CSCI 1515 Applied Cryptography

This Lecture:

- Private Set Intersection
- Privacy-Preserving Machine Learning

Private Set Intersection (PSI)

PSI: $f(X, Y) = X \cap Y$

PSI-CA: $f(X, Y) = |X \cap Y|$

Applications:

- Password Breach Alert (Chrome, Edge, Firefox, iOS keychain, ...)
- Ads Conversion Measurement (Google)
- Privacy-Preserving Inventory Matching (J.P. Morgan)
- Private Database Joins

Private Set Intersection (PSI)

Input: $X = \{x_1, x_2, \cdots, x_n\}$

Input: $Y = \{y_1, y_2, \cdots, y_n\}$

$H(x_1), \cdots, H(x_n)$ →

? ← $H(x')$

$H(y_1), \cdots, H(y_n)$

⇓

$X \cap Y$

Is it (semi-honest) secure?

Is it possible to achieve 2PC / MPC with 1 round of communication? NO!

x f y

m →

→ $f(x, y_1)$

$f(x, y_2)$

⋮

DDH-based PSI

Cyclic group G of order q with generator g, where DDH holds.

$H: \{0,1\}^* \to G$ (modeled as Random Oracle)

Input: $X = \{x_1, x_2, \cdots, x_n\}$

$a \xleftarrow{\$} \mathbb{Z}_q$

Input: $Y = \{y_1, y_2, \cdots, y_n\}$

$b \xleftarrow{\$} \mathbb{Z}_q$

$$\xleftarrow{H(Y)^b := \{H(y_1)^b, \cdots, H(y_n)^b\}}$$

$$\xrightarrow{H(x)^a,\ H(Y)^{a\cdot b}}$$

$H(x')^a = ?$

$H(x)^{a\cdot b} \cap H(Y)^{a\cdot b}$

$\Downarrow$

$X \cap Y$

Is it (semi-honest) secure?

PSI-CA?

PSI-CA: $f(X, Y) = |X \cap Y|$

Input: $X = \{x_1, x_2, \cdots, x_n\}$

$a \xleftarrow{\$} \mathbb{Z}_q$

Input: $Y = \{y_1, y_2, \cdots, y_n\}$

$b \xleftarrow{\$} \mathbb{Z}_q$

$$\xleftarrow{\quad H(Y)^b := \{H(y_1)^b, \cdots, H(y_n)^b\} \quad}$$

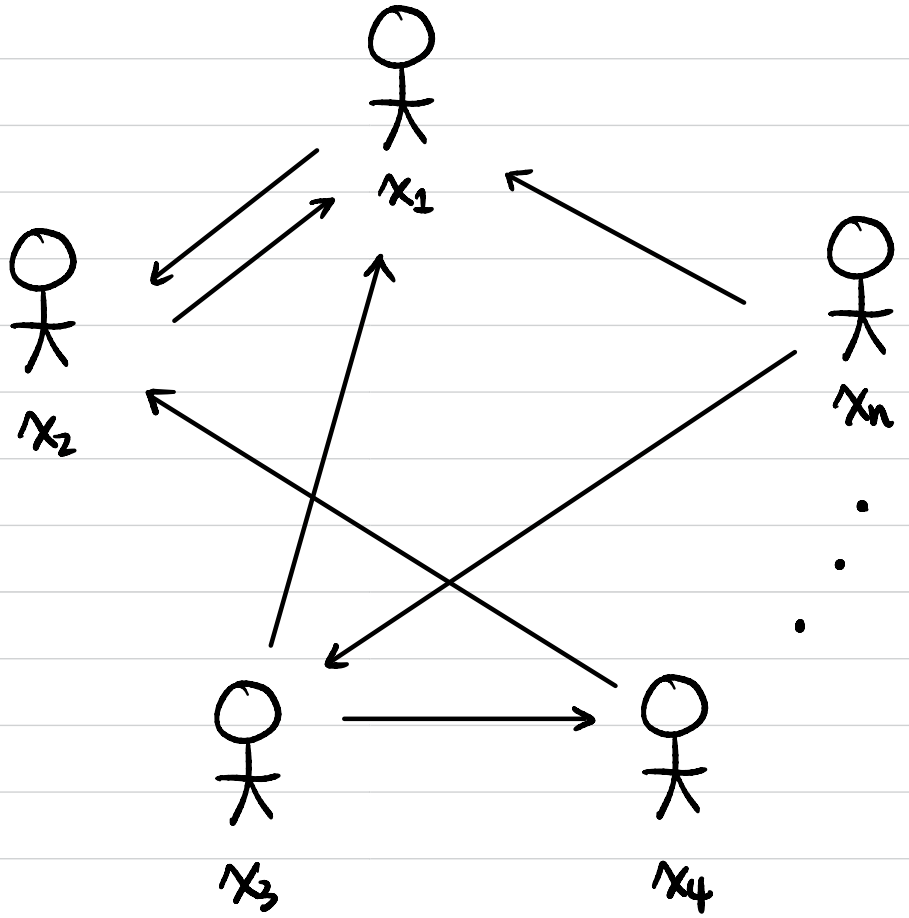
$$\xrightarrow{\quad H(x)^a, \{H(Y)^{a \cdot b}\}_{\text{shuffle}} \quad}$$

$H(x)^{a \cdot b} \cap \{H(Y)^{a \cdot b}\}_{\text{shuffle}}$

$\Downarrow$

$|X \cap Y|$

Privacy-Preserving Machine Learning (PPML)



⇓

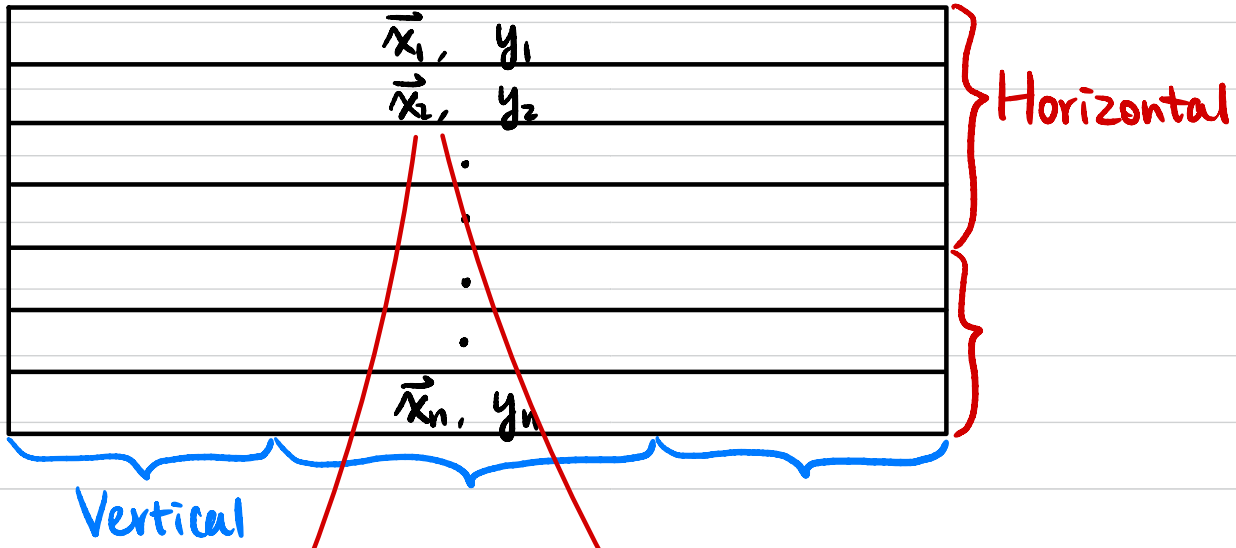
ML model

⇓

ML inference

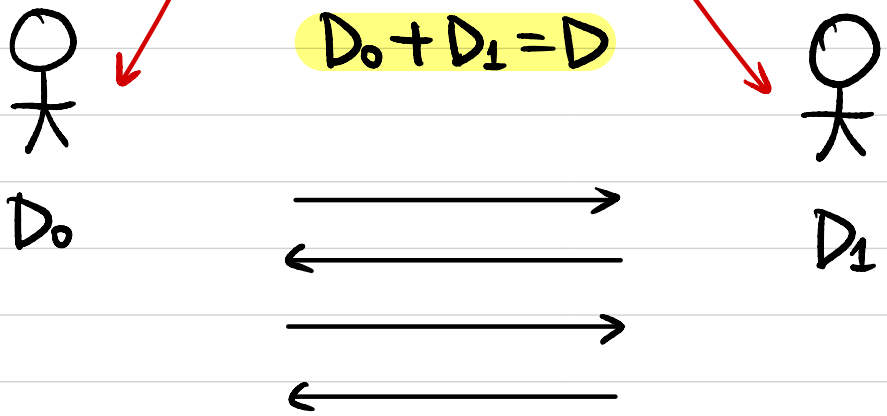
Privacy-Preserving Machine Learning (PPML)

Horizontal / Vertical Data Partitioning



Secret share

PPML Framework



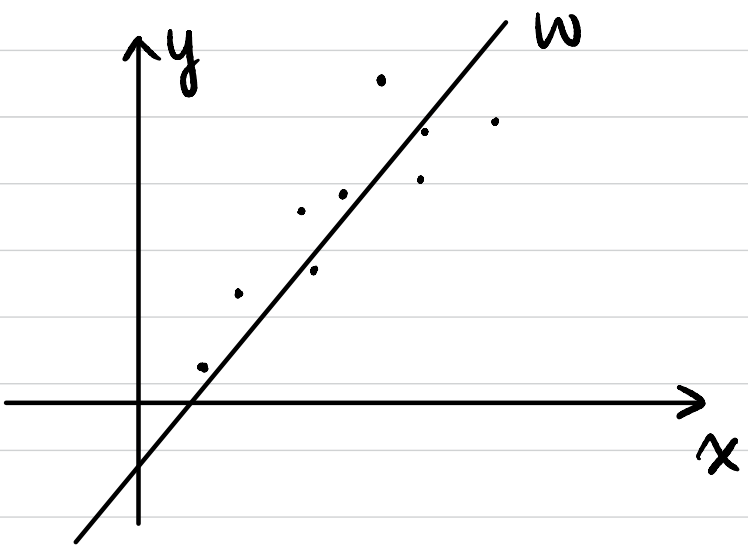
Machine Learning Background

Linear Regression

Data Points $(\vec{x}, y)$

ML Model: coefficient vector $\vec{w}$

$$g(\vec{x}) = \langle \vec{x}, \vec{w} \rangle$$



For each data point $(\vec{x}_i, y_i)$,

Define loss function

$$L_i(\vec{w}) := \frac{1}{2}(\langle \vec{x}_i, \vec{w} \rangle - y_i)^2$$

Total loss: $L(\vec{w}) := \frac{1}{N} \sum_{i \in [N]} L_i(\vec{w})$

Goal: Find $\vec{w}$ that minimizes $L(\vec{w})$.

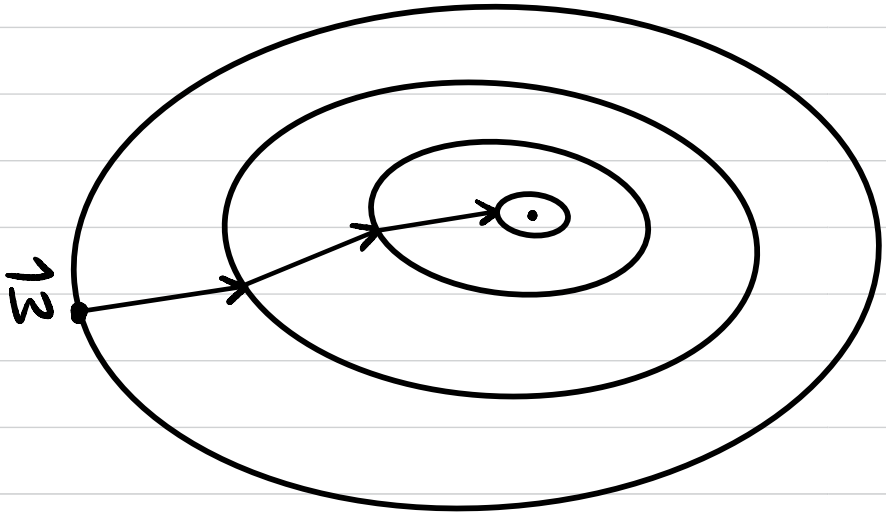
Machine Learning Background

Stochastic Gradient Descent (SGD)

- $\vec{w}$ initialized with arbitrary value
- Given a data point $(\vec{x}_i, y_i)$:

$$\vec{w} \leftarrow \vec{w} - \eta \cdot \nabla L_i(\vec{w})$$

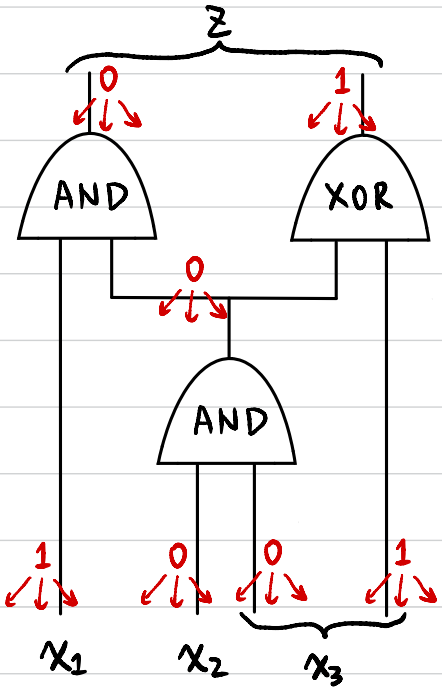
learning rate, gradient



$$\vec{w} \leftarrow \vec{w} - \eta \cdot (\underbrace{\langle \vec{x}_i, \vec{w} \rangle}_{y_i^* = \langle \vec{x}_i, \vec{w} \rangle \text{ (forward propagation)}} - y_i) \cdot \vec{x}_i$$

backward propogation

MPC for any function with $t \le n-1$ (GMW)



Each party P_i holds a random share $v_i^w \in \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

Inputs:

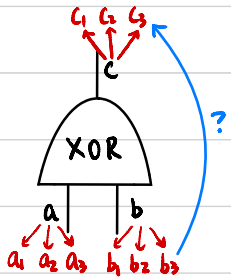
For each input wire w:

If it's from party P_k with input value $v^w \in \{0,1\}$,

P_k randomly samples $v_i^w \xleftarrow{\$} \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

↳ Sends v_i^w to party P_i.

XOR gates:

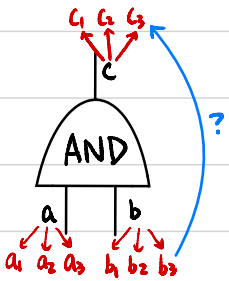


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \oplus b$

$c_i = a_i \oplus b_i$

AND gates:



GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \cdot b$

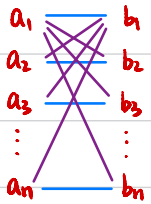
$c_i = a_i \cdot b_i \oplus \sum_{j \neq i} r_{i,j}^{(1)} \oplus \sum_{j \neq i} r_{j,i}^{(2)}$

$$a \cdot b = \left(\sum_{i=1}^{n} a_i\right) \cdot \left(\sum_{i=1}^{n} b_i\right) \mod 2$$

$$= \left(\sum_{i=1}^{n} a_i \cdot b_i\right) + \left(\sum_{i \neq j} a_i \cdot b_j\right) \mod 2$$

P_i locally

Reshare $r_{i,j}^{(1)}$ $r_{i,j}^{(2)}$



Outputs:

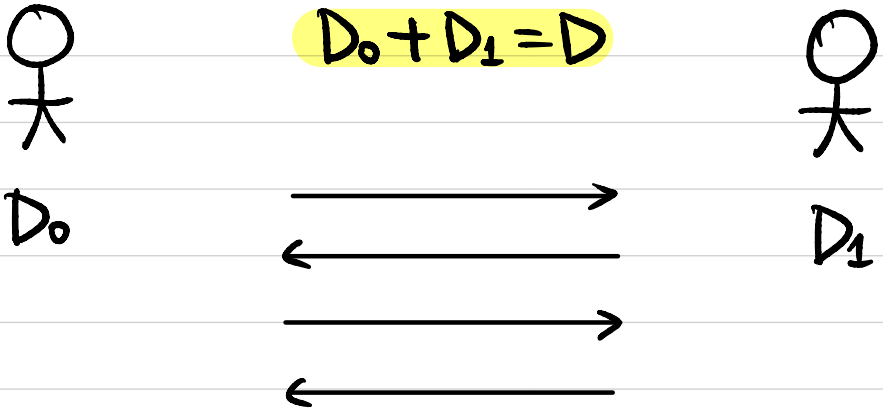
For each output wire w:

Each party P_i holds a random share $v_i^w \in \{0,1\}$

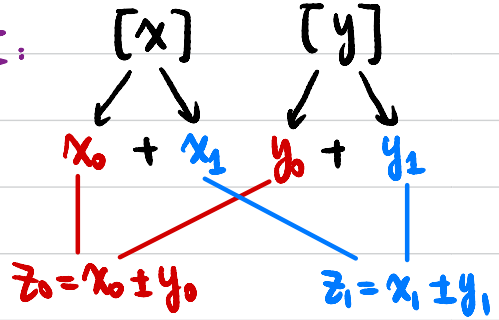
↳ Sends v_i^w to all parties

Each party computes the value $v^w = \bigoplus_{i=1}^{n} v_i^w$

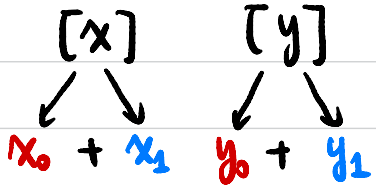
PPML for Linear Regression



$\pm$:



$\times$:



$$(x_0 + x_1) \cdot (y_0 + y_1)$$
$$= x_0 \cdot y_0 + x_1 \cdot y_1$$
$$+ x_0 \cdot y_1 + x_1 \cdot y_0$$

Reshare Reshare

Invariant: Two parties hold secret shares of $\vec{w}$.

Initialization: Randomly sample $\vec{w}_0, \vec{w}_1$ respectively.

SGD: $\vec{w} \leftarrow \vec{w} - \eta \cdot (\langle \vec{x}_i, \vec{w} \rangle - y_i) \cdot \vec{x}_i$ (η: public)

How to get a secret share of the updated $\vec{w}$?

Machine Learning Background

Logistic Regression

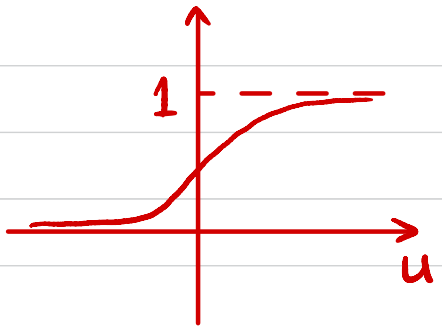
Data Points $(\vec{x}, y)$

ML Model: coefficient vector $\vec{w}$

$$g(\vec{x}) = f(\langle \vec{x}, \vec{w} \rangle)$$

f: activation function

Sigmoid $f(u) = \frac{1}{1+e^{-u}}$



For each data point $(\vec{x}_i, y_i)$,

Define loss function $L_i(\vec{w}) := -y_i \cdot \log y_i^* - (1-y_i) \cdot \log(1-y_i^*)$

$y_i^* := f(\langle \vec{x}_i, \vec{w} \rangle)$

Total loss: $L(\vec{w}) := \frac{1}{N} \sum_{i \in [N]} L_i(\vec{w})$

Goal: Find $\vec{w}$ that minimizes $L(\vec{w})$.

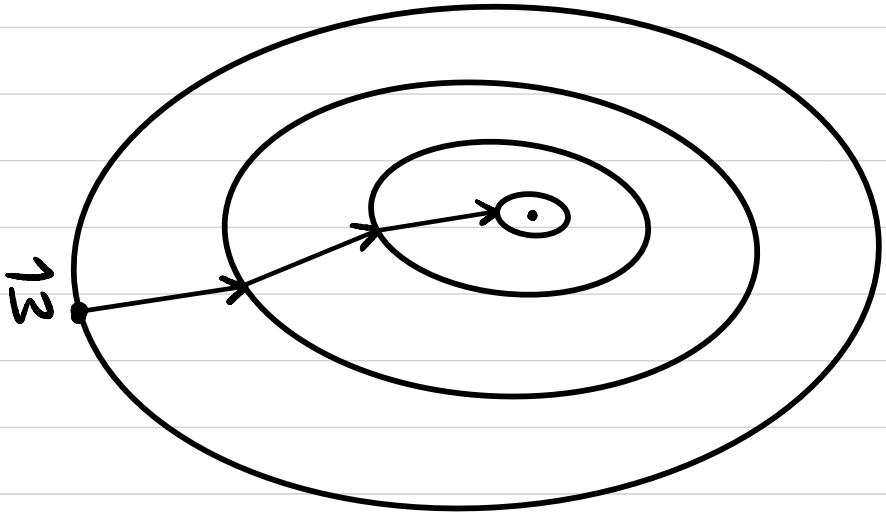
Machine Learning Background

Stochastic Gradient Descent (SGD)

- $\vec{w}$ initialized with arbitrary value
- Given a data point $(\vec{x}_i, y_i)$:

$$\vec{w} \leftarrow \vec{w} - \eta \cdot \nabla L_i(\vec{w})$$

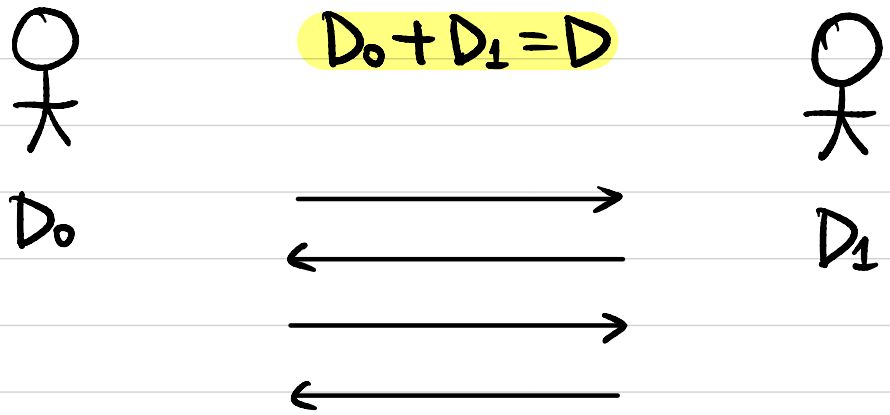
learning rate, gradient



$$\vec{w} \leftarrow \vec{w} - \eta \cdot (f(\underbrace{\langle \vec{x}_i, \vec{w} \rangle}_{y_i^* = \langle \vec{x}_i, \vec{w} \rangle \text{ (forward propagation)}}) - y_i) \cdot \vec{x}_i$$

backward propogation

PPML for Logistic Regression



Invariant: Two parties hold secret shares of $\vec{w}$.

Initialization: Randomly sample $\vec{w}_0, \vec{w}_1$ respectively.

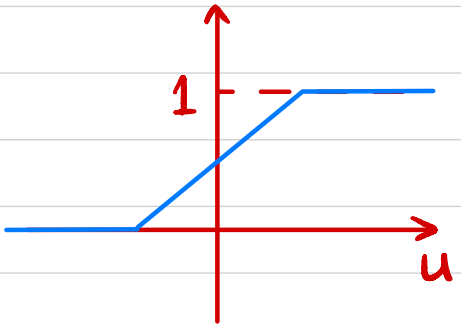
SGD: $\vec{w} \leftarrow \vec{w} - \eta \cdot (f(\langle \vec{x}_i, \vec{w} \rangle) - y_i) \cdot \vec{x}_i$

(η: public; $\vec{w}$: $[u]$; $f(\langle \vec{x}_i, \vec{w} \rangle)$: $[f(u)]$)

How to get a secret share of the updated $\vec{w}$?

PPML for Logistic Regression

Approximating $f(u)$



Piecewise polynomial $f(u) = \begin{cases} 0 & \text{if } u < -\frac{1}{2} \\ u + \frac{1}{2} & u \in [-\frac{1}{2}, \frac{1}{2}] \\ 1 & u > \frac{1}{2} \end{cases}$

$[b_1] := \begin{cases} 1 & \text{if } u < -\frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$

$[b_2] := \begin{cases} 1 & \text{if } u > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$

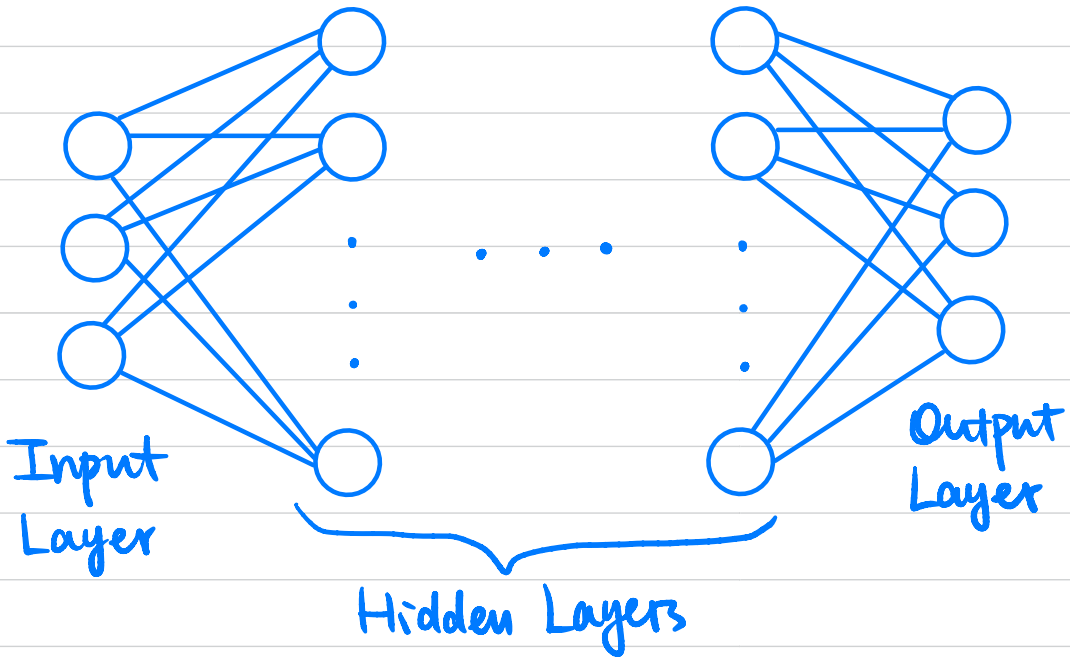
$$f(u) = b_1 \cdot 0 + (1 - b_1) \cdot (1 - b_2) \cdot (u + \tfrac{1}{2}) + b_2 \cdot 1$$

Machine Learning Background

Neural Network

Data Points $(\vec{x}, y)$

ML Model: coefficient vector $\vec{w}$



Each node in hidden layers: linear function + activation function

Total loss: $L(\vec{w}) := \frac{1}{N} \sum_{i \in [N]} Li(\vec{w})$

Goal: Find $\vec{w}$ that minimizes $L(\vec{w})$.