TUT 8 9 ARIMA

概念

ARIMA

自回归综合移动平均（Auto-Regressive Integrated Moving Averages）

前提假设

TS 需要是 stationarity 的

组成

ARIMA有三个分量：AR(**A**uto**r**egressive)、I(差分项)和MA(**M**oving **A**verage model)

- AR项是指用于预测下一个值的过去值。AR项由ARIMA中的参数‘p’定义。“p”的值是由PACF图确定的。
- MA项定义了预测未来值时过去预测误差的数目。ARIMA中的参数‘q’代表MA项。ACF图用于识别正确的‘q’值。
- 差分顺序规定了对序列执行差分操作的次数，对数据进行差分操作的目的是使之保持平稳。像ADF和KPSS这样的测试可以用来确定序列是否是平稳的，并有助于识别d值。

ACF 和 PACF

ACF

定义：度量时间序列中每隔 k 个时间单位（yt 和 yt–k）的观测值之间的相关性

$\rho_k = Corr(Y_t, Y_{t+(or-)k})$

画 ACF 图

smt.graphics.tsa.plot_acf

- data 分析的数据
- lags = 30 与 t 相差多少个时间单位
- alpha = 0.05 置信区间

```
smt.graphics.tsa.plot_acf(data, lags = 30, alpha = 0.05)
```

判断是否 stationary

如果 ACF 图的值随着 lag 的增加迅速降低至 0 ，则原数据是 stationary 的(和 t 无关)

推导在L7 P31 到 P34

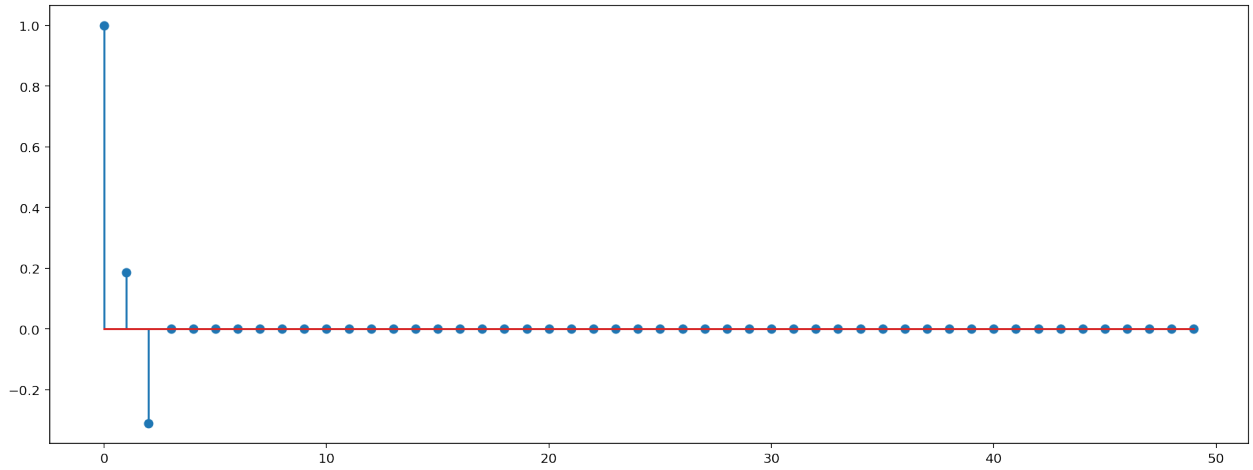
假设 数据是 stationary， AR(1) 的 ACF

$$\rho_k = Cov(Y_t, Y_{t-k})/Var(Y_t) = \phi_1^k$$

判断 MA 的阶数 q

一般来说是看 ACF 图，如果 ACF 图 中有 **the cutting off value** 就是

换句话说就是在 lag 等于这个值之后的 ACF 值都在区间（接近于0）内。



the cutting off value = 2

PACF

部分自相关函数是在去除其他变量的影响后(yt–1, yt–2, ..., yt–k–1)的存在之后，以k个时间单位分隔的时间序列（yt和yt-k）的观测值之间的相关性。

$$\hat{X}_{t-1,1} = \alpha_1 X_{t-1} + \ldots + \alpha_h X_{t-h}.$$

PACF就是上式中的系数，alpha[h]就是X(t)和X(t-h)的偏相关系数。

画 PACF

smt.graphics.tsa.plot_pacf

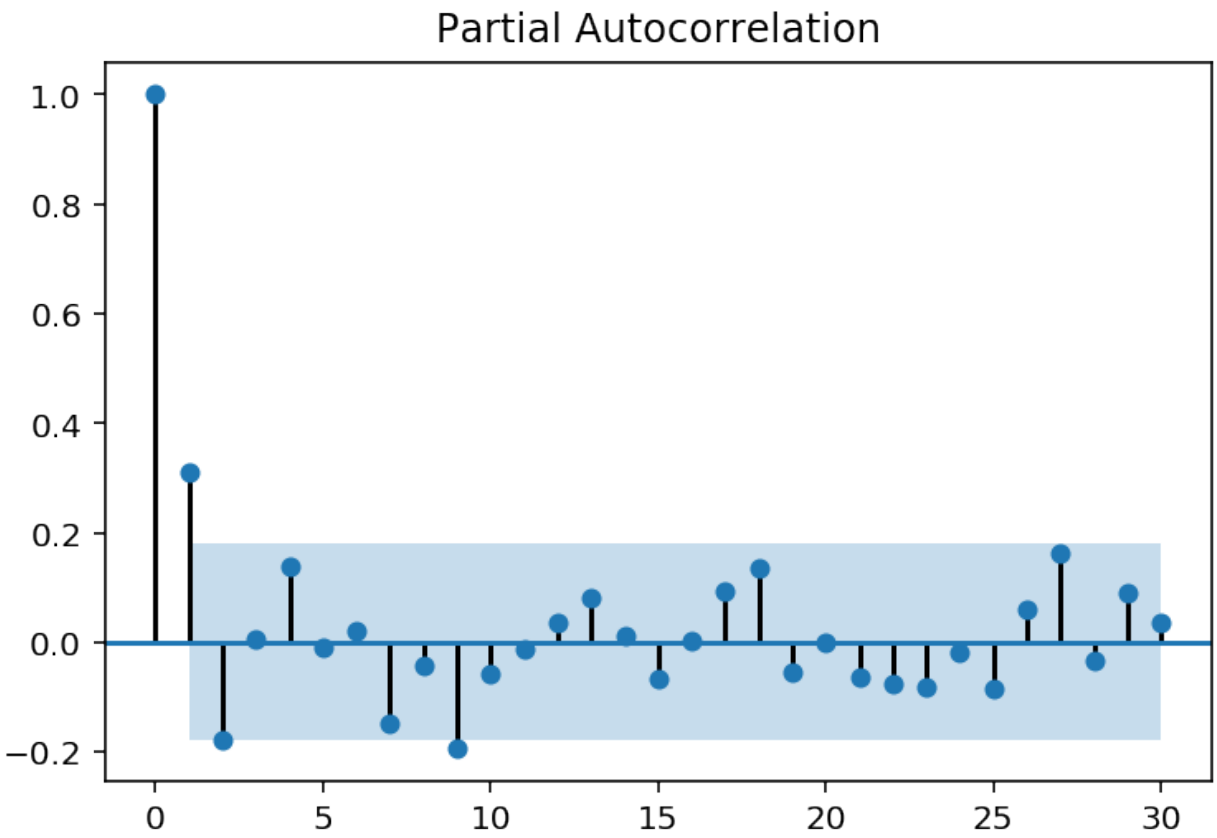
- data 分析的数据
- lags = 30 与 t 相差多少个时间单位
- alpha = 0.05 置信区间

```
smt.graphics.tsa.plot_pacf(data, lags=30, alpha = 0.05);
```

判断 AR 的阶数p

一般来说是看 PACF 图，如果 PACF 图 中有 **the cutting off value** 就是

换句话说就是在 lag 等于这个值之后的 PACF 值都在区间（接近于0）内。



the cutting off value = 1

AR(Auto Regression) 模型

描述当前值与历史值之间的关系，用变量自身的历史时间数据对自身进行预测。

AR(p):

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \ldots + \alpha_p X_{t-p} + \varepsilon_t$$

自回归模型首先需要确定一个阶数p，表示用几期的历史值来预测当前值，阶数可以通过看 PACF 图 和 AIC/BIC 得到

MA() 模型

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

y_t值可以被认为是过去几个误差的加权移动平均值

生成 AR MA 模型数据

随机数种子

```
np.random.seed(1)
```

设置参数

```
arparams = np.array([0.9])
maparams = np.array([0.6, -0.5])

ar = np.r_[1, -arparams]
ma = np.r_[1, maparams]
zero_lag = np.array([1])

c = 0
sigma2 = 1
```

- 第四行 ar 的参数是负数的原因是

  statmodel 的 armaprocess 里的定义

$$y_t = c - \phi_1 y_{t-1} - \cdots - \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

生成数据

ArmaProcess

生成 AR(p) 数据

```
sm.tsa.arima_process.ArmaProcess(ar = ar, ma = zero_lag)
```

- Zero_lag 是因为

> Both the AR and MA components must include the coefficient on the zero-lag.In almost all cases these values should be 1

生成 MA(q) 数据

```
ma_model = sm.tsa.arima_process.ArmaProcess(ar = zero_lag, ma = ma)
```

生成 ARMA(p,q) 数据

```
arma_process = sm.tsa.arima_process.ArmaProcess(ar, ma)
```

判断生成数据的 stationary 和 invertible

```
arma_process.isstationary
arma_process.isinvertible
```

完整的 ARIMA 过程

步骤

1. 加载数据：构建模型的第一步当然是加载数据集。
2. 预处理：根据数据集定义预处理步骤。包括创建时间戳、日期/时间列转换为d类型、序列单变量化等。
3. 序列平稳化：为了满足假设，应确保序列平稳。这包括检查序列的平稳性和执行所需的转换。
4. 确定d值：为了使序列平稳，执行差分操作的次数将确定为d值。
5. 创建ACF和PACF图：这是ARIMA实现中最重要的一步。用ACF PACF图来确定ARIMA模型的输入参数。
6. 确定p值和q值：从上一步的ACF和PACF图中读取p和q的值。或者用 AIC 和 BIC
7. 拟合ARIMA模型：利用我们从前面步骤中计算出来的数据和参数值，拟合ARIMA模型。
8. 在验证集上进行预测：预测未来的值。
9. 计算误差：通过检查RMSE值来检查模型的性能，用验证集上的预测值和实际值检查RMSE值。

步骤3 log transform

为了减少最高数据和最低数据的距离，使序列更平缓

```
ts_log = np.log(ts)

plt.figure()
plt.plot(ts_log)
plt.title("Air passenger data (log)");
```

步骤4 差分 确定 i

Box and Jenkins advocate difference transforms to achieve stationarity, e.g

$$\Delta Y_t = Y_t - Y_{t-1}$$

$$\Delta^2 Y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-1}$$

为了使 TS 达到 stationary 我们用差分变换

tutorial :

```
ts_log_diff = ts_log - ts_log.shift()
ts_log_diff.dropna(inplace=True)
```

或者调库 (Series.diff)[https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.diff.html#pandas-series-diff]

```
s = s.diff()
s.dropna(inplace=True)
```

然后通过 画 ACF 或者 之前学的AD Fuller test 判断是否 stationary，如果不 stationary继续做2阶 3阶差分

i 的值就是差分的阶数

步骤5 画 ACF 和 PACF 确定 p 和 q

```
smt.graphics.tsa.plot_acf(ts_log_diff, lags=30, alpha = 0.05)
smt.graphics.tsa.plot_pacf(ts_log_diff, lags=30, alpha = 0.05)
```

步骤6 拟合ARIMA模型

statsmodels.tsa.arima_model.ARMA(data , **order**) 根据选定的ARIMA oder(p,i,q) 和数据 拟合 AR 和 MA 的参数 ϕ 和 θ

```
model_y2 = sm.tsa.arima_model.ARMA(y2, (1,1,2)).fit(trend = 'nc')
print("Estimated Model Parameters: " + str(model_y2.params))

fitted = model_y2.predict(typ = 'levels', dynamic = False)

model_y2.plot_predict(dynamic = False)
plt.show()
```

- ARMA.fit() 拟合
  - **trend** (*str {'c','nc'}*) – Whether to include a constant or not. ‘c’ includes constant, ‘nc’ no constant.
  - **disp** If True, convergence information is printed. For the default l_bfgs_b solver, disp controls the frequency of the output during the iterations. **disp < 0 means no output in this case.**
  - 返回 ARMAResults 对象
- ARMAResults
  - `predict()` 生成拟合结果
  - plot_predict 画出拟合图像
    - *start* 起点
    - *end* 终点

- *dynamic* The dynamic keyword affects in-sample prediction. If dynamic is False, then the in-sample lagged values are used for prediction. If dynamic is True, then in-sample forecasts are used in place of lagged dependent variables. The first forecasted value is start.

步骤7 计算RSS

```
residuals = pd.DataFrame(model_y2.resid)

plt.figure()
plt.plot(residuals)
plt.title('ARIMA(2,1,0) RSS: %.4f'% sum((results_AR.resid.values)**2))
```

步骤8 在验证集上进行预测

ARMAResults.forecast

- steps 预测的时间长度
- **alpha** 置信区间

```
forecast, stderr, conf_int = model_y2.forecast(steps = 10)
plt.figure()
plt.plot(forecast)
```