Why Adversarial Interaction Creates Non-Homogeneous Patterns: A Pseudo-Reaction Diffusion Model for Turing Instability

Litu Rout

Association for the Advancement of Artificial Intelligence (AAAI-21)

Preprint: https://liturout.github.io/data/aaai21_preprint.pdf

June 2, 2021

Why Adversarial Interaction Creates Non-Homogeneous Patterns: A Pseudo-Reaction Diffusion Model for Turing Instability

Why Adversarial Interaction Creates Non-Homogeneous Patterns: A Pseudo-Reaction Diffusion Model for Turing Instability

- Adversarial Interaction
  - Generative Adversarial Networks (GANs)
  - Application of conditional GANs
- Non-Homogeneous Patterns
  - Homogeneous patterns
  - Supervised learning

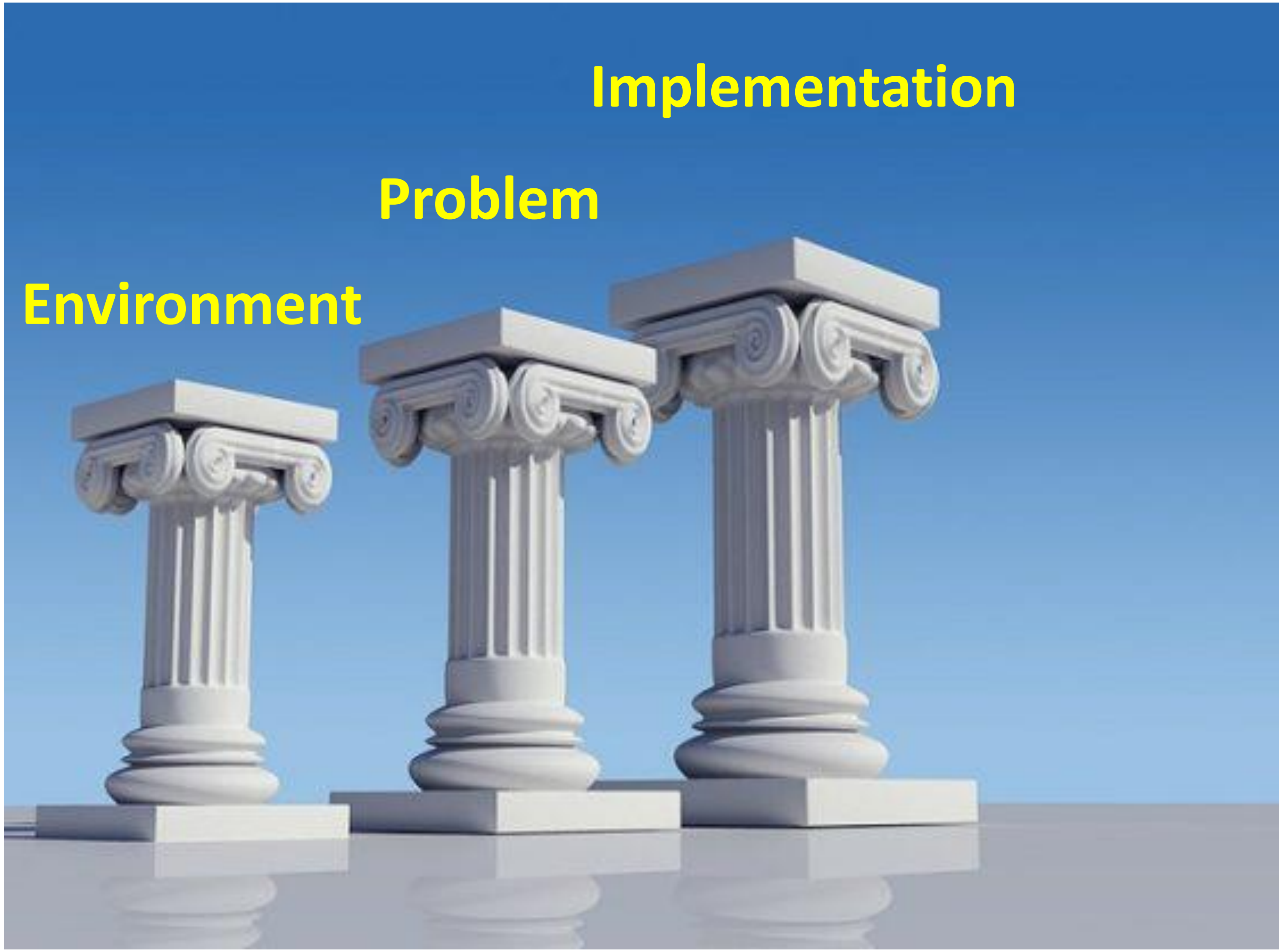
Why Adversarial Interaction Creates Non-Homogeneous Patterns: A Pseudo-Reaction Diffusion Model for Turing Instability

- Adversarial Interaction
  - Generative Adversarial Networks (GANs)
  - Application of conditional GANs
- Non-Homogeneous Patterns
  - Homogeneous patterns
  - Supervised learning
- Reaction-Diffusion
  - Turing’s RD model (1952)
  - Gray-Scott RD model (1984)
- Turing Instability
  - Reaction dynamics
  - Diffusion dynamics

Why Adversarial Interaction Creates Non-Homogeneous Patterns: A Pseudo-Reaction Diffusion Model for Turing Instability

- Adversarial Interaction
  - Generative Adversarial Networks (GANs)
  - Application of conditional GANs
- Non-Homogeneous Patterns
  - Homogeneous patterns
  - Supervised learning
- Reaction-Diffusion
  - Turing's RD model (1952)
  - Gray-Scott RD model (1984)
- Turing Instability
  - Reaction dynamics
  - Diffusion dynamics

Three Pillars of Deep Learning



Three Pillars of Deep Learning

- Setting Up DL Environment
- Defining Problem Statement
- Implementation Details

Setting Up DL Environment

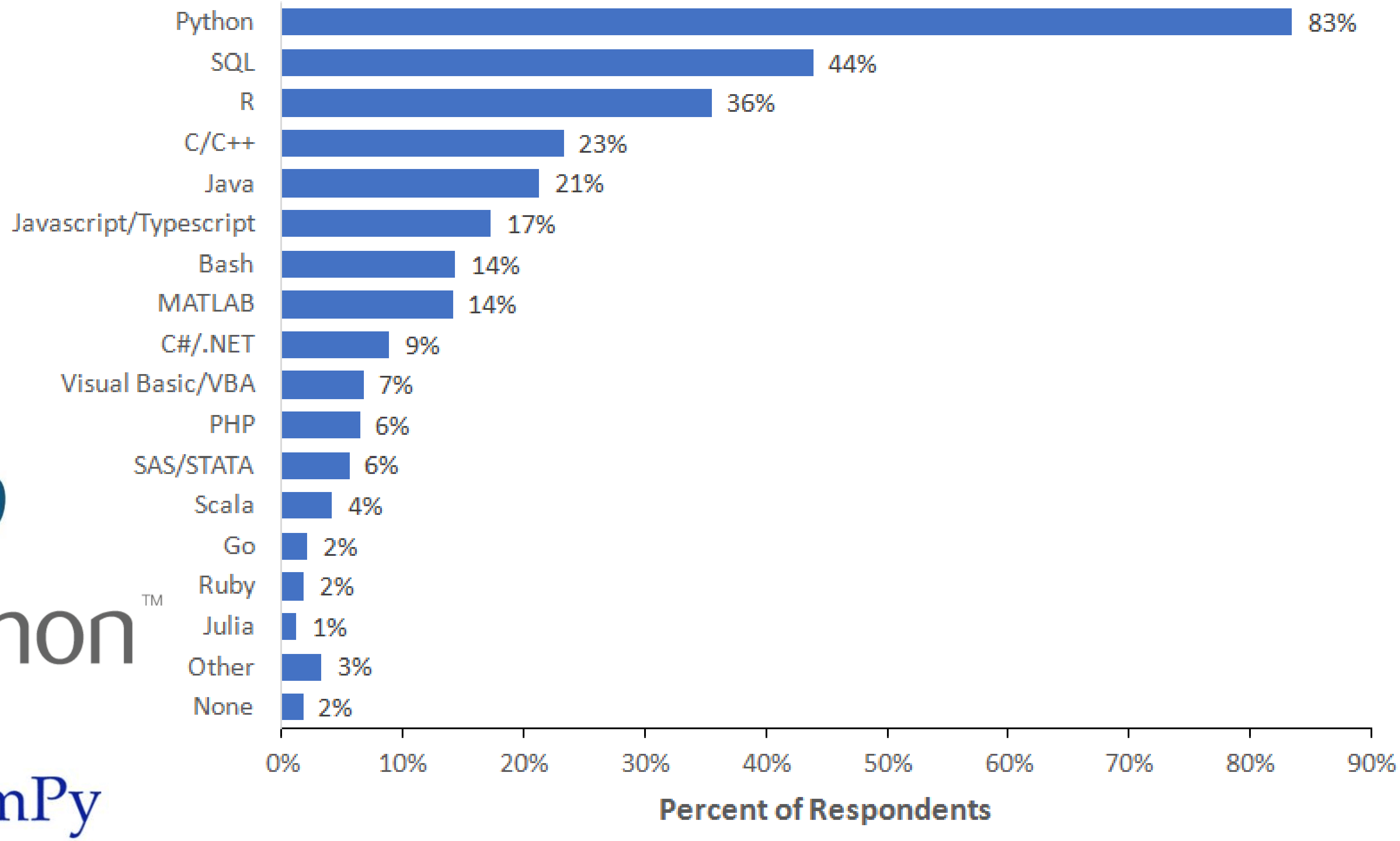
- Data Processing
- Network Design
- Visualization

Setting Up DL Environment

- Data Processing
- Network Design
- Visualization

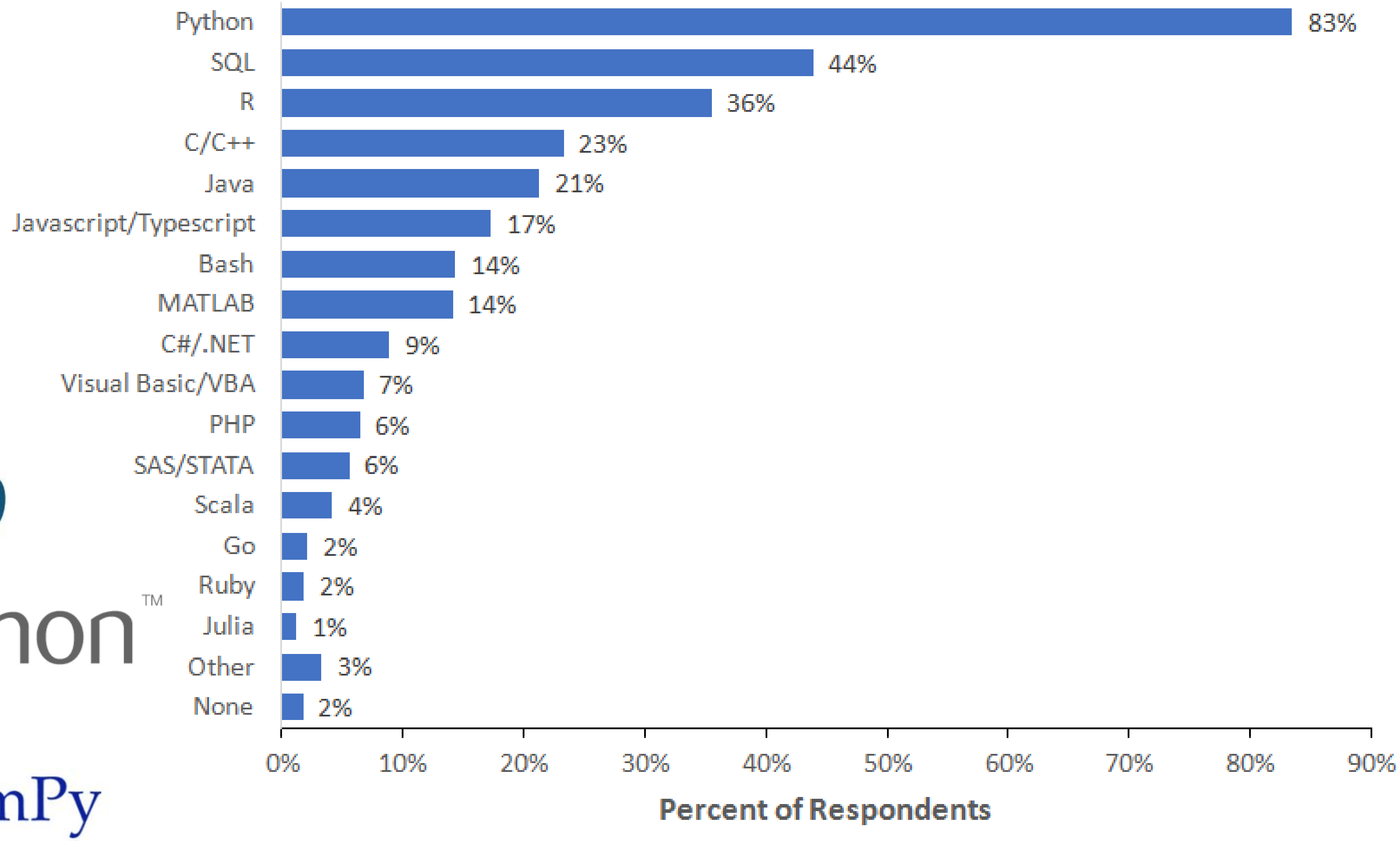
Data Processing

- Programming Language
- Image Processing
- Numeric Computation
- Data Manipulation



Data Processing

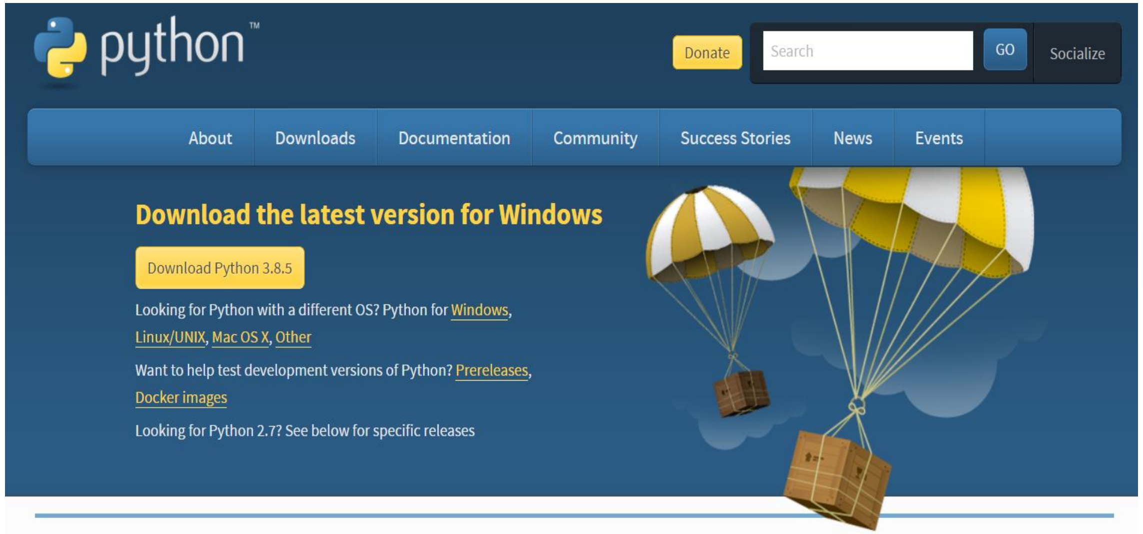
- Programming Language
- Image Processing
- Numeric Computation
- Data Manipulation



Data Processing

- Package Installation via “pip”

  >> pip install package



- Package Installation via “conda”

  >> conda install package

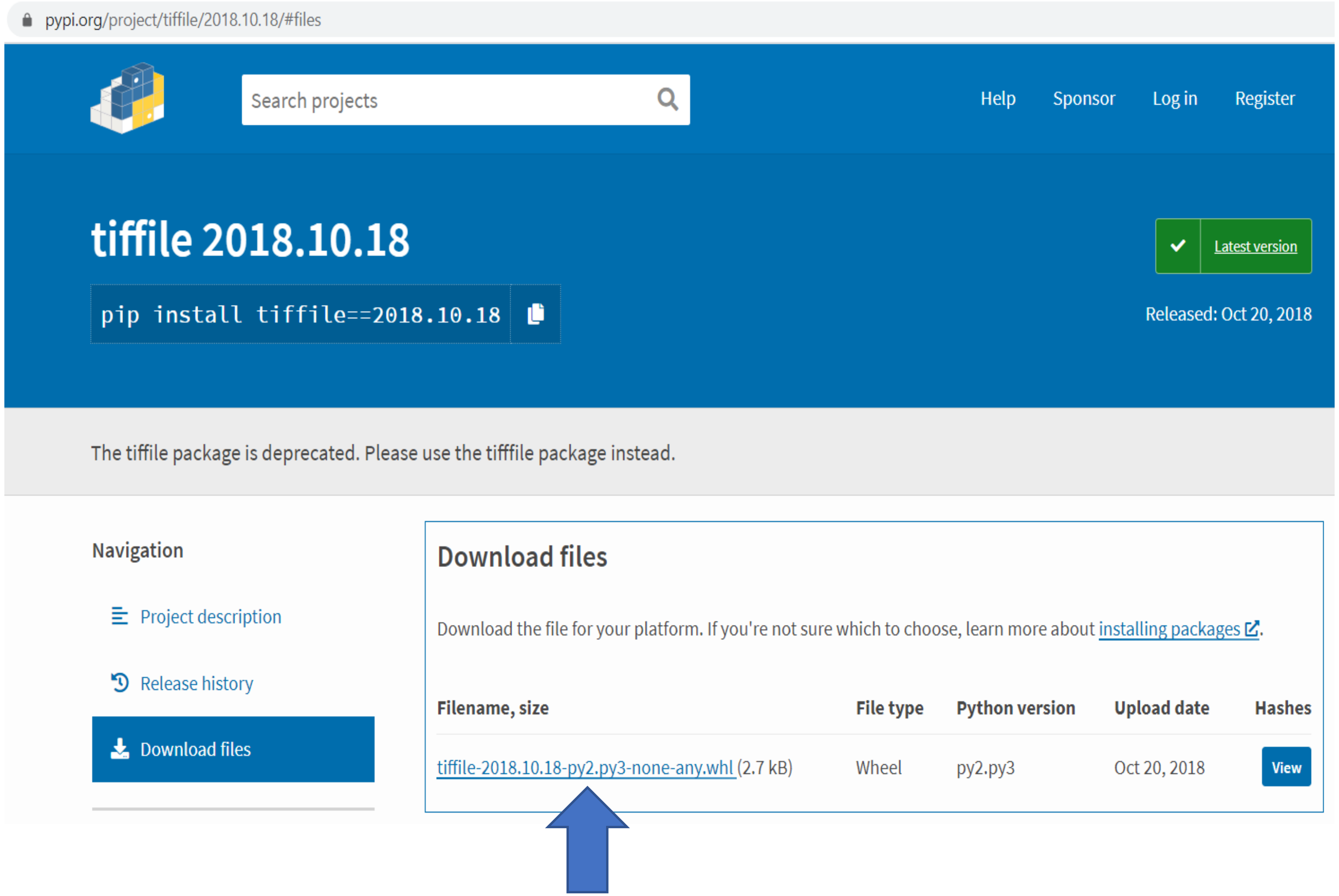
ANACONDA®

Many packages ship pre-installed in Anaconda

Data Processing

- Offline Installation
  - Download on Thin Client

>> pip install package.whl
or
>> pip install package.tar.gz



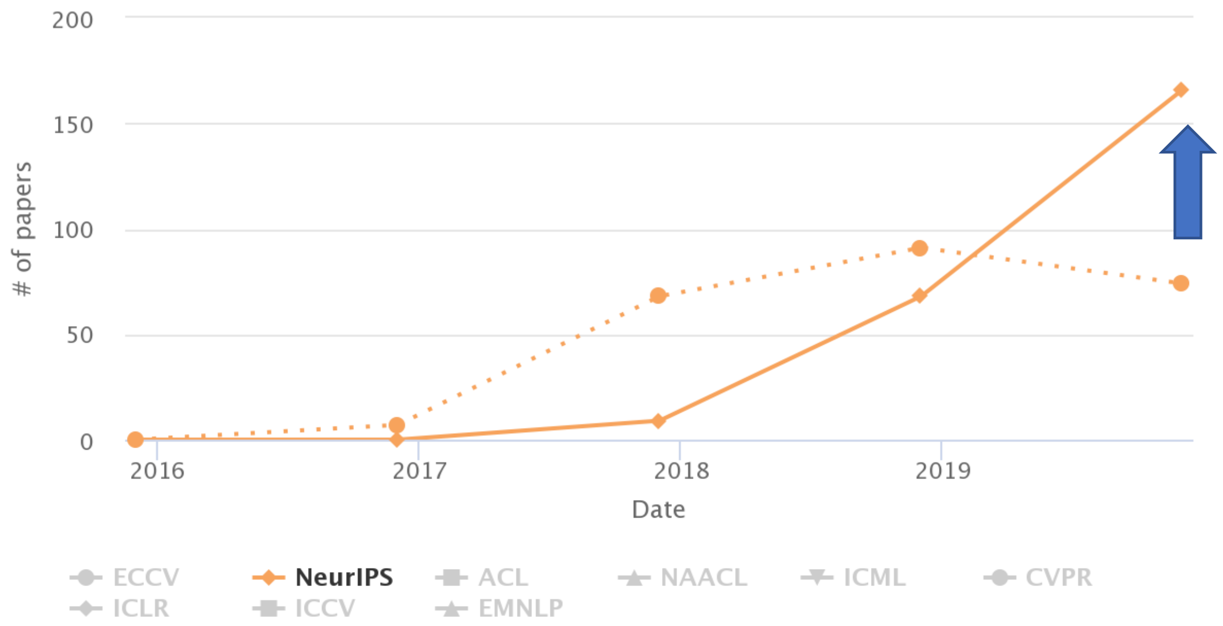
Setting Up DL Environment

- Data Processing
- Network Design
- Visualization

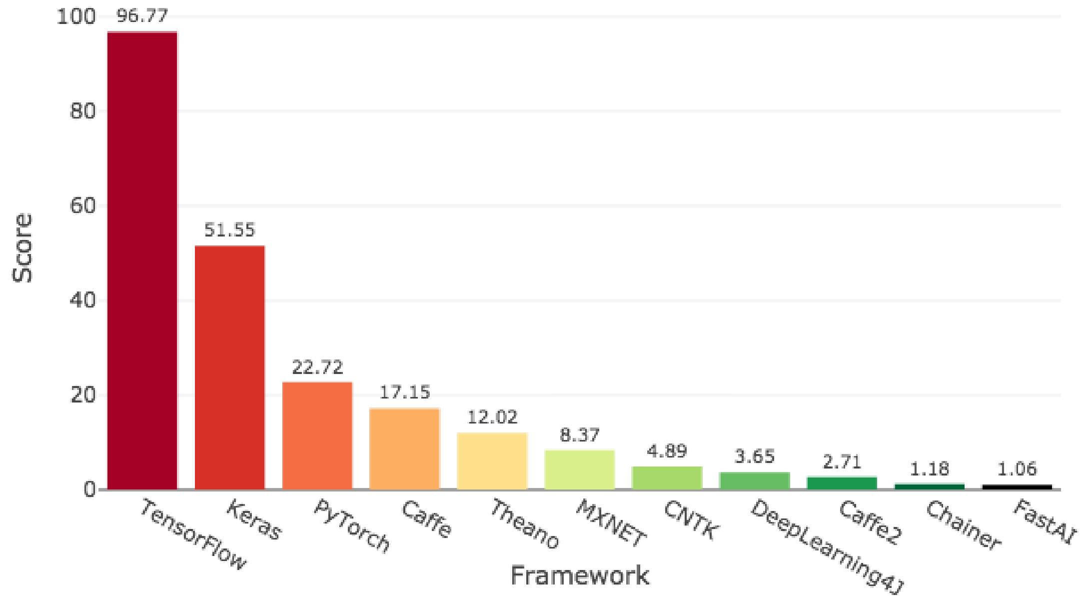
Network Design

- Popular Libraries

PyTorch (Solid) vs TensorFlow (Dotted) Raw Counts



DL Libraries in 2018



Setting Up DL Environment

- Data Processing
- Network Design
- Visualization

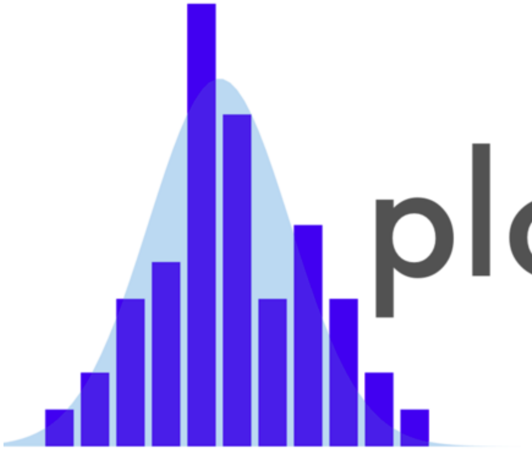
Visualization of Loss Surface

Visualization

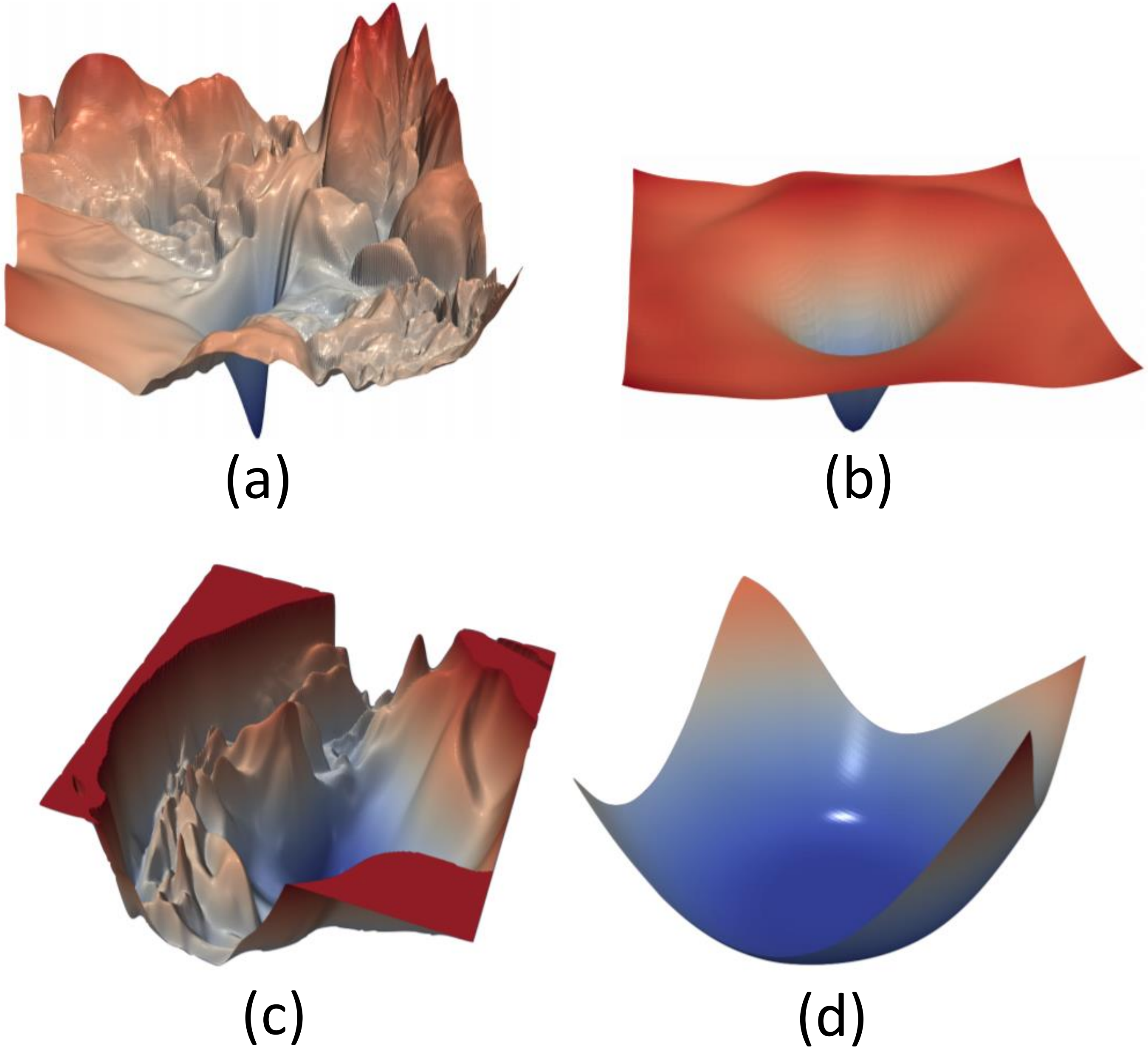
- Popular Libraries

matplotlib

Seaborn



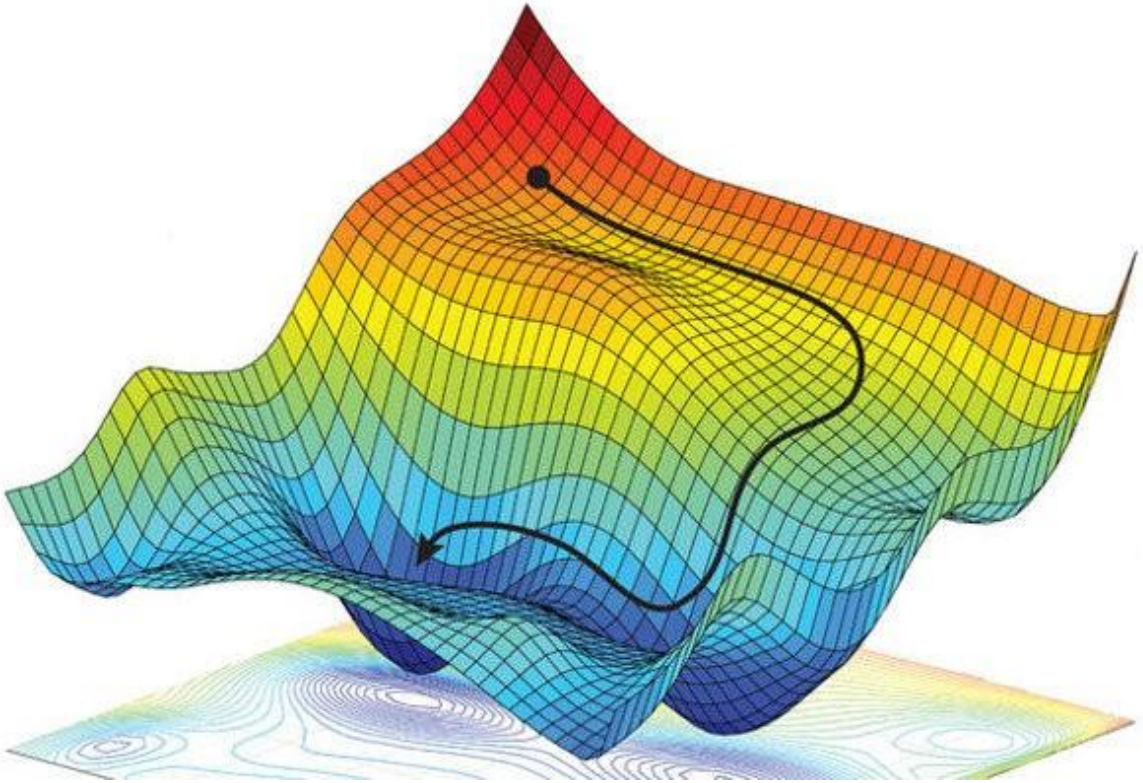
plotly



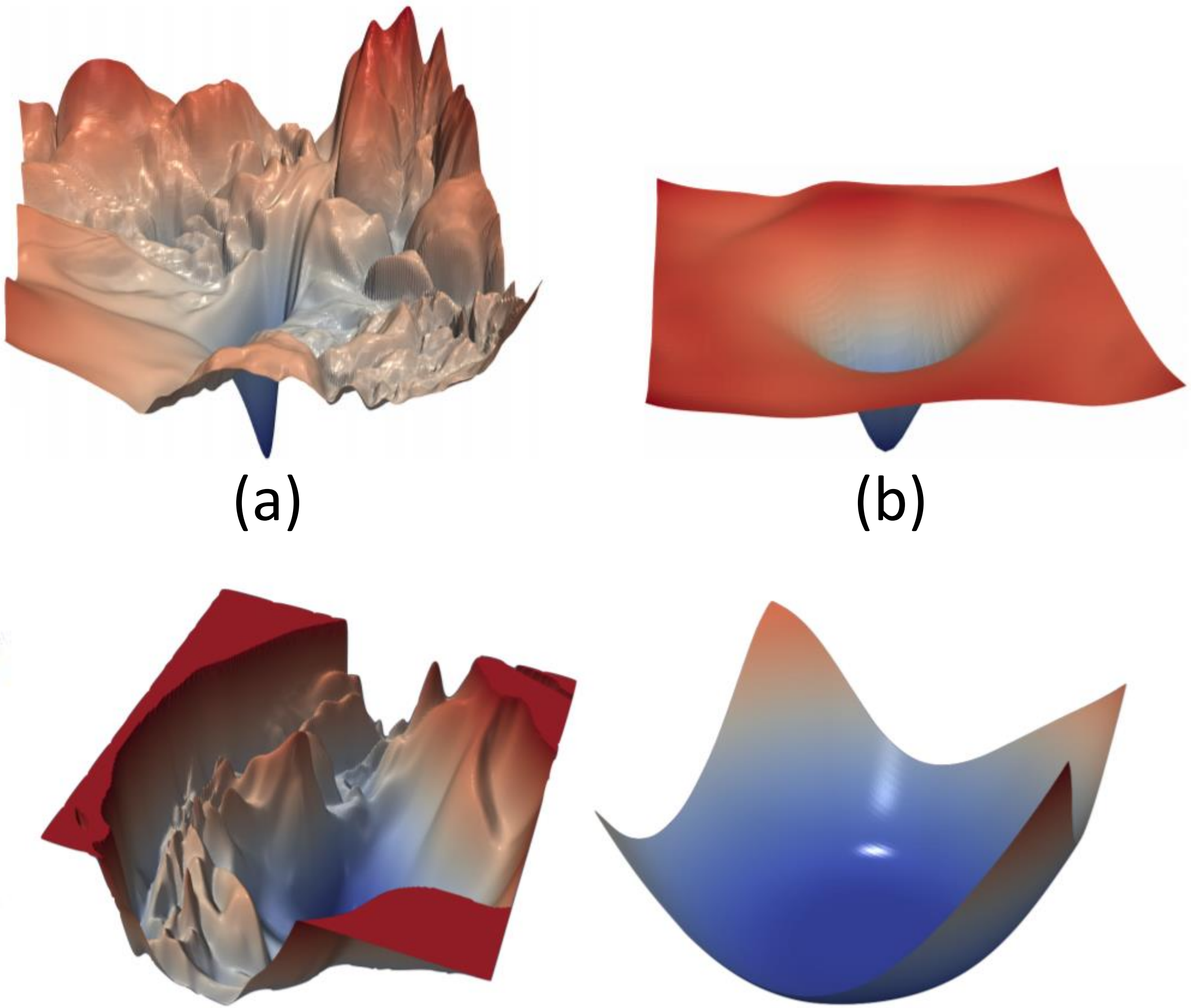
Li et al. NeurIPS 2018

Visualization

- Popular Libraries



Visualization of Loss Surface



(a)

(b)

(c)

(d)

Li et al. NeurIPS 2018

Three Pillars of Deep Learning

- Setting Up DL Environment
- Defining Problem Statement
- Implementation Details

Defining Problem Statement

- Linear Function Approximation
  - Dataset Preparation

    $$\{(x_p, y_p)\}_{p=1}^{n} \subset R^{d_{in} \times d_{out}}$$

  - Function Approximator

    $$f(m, c, x) = m\,x + c$$

  - Goal

    $$m = ?\,, c = ?$$

$f(.)$

x	y
-10	-48
-9	-43
…	…
10	52

Defining Problem Statement

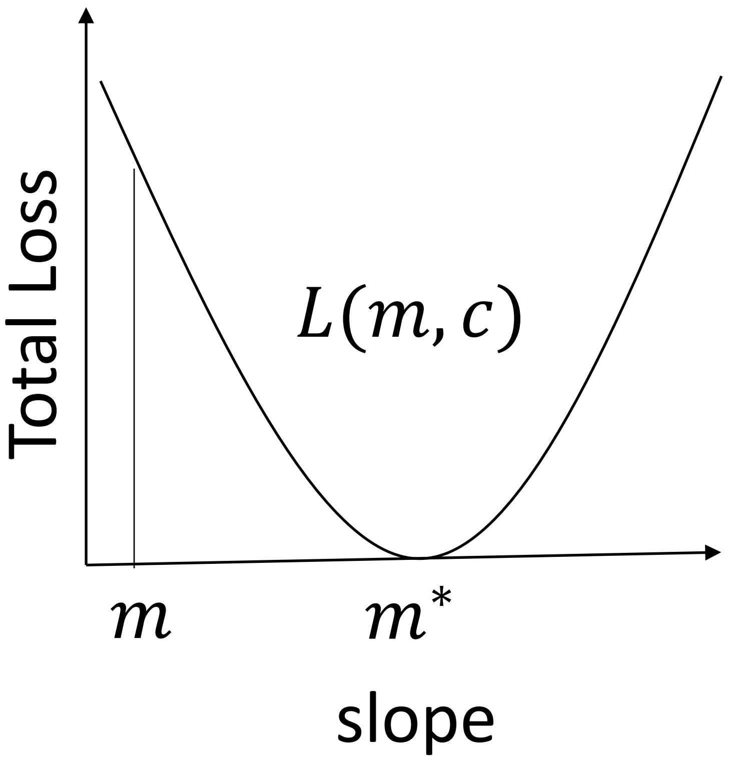
- Linear Function Approximation
  - Error Computation

$$l(f(m,c,x),y) = \frac{1}{2}(f(m,c,x) - y)^2$$

$$L(m,c) = \frac{1}{2n}\sum_{p=1}^{n}\left(f\left(m,c,x_p\right) - y_p\right)^2$$

  - Optimization

$$(m^*, c^*) = \arg min_{((m,c)\in R^{1\times 1})} L(m,c)$$



Defining Problem Statement

- Linear Function Approximation
  - Optimization

$$(m^*, c^*) = \arg min_{((m,c)\in R^{1\times 1})} L(m, c)$$

  - Learning Algorithm: GD

$$\frac{dm}{dt} = -\eta \frac{\partial L(m,c)}{\partial m(t)}, \quad m(t+1) = m(t) - \eta \frac{\partial L(m,c)}{\partial m(t)}$$

$$\frac{dc}{dt} = -\eta \frac{\partial L(m,c)}{\partial c(t)}, \quad c(t+1) = c(t) - \eta \frac{\partial L(m,c)}{\partial c(t)}$$

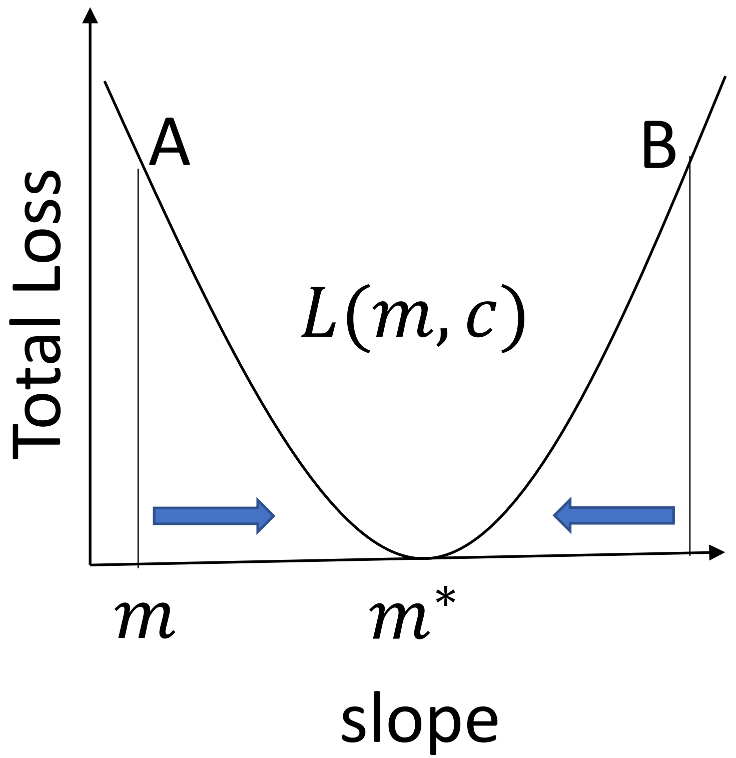
Defining Problem Statement

- Linear Function Approximation
  - Gradient Descent (GD)

$$m(t+1) = m(t) - \eta \frac{\partial L(m,c)}{\partial m(t)}$$

$$\frac{\partial L(m,c)}{\partial m(t)} < 0$$ at point A

$$\frac{\partial L(m,c)}{\partial m(t)} > 0$$ at point B



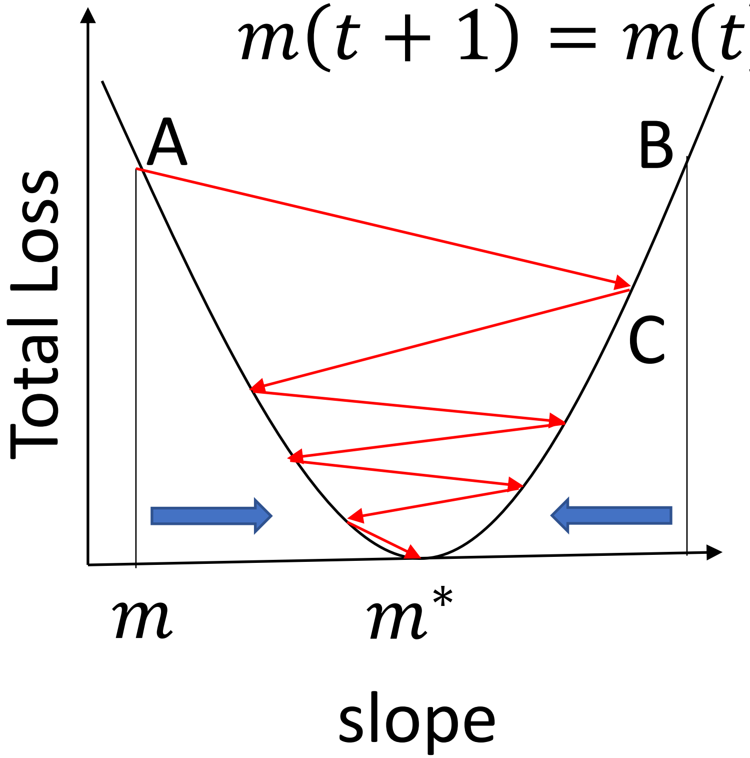
Defining Problem Statement

- Linear Function Approximation
  - Gradient Descent (GD)

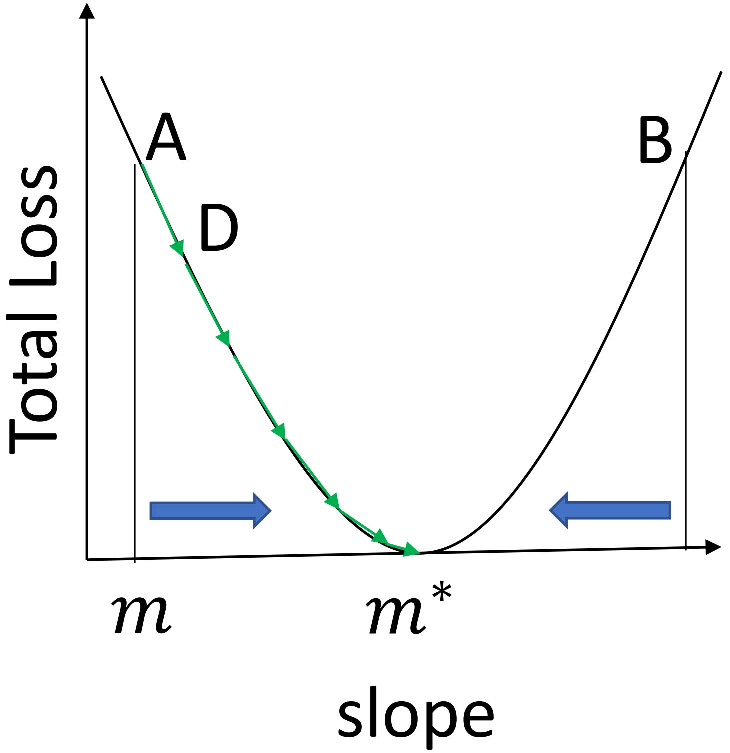
Learning Rate: η

$$m(t+1) = m(t) - \eta \frac{\partial L(m,c)}{\partial m(t)}$$

$\eta \gg$



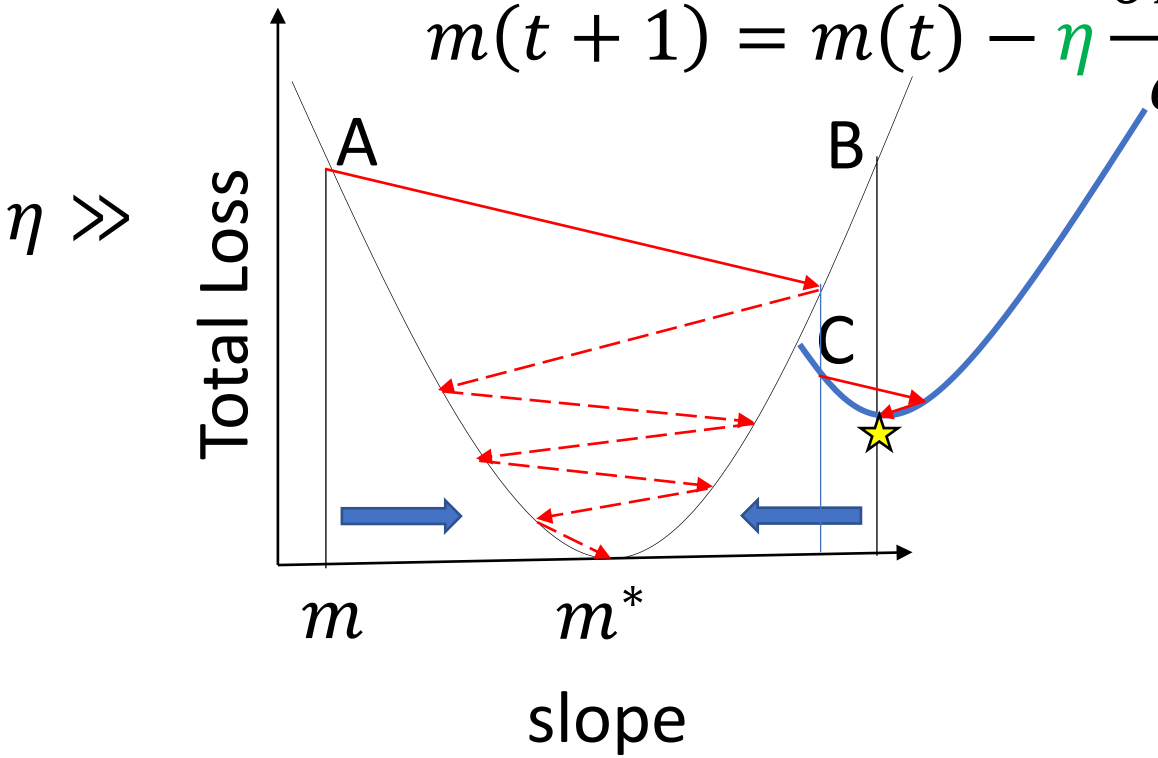
$\eta \ll$



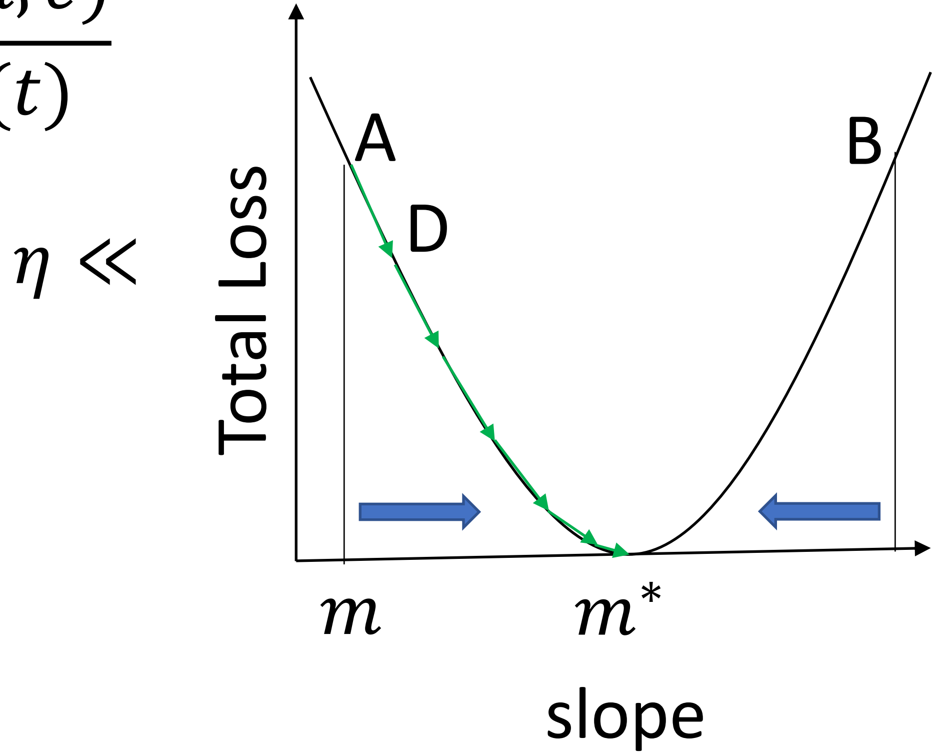
Defining Problem Statement

- Linear Function Approximation
    - Gradient Descent (GD)

$$m(t+1) = m(t) - \eta \frac{\partial L(m,c)}{\partial m(t)}$$



Learning Rate: η



Defining Problem Statement

- Linear Function Approximation
  - Gradient Descent (GD)

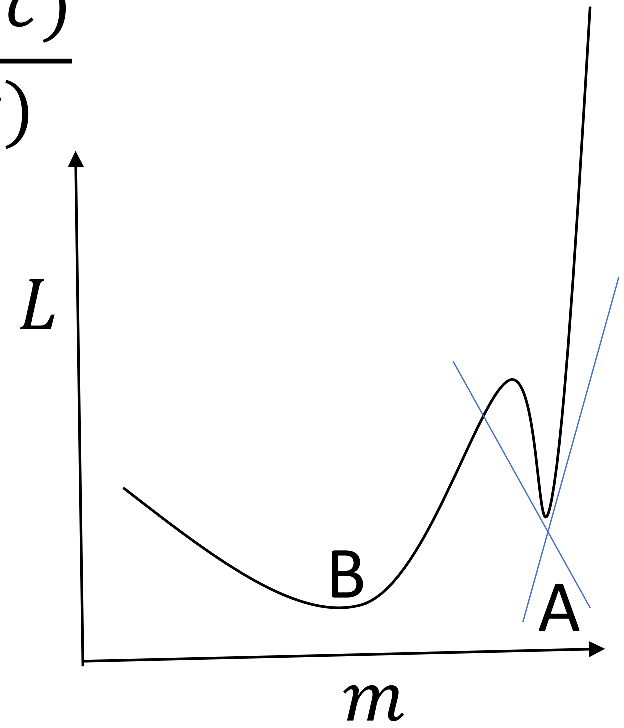
$$\eta = \beta^{-1}$$

$$\boxed{m_{t+1} = m_t - \beta^{-1} \nabla L(m_t)}, \nabla L(m_t) \triangleq \frac{\partial L(m,c)}{\partial m(t)}$$

  - Assumption: β-Smoothness

$$\|\nabla L(m_{t+1}) - \nabla L(m_t)\|_2 \leq \beta \|m_{t+1} - m_t\|_2$$

Or $\|\nabla^2 L(m_t)\|_2 \leq \beta$



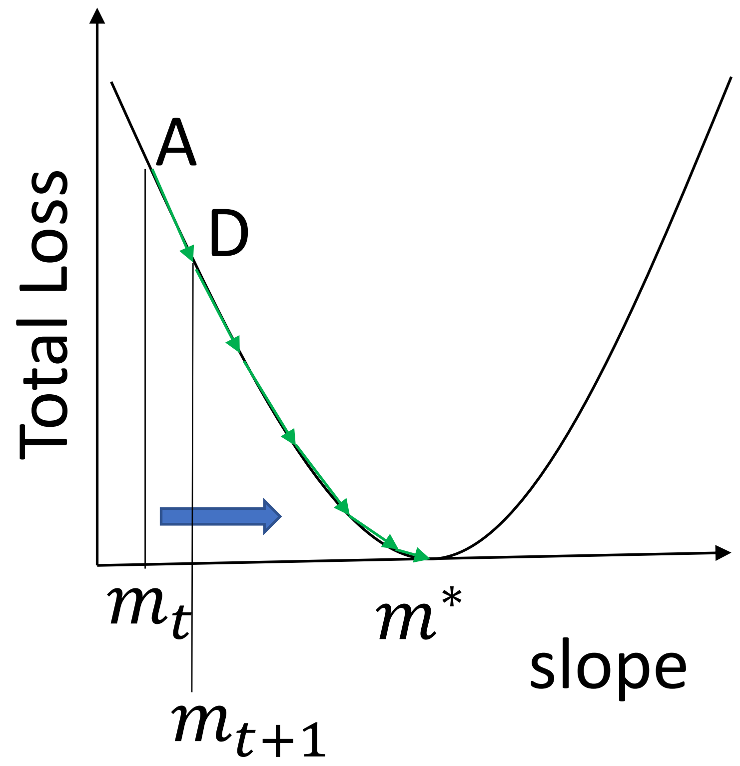
Defining Problem Statement

- Linear Function Approximation
  - Gradient Descent (GD)

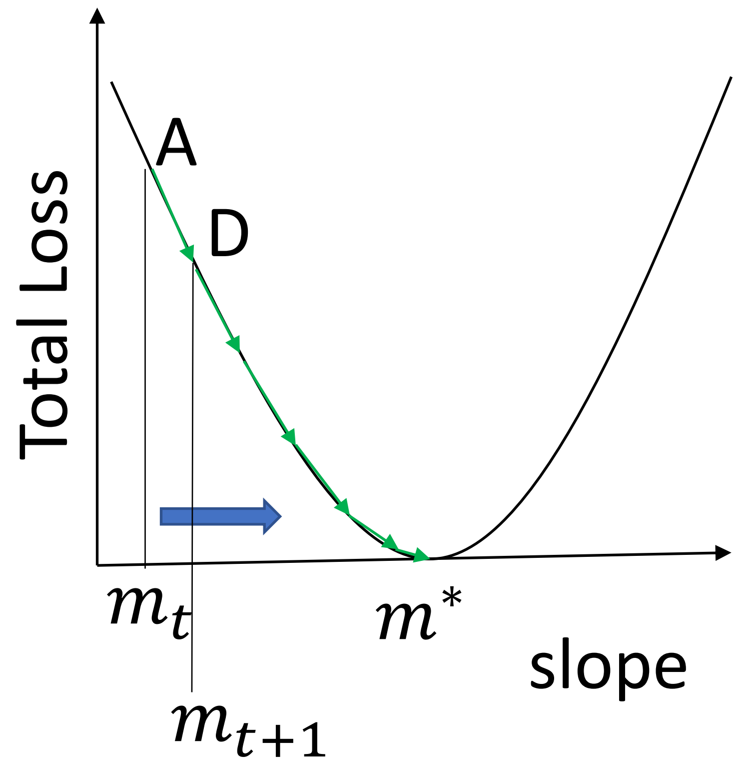
$$\boxed{m_{t+1} = m_t - \beta^{-1} \nabla L(m_t)}$$

  - Taylor Series Expansion

$$L(m_{t+1}) = L(m_t) + \langle \nabla L(m_t), m_{t+1} - m_t \rangle + \frac{1}{2}(m_{t+1} - m_t)^T \nabla^2 L(m_t)(m_{t+1} - m_t)$$



Defining Problem Statement



- Linear Function Approximation
  - Gradient Descent (GD)

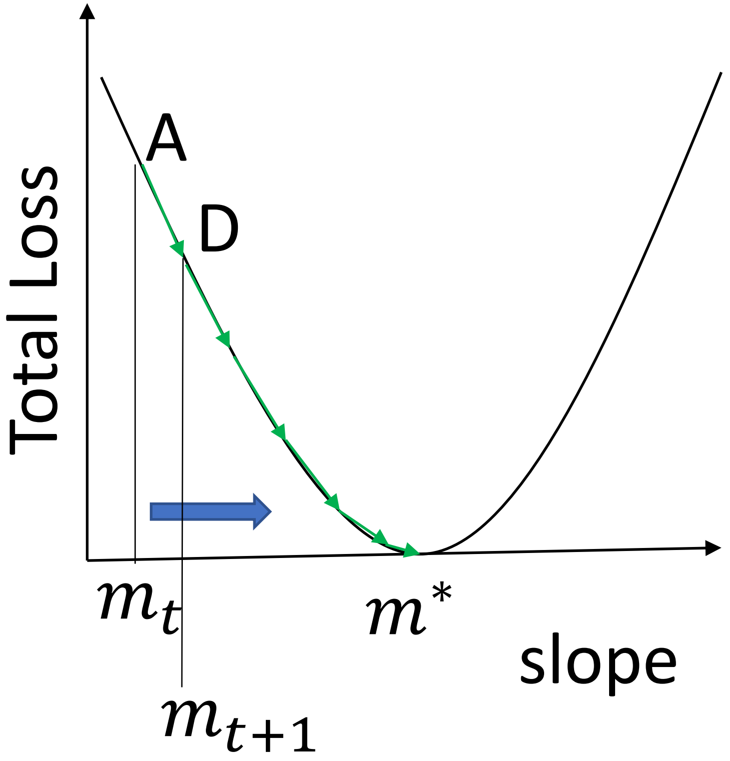
$$\boxed{m_{t+1} = m_t - \beta^{-1} \nabla L(m_t)}$$

  - Taylor Series Expansion

$$L(m_{t+1}) = L(m_t) + \langle \nabla L(m_t), m_{t+1} - m_t \rangle + \frac{1}{2}(m_{t+1} - m_t)^T \nabla^2 L(m_t)(m_{t+1} - m_t)$$

$$\leq L(m_t) + \langle \nabla L(m_t), m_{t+1} - m_t \rangle + \frac{\beta}{2}\|m_{t+1} - m_t\|_2^2, \because \|\nabla^2 L(m_t)\|_2 \leq \beta$$

Defining Problem Statement



- Linear Function Approximation
  - Gradient Descent (GD)

$$\boxed{m_{t+1} = m_t - \beta^{-1} \nabla L(m_t)}$$

  - Taylor Series Expansion

$$L(m_{t+1}) = L(m_t) + \langle \nabla L(m_t), m_{t+1} - m_t \rangle + \frac{1}{2}(m_{t+1} - m_t)^T \nabla^2 L(m_t)(m_{t+1} - m_t)$$

$$\leq L(m_t) + \langle \nabla L(m_t), m_{t+1} - m_t \rangle + \frac{\beta}{2}\|m_{t+1} - m_t\|_2^2, \because \|\nabla^2 L(m_t)\|_2 \leq \beta$$

$$\leq L(m_t) - \beta^{-1}\|\nabla L(m_t)\|_2^2 + \frac{\beta^{-1}}{2}\|\nabla L(m_t)\|_2^2,$$

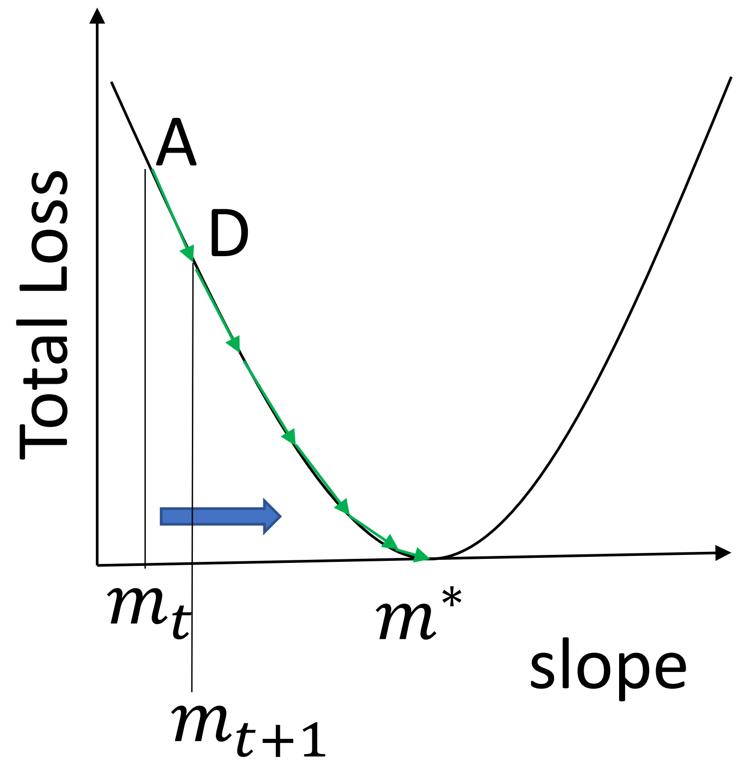
Defining Problem Statement

- Linear Function Approximation
  - Gradient Descent (GD)

$$\boxed{m_{t+1} = m_t - \beta^{-1} \nabla L(m_t)}$$

  - Taylor Series Expansion

$$\boxed{L(m_{t+1}) \leq L(m_t) - \frac{\beta^{-1}}{2} \|\nabla L(m_t)\|_2^2}$$



Defining Problem Statement

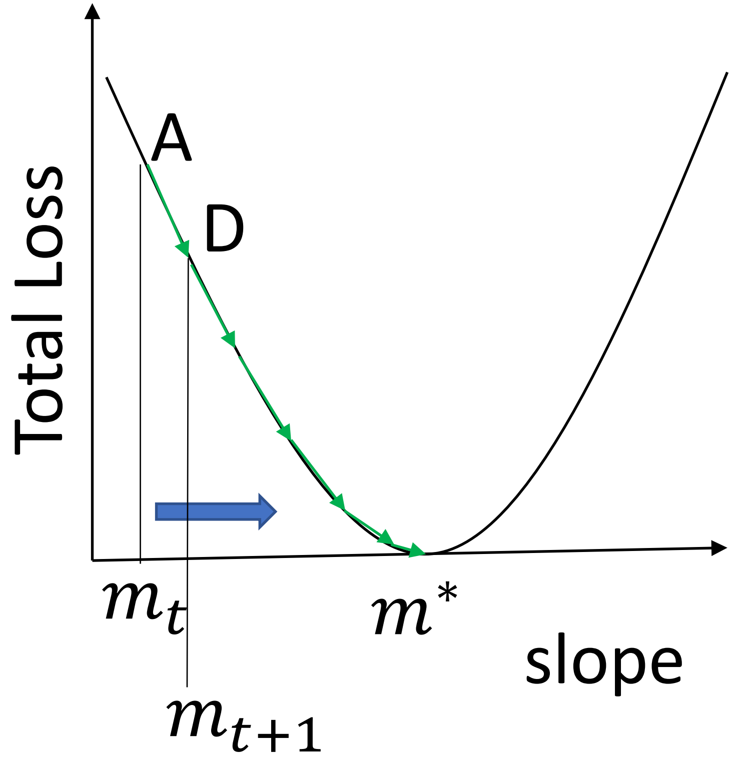
- Linear Function Approximation
  - Iteration Complexity

$$L(m_1) \leq L(m_0) - \frac{\beta^{-1}}{2} \|\nabla L(m_0)\|_2^2$$

$$L(m_2) \leq L(m_1) - \frac{\beta^{-1}}{2} \|\nabla L(m_1)\|_2^2$$

$$\vdots$$

$$L(m_T) \leq L(m_{T-1}) - \frac{\beta^{-1}}{2} \|\nabla L(m_{T-1})\|_2^2$$



Defining Problem Statement

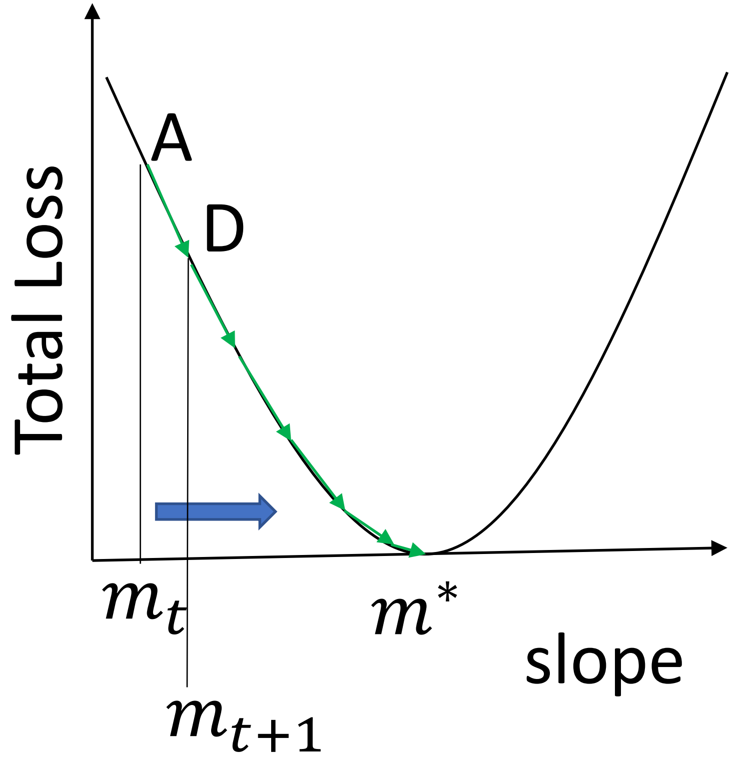
- Linear Function Approximation
  - Iteration Complexity

$$L(\cancel{m_1}) \leq L(m_0) - \frac{\beta^{-1}}{2}\|\nabla L(m_0)\|_2^2$$

$$L(\cancel{m_2}) \leq L(\cancel{m_1}) - \frac{\beta^{-1}}{2}\|\nabla L(m_1)\|_2^2$$

$$\vdots$$

$$L(m_T) \leq L(\cancel{m_{T-1}}) - \frac{\beta^{-1}}{2}\|\nabla L(m_{T-1})\|_2^2$$

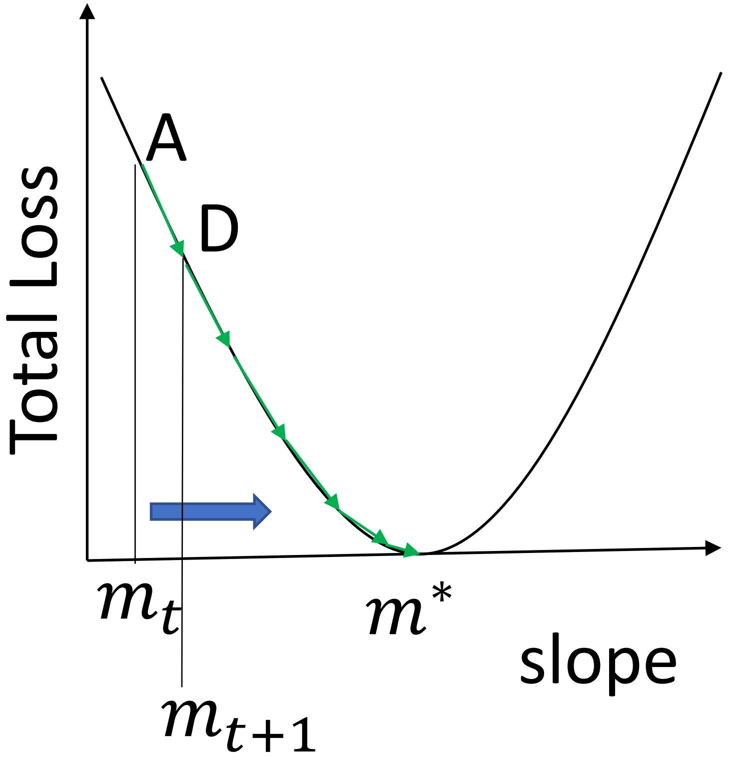


Defining Problem Statement

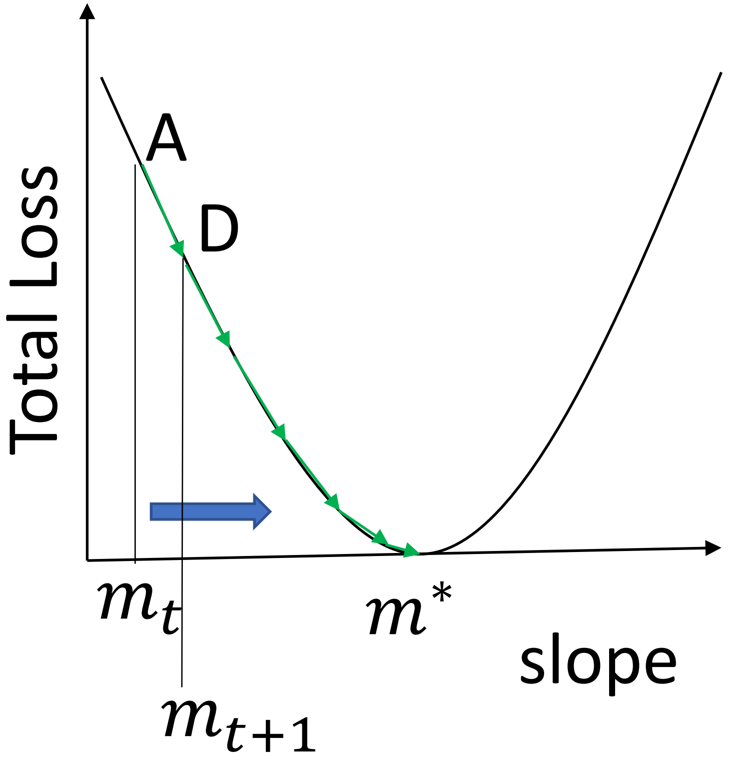
- Linear Function Approximation
  - Iteration Complexity

$$L(m_T) \leq L(m_0) - \frac{\beta^{-1}}{2} \sum_{t=0}^{T-1} \|\nabla L(m_t)\|_2^2$$

$$L(m_0) - L(m_T) \geq \frac{\beta^{-1}}{2} \sum_{t=0}^{T-1} \|\nabla L(m_t)\|_2^2$$



Defining Problem Statement

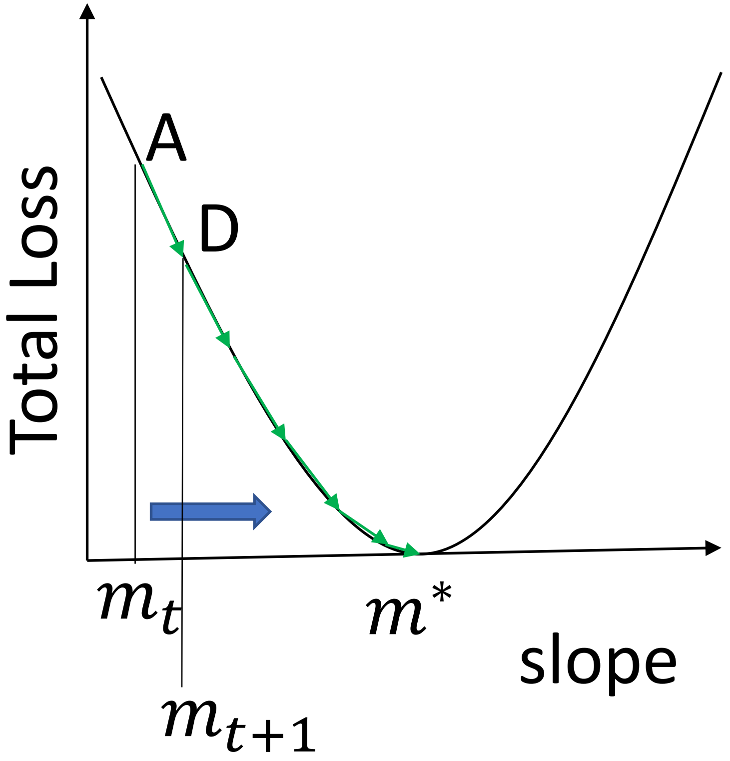


- Linear Function Approximation
  - $\in$-Stationary Solution

$\|\nabla L(m_T)\|_2 \leq \in$ ⟹ For all $t = 0, \ldots, T-1,$

$\|\nabla L(m_t)\|_2 > \in$

Defining Problem Statement



- Linear Function Approximation
  - ∈-Stationary Solution

$$\|\nabla L(m_T)\|_2 \leq \epsilon \Longrightarrow \text{For all } t = 0, \dots, T-1,$$

$$\|\nabla L(m_t)\|_2 > \epsilon$$

  - Iteration Complexity

$$L(m_0) - L(m_T) \geq \frac{\beta^{-1}}{2} \sum_{t=0}^{T-1} \|\nabla L(m_t)\|_2^2$$

Defining Problem Statement

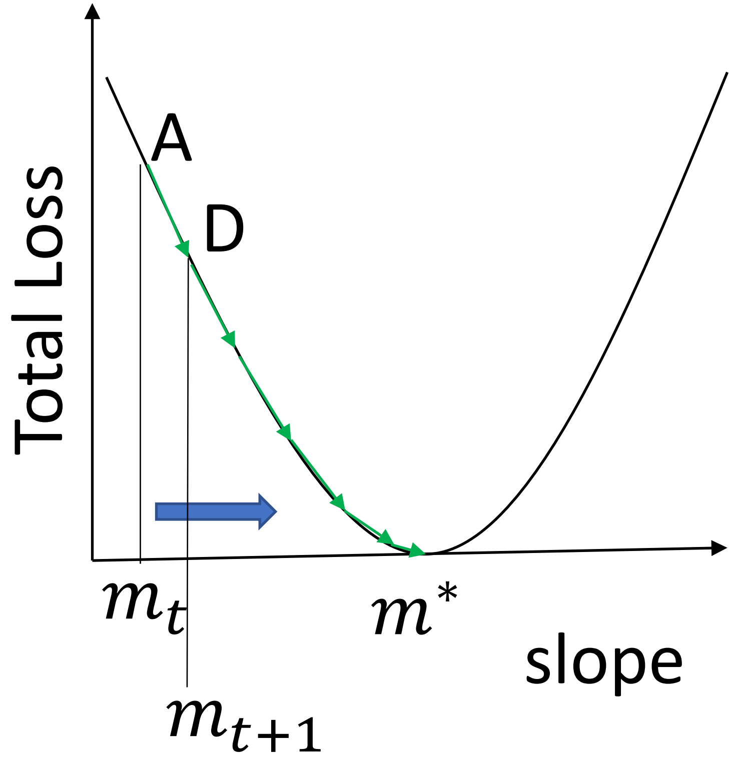
- Linear Function Approximation
  - $\in$-Stationary Solution

$\|\nabla L(m_T)\|_2 \leq \in$ ⟹ For all $t = 0, \dots, T-1,$

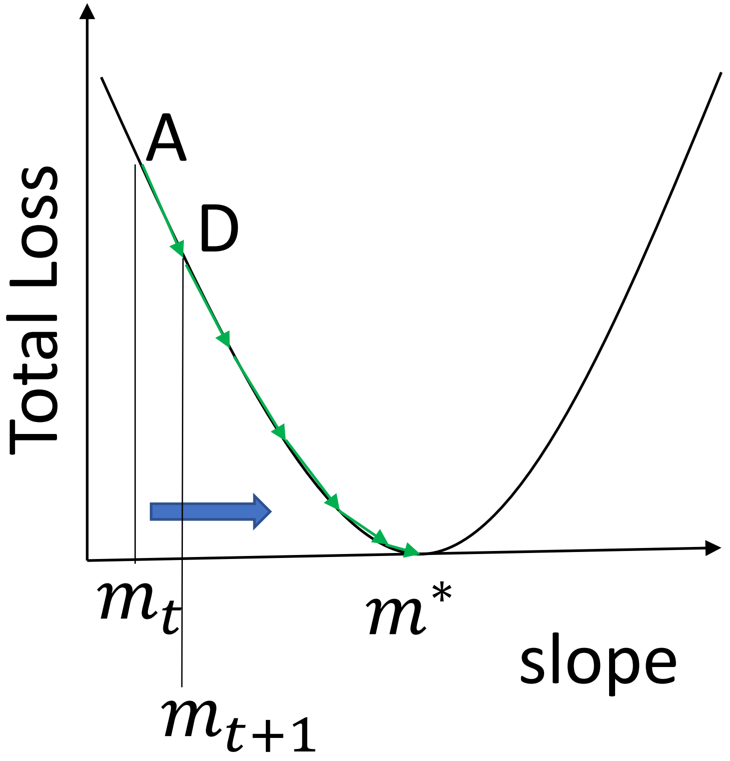
$\|\nabla L(m_t)\|_2 > \in$

  - Iteration Complexity

$$L(m_0) - L(m_T) \geq \frac{\beta^{-1}}{2} \sum_{t=0}^{T-1} \in^2$$



Defining Problem Statement



- Linear Function Approximation
  - ∈-Stationary Solution

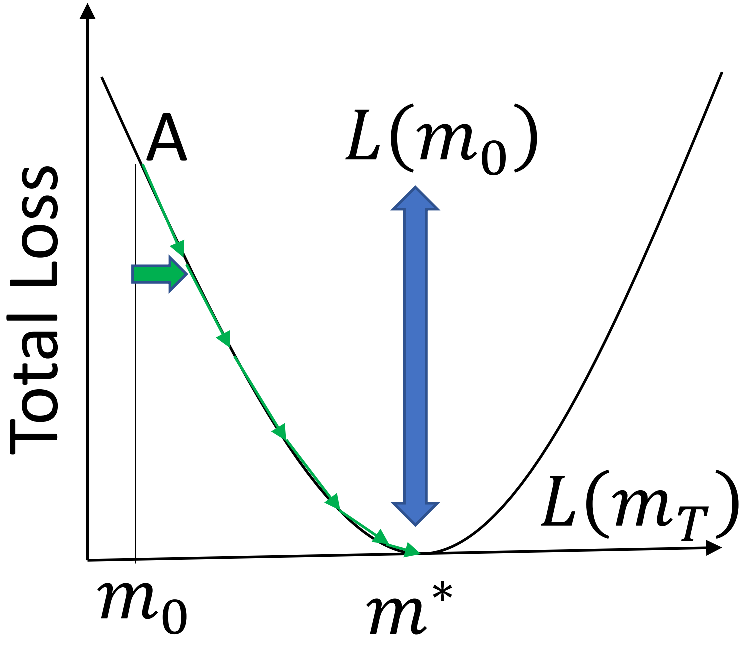
$$\|\nabla L(m_T)\|_2 \leq \in \Longrightarrow \text{For all } t = 0, \ldots, T-1,$$

$$\|\nabla L(m_t)\|_2 > \in$$

  - Iteration Complexity

$$L(m_0) - L(m_T) \geq \frac{\beta^{-1}}{2} T \in^2$$

Defining Problem Statement



- Linear Function Approximation
  - ∈-Stationary Solution

$\|\nabla L(m_T)\|_2 \leq \in$ ⟹ For all $t = 0, \ldots, T-1,$

$$\|\nabla L(m_t)\|_2 > \in$$

  - Iteration Complexity

$$L(m_0) - L(m_T) \geq \frac{\beta^{-1}}{2} T \in^2$$

$$\boxed{T \leq \frac{2\beta\big(L(m_0) - L(m_T)\big)}{\in^2} = \mathcal{O}\left(\frac{1}{\in^2}\right)}$$

Defining Problem Statement

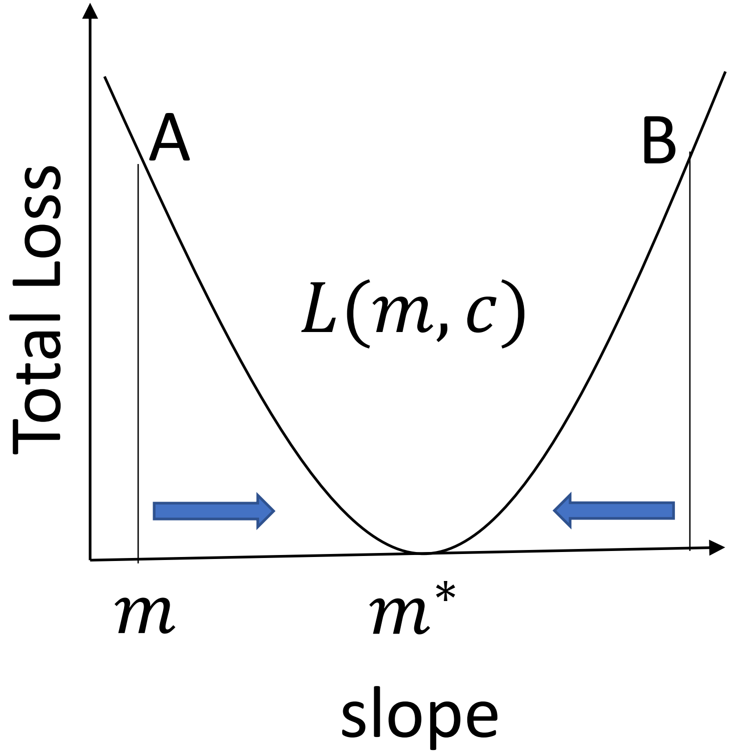
- Linear Function Approximation

    - Stochastic Gradient Descent (SGD)

$$L(m,c) = \frac{1}{2n}\sum_{p=1}^{n}\left(f\left(m,c,x_p\right) - y_p\right)^2$$

$$L_{\mathcal{B}}(m,c) = \frac{1}{2|\mathcal{B}|}\sum_{p=1}^{|\mathcal{B}|}\left(f\left(m,c,x_p\right) - y_p\right)^2$$

$$m(t+1) = m(t) - \eta\,\frac{\partial L_{\mathcal{B}}(m,c)}{\partial m(t)}$$



Defining Problem Statement

- Linear Function Approximation

$$l_p(m,c) = \left(f\left(m,c,x_p\right) - y_p\right)^2$$

  - Stochastic Gradient Descent (SGD)

$$L(m,c) = \frac{1}{2n}\sum_{p=1}^{n}\left(f\left(m,c,x_p\right) - y_p\right)^2$$

$$L(m,c) = \frac{1}{2n}\left(l_1(m,c) + l_2(m,c) + \cdots + l_n(m,c)\right)$$

Defining Problem Statement

- Linear Function Approximation

$$l_p(m,c) = \left(f\left(m,c,x_p\right) - y_p\right)^2$$

  - Stochastic Gradient Descent (SGD)

$$L(m,c) = \frac{1}{2n}\sum_{p=1}^{n}\left(f\left(m,c,x_p\right) - y_p\right)^2$$

$$L(m,c) = \frac{1}{2n}\left(l_1(m,c) + l_2(m,c) + \cdots + l_n(m,c)\right)$$

$$L_r(m,c) = \frac{1}{2}\left(l_r(m,c)\right)$$

Defining Problem Statement

- Linear Function Approximation

$$l_p(m,c) = \left(f(m,c,x_p) - y_p\right)^2$$

  - Stochastic Gradient Descent (SGD)

$$L(m,c) = \frac{1}{2n}\sum_{p=1}^{n}\left(f(m,c,x_p) - y_p\right)^2$$

$$L(m,c) = \frac{1}{2n}\left(l_1(m,c) + l_2(m,c) + \cdots + l_n(m,c)\right)$$

$$L_r(m,c) = \frac{1}{2}\left(l_r(m,c)\right)$$

$$m(t+1) = m(t) - \eta\,\frac{\partial L_r(m,c)}{\partial m(t)}$$

Defining Problem Statement

- Linear Function Approximation

$$l_p(m,c) = \left(f\left(m,c,x_p\right) - y_p\right)^2$$

  - Stochastic Gradient Descent (SGD)

$$L(m,c) = \frac{1}{2n}\left(l_1(m,c) + l_2(m,c) + \cdots + l_n(m,c)\right)$$

$$L_{\mathcal{B}}(m,c) = \frac{1}{2|\mathcal{B}|}\sum_{p=1}^{|\mathcal{B}|} l_p(m,c)$$

$$m(t+1) = m(t) - \eta\,\frac{\partial L_{\mathcal{B}}(m,c)}{\partial m(t)}$$

Three Pillars of Deep Learning

- Setting Up DL Environment
- Defining Problem Statement
- Implementation Details

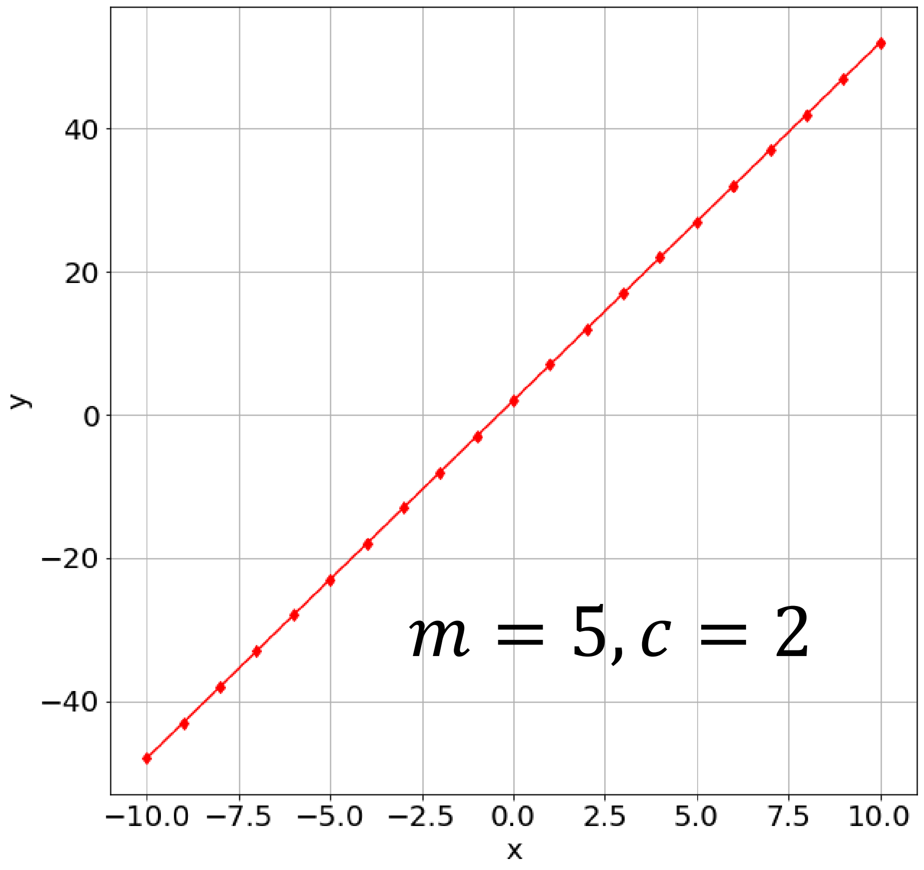
Implementation Details

- Linear Function Approximator
- One Layer Neural Network Function Approximator
- Two Layer Neural Network Function Approximator
- Three Layer Convolutional Neural Network Function Approximator

Implementation Details

Paired Training Data

- Linear Function Approximation

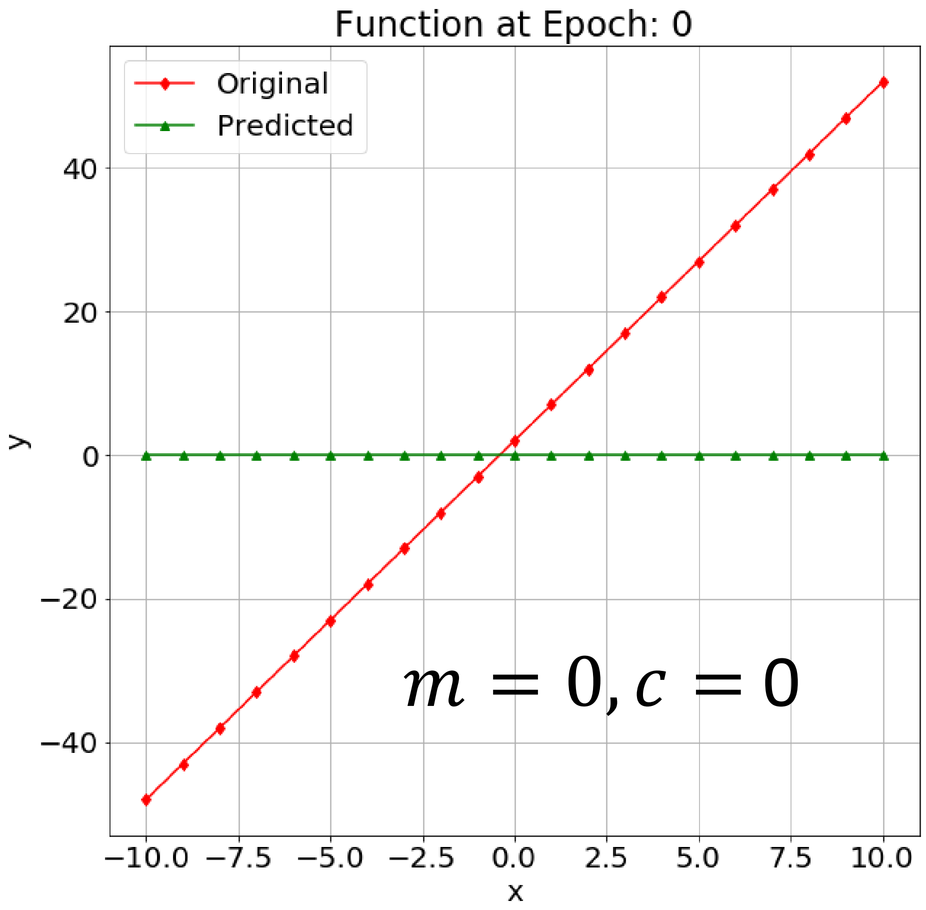


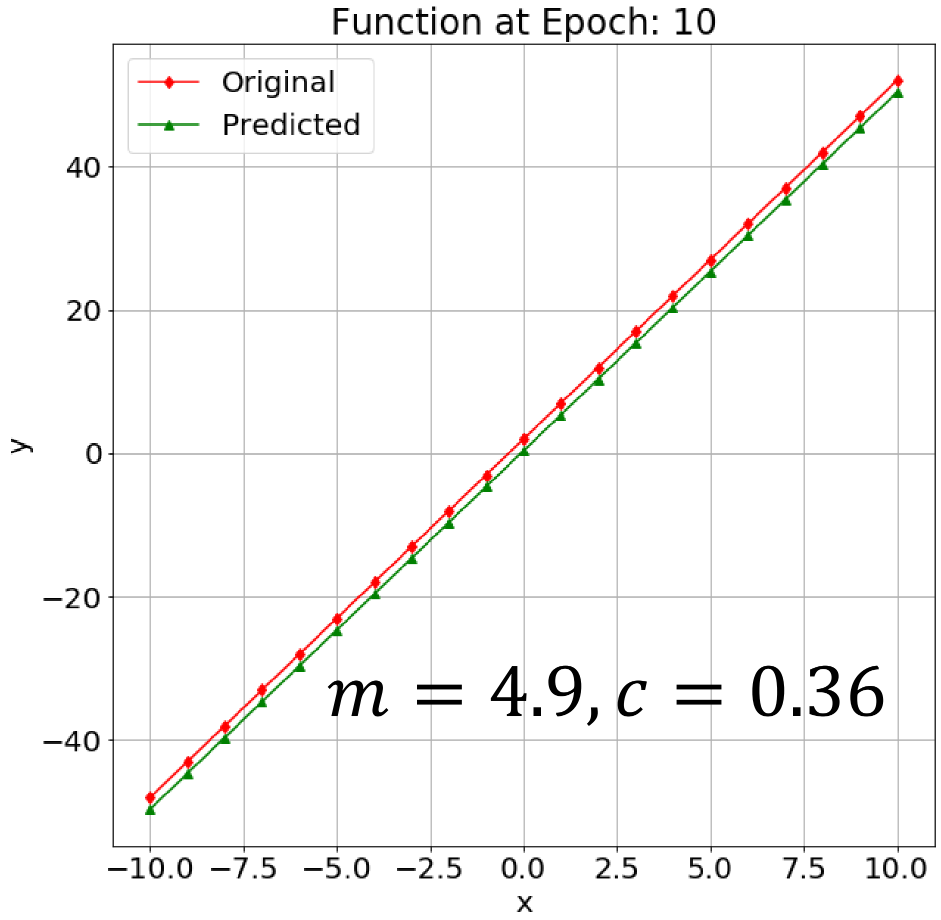
$f(.)$

x	y
-10	-48
-9	-43
…	…
10	52

Implementation Details

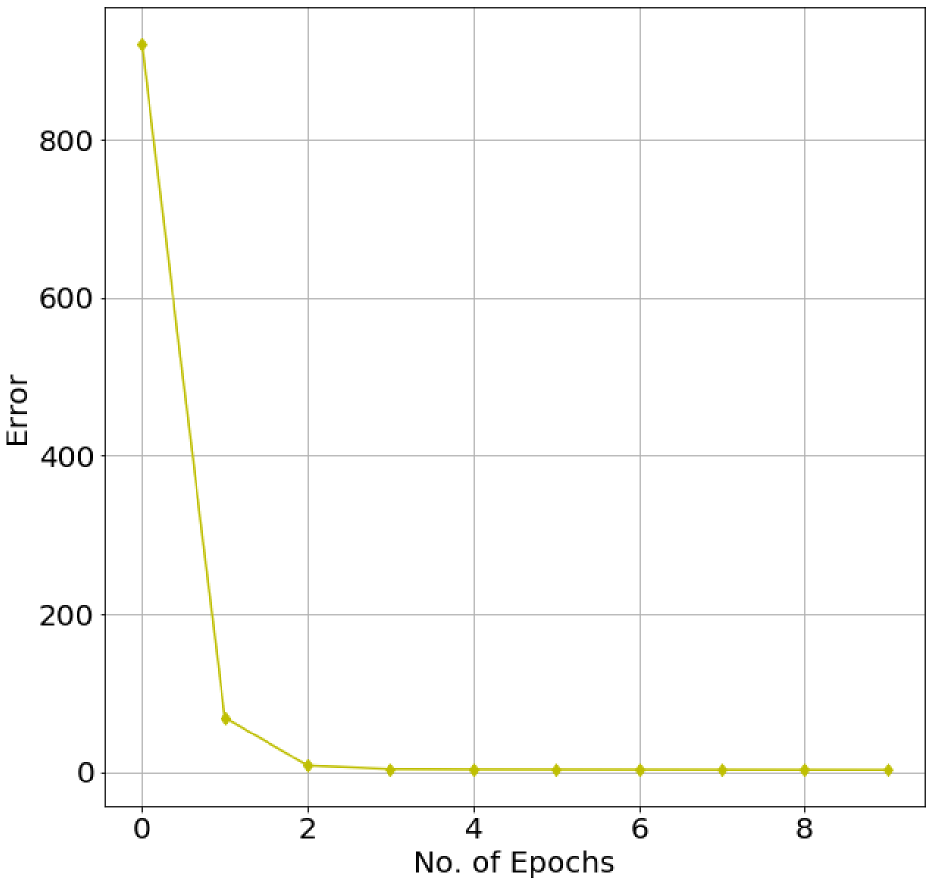
- Linear Function Approximation



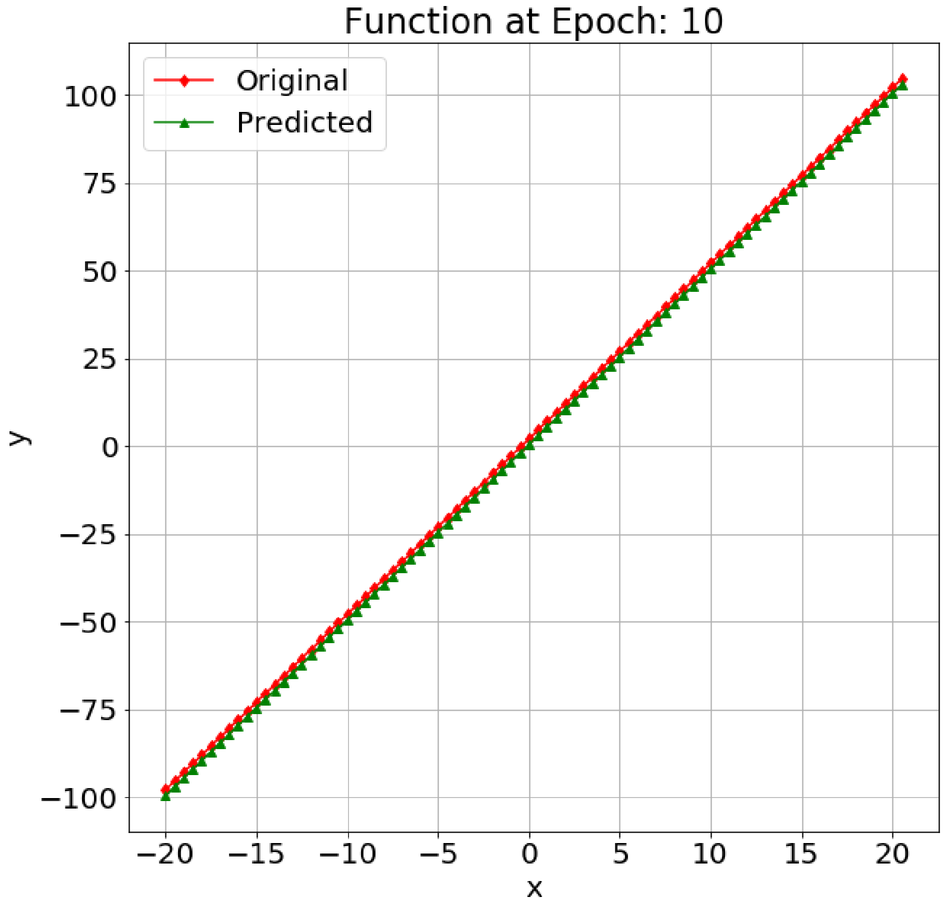


Implementation Details

- Linear Function Approximation



Inference Stage



Demo code 1

Implementation Details

- Linear Function Approximator
- One Layer Neural Network Function Approximator
- Two Layer Neural Network Function Approximator
- Three Layer Convolutional Neural Network Function Approximator

Implementation Details

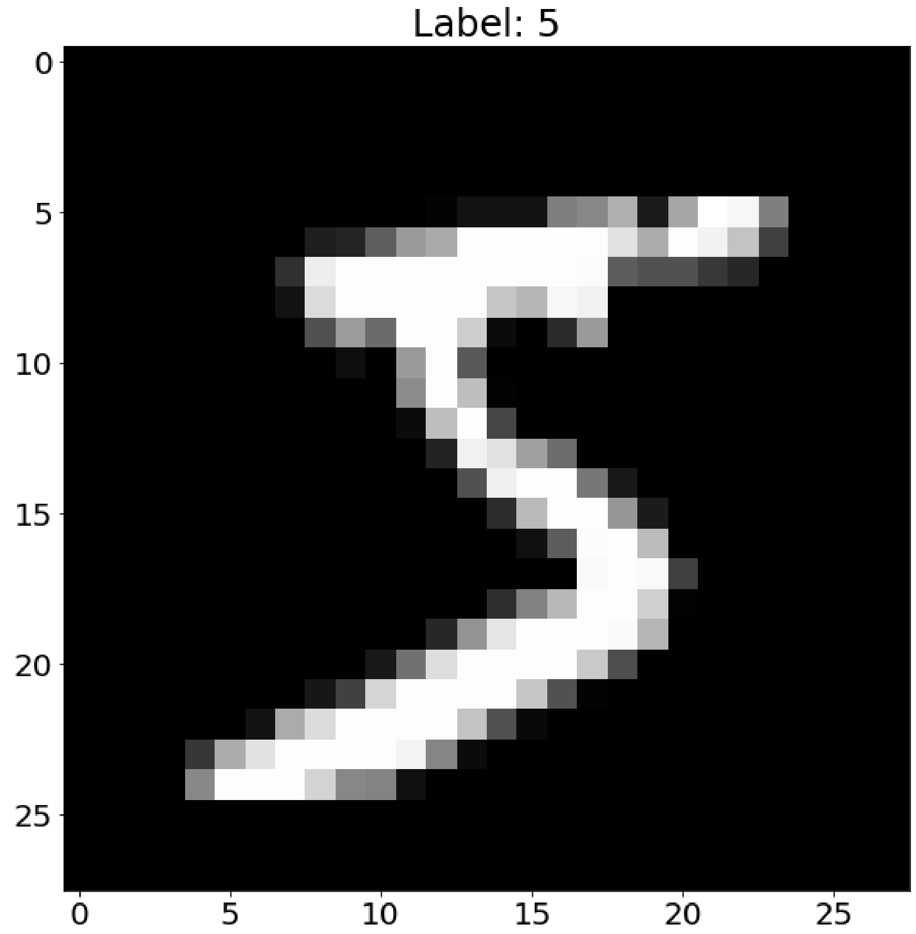
- One Layer Neural Network Function Approximator
  - Dataset Preparation

$$\{(x_p, y_p)\}_{p=1}^{n} \subset R^{d_{in} \times d_{out}}$$

$$d_{in} = 28 \times 28 = 784$$

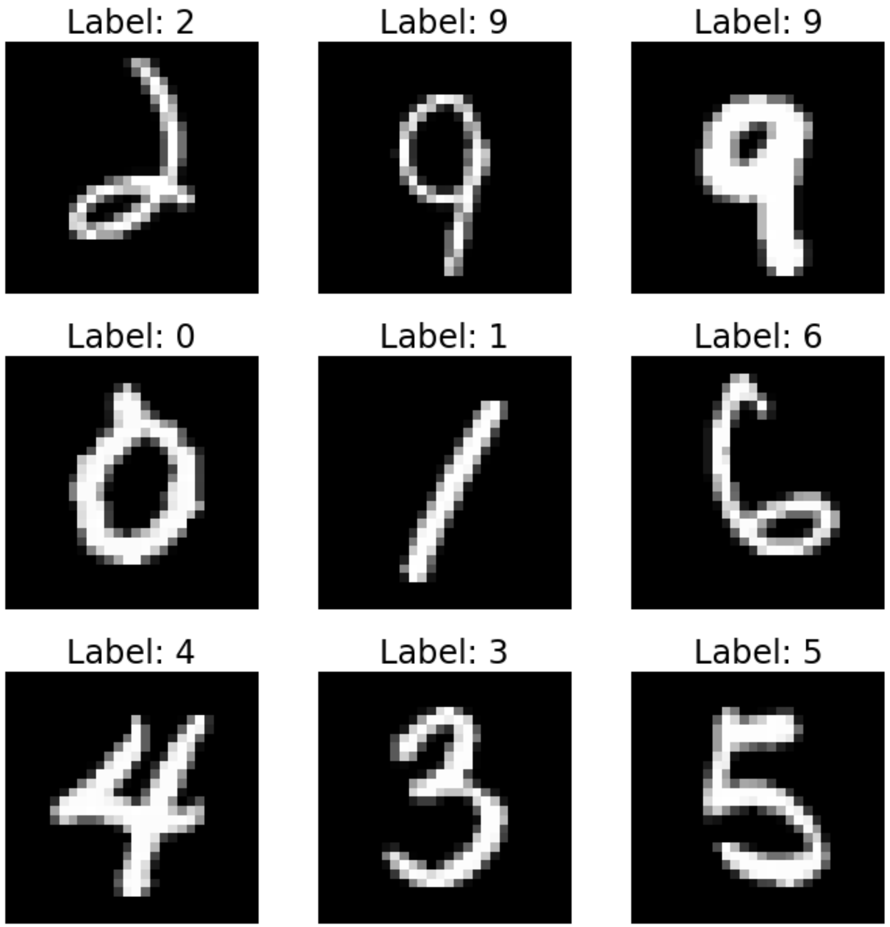
$$d_{out} = 10$$

$$n = 60000$$



Implementation Details

- One Layer Neural Network Function Approximator



$f(.)$

$x = (x_1, x_2, \dots, x_{784})$	$y = (y_1, y_2, \dots, y_{10})$
$(0,0.5, \dots, 1)$	$(1,0, \dots, 0)$
$(0.8,1, \dots, 0)$	$(0,1, \dots, 0)$
…	…
$(1,0, \dots, 0.2)$	$(0,0, \dots, 1)$

Implementation Details

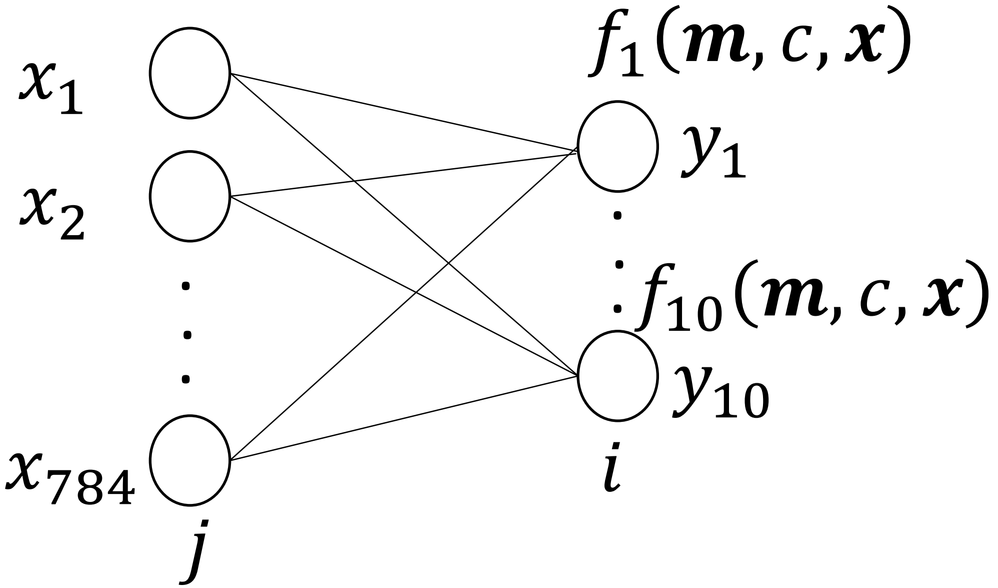
- One Layer Neural Network Function Approximator
  - Function Approximator

$$f_i(\boldsymbol{m}, c, \boldsymbol{x}) = m_1 x_1 + m_2 x_2 + \cdots + m_{784} x_{784} + c_i$$

$$= \sum_{j=1}^{784} m_j x_j + c_j$$

$$\boldsymbol{f}(f_1, f_2, \ldots, f_{10}) = M\boldsymbol{x} + \boldsymbol{c}$$

[10x1] = [10x784][784x1] + [10x1]



Implementation Details

- One Layer Neural Network Function Approximator
  - Function Approximator

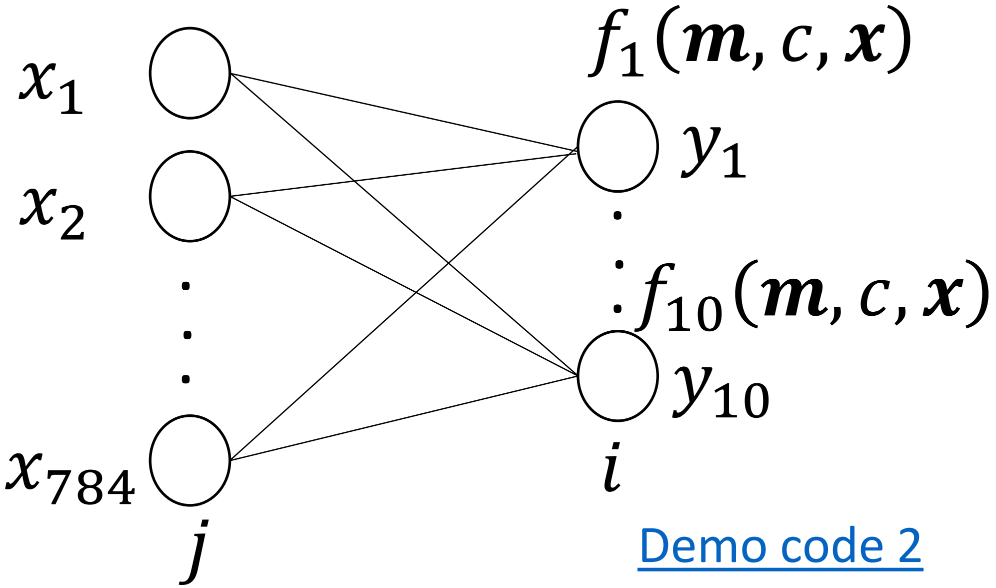
$$f_i(\boldsymbol{m}, c, \boldsymbol{x}) = m_1 x_1 + m_2 x_2 + \cdots + m_{784} x_{784} + c_i$$

$$= \sum_{j=1}^{784} m_j x_j + c_j$$

Trainable Parameters
10x784+10 = 7850

$$\boldsymbol{f}(f_1, f_2, \ldots, f_{10}) = M\boldsymbol{x} + \boldsymbol{c}$$

[10x1] = [10x784][784x1] + [10x1]



Demo code 2

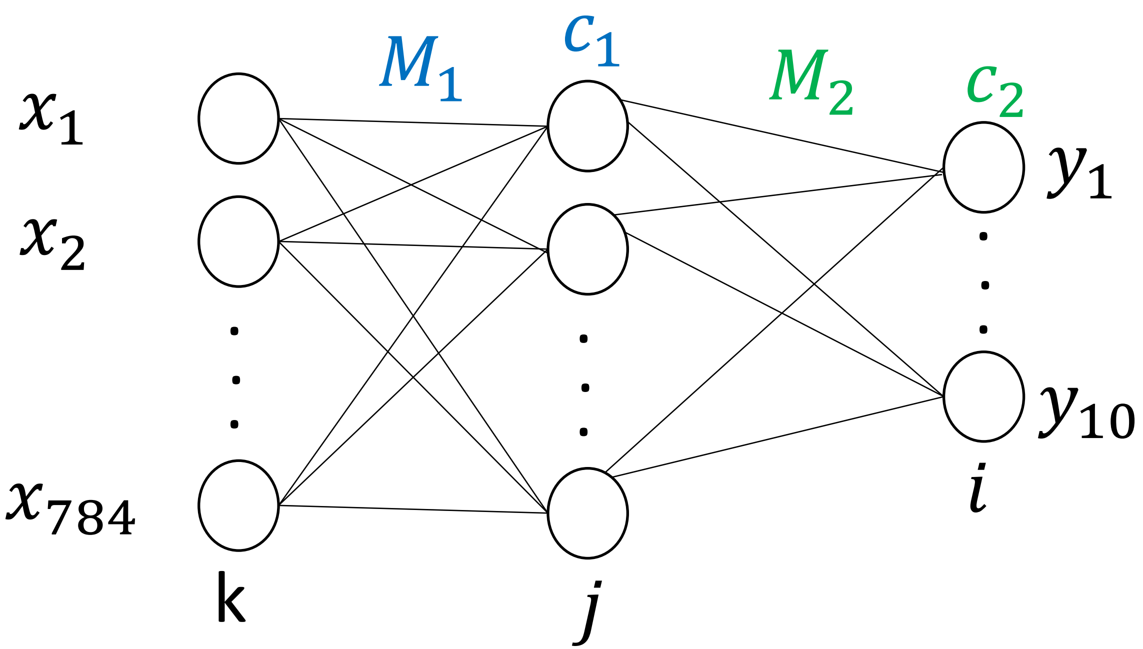
Implementation Details

- Linear Function Approximator
- One Layer Neural Network Function Approximator
- Two Layer Neural Network Function Approximator
- Three Layer Convolutional Neural Network Function Approximator

Implementation Details

- Two Layer Neural Network Function Approximator
  - Function Approximator

$$\boldsymbol{f}(f_1, f_2, \dots, f_{10}) = M_2(M_1\boldsymbol{x} + c_1) + \boldsymbol{c_2}$$

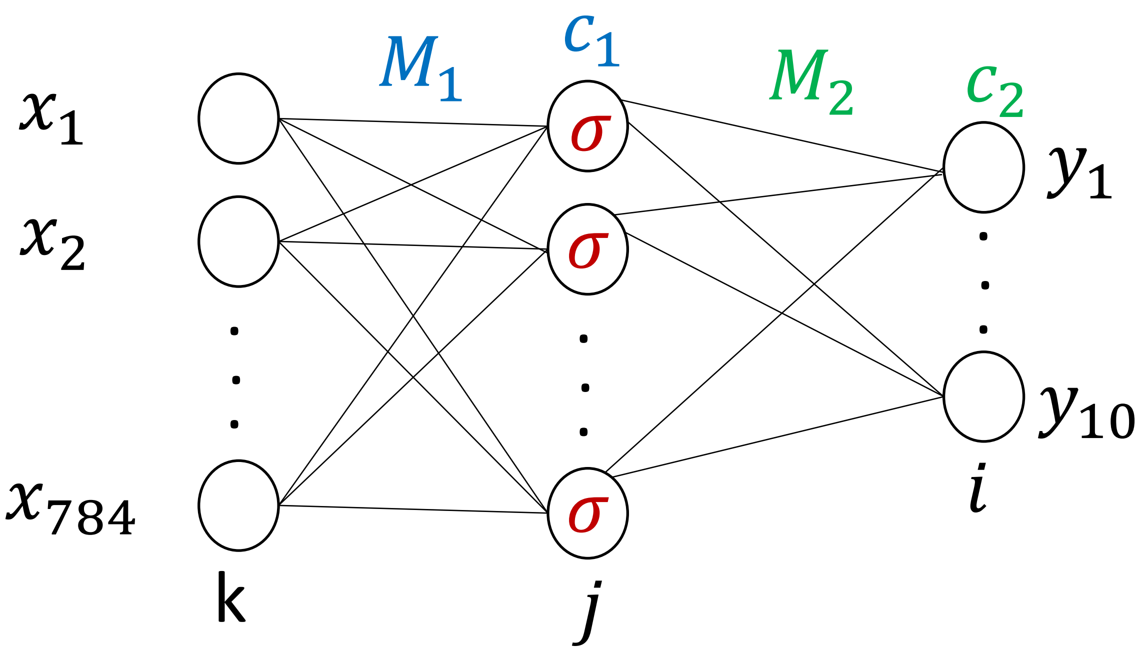


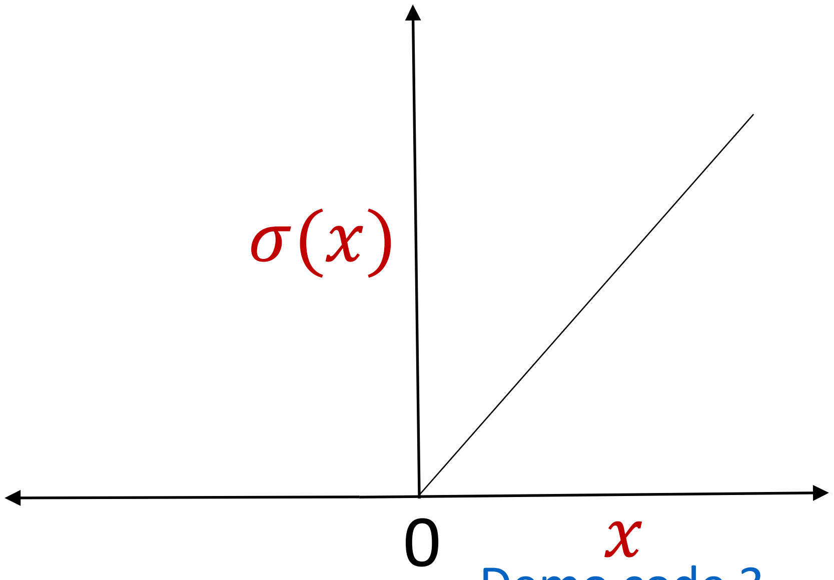
Implementation Details

- Two Layer Neural Network Function Approximator
  - Function Approximator

$$\sigma(x) = x, if\ x \geq 0$$
$$= 0, otherwise$$

$$\boldsymbol{f}(f_1, f_2, \ldots, f_{10}) = M_2(\sigma(M_1\boldsymbol{x} + c_1)) + \boldsymbol{c_2}$$





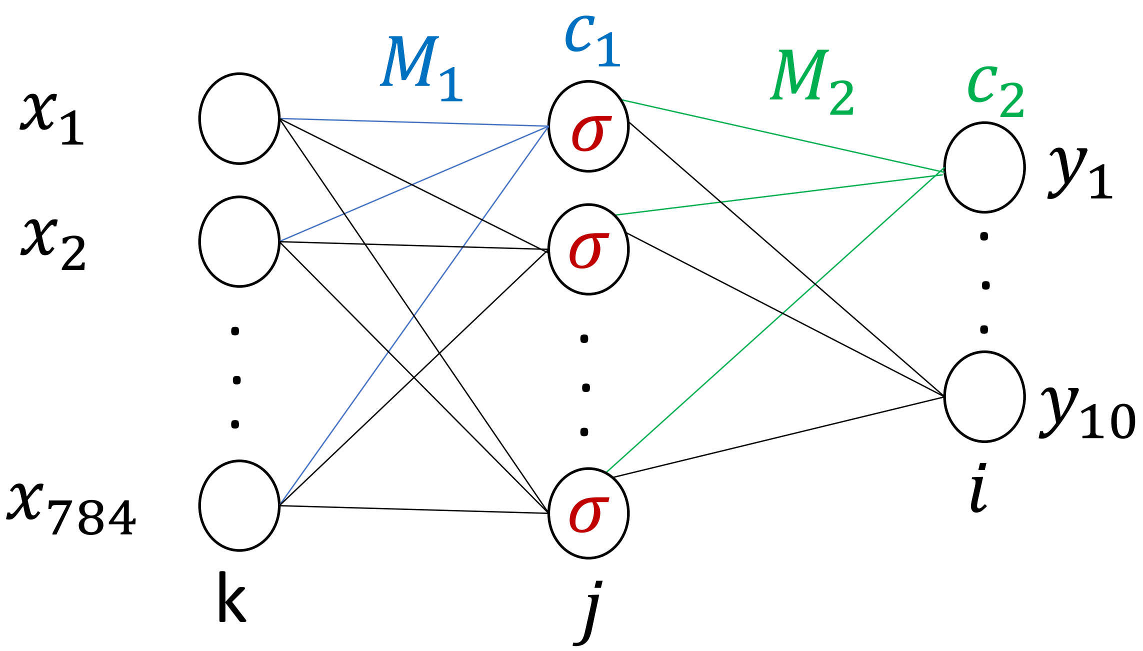
Demo code 3

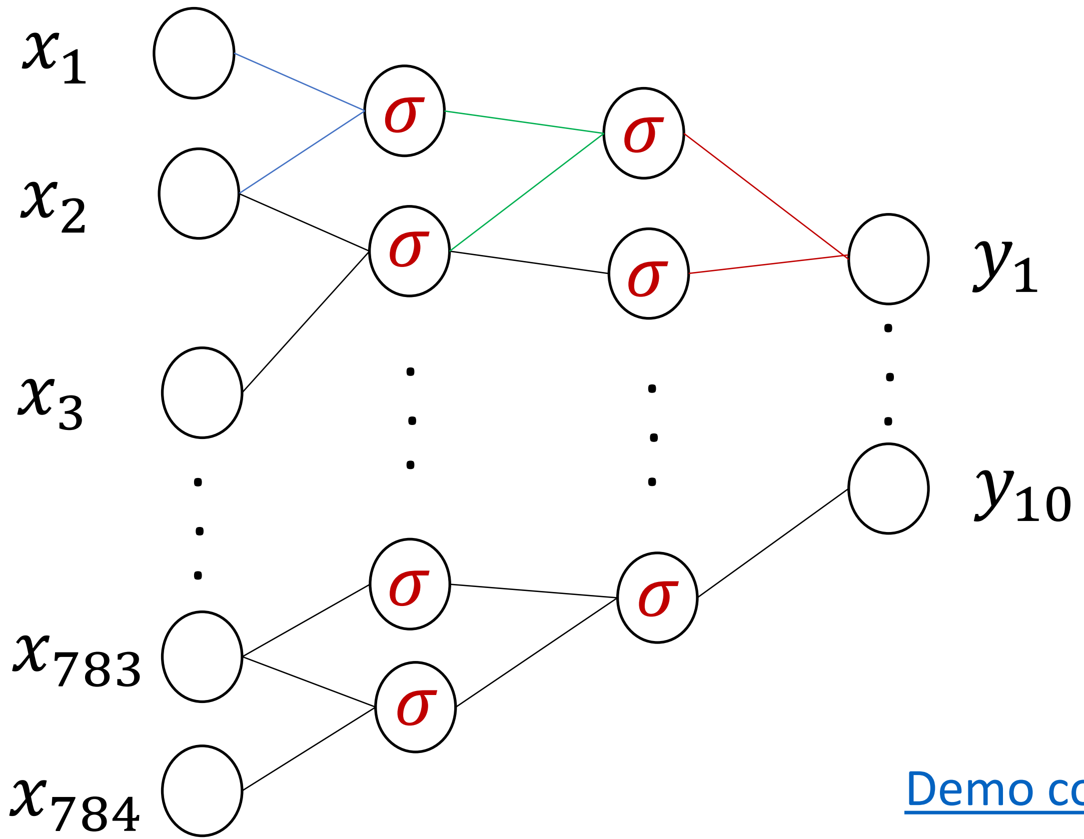
Implementation Details

- Linear Function Approximator
- One Layer Neural Network Function Approximator
- Two Layer Neural Network Function Approximator
- Three Layer Convolutional Neural Network Function Approximator

Implementation Details

- Three Layer Convolutional Neural Network Function Approximator
  - Local Connectivity
  - Weight Sharing



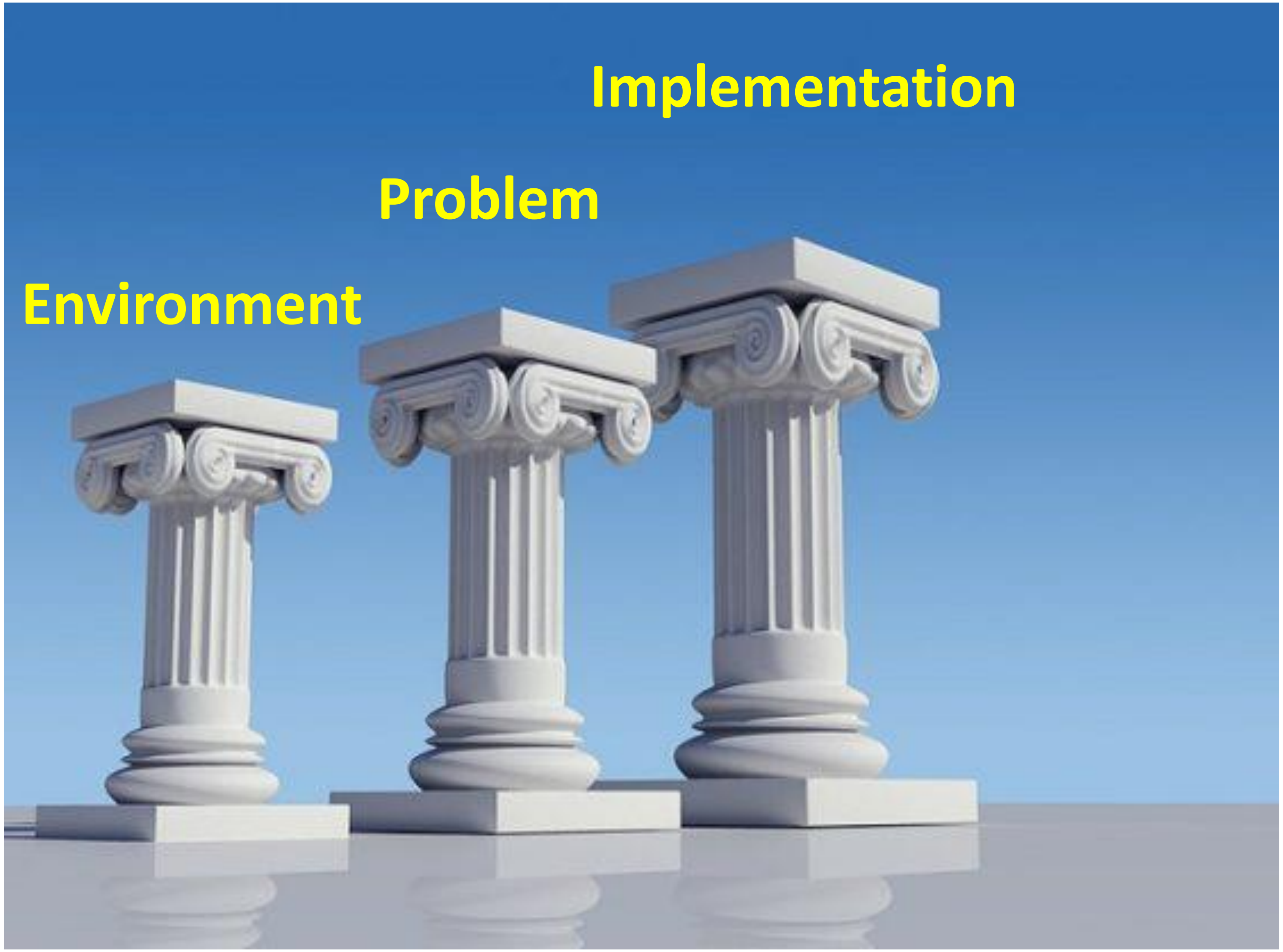


Demo code 4

Implementation Details

- Linear Function Approximator
- One Layer Neural Network Function Approximator
- Two Layer Neural Network Function Approximator
- Three Layer Convolutional Neural Network Function Approximator

Three Pillars of Deep Learning



Why Adversarial Interaction Creates Non-Homogeneous Patterns: A Pseudo-Reaction Diffusion Model for Turing Instability

- Adversarial Interaction
  - Generative Adversarial Networks (GANs)
  - Application of conditional GANs
- Non-Homogeneous Patterns
  - Homogeneous patterns
  - Supervised learning
- Reaction-Diffusion
  - Turing's RD model (1952)
  - Gray-Scott RD model (1984)
- Turing Instability
  - Reaction dynamics
  - Diffusion dynamics