E

IRIE NORIFUMI

Table of contents

JDLA E E2024#2

E
E GPT5

- E
-
-
- GPT5 SOTA

1

E E2024#2

Part I

1. (Mach Basics)

2 1. (Statistics)

2.0.1 1

KL $D_{KL}(P||Q)$ H(P,Q)

$$D_{KL}(P||Q) = H(P,Q) - H(P)$$

P Q H(P) P

A. Q P
B. P H(P) KL
C. KL P=Q 0
D. KL $D_{KL}(P||Q) = D_{KL}(Q||P)$

Part II

2. (Machine Learning)

3 1. (Machine Learning Basics)

3.0.1 1

MAP

MAP Maximum A Posteriori Estimation MLE Maximum Likelihood Estimation

I.

A. MLE MAP
B. MLE MAP
C. MLE MAP
D. MAP

MAP θ_{MAP} θ_{MAP}

$$\theta_{\text{MAP}} = \arg\max_{\theta} P(\theta \mid X) \ =$$

II.

A. $\arg\max_\theta \, P(X \mid \theta)$
B. $\arg\max_\theta \, \dfrac{P(\theta)}{P(X)}$
C. $\arg\max_\theta \, P(X \mid \theta)\, P(\theta)$
D. $\arg\max_\theta \, \log P(X)$

Part III

3. (Basic Deep-learning)

4 1. (Feedforward Neural Network)

4.0.1 1

2

```
import numpy as np

def binary_crossentropy(y_true, y_pred):
    """
    y_true:     [batch_size] (0 or 1)
    y_pred:    [batch_size] (0~1 )
    """
    #
    epsilon = 1e-15
    y_pred =

    #
    loss = -np.mean(y_true * np.log(y_pred) + (1 - y_true) * np.log(1 - y_pred))
    return loss
```

A. `np.clip(y_pred, epsilon, 1.0)`
B. `np.clip(y_pred, 0.0, 1 - epsilon)`
C. `np.clip(y_pred, epsilon, 1 - epsilon)`
D. `np.maximum(y_pred, epsilon)`

4.0.2 2

```
import numpy as np

def softmax(x):
    """
    x:    [batch_size, num_classes]
    """
    #
    x_max =
    x_shifted = x - x_max

    #
    exp_x = np.exp(x_shifted)
    sum_exp =

    return exp_x / sum_exp
```

A. : `np.max(x, axis=1, keepdims=True)`, : `np.sum(exp_x, axis=1, keepdims=True)`
B. : `np.max(x, axis=0, keepdims=True)`, : `np.sum(exp_x, axis=0, keepdims=True)`
C. : `np.max(x, axis=1)`, : `np.sum(exp_x, axis=1)`
D. : `np.maximum(x, 0)`, : `np.sum(exp_x)`

4.0.3 3

tanh

```
import numpy as np

def tanh(x):
    y =
    return y
```

A. `(np.exp(x) + np.exp(-x)) / (np.exp(x) + np.exp(-x))`
B. `(np.exp(x) + np.exp(-x)) / (np.exp(x) - np.exp(-x))`
C. `(np.exp(x) - np.exp(-x)) / (np.exp(x) - np.exp(-x))`
D. `(np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))`

4.0.4 4.

( ) ~ ( ) $\sigma(x)$ ( ) $\sigma(x)$ ( )
( ) ReLU ( ) ( ) ( )

( ) :
A. $\frac{1}{1+e^{-x}}$
B. $\frac{e^x}{1+e^x}$
C. $\frac{1}{1+e^x}$
D. $\frac{1}{1-e^x}$

( ) :
A. $1 - \sigma(x)$
B. $\sigma(x)(1 - \sigma(x))$
C. $x(1 - x)$
D. e^{-x}

( ) :
A.
B.
C.
D.

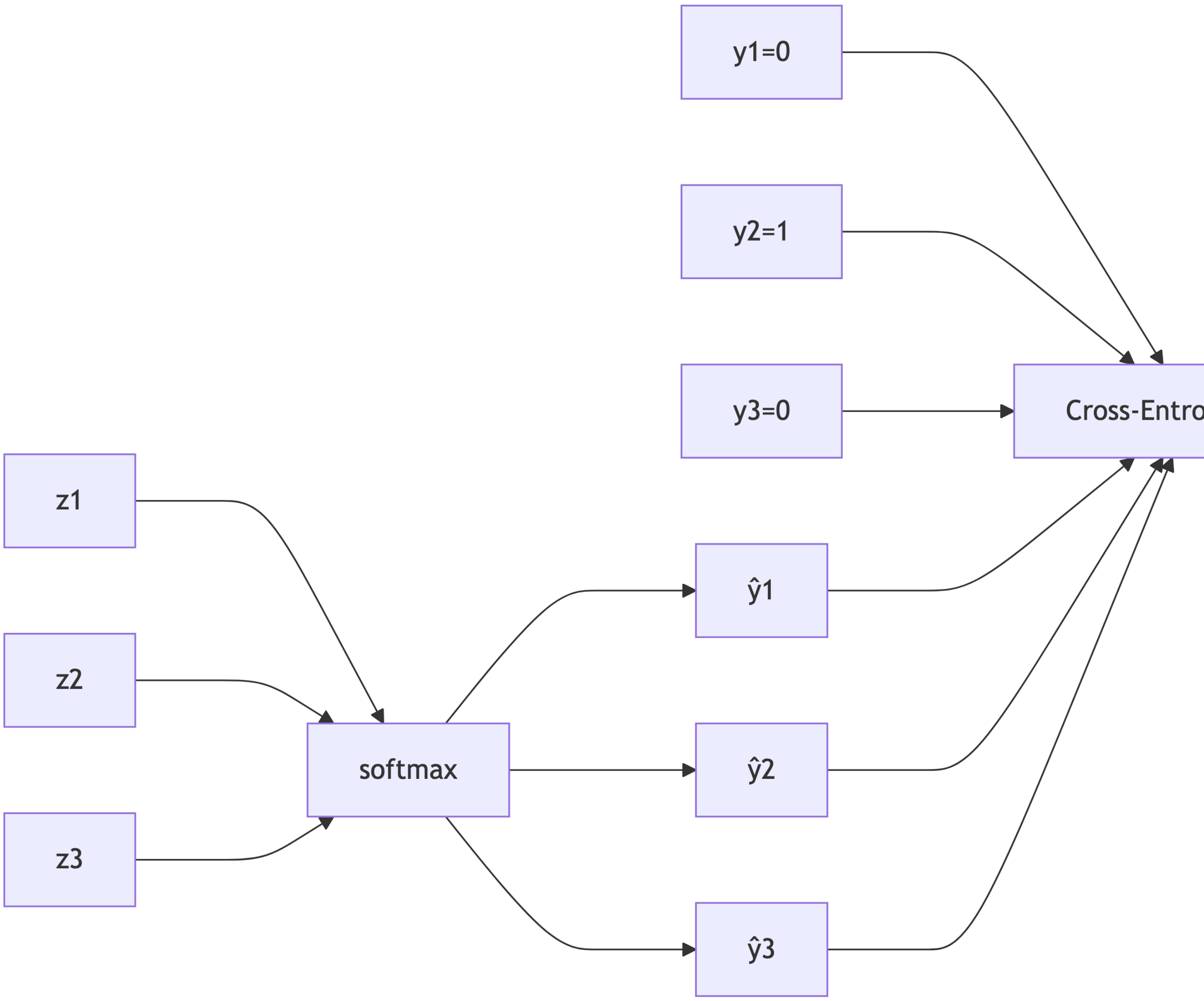
( ) :
A.
B. $(0, 1)$
C. ReLU
D. ReLU 1

( ) ( ) :
A. ResNet
B. Word2Vec Negative Sampling
C. GAN Discriminator
D. Transformer Multi-Head Attention

5 2. (Optimization)

5.0.1 1

(z $= [z_1, z_2, z_3]$) (L)



I. $\frac{\partial L}{\partial z_j}$

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_k \exp(z_k)}$$

$$L = -\sum_i y_i \log(\hat{y}_i)$$

A. $y_j - \hat{y}_j$
B. $\hat{y}_j - y_j$
C. $\frac{-y_j}{\hat{y}_j}$
D. $\hat{y}_j(1 - y_j)$

II.3 $y = [0, 1, 0]$ $\hat{y} = [0.2, 0.3, 0.5]$ $\frac{\partial L}{\partial z_2}$
A. -0.7
B. 0.7
C. -0.3
D. 0.3

5.0.2 2

AdaGrad

$$h_t = h_{t-1} + \nabla E(W_t) \odot \nabla E(W_t) \alpha_t = \alpha_0 \times (1/\sqrt{}(h_t + \varepsilon)) W_{t+1} = W_t - \alpha_t \odot \nabla E(W_t)$$

A. h_t
B. _t
C.
D. h_t

5.0.3 3

1 train_flag True False dropout_ratio

```python
import numpy as np

class Dropout:
    def __init__(self, dropout_ratio=0.5):
        self.dropout_ratio = dropout_ratio
        self.mask = None

    def forward(self, x, train_flg=True):
        if train_flg:
            self.mask = (   ) self.dropout_ratio
            return (   )
        else:
            return (   )

    def backward(self, dout):
        return (   )
```

A. `np.random.rand(*x.shape) >`
B. `np.random.rand(*x.shape) >`
C. `np.random.randn(*x.shape) <`
D. `np.random.rand(*x.shape) <`

A. `x * self.mask`
B. `x * (1.0 + self.mask)`
C. `x - self.mask`
D. `x * (1.0 - self.mask)`

A. `x * self.mask`
B. `x * (1.0 - self.mask)`
C. `x * self.dropout_ratio`
D. `x * (1.0 - self.dropout_ratio)`

A. `dout * self.mask`
B. `dout * (1.0 - self.mask)`
C. `dout * self.dropout_ratio`
D. `dout * (1.0 - self.dropout_ratio)`

6 3. (Convolutional Neural)

6.0.1 1.

1 1 im2col
5×5
3×3
1
0

I.
A. (2, 2)
B. (3, 3)
C. (4, 4)
D. (5, 5)

II. im2col im_col.shape 1 1

A. (3, 3)
B. (9, 9)
C. (9, 25)
D. (3, 9)

Part IV

4 . (Advanced Deep-learning)

7 1. (Image Recognition)

7.0.1 1

A.
B.
C.
D.

7.0.2 2

$C_{in} = 64$ $C_{out} = 256$ $H = W = 32$
3×3 ResNet50 MAC

1×1 conv : 64 → 64
3×3 conv : 64 → 64
1×1 conv : 64 → 256

A. 1/2
B. 1/4
C. 1/8
D. 1/16

7.0.3 3

Vision Transformer ViT 1
A. 1 CNN
B. Transformer
C. CNN
D.

8 2. (Object Detection)

8.0.1 1

FCOS
A. FCOS
B. FCOS Feature Pyramid Network(FPN)
C. FCOS
D. FCOS Center-ness

8.0.2 2

R-CNN (A) ROI ROI (B) R-CNN (C)

8.0.3 3

SSD Single Shot MultiBox Detector

A. Single Shot 2 MultiBox 1 1
B. Single Shot 1 MultiBox
C. Single Shot 1 MultiBox GPU
D. Single Shot YOLO MultiBox

9 3. Semantic Segmentation (Semantic Segmentation)

9.0.1 1

FCN Fully Convolutional Network CNN VG
ResNet FCN-32s FCN-16s FCN-8s
FCN

$$L = -\sum_{x\in\Omega}\sum_{c\in C} y_c(x) log P_c(x)$$

$y_c(x)$ x one-hot $P_c(x)$ c

A. ( ) ( ) Transposed Conv ( ) ( )
B. ( ) ( ) Interpolation ( ) ( ) ReLU
C. ( ) ( ) Transposed Conv ( ) ( )
D. ( ) ( ) ( ) ( )

10 4. (Natural Language Processing)

10.0.1 1

Seq2Seq

A. +logP y |

B. -logP y |x

C. +logP y |

D. -logP y x

10.0.2 2

Word2Vec

A. CBOW

B. Skip-gram

C.

D. LSI

10.0.3 3

GPT

A. GPT

B. GPT

C. GPT Masked Language Modeling

D. GPT Few-shot Learning Fine-tuning

10.0.4 4

Word2Vec Skip-gram softmax

A. softmax
B.
C.
D. skip-gram CBOW

11 5. Recurrent Neural Network (Recurrent Neural Network)

11.0.1 1

LSTM t LSTM RNN 3

```
n_input = 100
n_hidden = 256

w =
b = np.zeros(n_hidden * 4)

# x
# h, c
def lstm(x, h, c):
    #       Wx + Uh + b
    inputs = np.concatenate((x, h), axis=1)
    inputs = np.matmul(inputs, w) + b
    z, i, f, o = np.hsplit(inputs, 4)
    ...
```

( )

A. `np.random.randn(n_input * n_hidden, n_hidden * 3)`
B. `np.random.randn(n_input * n_hidden, n_hidden * 4)`
C. `np.random.randn(n_input + n_hidden, n_hidden * 3)`
D. `np.random.randn(n_input + n_hidden, n_hidden * 4)`

11.0.2 2

LSTM GRU(Gated Recurrent Unit)

12 6. (Generative Model)

12.0.1 1

1

A. VAE Reparameterization Trick
B. Denoising Autoencoder
C.
D. GAN VAE

13 7. (Reinforcement Learning)

13.0.1 1

MDP

$\gamma = 0.5$
s_0 a_0 $r_0 = 2$ s_1
s_1 $r_1 = 4$ 0

I. $V^{\pi}(s_1)$
II. $Q^{\pi}(s_0, a_0)$

13.0.2 2

DQN Q TD

$$TD \quad = [] - Q(s, a)$$

A. Q(s', a')
B.
C. $\gamma * Q(s, a)$
D. $R(s, a) + \gamma * maxQ(s', a')$

13.0.3 3

$Q(s, a)$ $\pi_\theta(a|s)$
$V(s_0)$ $\nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[()]$$

_ (|) ( )

A. $\nabla_\theta log \pi_\theta(a|s) Q^\pi(s, a)$
B. $\nabla_\theta \pi_\theta(a|s) Q^\pi(s, a)$
C. $\frac{\nabla_\theta \pi_\theta(a|s)}{Q^\pi(s,a)}$
D. $\frac{Q^\pi(s,a)}{\nabla_\theta \pi_\theta(a|s)}$

14 8. (Various Learning Methods)

14.0.1 1

Triplet Network Triplet Network Siamese Network Siamese Network 2 Triplet Network 3 3 d_p triplet loss L

L

A. $max(d_p - d_n + m, 0)$
B. $max(d_p - d_n, m)$
C. $max(-d_p + d_n, m)$
D. $max(d_p - d_n - m, 0)$

14.0.2 2

A. Triplet Network
B. Siamese Network 2
C. Siamese Network 2 Triplet Network 2
D. Triplet Network

15 9. (Explainability of Deep-learning)

15.0.1 1

Grad-CAM Gradient-weighted Class Activation Mapping

A. AlexNet CNN
B. Grad-CAM
C. Guided Grad-CAM Grad-CAM Guided Backpropagation
D. CNN

15.0.2 2

1

A. CAM Global Average Pooling
B. Grad-CAM Global Average Pooling
C. GoogLeNet Flatten→ CAM
D. VGG Global Average Pooling CAM

15.0.3 3

Shapley Shapley

A. 1
B. Permutation
C.
D.

Part V

5 . (Infrastructure)

16 4. (Accelerator)

16.0.1 1

AI Google TPU Tensor Processing Unit TPU GPU GPU

A.
B.
C.
D. 8bit 16bit

17

18 1 . (Mach Basics)

18.1 1. (Statistics)

1. D

19 2 . (Machine Learning)

19.1 1. (Machine Learning Basics)

1. I: A II: C

20 3 . (Basic Deep-learning)

20.1 1. (Feedforward Neural Network)

1. C
2. A
3. D
4. ( ):A

( ):B

$$\frac{d}{dx}\frac{1}{f(x)} = -\frac{f'(x)}{f(x)^2}$$

f(x) $= 1 + e^{-x}$

$$f(x) = 1 + e^{-x}, \quad f'(x) = -e^{-x}$$

$$\frac{d\sigma(x)}{dx} = \frac{d}{dx}\frac{1}{f(x)} = -\frac{f'(x)}{f(x)^2} = -\frac{-e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$$

(x)

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

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

( ):D

( ):B

( ) ( ):B,C

(D) Transformer Multi-Head Attention

20.2 2. (Optimization)

1. I: B II: A
2. D

0

20.3 3. (Convolutional Neural)

1. I: B II: B

21 4 . (Advanced Deep-learning)

21.1 1. (Image Recognition)

1. B

21.2 2. (Object Detection)

1. A
2. A: Region-based Convolutional Neural Network
B: Region Proposal( )
C: Selective Search
3. B

21.3 3. Semantic Segmentation (Semantic Segmentation)

1. C

21.4 4. (Natural Language Processing)

1. B
2. C
3. B
2. C

21.5 5. Recurrent Neural Network (Recurrent Neural Network)

1. D
2. LSTM: (Forget Gate) (Input Gate) (Output Gate) (Memory Cell)
GRU: (Reset Gate) (Update Gate)

21.6 6. (Generative Model)

1. C

21.7 7. (Reinforcement Learning)

1. I: 4 II: 4
2. D
3. A

21.8 8. (Various Learning Methods)

1. A
2. A

21.9 9. (Explainability of Deep-learning)

1. A
2. B
3. B

22 5 . (Infrastructure)

22.1 4. (Accelerator)

1. A

23

- Transformer - Multi-Head Attention vol.28
- KL
- E
- E
- e