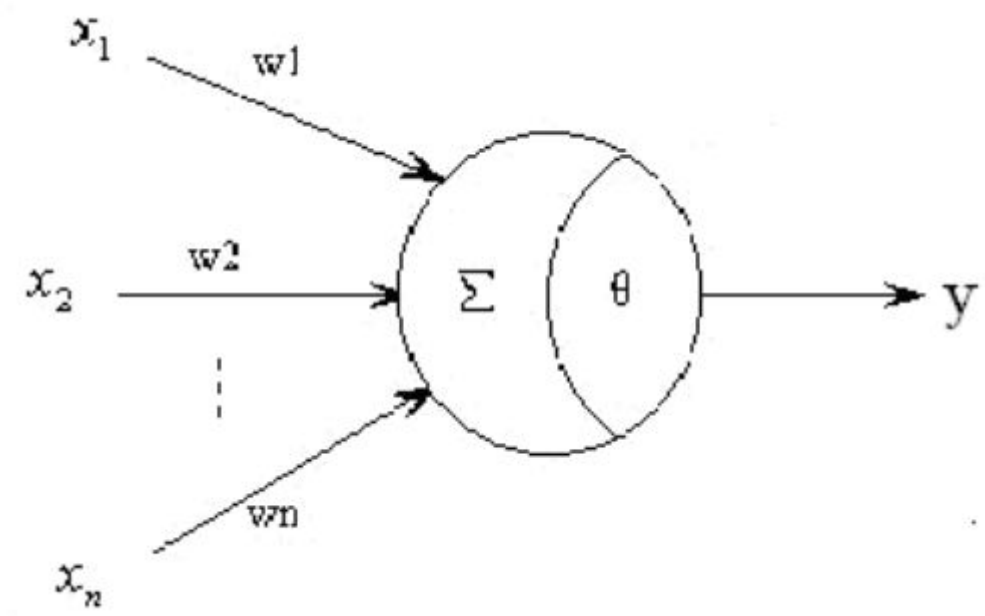
股票预测算法梳理

1．NN（Neural Networks）

1.1 人工神经元的数学模型

人工神经元是一个多输入单输出的信息处理单元，如图：



图中运算关系为：

$$x=\sum_{i=1}^{n} w_i x_i-\theta$$

x 为当前神经元所接收的输入信号的总和，其中 θ 为兴奋阈值。

$$y=f(\sum_{i=1}^{n} w_i x_i-\theta)$$

若将阈值也作为一个权值，并记 w 0 = － θ , x 0 = 1，则上述 y 也可写作：

$$y=f(\sum_{i=0}^{n} w_i x_i)$$

其中阈值函数（激励函数）可以有多种模型，如下几个常见函数模型：

- 阈值函数

$$f(x)=\begin{cases}1 & x>0\\ 0 & x\le 0\end{cases}$$

- 双向阈值函数

$$f(x)=\begin{cases}1 & x>0\\ -1 & x\le 0\end{cases}$$

- S 函数

$$f(x)=\frac{1}{1+e^{-x}}$$

- 双曲正切函数

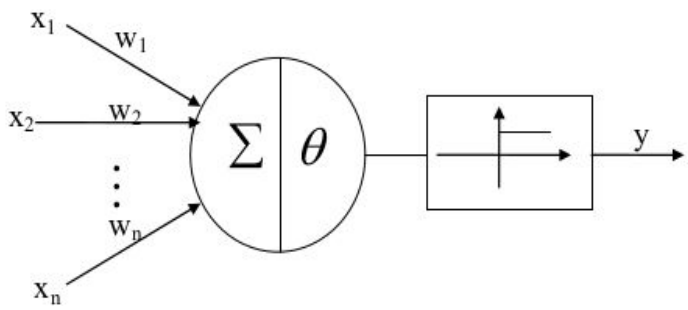
$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

- 高斯函数

$$f(x)=e^{-(x^2/\sigma^2)}$$

1.2 感知器

单层感知器可以看成是多输入单输出的单个神经元,其激励函数是阈值函数或双阈值函数。



单神经元感知器的输出是

$$y = f(\sum_{i=1}^{n} \omega_i x_i - \theta)$$

将输入和权值写成向量的形式,即记

$$\mathbf{x} = (x_1 \quad x_2 \quad \cdots \quad x_n)^T \qquad \boldsymbol{\omega} = (\omega_1 \quad \omega_2 \quad \cdots \quad \omega_n)^T$$

则有: $y = f(\boldsymbol{\omega}^T \mathbf{x} - \theta)$

记 $d(x) = \boldsymbol{\omega}^T \mathbf{x} - \theta$ ，称为决策函数，而 d(x) = 0 所形成的分界线可以将输入分成两类。

单神经元感知器是典型的有教师学习训练,设训练样本包含有 p 个输入输出对 $\{\mathbf{x}^l, \bar{\mathbf{y}}^l\}$ (l = 1 , 2 , " , p) , 相应的学习(或称训练)算法如下:

- 取算法部数 k=0,各个权值置为小的随机数 $\boldsymbol{\omega}(0) = (\omega_1(0) \quad \omega_2(0) \quad \cdots \quad \omega_n(0))^T$ 置阈值初值 θ (0)
- 任取样本集中的一对样本 $\{\mathbf{x}^l, \bar{\mathbf{y}}^l\}$,计算
- $y^l = f(\boldsymbol{\omega}^T(0)\mathbf{x}^l - \theta(0))$
- 对连接权值和阈值进行修正:

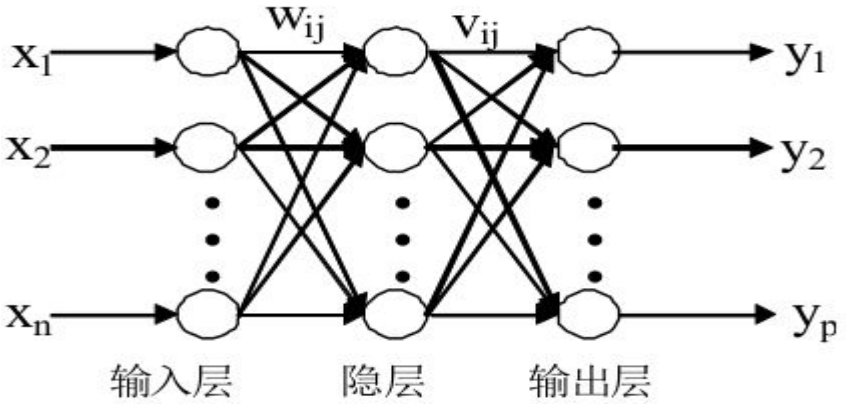
$$\boldsymbol{\omega}(k+1) = \boldsymbol{\omega}(k) + \alpha(\bar{y}^l - y^l)\mathbf{x}^l$$

- $\theta(k+1) = \theta(k) - \alpha(\bar{y}^l - y^l)$
- 置 k=k+1,重复以上 2、3 步骤,直到对所有的样本都成立:

$\boldsymbol{\omega}(k+1) = \boldsymbol{\omega}(k)$ 和 $\theta(k+1) = \theta(k+1)$

1.3 神经网络

前向网络是目前研究最多的网络形式之一,它包含输入层,隐含层和输出层,其中隐含层可为一层和多层。本节只考虑只具有一层隐含层的前向网络,其中输入层有 n 个节点,隐含层有 m 个节点,输出层有 p 个节点,



由于前向神经网络权值阈值修正算法推导较为复杂，在此不做推导，只列出计算步骤：

(1) 置各权值和阈值的初始值$v_{li}(0)$，$w_{ij}(0)$，隐含层神经元的阈值$\theta_i(0)$和输出层节点的阈值$\xi_l(0)$;设定修正项调节参数η和η'及期望误差$\varepsilon > 0$。

(2)输入样本训练集$\{\mathbf{x}_q, \bar{\mathbf{y}}_q\}$, $q = 1,2,\cdots,s$ (即设有s对样本)，对每个样本进行如下的(3)~(4).

(3)计算网络在k步迭代的各项数据，为此，按如下先后顺序计算

$$net_i^1(k) = \sum_{j=1}^{n} w_{ij}(k)x_j - \theta_i(k) \qquad z_i(k) = f(net_i^1(k))$$

$$net_l^2(k) = \sum_{i=1}^{m} v_{li}(k)z_i(k) - \xi_l(k) \qquad y_l(k) = f(net_l^2(k))$$

$$f'(net_l^2(k)) = y_l(k)(1 - y_l(k)) \qquad \delta_l^2(k) = -(\bar{y}_l - y_l(k)) \cdot f'(net_l^2(k))$$

$$f'(net_i^1(k)) = z_i(k)(1 - z_i(k)) \qquad \delta_i^1(k) = f'(net_i^1(k)) \cdot \sum_{l=1}^{p} \delta_l^2(k) \cdot v_{li}(k)$$

于是，对输出节点和隐含层节点的权值和阈值按如下方式修正

$$v_{li}(k+1)=v_{li}(k)-\eta\delta_l^2(k)z_i(k) \qquad w_{ij}(k+1)=w_{ij}(k)-\eta'\delta_i^1(k)x_j$$

$$\xi_l(k+1)=\xi_l(k)+\eta\delta_l^2(k) \qquad \theta_i(k+1)=\theta_i(k)+\eta'\delta_i^1(k)$$

同时，计算出当前样本(设为第s个)的误差函数

$$E_s(k+1)=\frac{1}{2}\sum_{l=1}^{p}(\bar{y}_l-y_l(k+1))^2$$

(4)当样本结中的所有样本都经历了(3)～(4)步，即完成了一个训练周期。计算

$$E(k+1)=\sum_{s=1}^{q}E_s(k+1)$$

如果 $E(k+1)\le\varepsilon$ ，则当前 $k+1$ 步所得到的权值和阈值 $T_{li}(k+1),\ w_{ij}(k+1),\ \xi_l(k+1),\ \theta_i(k+1)$ 作为网络训练成功的权值和阈值，计算结束退出。否则返回第2步。

注：以上计算均要求对 $j=1,2,\cdots,n;\ i=1,2,\cdots,m$ 和 $l=1,2,\cdots,p$ 进行。

1.4 使用 Tensorflow 构建神经网络

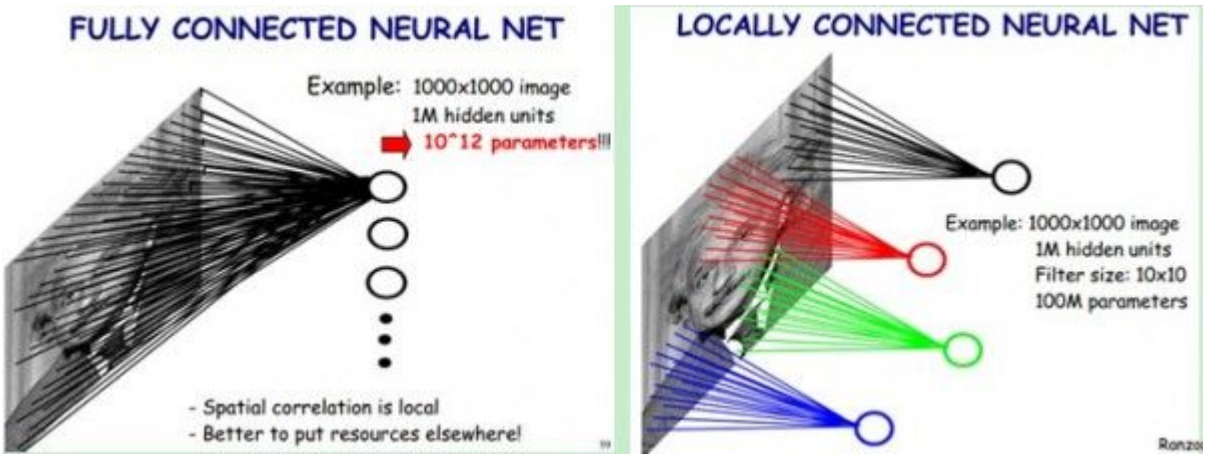
详见附录 digit_nn.py

2. CNN（convolutional neural network）

在图像处理中，往往把图像表示为像素的向量，比如一个 1000×1000 的图像，可以表示为一个 1000000 的向量。在上一节中提到的神经网络中，如果 隐含层数目与输入层一样，即也是 1000000 时，那么输入层到隐含层的参数数据为 1000000×1000000=10^12，这样就太多了，基本没法 训练。所以图像处理要想练成神经网络大法，必先减少参数加快速度。

2.1 局部感知

卷积神经网络有两种神器可以降低参数数目，第一种神器叫做局部感知野。一般认为人对外界的认知是从局部到全局的，而图像的空间联系也是局部的像素联系较为紧密，而距离较远的像素相关性则较弱。因而，每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知，然后在更高层将局部的信息综合起来 就得到了全局的信息。网络部分连通的思想，也是受启发于生物学里面的视觉系统结构。视觉皮层的神经元就是局部接受信息的（即这些神经元只响应某些特定区域 的刺激）。如下图所示：左图为全连接，右图为局部连接



在上右图中，假如每个神经元只和 10×10 个像素值相连，那么权值数据为

1000000×100 个参数，减少为原来的千分之一。而那 10×10 个像素值对应的 10×10 个参数，其实就相当于卷积操作。

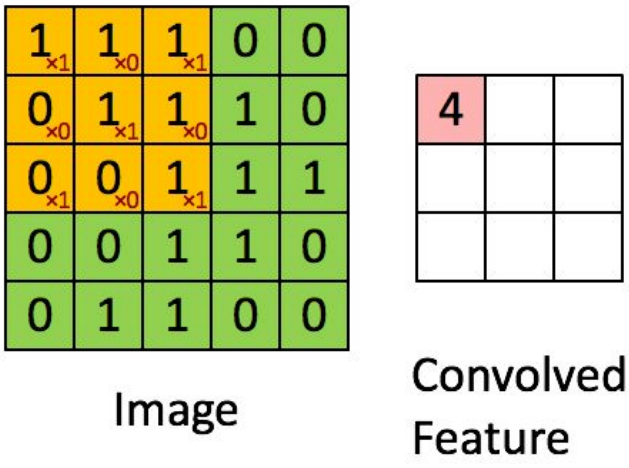
2.2 参数共享

但其实这样的话参数仍然过多，那么就启动第二级神器，即权值共享。在上面的局部连接中，每个神经元都对应 100 个参数，一共 1000000 个神经元，如果这 1000000 个神经元的 100 个参数都是相等的，那么参数数目就变为 100 了。

怎么理解权值共享呢？我们可以这 100 个参数（也就是卷积操作）看成是提取特征的方式，该方式与位置无关。这其中隐含的原理则是：图像的一部分的统 计特性与其他部分是一样的。这也意味着我们在这一部分学习的特征也能用在另一部分上，所以对于这个图像上的所有位置，我们都能使用同样的学习特征。

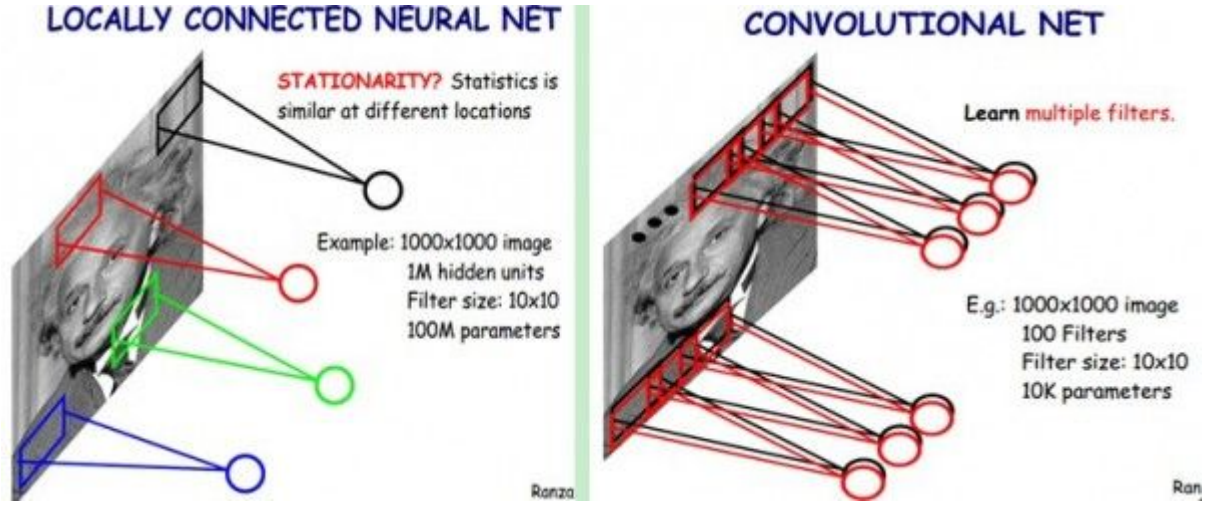
更直观一些，当从一个大尺寸图像中随机选取一小块，比如说 8×8 作为样本，并且从这个小块样本中学习到了一些特征，这时我们可以把从这个 8×8 样本中学习到的特征作为探测器，应用到这个图像的任意地方中去。特别是，我们可以用从 8×8 样本中所学习到的特征跟原本的大尺寸图像作卷积，从而对这个大尺寸图像上的任一位置获得一个不同特征的激活值。

如下图所示，展示了一个 33 的卷积核在 55 的图像上做卷积的过程。每个卷积都是一种特征提取方式，就像一个筛子，将图像中符合条件（激活值越大越符合条件）的部分筛选出来。动态效果请看附录中“卷积过程.gif”



2.3 多卷积核

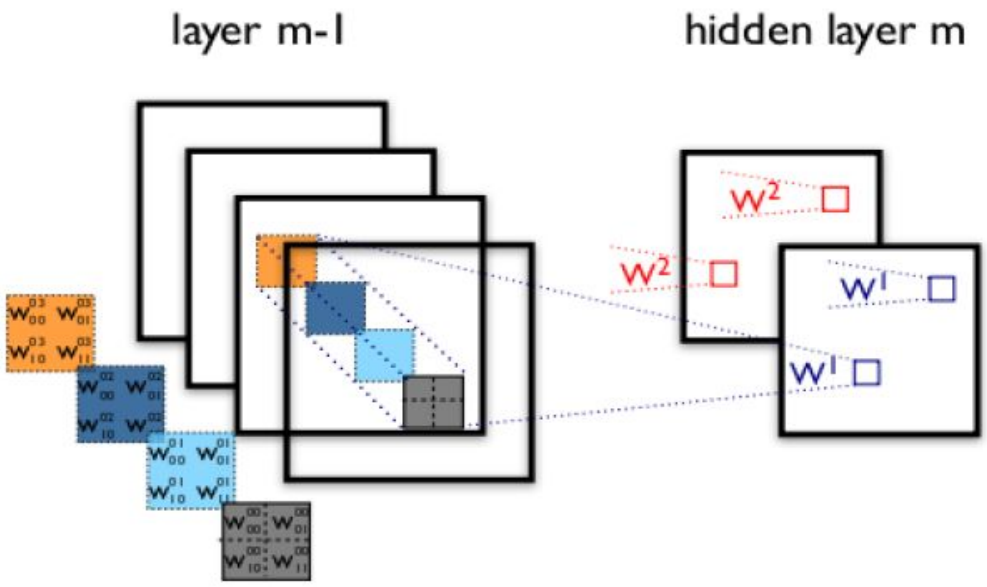
上面所述只有 100 个参数时，表明只有 1 个 100*100 的卷积核，显然，特征提取是不充分的，我们可以添加多个卷积核，比如 32 个卷积核，可以学习 32 种特征。在有多个卷积核时，如下图所示：



上图右，不同颜色表明不同的卷积核。每个卷积核都会将图像生成为另一幅图像。比如两个卷积核就可以将生成两幅图像，这两幅图像可以看做是一张图像的不同

的通道。如下图所示，下图有个小错误，即将 w1 改为 w0，w2 改为 w1 即可。下文中仍以 w1 和 w2 称呼它们。

下图展示了在四个通道上的卷积操作，有两个卷积核，生成两个通道。其中需要注意的是，四个通道上每个通道对应一个卷积核，先将 w2 忽略，只看 w1，那么在 w1 的某位置（i,j）处的值，是由四个通道上（i,j）处的卷积结果相加然后再取激活函数值得到的。



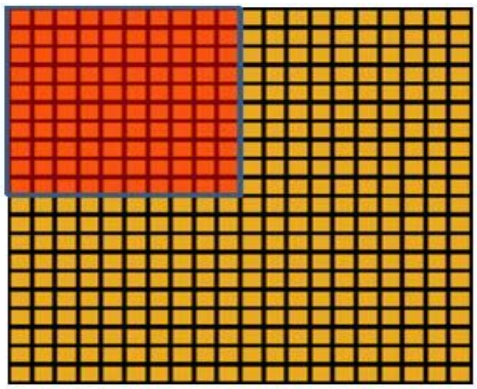
所以，在上图由 4 个通道卷积得到 2 个通道的过程中，参数的数目为 4×2×2×2 个，其中 4 表示 4 个通道，第一个 2 表示生成 2 个通道，最后的 2×2 表示卷积核大小。

2.4 Down-pooling

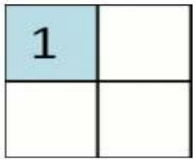
在通过卷积获得了特征 (features) 之后，下一步我们希望利用这些特征去做分类。理论上讲，人们可以用所有提取得到的特征去训练分类器，例如 softmax 分类器，但这样做面临计算量的挑战。例如：对于一个 96X96 像素的图像，假设

我们已经学习得到了 400 个定义在 8X8 输入上的特征，每一个特征和图像卷积都会得到一个 $(96-8+1)\times(96-8+1)=7921$ 维的卷积特征，由于有 400 个特征，所以每个样例 (example) 都会得到一个 $892\times400=3,168,400$ 维的卷积特征向量。学习一个拥有超过 3 百万特征输入的分类器十分不便，并且容易出现过拟合 (over-fitting)。

为了解决这个问题，首先回忆一下，我们之所以决定使用卷积后的特征是因为图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在 另一个区域同样适用。因此，为了描述大的图像，一个很自然的想法就是对不同位置的特征进行聚合统计，例如，人们可以计算图像一个区域上的某个特定特征的平均值(或最大值)。这些概要统计特征不仅具有低得多的维度 (相比使用所有提取得到的特征)，同时还会改善结果(不容易过拟合)。这种聚合的操作就叫做池化 (pooling)，有时也称为平均池化或者最大池化 (取决于计算池化的方法)。



Convolved feature



Pooled feature

至此，卷积神经网络的基本结构和原理已经阐述完毕。

2.5 多层卷积

在实际应用中，往往使用多层卷积，然后再使用全连接层进行训练，多层卷积的目的是一层卷积学到的特征往往是局部的，层数越高，学到的特征就越全局化。

2.6 使用 tensorflow 构建卷积神经网络

参见附录 mnist_convolutional 文件夹中的内容。

3．Arima

3.1 AR（p）(p 阶自回归模型）

$$x_t = \delta + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + u_t$$

其中 ut 白噪声序列，δ是常数（表示序列数据没有 0 均值化）AR（p）等价于

$$(1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p) x_t = \delta + u_t$$

AR（p）的特征方程是：

$$\Phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p = 0$$

AR（p）平稳的充要条件是特征根都在单位圆之外。

3.2 MA（q）（q 阶移动平均模型）

$$x_t = \mu + u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q}$$

$$x_t - \mu = (1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q) u_t = \Theta(L) u_t$$

其中{ut}是白噪声过程

MA（q）平稳性

MA（q）是由 ut 本身和 q 个 ut 的滞后项加权平均构造出来的，因此它是平稳的。 MA（q）

可逆性（用自回归序列表示 ut）

$$u_t = [\Theta(L)]^{-1} x_t$$

可逆条件：即 $u_t = [\Theta(L)]^{-1} x_t$ 收敛的条件。即Θ（L）每个特征根绝对值大于 1，即全部特征根在单位圆之外。

3.3 ARMA（p，q）（自回归移动平均过程）

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \delta + u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q}$$

$$\Phi(L)x_t = (1-\phi_1 L-\phi_2 L^2-\cdots-\phi_p L^p)x_t$$
$$= \delta + (1+\theta_1 L+\theta_2 L^2+\cdots+\theta_q L^q)u_t = \delta + \Theta(L)u_t$$

$$\Phi(L)x_t = \delta + \Theta(L)u_t$$

ARMA（p，q）平稳性的条件是方程Φ（L）=0 的根都在单位圆外；可逆性条件是方程Θ（L）=0 的根全部在单位圆外。

3.4 ARIMA（p，d，q）（单整自回归移动平均模型）

差分算子：

$$\Delta x_t = x_t - x_{t-1} = x_t - Lx_t = (1-L)x_t$$
$$\Delta^2 x_t = \Delta x_t - \Delta x_{t-1} = (1-L)x_t - (1-L)x_{t-1} = (1-L)^2 x_t$$
$$\Delta^d x_t = (1-L)^d x_t$$

对 d 阶单整序列 xt~I(d)

$$w_t = \Delta^d x_t = (1-L)^d x_t$$

则 wt 是平稳序列，于是可对 wt 建立 ARMA（p，q）模型，所得到的模型称为 xt~ARIMA（p，d，q），模型形式是

$$w_t = \phi_1 w_{t-1} + \phi_2 w_{t-2} + \cdots + \phi_p w_{t-p} + \delta + u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q}$$

$$\Phi(L)\Delta^d x_t = \delta + \Theta(L)u_t$$

由此可转化为 ARMA 模型。

ARMA（p，q）有拖尾特征，p 和 q 的识别通过从低阶逐步试探直到合适的模型为止

3.5 基本程序

第一步，根据时间序列的散点图、自相关函数和偏自相关函数图以 ADF 单位根检验其方差、趋势及其季节性变化规律，对序列的平稳性进行识别。一般来讲，经济运行的时间序列都不是平稳序列。

第二步，对非平稳序列进行平稳化处理。如果数据序列是非平稳的，并存在一定的增长或下降趋势，则需要对数据进行差分处理，如果数据存在异方差，则需对数据进行技术处理，直到处理后的数据的自相关函数值和偏相关函数值无显著地异于零。

第三步，根据时间序列模型的识别规则，建立相应的模型。若平稳序列的偏相关函数是截尾的，而自相关函数是拖尾的，可断定序列适合 AR 模型；若平稳序列的偏相关函数是拖尾的，而自相关函数是截尾的，则可断定序列适合 MA 模型；若平稳序列的偏相关函数和自相关函数均是拖尾的，则序列适合 ARMA 模型。

第四步，进行参数估计，检验是否具有统计意义。 第五步，进行假设检验，诊断残差序列是否为白噪声。 第六步，利用已通过检验的模型进行预测分析。

代码参考 arima_find_pdq.py

4. GA (Genetic Algorithm)

4.1 基本概述

遗传算法是建立在自然选择和自然遗传学机理基础上的迭代自适应概率性搜索算法,达尔文的“适者生存”基本理论贯彻于整个算法。它采用选择、复制、交叉和变异等基因操作作为其模型,并将这些算子应用于抽象的染色体上,使得(父代)优良品质得到保留并重新组合得到更好的染色体。随着这些操作不断进行,好的特征被(后代)继承下来,差的特征逐渐被淘汰,这样,整个群体就得到了进化,以此向最优解收敛,从而达到对最优化问题的求解。

遗传优化与传统优化

优化问题是指在满足等式或不等式表示的约束条件下的多变量函数的最小化和最大化问题。传统的优化方法主要有三种类型:解析法、枚举法和随机法。其中解析法是传统方法中研究得最多的一种。解析法又可以分为直接法和间接法。间接法是通过令目标函数的梯度为零,进而求解一组非线性方程来寻找局部极值的方法;直接法是按照梯度提供的信息确定一个寻优方向,并按该方向逐步进行寻优的方法,如最速下降法。具体描述如下:

考虑无约束最小化问题

$$\min_{x\in D} f(x)$$

其中 $D\in\mathbf{R}^n$, $f:\mathbf{R}^n\to\mathbf{R}$ 。定义

$$\nabla f(x)=\left(\frac{\partial f(x)}{\partial x_1}\quad\frac{\partial f(x)}{\partial x_2}\quad\cdots\quad\frac{\partial f(x)}{\partial x_n}\right)^T$$

称 ∇ f(x) 为函数 f(x)沿着向量 x 的梯度。如果已知函数 f(x)在定义域 D 内达到最小值,则最小值必在据点,即 D 中满足如下方程组的解集中达到: ∇ f(x) = 0。该方法称为间接法。

直接法的代表是最速下降法,它是一种迭代方法,极值点按梯度的负方向反复迭代而得到,具体算法如下:

最速下降法算法:

Step1:选取初始点 x 0 ,给定允许误差 ε > 0 ,令 k=0

Step 2: 计算$d_k = -\nabla f(x_k)$,若 $\|d_k\| < \varepsilon$,则迭代终止,x k 即为最优问题(5.1)的近似最优解;否则,转入 Step3

Step 3:进行一维搜索,求取 λ_k 和下一步迭代点x_{K+1}。

$$f(x_k + \lambda_k d_k) = \min_{\lambda \geq 0} f(x_k + \lambda d_k)$$

$$x_{k+1} = x_k + \lambda_k d_k$$

从上面可以看出,解析法求解优化问题,要求目标函数必须连续、光滑和可导。通过解析法寻优只能得到获得局部最优解。

遗传算法的特点:

1)遗传算法以决策变量的编码作为运算对象。

2)遗传算法对解空间的搜索是基于一组可能解组成的种群而进行的,而初始种群通常由随机个体产生。

3)遗传算法采用关于每个可能解的评价来引导搜索。

4)遗传算法的各种操作是基于概率性规则而不是确定性规则进行的

5)遗传算法对目标函数的类型和性质基本没有要求,对函数的连续性、可微性和光滑不作要求。目标函数可以是解析表达的显函数,也可以是映射矩阵或神经网络等各种隐函数,所以,具有很强的通用性。

4.2 遗传算法的基本原理

遗传算法的基本思想是,由一个个体(由编码表示,如二进制编码)(Individual)组成一个群体(Population)。每一个个体代表问题的一个潜在的解(可行解)。每一个个体均受到评价并得到相应的适应值(Fitness),以反映该个体对环境的适应度。种群中的部分个体需要经过遗传操

作(Genetic Operation)并由此产生新个体。遗传操作其实际是随机变换,主要有选择、复制、交叉和变异。受遗传算子直接操作的个体称为父代(Parent)个体,新产生的个人称为后代或子代(Offspring)个体。从父代种群中选取比较优秀的个体就形成了新的种群,新的种群称为新的父代。这样的过程循环往复,经过若干代后,算法就收敛到一个最优个体,即优化问题的最优解或者次优解。

遗传算法的基本操作

遗传算法有五个基本组成部分(1)问题可行解的遗传表示;(2)创建解的初始种群的方法;(3)根据个体适应值对其进行优劣判断的评价函数;(4)用来改变复制过程中产生的子代个体的遗传算子;(5)遗传算法的参数修改。

可行解是由很多个体所组成。遗传算法的个体也称为染色体,遗传算法中的个体以串位式的编码表示(如二进制编码),编码中的每一位称为基因。在下面的阐述中,将不加区别地使用个体、染色体和串位这三个术语。下面结合一个例子来简单介绍遗传算法的基本操作。

设待求解的优化问题是 $\max_{x\in D} f(x)$,其中 $f(x)=x^2$,而可行解为 D = { 1 , 2 , " , 31 }。

第一步:根据求解精度,选 5 位二进制来表示可行解。如 x = 12 → 01100 , x = 31 → 11111 。

第二步:遗传算法中"代"的概念是通过种群来表示的。若干个个体组成一个种群,遗传算法的操作是从某个种群开始的,而该种群就称为初始种群。初始种群通常是随机产生。对本例,假设随机产生 4 个个体:01101,11000,01000,10011 组成初始种群。

第三步:选择和复制:首先要确定出一个适应度函数,用于判断个体对环境的适应能力。适应度函数一般由目标函数构造而成,本例直接以目标函数 f(x)=x 2 作为适应度函数。对种群中的个体进行逐个解码并根据适应度函数计算出它们的适应值,以此为根据确定出各个个体被复制的概率。如果优化问题是使得适应值最大化,并且各个染色体的适应值均为正,那么就

采用轮盘赌的方式进行选择。基本原理是:根据每个染色体适应值在整个种群中所占的比例来画出一个圆盘,然后转动圆盘若干次来选择个体。实际计算时,可以按下式计算各个个体的期望复制数,然后通过取整得到实际复制数:

$$N_i = N\frac{f_i}{\sum_i f_i}$$

其中,N 是种群规模(即种群的个体个数), f i 是第 i 个个体的适应值, ∑ f i 是当前种群所有个体 i 适应值之和,N i 是第 i 个个体的复制数。新一代种群产生过程见下表:

序号	当前种群	x值	$f(x)=x^2$	选择概率 $f_i/\sum_i f_i$	期望复制数 $Nf_i/\sum_i f_i$	实际复制数	产生的新种群
1	01101	13	169	14.4%	0.576	1	
2	11000	24	576	49.2%	1.968	2	
3	01000	8	64	5.5%	0.22	0	01101
4	10011	19	361	30.9%	1.236	1	11000
总计			1170	100	4.0	4	11000
平均值			293	25	1.0	1	10011
最大值			576	49.2	1.968	2	

第四步:交叉:交叉分两步进行,首先对经过复制生成的新的种群中的部分或全部染色体随机地进行两两配对;然后对配对的染色体进行部分的基因位的交叉操作,即首先选择一个交叉点,并把交叉点以后的部分互换而产生两个子代染色体,这种交叉方式称为单点交叉。交叉的作用是实现优势互补,通过交叉,种群的总体适应值和平均适应性能得到改善,当然,不排除个体性能变差的情况。交叉操作如下图:

序号	当前种群	配对情况	交叉点	新种群	x值	$f(x)=x^2$
1	01101	1与2	4	01100	12	144
2	11000			11001	25	625
3	11000	3与4	2	11011	27	729
4	10011			10000	16	256
总值						1754
平均值						439
最大值						729

第五步:变异:变异是随机地将染色体的某些基因的状态变成其它状态而产生新的染色体的过程。对于二进制位串,变异就是对随机选取的位进行 1 → 0 或 0 → 1 的操作。变异的概率通常很小,其经典取值为 0.001~0.01。对于本例,种群共有 4×5=20 位,如果变异的概率是 0.001,那么变异的位数是 20×0.01=0.2 位,所有的基因均不参与变异。而如果变异的概率是 0.05,则变异的概率是 20×0.05=1 位,变异位随机选取。如果变异为刚好是新种群的第 3 位,则个体 11011 →11111,则通过变异达到最优点(最大适应值)。

4.3 DEAP 实现遗传算法

DEAP 在实现 GA 过程中的思路一般是很清晰的

1．确定 Types——creator 创建，选好解决问题的类型

2．初始化 Initialization ——toolbox 注册个体啊，种群啊等等函数。

3．Operator ——算子选择，交叉，变异，选择，进化等等。每个算子都有不同的算法，可以选择的！

4．Algorithms ——算法就是具体将上面注册的函数啊，等等应用结合起来，编写流程。

例子可以参照附录 GA_deap