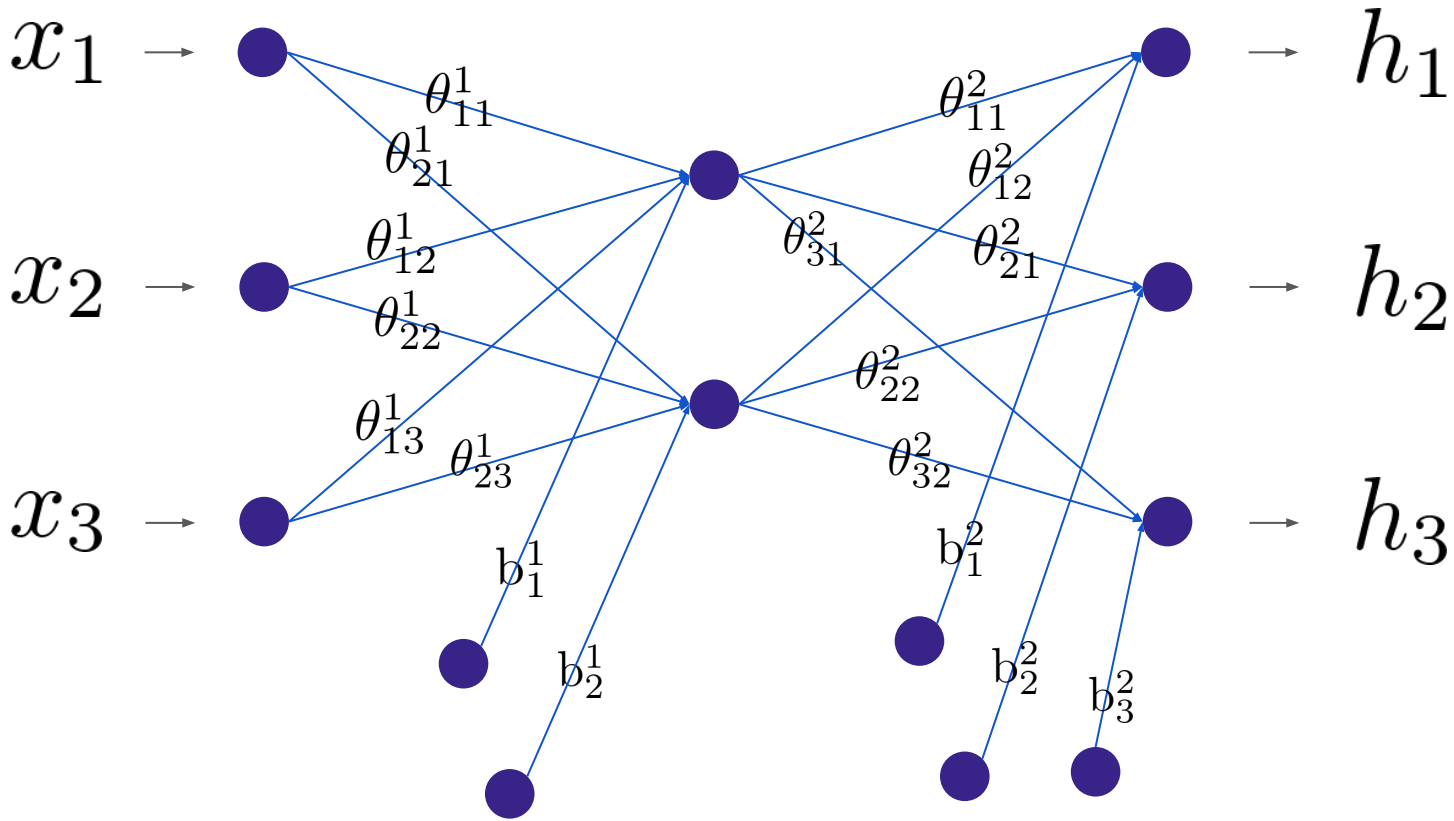
```
>> import torch.nn as nn
>> net =
nn.Sequential(nn.Linear(4, 2),
nn.Sigmoid(), nn.Linear(2, 2))
>> net
Sequential (
  (0): Linear (4 -> 2)
  (1): Sigmoid ()
  (2): Linear (2 -> 2)
)
```

Feedforward feature vector

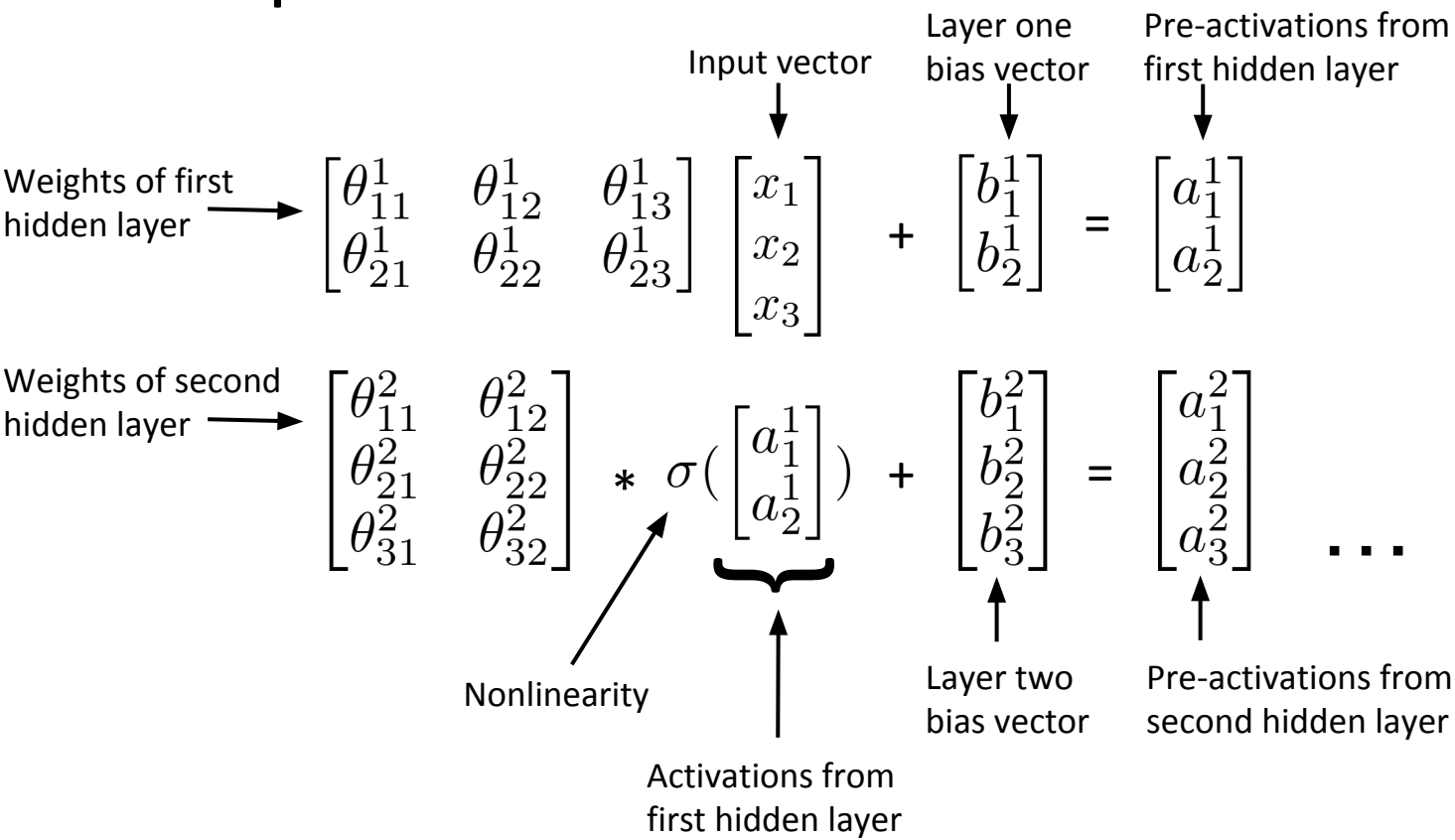
```
>> from torch.autograd import Variable
>> image = Variable(image.view(1,4))
>> net(image)
Variable containing:
 0.0817  0.3724
[torch.FloatTensor of size 1x2]
```

MACHINE INTELLIGENCE COMMUNITY

More detailed view of MLP



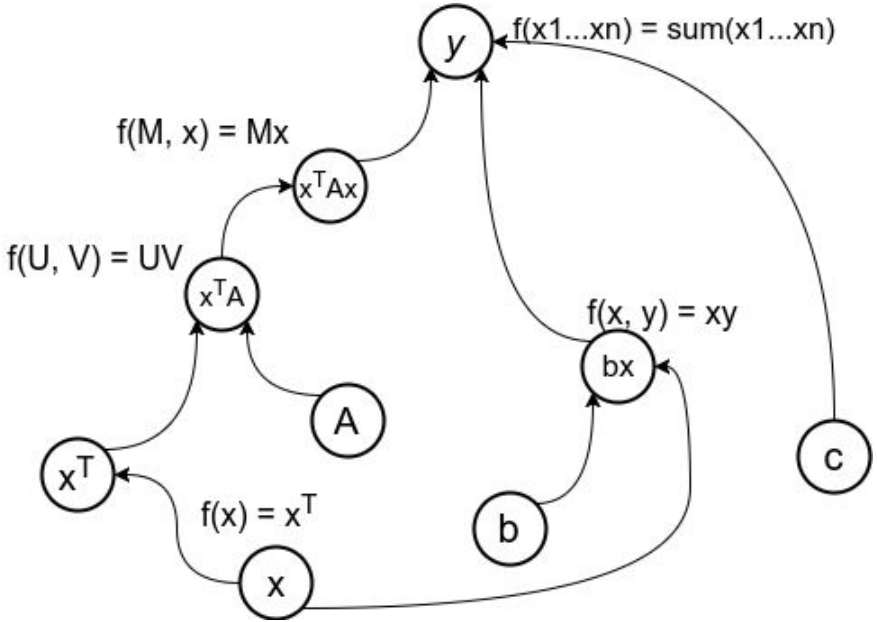
Matrix Representation

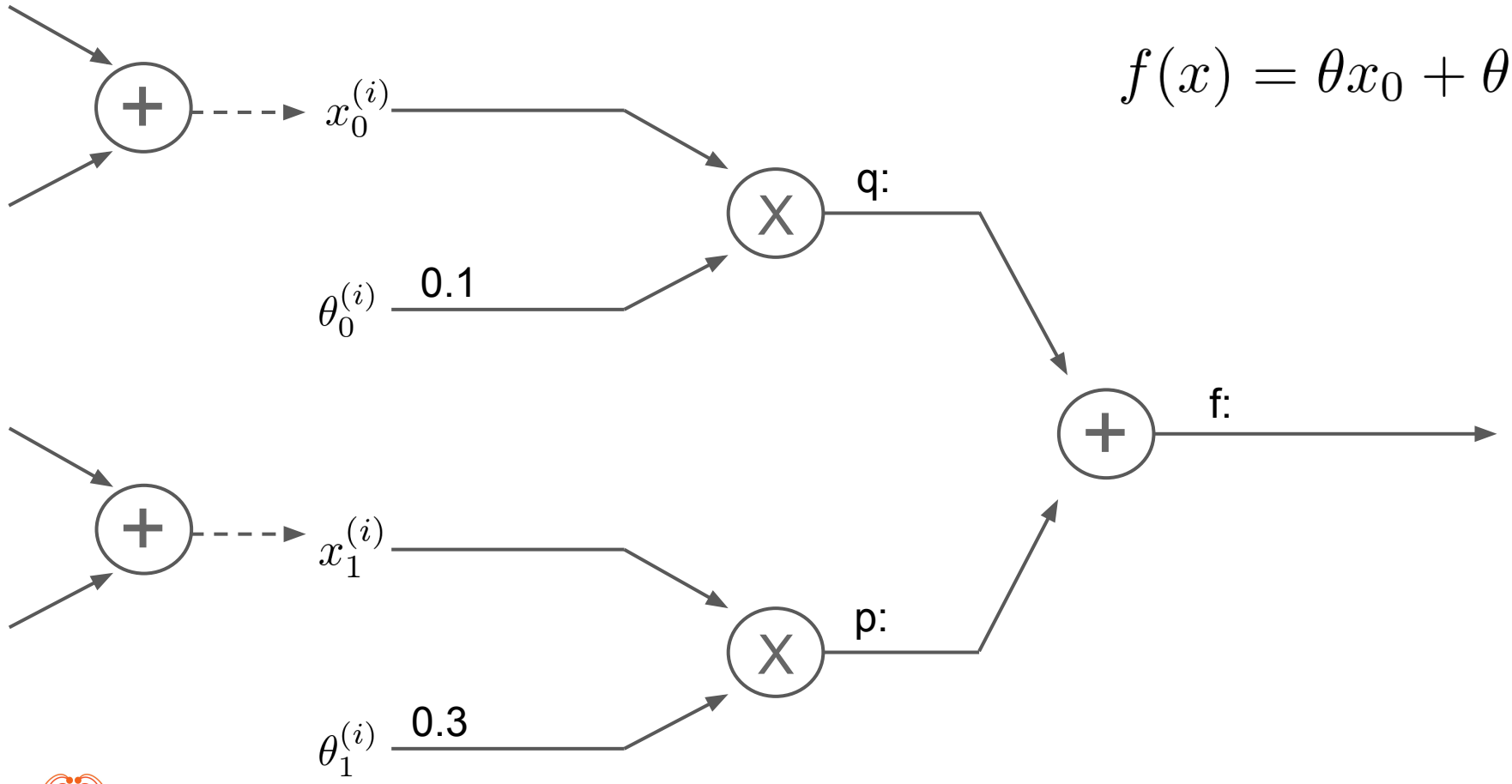


MACHINE INTELLIGENCE COMMUNITY

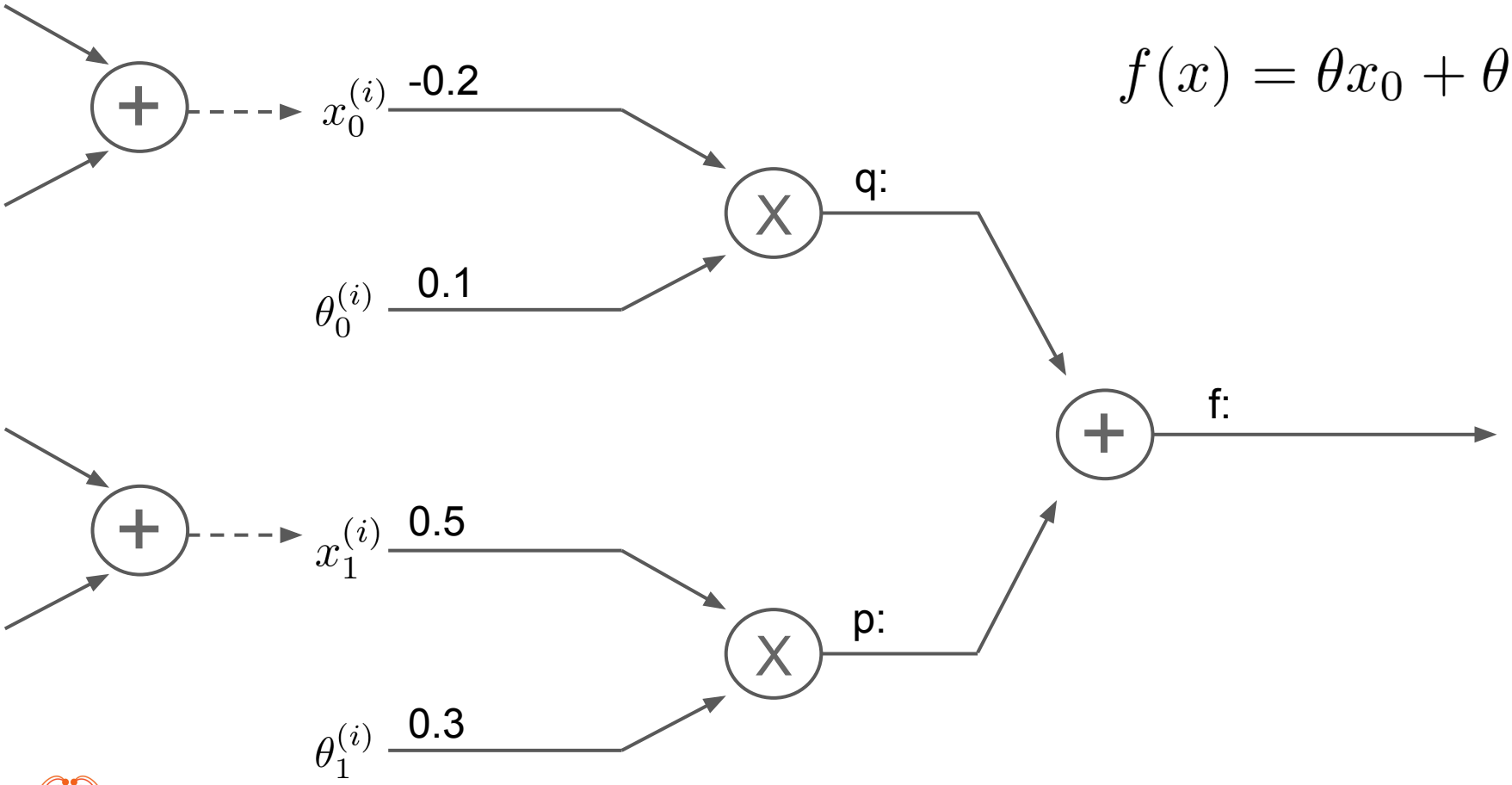
Computation Graph Representation

$$y = Ax + b \cdot x + c$$

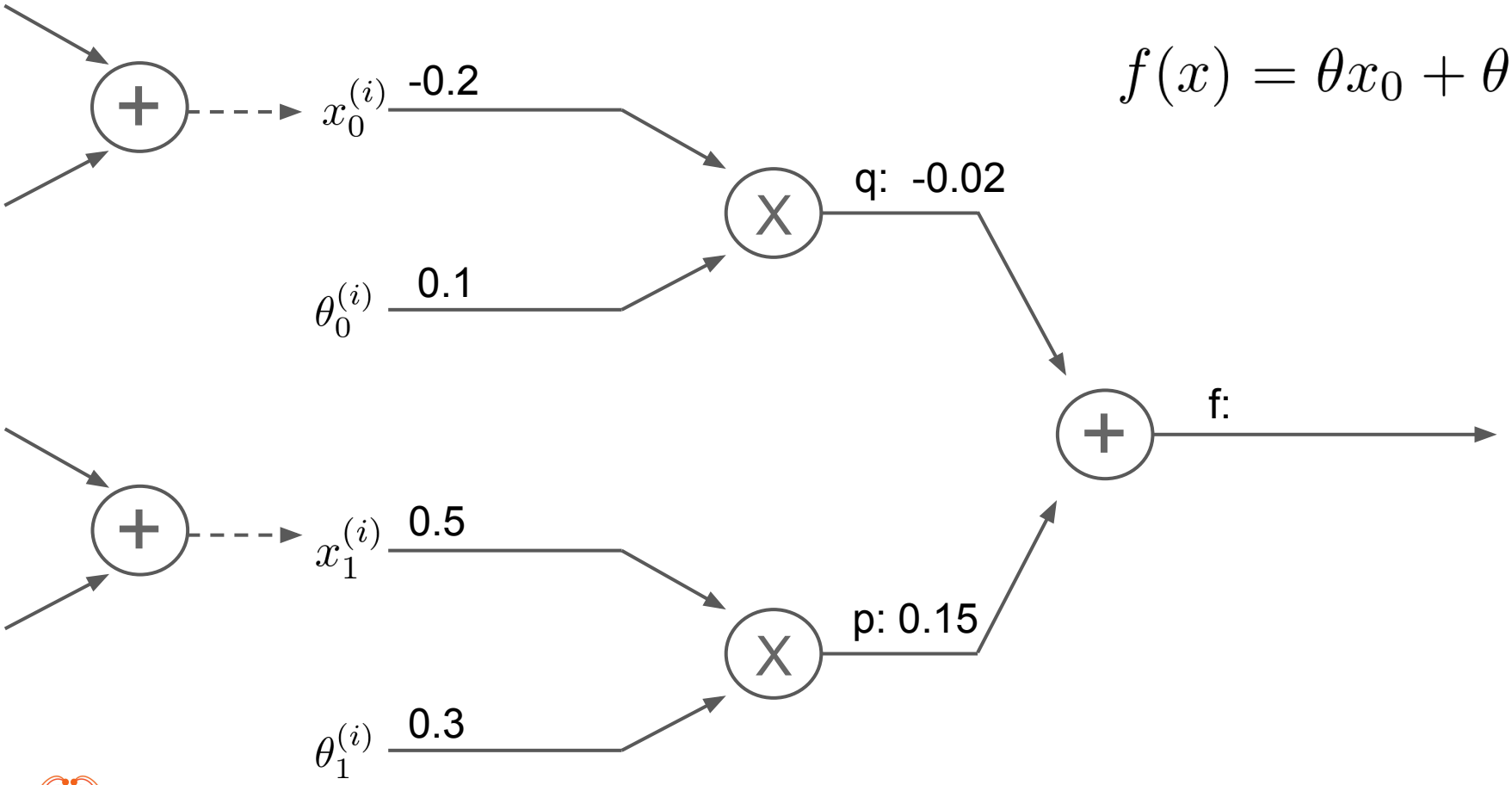




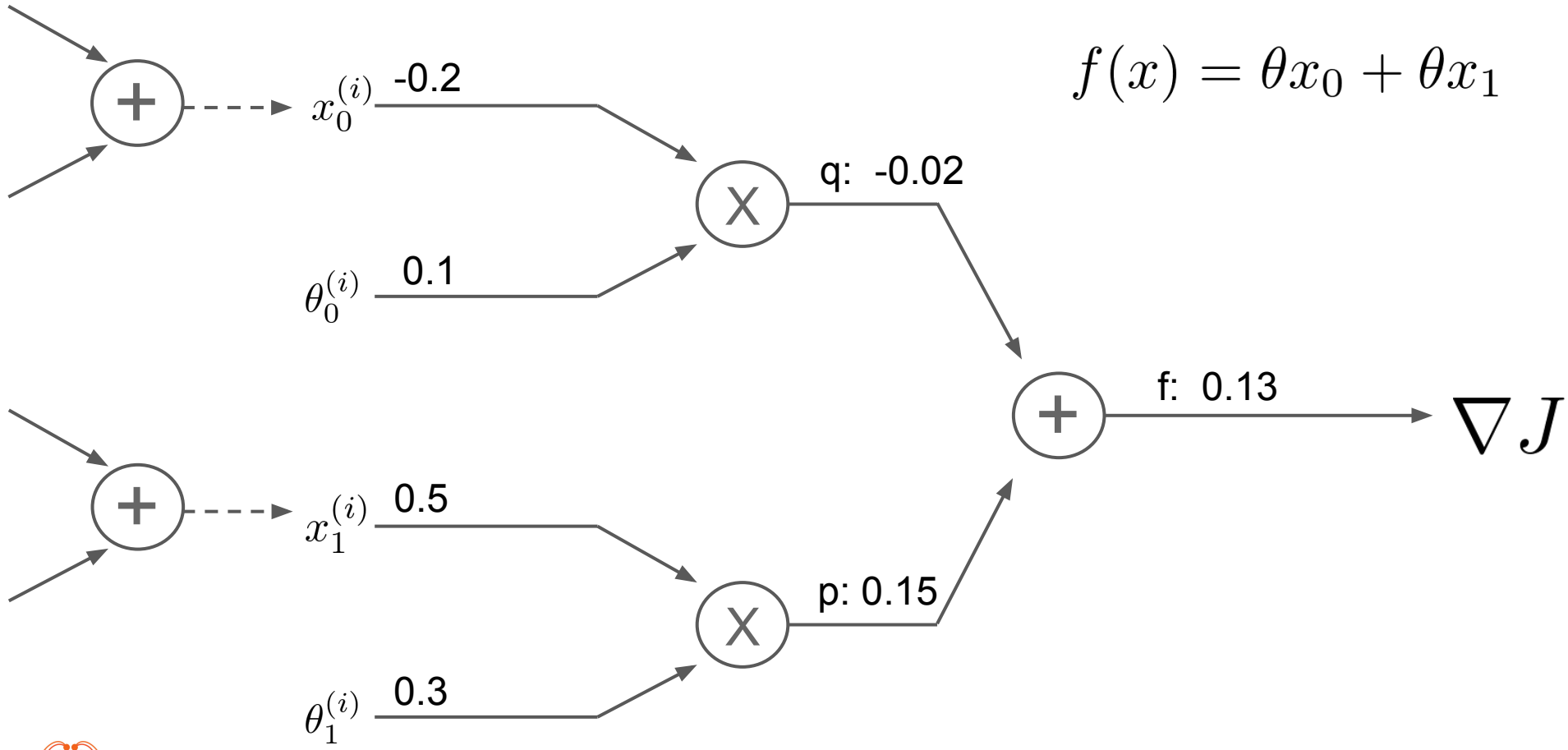
MACHINE INTELLIGENCE COMMUNITY



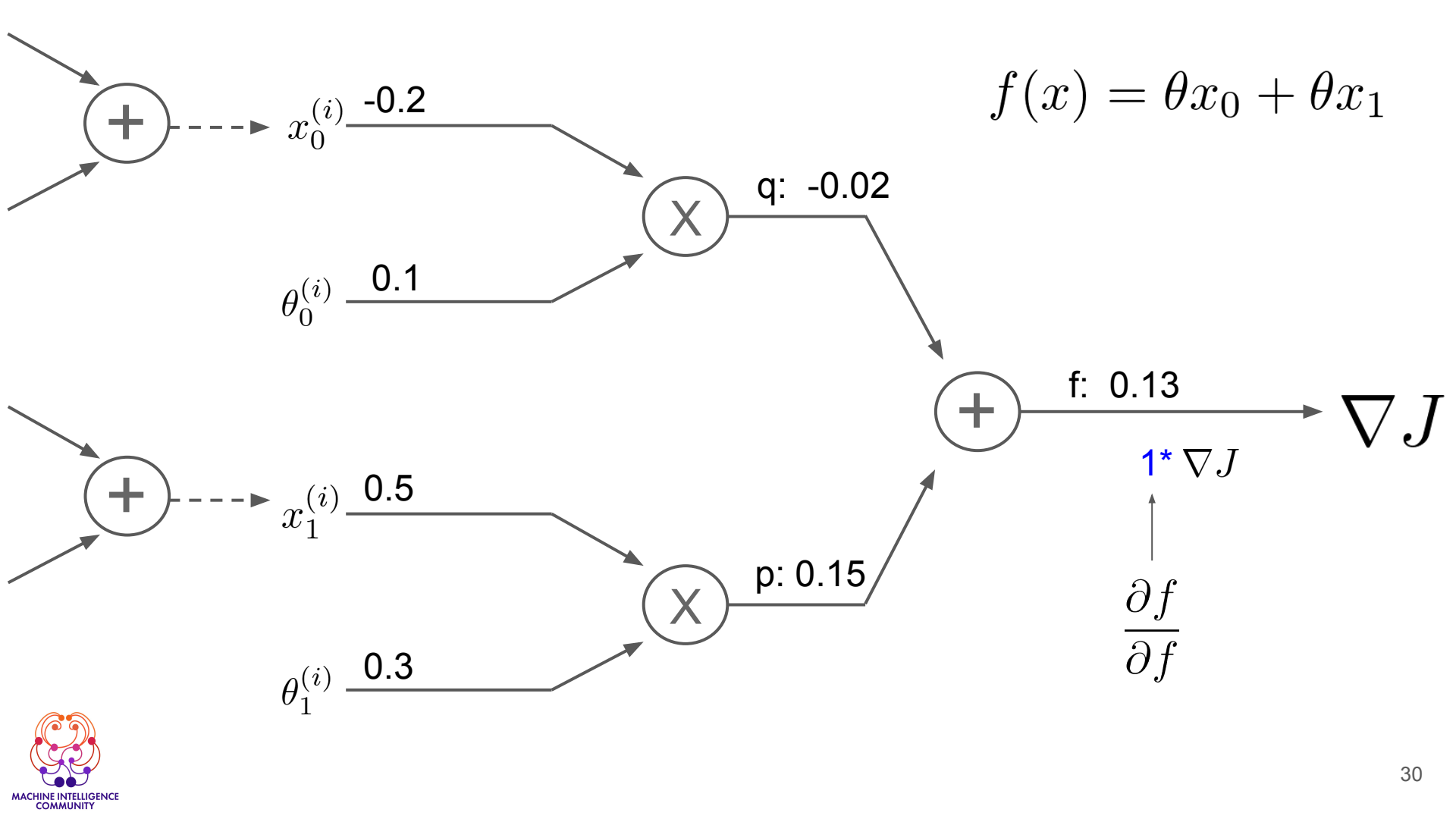
MACHINE INTELLIGENCE COMMUNITY



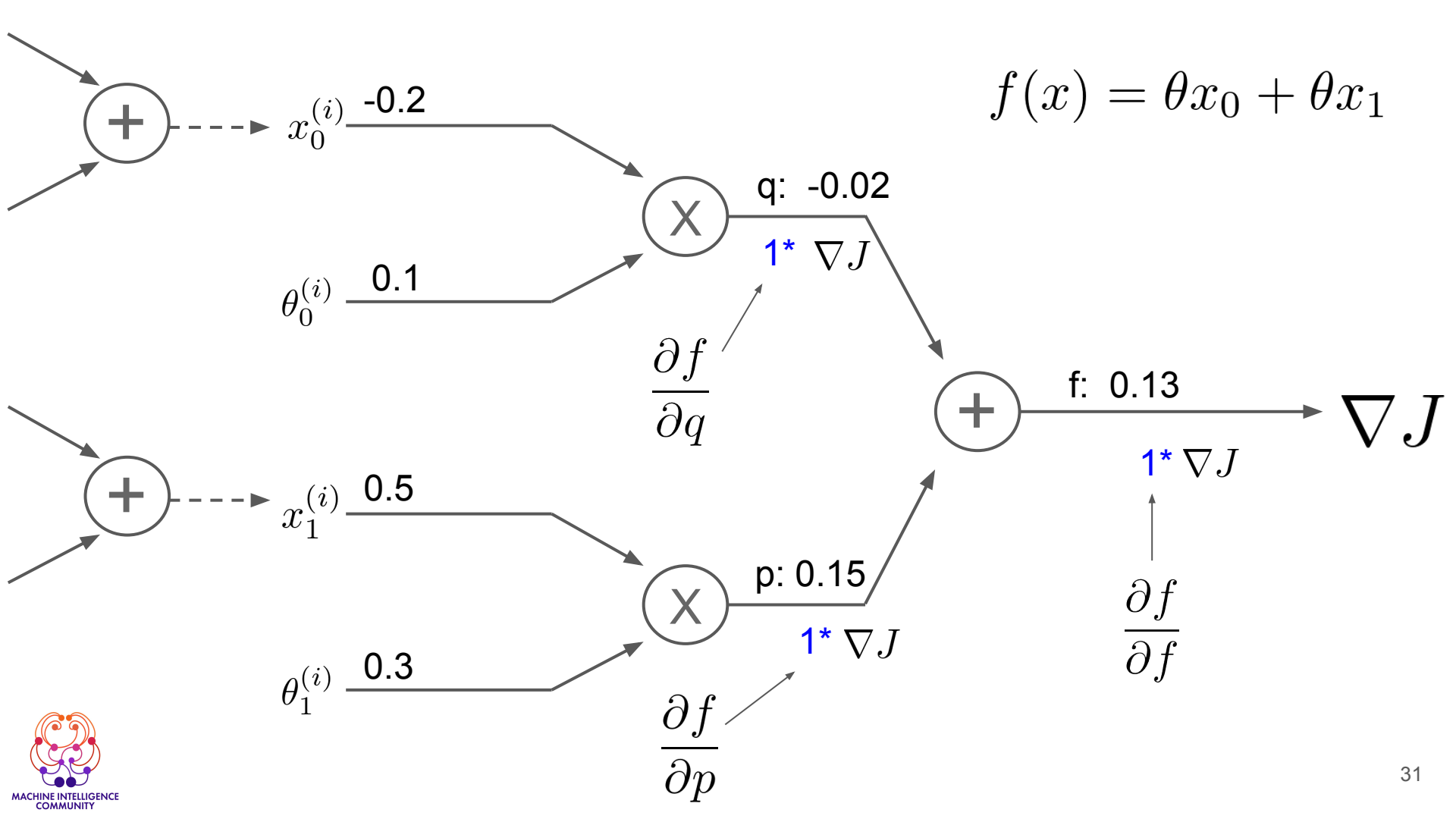
MACHINE INTELLIGENCE COMMUNITY



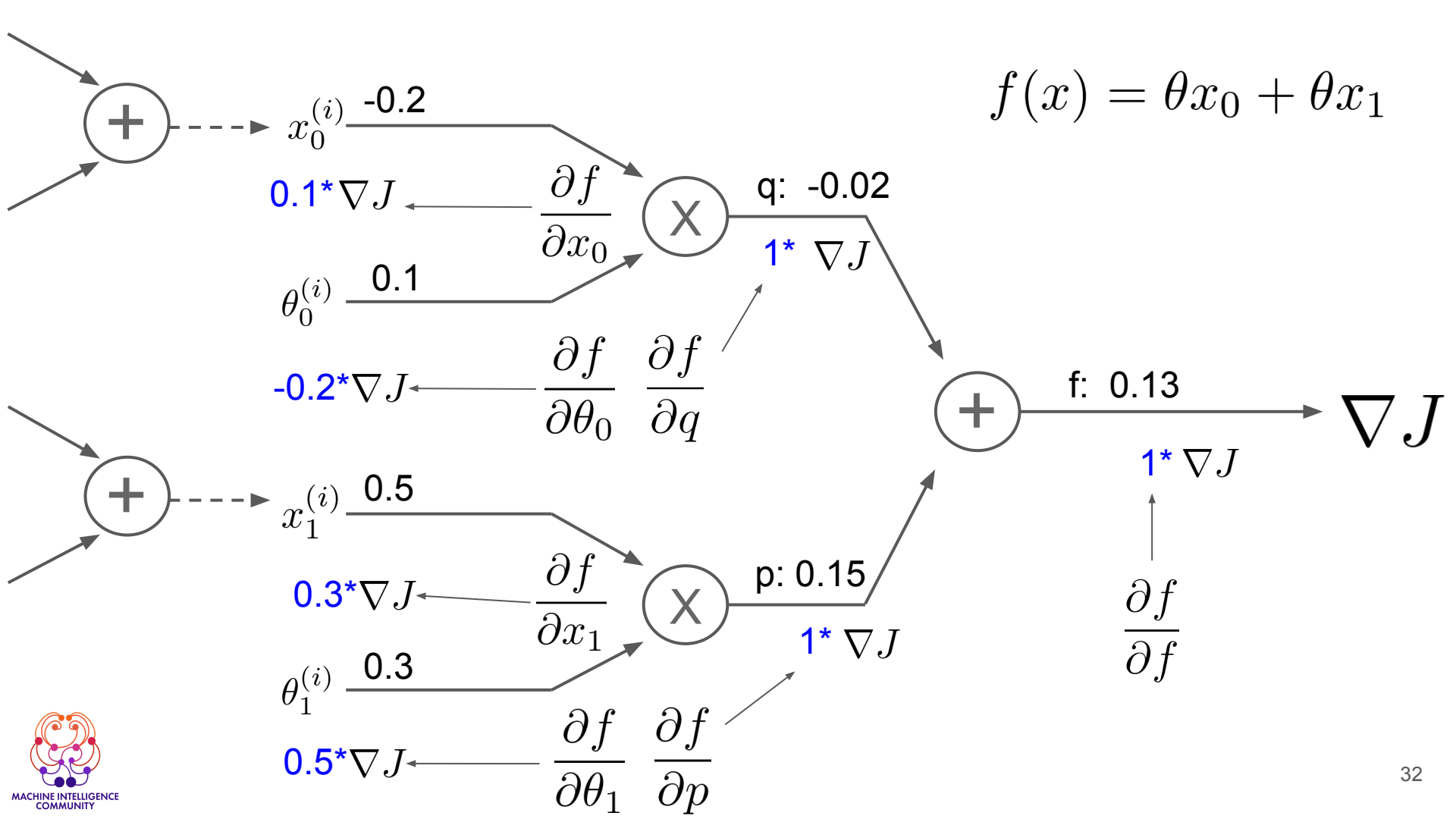
MACHINE INTELLIGENCE COMMUNITY



MACHINE INTELLIGENCE COMMUNITY



MACHINE INTELLIGENCE COMMUNITY



MACHINE INTELLIGENCE COMMUNITY

Static Computation Graphs (SCG)

- Generally, a static computation graph does static declaration to define whole graph first
  - Once defined, it cannot be changed during computation
- Steps:
  - Define an architecture
  - Run data through graph to train the weights
- Good for computational efficiency, but will run into trouble when architecture needs to be changed (e.g. recurrent neural nets)
- Also, language to define computational graph is very complex
  - TensorFlow currently most popular for using static computation graph

Dynamic Computation Graphs (DCG)

- Used for weird looking data e.g. graph data, tree data, jagged arrays
  - Separate execution and definition of the computation graph
  - Less invasive library, or syntax
- Allows for architectural changes, so works well for recurrent neural nets, for example
- Currently, PyTorch is the most popular framework for dynamic computation graphs and is becoming the standard in industry

MACHINE INTELLIGENCE COMMUNITY

Additional Resources

Andrew Ng's deeplearning.ai course:

https://www.coursera.org/learn/neural-networks-deep-learning/lecture/Cuf2f/welcome

PyTorch 60minute Blitz:

http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Efficient Backprop by Yann LeCun:

http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

BUMIC GitHub Link:

https://github.com/bumic/workshops

MACHINE INTELLIGENCE COMMUNITY

Shoutout to our Sponsors

Boston University Computer Science

Boston University SPARK!

Boston University Software Application and Innovation Lab

MACHINE INTELLIGENCE COMMUNITY

Upcoming Events

MIC Paper signup: https://goo.gl/iAm6TL
BUMIC Projects signup: https://goo.gl/GmP9oK

MIT MIC reading group:

Paper: Deepface
Location: MIT 56-154 (building 56, room 154)
Date: 10.05.17 Time: 5 PM

BU MIC reading group:

Paper: Character-level Convolutional Networks for Text Classification
Date: 10.06.17 Time: 7 PM

Next workshop:

Location: BU Hariri Seminar Room
Date: 10.10.17 Time: 7 PM

MACHINE INTELLIGENCE COMMUNITY