Sampling Techniques - Continuation

05/02/2025

MALA - Metropolis Adjusted Langevin Algorithm

Markov Chain Monte Carlo Method (MCMC) that combines Langevin Dynamics (a gradient-based proposal mechanism) with the Metropolis-Hastings algorithm (an acceptance-rejection step). This combination ensures that the Markov chain converges to the correct target dist. $p(x)$, even when the langevin dynamics are discretized.

MALA uses langevin dynamics to propose new samples :-

$$\left(dX_t = -\nabla U(X_t)\,dt + \sqrt{2}\,d\mathcal{B}_t \right).$$

- $\nabla U(X_t)$: gradient of the potential function
- U : potential function
- $d\mathcal{B}_t$: increment of standard Brownian motion.

Discretized using Euler - Maruyama Scheme.

$$x_{k+1} = x_k - \gamma \nabla U(x_k) + \sqrt{2\gamma}\, z_k .$$

γ : step size

Langevin Dynamics.

z_k : Standard normal R.V.

$$z_k \sim N(0, I).$$

The proposal is biased towards regions of higher probability (lower $U(x)$) due to the gradient term $-\gamma \nabla U(x_k)$.

The proposal dist $Q(x'|x)$ describes the probability of proposing a new state x' given the current state x.

For MALA - the dist is Gaussian:-

$$Q(x'|x) \propto \exp\left(-\frac{1}{4\gamma} \| x' - x + \gamma \nabla U(x) \|^2\right)$$

$x - x + \gamma \nabla U(x)$ → diff between x' (proposed state)

and the predicted state (using Langevin dynamics)

$$(x - \delta \nabla U(x))$$

The Gaussian dist ensures that the proposals are more likely to be close to the state predicted by the Langevin Dynamics.

To correct the discretization error in Langevin Dynamics, MALA uses Metropolis Hastings acceptance probability :-

without this the discretization error could cause the Markov chain to converge to wrong prob. dist.

$$\alpha = \min\left\{1, \frac{p(x_{k+1})\, Q(x_k \mid x_{k+1})}{p(x_k)\, Q(x_{k+1} \mid x_k)}\right\}$$

$p(x) \propto e^{-U(x)}$ → Target distribution

$Q(x'/x)$ → proposal distribution → Gaussian

Acceptance ratio :- $\dfrac{p(x_{k+1})\, Q(x_k \mid x_{k+1})}{p(x_k)\, Q(x_{k+1} + x_k)}$

This ensures that the Markov chain satisfies the detailed balance equation which guarantees the convergence to the target distribution $p(x)$

Step By Step MALA Algorithm :-

→ x_0 -> start with the initial state

→ For each step k :-

- Propose a new state x_{k+1} → using Langevin dynamics proposal.

$$x_{k+1} = x_k - \gamma \nabla U(x_k) + \sqrt{2\gamma}\, x_k .$$

- Compute the acceptance probability α

$$\alpha = \min\left\{1, \frac{p(x_{k+1})\, Q(x_k | x_{k+1})}{p(x_k)\, Q(x_{k+1} | x_k)}\right\}$$

- Accept x_{k+1} with prob. α, otherwise keep x_k

→ Burn-in → discard the first few samples to allow the chain to converge.

Metropolis Hastings Algorithm :-

MCMC method used to sample from a target prob. dist $P(x)$. It is particularly useful when $P(x)$ is difficult to sample from directly, but its unnormalized density can be evaluated.

Idea :-

Construct a markov chain whose stationary dist is the target distribution $P(x)$.

Proposing a new state x_i from a proposal dist. $Q(x|x_t)$, accepting/rejecting the proposed state based on acceptance prob. A.

P.T.O

Metropolis Hastings Algorithm :-

Start → initial state $x_t = x_0$

Propose a new state x_i from the proposal dist $Q(x|x_t)$.

Compute acceptance probability A :-

$$A = \min\left(\frac{P(x_i)\, Q(x_t|x_i)}{P(x_t)\, P(x_i|x_t)}, 1\right)$$

- this ratio ensures that the Markov chain satisfies detailed balance, which guarantees convergence to the target dist $P(x)$.

- ## Accept / Reject :-

  - Sample $k_i \sim \text{Uniform}[0,1]$

    if $k_i < A$, then accept the proposed state $x_{t+1} = x_i$ otherwise, reject, i.e, $x_{t+1} = x_t$.

Theory of Input/Outputs via Model.

Target = T (Ground Truth, Labels)

Obtained = Y (model.forward()).

↳ Model's output.

10 digit - binary numbers.

To model this.

---------- = 2^{10}

#1's > #0's ? → 1/0 → model.

1010110110 = 1

1111100000 = 1

0000000010 = 0

1111100000 = 0

Activation?

0/1

Sigmoid

I/P → | Model | → O/P.

(1 × 10) (10 × 1) ⇒ 1 × 1

~~1 × 10~~

~~10 × 1~~

(0 - 1)

(−1 to 1)

0 1

{0 or +ve} = valid.

(∞ → NaN) → corrupt.

#1 > #0's

6 → 4

o/p → 1

W

1 − 1
1 − 1
1 − 1
0 − 1
0 − 1
0 − 1
1 − 1
1 − 1
0 = 1

b = -5

||x_i||

#1s → 5 , #0s → 5

6 − 5 ⇒ 1

1000

10 − 5 ⇒ 5

7 8 9 10 3 2 1 0

(0 to 1) → Sigmoid.

LR = 10^{-4}

$w_{t+1} = w_t - \eta \frac{\partial L}{\partial w}$

W = 1000

1000

10^{-4}

Pipeline

- Data → simulate / Web-scrap collect.
- Model → Pytorch classes.
- I/p → Model → prediction → training/valid/test
- Visualization / Analysis. → Metrics

Dataloader → raw data → preprocessing

Augmentations

[0-255] → Rescale? → [0-1] (images)

(-1 to 1)

weights

(exploding / NaN) ○ complexity

fp 16

fp·32

weights (-10000, 10000)

(-30, 30)

Model → Pytorch.

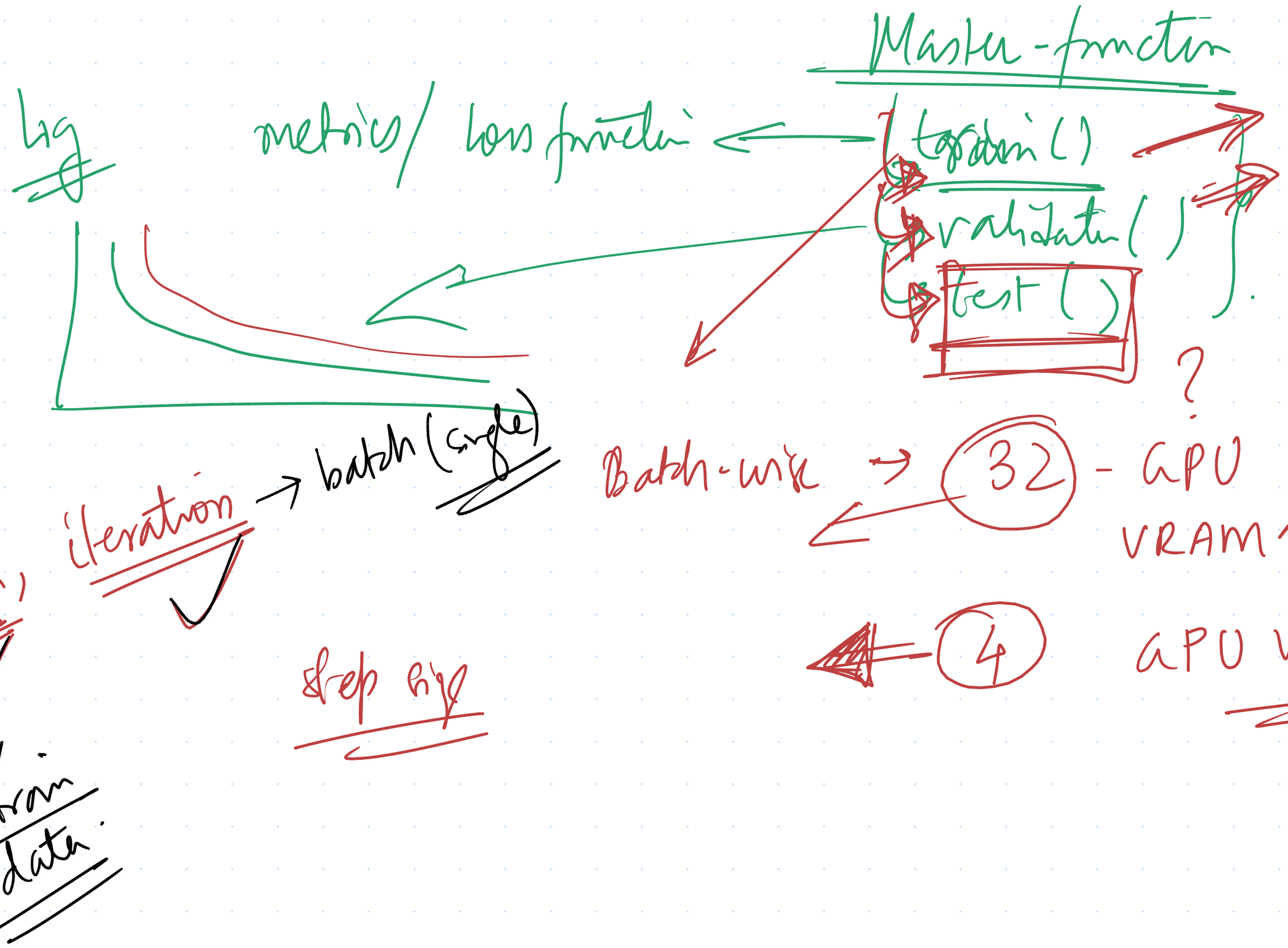
train(), validation(), test().

model.train() → (epoch wise) optimize the model.

validation() → torch.no_grad() → model.eval()

test() → model.eval() → torch.no_grad()

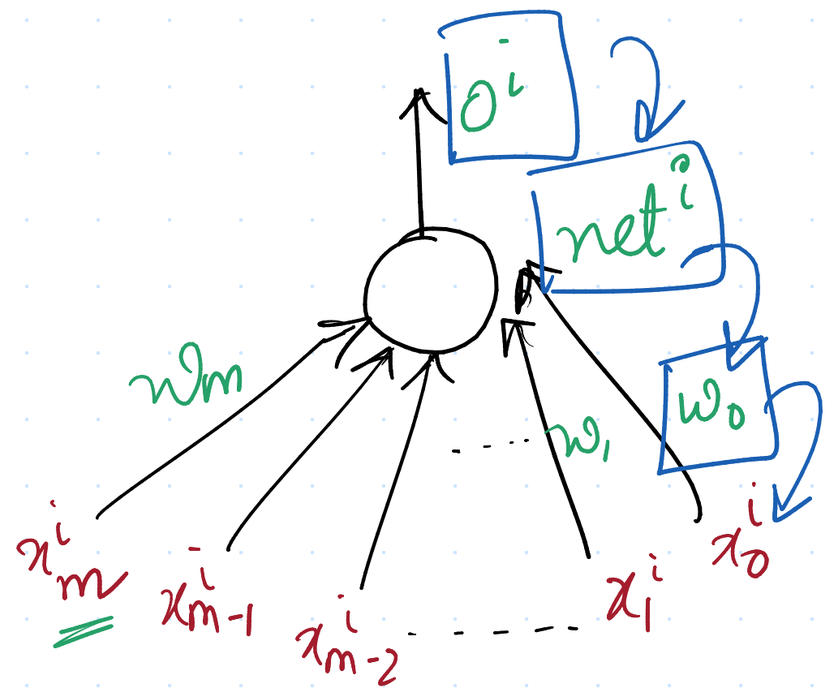
model checkpointing, model saving, model loading



Target → T , obtained ⇒ Y

Sigmoid

$$O^i = \frac{1}{1+e^{-x}} \rightarrow net^i$$



$$O^i = \frac{1}{1+e^{-net^i}}$$

$x^i \rightarrow$ ith sample.
$x_j \rightarrow$ jth scalar

$$net^i = W \cdot X^i = \sum_{j=0}^{m} w_j x_j^i$$

$X =$ 0th, 1st, 2nd, 3rd.

$x_j^i \rightarrow x_2^0$

$[1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0]$ → 0
$[1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0]$ → 1
$[1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0]$ → 1 (x_8^2)
$[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$ → 1. (x_4^3)

y

y^0

$X^0 \Leftarrow [x_0^0, x_1^0, x_2^0, \ldots x_m^0] \rightarrow [y^0]$

$X^1 \Leftarrow [x_0^1, x_1^1, x_2^1, \ldots x_m^1] \rightarrow [y^1]$

$X^{n-1} \Leftarrow [x_0^{n-1}, x_1^{n-1}, \ldots, x_m^{n-1}] \rightarrow [y^{n-1}]$

$(m+1)$

Vectors → W capital letters.

scalar → small letters

$X^i \rightarrow i^{th}$ input vector.

$O_i \rightarrow$ output scalar.

$net_i \rightarrow WX^i$ (n-th i/p / o/p)

net^i → same → net_i

$W \rightarrow$ weight vector.

W & X^i has $(m+1$ components$)$

$W = \langle w_m, w_{m-1}, \ldots w_2, w_1, w_0 \rangle$

$x^i = \langle x^i_m, x^i_{m-1}, \ldots, x^i_2, x^i_0 \rangle$.

Derivative of Sigmoid :-

$$o^i = \frac{1}{1+e^{-net_i}}$$

$$\log_e(o^i) = -\ln\left(1+e^{-net_i}\right)$$

Derivative w.r.t. net^i :-

$$\frac{1}{o^i}\cdot\frac{\partial o^i}{\partial net^i} = +\frac{1}{\left(1+e^{-net_i}\right)}\left(+e^{-net_i}\right)$$

$$\frac{1}{o^i}\frac{\partial o^i}{\partial net_i} = \frac{e^{-net_i}}{1+e^{-net_i}} = 1 - \frac{1}{1+e^{-net_i}}$$

$\downarrow$

o^i

$$1+x \;\overline{)\; x \;(\; 1}$$

$$x+1$$

$$-1$$

$$\left(1 - \frac{1}{1+x}\right) \qquad \frac{x}{1+x}.$$

$\downarrow$

$$\frac{1}{o^i}\frac{\partial o^i}{\partial net^i} = (1-o^i)$$

$$\frac{1+x-x}{1+x} \Rightarrow \frac{x}{1+x}.$$

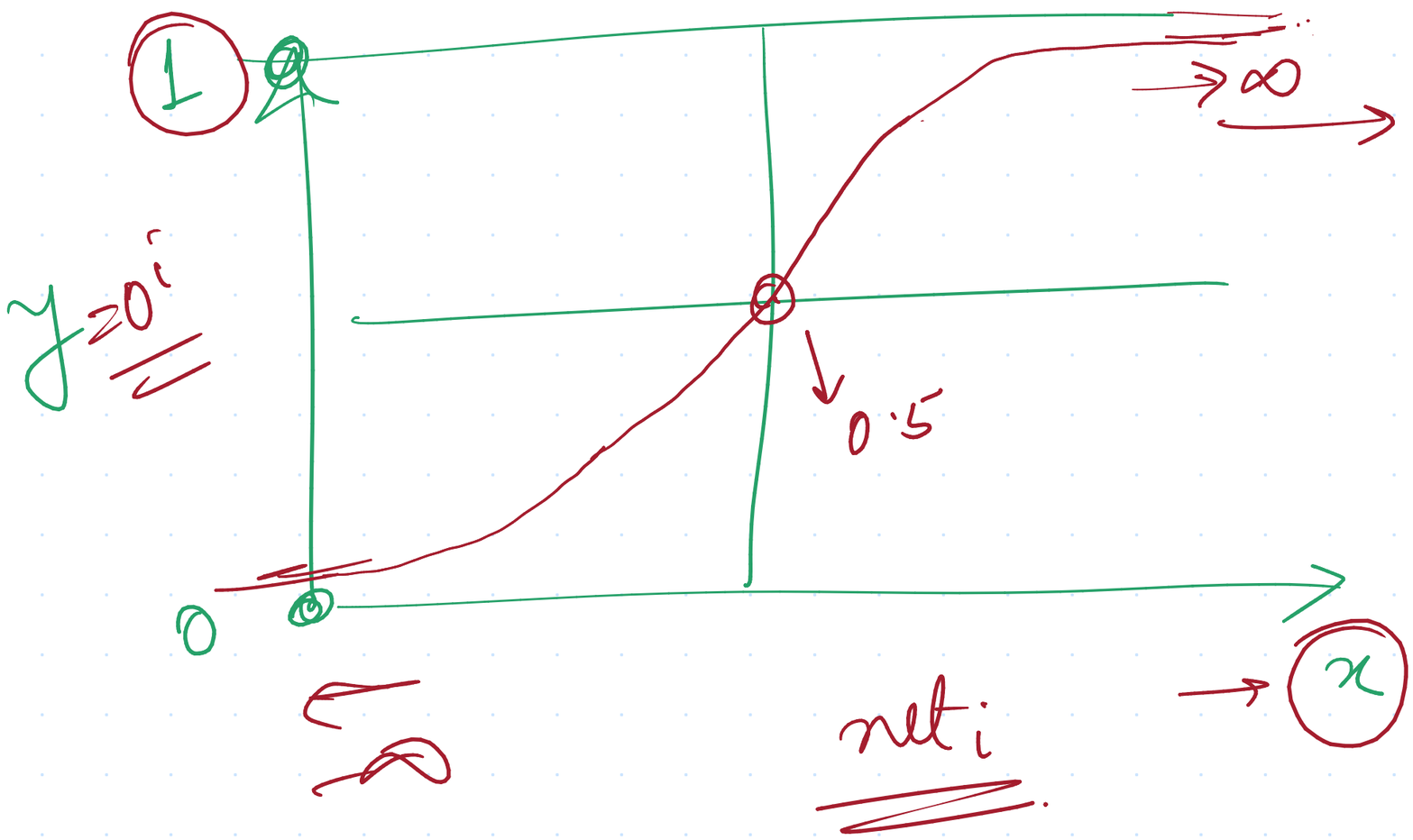
$$\frac{\partial o^i}{\partial net^i} = o^i(1-o^i)$$

when $x = \infty$,

$y = 1$

when $x = -\infty$

$y = 0$



where $x = 0.5$,

$y = ?$

$y = ?$ $\quad x = 0.5$

where is the derivative of sigmoid maximum?

$$y = \frac{1}{1 + e^{-0.5}}$$

$$y = 0.6224$$

$$\frac{\partial}{\partial x}\left(x(1-x)\right) \Rightarrow \frac{d}{dx}\left(x - x^2\right) \Rightarrow 1 - 2x$$

Derivative is maximum at $1 - 2x = 0$.

$x = \frac{1}{2}$

$\therefore \boxed{x = \frac{1}{2}}$.

But maximum value of derivative = ?

maximum value of derivative

$\frac{1}{4}$ maximum value of derivative of sigmoid.

$x(1-x) \Rightarrow \frac{1}{2}\left(1 - \frac{1}{2}\right)$

$\Rightarrow \frac{1}{2} \times \frac{1}{2} \Rightarrow 0.25$

When $x = \frac{1}{2}$ derivative of sigmoid is maximum.

※

value of sigmoid $= \frac{1}{1+e^{-(x)}}$ $\Rightarrow \frac{1}{1+e^{-(\frac{1}{2})}} \simeq 0.62$

{ Derivative of sigmoid function.

$\Rightarrow o_i(1-o_i)$

$\Rightarrow \frac{1}{(1+e^{-x})}\left(1-\frac{1}{1+e^{-x}}\right)$

$\Rightarrow 0.62 \times (1-0.62)$

$\Rightarrow 0.2356$

$net_i = \infty$.

$o^i = \frac{1}{1+e^{-net_i}}$

$net_i = -\infty$

$o^i = \frac{1}{1+e^{\infty}} = \frac{1}{e^{\infty}} = 0$.

$o^i = \frac{1}{1+e^{-\infty}}$

$= \frac{1}{1+\frac{1}{e^{\infty}}}$

$o^i = \frac{1}{1+0} = 1$

$$\frac{\partial}{\partial x}\left(\frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right)\right)=0.$$

x → different. ?

$\frac{1}{2}x$ different x.