ICTT

冯哲*
计算机科学与技术学院

2020 年 5 月 30 日

目录

*电子邮件: 1194585271@qq.com

1 命题时序逻辑

1.1 语法

① notation:

Z：整数;

N：正整数;

N_0：非负整数;

Prop：可数原子命题;

operate：$\neg, \Box, \bigcirc, \Diamond, ;, \vee, \wedge, +, *$

② 归纳定义命题公式:

1. $p \in Prop, p\ is\ a\ formula.$
2. $p, q \in Prop, so\ are\ the\ constructs.$

$p, q, p \wedge q, p; q, \bigcirc p, \Box q, p^+, q^*. and\ so\ on.$

1.2 语义

① 状态 S , B={true,false}

S : Prop $\rightarrow$ B　　{原子命题 $\rightarrow$ B}

状态是原子命题到真假集合的映射。

② 区间

区间 σ : 非空状态序列。

区间长度： 有限区间　finite　$|\sigma|$ ，　$|\sigma| = status - 1$

无限区间　infinite　w

统一★, $N_w = N_0 \vee \{w\}$

operator $=, <, \leq, \preceq$

对于　N_w ，$w = w$

对于 $i \in N_0$, $i < w$

$\preceq$ as $\leq -\{w, w\}$

定义:

1. $\sigma\ as\ < s_0, ..., s_{|\sigma|} >$, 若 $\sigma\ is\ infinite.$ $s_{|\sigma|}$ 无定义。
2. $\sigma_{(i,...j)}\ as\ (0 \leq i \preceq \leq |\sigma|)\ < s_i, ...s_j >$
3. $\sigma^{(k)}\ as\ (0 \leq k \preceq |\sigma|)\ < s_k, ..., s_{|\sigma|} >$

2 PTL练习题

常用基本符号优先级:

1. $\neg$
2. $\bigcirc, \Diamond, \Box, +, *$
3. $\wedge, \vee$
4. $\rightarrow, \leftrightarrow$
5. ;

2018/7/22

题 1 : 存在原子命题A,B，将下列命题用PTL公式表示:

① A发生，那么之后B总会发生（包括当前状态)。

② A发生，那么之前B总会发生（包括当前状态)。

答:

① $\Box(A \rightarrow \Diamond B)$

② $\neg(\Box\neg B; A)$

理解: chop 分为两个区间，两个区间上分别真假共四种情况，而此时否定一种，剩三种，与①的蕴含效果一样，具体情况为$\neg(P;Q)$被这样的区间满足: 对于该区间的任意一种两段划分，要么第一段不满足P，要么第二段不满足Q （可以第一段不满足P 同时第二段不满足Q)。

2018/7/23

题 2 : 存在原子命题A,B，假设有个词语"严格之后"表示之后的状态，但不包括当前状态。 "严格之前"同理，将下列命题用PTL公式表示:

① A发生，那么严格之后B总会发生（不包括当前状态)。

② A发生，那么严格之前B总会发生（不包括当前状态)。

答:

① $\Box(A \rightarrow \bigcirc\Diamond B)$

② $\neg((\Box(more \wedge \neg B)); A)$ × □ 和 more连用表示无穷区间

② $\neg((\Box(\ominus\neg B)); A)$ × $\ominus$不能和always直接连用，因为第一个状态没有上一状态。

② $(\neg A) \wedge ((\Diamond B; \bigcirc A) \vee (\Box\neg A))$?

② $(\neg A) \wedge \neg(\Box\neg B; skip; A)$ √ 题目的隐含意思有A不可能发生在起始状态

② $(\neg A) \wedge \neg(\Box\neg B; \bigcirc A)$ √ 题目的隐含意思有A不可能发生在起始状态

2018/7/25 Kun 整理笔记

时序逻辑性质表达 -2018/5/21

是朋友才能发消息，如何表示?
是朋友： f ； 发消息： s 。

① 经典逻辑表达：
a). $s \to f$ 发消息必是朋友。
b). $\neg f \to \neg s$ 不是朋友不能发消息
c). $\neg(\neg f \wedge s)$ 不是朋友且发消息是不对的 （与b)等价）
d). $f \vee \neg s$ 要么是朋友要么不发消息 （与b),c)等价）

② 时序逻辑表示：
a).将来能发消息，现在一定是朋友。

$$\Diamond(\Diamond s \to f)\ weak \qquad \Box(s \to f)$$

b).现在不是朋友，将来不能发消息。

$$\Box(\neg f \to \neg s)$$

c).现在不是朋友，将来能发消息是不对的。

$$\neg\Diamond(\Box\neg f; s) \qquad \Box\neg(\neg f \wedge s)$$

d).任何时刻，要么是朋友，要么不发消息。

$$\Box(f \vee \neg s)$$

e).在是朋友之前不能发消息。

$$\Diamond(s \to \boxdot f)$$

Runtime verification for LTL and TLTL ：

$$\neg s\ until\ f \equiv \Box\neg s; skip; f$$

第一部分　paper

3　A decision procedure for propositional projection temporal logic with infinite models

无限模型下命题投影时间逻辑（PPTL）的决策过程

(1) 无限模型下命题投影时间逻辑（PPTL）的可满足性;
(2) PPTL 公式的决策过程;
　(a) PPTL 的范式;
　(b) PPTL 的范式图;
　(c) 根据公式转换为范式;
　(d) 构造给定公式的范式图;
(3) 扩展 PPTL 的决策过程;

Question ① ITL, PTL, PPTL?

ITL(Interval Temporal Logic) 区间时序逻辑，规范和验证并发系统。

PTL 投影时序逻辑，相对于 ITL 扩展了无限模型和投影操作。

PPTL 命题投影时序逻辑。

决策过程是为了检查 PPTL 公式的可满足性条件。

3.1　PPTL

3.1.1　语法

$$P ::= p \mid \bigcirc P \mid \neg P \mid P_1 \vee P_2 \mid (P_1, \ldots, P_m)\ prj\ P$$

$\bigcirc$，prj，基本的时间运算符。

Question ② PPTL 的投影操作?

3.1.2　语义

1. state（状态）:
　命题公式到真假集的映射。$B = \{true, false\}$, $s : \text{Prop} \longrightarrow B$
2. interval（区间）:
　有限无限区间的统一定义;

区间长度比较操作：$=, <, \leq, \preceq$

3. `interpretation`（解释）：

三元组：$\mathcal{I} = (\sigma,\ k,\ j)$

σ，区间。

$k \preceq j \leq |\sigma|$

3.1.3 运算法则

L_1	$\Box(P \wedge Q) \equiv \Box P \wedge \Box Q$
L_2	$\Diamond(P \vee Q) \equiv \Diamond P \vee \Diamond Q$
L_3	$\bigcirc(P \vee Q) \equiv \bigcirc P \vee \bigcirc Q$
L_4	$\bigcirc(P \wedge Q) \equiv \bigcirc P \wedge \bigcirc Q$
L_5	$R;(P \vee Q) \equiv (R; P) \vee (R; Q)$
L_6	$(P \vee Q); R \equiv (P; R) \vee (Q; R)$
L_7	$\Diamond P \equiv P \vee \bigcirc\Diamond P$
L_8	$\Box P \equiv P \wedge \odot \Box P$
L_9	$\bar{\varepsilon} \wedge \neg \bigcirc P \equiv \bar{\varepsilon} \wedge \bigcirc\neg P$
L_{10}	$\neg \odot P \equiv \bigcirc\neg P$
L_{11}	$\bigcirc P; Q \equiv \bigcirc(P\ ; Q)$
L_{12}	$w \wedge (P\ ; Q) \equiv (w \wedge P)\ ; Q$
L_{13}	$Q\ prj\ \varepsilon \equiv Q$
L_{14}	$\varepsilon\ prj\ Q \equiv Q$
L_{15}	$(P_1, \ldots P_m)\ prj\ \varepsilon \equiv P_1; \ldots; P_m$
L_{16}	$(P, \varepsilon)\ prj\ Q \equiv (P \wedge \Diamond\varepsilon)\ prj\ Q$
L_{17}	$(P_1, \ldots, P_t, w \wedge \varepsilon, P_{t+1}, \ldots, P_m)\ prj\ Q \equiv (P_1, \ldots, P_t, w \wedge P_{t+1}, \ldots, P_m) prj Q$
L_{18}	$(P_1, \ldots, (P_i \vee P_i'), \ldots, P_m)\ prj\ Q \equiv ((P_1, \ldots, P_t, \ldots, P_m)\ prj\ Q) \vee ((P_1, \ldots, P_t', \ldots, P_m)\ prj\ Q)$
L_{19}	$(P_1, \ldots, P_m)\ prj\ (P \vee Q) \equiv ((P_1, \ldots, P_m)\ prj\ P) \vee ((P_1, \ldots, P_m)\ prj\ Q)$
L_{20}	$(P_1, \ldots, P_m)\ prj\ \bigcirc Q \equiv (P_1 \wedge \bar{\varepsilon}; (P_2, \ldots, P_m)\ prj\ Q) \vee (P_1 \wedge \varepsilon\ ; (P_2, \ldots, P_m)\ prj\ \bigcirc Q)$
L_{21}	$(\bigcirc P_1, \ldots, P_m)\ prj\ \bigcirc Q \equiv \bigcirc(P_1\ ; (P_2, \ldots, P_m)\ prj\ Q)$
L_{22}	$(w \wedge P_1, \ldots, P_m)\ prj\ Q \equiv w \wedge ((P_1, \ldots, P_m)\ prj\ Q)$
L_{23}	$(P_1, \ldots, P_m)\ prj\ (w \wedge Q) \equiv w \wedge ((P_1, \ldots, P_m)\ prj\ Q)$

第二部分 Weekly conversation

4 2019/7/16

1. 范式:将 PPTL 公式化成规范形式。

$$Q \equiv \bigvee_{j=1}^{n_0} (Q_{ej} \wedge \varepsilon) \vee \bigvee_{i=1}^{n} (Q_{ci} \wedge \bigcirc Q_i')$$

Q，PPTL范式；Q_p，原子命题公式；Q_i'，普通 PPTL 公式。

那么PPTL 公式怎么转为范式的四种情况：（最后一种为转为完全范式）。

(1) PPTL 公式转 $\mathrm{N_F}$。

(2) project 操作转 $\mathrm{N_F}$。

(3) chop 操作转 $\mathrm{N_F}$。

(4) $\mathrm{N_F} \rightarrow C\mathrm{N_F}$

例 $\mathrm{N_F}$ 转 $C\mathrm{N_F}$

$$\begin{aligned}(p \wedge \bigcirc P') \vee (q \wedge \bigcirc Q') \equiv & ((p \wedge q) \wedge \bigcirc (P' \vee Q')) \vee ((p \wedge \neg q) \wedge \bigcirc P') \\ & \vee ((\neg p \wedge q) \wedge \bigcirc Q') \vee ((\neg p \wedge \neg q) \wedge \bigcirc false)\end{aligned}$$

$$(p \wedge \bigcirc P') \vee (q \wedge \bigcirc Q') = false \wedge \epsilon \vee (p \wedge \bigcirc P') \vee (q \wedge \bigcirc Q')$$

2. 完全范式(作用：PPTL 完全范式否定简易)

$$Q \equiv (Q_e \wedge \varepsilon) \vee V_{i=1}^{n} (Q_i \wedge \bigcirc Q_i')$$

其中 $\vee_i Q_i \equiv true$ and $V_{i \neq j} (Q_i \wedge Q_j) \equiv false$。

3. 完全范式的否定

例 $C\mathrm{N_F}$ 的否定

$$\begin{aligned}\neg((p \wedge \bigcirc P') \vee (q \wedge \bigcirc Q')) \equiv & \neg\Big(((p \wedge q) \wedge \bigcirc (P' \vee Q')) \vee ((p \wedge \neg q) \wedge \bigcirc P') \\ & \vee ((\neg p \wedge q) \wedge \bigcirc Q') \vee ((\neg p \wedge \neg q) \wedge \bigcirc false)\Big) \\ \equiv & ((p \wedge q) \wedge \bigcirc \neg (P' \vee Q')) \vee ((p \wedge \neg q) \wedge \bigcirc \neg P') \\ & \vee ((\neg p \wedge q) \wedge \bigcirc \neg Q') \vee ((\neg p \wedge \neg q) \wedge \bigcirc \neg false)\end{aligned}$$

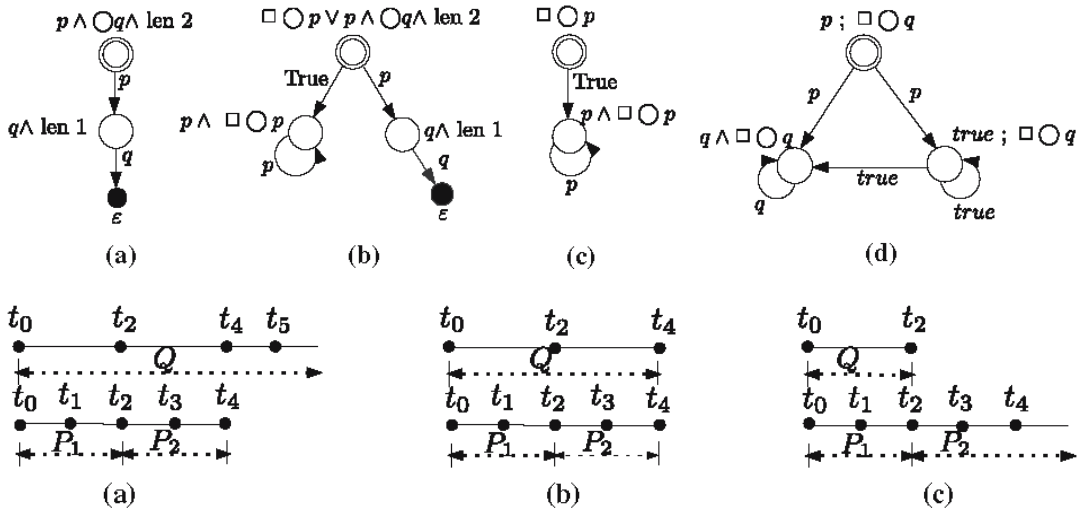
4. 范式图（重点）

PPTL 公式决策过程的可视化

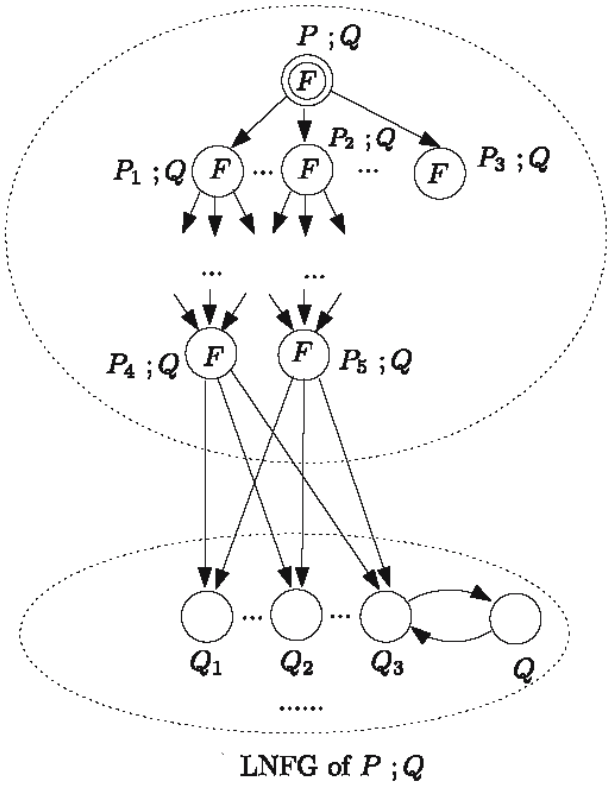
NFG 举例：

5. 带标签的范式图

由于 prj 操作有明确的语义，$(P_1, P_2)\ prj\ Q$ 的语义：



即就是 PPTL 公式 P_1 必须是有穷区间上的。但这一点在范式图上无法限制，所以加入标签。



4.1 练习

$$
\begin{aligned}
&\mathrm{NF}\left(\mathrm{O}^2(\Box \bigcirc p; q) \vee (p; \Box \bigcirc q)\right) \\
&\equiv \mathrm{NF}\left(\bigcirc^2(\Box \bigcirc p; q)\right) \vee \mathrm{NF}(p; \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee \mathrm{CHOP}(p; \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee \mathrm{CHOP}(\mathrm{NF}(p); \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee \mathrm{CHOP}(p \wedge \varepsilon \vee p \wedge \ \texttt{Otrue;}\ \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee \mathrm{CHOP}(p \wedge \varepsilon; \Box \bigcirc q) \vee \mathrm{CHOP}(p \wedge \ \texttt{Otrue;}\ \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee p \wedge \mathrm{CHOP}(\varepsilon; \Box \bigcirc q) \vee p \wedge \mathrm{CHOP}(\bigcirc true; \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee p \wedge (\mathrm{NF}(\bigcirc q \wedge \varepsilon) \vee \mathrm{NF}(\bigcirc q \wedge \bigcirc \Box \bigcirc q)) \vee p \wedge \bigcirc(\texttt{true}; \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee p \wedge (false \vee \bigcirc(q \wedge \Box \bigcirc q)) \vee p \wedge \bigcirc(\texttt{true}; \Box \bigcirc q) \\
&\equiv \bigcirc(\bigcirc(\Box \bigcirc p; q)) \vee p \wedge \bigcirc(q \wedge \Box \bigcirc q) \vee p \wedge \bigcirc(true; \Box \bigcirc q)
\end{aligned}
$$

第三部分　神经网络形式化

5　全连接神经网络

5.1　数学推导

5.1.1　作用过程示例

变量说明：

- 以零为下角标的变量为偏置（bias）。
- x 表示输入数据（一个训练样本为一个行向量），y 表示标签（one-hot 编码，为一个行向量），s 为求和值，a 为激活值。
- x_j^i 的上标表示第 i 行数据样本，下标表示该行数据样本的第 j 个值。
- x, s, a 若无下角标，则表示向量化。
- 多分类标签 y 采用 one-hot 编码。

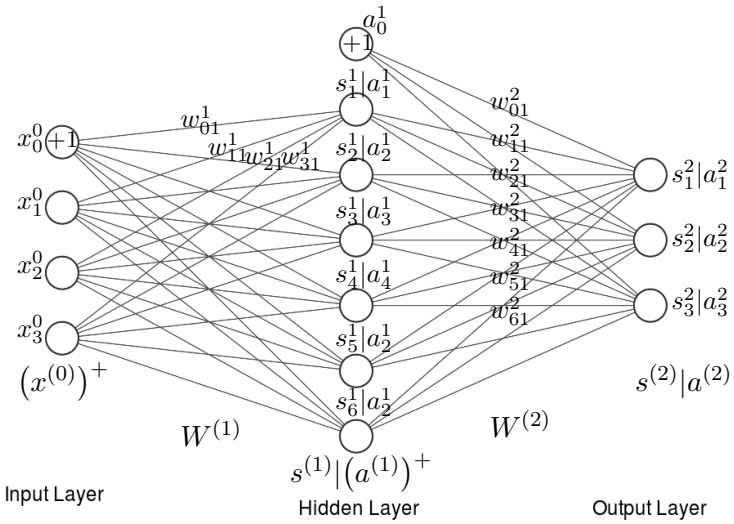
定义 1. 向量上标+ 运算符：为该向量添加*Bias*列，具体在第*0*列添加全*1*。

向量上标- 运算符：为该向量删去*Bias*列，具体删除第*0*列。

例 1.
$$a=\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad a^+=\begin{pmatrix} 1 & a & b & c \\ 1 & d & e & f \\ 1 & g & h & i \end{pmatrix} \quad a^-=\begin{pmatrix} b & c \\ e & f \\ h & i \end{pmatrix}$$

定义 2. 同维矩阵对应位置相乘运算：$A \circ B$.

输入层三个神经元，一个隐藏层六个神经元，输出层三个神经元(三分类)为例。激活函数以逻辑函数，损失函数以均方差损失为例。



作用过程：

1．向前传播计算误差：

$$\left(x^{(0)}\right)^{+}w^{(1)}=s^{(1)},\quad \sigma(s^{(1)})=a^{(1)}$$
$$\left(a^{(1)}\right)^{+}w^{(2)}=s^{(2)},\quad \sigma(s^{(2)})=a^{(2)}$$
$$Loss\quad L=\frac{1}{2}\sum_{k=1}^{3}(a_k^{(2)}-y_k)^2$$

假设有 1 个训练样本：

(1) x 的维度就是 1×3。

(2) x^{+} 的维度就是 1×4。

(3) $w^{(1)}$ 的维度就是 4×6。

(4) $s^{(1)}$ 的维度就是 1×6。

(5) $a^{(1)}$ 的维度就是 1×6。

(6) $\left(a^{(1)}\right)^{+}$ 的维度就是 1×7。

(7) $w^{(2)}$ 的维度就是 7×3。

(8) $s^{(2)}$ 的维度就是 1×3。

(9) $a^{(2)}$ 的维度就是 1×3。（输出层）

2．误差反向传播计算权值梯度：

(1) 输出层权值梯度：

$$Loss,\quad L=\frac{1}{2}\sum_{i=1}^{K}(a_i^{(2)}-y_i)^2$$
$$\frac{\partial L}{w_{ik}^{(2)}}=\frac{\partial}{\partial w_{ik}^{(2)}}\frac{1}{2}\sum_{i=1}^{K}(a_i^{(2)}-y_i)^2$$
$$=\left(\sigma(s_k^{(2)})-y_k\right)\sigma(s_k^{(2)})\left(1-\sigma(s_k^{(2)})\right)\frac{\partial s_k^{(2)}}{\partial w_{ik}^{(2)}}$$
$$=\left(\sigma(s_k^{(2)})-y_k\right)\sigma(s_k^{(2)})\left(1-\sigma(s_k^{(2)})\right)\left(a_i^{(1)}\right)^{+}$$
$$令,\quad \delta_k^{(2)}=\left(\sigma(s_k^{(2)})-y_k\right)\sigma(s_k^{(2)})\left(1-\sigma(s_k^{(2)})\right)$$
$$=\left(a_k^{(2)}-y_k\right)a_k^{(2)}\left(1-a_k^{(2)}\right)$$
$$则,\quad \frac{\partial L}{w_{ik}^{(2)}}=\delta_k^{(2)}\left(a_i^{(1)}\right)^{+}$$

(2) 隐藏层权值梯度：

$$\frac{\partial L}{\partial w_{ij}^{(1)}}=\frac{\partial}{\partial w_{ij}^{(1)}}\frac{1}{2}\sum_{k\in K}(a_k^{(2)}-y_k)^2$$
$$=\sum_{k\in K}(a_k^{(2)}-y_k)\frac{\partial\sigma(s_k^{(2)})}{\partial w_{ij}^{(1)}}$$

$$
\begin{aligned}
&= \sum_{k\in K}(a_k^{(2)}-y_k)\sigma(s_k^{(2)})\big(1-\sigma(s_k^{(2)})\frac{\partial s_k^{(2)}}{\partial w_{ij}^{(1)}} \\
&= \sum_{k\in K}(a_k^{(2)}-y_k)\sigma(s_k^{(2)})\big(1-\sigma(s_k^{(2)})\frac{\partial s_k^{(2)}}{\partial a_j^{(1)}}\frac{\partial a_j^{(1)}}{\partial w_{ij}^{(1)}} \\
&= \sum_{k\in K}(a_k^{(2)}-y_k)\sigma(s_k^{(2)})\big(1-\sigma(s_k^{(2)})w_{jk}^{(2)}\frac{\partial a_j^{(1)}}{\partial w_{ij}^{(1)}} \\
&= \frac{\partial a_j^{(1)}}{\partial w_{ij}^{(1)}}\sum_{k\in K}(a_k^{(2)}-y_k)\sigma(s_k^{(2)})\big(1-\sigma(s_k^{(2)})w_{jk}^{(2)} \\
&= a_j^{(1)}\big(1-a_j^{(1)}\big)\frac{\partial s_j^{(1)}}{w_{ij}^{(1)}}\sum_{k\in K}\delta_k^{(2)}w_{jk}^{(2)} \\
&= a_j^{(1)}\big(1-a_j^{(1)}\big)\big(x_i^{(0)}\big)^{+}\sum_{k\in K}\delta_k^{(2)}w_{jk}^{(2)}
\end{aligned}
$$

令，$\delta_j^{(1)} = a_j^{(1)}(1-a_j^{(1)})\sum_{k\in K}\delta_k^{(2)}w_{jk}^{(2)}$

则，$\dfrac{\partial L}{\partial w_{ij}^{(1)}} = \delta_j^{(1)}\big(x_i^{(0)}\big)^{+}$

3．更新权值：

$$W = W - \eta\frac{\partial Loss}{\partial W}$$

5.1.2 作用过程推广

用户指定神经网络相关参数：

变量名称	符号	备注
训练样本维度	N_t	
训练样本数量	N_T	
隐藏层数	H	包含H个隐藏层
第i层神经元个数	$N^{(i)}$	$i=0,1,\cdots,H,H+1$ $i=H+1$，输出层，$N^i=K$ 这里定义不带 bias 神经元。
第i层的激活函数	$\sigma^{(i)}$	在常用的 sigmoid，tanh，relu，leaky relu 中选择
分类类别数	K	
损失函数	S_l	在常用的 MeanSquareError，CorssEntropyLoss 中选择

常见激活函数：

1. sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$\sigma'(z) = \sigma(z)(1-\sigma(z))$$

2. tanh

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 1 - \frac{2e^{-z}}{e^z + e^{-z}}$$
$$\sigma'(z) = 1 - \sigma(z)^2$$

3. relu

$$\sigma(z) = \begin{cases} 0, & \texttt{if } z < 0 \\ z, & \texttt{otherwise.} \end{cases}$$
$$\sigma'(z) = \begin{cases} 0, & \texttt{if } z < 0 \\ 1, & \texttt{otherwise.} \end{cases}$$

4. leaky relu

$$\sigma(z) = \begin{cases} z, & \texttt{if } z > 0 \\ az, & \texttt{otherwise.} \end{cases}$$
$$\sigma'(z) = \begin{cases} 1, & \texttt{if } z > 0 \\ a, & \texttt{otherwise.} \end{cases}$$

5. softmax

第 i 个类目的概率，$p_i = a_i^{(N+1)} = \sigma(s_i^{(N+1)}) = \frac{e^{s_i^{(N+1)}}}{\sum_{j=1}^{K} e^{s_j^{(N+1)}}}$

$$\frac{\partial p_i}{\partial s_j} = \frac{\partial \frac{e^{s_i}}{\sum_{k=1}^{N} e^{s_k}}}{\partial s_j}$$
$$f(x) = \frac{g(x)}{h(x)}, \qquad f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{h^2(x)}$$
$$g(x) = e^{s_i}, \qquad h(x) = \sum_{k=1}^{N} e^{s_k}$$

$i == j$ $\qquad\qquad$ $i \neq j$

$$\frac{\partial \frac{e^{s_i}}{\sum_{k=1}^{N} e^{s_k}}}{\partial s_j} = \frac{e^{s_i}\sum_{k=1}^{N} e^{s_k} - e^{s_j}e^{s_i}}{\left(\sum_{k=1}^{N} e^{s_k}\right)^2} = \frac{e^{s_i}\left(\sum_{k=1}^{N} e^{s_k} - e^{s_j}\right)}{\left(\sum_{k=1}^{N} e^{s_k}\right)^2} = p_i(1-p_j)$$

$$\frac{\partial \frac{e^{s_i}}{\sum_{k=1}^{N}}}{\partial s_j} = \frac{0 - e^{s_j}e^{s_i}}{\left(\sum_{k=1}^{N} e^{s_k}\right)^2} = -\frac{e^{s_i}}{\sum_{k=1}^{N} e^{s_k}}\frac{e^{s^j}}{\sum_{k=1}^{N} e^{s_k}} = -p_i p_j$$

总结：

$$\delta_{ij} = \begin{cases} 1 & i == j \\ 0 & i \neq j \end{cases}$$

$$\Rightarrow \quad \frac{\partial p_i}{\partial s_j} = p_i(\delta_{ij} - p_j)$$

常见损失函数：

1. Mean Square Error

$$Loss(z,y) = \frac{1}{2}\sum_{i=1}^{K}(z_i - y_i)^2$$

$$\frac{\partial Loss}{\partial z} = (z - y) \quad \text{向量相减}$$

2. Cross Entropy Loss

$$Loss(z,y) = -\sum_{i=1}^{K} y_i \log z_i = -y_k \log z_k (\texttt{one-hot 只有第k个分量为 1})$$

$$\frac{\partial Loss}{\partial z} = -\frac{y}{z} \quad \text{向量相除}$$

一般如果多分类问题输出层使用 `CrossEntropy` 损失，并用 `Softmax` 激活。
因为有：

$$\frac{\partial Loss(a^{(N+1)}, y)}{\partial s_j^{(N+1)}} = a_j^{(N+1)} - y_j$$

推导：

$$\frac{\partial Loss(a^{(N+1)}, y)}{\partial s_j^{(N+1)}} = \sum_{i=1}^{K} \frac{\partial Loss(a^{(N+1)}, y)}{\partial a_i}\frac{\partial a_i}{\partial s_j}$$

$$
\begin{aligned}
&= -\sum_{i=1}^{K} \frac{y_i}{a_i}\frac{\partial a_i}{\partial s_j} \\
&= \big(-\frac{y_i}{a_i}\frac{\partial a_i}{\partial s_j}\big)_{i=j} - \sum_{i=1,i\neq j}^{K} \frac{y_i}{a_i}\frac{\partial a_i}{\partial s_j} \\
&= -\frac{y_j}{a_j}a_j(1-a_j) - \sum_{i=1,i\neq j}^{K} \frac{y_i}{a_i}\cdot -a_i a_j \\
&= -y_j + y_j a_j + \sum_{i=1,i\neq j}^{K} y_i a_j \\
&= -y_j + a_j \sum_{i=1}^{K} y_i \\
&= a_j^{(N+1)} - y_j
\end{aligned}
$$

变量说明：

变量	含义	备注
X	输入训练样本	维度 $N_T \times N_t$，相当于激活值矩阵$A^{(0)}$.
X^+	输入训练样本加偏置列	维度 $N_T \times (N_t+1)$，相当于激活值矩阵$\big(A^{(0)}\big)^+$.
$S^{(i)}$	第 i 层求和值矩阵	维度 $N_T \times N^{(i)}$，$i = 1, \cdots, H, H+1$.
$A^{(i)}$	第 i 层激活值矩阵	维度 $N_T \times N^{(i)}$，$i = 0, 1, \cdots, H, H+1$.
$\big(A^{(i)}\big)^+$	第 i 层激活值矩阵加偏置列	维度 $N_T \times (N^{(i)}+1)$，$i = 0, 1, \cdots, H$.
$W^{(i)}$	第 i 层权值矩阵	维度 $N^{(i-1)} \times N^{(i)}$，$i = 1, \cdots, H, H+1$.
$W^{(H+1)}$	输出层权值矩阵	维度 $N^{(N)} \times K$，标签有 K 个类目。
$W_b^{(i)}$	第 i 层权值偏置矩阵	维度 $(N^{(i-1)}+1) \times N^{(i)}$，$i = 1, \cdots, H, H+1$.
$W_b^{(H+1)}$	输出层权值偏置矩阵	维度 $(N^{(N)}+1) \times K$，标签有 K 个类目。
$S^{(H+1)}$	输出层求和值矩阵	维度 $N_T \times K$.
$A^{(H+1)}$	输出层激活值矩阵	维度 $N_T \times K$.
Y	训练数据标签	维度 $N_T \times K$ (ONE-HOT).
$\Delta^{(i)}$	第 i 层的反向传播中间变量矩阵	维度 $N_T \times N^{(i)}$，$i = 1, \cdots, H, H+1$.
$\Delta^{(H+1)}$	输出层反向传播中间变量矩阵	维度 $N_T \times K$.

神经网络向前传播

$$
\begin{aligned}
\big(A^{(0)}\big)^+ W_b^{(1)} = S^{(1)} \xrightarrow{\sigma+b} \big(A^{(1)}\big)^+ &\Rightarrow \cdots \Rightarrow \cdots \\
&\Rightarrow \big(A^{(H-1)}\big)^+ W_b^{(H)} = S^{(H)} \xrightarrow{\sigma+b} \big(A^{(H)}\big)^+ \\
&\Rightarrow \big(A^{(H)}\big)^+ W_b^{(H+1)} = S^{(H+1)} \xrightarrow{\sigma} A^{(H+1)} \\
&\Rightarrow \mathcal{L}(A^{(H+1)}, Y)
\end{aligned}
$$

神经网络反向传播

1．输出层权值求导：

$$\nabla_{W_b^{(H+1)}}\mathcal{L}(A^{(H+1)},Y)=\frac{1}{H_T}(A^{(H)^+})^T\Delta^{(H+1)} \qquad \text{（矩阵化）}$$

$$\Delta^{(H+1)}=\frac{\partial\mathcal{L}(A^{(H+1)},Y)}{\partial A^{(H+1)}}\circ\sigma'(S^{(H+1)})$$

当输出层使用 `Softmax+CrossEntropy` 时：

$$\Delta^{(H+1)}=A^{(H+1)}-Y$$

2．隐藏层权值求导：

$$\nabla_{W_b^{(H)}}\mathcal{L}(A^{(H+1)},Y)=\frac{1}{H_T}\left(A^{(H-1)^+}\right)^T\Delta^{(H)}$$

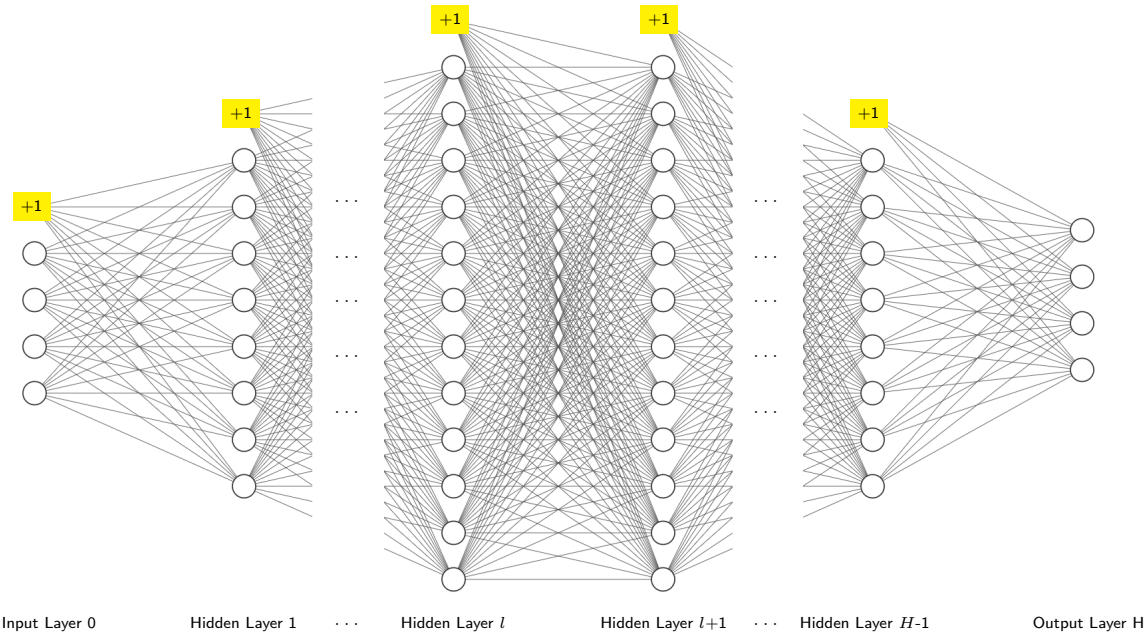
$$\text{令：}\ \Delta^{(H)}=\left(\Delta^{(H+1)}(W^{(H+1)})^T\right)\circ\sigma'^{(H)}(S^{(H)})$$

扩展： 对于第 i 个隐藏层的权值矩阵的梯度为($i=1,2,\cdots,H.$)：

$$\nabla_{W_b^{(i)}}\mathcal{L}(A^{(H+1)},Y)=\frac{1}{H_T}\left(A^{(i-1)^+}\right)^T\Delta^{(i)}$$

$$\text{令：}\ \Delta^{(i)}=\Delta^{(i+1)}(W^{(i+1)})^T\circ\sigma'^{(i)}(S^{(i)})$$

全连接神经网络结构图



神经网络张量流：

前向传播	维度	反向传播	维度
X	$N_T \times (N_t+1)$	LOSS	Scalar
$S^{(1)}$	$N_T \times N^{(1)}$	$\Delta^{(H+1)}$	$N_T \times K$
$(A^{(1)})^+$	$N_T \times (N^{(1)}+1)$	$\nabla_{W^{(H+1)}}\mathcal{L}$	$(N^{(H)}+1) \times K$
$S^{(2)}$	$N_T \times N^{(2)}$	$\Delta^{(H)}$	$N_T \times (N^{(H)}+1)$
$(A^{(2)})^+$	$N_T \times (N^{(2)}+1)$	$\nabla_{W^{(N)}}\mathcal{L}$	$(N^{(H-1)}+1) \times N^{(H)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$S^{(H)}$	$N_T \times N^{(H)}$	$\Delta^{(i)}$	$N_T \times (N^{(i)}+1)$
$(A^{(H)})^+$	$N_T \times (N^{(H)}+1)$	$\nabla_{W^{(i)}}\mathcal{L}$	$(N^{(i-1)}+1) \times N^{(i)}$
$S^{(H+1)}$	$N_T \times K$	$\Delta^{(1)}$	$N_T \times (N^{(1)}+1)$
$A^{(H+1)}$	$N_T \times K$	$\nabla_{W^{(1)}}\mathcal{L}$	$(N_t+1) \times N^{(1)}$
comput Loss		update Weight	

优化算法

批量梯度下降（Batch Gradient Descent，BGD） 批量梯度下降法是最原始的形式，它是指在每一次迭代时使用所有样本来进行梯度的更新。

1. 对目标函数求导：

$$\frac{\Delta J(\theta_0,\theta_1)}{\Delta\theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)x_j^{(i)}$$

其中 $i=1,2,\cdots,m$ 表示样本数，$j=0,1$ 表示特征数，这里我们使用了偏置项 $x_0^{(i)}=1$ 。

2. 每次迭代对参数进行更新：

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)x_j^{(i)}$$

伪码：

repeat{

$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$

(for $j = 0, 1$)

}

1．优点：

- 一次迭代是对所有样本进行计算，此时利用矩阵进行操作，实现了并行。
- 由全数据集确定的方向能够更好地代表样本总体，从而更准确地朝向极值所在的方向。当目标函数为凸函数时，BGD一定能够得到全局最优。

2．缺点：

- 当样本数目 m 很大时，每迭代一步都需要对所有样本计算，训练过程很慢。

随机梯度下降（Stochastic Gradient Descent，SGD） 随机梯度下降法不同于批量梯度下降，随机梯度下降是每次迭代使用一个样本来对参数进行更新。使得训练速度加快。

1．对一个样本的目标函数求偏导：

$$\frac{\Delta J^{(i)}\left(\theta_0, \theta_1\right)}{\theta_j} = \left(h_\theta \left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

2．参数更新：

$$\theta_j := \theta_j - \alpha \left(h_\theta \left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

伪码：

repeat{

for $i = 1, \cdots, m$ {

$\theta_j := \theta_j - \alpha \left(h_\theta \left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$

(for $j = 0, 1$)

}

}

1．优点：

- 由于不是在全部训练数据上的损失函数，而是在每轮迭代中，随机优化某一条训练数据上的损失函数，这样每一轮参数的更新速度大大加快。

2．缺点：

- 准确度下降。由于即使在目标函数为强凸函数的情况下，SGD仍旧无法做到线性收敛。
- 可能会收敛到局部最优，由于单个样本并不能代表全体样本的趋势。
- 不易于并行实现。

小批量梯度下降（Mini-Batch Gradient Descent, MBGD） 是对 BGD 以及 SGD 的一个折中办法。其思想是：每次迭代使用 batch_size 个样本来对参数进行更新。

这里我们假设 batch_size=10 ，样本数 m=1000 。

伪码：

repeat{

for i $= 1, 11, 21, 31, \ldots, 991$ {

$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{(i+9)} \left(h_\theta\left(x^{(k)}\right) - y^{(k)}\right) x_j^{(k)}$

(for $j = 0, 1$)

}

}

1．优点：

- 通过矩阵运算，每次在一个batch上优化神经网络参数并不会比单个数据慢太多。
- 每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。(比如上例中的30W，设置batch_size=100时，需要迭代3000次，远小于SGD的30W次)
- 可实现并行化。

2．缺点：

- batch_size的不当选择可能会带来一些问题。

5.2 MSVL 语言框架

数据类型：单精度浮点数。

5.2.1 辅助函数

1. absolute
2. min
3. equal
4. F2S
5. sigmoid
6. tanh
7. relu
8. leakyRelu

5.2.2 矩阵操作函数

1. MatCreate
2. MatDelete
3. MatSetVal
4. MatShape
5. MatDump
6. MatAdd
7. MatSub
8. MatMul
9. MatProduct
10. MatNumMul
11. MatNumAdd
12. MatTrans
13. MatZeros
14. MatOnes
15. MatEye
16. MatRowSum
17. MatRowMax
18. MatExp
19. MatSquare
20. MatVectorSub
21. MatVectorDiv

22. MatCopy
23. MatPlusRow
24. MatPlusCol

5.2.3 矩阵激活函数

1. MatNoneActi
2. MatSoftmax
3. MatSigmoid
4. MatTanh
5. MatRelu
6. MatLeakyRelu

5.2.4 矩阵激活函数导数函数

1. MatDerivationNoneActi 0
2. MatDerivationSigmoid 1
3. MatDerivationTanh 2
4. MatDerivationRelu 3
5. MatDerivationLeakyRelu 4
6. MatDerivationSoftmax 5

5.2.5 损失函数

1. OneHot
2. MSE
3. CrossEntropy

5.2.6 损失函数的导函数

1. MSEDerivative
2. CrossEntropyDerivative

5.2.7 神经网络的初始化

神经网络初始化介绍

1. AllZero initialization
2. Random initialization
3. Xavier initialization
4. He initialization

相关函数

1. gaussrand_NORMAL 标准高斯随机生成
2. gaussrand 高斯随机生成
3. MatInitZero 全零初始化
4. MatInitRandomNormalization 随机初始化
5. MatInitXavier Xavier初始化
6. MatInitHe 凯明初始化

5.2.8 神经网络用户自定义参数

用户自定义参数	参数表示	数据类型	备注
输入训练样本的数量	N_sample	int	NULL
单一训练样本的维度	D_sample	int	NULL
训练样本的真值	Xval	float *	数组长度为 N_sample * D_sample
训练样本的标签	Yval	float *	数组长度为 N_sample
权值初始化方式	Style_initWeight	int	0 ->全0初始化 1 ->随机初始化 2 ->Xavier初始化 3 ->何凯明初始化
神经网络隐藏层层数	N_hidden	int	注意：这里是隐藏层层数。
各层神经元个数	N_layerNeuron	int *	数组长度为 0(输入层),1,...,N_hidden,N_hidden+1(输出层).
各层激活函数	Nstr_ActiFsHidden	int *	数组长度为 0(输入层),1,...,N_hidden,N_hidden+1(输出层). 取值的映射关系： 0 ->no activation（input layer） 1 ->sigmoid 2 ->tanh 3 ->relu 4 ->leaky relu 5 ->softmax (output layer)
分类类别数	N_out	int	NULL
输出层损失函数	Nstr_LossF	int	取值映射关系： 0 ->MSE 1 ->CE

5.2.9 神经网络的运算所需空间变量及创建所需空间函数

神经网络的运算所需空间变量	参数表示	数据类型	备注
神经网络激活值矩阵	P_ActiMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层)
神经网络激活值矩阵加偏置列	P_ActiMatPlus	Mat *	列表索引i=0(输入层),1,...,N_hidden 加偏置列偏置列全部置1.输出层不需要Plus
神经网络求和矩阵	P_SumMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层) i=0 时输入层求和矩阵行列置零无实际含义
神经网络权值矩阵	P_WeightMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层) i=0 时输入层求和矩阵行列置零无实际含义
神经网络权值偏置矩阵	P_WeightBiasMat	Mat*	列表索引i = 0(输入层), 1, ..., N hidden, N hidden + 1(输出层) i = 0 时输入层求和矩阵行列置零无实际含义
训练数据标签	Mat_oneHot	Mat	row=N_sample col=N_out
反向传播中间变量矩阵	P_DeltaMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层) i=0 时输入层求和矩阵行列置零无实际含义
权值偏置矩阵导数变量矩阵	P_NablaWbMat	Mat*	列表索引i = 0(输入层), 1, ..., N hidden, N hidden + 1(输出层) i = 0 时输入层求和矩阵行列置零无实际含义
激活函数对求和值导数矩阵	P_ActiFunDerivation	Mat*	列表索引i = 0(输入层), 1, ..., N hidden, N hidden + 1(输出层) i = 0 时输入层求和矩阵行列置零无实际含义

创建所需空间函数：

1. SpaceCreateActi 创建激活值矩阵所需空间
2. SpaceCreateActiPlus 创建激活值Plus矩阵所需空间
3. SpaceCreateSum 创建求和值矩阵所需空间
4. SpaceCreateWeight 创建权值矩阵所需空间
5. SpaceCreateWeightBias 创建权值偏置矩阵所需空间
6. SpaceCreateDelta 创建反向传播中间变量矩阵所需空间

5.2.10 初始化神经网络

函数：

1. NNinit

说明：

1. 输入数据喂入激活值（plus）矩阵
2. 标签数据喂入矩阵并且进行独热编码
3. 选择常用的初始化方式进行权值初始化

5.2.11 神经网络前向传播

神经网络向前传播所需参数变量	参数表示	数据类型	备注
神经网络激活值矩阵	P_ActiMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层)
神经网络激活值矩阵加偏置列	P_ActiMatPlus	Mat *	列表索引i=0(输入层),1,...,N_hidden 加偏置列偏置列全部置1.输出层不需要Plus
神经网络求和矩阵	P_SumMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层) i=0 时输入层求和矩阵行列置零无实际含义
神经网络权值偏置矩阵	P_WeightBiasMat	Mat *	列表索引i = 0(输入层), 1, ..., N hidden, N hidden + 1(输出层) i = 0 时输入层求和矩阵行列置零无实际含义
训练数据标签	Mat_oneHot	Mat	row=N_sample col=N_out
神经网络隐藏层层数	N_hidden	int	注意：这里是隐藏层层数。
各层激活函数	NStr_ActiFsHidden	int *	数组长度为 0(输入层),1,...,N hidden,N hidden+1(输出层). 取值的映射关系： 0 ->no activation（input layer） 1 ->sigmoid 2 ->tanh 3 ->relu 4 ->leaky relu 5 ->softmax (output layer)
输出层损失函数	Nstr_LossF	int	取值映射关系： 0 ->MSE 1 ->CE

1. MatActivate 激活函数选择函数
2. LossFunction 损失函数选择函数
3. NNforward 神经网络前向传播函数-返回损失

5.2.12 神经网络反向传播

神经网络反向传播所需参数变量	参数表示	数据类型	备注
神经网络隐藏层层数	N_hidden	int	注意：这里是隐藏层层数
输入样本的数量	N_sample	int	样本数量（训练样本）
各层神经元个数	N_layerNeuron	int *	数组长度为 0(输入层),1,...,N hidden,N hidden+1(输出层)
神经网络求和矩阵	P_SumMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层)
i=0 时输入层求和矩阵行列置零无实际含义	P_ActiMat	Mat *	列表索引i=0(输入层),1,...,N_hidden, N_hidden+1(输出层)
神经网络激活值矩阵加偏置列	P_ActiMatPlus	Mat *	列表索引i=0(输入层),1,...,N_hidden 加偏置列偏置列全部置1.输出层不需要Plus
反向传播中间变量矩阵	P_DeltaMat	Mat *	列表索引i = 0(输入层), 1, ..., N hidden, N hidden + 1(输出层) i = 0 时输入层求和矩阵行列置零无实际含义
训练数据标签	Mat_oneHot	Mat	row = N_sample col = N_out
训练数据标签	Mat_oneHot	Mat	row=N_sample col=N_out
各层激活函数	NStr_ActiFsHidden	int *	数组长度为 0(输入层),1,...,N hidden,N hidden+1(输出层). 取值的映射关系： 0 ->no activation（input layer） 1 ->sigmoid 2 ->tanh 3 ->relu 4 ->leaky relu 5 ->softmax (output layer)
输出层损失函数	Nstr_LossF	int	取值映射关系： 0 ->MSE 1 ->CE
激活函数对求和值导数矩阵	P_ActiFunDerivation	Mat *	列表索引i = 0(输入层), 1, ..., N hidden, N hidden + 1(输出层) i = 0 时输入层求和矩阵行列置零无实际含义

1. ActiFunDerivation

2. LossFunDerivation
3. NNOuputLayerBackward
4. NNBackward

5.2.13 优化算法

1. BGD（批量梯度下降）
2. SGD（随机梯度下降）
3. MBGD（小批量梯度下降）
4. Adam

demo 成型，损失减小，可收敛。

5.3 MSVL 语言框架重构

5.3.1 对象结构体

1. 矩阵 结构体

```
typedef struct {
   int row, col;     // rowNum and columnNum [int]
   float** element; // element, two dimensions
}Mat;
```

2. BP单层 结构体

```
typedef struct{
   Mat ActiMat;              // active value Matrix without bias column
   Mat ActiMatPlus;          // active value Matrix with bias column
   Mat SumMat;               // sum value Matrix
   Mat WeightMat;            // weight value Matrix without bias row
   Mat WeightBiasMat;        // weight value Matrix with bias row
   Mat DeltaMat;             // backtrack temporary variable Matrix
   Mat NablaWbMat;           // gradient Matrix for weight with bias
   Mat ActiFunDerivationMat; // active Function Dervation Matrix

   int NeuronNum;            // number of neuron [int]
   int AcitFuncNum;          // active function [int]
}FCLayer;
```

3. BP神经网络 结构体

```
typedef struct{
   int CurrentSampleNum;          // number of current samples [int]
   int SampleDimensionNum; // dimensions(features) of sample [int]
   int HiddenLayerNum;       // number of hidden layer [int]
   int WeightInitWayNum;     // weight initialization mode [int]

   FCLayer *Layer;           // layers of FCNN
   Mat OnehotMat;            // onehot code Matrix

   int ClassificationNum;    // Number of categories classified [int]
   int LossFuncNum;          // loss function [int]
}FCNN;
```

4. 用户自定义参数 结构体

```
typedef struct{
    int CompleteSampleNum;  // number of all samples [int]
   int TrainSampleNum;      // number of training samples [int]
   int TestSampleNum;       // number of test samples [int]
   int ValidationNum;       // number of validation samples [int]
   int SampleDimensionNum; // dimensions(features) of sample [int]
   int HiddenLayerNum;      // number of hidden layer [int]
   int WeightInitWayNum;    // weight initialization mode [int]
   float *XValArray;        // samples features value [float*]
   float *YValArray;        // samples labels value [float*]
   int *NeuronNumArray;     // numbers of every layers Neuron [int*]
   int *ActiFuncNumArray;   // activate functions of every layers [int*]
   int ClassificationNum;   // Number of categories classified [int]
   int LossFuncNum;         // loss function [int]
}Custom;
```

5. 数据集 结构体

```
typedef struct{
   Mat CompleteFeatureDataSet; // complete featrue Mat for FCNN
   Mat CompleteLabelDataSet; // complete label Mat without onehot
   Mat CompleteTrainFeature; // complete featrue Mat for FCNN training
   Mat CompleteTrainLabel;  // complete label Mat without onehot
   Mat *BatchTrainFeature;  // batch featrue Mat for FCNN training [three
       dimensions]
   Mat *BatchTrainLabel;    // batch label Mat without onehot [three
       dimensions]
   Mat *BatchTrainLabelOneHot; // batch label Mat with onehot [three
       dimensions]
   Mat TestFeature;      // featrue Mat for FCNN test
   Mat TestLabel;        // label Mat without onehot
   Mat TestLabelOneHot;     // label Mat with onehot
   //Mat ValidationFeature; // featrue Mat for FCNN Validation
   //Mat ValidationLabel;  // label Mat withoutone


   int CompleteSampleNum;  // number of all samples [int]
   int TrainSampleNum;      // number of training samples [int]
   int TestSampleNum;       // number of test samples [int]
   //int ValidationNum;     // number of validation samples [int]
```

```
    int SampleDimensionNum; // dimensions(features) of sample [int]
    int ClassificationNum;  // Number of categories classified [int]
    int BatchSize;          // batch size for dataset
    int BatchNum;           // number of batch
    int remainder;          // the last batch size
}DataSet;
```

batch 机制的实现:建立 batch 结构体，分割数据集，复制到 FCNN 中。

其中 TrainSampleNum = BatchSize*BatchNum+remainder，当remainder = 0时，代表能被整除。

5.3.2 开发流程

1. 用户自定义参数及总数据录入;
2. 数据集构建;
3. 神经网络初始化;
4. 前向传播构建;
5. 反向传播构建;
6. 优化算法构建;
7. 神经网络训练构建;
8. 神经网络测试构建。

神经网络用户自定义参数及总数据录入 目的是将相关参数及总的数据赋值给用户自定义结构体中（其中注意对数组数据的赋值，传引用即可），再将各个参数分发给各个结构体。

数据集构建 根据用户输入的数据相关参数将训练测试数据集构建完成。构建路线:

1. 根据训练集数据量以及 Batch_Size 大小计算并导入相关参数。
2. 根据整体数据集的各个参数，开辟所需的全部空间。
3. 将整体数据集数组根据训练测试集数目划分成两个数组，包括训练特征数组和训练标签数组，测试特征数组和测试标签数组。
4. 再将训练特征数组和训练标签数组划分成若干个块大小的块训练特征数组和块训练标签数组。
5. 最后将数据喂入到相应的矩阵结构体当中。

神经网络初始化

1. 开辟神经网络所需空间并导入每层相关参数注意创建顺序不可变化。
   (1) 创建神经网络层空间
      <1> FCLayer
   (2) 导入每层参数
      <1> NeuronNum
      <2> AcitFuncNum
   (3) 创建神经网络运算空间
      <1> ActiMat
      <2> ActiMatPlus
      <3> SumMat
      <4> ActiFunDerivationMat
      <5> DeltaMat
      <6> OnehotMat（FCNN）
   (4) 创建神经网络参数空间(参数空间的大小不随输入神经网络样本大小而改变)
      <1> WeightMat
      <2> WeightBiasMat
      <3> NablaWbMat
2. 进行权值初始化
   (a) WeightMat
   (b) WeightBiasMat

神经网络前向传播求损失 神经网络向前传播将小批量输入样本导入神经网络输入层激活值矩阵，并将对应的标签数据进行独热编码，然后进行矩阵运算损失处理得到标量损失值。其中要注意小批量输入样本的数量不一定和已经开辟的神经网络有关矩阵空间的数量相一致，要做检查处理。

1. 小批量余数处理
2. 输入特征数据拷贝到激活值矩阵
3. 输入标签独热编码数据拷贝到神经网络独热编码矩阵中
4. 前向传播矩阵相乘，激活，扩展激活值矩阵
5. 预测值矩阵与真实值矩阵求损失，返回

神经网络反向传播求梯度 神经网络先经过一次正向传播得到损失之后，再求其损失对所有权值偏置的导数得到梯度，用到了反向传播算法。注意，每一次反向传播是建立

在一次正向传播求得损失的基础之上的。

1. 输出层反向传播，求得输出层的中间变量矩阵以及权值导数
2. 隐藏层反向传播，求得各隐藏层的中间变量矩阵以及权值导数

Adam算法构建 Adam算法使用了动量变量 v_t 和RMSProp算法中小批量随机梯度按元素平方的指数加权移动平均变量 s_t，并在时间步0将它们中每个元素初始化为0。
默认参数变量：$\beta_1 = 0.9, \beta_2 = 0.999, \eta = 10^{-3}, \epsilon = 10^{-8}$。

$$v_t = \beta_1 v_{t-1} + (1-\beta_1) g_t \tag{1}$$

$$s_t = \beta_2 s_{t-1} + (1-\beta_2) g_t \odot g_t \tag{2}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_1^t} \tag{3}$$

$$\hat{s}_t = \frac{s_t}{1-\beta_2^t} \tag{4}$$

$$g_t' = \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon} \tag{5}$$

$$x_t = x_{t-1} - g_t' \tag{6}$$

1. 创建Adam参数结构体。
2. 初始化结构体参数。
   (a) 初始化参变量为默认值。
   (b) 创建变量所需空间。
   (c) 时间步设置为0。
3. 根据六个公式更新梯度。

5.4 FCNN应用

5.4.1 Minst 手写字体识别

1. 数据集构建。

   minst 手写字体识别数据集包括一共 70000 张灰度图片，每张图片 28×28 个像素，图片数据集包括 0∼9 十类。全体数据集是经过出最大值归一化后的数据集。

2. 神经网络搭建。

```
CompleteSampleNum: 70000
TrainSampleNum:    60000
TestSampleNum:  10000
SampleDimensionNum: 784
HiddenLayerNum:    2
WeightInitWayNum: 3
ClassificationNum: 10
LossFuncNum:    1
BatchSize:   100
XValArray:(length = 54880000)
YValArray:(length = 70000)
NeuronNumArray:
784   512   256   10
ActiFuncNumArray:(include output layer Acti-Function)
0  3  3  5
OptimizationMethod: Mini-Batch Stochastic gradient descent
```

1. BSGD 收敛性不佳，指的是时间以及准确度上都达不到要求；
2. 开发Adam优化算法，提高准确度提高了五个百分点，以及收敛时间都缩短了1.5倍。

3. 对稀疏矩阵的运算做优化，收敛时间有缩短了一倍。

第四部分　开题报告

6　题目

基于MSVL的神经网络基础建模框架的设计与实现

Design and Implementation of Fundamental Modeling Framework for Neural Networks based on MSVL

7　论文概况

7.1　选题来源

国家自然科学基金项目

7.2　摘要

本文对神经网络的形式化建模方法进行研究，旨在为神经网络系统进行形式化验证提供模型理论基础。随着人工智能和神经网络的不断发展以及在计算机视觉，数据处理等领域的广泛应用，如何保障神经网络系统安全可靠成为了一个迫切需要解决的问题。目前，国内外关于神经网络系统安全性的研究集中在攻防方法的开发上，较少有工作使用形式化的手段对这类系统的安全性予以验证。本文以建模仿真语言MSVL以及投影时序逻辑PTL为基础，拟对神经网络的张量运算，初始化，激活函数，损失函数，优化方法等基本操作或者构件进行形式化建模研究，开发一套神经网络的基础建模框架，为CNN，RNN等其他类型的深度学习模型的形式化建模提供理论基础，为神经网络的形式化验证提供建模基础。

8　选题依据

8.1　选题意义

在如今飞速发展的机器学习和人工智能领域中，神经网络已经占据了重要地位。借助于卷积神经网络、循环神经网络等神经网络模型以及深度学习技术，神经网络系统拥有了出色的数据运算和类别预测能力。这类系统已成功应用于计算机视觉、自然语言处理、数据挖掘等领域，具备图像分类、语音识别、数据分析、智能推荐等丰富而方便的功能。其中不乏人们所熟知的系统，如人脸识别系统、自动驾驶汽车控制系统、阿尔法围棋等。

然而，即便神经网络系统已经展示了在处理复杂问题时所取得的成功，但是它们也

有其严重的安全漏洞。对于图像系统来说，具有较小扰动的对抗性样本不能为人所察觉，但是它们可以轻而易举地愚弄了神经网络模型。这种对抗攻击对神经网络系统造成了一系列威胁，也促使了这个方向上的大量研究投入。谷歌大脑也有研究表明，任何机器学习分类器都可能被欺骗，给出不正确的预测。在自动语音识别（ASR）系统中，深度循环网络已经取得了一定的成功，但是许多人已经证明，小的对抗干扰同样可以欺骗深层神经网络。

具体来说，在2016年6月，美国特斯拉公司的一辆 Model-S 型电动轿车在自动驾驶模式下撞上了一辆大型牵挂型卡车，导致一人当场身亡。事故原因是自动驾驶系统未能识别出前方卡车的白色侧面，以致没有及时启动刹车。2017年 11月的一个晚上，德国警方闯入一间公寓，因为他们接到邻居的报警，抱怨大清早就被隔壁震耳欲聋的音乐吵醒，然而这不是一个聚会，而是亚马逊的 Echo 在主人外出时，随机播放了音乐。2018年3月18日，一辆Uber自动驾驶车辆在美国亚利桑那州与一名行人相撞并致其死亡，成为全球首例自动驾驶致行人死亡事故。

值得指出的是，神经网络系统的形式化建模与验证并不是一个简单的问题。据我们了解，目前这一方面的研究工作也有限。究其原因，神经网络系统的运行依赖于由机器学习而得到的参数，其行为尚不透明、不可解释。如何为这类系统建立形式化模型以及如何对系统的安全性质进行形式化定义都是不小的挑战。此外，神经网络系统与大数据密切相关，系统的训练样本集和内部参数往往规模庞大，这无疑增加了验证的难度。神经网络系统的形式化建模与验证是一个迫切需要研究的问题，而这一问题的基础在于为神经网络建立形式化模型。

8.2 国内外研究现状

近年来，随着神经网络系统大规模的开发和应用，它们的安全性受到了研究人员越来越多的关注。由此开展了众多关于神经网络系统所面临的安全威胁及相应的防御技术的研究。

Szegedy 等人首次证明了可以通过对图像添加小量的人类察觉不到的扰动误导神经网络做出误分类。他们首先尝试求解让神经网络做出误分类的最小扰动的方程。但由于问题的复杂度太高，他们转而求解简化后的问题，即寻找最小的损失函数添加项，使得神经网络做出误分类，这就将问题转化成了凸优化过程。Ian J.Goodfellow 等人利用符号函数有目的地在图像样本上施加微小的扰动，提出了一种快速产生对抗样本的方法，并以高可信度在手写识别数据集上得到了验证。Moosavi-Dezfooli 等人通过迭代计算的方法生成最小规范对抗扰动，将位于分类边界内的图像逐步推到边界外，直到出现错误分类，结果证明他们生成的扰动比 FGSM 更小，同时有相似的欺骗率等。除了在神经网络系统分类任务攻击以外，还可以对自编码器和生成模型，循环神经网络，深度

强化学习，语义分割和物体检测上进行攻击，攻击效果也非常可观。而在这种对抗攻击上的防御策略也主要存在三个方向，修改训练过程或者样本；修改网络；附加网络。但都未利用到形式化的手段。

近些年来，有为数不多的利用形式化手段对神经网络的攻击与防御进行验证的工作。Huang 等人通过定义一组操纵对标准样本的邻域空间离散化，由此建立一组逻辑约束，并使用可满足模理论（SMT）求解器检验标准样本邻域空间中是否存在对抗样本。同样采用 SMT 求解技术，Scheibler 等人验证了倒立摆控制系统的特定性质，如状态的可达性。然而，其神经网络规模较小，仅包含 4 层共 16 个神经元。Katz 等人提出了一种针对仅采用修正线性单元（ReLU）激活函数的深度神经网络的验证方法。该方法通过对线性规划的单纯形法进行扩展，使其支持 ReLU 函数，进而对这类神经网络的线性性质加以验证。此外，Ghodsi 等人提出了一种对部署在云端的神经网络系统的验证方法。该方法对神经网络的结构做出了一些限制，如数据在有穷域内、仅采用二次激活函数等，通过一定的交互式证明协议验证系统的正确性。

目前为止，对于神经网络的验证都是基于神经网络的朴素的数学模型，并且仅考虑了模型中特定的方面，做了特定的简化假设，进而验证特定的性质。据本人所知，目前还没有工作采用一种形式化语言对神经网络的整体进行建模。本工作正是针对现状的这一不足旨在利用MSVL这种形式化语言，对神经网络的结构、行为（包括训练和预测）进行综合性建模。为进一步验证神经网络的性质提供基础。

9 研究方案

9.1 研究目标

对神经网络的形式化建模进行研究，形成基础的神经网络建模理论方法，支持对神经网络的训练、测试进行综合性建模。开发BP神经网络的建模工具，该工具可由用户自定义神经网络参数，生成神经网络的综合性形式化模型。

9.2 研究内容

为实现上述研究目标，本文基于形式化语言MSVL，研究神经网络各要素的建模方法，具体包括以下几点。

(1) 对逻辑，算数和矩阵运算进行形式化建模。
(2) 对神经网络的几种权值初始化方式进行形式化建模。
(3) 对神经网络的几种激活函数进行形式化建模。
(4) 对神经网络的几种损失函数进行形式化建模。
(5) 对神经网络的前项传播进行形式化建模。

(6) 对神经网络的反项传播进行形式化建模。

(7) 对基于梯度的优化方法进行形式化建模。

9.3 拟解决关键问题

(1) 如何利用基于时序逻辑的语言MSVL对神经网络进行形式化建模。

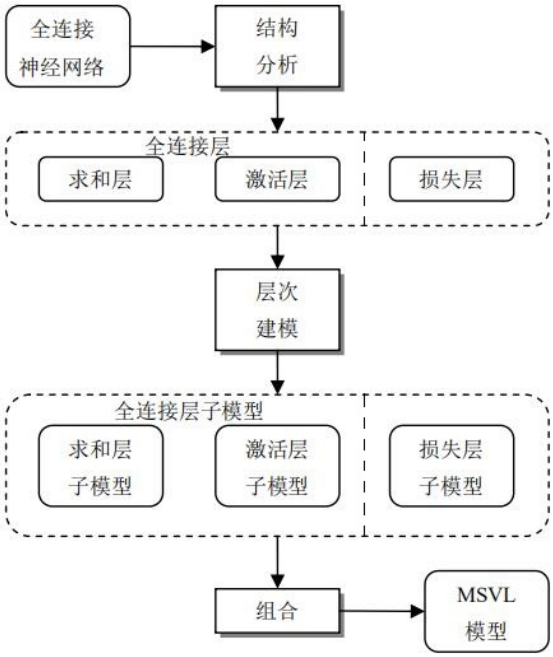
(2) 如何建立综合性的、可扩展的神经网络形式化模型。

9.4 拟采取的研究方法、技术路线、实验方案及可行性研究

(1) 研究方法:

实验研究，对现有的理论进行创新、完善或扩充，理论扩展其应用范围，并完成相应的模块单元测试，实现具体的模块功能。

(2) 技术路线:



关于形式化模型的选择，拟采用基于 MSVL 语言的模型。一方面，MSVL 语言由投影时序逻辑 PTL 所定义，具备严格的逻辑基础。另一方面，该语言支持多种基本类型与复合类型，函数、框架等重要机制以及顺序、并发、选择、循环等多种控制结构，适用于此系统的建模。

神经网络系统的形式化建模从系统的结构出发，逐层开展，完整的系统模型由各层次的模型组合而得。其核心步骤是包括数据结构和运算行为进行建模。之后的技术路线:

(a) 神经网络的前向传播进行建模。

(b) 神经网络的反向传播进行建模。

(c) 对优化算法进行建模。

例如，对于矩阵以及一个单层的神经网络课建模为如下形式化结构：

```
typedef struct {
   int row, col;     // rowNum and columnNum [int]
   float** element; // element, two dimensions
}Mat;
```

```
typedef struct{
   Mat ActiMat;              // active value Matrix without bias column
   Mat ActiMatPlus;          // active value Matrix with bias column
   Mat SumMat;               // sum value Matrix
   Mat WeightMat;            // weight value Matrix without bias row
   Mat WeightBiasMat;        // weight value Matrix with bias row
   Mat DeltaMat;             // backtrack temporary variable Matrix
   Mat NablaWbMat;           // gradient Matrix for weight with bias
   Mat ActiFunDerivationMat; // active Function Dervation Matrix

   int NeuronNum;            // number of neuron [int]
   int AcitFuncNum;          // active function [int]
}FCLayer;
```

(3) 实验方案：

(a) 查阅相关文献，学习投影时序逻辑的相关理论；

(b) 查阅文献和观看视频资料，学习经典神经网络的结构和建模细节；

(c) 利用MSVL进行神经网络模型搭建。

(4) 可行性分析：

(a) 论文《MSVL a Typed Language for Temporal Logic Programming》使用逻辑定义了一种编程语言，对常用的数据类型，顺序，条件，循环结构等进行了严格的逻辑定义，并开发了一套程序语言MSVL。

(b) 论文《Full Regular Temporal Property Verification as Dynamic Program Execution》提出了一种程序可动态执行统一模型检测方法，同时开发了统一模型检测器 UMC4MSVL；

(c) 论文《A compiler for MSVL and its applications》基于 LLVM 完成一个名为 MC 的编译器服务于建模仿真验证语言 MSVL，同时分析了 MC 应用于人工智能 AI 的可行性。

时间节点	预期研究成果
2020.03-2020.05	查阅相关文献，完成基本的调研任务，包括对 PTL 词法语法的理解；
2020.05-2020.06	研究 MC 编译器相关的内容，了解 MSVL 相关的编程技术；
2020.06-2020.08	研究经典深度神经网络的结构和训练细节；
2020.08-2020.10	完成逻辑运算，算数运算，矩阵运算的原型系统建模工作，分析并测试相关性质；
2020.10-2020.12	完成神经网络的初始化，几种损失函数，激活函数的建模工作，分析并测试相关性质；
2021.12-2021.03	完成神经网络的前向传播，梯度优化方式，反向传播的建模工作，并分析测试相关性质；
2021.03-2021.04	完善整体设计的流程，改善系统框架的结构，测试并验证最终的BP神经网络工具的分类回归性能；
2021.04-2021.05	撰写论文。

9.5 研究计划及预期取得的研究成果

10 研究基础

10.1 已具备的实验条件和研究工作积累

1. 研究条件：电脑、MC 编译器、C2MSVL 转换器、UMC4MSVL 验证器。
2. 研究工作积累：实验室具有一套完整的 PTL 模型理论，通过学习，对投影时序逻辑有了一定的认知，并且通过查阅深度神经网络技术相关的论文，有了比较成熟的设计思路。

10.2 已取得的科研成果

无。