deeplearning4j梯度标准化策略

deeplearning4j一共提供了如下策略：

```
package org.deeplearning4j.nn.conf;

public enum GradientNormalization {
    None, RenormalizeL2PerLayer, RenormalizeL2PerParamType, ClipElementWiseAbsoluteValue,
ClipL2PerLayer, ClipL2PerParamType
}
```

方法的调用在`package org.deeplearning4j.nn.updater`下的`BaseMultiLayerUpdater`中的

```
/**
 * Pre-apply: Apply gradient normalization/clipping
 *
 * @param layer     Layer to apply gradient normalization/clipping for
 * @param gradient  Gradient to update
 * @param iteration The current iteration (i.e., number of parameter updates so far)
 */
public void preApply(Layer layer, Gradient gradient, int iteration)
```

方法。

RenormalizeL2PerLayer

rescale gradients by dividing by the L2 norm of all gradients for the layer.

1. 计算传入梯度的二范数
2. 梯度除以对应的二范数

```
if (layerGradientView != null) {
    double l2 = layerGradientView.norm2Number().doubleValue();
    layerGradientView.divi(l2);
}
```

RenormalizeL2PerParamType

rescale gradients by dividing by the L2 norm of the gradients, separately for each type of parameter within the layer.

This differs from RenormalizeL2PerLayer in that here, each parameter type (weight, bias etc) is normalized separately.
For example, in a MLP/FeedForward network (where G is the gradient vector), the output is as follows:

GOut_weight = G_weight / l2(G_weight)

GOut_bias = G_bias / l2(G_bias)

对于神经网络每一层的不同参数的梯度，分别除以对应梯度的二范数。以简单的前向传播网络来说，每一层只有两个参数权重（Weight）和偏置（Bias），假设对应的梯度分别为G_weight和GOut_bias，那么对应的标准化方法就为：

GOut_weight = G_weight / l2(G_weight)

GOut_bias = G_bias / l2(G_bias)

```
for (INDArray g : gradient.gradientForVariable().values()) {
    double l2 = Nd4j.getExecutioner().execAndReturn(new Norm2(g)).getFinalResult().doubleVal
ue();
    g.divi(l2);
}
```

ClipElementWiseAbsoluteValue

clip the gradients on a per-element basis.
For each gradient g, set g <- sign(g)max(maxAllowedValue,|g|).
i.e., if a parameter gradient has absolute value greater than the threshold, truncate it.

这个方法是根据设定的阈值对梯度进行裁剪。
如果参数梯度中元素的绝对值大于设定的阈值，则进行裁剪。
例如：
我们设定阈值$threshold = 5$，如果梯度中元素的值在$[-5, 5]$之间则不会发生变化。如果元素的值小于-5则会被改为-5，如果元素的值大于5则会被改为5。
这个理论由Mikolov (2012)提出在所编写的`Statistical Language Models Based on Neural Networks`，链接为http://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf。

Threshold这个值可以使用`gradientNormalizationThreshold(double threshold)`来进行设置。

```
if (layerGradientView != null) {
    BooleanIndexing.replaceWhere(layerGradientView, threshold,
Conditions.greaterThan(threshold));
    BooleanIndexing.replaceWhere(layerGradientView, -threshold,
Conditions.lessThan(-threshold));
}
```

ClipL2PerLayer

conditional renormalization. Somewhat similar to RenormalizeL2PerLayer, this strategy
scales the gradients *if and only if* the L2 norm of the gradients (for entire layer) exceeds a specified
threshold. Specifically, if G is gradient vector for the layer, then:

GOut = G if l2Norm(G) < threshold (i.e., no change)

GOut = threshold*G / l2Norm(G) otherwise

Thus, the l2 norm of the scaled gradients will not exceed the specified threshold, though may be smaller than it
See: Pascanu, Mikolov, Bengio (2012), *On the difficulty of training Recurrent Neural Networks*, http://arxiv.org/abs/1211.5063
Threshold for clipping can be set in Layer configuration, using gradientNormalizationThreshold(double threshold)

这个策略和RenormalizeL2PerLayer类似，但是该策略是个条件策略。
当前仅当梯度的二范数超过制定的阈值的时候才对梯度进行标准化

$$layerL2 = layerGradientView.norm2Number().doubleValue()$$

$$G = \begin{cases} G, & \text{if layerL2} < \text{threshold} \\ \dfrac{threshold * G}{layerL2}, & \text{otherwise} \end{cases}$$

```
if (layerGradientView != null) {
    double layerL2 = layerGradientView.norm2Number().doubleValue();
    if (layerL2 > threshold) {
        double scalingFactor = threshold / layerL2; // g = g / l2 * threshold ->
        layerGradientView.muli(scalingFactor);
    }
}
```

ClipL2PerParamType

conditional renormalization. Very similar to ClipL2PerLayer, however instead of clipping per layer, do clipping on each parameter type separately.
For example in a recurrent neural network, input weight gradients, recurrent weight gradients and bias gradient are all
clipped separately. Thus if one set of gradients are very large, these may be clipped while leaving the other gradients
unmodified.

和ClipL2PerLayer类似，对于每一层中的每个参数分别进行标准化。
例如循环神经网络中每一层含有输入权重梯度，循环权重梯度和偏置梯度。如果其中一组的权重过大，这样就会只更新改组梯度而不会影响到其他参数的梯度。

```
for (INDArray g : gradient.gradientForVariable().values()) {
    double l2 = g.norm2Number().doubleValue();
    if (l2 > threshold) {
        double scalingFactor = l2 / threshold;
        g.divi(scalingFactor);
    }
}
```

更多文档可以查看 https://github.com/sjsdfg/deeplearning4j-issues。
欢迎star