(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 107797481 B
(45)授权公告日 2022.08.02

(21)申请号 201710799864.8

(22)申请日 2017.09.07

(65)同一申请的已公布的文献号
申请公布号 CN 107797481 A

(43)申请公布日 2018.03.13

(30)优先权数据
102016216948.3 2016.09.07 DE

(73)专利权人 罗伯特·博世有限公司
地址 德国斯图加特

(72)发明人 A.贡托罗

(74)专利代理机构 中国专利代理(香港)有限公司 72001
专利代理师 臧永杰 杜荔南

(51)Int.Cl.
G05B 19/042(2006.01)

(56)对比文件
CN 105981055 A,2016.09.28
CN 103282891 A,2013.09.04
CA 2149478 A1,1996.01.29
US 2009248450 A1,2009.10.01
詹璨铭.《将神经网络控制器用于VLSI设计的方法研究》.《微电子学与计算机》.2015,(第7期),
Zhai Jun.《A Method of Adaptive Neuron Model (AUILS) and Its Application》.《2006 5th IEEE International Conference on Cognitive Informatics》.2006,

审查员 郭向尚

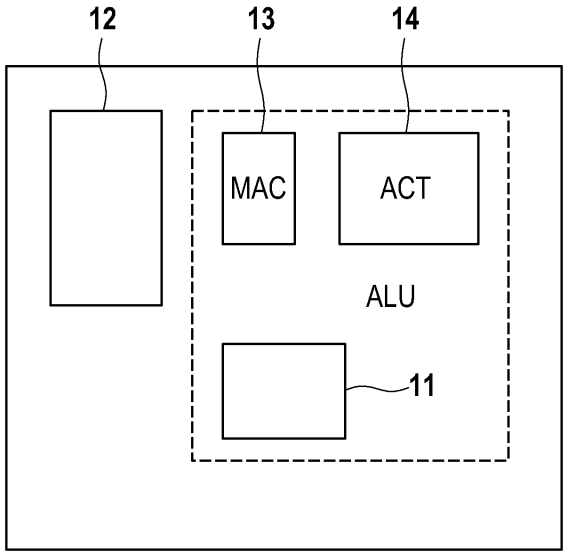
权利要求书1页 说明书7页 附图4页

(54)发明名称

用于计算神经元层的模型计算单元和控制设备

(57)摘要

本发明涉及一种用于选择性地计算多层感知器模型的层和至少一个另外的基于数据的函数模型的模型计算单元,其具有以硬件构造的固定接线的计算内核,计算内核用于计算在耦合的功能块中固定地预先给定的计算算法,其中,计算内核具有状态机和运算块,其中状态机预先给定用于计算多层感知器模型的所述层和所述至少一个另外的基于数据的函数模型的计算操作,其中状态机此外还被构造用于可选择地在计算感知器模型的所述层之前或者计算所述至少一个另外的基于数据的函数模型之前进行对输入参量的输入变换和/或在计算感知器模型的所述层之后或者计算所述至少一个另外的基于数据的函数模型之后进行对输出参量的输出变换。



CN 107797481 B

1.一种用于选择性地计算多层感知器模型的神经元层和至少一个另外的基于数据的函数模型的模型计算单元(22),所述模型计算单元具有以硬件构造的固定接线的计算内核,所述计算内核用于计算在耦合的功能块中固定地预先给定的计算算法,其中,所述计算内核具有状态机(11)和运算块(13、14),其中所述状态机(11)预先给定用于计算所述多层感知器模型的神经元层和所述至少一个另外的基于数据的函数模型的计算操作,

其中所述状态机(11)此外还被构造用于可选择地在计算所述感知器模型的神经元层之前或者计算所述至少一个另外的基于数据的函数模型之前进行对输入参量的输入变换和/或在计算所述感知器模型的神经元层之后或者计算所述至少一个另外的基于数据的函数模型之后进行对输出参量的输出变换。

2.按照权利要求1所述的模型计算单元(22),其中,所述计算内核被构造用于,对于具有一定数目的神经元(20)的多层感知器模型的神经元层,根据输入参量向量ut的一个或多个输入参量、具有权重因子v_{jk}的权重矩阵和针对每个神经元(20)所预先给定的偏置值来计算每个神经元(20)的输出参量y[j],其中针对每个神经元(20)而言,用预先给定给所述神经元(20)的偏置值来加载利用由所述神经元(20)和所述输入参量所确定的权重因子v_{jk}所加权的输入参量的值的总和,并且利用激活函数act对结果进行变换,以便得到所述神经元(20)的输出参量y[j]。

3.按照权利要求1或2所述的模型计算单元(22),其中,所述状态机(11)此外还被构造用于可选择地在计算所述感知器模型的神经元层之前或者计算所述至少一个另外的基于数据的函数模型之前进行所述输入参量到另一存储区的复制过程和/或在计算所述感知器模型的神经元层之后或者计算所述至少一个另外的基于数据的函数模型之后进行所述输出参量到另一存储区的复制过程。

4.按照权利要求1或2所述的模型计算单元(22),其中,所述计算内核被构造用于根据选择参量cfg_activation_function选择用于所述多层感知器模型的激活函数的类型和/或借助于另一选择参量来选择:应该计算高斯过程模型或RBF模型还是应该计算所述感知器模型的神经元层。

5.按照权利要求1或2所述的模型计算单元(22),其中,所述计算内核被构造在集成模块的表面区域内。

6.用于运行按照权利要求1至5之一所述的模型计算单元(22)的方法,其中,根据对所述感知器模型的神经元层的计算和/或根据选择变量来跳过对输入参量的所述输入变换和/或对输出参量的所述输出变换。

7.控制设备,所述控制设备具有微处理器和一个或多个按照权利要求1至5之一所述的模型计算单元。

8.按照权利要求7所述的控制设备,其中,所述控制设备(2)被构造为集成电路。

9.按照权利要求7或8所述的控制设备作为用于控制机动车中的发动机系统(1)的控制设备的用途。

用于计算神经元层的模型计算单元和控制设备

技术领域

[0001] 本发明涉及对单独的硬接线的模型计算单元中的函数模型、尤其是用于计算多层感知器模型的函数模型的计算。

背景技术

[0002] 经常利用模型来实现对技术系统(诸如内燃机、电动驱动装置、蓄电池和诸如此类)的控制的功能,所述模型表示真实系统的数学模型(Abbild)。然而,在物理模型情况下、尤其是在复杂关系的情况下缺乏必要的计算精确度,并且在如今的计算能力的情况下通常难以在对于控制设备来说所需的实时要求之内对这样的模型计算。针对这种情况设想使用基于数据的模型,所述基于数据的模型仅仅基于借助于试验台或者诸如此类所得到的训练数据来描述在输出参量和输入参量之间的关系。尤其是,基于数据的模型适合用于对复杂关系进行建模,在所述复杂关系的情况下,在模型中以适当的方式考虑多个输入参量,在所述输入参量之间存在相互关系。此外,借助于基于数据的模型进行的建模还提供如下可能性:通过添加单个输入参量来对该模型补充。

[0003] 基于数据的函数模型通常基于大量控制点(Stützstellen),以便实现对于相应应用足够的建模精确度。由于大量控制点,为了利用基于数据的函数模型(诸如高斯过程模型)计算模型值,需要高计算能力。因而,为了可以在控制设备应用中实时地计算这样的基于数据的函数模型,可以设置基于硬件设计方案的模型计算单元。

发明内容

[0004] 根据本发明,设置用于计算多层感知器模型的层的模型计算单元以及控制设备和所述控制设备的用途。

[0005] 在具体实施方式中说明另外的设计方案。

[0006] 上述模型计算单元规定如下设计方案,所述设计方案实现:计算具有可变数目的神经元的多层感知器模型(MLP模型)的神经元层或者至少一个另外的基于数据的函数模型。

[0007] 根据第一方面是一种用于选择性地计算多层感知器模型的神经元层和至少一个另外的基于数据的函数模型的模型计算单元,其中所述模型计算单元具有以硬件构造的固定接线的计算内核(Rechenkern),所述计算内核用于计算在耦合的功能块(Funktionsblöcken)中固定地预先给定的计算算法,其中所述计算内核具有状态机和运算块,其中所述状态机预先给定用于计算多层感知器模型的神经元层和至少一个另外的基于数据的函数模型的计算操作,其中所述状态机此外还被构造用于可选择地在计算感知器模型的神经元层之前或者计算至少一个另外的基于数据的函数模型之前进行对输入参量的输入变换和/或在计算感知器模型的神经元层之后或者计算至少一个另外的基于数据的函数模型之后进行对输出参量的输出变换。

[0008] 上述模型计算单元的思想在于:所述模型计算单元在硬件结构中在控制设备中的

计算内核中单独地被构造用于计算多层感知器模型的神经元层和至少一个另外的基于数据的函数模型。以这种方式可以提供基本上固定接线的硬件电路用于实现如下功能，所述功能实现：计算多层感知器模型的一个或多个神经元层并且在此在控制设备的软件控制的微处理器中只引起非常小的计算负荷。通过由模型计算单元提供的硬件加速可以实时地计算多层感知器模型或者另一基于数据的函数模型，使得将这种模型用于机动车中的内燃机的控制设备应用的用途变得令人感兴趣。

[0009] 由于进行或跳过对输入参量或输出参量的输入变换和/或输出变换的可能性，可以根据要计算的模型的类型而定地来进行输入参量或输出参量的适配。

[0010] 该模型计算单元可以配备接口，以便逐层地计算MLP模型，使得MLP神经元层的数目和在每个神经元层中的神经元的数目都可以被自由地选择。通过所述逐层的划分，可以为了每个神经元层单独地预先给定参数、诸如突触权重。

[0011] 计算内核可以被构造用于，为了计算具有一定数目的神经元的多层感知器模型的神经元层，根据输入参量向量的一个或多个输入参量、具有权重因子的权重矩阵和针对每个神经元所预先给定的偏置值来计算每个神经元的输出参量，其中针对每个神经元来计算利用由神经元和输入参量所确定的权重因子所加权的输入参量的值的总和与预先给定给所述神经元的偏置值，并且利用激活函数对结果进行变换，以便得到所述神经元的输出参量。

[0012] 此外，根据一种实施方式，状态机可被构造用于，可选择地在计算感知器模型的神经元层之前或者计算至少一个另外的基于数据的函数模型之前进行输入参量到另一存储区的复制过程和/或在计算感知器模型的神经元层之后或者计算至少一个另外的基于数据的函数模型之后进行输出参量到另一存储区的复制过程。

[0013] 计算内核可以被构造用于根据选择参量(cfg_activation_function)选择用于多层感知器模型的激活函数的类型和/或借助于另一选择参量来选择：应该计算高斯过程模型或RBF模型还是应该计算感知器模型的神经元层。

[0014] 计算内核可以被构造在集成模块的表面区域(Flächenbereich)内。

[0015] 根据另一方面，设置一种用于运行上述模型计算单元的方法，其中在计算感知器模型的神经元层时和/或或者根据选择变量来跳过对输入参量的输入变换和/或对输出参量的输出变换。

[0016] 根据另一方面，设置一种具有微处理器和一个或多个上述模型计算单元的控制设备。

[0017] 所述控制设备尤其可以被构造为集成电路。

[0018] 根据另一方面，设置上述控制设备作为用于控制机动车中的发动机系统的控制设备的用途。

附图说明

[0019] 随后依据随附的附图进一步阐述多种实施方式。其中：

[0020] 图1示出用于针对机动车中的发送机系统来使用的控制设备的示意图；

[0021] 图2示出作为控制设备的部分的计算单元的示意图；

[0022] 图3示出MLP模型的神经元层的示意图；和

[0023] 图4a-4d示出可能的激活函数的图。

具体实施方式

[0024] 图1示例性示出用于作为要控制的技术系统的具有内燃机3的发动机系统1的控制设备2。可替代的系统、诸如具有电动机的发动机系统或者电池系统可以类似的方式由这样的控制设备来控制。控制设备2包括微处理器21和模型计算单元22,所述微处理器和模型计算单元可以被构造为单独的构件或者以集成的方式被构造在芯片上的单独的表面区域内。模型计算单元22尤其是硬件电路,所述硬件电路在结构上可以与微处理器21的计算内核分开。

[0025] 模型计算单元22基本上硬接线并且与此相应地没有像微处理器21那样被构造用于实施软件代码并且由此实施可变的、通过软件所预先给定的函数。换言之,在模型计算单元22中没有设置处理器,使得所述模型计算单元不能通过软件代码来运行。通过聚焦到(Fokussierung auf)预先给定的模型函数上,使得能够资源优化地实现这种模型计算单元22。可以以集成的结构方式以面优化的方式来实现模型计算单元22,所述模型计算单元此外还实现快速的计算。

[0026] 控制设备2基本上用于:处理由内燃机3中的传感装置所检测的传感器信号S或传感器参量和/或外部的预给定参数(Vorgabe)V,并且以固定地预先给定的时间间隔循环地、也就是说为了控制内燃机而周期性地在例如1ms与100ms值之间的循环时间之内将一个或多个相应的操控参量A的值施加给内燃机3,或者以角度同步(与曲轴的位置同步)的方式根据所运行的内燃机的曲轴角来将一个或多个相对应的操控参量A的值施加给内燃机3(oder winkelsynchron (synchron zur Stellung einer Kurbelwelle) in Abhängigkeit zu einem Kurbelwellenwinkel eines betriebenen Verbrennungsmotors von einer oder mehreren entsprechenden Ansteuergrößen A an den Verbrennungsmotor 3 anzulegen),使得该内燃机能够以本身已知的方式来运行。

[0027] 在图2中更详细地示出了模型计算单元22。模型计算单元22包括状态机11、存储器12和一个或多个运算块,所述运算块例如包括一个或多个MAC块13(MAC:Multiply-Accumulate(乘积累加),用于定点和浮点计算)和用于计算激活函数的激活函数计算块14。所述状态机11和所述一个或多个运算块13、14形成模型计算单元22的计算内核ALU。运算块可以对于MAC块而言附加地或者替代地包括乘法块和加法块。

[0028] 借助于状态机11,可以通过重复的循环计算(Schleifenberechnung)来对保存在存储器12的输入参量存储区中的输入参量的值结算(verrechnen),以便得到中间参量或输出参量,所述输出参量被写到存储器12的相应的输出参量存储区中。

[0029] 状态机11因此被设计用于计算多层感知器模型的单个神经元层。状态机11可依据随后的伪代码来描述:

[0030] /*输入变换 */

[0031] for (k=0; k<p7; k++) {

[0032] ut[k] = u[k]*p1[k] + p2[k];

[0033] }

[0034] /* 循环计算 */

[0035] for (j=p8; j<p6; j++) {

[0036] i = j * p7;

[0037] t = p3[j];

[0038] for (k=0; k<p7; k++) {

[0039] t += V[i+k] * ut[k];

[0040] }

[0041] y[j] = act(t);

[0042] }

[0043] /* 输出变换 */

[0044] for (k=0; k<p6; k++) {

[0045] z[k] = y[k] * p4[k] + p5[k];

[0046] 其中：

[0047] p7:用于输入参量向量的输入参量的最大索引值

[0048] p8:用于神经元的数目的最小索引值或预先给定的初始索引

[0049] p6:用于神经元的数目的最大索引值

[0050] p3:偏置值

[0051] p1、p2:用于输入变换的变量

[0052] p4、p5:用于输出变换的变量。

[0053] 借助于上述伪代码可以执行针对计算的神经元层的每个神经元的如下计算：

[0054] 对于j=0…p6-1: $y[j] = act\left(p3[j] + \sum_{k=0}^{p7-1} v_{j,k} * ut[k]\right)$

[0055] 如在图3中示出的那样，这表示针对多层感知器模型的神经元层的计算。

[0056] 图3示出多个神经元20的神经元层，输入参量向量$ut_0 \cdots ut_{p6-1}$的输入参量的值被输送给所述多个神经元。输入参量的值借助于相应的预先给定的由权重值所构成的权重矩阵来加权，所述权重值被设置为权重因子$V_{0\cdots p7-1,0\cdots p6-1}$。通常，通过用分配的权重因子$V_{0\cdots p7-1,0\cdots p6-1}$以相乘方式加载(beaufschlagen mit…)来进行加权，然而也可以以其他方式加载输入参量向量的值。

[0057] 输入参量向量$ut_0 \cdots ut_{p6-1}$的经加权的值的总和分别被加载有偏置值$O_0 \cdots O_{p6-1}$、尤其是以相加方式被加载。利用预先给定的激活函数“act”对结果变换。作为结果得到输出参量向量$y_0 \cdots y_{p6-1}$的相应的值。由于针对每个神经元设置偏置值，存在对于形成模型而言的另一自由度。

[0058] 通过规定控制变量(Laufvariable)p6可以调整要计算的神经元层的神经元20的数目。多层感知器模型可以通过将神经元层的输出参量向量$y_0 \cdots y_{p6-1}$的值用作为输入参量向量来被用于计算在模型计算单元22中的随后的神经元层，使得通过根据上述伪代码重复地调用函数或通过利用不同的参数重复地调用模型计算单元22可以实现多层感知器模型的神经元层的所述数目。

[0059] 借助于针对每个神经元预先给定的标准化变量(Normierungsvariable)p1和p2或p4和p5可以进行对输入参量向量的输入参量或输出参量向量的输出参量的输入变换和/或输出变换。

[0060] 对MLP模型的逐层计算实现模型计算单元22的纤细的(schlanke)设计方案，使得

其面积需求以集成的结构方式是小的。尽管如此，模型计算单元22仍实现：以简单的方式通过将输出参量向量的输出参量的值反馈(rückführen)或者重新定义为用于计算另一神经元层的输入参量向量的输入参量，计算多层感知器模型。

[0061] 作为激活函数“act”可以提供能通过模型计算单元22的激活函数计算块14来计算的多个激活函数之一。激活函数例如可以使用折线函数(Knickfunktion)、Sigmoid函数、正切双曲线函数或者线性函数，如它们在图4a至4d中相应示出的那样。

[0062] 此外，通过神经元模型的单层的构造而可能的是，通过简单地修改，除了MLP模型的神经元层之外也计算高斯过程模型或RBF模型(RBF：径向基函数)，其中通过上述伪代码实现所述神经元模型。为此，权重值不是以相乘方式加载到输入参量的值上，而是以相加或相减的方式加载到输入参量的值上。此外，计算距离平方(quadratische Abstand)，所述距离平方利用预先给定的长度标度L[k]来加权。此外，针对RBF模型选择指数函数作为激活函数。因此可以对应于

[0063] $$y = \sum_{j=0}^{p6-1} p3[j] \cdot exp\left(-\sum_{k=0}^{p7-1} L[k] \cdot (-v_{j,k} + ut[k])^2\right)$$

[0064] 通过对伪代码的修改选择性地如下计算高斯过程模型。

```
[0065] /* 输入变换*/
[0066] for (k=0; k<p7; k++) {
[0067] ut[k] = u[k]*p1[k] + p2[k];
[0068] }
[0069] /* 循环计算*/
[0070] for (j=p8; j<p6; j++) {
[0071] i = j * p7;
[0072] t = (cfg_mlp) ?p3[j] : 0.0f; // 以用于MLP的偏置来初始
[0073] for (k=0; k<p7; k++) {
[0074] if (cfg_mlp) {
[0075] t += V[i+k] * ut[k];
[0076] }
[0077] else {
[0078] d = V[i+k] - ut[k];
[0079] d = d * d;
[0080] t += L[k] * d
[0081] }
[0082] }
[0083] if (cfg_mlp) {
[0084] switch (cfg_activation_function) {
[0085] case 1:
[0086] e = (t>=0.0f) ?t : 0.0f; // 折线函数
[0087] break;
[0088] case 2: // Sigmoid函数
```

```
[0089]  e = sigmoid(t);
[0090]  break;
[0091]  case 3: // tanh函数(正切双曲线函数)
[0092]  e = tanh(t);
[0093]  break;
[0094]  default: // 线性函数
[0095]  e = t;
[0096]  }
[0097]  y[j] = e;
[0098]  }
[0099]  else { // 对于高斯过程模型/RBF模型
[0100]  e = exp(-t);
[0101]  y[0] += p3[j] * e;
[0102]  }
[0103]  /* 输出变换*/
[0104]  j = (cfg_mlp)?p6 : 1;
[0105]  for (k=0; k<j; k++) {
[0106]  z[k] = y[k] * p4[k] + p5[k];
[0107]  }
```

[0108] 识别出:在执行循环函数时,通过变量cfg_mlp可以执行情况区分。在cfg_mlp=1的情况下选择对神经元层的计算,并且可以利用cfg_activation_function=0…3选择上面所描述的激活函数的类型。

[0109] 在cfg_mlp=0时,替代MLP感知器模型,计算高斯过程模型或者RBF模型。这里,不需要选择激活函数,因为始终利用指数函数来计算所述激活函数。以这种方式可能的是,不仅为了计算高斯过程模型和RBF模型而且也为了计算MLP模型的神经元层来使用模型计算单元22,并且在此以状态机的集成结构方式需要仅仅很小的面积需求。

[0110] 对于计算高斯过程模型、RBF模型或诸如此类需要输入变换和输出变换,而这对于计算感知器模型的神经元层来说不是强制性的。因而,替代输入变换和/或输出变换,进行对输入参量和输出参量的简单的复制操作可以是足够的。此外,如果输入参量和输出参量已经处在为此设置的存储区内,可以省去复制操作。

[0111] 因而,对于输入变换来说,借助于输入变换说明cfg_skip_input_scaling和输入参量复制说明cfg_copy_input来进行控制,其中所述输入变换说明cfg_skip_input_scaling说明是否应该进行输入变换,所述输入参量复制说明cfg_copy_input说明所提供的输入参量是否应该被复制到另一存储区内或者所提供的输入参量是否应该以变换方式在所述另一存储区内被提供。

[0112] 用于输入变换的伪代码如下:

```
[0113]  /* 输入变换*/
[0114]  if (!cfg_skip_input_scaling) {
[0115]  for (k=0; k<p7; k++) {
```

```
[0116]  if (cfg_copy_input) {
[0117]  ut[k] = u[k];
[0118]  }
[0119]  else {
[0120]  // 输入变换
[0121]  ut[k] = u[k]*p1[k] + p2[k];
[0122]  }
[0123]  }
[0124]  }
```

[0125] 相应地,对于输出变换来说,借助于输出变换说明cfg_skip_output_scaling和输出参量复制说明cfg_copy_output来进行控制,其中所述输出变换说明cfg_skip_output_scaling说明是否应该进行输出变换,所述输出参量复制说明cfg_copy_output说明所提供的输出参量是否应该被复制到另一存储区内或者所提供的输出参量是否应该以变换方式在所述另一存储区内被提供。

[0126] 用于输出变换的伪代码如下:

```
[0127]  /* 输出变换*/
[0128]  if (!cfg_skip_output_scaling) {
[0129]  for (k=0; k<j; k++) {
[0130]  if (cfg_copy_output) {
[0131]  z[k] = y[k]; /* 到另一存储区内的复制操作 */
[0132]  }
[0133]  else {
[0134]  // 输出变换
[0135]  z[k] = y[k] * p4[k] + p5[k];
[0136]  }
[0137]  }
```

[0138] 由此,整体上选择性地可能的是,与输入变换说明cfg_skip_input_scaling、输入参量复制说明cfg_copy_input、输出变换说明cfg_skip_output_scaling和输出参量复制说明cfg_copy_output对应地来执行输入变换和/或输出变换和/或输入参量到另一存储区内的复制过程以及输出参量到另一存储区内的复制过程。由此,尤其是在计算感知器模型的神经元层的情况下可以跳过输入变换和/或输出变换,使得计算可以整体上被加速。

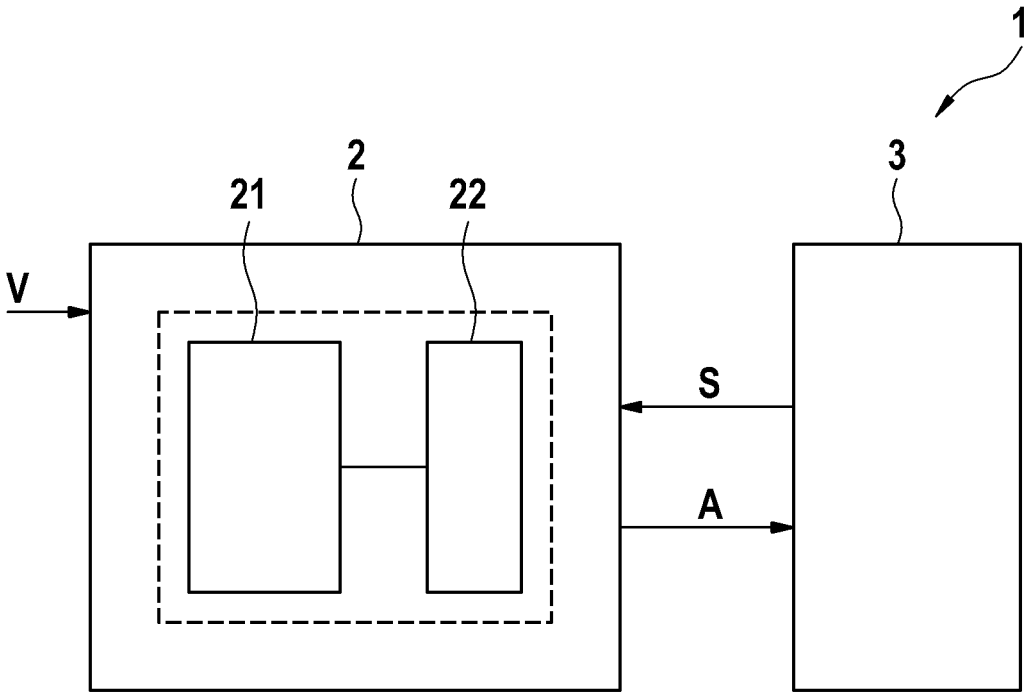


图 1

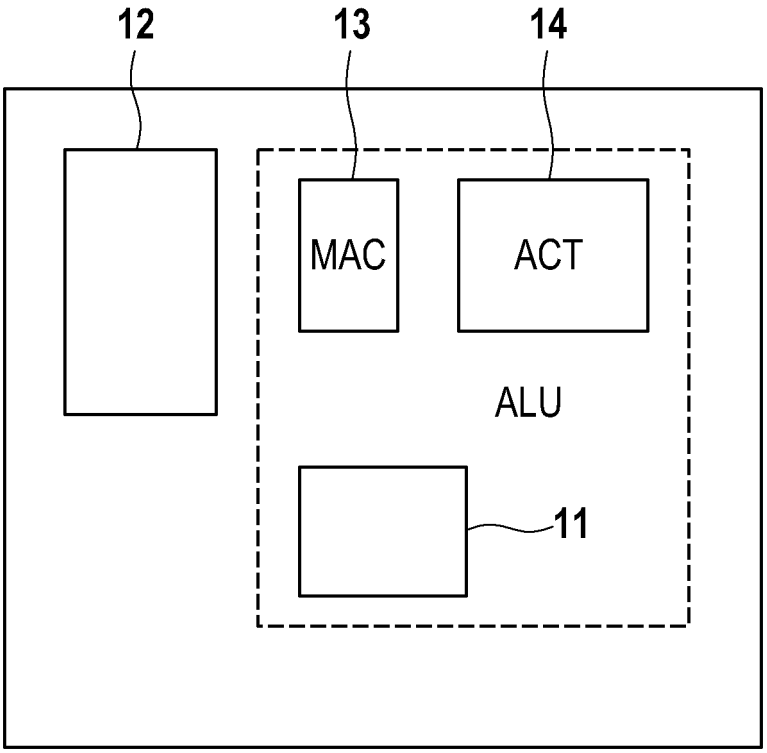


图 2

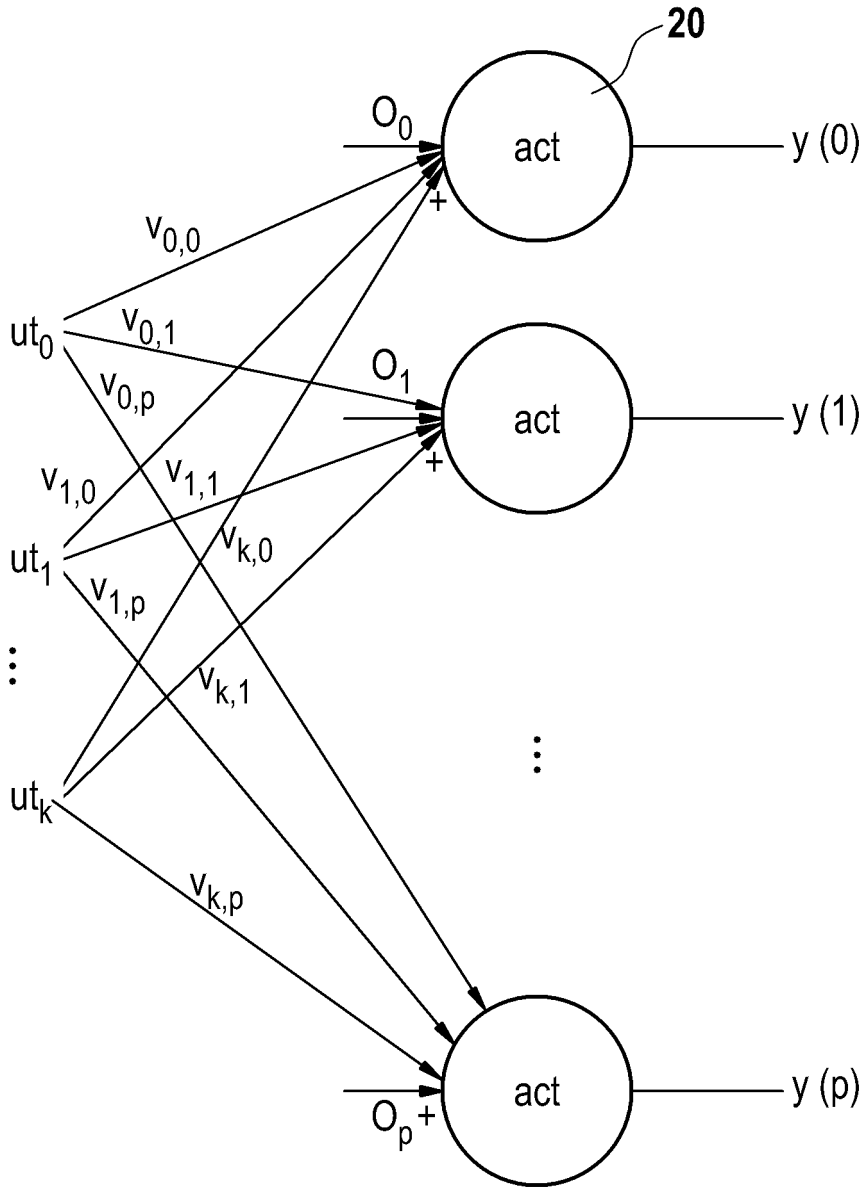

20
act
act
act
O_0
O_1
O_p
y (0)
y (1)
y (p)
ut_0
ut_1
ut_k
$v_{0,0}$
$v_{0,1}$
$v_{0,p}$
$v_{1,0}$
$v_{1,1}$
$v_{1,p}$
$v_{k,0}$
$v_{k,1}$
$v_{k,p}$
+
+
+

图 3

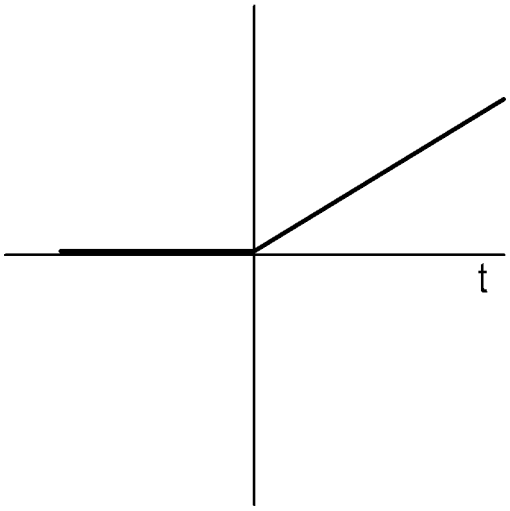


图 4A

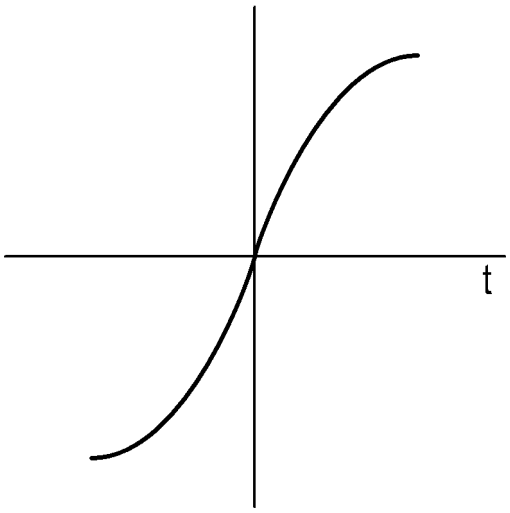


图 4B

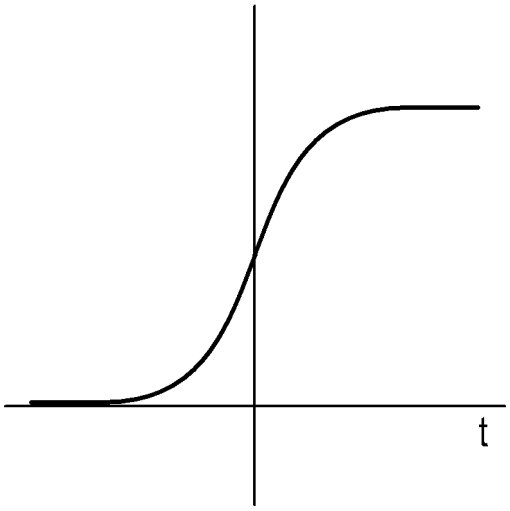


图 4C

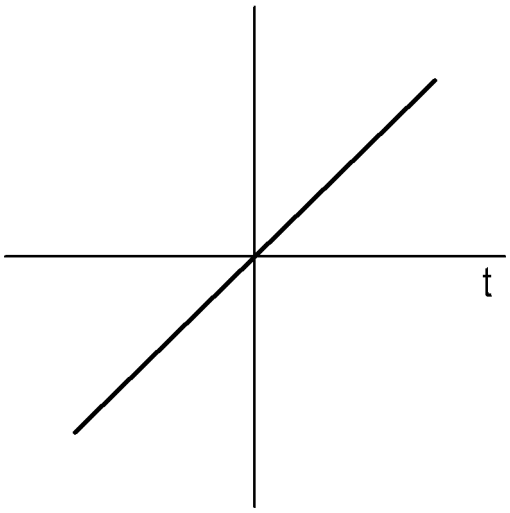


图 4D