https://www.jianshu.com/p/d2b125c0dd6b

http://www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf

https://en.wikipedia.org/wiki/Dual\_number

https://blog.csdn.net/okcd00/article/details/78294212

https://alexey.radul.name/ideas/2013/introduction-to-automatic-differentiation/

# 自动求导

经常需要使用函数和对应的导数，例如使用牛顿法解非线性等式：

牛顿迭代的方法是：

因此需要在知道的情况下计算。

常见的自动求导方法包括：数值微分、符号微分、前向模式和反向模式。

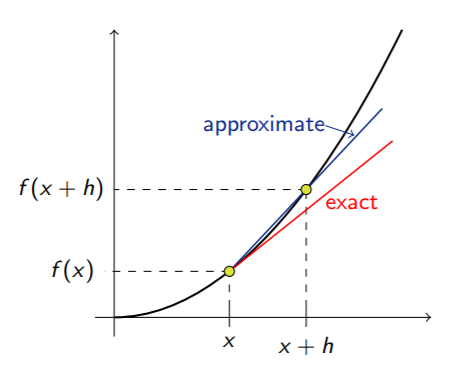
## 手动求解

求解出梯度公式，然后编写代码，代入实际数值得出真实的梯度。每一次修改模型都要修改对应的梯度求解算法。

## 数值微分

最直接且最简单的一种自动求导方式。从**导数的定义**中，可以直观看到：

当h接近0时，导数是可以近似计算出来的。这个方法可以使用几乎所有形式，除非该点不可导。



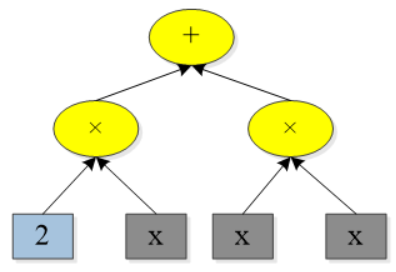
但是有两个问题：**求出的导数可能不准确**，如再极值点附近，如果h选取不当，可能得到符号相反的结果。参数比较多时计算**不够高效**，每计算一个参数的导数，都需要重新计算。

可以用这种不高效但简单的方法做**梯度检查**，检查其他方法得到的梯度是否正确。

## 符号微分

符号微分适合符号表达式的自动求导，原理是基于简单的**求导规则**：

当我们将符号表达式用表达树表示时，可以利用**加法规则**和**乘法规则**进行自动求导。



利用求导规则，可以求出：

符号微分不一定会得到简化的导数，如果表达式树结构比较复杂时，得到的导数表达式会相当复杂，也许会出现**表达式爆炸**的现象。

## 自动微分

自动微分法是**一种介于符号微分和数值微分的方法**：数值微分强调一开始直接代入数值近似求解，符号微分强调直接对式子进行求解，最后才代入数值。

自动微分将符号微分应用于最基本的算子，比如常数、幂函数、指数函数、对数函数、三角函数等，然后代入数值保留中间结果，最后再应用于整个函数。可做到完全向用户隐藏微分求解过程。使用自动微分和不使用自动微分对**代码总体改动非常小**，并且由于它的计算**实际是一种图计算**，可以对其做很多优化。

## 前向模式

前向模式基于**二元数（dual number）**：，其中a和b都是实数，是无穷小量，其无限接近0，但是不等于0，。比如，可以认为是一个接近5的数。

对于二元数满足简单的加法和乘法规则：

另外一个特性是：

这意味着：当b=1时，**只需要计算出，就可以得到和其对应的导数**。所以一个**前向计算过程**可以**同时得到函数值和其导数**。例如，要计算在x=2处的函数值和导数：

**特性的证明**：

对于一个实多项式：

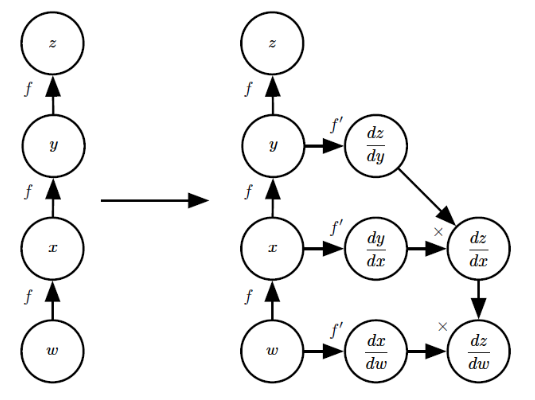
可以从实值扩展为dual numbers：

## 符号到数值的微分（Torch）

计算图和一组用于输入的数值，返回在这些输入值处的梯度。这种方法用在Torch和Caffe中。

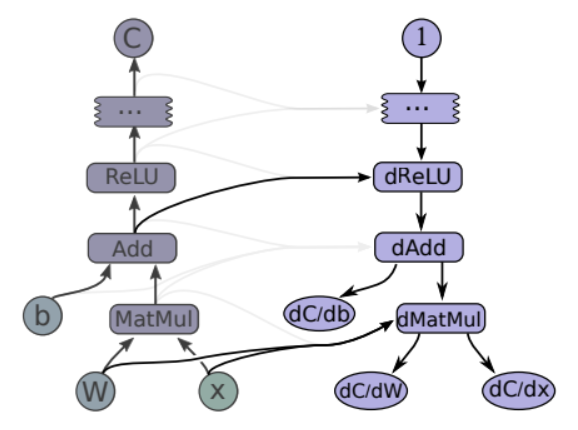
## 符号到符号的微分（Tensorflow）

采用计算图以及添加一些额外的节点到计算图中，这些额外的节点提供了所需导数的符号描述。这是Theano和Tensorflow中采用的方法。这种方法的主要优点是导数可以使用和原始表达式相同的语言来描述：



在这种方法中，反向传播算法不需要访问任何实际的特定数值，它将节点添加到计算图中来描述如何计算这些导数，可以在随后计算任何特定数值的导数。

Tensorflow中实现的自动求导方式是利用**反向传递**与**链式法则**建立一张对应原计算图的梯度图：



微积分中的链式法则用于计算复合函数的导数，反向传播是一种计算链式法则的算法。设是实数，和是从实数映射到实数的函数，假设且，那么链式法则是：

可以对标量情况进行扩展。假设，*g*是从到的映射，*f*是从到的映射。如果且，那么：

通常不仅将反向传播算法用于向量，还要用于任意维度的张量。在反向传播之前，将每个张量变平成为一个向量，计算该向量值的梯度，然后将该梯度重新构造成一个张量。

为了计算某个标量关于图中节点的梯度，首先计算它关于的梯度，然后计算到x路径上其它节点的梯度，以这种方式计算各个节点的梯度直到到达x。对于从z出发，经过两个或更多路径向后到达的任意节点，可以简单地对该节点来自不同路径上的梯度求和。

Tensorflow中每个操作op都有对应的bprop实现。

