QBUS6840 TUT 7 exponential smoothing(seasonal)

之前讨论的模型在预测时并没有考虑到数据集的季节性

Additive Holt-Winters smoothing

Lecture6 p3

- The ideal scenario

trend

$$y_t = \omega_0 + \omega_1 t + S_t + \varepsilon_t$$

- Additive decomposition model: assuming ω_0, ω_0 and S_t (M different values) are fixed constants.
- Simple exponential method: modelling the case where $S_t = 0$, $\omega_1 = 0$ (or constant) and ω_0 changes with time
- Trend corrected exponential method: modelling the case where $S_t = 0$, both ω_1 and ω_0 are changing
- How to model the data if the level, the level growth rate (the trend), and seasonal patterns are changing?

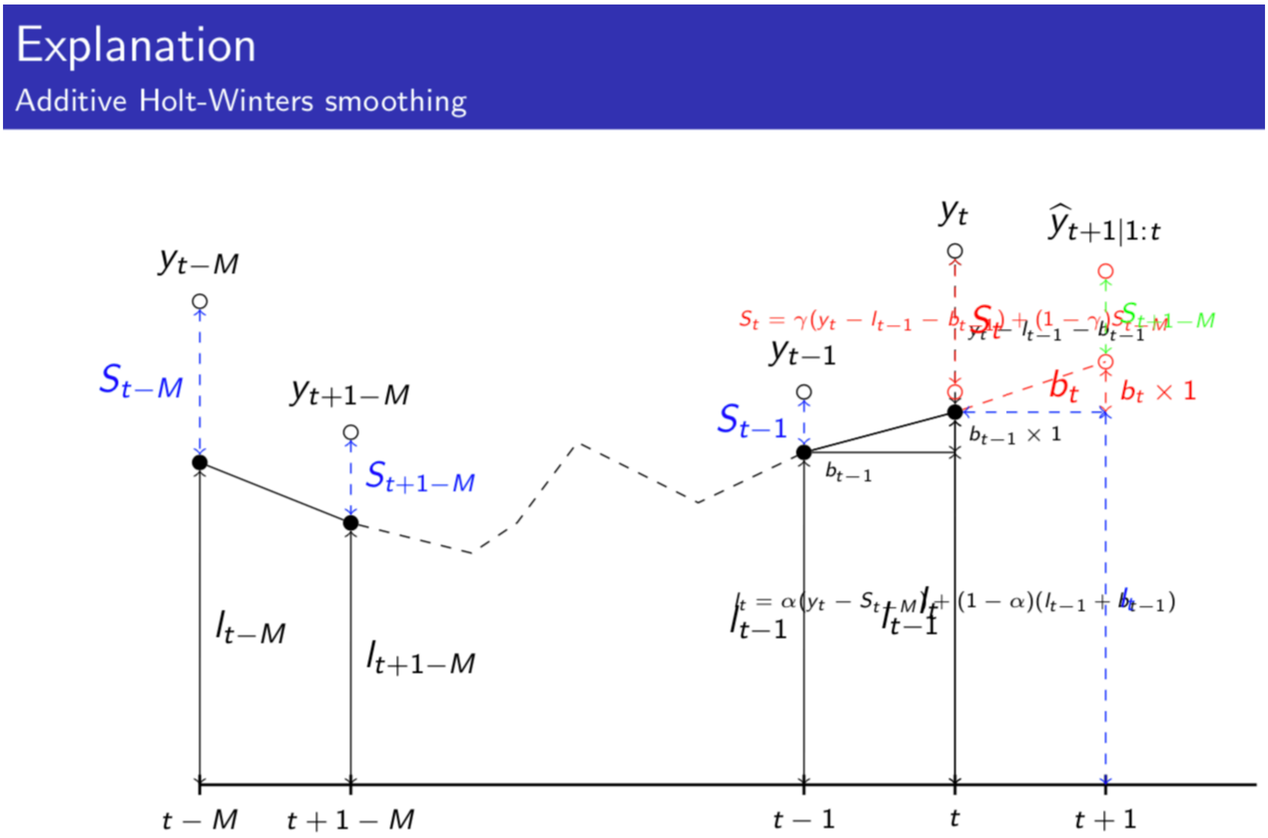
- 联系上 第 2 周和第3 周我们学过的

  Additive model 的分解公式：

$$y_t = T_t + S_t + C_t + e_t$$

- 第5周学的 Holt linear model （l 和 b）只是对 $T \times C$ 部分的预测，如果要完善，需要我们添加对 S 部分的预测

图示



递推公式

$$
\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},
\end{aligned}
$$

level 是在时间 t 的 the seasonally adjusted observation $(y_t - s_{t-m})$ 和 non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ 的加权平均值.

trend 是在 时间 t 的 预估 trend $\ell_t - \ell_{t-1}$, 和 之前的预估 trend b_{t-1}的加权平均值.

seasonal 是 the current seasonal index, $(y_t - \ell_{t-1} - b_{t-1})$ 和 上一周期的相同时间的 the seasonal index 的加权平均值.

- hyper parameters : α β γ
- l_0 b_0 和 $\hat{s}_t$(seasonal indice) 对预测的结果影响很大

手动实现 Additive Holt-winters smoothing

初始化参数

用 linear regression 找到 l_0 b_0

Additive Holt-Winters smoothing

Choice of initial values

How should we set the initial values l_0, b_0, s_0, s_{-1}, ..., s_{2-M}, s_{1-M}?

Suggested Method

1. Do a linear least square regression over the data $y_1, \ldots, y_T$ to find out
$$\widehat{y}_t = \widehat{\omega}_0 + \widehat{\omega}_1 t$$
2. Take $l_0 = \widehat{\omega}_0$ and $b_0 = \widehat{\omega}_1$
3. Find out $\widehat{s}_t = y_t - \widehat{y}_t$, then take the average of $\widehat{s}_t$ as one of s_0, s_{-1}, ..., s_{2-M}, s_{1-M} according to each season.

```python
1   from sklearn.linear_model import LinearRegression
2   def linearOptimization(X, m):
3       x = np.linspace(1, len(X), len(X))
4       x = np.reshape(x, (len(x), 1))
5       y = np.reshape(X, (len(X), 1))
6   
7       # 大 X 是原数据 y_t
8   
9       lm = LinearRegression().fit(x, y)
10      l0 = lm.intercept_
11      b0 = lm.coef_[0]
```

- 第3,4行生成 从1开始的数字序列组成的向量
- 第5行生成 y_t 的二维向量
- 第9行 用 LinearRegression 库做 线性回归， l_0 就是 intercept , l_1 就是斜率

计算 s_0 到 s_{M-1}

```python
res = y - lm.predict(x) + 0.
res = np.reshape(res,(m,int(len(X)/m)))
s = np.mean(res,axis = 0)

return l0[0], b0, s.tolist()
```

- 第1行 $\hat{s} = y - \hat{y}$； +0. 是转成浮点数
- 第2行 转成大小为 (周期, 周期数) 的向量
- 第3行 对每个周期时间点取 平均数
- 返回值 l_0 b_0 和 $\overline{s}_m$(seasonal indice)

- The usual assumption (under monthly data) is that

$$S_t = S_{t-12} = S_{t+12} \Rightarrow \widehat{S}_t = \overline{S}_m$$

where $\overline{S}_m$ is called seasonal index and they ($m = 1, 2, \ldots, M$)are the normalized average of all observations in the month m of historical data, i.e.,

$$\sum_{m=1}^{M} \overline{S}_m = M.$$

用迭代公式 进行预测

```
def addSeasonal(x, m, fc, alpha = None, beta = None, gamma = None, l0 = None, b0 = None, s = None):
    Y = x[:]
    l = []
    b = []
    s = []

    if (alpha == None or beta == None or gamma == None):
        alpha, beta, gamma = 0.1, 0.1, 0.1

    if (l0 == None or b0 == None or s == None):
        l0,b0,s = linearOptimization(Y,m)
        l.append(l0)
        b.append(b0)
    else:
        l = l0
        b = b0
        s = s

    forecasted = []
    rmse = 0

    for i in range(len(x) + fc):
        if i == len(Y):
            Y.append(l[-1] + b[-1] + s[-m])

        l.append(alpha * (Y[i] - s[i-m]) + (1 - alpha) * (l[i] + b[i]))

```

```
28          b.append(beta * (l[i + 1] - l[i]) + (1 - beta) * b[i])
29
30          s.append(gamma * (Y[i] - l[i] - b[i]) + (1 - gamma) * s[i-m])
31
32          forecasted.append(l[i] + b[i] + s[i-m])
33
34
35      rmse = sqrt(sum([(m - n + 0.) ** 2 for m, n in zip(Y[:-fc], y[:-fc-1])]) / len(Y[:-fc]))
36
37      return forecasted, Y[-fc:], l, b, s, rmse
```

- 函数的参数中 x 是原数据 y_t ， m 是周期， fc 是要预测的时间长度
- 2 到 5 行 新建变量存储结果
- 7到8 行 确定 α β γ
- 10到17行，确定 l_0 b_0 和 $\bar{s}_m$, 如果没有传参数，用上面提到的 linearOptimization 函数计算这三个参数。
- 19到20 行 存储 forecast 的结果以及对应的 rmse
- 22到 32 行 迭代公式进行预测
    - 23 行，如果 i 等于 Yt 的个数，那么已经过了最后一个 y_t , 根据最后一个预测的 l, b 和 s[-m] 生成新的 Y 值作为 y_{t+1} .
    - 30 行 32 行的 s[i-m] 指的是上一周期的相同时间的 the seasonal index 的值
    - 26 行 的 s[-m] 指的是 s_{t+1-m} 因为 s[-1] 指的是 s_t ， s[-1-m] 是 s_{t-m}
- 35行 根据公式计算 rmse.
- zip(a,b) 函数将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表
- 返回值中
    - forecasted 是包含预测的smooth结果，
    - Y[-fc:] 预测的结果

```
a = [1,2,3]
b = [4,5,6]
zipped = zip(a,b)     # 打包为元组的列表
[(1, 4), (2, 5), (3, 6)]
```

statsmodels 实现

- holt winter 有 additive 和 multiplicative 两种

Multiplicative Holt-Winters smoothing

Model

$$
\begin{aligned}
l_t &= \alpha(y_t/S_{t-M}) + (1-\alpha)(l_{t-1}+b_{t-1}),\\
b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1},\\
S_t &= \gamma(y_t/l_t) + (1-\gamma)S_{t-M},
\end{aligned}
$$

$$
y_{t+1} = (l_t + b_t) \times S_{t+1-M} + \varepsilon_{t+1}, \qquad \varepsilon_{t+1} \sim N(0, \sigma^2).
$$

We can chose the parameters α, β and γ by minimising

$$
SSE = \sum_{t=1}^{n} (y_t - (l_{t-1} + b_{t-1})S_{t-M})^2
$$

ExponentialSmoothing 函数

专门用来做Holt Winter’s Exponential Smoothing 的库函数

(endog, trend=None, damped=False, seasonal=None, seasonal_periods=None）

- endog (array-like) – **Time series**
- trend ({"add", "mul", "additive", "multiplicative", None}, optional) – **Type of trend component.**
- damped (bool, optional) – **Should the trend component be damped.**
- seasonal ({"add", "mul", "additive", "multiplicative", None}, optional) – **Type of seasonal component**.
- seasonal_periods (int, optional) – **周期**

```
# additive
fit1 = ExponentialSmoothing(y, seasonal_periods = 12, trend = 'add', seasonal = 'add').fit()

# multiplicative
fit2 = ExponentialSmoothing(y, seasonal_periods = 12, trend = 'add', seasonal = 'mul').fit()
```

方法：

`fit` () 用来生成 smoothed 结果。返回一个 HoltWintersResults 对象

class HoltwintersResults

ExponentialSmoothing.fit() 函数返回的结果，也就是Holt Winter's Exponential Smoothing 的结果

属性：

- **params** smoothing 的所有参数
  - "α","β","ϕ","γ", "l_0","b_0,
- **fittedvalues** 拟合的结果
- **sse** 我们用 sse 判断是用 additive 还是 multiplicative
- **level** 构成 $\hat{y}_t$ 的 level 部分也就是 l_t
- **slope** b_t
- **season** $\hat{s}_t$

方法：

`forecast`(fc) fc 是预测的时间长度

返回 预测的结果组成的 array

Tutorial 之后都是 画图 和表格的形式展示 smoothing 的结果和预测的结果，根据 tutorial 上的代码讲

补充

`numpy.c_`

np.c_是按行连接两个矩阵

```
a = np.array([[1, 2, 3],[7,8,9]])
b=np.array([[4,5,6],[1,2,3]])
c=np.c_[a,b]

c
Out[7]:
array([[1, 2, 3, 4, 5, 6],
       [7, 8, 9, 1, 2, 3]])
```