自适应滤波器 - Adaptive Filter

Wikipedia 的解释

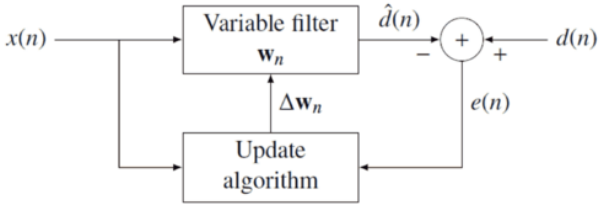
自适应滤波器是能够根据输入信号自动调整性能进行数字信号处理的数字滤波器。作为对比，非自适应滤波器有静态的滤波器系数，这些静态系数一起组成传递函数。

对于一些应用来说，由于事先并不知道所需要进行操作的参数，例如一些噪声信号的特性，所以要求使用自适应的系数进行处理。在这种情况下，通常使用自适应滤波器，自适应滤波器使用反馈来调整滤波器系数以及频率响应。

总的来说，自适应的过程涉及到将**代价函数**用于确定如何更改滤波器系数从而减小下一次迭代过程成本的算法。价值函数是滤波器最佳性能的判断准则，比如减小输入信号中的噪声成分的能力。

随着数字信号处理器性能的增强，自适应滤波器的应用越来越常见，时至今日它们已经广泛地用于手机以及其它通信设备、数码录像机和数码照相机以及医疗监测设备中。

下面图示的框图是最小均方滤波器（LMS）和递归最小平方（en:Recursive least squares filter, RLS，即我们平时说的最小二乘法)这些特殊自适应滤波器实现的基础。框图的理论基础是可变滤波器能够得到所要信号的估计。



在开始讨论结构框图之前，我们做以下假设：

- 输入信号是所要信号$d(n)$和干扰噪声 $v(n)$之和

$$x(n) = d(n) + v(n)$$

- 可变滤波器具有有限脉冲响应结构，这样结构的脉冲响应等于滤波器系数。p 阶滤波器的系数定义为

$$\boldsymbol{w_n} = [w_n(0), w_n(1), w_n(2), \dots w_n(p)]^T$$

- 误差信号或者叫作代价函数，是所要信号与估计信号之差

$$e(n) = d(n) - \hat{d}(n)$$

- 可变滤波器通过将**输入信号**与**脉冲响应**作卷积，估计所要信号，用向量表示为

$$\hat{d}(n) = \boldsymbol{w_n} * \boldsymbol{x(n)}$$

其中

$$\boldsymbol{x(n)} = [x(n), x(n-1), \dots, w_n(n-p)]^T$$

是输入信号向量。另外，可变滤波器每次都会马上改变滤波器系数

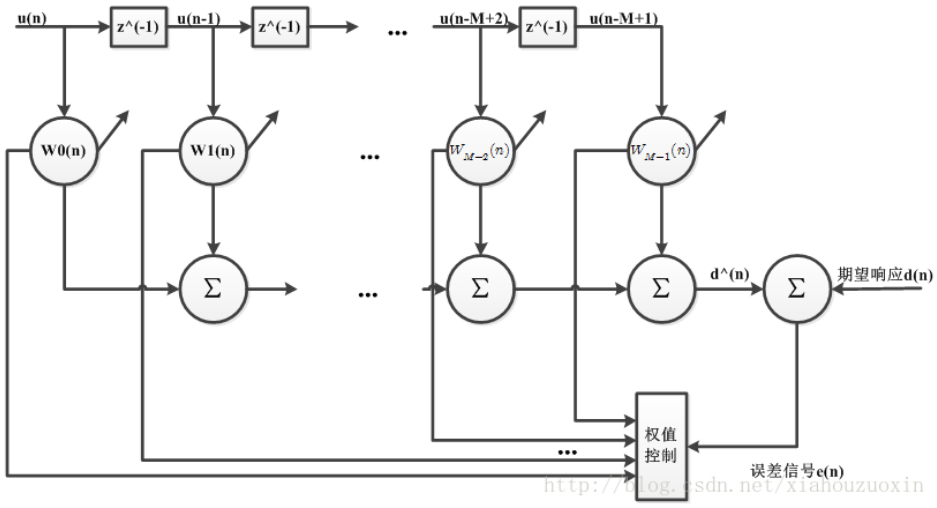
$$\boldsymbol{w_{n+1}} = \boldsymbol{w_n} + \Delta\boldsymbol{w_n}$$

其中$\Delta\boldsymbol{w_n}$是滤波器系数的校正因子。

自适应算法根据输入信号与误差信号生成这个校正因子，LMS 和 RLS 是两种不同的系数更新算法。

相对于其它类型的滤波器，自适应滤波器效果更好的关键是：**自适应滤波器是反馈结构。**

自适应滤波器的自适应过程是：用自适应算法（Update Algorithm）调节 FIR 或 IIR 滤波器的系数，使误差信号逼近于 0。



LMS 算法

LMS 算法可认为是机器学习里面最基本也比较有用的算法，神经网络中对参数的学习使用的就是 LMS 的思想，在通信信号处理领域 LMS 也非常常见，比如自适应滤波器。

本文主要对 LMS（Least Mean Square）算法进行简单的整理，包括内容：

（1） 理论上介绍基于 LMS 的梯度下降算法（包括 BACH/STOCHASTIC），给出一个 matlab 的实现

（2） DSP 上的实现，主要使用 **C 语言**

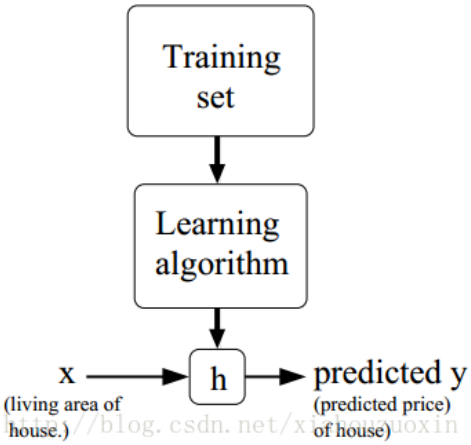
LMS 算法理论

问题引出

因为本人感兴趣的领域为机器学习，因此这里先说明下学习的过程，给定这样一个问题：某地的房价与房地面积和卧室的数量之间成如下表的关系，

Living area (feet2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

据此，我们要通过分析上面的数据学习出一个模型，用于预测其它情况（比如面积 2000，卧室数 5）的房价。这就是一个学习问题，更简洁的说，就是一个概率里的回归问题。这里固定几个符号：*x* 表示输入（[Living area，bedrooms]），*y* 表示输出（Price），*h* 表示要学习的模型，*m* 表示输入每个数据维度（这里是 2），n 表示输入数据的个数（这里是 5）。该学习过程的可以描述如下图，



h 必定与面积和卧室数相关，.这里不考虑复杂的情况，假设模型是线性的（**实际其它问题中很可能是其它关系模型，比如 exp**）

$$h(x) = w_0 + w_1 * x_1 + w_2 * x_2$$

令$x_1 = 1$，则$h(x) = w^T x$ 这里，我们考虑上面的房价问题，还是将w_0忽略。

为了获得$h(x)$，现在的问题是什么呢？那就是：怎样获得$h(x)$的$w_1 \sim w_2$的值。

我们再对问题进行描述：

- 已知——上面的数据表格，线性模型（不知道参数）
- 求解——参数$w_1 \sim w_2$

引入一个函数，叫损失函数

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}\{h(x^{(i)}) - (y^{(i)})\}^2$$

就是最小二乘法中计算误差的函数，只是前面添加了 1/2，表示什么意思呢？损失函数越小，说明模型与当前已知数据的拟合程度越好，否则越差。因此，求解$w_1 \sim w_2$的目标就是求解$J(w)$最小，这就用到了 LMS 算法。

LMS 算法

LMS 算法是一个搜索算法，假设w从某个给定的初始值开始迭代，逐渐使$J(w)$朝着最小的方向变化，直到达到一个值使$J(w)$收敛。考虑梯度下降算法（gradient descent algorithm），它通过给定的w值快速的执行如下的更新操作：

$$w_i = w_i + \alpha\frac{\partial J(w_i)}{\partial w_i}$$

其中α为学习率（Learning rate）。

要对w更新，首先需要完成上面的求导，求导的结果参见下面的算法流程。

对一个单一的训练实例 j，

$$w_i = w_i + \alpha\left(y^{(j)} - h(x^{(j)})\right)x_i^j$$

LMS 算法的求导推算

令$d(k)$代表所期望的响应，定义一个误差信号：

$$\varepsilon(x) = \mathrm{d}(k) - \mathrm{y}(k) = d(k) - \sum_{i=1}^{m} w_i x(k-i)$$

转换为向量形式，得到：

$$\varepsilon(x) = d(k) - W^T X(k) = X^T(k)W$$

误平方差为：

$$\varepsilon^2(x) = d^2(k) - 2d(k) \times X^T(k)W + W^T X(k)X^T(k)W$$

上式两边取数学期望后得：

$$E\{\varepsilon^2(x)\} = E\{d^2(x)\} - 2E\{d(k)X^T(k)\}W + W^T E\{X(k)X^T(k)\}W$$

定义互相关函数：

$$R_{Xd}^T = E\{d(k)X^T(k)\}$$

和自相关函数矩阵:

$$R_{XX} = E\{X(k)X^T(k)\}$$

则均方误差可以变为：

$$E\{\varepsilon^2(x)\} = E\{d^2(x)\} - 2R_{Xd}^T W + W^T R_{XX} W$$

这表面，均方误差是加权系数向量W的二次函数，它是一个中间向上凹的抛物形曲线，是具有唯一最小值的函数，调节权系数使均方差为最小，相当于沿抛物线曲线下降找最小值，可以用梯度法来求最小值。

将上式对权系数W求导，得到均方误差函数的梯度：

$$\nabla(k) = \nabla E\{\varepsilon^2(x)\} = \left[\frac{2E\{\varepsilon^2(x)\}}{\partial W_1}, \ldots, \frac{E\{\varepsilon^2(x)\}}{\partial W_M}\right]^T = -2R_{Xd} + 2R_{XX}W$$

要令$\nabla(k) = 0$，即可以求出**最佳权系数向量：**

$$W_{opt} = R_{XX}^{-1} R_{Xd}$$

这也是研究 Wiener 滤波器遇到过的 Wiener-Hoof 方程。因此**最佳权系数向量**W_{opt}通常也叫 Wiener 权系数向量，代入均方差公式可以得到最小均方差误差：

$$E\{\varepsilon^2(x)\}_{min} = E\{d^2(x)\} - R_{Xd}^T W_{opt}$$

利用上面的公式求最佳权系数的精确解需要知道R_{XX}和R_{Xd}的先验统计参数，而且还需要进行矩阵求逆运算，Widrow and Hoff(1960)提出了一种在这样先验统计为止的情况下求W_{opt}的近似值的方法，习惯上称 Widrow and Hoff LMS 算法，**这种算法是根据最优化方法中的最速下降法**，根据最速下降法，“下一时刻”的权系数$W(k+1)$应该等于“现时刻”的权系数$W(k)$加上一个负均方误差梯度$-\nabla(k)$的比例项，即

$$W(k+1) = W(k) - \mu\nabla(k)$$

上面中μ是一个控制收敛速度与稳定性的常数，称之为**收敛因子**。

不难看出，LMS 算法有两个关键：梯度$\nabla(k)$的计算以及收敛因子μ的选择

$\nabla(k)$的近似运算

精确计算梯度$\nabla(k)$是非常困难的，一种粗略但十分有效的计算$\nabla(k)$的近似方法是：直接取$\varepsilon^2(x)$作为均方误差$E\{\varepsilon^2(x)\}$的估算值，即：

$$\hat{\nabla}(k) = \nabla\{\varepsilon^2(x)\} = 2\varepsilon(k)\nabla[\varepsilon(k)]$$

公式中的$\nabla[\varepsilon(k)]$为：

$$\nabla[\varepsilon(k)] = \nabla[d(k) - W^T X(k)] = -X(k)$$

结合上面 2 个公式，得到梯度估计值：

$$\hat{\nabla}(k) = -2\varepsilon(k)X(k)$$

于是 Widrow and Hoff LMS 的最终式为

$$W(k+1) = W(k) + 2\varepsilon(k)X(k)$$

下面分析下梯度估计值$\hat{\nabla}(k)$的无偏性，$\hat{\nabla}(k)$的数学期望为：

$$\begin{aligned} E\{\hat{\nabla}(k)\} &= E\{-2\varepsilon(k)X(k)\} \\ &= -2E\{X(k)[d(k) - X^T(k)W(k)]\} \\ &= -2[R_{Xd} - R_{XX}]W(k) \\ &= \nabla(\mathrm{k}) \end{aligned}$$

上面的推导过程中，利用了$d(k)$和$\varepsilon(k)$二者皆为标量的事实，在得到最后的结果时，利用了$\nabla(k) = -2R_{Xd} + 2R_{XX}W$，所以在无偏性估计得到$\hat{\nabla}(k)$是无偏估计。

按照上述的更新方法，对多个实例的更新规则为

Repeat until convergence {

for every *j*, exec

$$w_i = w_i + a\sum_{j=1}^{m}\left(y^{(j)} - h(x^{(j)})\right)x_i^j$$

}

这种更新的梯度下降方法称为 batch gradient descent。还有一种更新的方式：采用随机的样本数据实例，如下

Repeat until convergence {

for every *j*, exec

$$w_i = w_i + a\left(y^{(j)} - h(x^{(j)})\right)x_i^j$$

}

这种方法称为 stochastic gradient descent (或者 incremental gradient descent)。

两种方法的明显区别是 batch 的训练时间要比 stochastic 长，但效果可能更好。实际问题中，因为我们只需要找到一个接近使$J(w)$最小的值即可，因此 stochastic 更常用。

说了这么久，LMS 到底能用来干嘛，其实上面已经很清楚了：**参数训练中的求极值。**

在 matlab 上对 stochastic gradient descent 的实现如下：

LMS.m

基于 LMS 的梯度下降算法在 DSP 上的实现

下面是在 DSP6713 上使用软件仿真实现的 LMS 算法，

zx_lms.h

zx_lms.c

输入、输出、初始权值为

```
static double init_y[] = {4.00,3.30,3.69,2.32};
static double init_x[] = {          /* 用一维数组保存 */
                        2.104, 3,
                        1.600, 3,
                        2.400, 3,
                        3.000, 4
                        };
static double weight[2] = {0.1, 0.1};
```

main 函数中只需要调用 zx_lms()就可以运行了，本文对两种梯度下降方法做了个简单对比，

max_iter=1000	w1	w2	error	CPU Cycles
batch	-0.6207369	1.419737	0.20947	2181500
stochastic	0.145440	0.185220	0.130640	995

需要说明的是：batch 算法是达到最大迭代次数 1000 退出的，而 stochastic 是收敛退出的，因此这里 batch 算法应该没有对数据做到较好的拟合。stochastic 算法则在时钟周期上只有 995，远比 batch 更有时间上的优势。

注：这里的 error 没有太大的可比性，因为 batch 的 error 针对的整体数据集的 error，而 stochastic 的 error 是针对一个随机的数据实例。

LMS 有个很重要的问题：收敛。开始时可以根据给定数据集设置 *w* 值，使 *h(x)*尽可能与接近 *y*，如果不确定可以将 *w* 设置小一点。

这里顺便记录下在调试过程中遇到的一个问题：在程序运行时发现有变量的值为 1.#QNAN。

解决：QNAN 是 Quiet Not a Number 简写，是常见的浮点溢出错误，在网上找到了解释

A QNaN is a NaN with the most significant fraction bit set. QNaN's propagate freely through most arithmetic operations. These values pop out of an operation when the result is not mathematically defined.

在开始调试过程中因为迭代没有收敛，发散使得 w 和 error 等值逐渐累积，超过了浮点数的范围，从而出现上面的错误，通过修改使程序收敛后上面的问题自然而然解决了。

LMS 算法实现

对于初学者，实际问题中，比如一个不知道分布状态的声音信号，**期望信号 d(x)的确定很难理解**(既然知道 d(x)干嘛还要滤波？？都知道答案了干嘛还要计算？？)，而很多文章都没有提到这点！在上面提到的自适应滤波器的 4 种应用中，我觉得最大的不同也在于 d(x)的不同。

从百度直到上找到一点答案：

1) 系统辨识：这时参考信号就是未知系统的输出，当误差最小时，此时自适应滤波器就与未知系统具有相近的特性，自适应滤波器用来提供一个在某种意义上能够最好拟合未知装置的线性模型
2) 逆模型：在这类应用中，自适应滤波器的作用是提供一个逆模型，该模型可在某种意义上最好拟合未知噪声装置。理想地，在线性系统的情况下，该逆模型具有等于未知装置转移函数倒数的转移函数，使得二者的组合构成一个理想的传输媒介。该系统输入的延迟构成自适应滤波器的期望响应。在某些应用中，该系统输入不加延迟地用做期望响应。
3) 预测：在这类应用中，自适应滤波器的作用是对随机信号的当前值提供某种意义上的一个最好预测。于是，信号的当前值用作自适应滤波器的期望响应。信号的过去值加到滤波器的输入端。取决于感兴趣的应用，自适应滤波器的输出或估计误差均可作为系统的输出。在第一种情况下，系统作为一个预测器；而在后一种情况下，系统作为预测误差滤波器。
4) 干扰消除：在一类应用中，自适应滤波器以某种意义上的最优化方式消除包含在基本信号中的未知干扰。基本信号用作自适应滤波器的期望响应，参考信号用作滤波器的输入。参考信号来自定位的某一传感器或一组传感器，并以承载新息的信号是微弱的或基本不可预测的方式，供给基本信号上。

这也就是说，得到期望输出往往不是引入自适应滤波器的目的，**引入它的目的是得到未知系统模型、得到未知信道的传递函数的倒数、得到未来信号或误差和得到消除干扰的原信号**

这里也有部分关于参考信号的讨论：http://www.amobbs.com/thread-5535155-1-1.html

======= Matlab 代码演示 =======

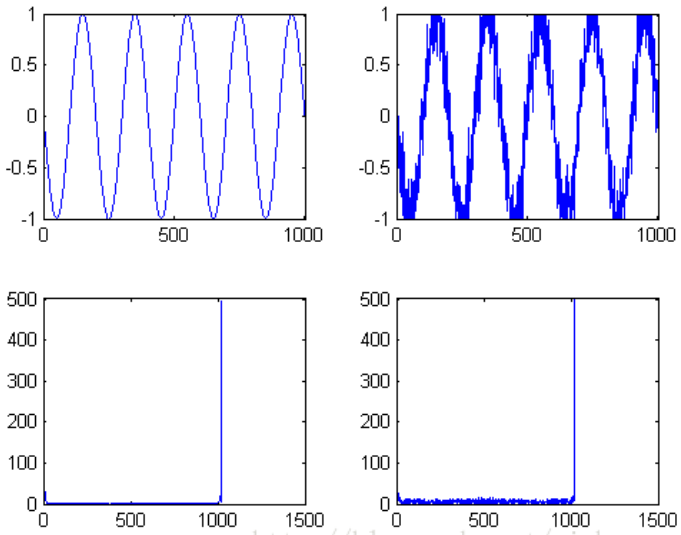
自适应滤波器的函数

zx_lms.m

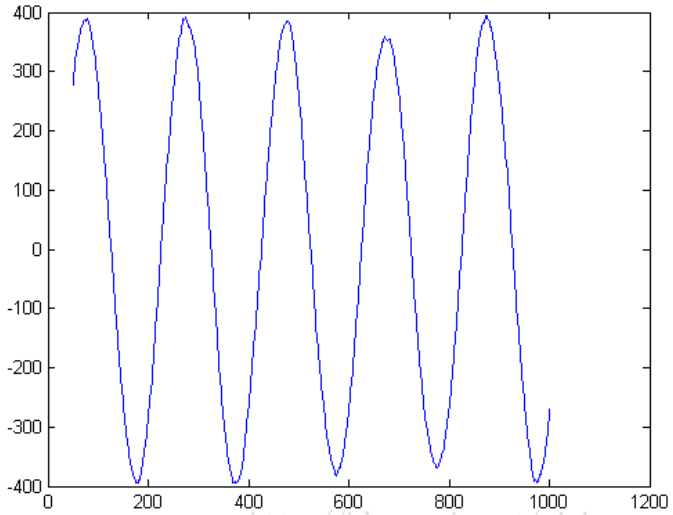
调用自适应滤波器实例，参考信号 d(x)为正弦信号，加高斯白噪声后构成输入信号 x(n)，

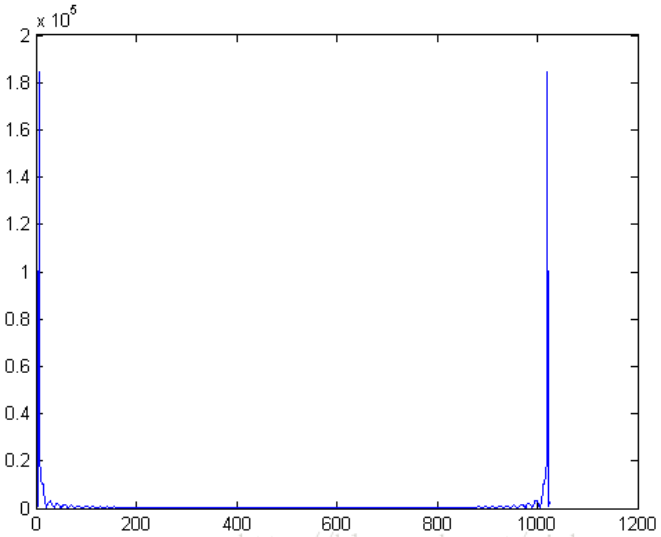
lms_fft.m

使用 LMS 自适应滤波器前原始信号和加高斯白噪声后的信号（时域+频域）如下，



使用 LMS 对添加了高斯白噪声的信号滤波后效果如下，





-温嘉颖整理，非原创

2016@SUTD

http://blog.csdn.net/xiahouzuoxin/article/details/11138211

http://blog.csdn.net/xiahouzuoxin/article/details/9749689

http://www.cnblogs.com/guluxuanyuan/p/4062176.html