SoftMax 层

初始示例代码

```python
import numpy as np
from cuda import cudart
import tensorrt as trt

nIn, cIn, hIn, wIn = 1, 3, 3, 3  # 输入张量 NCHW
data = np.arange(1, 1 + cIn * hIn * wIn, dtype=np.float32).reshape(nIn, cIn, hIn, wIn)  # 输入数据

np.set_printoptions(precision=8, linewidth=200, suppress=True)
cudart.cudaDeviceSynchronize()

logger = trt.Logger(trt.Logger.ERROR)
builder = trt.Builder(logger)
network = builder.create_network(1 << int(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH))
config = builder.create_builder_config()
inputT0 = network.add_input('inputT0', trt.DataType.FLOAT, (nIn, cIn, hIn, wIn))
#---------------------------------------------------------- --------------------# 替换部分
softMaxLayer = network.add_softmax(inputT0)
#---------------------------------------------------------- --------------------# 替换部分
network.mark_output(softMaxLayer.get_output(0))
engineString = builder.build_serialized_network(network, config)
engine = trt.Runtime(logger).deserialize_cuda_engine(engineString)
context = engine.create_execution_context()
_, stream = cudart.cudaStreamCreate()

inputH0 = np.ascontiguousarray(data.reshape(-1))
outputH0 = np.empty(context.get_binding_shape(1), dtype=trt.nptype(engine.get_binding_dtype(1)))
_, inputD0 = cudart.cudaMallocAsync(inputH0.nbytes, stream)
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)

cudart.cudaMemcpyAsync(inputD0, inputH0.ctypes.data, inputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
context.execute_async_v2([int(inputD0), int(outputD0)], stream)
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)
cudart.cudaStreamSynchronize(stream)

print("inputH0 :", data.shape)
print(data)
print("outputH0:", outputH0.shape)
print(outputH0)

cudart.cudaStreamDestroy(stream)
cudart.cudaFree(inputD0)
cudart.cudaFree(outputD0)
```

- 输入张量形状 (1,3,3,3)

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}1. & 2. & 3.\\4. & 5. & 6.\\7. & 8. & 9.\end{bmatrix}\begin{bmatrix}10. & 11. & 12.\\13. & 14. & 15.\\16. & 17. & 18.\end{bmatrix}\begin{bmatrix}19. & 20. & 21.\\22. & 23. & 24.\\25. & 26. & 27.\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 输出张量形状 (3, 3, 3)，默认在“非 batch 维的最高维”上计算 softmax，各通道相同 HW 位置上元素之和为 1

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}0.00000002 & 0.00000002 & 0.00000002\\0.00000002 & 0.00000002 & 0.00000002\\0.00000002 & 0.00000002 & 0.00000002\end{bmatrix}\begin{bmatrix}0.00012339 & 0.00012339 & 0.00012339\\0.00012339 & 0.00012339 & 0.00012339\\0.00012339 & 0.00012339 & 0.00012339\end{bmatrix}\begin{bmatrix}0.9998766 & 0.9998766 & 0.9998766\\0.9998766 & 0.9998766 & 0.9998766\\0.9998766 & 0.9998766 & 0.9998766\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 计算过程：

$$\frac{\left[e^1, e^{10}, e^{19}\right]}{e^1+e^{10}+e^{19}} = \left[1.523\times 10^{-8}, 1.234\times 10^{-4}, 9.998\times 10^{-1}\right]$$

---

axes

```
axesIndex = 0
softMaxLayer = network.add_softmax(inputT0)
softMaxLayer.axes = 1 << axesIndex  # 运算的轴号，默认值 1<<1
```

- 指定 axes=1<<0（在最高维上计算 softmax），输出张量形状 (1,3,3,3)，只有一个元素参与，结果全都是 1

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}1. & 1. & 1.\\1. & 1. & 1.\\1. & 1. & 1.\end{bmatrix}\begin{bmatrix}1. & 1. & 1.\\1. & 1. & 1.\\1. & 1. & 1.\end{bmatrix}\begin{bmatrix}1. & 1. & 1.\\1. & 1. & 1.\\1. & 1. & 1.\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 指定 axes=1<<1（在次高维上计算 softmax），输出张量形状 (1,3,3,3)，结果与初始示例代码相同
- 指定 axes=1<<2（在季高维上计算 softmax），输出张量形状 (1,3,3,3)，每个通道内同一列元素之和为 1

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}0.00235563 & 0.00235563 & 0.00235563\\0.04731416 & 0.04731416 & 0.04731416\\0.95033026 & 0.95033026 & 0.95033026\end{bmatrix}\begin{bmatrix}0.00235563 & 0.00235563 & 0.00235563\\0.04731416 & 0.04731416 & 0.04731416\\0.95033026 & 0.95033026 & 0.95033026\end{bmatrix}\begin{bmatrix}0.00235563 & 0.00235563 & 0.00235563\\0.04731416 & 0.04731416 & 0.04731416\\0.95033026 & 0.95033026 & 0.95033026\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 指定 axes=1<<3（在叔高维上计算 softmax），输出张量形状 (1,3,3,3)，每个通道内同一行元素之和为 1

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}0.09003057 & 0.24472848 & 0.66524094\\0.09003057 & 0.24472848 & 0.66524094\\0.09003057 & 0.24472848 & 0.66524094\end{bmatrix}\begin{bmatrix}0.09003057 & 0.24472848 & 0.66524094\\0.09003057 & 0.24472848 & 0.66524094\\0.09003057 & 0.24472848 & 0.66524094\end{bmatrix}\begin{bmatrix}0.09003057 & 0.24472848 & 0.66524094\\0.09003057 & 0.24472848 & 0.66524094\\0.09003057 & 0.24472848 & 0.66524094\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 不能同时指定两个及以上的 axes，如使用 axes=(1<<0)+(1<<1)，会收到报错

```
[TRT] [E] 3: [layers.h::setAxes::600] Error Code 3: API Usage Error (Parameter check failed at:
/_src/build/cuda-11.4/8.2/x86_64/release/optimizer/api/layers.h::setAxes::600, condition:
isSingleBit(axes)
)
```