Python\_API\_Guides\_Neural\_Network

目录

[神经网络(Neural Network) 1](#_Toc497234208)

[激活函数（Activation Functions） 1](#_Toc497234209)

[卷积函数（Convolution） 2](#_Toc497234210)

[池化函数（Pooling） 2](#_Toc497234211)

[数据标准化（Normalization） 3](#_Toc497234212)

[损失函数（Losses） 4](#_Toc497234213)

[分类函数（Classification） 4](#_Toc497234214)

[符号嵌入（Embeddings） 5](#_Toc497234215)

[循环神经网络（Recurrent Neural Networks） 5](#_Toc497234216)

[求值网络（Evaluation） 6](#_Toc497234217)

[监督候选采样网络（Candidate Sampling） 7](#_Toc497234218)

# 神经网络(Neural Network)

## 激活函数（Activation Functions）

| **操作** | **描述** |
| --- | --- |
| tf.nn.relu(features, name=None) | 整流函数：max(features, 0) |
| tf.nn.relu6(features, name=None) | 以6为阈值的整流函数：min(max(features, 0), 6) |
| tf.nn.elu(features, name=None) | elu函数，exp(features) - 1 if < 0,否则features |
| tf.nn.softplus(features, name=None) | 计算softplus：log(exp(features) + 1) |
| tf.nn.dropout(x, keep\_prob,  noise\_shape=None, seed=None, name=None) | 计算dropout，keep\_prob为keep概率 noise\_shape为噪声的shape |
| tf.nn.bias\_add(value, bias, data\_format=None, name=None) | 对value加一偏置量 此函数为tf.add的特殊情况，bias仅为一维， 函数通过广播机制进行与value求和, 数据格式可以与value不同，返回为与value相同格式 |
| tf.sigmoid(x, name=None) | y = 1 / (1 + exp(-x)) |
| tf.tanh(x, name=None) | 双曲线切线激活函数 |

## 卷积函数（Convolution）

| **操作** | **描述** |
| --- | --- |
| tf.nn.conv2d(input, filter, strides, padding,  use\_cudnn\_on\_gpu=None, data\_format=None, name=None) | 在给定的4D input与 filter下计算2D卷积 输入shape为 [batch, height, width, in\_channels] |
| tf.nn.conv3d(input, filter, strides, padding, name=None) | 在给定的5D input与 filter下计算3D卷积 输入shape为[batch, in\_depth, in\_height, in\_width, in\_channels] |

## 池化函数（Pooling）

| **操作** | **描述** |
| --- | --- |
| tf.nn.avg\_pool(value, ksize, strides, padding,  data\_format=’NHWC’, name=None) | 平均方式池化 |
| tf.nn.max\_pool(value, ksize, strides, padding,  data\_format=’NHWC’, name=None) | 最大值方法池化 |
| tf.nn.max\_pool\_with\_argmax(input, ksize, strides, padding, Targmax=None, name=None) | 返回一个二维元组(output,argmax),最大值pooling，返回最大值及其相应的索引 |
| tf.nn.avg\_pool3d(input, ksize, strides,  padding, name=None) | 3D平均值pooling |
| tf.nn.max\_pool3d(input, ksize, strides,  padding, name=None) | 3D最大值pooling |

## 数据标准化（Normalization）

| **操作** | **描述** |
| --- | --- |
| tf.nn.l2\_normalize(x, dim, epsilon=1e-12, name=None) | 对维度dim进行L2范式标准化 output = x / sqrt(max(sum(x\*\*2), epsilon)) |
| tf.nn.sufficient\_statistics(x, axes, shift=None,  keep\_dims=False, name=None) | 计算与均值和方差有关的完全统计量 返回4维元组,\*元素个数，\*元素总和，\*元素的平方和，\*shift结果 |
| tf.nn.normalize\_moments(counts, mean\_ss, variance\_ss, shift, name=None) | 基于完全统计量计算均值和方差 |
| tf.nn.moments(x, axes, shift=None,  name=None, keep\_dims=False) | 直接计算均值与方差 |

## 损失函数（Losses）

| **操作** | **描述** |
| --- | --- |
| tf.nn.l2\_loss(t, name=None) | output = sum(t \*\* 2) / 2 |

## 分类函数（Classification）

| **操作** | **描述** |
| --- | --- |
| tf.nn.sigmoid\_cross\_entropy\_with\_logits (logits, targets, name=None)\* | 计算输入logits, targets的交叉熵 |
| tf.nn.softmax(logits, name=None) | softmax[i, j] = exp(logits[i, j]) / sum\_j(exp(logits[i, j])) |
| tf.nn.log\_softmax(logits, name=None) | logsoftmax[i, j] = logits[i, j] - log(sum(exp(logits[i]))) |
| tf.nn.softmax\_cross\_entropy\_with\_logits (logits, labels, name=None) | 计算logits和labels的softmax交叉熵 logits, labels必须为相同的shape与数据类型 |
| tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits (logits, labels, name=None) | 计算logits和labels的softmax交叉熵，label为[batch\_size]，每个label为[0, num\_class)的一个数 |
| tf.nn.weighted\_cross\_entropy\_with\_logits (logits, targets, pos\_weight, name=None) | 与sigmoid\_cross\_entropy\_with\_logits()相似， 但给正向样本损失加了权重pos\_weight |

## 符号嵌入（Embeddings）

| **操作** | **描述** |
| --- | --- |
| tf.nn.embedding\_lookup (params, ids, partition\_strategy=’mod’,  name=None, validate\_indices=True) | 根据索引ids查询embedding列表params中的tensor值 如果len(params) > 1，id将会安照partition\_strategy策略进行分割 1、如果partition\_strategy为”mod”， id所分配到的位置为p = id % len(params) 比如有13个ids，分为5个位置，那么分配方案为： [[0, 5, 10], [1, 6, 11], [2, 7, 12], [3, 8], [4, 9]] 2、如果partition\_strategy为”div”,那么分配方案为： [[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10], [11, 12]] |
| tf.nn.embedding\_lookup\_sparse(params,  sp\_ids, sp\_weights, partition\_strategy=’mod’,  name=None, combiner=’mean’) | 对给定的ids和权重查询embedding 1、sp\_ids为一个N x M的稀疏tensor， N为batch大小，M为任意，数据类型int64 2、sp\_weights的shape与sp\_ids的稀疏tensor权重， 浮点类型，若为None，则权重为全’1’ |

## 循环神经网络（Recurrent Neural Networks）

| **操作** | **描述** |
| --- | --- |
| tf.nn.rnn(cell, inputs, initial\_state=None, dtype=None,  sequence\_length=None, scope=None) | 基于RNNCell类的实例cell建立循环神经网络 |
| tf.nn.dynamic\_rnn(cell, inputs, sequence\_length=None,  initial\_state=None, dtype=None, parallel\_iterations=None,  swap\_memory=False, time\_major=False, scope=None) | 基于RNNCell类的实例cell建立动态循环神经网络 与一般rnn不同的是，该函数会根据输入动态展开 返回(outputs,state) |
| tf.nn.state\_saving\_rnn(cell, inputs, state\_saver, state\_name,  sequence\_length=None, scope=None) | 可储存调试状态的RNN网络 |
| tf.nn.bidirectional\_rnn(cell\_fw, cell\_bw, inputs,  initial\_state\_fw=None, initial\_state\_bw=None, dtype=None, sequence\_length=None, scope=None) | 双向RNN, 返回一个3元组tuple (outputs, output\_state\_fw, output\_state\_bw) |

## 求值网络（Evaluation）

| **操作** | **描述** |
| --- | --- |
| tf.nn.top\_k(input, k=1, sorted=True, name=None) | 返回前k大的值及其对应的索引 |
| tf.nn.in\_top\_k(predictions, targets, k, name=None) | 返回判断是否targets索引的predictions相应的值 是否在在predictions前k个位置中， 返回数据类型为bool类型，len与predictions同 |

## 监督候选采样网络（Candidate Sampling）

| **操作** | **描述** |
| --- | --- |
| ***Sampled Loss Functions*** |  |
| tf.nn.nce\_loss(weights, biases, inputs, labels, num\_sampled, num\_classes, num\_true=1, sampled\_values=None, remove\_accidental\_hits=False, partition\_strategy=’mod’, name=’nce\_loss’) | 返回noise-contrastive的训练损失结果 |
| tf.nn.sampled\_softmax\_loss(weights, biases, inputs, labels,  num\_sampled, num\_classes, num\_true=1, sampled\_values=None, remove\_accidental\_hits=True, partition\_strategy=’mod’,  name=’sampled\_softmax\_loss’) | 返回sampled softmax的训练损失 |
| ***Candidate Samplers*** |  |
| tf.nn.uniform\_candidate\_sampler(true\_classes, num\_true,  num\_sampled, unique, range\_max, seed=None, name=None) | 通过均匀分布的采样集合 返回三元tuple 1、sampled\_candidates 候选集合。 2、期望的true\_classes个数，为浮点值 3、期望的sampled\_candidates个数，为浮点值 |
| tf.nn.log\_uniform\_candidate\_sampler(true\_classes, num\_true, num\_sampled, unique, range\_max, seed=None, name=None) | 通过log均匀分布的采样集合，返回三元tuple |
| tf.nn.learned\_unigram\_candidate\_sampler (true\_classes, num\_true, num\_sampled, unique,  range\_max, seed=None, name=None) | 根据在训练过程中学习到的分布状况进行采样 返回三元tuple |
| tf.nn.fixed\_unigram\_candidate\_sampler(true\_classes, num\_true, num\_sampled, unique, range\_max, vocab\_file=”,  distortion=1.0, num\_reserved\_ids=0, num\_shards=1,  shard=0, unigrams=(), seed=None, name=None) | 基于所提供的基本分布进行采样 |