Machine Learning

Assignment 3

姓名：
日期：2024/5/16

Lecture 5 Model Selection and Evaluation

1 Short Answers [必做，不用提交不算分数，后续公布答案后自行核对]

(1) [2pts] Explain overfitting and underfitting of machine learning models.

Overfitting: When the learner learns the training examples "too well", it is likely that some peculiarities of the training examples are taken as general properties that all potential samples will have, resulting in a reduction in generalization performance.

Underfitting: The learner failed to learn the general properties of training examples.

[2pts] Write down two aspects affecting the risk the overfitting.

i. the overly strong learning ability (overly high model complex)
ii. insufficient training samples

[2 pt] What strategies do decision trees use to reduce the risk of overfitting?

pruning

[2 pt] What strategies do NNs use to reduce the risk of overfitting?

Early stopping

(2) [6pts] Derive Precision and Recall from confusion matrix (i.e., TP, TN, FP, FN). Write down the definition of F1 score given precision and recall.

Precision=$\frac{TP}{TP+FP}$ Recall=$\frac{TP}{TP+FN}$

$$F1 = \frac{2\times P\times R}{P+R} = \frac{2\times TP}{总+TP-TN}$$

(3) [6pts] Describe K-fold cross-validation. Are the error rates of different folds independent when K>2 and why?

K-fold cross-validation is a technique used to assess the performance of a machine learning model. It involves splitting the dataset into K non-overlapping subsets, or folds. The model is then trained K times, each time using a different fold as the validation set and the remaining folds as the training set. Finally, the performance is evaluated by averaging the results of the K validations.

When K>2, the error rates of different folds are typically independent. This is because each fold is randomly sampled from the original dataset, and each fold is treated independently during training and validation. Therefore, the performance evaluations of each fold are usually independent of each other.

2 AUC [证明题，选做]

对于有限样例，请证明

$$\mathrm{AUC} = \frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}\left(\mathbb{I}(f(x^+) > f(x^-)) + \frac{1}{2}\mathbb{I}(f(x^+) = f(x^-))\right)$$

Proof. 此处用于写证明(中英文均可)

正例数量为m^+，反例数量为m^-

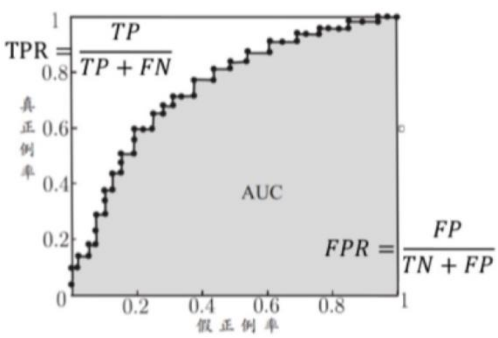
$$AUC = \sum_{i=1}^{m}\left[\frac{1}{2}(x_{i+1}-x_i)(y_{i+1}-y_i)\right]$$

$$= \sum_{x^-\in D^-}\frac{1}{2m^-}\left[\frac{2}{m^+}\sum_{x^+\in D^+}\mathbb{I}(f(x^+)>f(x^-)) + \frac{1}{m^+}\sum_{x\in D}\mathbb{I}(f(x^*)=f(x^-))\right]$$

$$= \sum_{x^+\in D^+}\left[\frac{1}{m^+m^-}\sum_{x^+\in D^+}\mathbb{I}(f(x^+)>f(x^-)) + \frac{1}{2}\cdot\frac{1}{m^+m^-}\sum_{x\in D}\mathbb{I}(f(x^*)=f(x^-))\right]$$

$$= \frac{1}{m^+m^-}\sum_{x^+\in D^+}\left[\sum_{x^-\in D^-}\mathbb{I}(f(x^+)>f(x^-)) + \frac{1}{2}\cdot\sum_{x^-\in D^-}\mathbb{I}(f(x^+)=f(x^-))\right]$$

$$= \frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x\in D}\left(\mathbb{I}(f(x^+)>f(x^-)) + \frac{1}{2}\mathbb{I}(f(x^+)=f(x^-))\right)$$



基于有限样例绘制的 ROC 曲线与 AUC

Lecture 6 Neural Networks

0. BP 算法推导 [必做，不提交不算分数，可根据南瓜书自行核对]

请将西瓜书教材中的标准 BP 算法的推导过程进行完整的推导，注意符号的一致性。

Solution（中英文均可）

Prove that

$$\Delta w_{hj} = \eta g_j b_h \qquad (1)$$

$$\Delta \theta_j = -\eta g_j \qquad (2)$$

$$\Delta v_{ih} = -\eta e_h x_i \qquad (3)$$

$$\Delta \gamma_h = -\eta e_h \qquad (4)$$

For (1), we have

$$\Delta w_{hj} = -\frac{\eta \partial E_k}{\partial w_{hj}}$$

Then

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

$$= (\hat{y}_j^k - y_j^k) \cdot f'(\beta_j - \theta_j) \cdot b_h$$

$$= (\hat{y}_j^k - y_j^k) \cdot b_h \cdot [f(\beta_j - \theta_j) \times (1 - f(\beta_j - \theta_j))]$$

$$= (\hat{y}_j^k - y_j^k) \cdot b_h \cdot \hat{y}_j^k (1 - \hat{y}_j^k)$$

$$= -g_j b_h.$$

Thus.

$$\Delta w_{hj} = -\frac{\eta \partial E_k}{\partial w_{hj}} = \eta g_j b_h.$$

For (2), we have

$$\Delta \theta_j = -\frac{\eta \partial E_k}{\partial \theta_j}$$

Then,

$$\frac{\partial E_k}{\partial \theta_j} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \theta_j}$$

$$= \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial f(\beta_j - \theta_j)}{\partial \theta_j}$$

$$= \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot f'(\beta_j - \theta_j) \times (-1)$$

$$= \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot f(\beta_j - \theta_j) \times [1 - f(\beta_j - \theta_j)] \times (-1)$$

$$= \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \hat{y}_j^k (1 - \hat{y}_j^k) \times (-1)$$

$$= \frac{\partial [\frac{1}{2} \sum_{j=1}^{l} (\hat{y}_j^k - y_j^k)^2]}{\partial \hat{y}_j^k} \cdot \hat{y}_j^k (1 - \hat{y}_j^k) \times (-1)$$

$$= \frac{1}{2} \times 2 (\hat{y}_j^k - y_j^k) \times 1 \cdot \hat{y}_j^k (1 - \hat{y}_j^k) \times (-1)$$

$$= (y_j^k - \hat{y}_j^k) \hat{y}_j^k (1 - \hat{y}_j^k)$$

$$= g_j$$

Thus. $\Delta \theta_j = -\frac{\eta \partial E_k}{\partial \theta_j} = -\eta g_j$

For (3). We have. $\Delta v_{ih} = -\frac{\eta \partial E_k}{\partial v_{ih}}$

Then. $$\frac{\partial E_k}{\partial v_{ih}} = \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{ih}}$$

$$= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot x_i$$

$$= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot f'(\alpha_h - \gamma_h) \cdot x_i$$

$$= -f'(\alpha_h - \gamma_h) \sum_{j=1}^{l} g_j \cdot w_{hj} \cdot x_i$$

$$= -b_h (1 - b_h) \sum_{j=1}^{l} g_j \cdot w_{hj} \cdot x_i$$

$$= -e_h \cdot x_i$$

So. $\Delta v_{ih} = -\frac{\eta \partial E_k}{\partial v_{ih}} = \eta e_h x_i$

For (4) We have.

$$\Delta \gamma_h = -\frac{\eta \partial E_k}{\partial \gamma_h}$$

Then, $$\frac{\partial E_k}{\partial \gamma_h} = \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \gamma_h}$$

$$= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot f'(\alpha_h - \gamma_h) \times (-1)$$

$$= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot w_{hj} \cdot f'(\alpha_h - \gamma_h)$$

$$= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot w_{hj} \cdot b_h (1 - b_h)$$

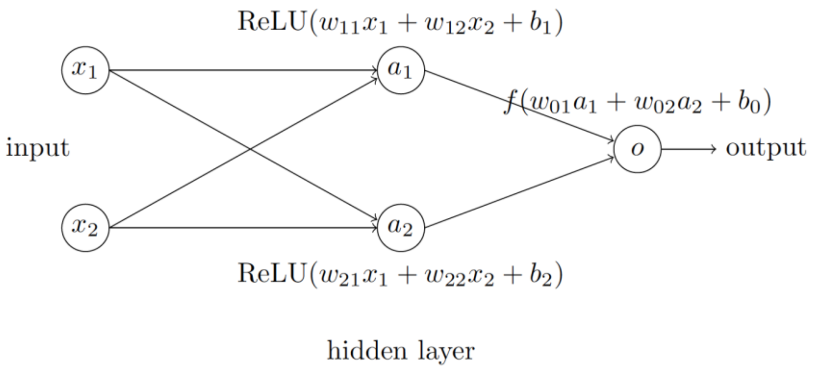
$$= \sum_{j=1}^{l} g_j \cdot w_{hj} \cdot b_h (1 - b_h)$$

$$= e_h.$$

So. $\Delta \gamma_h = -\frac{\eta \partial E_k}{\partial \gamma_h} = -\eta e_h.$

1. [25pts] Multi-layer Perception

Suppose that we apply neural networks on a problem which has boolean inputs $x \in \{0,1\}^p$ and boolean output $y \in \{0, 1\}$. The network structure example is showed as below. In this example we set $p = 2$, single hidden layer with 2 neurons, activation function $ReLU(u) = u$ if $u > 0$ otherwise 0, and an additional threshold function (e.g., $f(v) = 1$ if $v > 0$, otherwise $f(v) = 0$) for output layer.



(1) [5pts] Using the structure and settings of neural network above, show that such a simple neural network could output the function $x_1\ XOR\ x_2$ (equals to 0 if $x_1 = x_2$ and otherwise 1), which is impossible for linear models. State the values of parameters (i.e., w_{ij} and b_i) you found.

(2) [20pts] Now we allow the number of neurons in the hidden layer to be more than 2 but finite. Retain the structure and other settings. Show that such a neural network with single hidden layer could output an arbitrary binary function $h: \{0,1\}^p \mapsto \{0,1\}$. You can apply threshold function after each neuron in the hidden layer.

(1) Using the structure and setting of NN above, we can obtain that the weight matrix and bias matrix of hidden layer

$$W^{(1)} = \begin{bmatrix} w_{11} & w_{21} \\ w_{21} & w_{22} \end{bmatrix},\quad b^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

The weight matrix and bias matrix of output layer are

$$W^{(2)} = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix},\quad b^{(2)} = [b_0]$$

Then we use $X = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

denotes the four possible inputs. Substitute into NN. the output of the first layer is

$$H^{(1)} = ReLU(W^{(1)T}X + B^{(1)})$$

$$= ReLU\left(\begin{bmatrix} 0 & w_{12} & w_{11} & w_{11}+w_{12} \\ 0 & w_{22} & w_{21} & w_{21}+w_{22} \end{bmatrix} + \begin{bmatrix} b_1 & b_1 & b_1 & b_1 \\ b_2 & b_2 & b_2 & b_2 \end{bmatrix}\right)$$

$$= ReLU\left(\begin{bmatrix} b_1 & w_{12}+b_1 & w_{11}+b_1 & w_{11}+w_{12}+b_1 \\ b_2 & w_{22}+b_2 & w_{21}+b_2 & w_{21}+w_{22}+b_2 \end{bmatrix}\right)$$

At the second layer. let its activation function be the identity function, then solve the equation

$$H^{(2)} = W^{(2)T}H^{(1)} + B^{(2)} = [0\ \ 1\ \ 1\ \ 0]$$

Namely

$$\begin{cases} w_{01}ReLU(b_1) + w_{02}ReLU(b_2) = 0 \\ w_{01}ReLU(w_{12}+b_1) + w_{02}ReLU(w_{22}+b_2) = 1 \\ w_{01}ReLU(w_{11}+b_1) + w_{02}ReLU(w_{21}+b_2) = 1 \\ w_{01}ReLU(w_{11}+w_{12}+b_1) + w_{02}ReLU(w_{21}+w_{22}+b_2) = 0 \end{cases}$$

The system of equations has multiple solutions, such as

$$W^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},\ b^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad W^{(2)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix},\ b^{(2)} = [0]$$

However, xor is not impossible for linear model
The convex sets formed by the positive instance point set and the negative instance point set may intercet, which will lead to linear inseparability.

(2) The original question is equivalent to proving that any Boolean function of p-element input can be approximated by a neural network with ReLU as the activation function.

Lemma 1. Any Boolean function containing p variables can be uniquely expressed as a Disjunctive Normal Form(DNF)

Lemma 2. A single hidden neuron can represent any conjunctive normal form.

$proof.$

The hidden layer:

$$a_k = ReLU(\sum_{i=1}^{p} w_{ki}X_i + b_k)$$

Lemma 1 and Lemma 2 show that a Boolean function containing p variables needs to be represented by at most 2^{p-1} neurons.

If

$$o = f(\sum_{i=1}^{n} w_{0i}a_i + b_0)$$

where n is the number of hidden neurons represents the output layer, then let $w_{0i} = 1,\ b_0 = 0$. In this way, if and only when all hidden neurons are not activated, 0 will be output, otherwise a positive number will be output, which plays the role of disjunction.

Lemma 3. The ReLU function is discriminatory

$proof.$

Let μ be a signed Borel measure, and assume the following holds for any $y\epsilon R$ and $\theta\epsilon R$:

$$\int ReLU(yx+\theta)d\mu(x) = 0$$

Aming to show that $\mu = 0$.

Consider the function:

$$f(x) = \begin{cases} 0\ if\ x < 0 \\ x\ if\ x\epsilon[0,1] \\ 1\ if\ x > 1 \end{cases}$$

Then

$$g(x) = ReLU(yx+\theta_1) - ReLU(yx+\theta_2)$$

Letting $\theta_1 = -\frac{\theta}{y}$ and $\theta_2 = -\frac{1-\theta}{y}$. if $y = 0$, then:

$$g(x) = f(\theta) = \begin{cases} ReLU(f(\theta)), if\ f(\theta) \geq 0 \\ -ReLU(-f(\theta)), if\ f(\theta) \leq 0 \end{cases}$$

For any $y\epsilon R$ and $\theta\epsilon R$:

$$\int f(yx+\theta)d\mu(x) = \int ReLU(yx+\theta_1) - ReLU(yx+\theta_2)d\mu(x)$$

$$= \int ReLU(yx+\theta_1)d\mu(x) - \int ReLU(yx+\theta_2)d\mu(x)$$

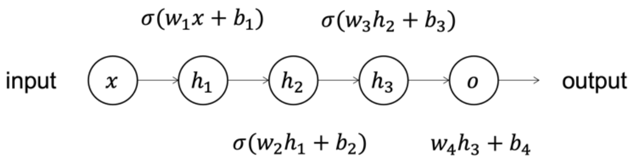
$$= 0 - 0$$

$$= 0$$

Then finish the proof.

2. [10pts] Gradient explosion and gradient vanishing

As shown in the figure below, the neural network has three hidden layers, each with only one neuron, and the activation function is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Let the input be $x = 3$. Use backpropagation to calculate the gradient, and experience the gradient explosion and gradient vanishing issues [1].



(1) [5pts] If $w_1 = 100, w_2 = 150, w_3 = 200, w_4 = 200, b_1 = -300, b_2 = -75, b_3 = -100, b_4 = 10$, calculate the gradient $\frac{\partial o}{\partial b_1}$.

(2) [5pts] If $w_1 = 0.2, w_2 = 0.5, w_3 = 0.3, w_4 = 0.6, b_1 = 1, b_2 = 2, b_3 = 2, b_4 = 1$, calculate the gradient $\frac{\partial o}{\partial b_1}$.

(1) $h_1 = \sigma(w_1 x + b_1) = \sigma(0) = \frac{1}{2}$

$h_2 = \sigma(w_2 h_1 + b_2) = \sigma(0) = \frac{1}{2}$

in the same way, $h_3 = \frac{1}{2}$, $o = 110$.

$$\frac{\partial o}{\partial b_1} = \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1}$$

$$= w_4 \cdot w_3 \sigma'(w_3 h_2 + b_3) \cdot w_2 \sigma'(w_2 h_1 + b_2) \cdot \sigma'(w_1 x + b_1)$$

$$= w_4 \cdot \prod_{i=2}^{3} w_i \sigma(w_i h_{i-1} + b_i)(1 - \sigma(w_i h_{i-1} + b_i)) \cdot \sigma(w_1 x + b_1)(1 - \sigma(w_1 x + b_1))$$

$$= w_4 \cdot \prod_{i=2}^{3} w_i h_i (1 - h_i) \cdot h_1 (1 - h_1) = 93750$$

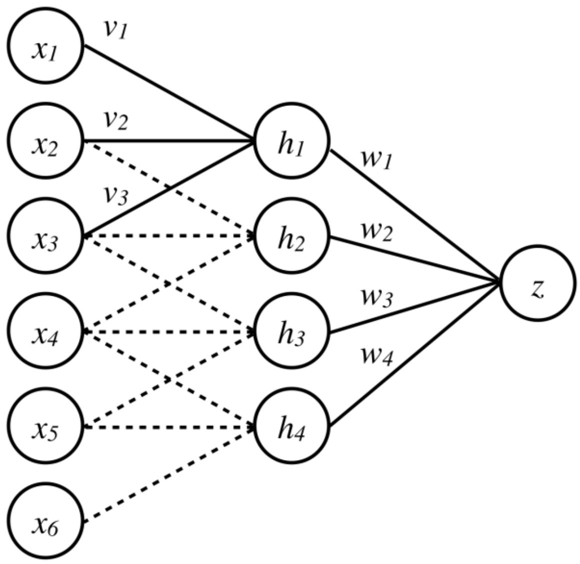
(2) By applying the same method in (1), we have $h_1 \approx 0.8332$, $h_2 \approx 0.918$. $h_3 \approx 0.88$.

$o = h_4 \approx 1.828$.

$$\frac{\partial o}{\partial b_1} = w_4 \cdot \prod_{i=2}^{3} w_i h_i (1 - h_i) \cdot h_1 (1 - h_1) \approx 0.000100$$

3. [25pts] CNN

Consider this **one-dimensional** convolutional neural network architecture.



In the first layer, we have a one-dimensional convolution with a single filter of size 3 such that $h_i = \sigma\left(\sum_{j=1}^{3} v_j\, x_{i+j-1}\right)$. The second layer is fully connected, such that $z = \sigma(\sum_{i=1}^{4} w_i\, h_i)$. The hidden units and output unit's activation function $\sigma(x)$ is the logistic (sigmoid) function with derivative $\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right)$. We perform gradient descent on the loss function $L = (y - z)^2$, where y is the training label for x.

a) [5pts] What is the total number of parameters in this neural network? Recall that convolutional layers share weights. There are no bias terms.

b) [10pts] Compute $\frac{\partial L}{\partial w}$

c) [10pts] Compute $\frac{\partial L}{\partial v_i}$

(a) There are $3+4=7$ parameters, namely $v_1, v_2, v_3, w_1, w_2, w_3$ and w_4.

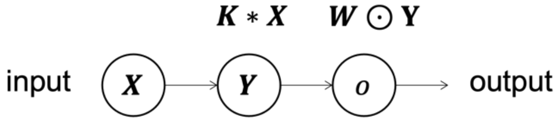
(b) $\frac{\partial R}{\partial w} = \frac{\partial R}{\partial z} \cdot \frac{\partial z}{\partial h^T w} \cdot \frac{\partial h^T w}{\partial w} = -2(y-z) \cdot z(1-z) \cdot h = 2(z-y)\, z(1-z)\, h.$

(c) $\frac{\partial R}{\partial v_i} = \frac{\partial R}{\partial z} \cdot \frac{\partial z}{\partial h^T w} \cdot \sum_{j=1}^{4} \frac{\partial h^T w}{\partial h_j} \cdot \frac{\partial h}{\partial v^T x} \cdot \frac{\partial v^T x}{\partial v_j}$

$= -2(y-z) \cdot z(1-z) \sum_{j=1}^{4} w_j h_j (1-h_j) x_{i+j-1}$

4. [30pts] CNN

As shown in the figure below, the neural network consists of a convolutional layer and a fully connected layer, with input as $\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}$, convolutional kernel (filter) as $\boldsymbol{K} = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$, convolution result as $\boldsymbol{Y} = \boldsymbol{K} * \boldsymbol{X} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$, and output as $o = \sigma(w_{11}y_{11} + w_{12}y_{12} + w_{21}y_{21} + w_{22}y_{22}), \sigma$ is sigmoid function. The sample label is z, and E is the mean squared error ($E = \frac{1}{2}\|z - o\|^2$). Find the derivative of the convolutional kernel [2].



Where, "*" is convolution, "⊙" is Hadamard product which is element-wise product.

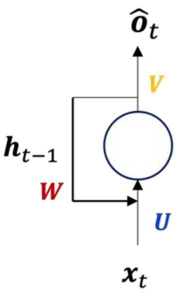
Solution. $\frac{\partial E}{\partial \boldsymbol{K}} = \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial Y} \cdot \frac{\partial Y}{\partial K}$

$= (z-o) \cdot \sigma'(W \odot Y) W \cdot \mathbb{I}_{2\times 2} * X$

$= (z-o) \cdot o(1-o) \cdot W \cdot \begin{bmatrix} x_{11} + x_{22} & x_{12} + x_{33} \\ x_{21} + x_{32} & x_{22} + x_{33} \end{bmatrix}$

5. [30pts] RNN: BPTT

Provide the detailed derivation process of the BPTT (Backpropagation Through Time) algorithm, ensuring that the symbols are consistent with those in the Lecture PPT.



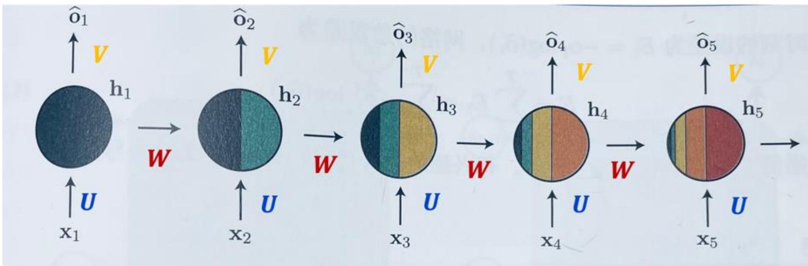


图 4.63 循环神经网络的基本结构

Solution.

$$s_t = U \cdot x_t + W \cdot h_{t-1}$$

$$h_t = \tanh(s_t) = \frac{e^{s_t} - e^{-s_t}}{e^{s_t} + e^{-s_t}}$$

$$z_t = V \cdot h_t$$

$$\hat{o}_t = g(V \cdot h_t) = \mathrm{softmax}(z_t)$$

$$E_t = -o_t \log \hat{o}_t$$

$$E = \sum_{t=1}^{T} E_t = \sum_{t=1}^{T} -o_t \log \hat{o}_t$$

BPTT:

1. $$\frac{\partial E}{\partial V} = \sum_{t=1}^{T} \frac{\partial E}{\partial z_t} \frac{\partial z_t}{\partial V} = \sum_{t=1}^{T} \sum_{k=1}^{T} \frac{\partial E}{\partial E_k} \frac{\partial E_k}{\partial \hat{o}_k} \frac{\partial \hat{o}_k}{\partial z_t} \cdot h_t$$

where $\frac{\partial E}{\partial E_k} = 1$. $\frac{\partial E_k}{\partial \hat{o}_k} = -\frac{o_k}{\hat{o}_k}$.

For $\frac{\partial \hat{o}_k}{\partial z_t}$: $\hat{o}_k = \mathrm{softmax}(z_k) = \frac{e^{z_k}}{\sum_{i=1}^{T} e^{z_i}}$

When $k = t$:

$$\frac{\partial \hat{o}_t}{\partial z_t} = \frac{\partial}{z_t}\left(\frac{e^{z_t}}{\sum_{j=1}^{T} e^{z_i}}\right) = \frac{e^{z_t}(\sum_{i=1}^{T} e^{z_i}) - e^{z_t} e^{z_t}}{(\sum_{i=1}^{T} e^{z_i})^2}$$

$$= \frac{e^{z_t}}{\sum_{i=1}^{T} e^{z_i}}\left(1 - \frac{e^{z_t}}{\sum_{i=1}^{T} e^{z_i}}\right) = \hat{o}_t (1 - \hat{o}_t).$$

Reference

[1] 王贝伦, “习题 4.24,” 出处 *机器学习*, 东南大学, 2021.

[2] 王贝伦, “习题 4.25,” 出处 *机器学习*, 东南大学, 2021.