CSLT TECHNICAL REPORT-20160017 [2017 年 1 月 6 日]

How to Config Kaldi nnet3 (in Chinese)

Zhiyuan Tang[1]
and Dong Wang[1*]

*Correspondence:
wangd99@mails.tsinghua.edu.cn,
[1]Center for Speech and Language Technologies, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
Full list of author information is available at the end of the article

Abstract

Kaldi nnet3 的神经网络结构是通过读取包含相关组件信息的配置文件生成的。配置文件由实验人员直接编写，它描述了网络中各个组件的拓扑关系。本文梳理了 Kaldi 中不同组件的使用方法，为配置不同的神经网络结构作参考，并展示了如何按照规定语法编写常见神经网络的配置文件。

本文结构如下：第一部分简介 nnet3 配置文件的语法规则，第二部分描述了 Kaldi 中不同组件的使用方法，第三部分列举了几种典型神经网络的配置方法。

1 配置文件的语法规则

Kaldi nnet3 借鉴了计算网络（Computational Network）的思想，即将网络看成是由多个不同计算单元或步骤按特定顺序连接起来的图，并对该计算图进行编译和执行。Kaldi nnet3 的配置文件则是对图的构造进行详细的描述，用于网络的初始化，比如含有一层隐含层（激活函数为 Rectifier）的前向神经网络可以描述为：

```
# First the components
component name=affine1 type=AffineComponent input-dim=30 output-dim=1000
component name=relu1 type=RectifiedLinearComponent dim=1000
component name=affine2 type=AffineComponent input-dim=1000 output-dim=800
component name=logsoftmax type=LogSoftmaxComponent dim=800
# Next the nodes
input-node name=input dim=10
component-node name=affine1_node component=affine1 input=Append(Offset(input, -2), Offset(input, 0), Offset(input, 1))
component-node name=nonlin1 component=relu1 input=affine1_node
component-node name=affine2 component=affine2 input=nonlin1
```

```
component-node name=output_nonlin component=logsoftmax input=affine2
output-node name=output input=output_nonlin objective=quadratic
```

1.1 component 和 component-node

上例中可以看出，配置文件分为两大部分：component 和 component-node，它们之间的关系可以类比于“类的实例化”。component 可以看作是类，它定义了网络中所有情形的组件，每个类（component）都有独一无二的名称，属性的设置相互不受干扰，属性包括类型（比如全连接，不同的激活函数，softmax等），输入输出的维度，以及该类的特性。component-node 则是对 component 的实例化，在对它进行定义时，它所属的类由后面的 component 选项给出，并且需要明确它的输入（另一个 component-node）。比如上例中，component-node affine2 实例化了 component affine2，并且指定了它的输入为 component-node nonlin1， nonli1 的维度与 affine2 所要求的输入维度是相同的。

input-node 和 output-node 是特殊的标识符，不需要指定类（component），它们分别代表神经网络的输入和输出节点。output-node 中可以通过 objective 选项指定不同的损失函数，如 linear（交叉熵，不指定 objective 选项时的默认设置）、quadratic （均方误差）。还有一个不需要类的特殊标识符 dim-range-node，它用于截取一个组件的部分输入，比如我们可以通过以下语句截取 node1 传入值（默认为向量）的前100维，并赋值给 node2：

```
dim-range-node name=node2 input-node=node1 dim-offset=0 dim=100
```

一个 component 可以对应多个 component-node，此时，这些 component-node 也将共享相同的参数。component 可以不被实例化，component-node 必须要有定义了的 component。

1.2 属性与 Descriptor

对于 component 中的属性，可以参考源文件 nnet3/nnet-simple-component.h 和 nnet3/nnet-general-component.h，其中的方法 InitFromConfig 表示了网络在初始化过程中需要从配置文件中读取的信息，常见的属性有 type（直接使用类名，如激活函数 SigmoidComponent、TanhComponent、RectifiedLinearComponent，全连接 AffineComponent、NaturalGradientAffineComponent 等），input-dim 和 output-dim（对于输入输出维度相同的组件来说，直接用 dim 表示二者），还有一些组件有自己特定的属性，比如卷积层 ConvolutionComponent 需要指定滤镜的大小、步长和个数。

上例中出现的其他关键字（Append、Offset）是 nnet3 的 Descriptor，出现于 component-node 的声明中，由源文件 nnet3/nnet-descriptor.h 定义。各个组件相互粘合时，Descriptor 常常用于 input 选项的量化说明，比如上例中 affine1_node 的输入是由3个不同时步（time-step）的原始输入（input-node）拼接（Append）而成的，而 Offset 则表示了某个时步的输入与当前时步的相对时间偏移。对于语音识别任务来说，affine1_node 的输入是由当前帧的上上帧、当前帧以及当前帧的下一帧拼接而成，共有 $10 * 3 = 30$ 维。

Descriptor 的使用方法为： **input=[Descriptor]**, 现将 nnet3 中提供的 Descriptor 总结如下，Descriptor 是可以嵌套使用的，比如 node1 也可以是另一个 Descriptor：

Descriptor	功能
node1	最基本形式，没有修饰语，node1 的值直接传入。
Append(node1,node2)	将 node1、node2 传入的值拼接起来，结果的维度是二者维度之和。
Offset(node1，value)	node1 传过来的值是相对于当前时标偏离 value（可正可负）时步时的结果。
Sum(node1,node2)	将 node1、node2 传入的值加起来，node1、node2 以及最终结果这三者的维度相同。
Failover(node1,node2)	如果 node1 是不可计算的，则使用 node2。
IfDefined(node1)	如果 node1 目前还没有计算出来，先以 “0” 值代替。
Switch(node1,node2,...)	按顺序，每个时步使用其中一个 node，该 node 的序号是当前时步对 node 总数取模的结果。
Round(node1,modulus)	每个时步，将 node1 的时标（time-index）设置为小于当前时步且又是 modulus 的最大倍数。
ReplaceIndex(node1,value)	每个时步，node1 的时标（time-index）保持不变，设为 value。

最后，nnet3 通过以上语法规则将不同的组件组织起来并初始化，在设计网络结构时，可以通过 nnet3-init 命令校验配置文件能否初始化成功。

2 不同组件的使用方法

Kaldi nnet3 实现了两大类组件：简单组件和通用组件，分别对应源文件 nnet3/nnet-simple-component.h 和 nnet3/nnet-general-component.h。接下来将逐一展示这些组件的使用方法，这是对上一章配置文件中 component 的类型和属性的拓展，本章的示例也将以 component 的定义为主，component-node 的实例化不在赘述。

不同组件的属性各有不同，但定义 component 时都必须指定输入输出的维度。维度的赋值可以通过以下两种方式：(1) 显式赋值。这是大多数组件使用的方式。如果组件的输入输出维度不一定相同，则该组件使用 input-dim 和 ouput-dim 进行赋值；如果二者维度必须保持一致，则该组件使用 dim 进行统一赋值。(2) 隐式赋值。部分组件的属性可以载入预先定义好的矩阵或向量，通过该矩阵或向量可以“数”出输入输出的维度，这样则不需要显式地指明输入输出维度。下文将省去

input-dim、output-dim 和 dim 这3个属性的相关使用说明，只对各个组件的特有属性进行描述。

本文将 Kaldi 目前实现的组件分为以下几类，并分别进行讲解：

(1) **常规运算组件**。这类组件除了 input-dim、output-dim 或 dim，没有其他特有属性，语法格式较为一致，没有参数需要更新。常用的激活函数属于此类。

(2) **连接层组件**。这类组件对应网络中的权重部分，参数需要更新。此类组件之间的差别在于不同的连接方式或更新方式。

(3) **一对一运算组件**。这类组件对输入的向量进行一对一的乘或加运算，参数需要更新。

(4) **固定参数组件**。这类组件的部分或全部参数是固定的，固定部分不可更新。

(5) **增强组件**。这类组件的存在是为了增强网络的鲁棒性和训练的稳定性、收敛性。

(6) **卷积组件**。设计卷积神经网络所需组件。

(7) **位置运算组件**。这类组件在进行计算时，考虑了输入（向量）中各元素的位置。

(8) **整合组件**。这类组件可以整合多个其他组件或将复杂功能整合为一个特定的组件。

(9) **通用组件**。有别于上面所有的组件，这类组件的计算是基于 Kaldi 定义的三元组 Index(index,time,extra index) 的，使用也较为复杂。

下文的解说中，x 表示输入向量 X 的单个元素，y 表示输出向量 Y 的单个元素。

2.1 常规运算组件

各个常规运算组件的功能描述：

Component	功能
PnormComponent	p-norm 激活函数，输入维度是输出维度的整数倍，Kaldi 目前使用 2-norm，即 $y_j = \sqrt{\sum_{i=1}^{n} x_{ij}^2}, n = input\text{-}dim/output\text{-}dim$ (X 分为 $output\text{-}dim$ 组，每组 n 个元素，x_{ij} 表示第 j 组的第 i 个元素)。
SigmoidComponent	sigmoid 激活函数。
TanhComponent	tanh 激活函数。
RectifiedLinearComponent	rectifier 激活函数。
SoftmaxComponent	softmax 运算。
LogSoftmaxComponent	softmax 运算之后再进行log 运算。
SumReduceComponent	输入分块后求和，即先将 X 分为 $n = input\text{-}dim/output\text{-}dim$ 块，每块相同位置的元素相加构成输出的一个元素，表示为 $y_j = x_j + x_{l+j} + x_{2l+j} + ...$，$l$ 表示块的长度，等于 $output\text{-}dim$。
ElementwiseProductComponent	对折相乘，即将 X 截为两半，两部分相同位置的元素一对一相乘，输入维度是输出维度的2倍。
NoOpComponent	不做具体运算，只是简单的传值，输入与输出一样。

2.2 连接层组件

各个连接层组件的功能描述：

Component	功能
AffineComponent	常规的全连接，以下几种都是由它衍生而来。
NaturalGradientAffine-Component	与 AffineComponent 的唯一区别在于它用 Natural Gradient Descent (NGD) [1] 的方法更新权值。
RepeatedAffineComponent	X 均分为 n 个块 X_1、X_2 ...，Y 也均分为相同数目的块 Y_1、Y_2 ...，一一对应起来并全连接，即 X_1 与 Y_1 全连接， X_2 与 Y_2 全连接...，这些全连接共享一个权值矩阵。
NaturalGradient-RepeatedAffineComponent	与RepeatedAffineComponent的区别在于它用 NGD 的方法更新权值。
BlockAffineComponent	权值矩阵对角线由数个维度相等的矩阵块对角依次相连并铺满，所以矩阵块数目可以整除输入输出维度。这些矩阵块的权值可更新，其他地方的权值设为0 且不更新。

对于连接层、一对一运算等参数可更新组件，学习率都可以通过以下属性进行设置：

参数更新所需属性	功能
learning-rate	学习率。
learning-rate-factor	学习率的额外系数。
max-change	对参数的变化进行控制，类似 L2-norm，设为正数时有效。

连接层组件都涉及权值的初始化，它们是有公有属性的：

连接层组件公有属性	功能
matrix	外部传入矩阵对全连接参数矩阵进行初始化，部分组件未实现。
param-stddev	随机初始化全连接参数时的参数（标准差）。
bias-stddev	随机初始化偏置（bias）参数时的参数（标准差）。
bias-mean	随机初始化偏置（bias）参数时的参数（均值），部分组件未添加。

各个连接层组件特有的属性分别列举如下：

NaturalGradientAffine-Component 特有属性	功能
num-samples-history alpha rank-in, rank-out update-period	NGD 相关配置参数。 可以参考源文件 nnet-precondition-online.h， 使用默认值即可。
max-change-per-sample	对一个 minibatch 下所有参数的变化之和进行控制，类似 L2-norm，设为正数时有效。已废弃。使用 max-change 替代。

(NaturalGradient)Repeated-AffineComponent 特有属性	功能
num-repeats	局部全连接的个数，可以整除输入输出维度。

BlockAffineComponent 特有属性	**功能**
num-blocks	对角线上子矩阵块的个数，可以整除输入输出维度。

2.3 一对一运算组件

这类组件的参数是一个与输入同维度的向量，并与输入在元素级别上进行一对一的乘或加运算：

Component	**功能**
PerElementScaleComponent	输入的每个元素都有一个参数，并与之相乘。
NaturalGradientPerElement-ScaleComponent	运算同 PerElementScaleComponent，使用 NGD 更新参数。
PerElementOffsetComponent	输入的每个元素都有一个参数，并与之相加。

NaturalGradientPerElementScaleComponent 的属性中有一部分是 NGD 的配置，参考 NaturalGradientAffineComponent，3个组件的其他属性均是关于参数的初始化：

一对一运算公有属性	**功能**
vector/scales	外部传入的向量初始化参数。
param-mean param-stddev	随机初始化参数的配置（均值）。 随机初始化参数的配置（标准差）。

2.4 固定参数组件

各个固定参数组件的功能描述：

Component	**功能**
FixedAffineComponent	通过 matrix 传入一个固定的全连接矩阵。
FixedScaleComponent	通过 scales 传入一个固定的向量与输入元素上一对一相乘。
FixedBiasComponent	通过 bias 传入一个固定的向量与输入元素上一对一相加。
ConstantFunctionComponent	输出与输入无关，等价于 AffineComponent 的全连接参数固定为0，只设置偏置（bias）。

ConstantFunctionComponent 的参数可通过以下属性进一步设置：

ConstantFunctionComponent 特有参数	**功能**
is-updatable	是否可更新。
use-natural-gradient	是否使用 NGD 方法更新。
output-mean	随机初始化参数的配置（均值）。
output-stddev	随机初始化参数的配置（标准差）。

2.5 增强组件

各个增强组件的功能描述：

Component	功能
DropoutComponent	将输入的元素以一定概率（dropout-proportion 属性提供）设置为0。
NormalizeComponent	将输入标准化且均方根为 target-rms（默认为1.0），add-log-stddev (默认false) 表示是否输出 log 计算后的标准差作为额外的输出。
ClipGradientComponent	防止反向传播时梯度爆炸，对过大的梯度值进行剪切。

ClipGradientComponent 的相关参数：

ClipGradientComponent 特有属性	功能
clipping-threshold	对梯度进行剪切的阈值，处理后梯度的绝对值都不超过该阈值。
norm-based-clipping	是否以标准化的方式对整个输入进行相同比例的缩小。
self-repair-clipped-proportion-threshold	如果剪切的数目超过一定百分比，则进行自我修复。
self-repair-target	修复后的预期值。
self-repair-scale	修复时进行缩放。

2.6 卷积组件

卷积组件只有两个：

Component	功能
ConvolutionComponent	进行卷积操作，Kaldi 中只实现了2维卷积。
MaxpoolingComponent	下采样操作。

ConvolutionComponent 属性	功能
input-x-dim	输入的维度，比如 x 代表时域，
input-y-dim	y 代表频域，
input-z-dim	z 代表信道。
input-vectorization-order	输入的拼接顺序，zyx 或者 yzx。
num-filters	2维 filter 的个数。
filt-x-dim, filt-y-dim	2维 filter 的大小。
filt-x-step, filt-y-step	2维 filter 的步长。
matrix	外部传入矩阵对权值进行初始化。
param-stddev, bias-stddev	随机初始化权值和偏置所需的配置（标准差）。

MaxpoolingComponent 属性	功能
input-x-dim	输入的维度，比如 x 代表时域，
input-y-dim	y 代表频域，
input-z-dim	z 代表 filter 的个数，输入拼接方式为 zyx。
pool-x-dim, pool-y-dim, pool-z-dim	3维下采样的大小。
pool-x-step, pool-y-step, pool-z-step	3维下采样的步长。

2.7 位置运算组件

位置运算组件的功能描述：

Component	功能
SumGroupComponent	将输入分组相加，通过 sizes 属性设置相加的范围，如 $sizes = [3, 2], input\text{-}dim = 3 + 2, output\text{-}dim = 2, y_1 = x_1 + x_2 + x_3, y_2 = x_4 + x_5$ 。
PermuteComponent	将输入中各元素按 column-map 属性指定的新位置重新排列作为输出。

2.8 整合组件

整合组件的功能描述：

Component	功能
LstmNonlinearityComponent	论文 [2] 中的 LSTM 作为一个组件，通过 cell-dim 设置细胞数，且有自我修复属性，用法参见ClipGradientComponent。
CompositeComponent	整合多个组件为一个组件。将多个组件的定义放在一起作为一个参数。

CompositeComponent 属性	功能
num-components	将要整合的组件总数。
max-rows-process	能同时处理的最大输入个数。因该组件较为庞大，做此限制可防止消耗过大内存。

2.9 通用组件

通用组件的功能描述：

Component	功能
DistributeComponent	将输入均分为几部分，对应多个输出。
StatisticsExtractionComponent	收集输入的统计信息。
StatisticsPoolingComponent	与 StatisticsExtractionComponent 合用，在时序上累积统计信息。
BackpropTruncationComponent	对梯度进行截取，防止梯度爆炸，有类似ClipGradientComponent的功能，常用于RNN。

通用组件的更多细节将在实用过程中进一步说明。

3 几种网络的配置方法

本章针对不同的神经网路结构，先说明了其配置文件的要点。

CNN 的配置需要使用卷积组件（ConvolutionComponent、MaxpoolingComponent）。

Residual Network 需要额外引入一个 node 用于残差加法运算。

Siamese Network 可以使用 (NaturalGradient)RepeatedAffineComponent，或者用同一个 component 实例化两个 component-node。

Multi-task 需要至少两个output-node。

RNN、LSTM、GRU的配置需要用到 Offset 这个 Descriptor，用于产生延时。

具体的配置实例后续将添加。

Author details

[1]Center for Speech and Language Technologies, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [2]Center for Speech and Language Technologies, Division of Technical Innovation and Development, Tsinghua National Laboratory for Information Science and Technology, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [3]Chengdu Institute of Computer Applications, Chinese Academy of Sciences, 610041 Chengdu, China.

References

1. Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur, "Parallel training of dnns with natural gradient and parameter averaging," *arXiv preprint arXiv:1410.7455*, 2014.
2. Hasim Sak, Andrew W Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling.," in *INTERSPEECH*, 2014, pp. 338–342.