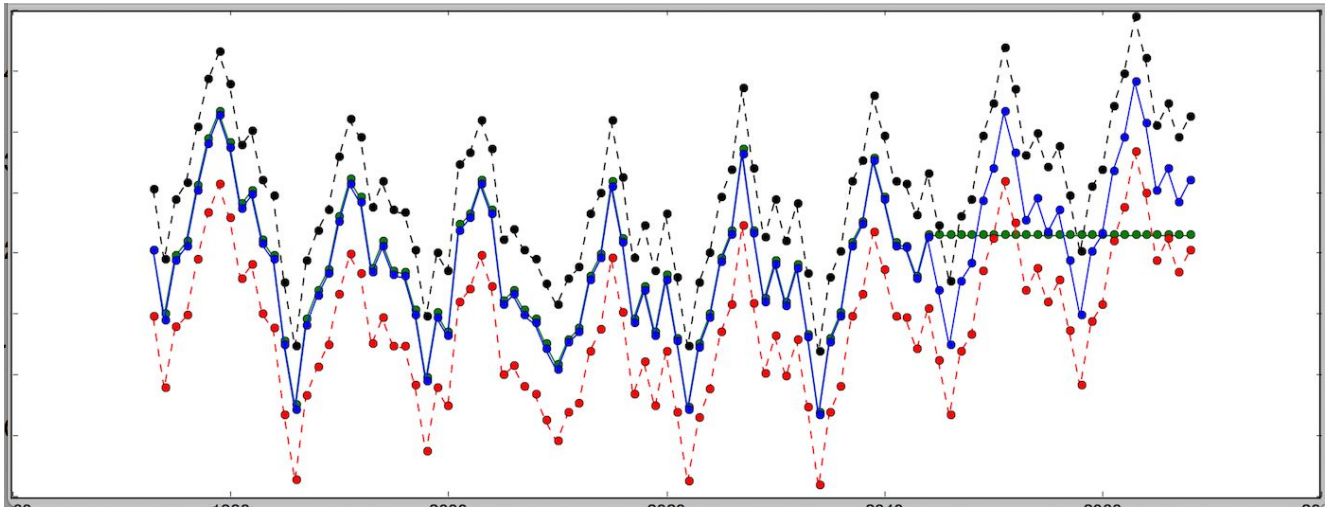
Suavizamiento exponencial

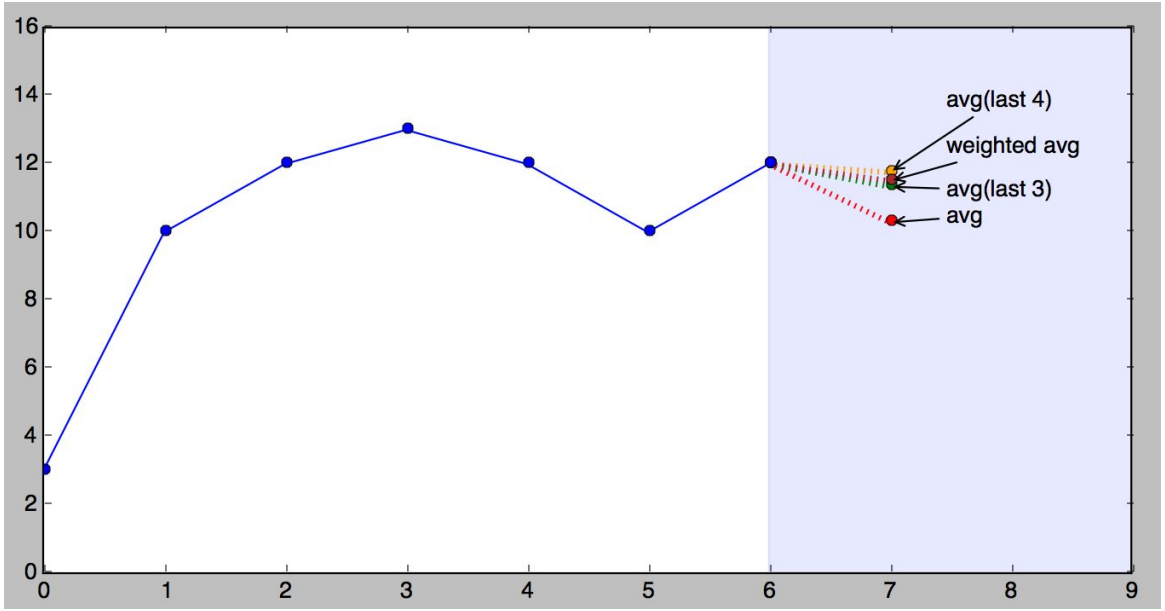
Holt Winters

Introducción

El suavizamiento exponencial es uno de los muchos métodos o algoritmos que se pueden usar para pronosticar puntos de datos en una serie, siempre que la serie sea "estacional", es decir, repetitiva durante un cierto período.



Tipos de predicción



Suavizamiento exponencial sencillo

Fórmula:

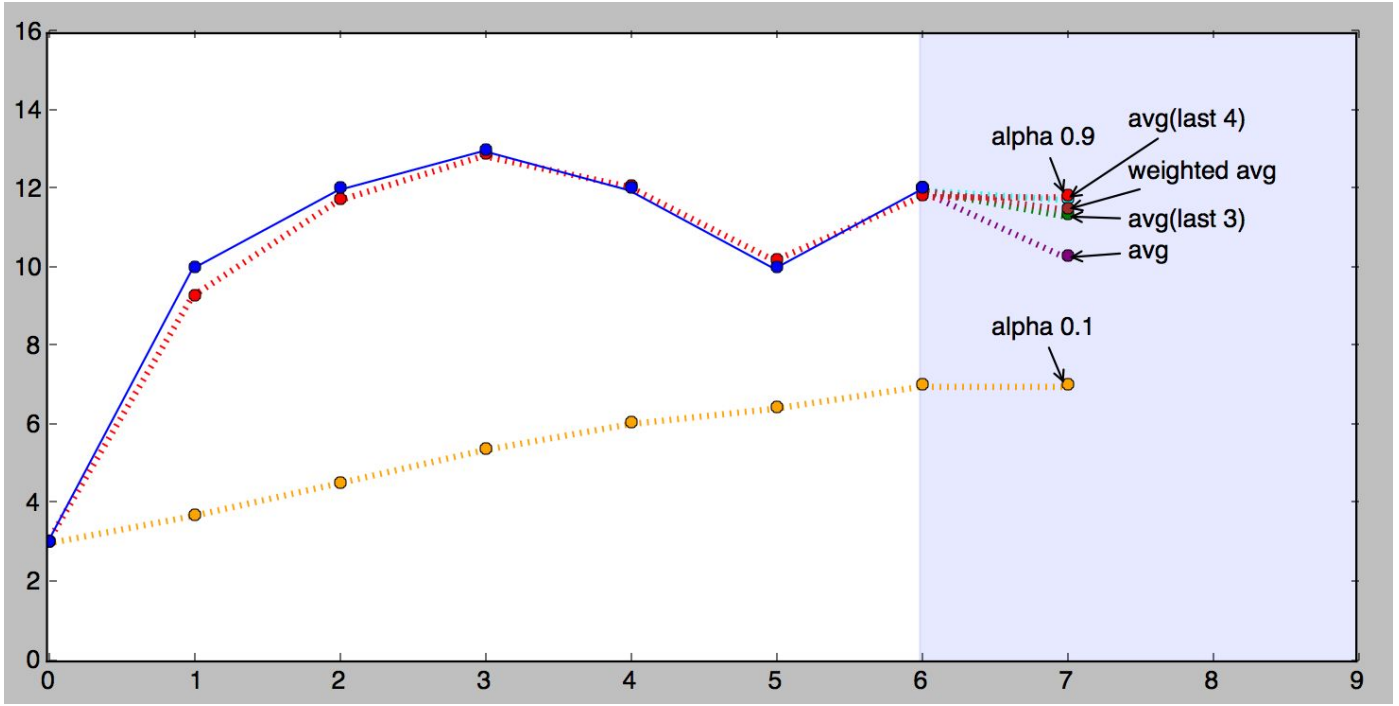
$$\hat{y}_x = \alpha \cdot y_x + (1 - \alpha) \cdot \hat{y}_{x-1}$$

Implementación:

```python
# given a series and alpha, return series of smoothed points
def exponential_smoothing(series, alpha):
    result = [series[0]] # first value is same as series
    for n in range(1, len(series)):
        result.append(alpha * series[n] + (1 - alpha) * result[n-1])
    return result

# >>> exponential_smoothing(series, 0.1)
# [3, 3.7, 4.53, 5.377, 6.0393, 6.43537, 6.991833]
# >>> exponential_smoothing(series, 0.9)
# [3, 9.3, 11.73, 12.873000000000001, 12.0873, 10.20873, 11.820873]
```

Gráfica


16
14
12
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
9
alpha 0.9
avg(last 4)
weighted avg
avg(last 3)
avg
alpha 0.1

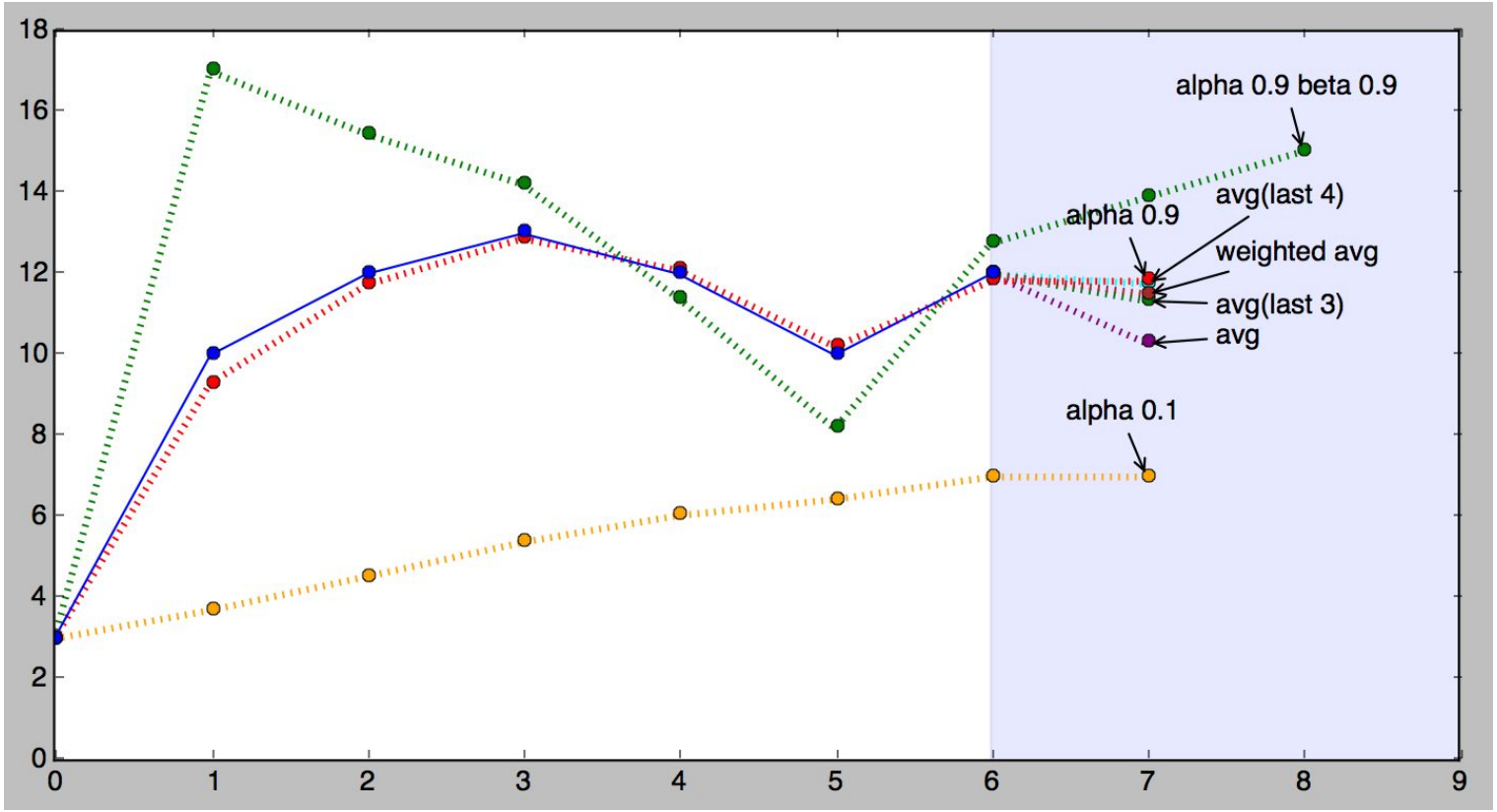
Suavizamiento exponencial doble

$$\ell_x = \alpha y_x + (1-\alpha)(\ell_{x-1} + b_{x-1}) \qquad \text{level}$$
$$b_x = \beta(\ell_x - \ell_{x-1}) + (1-\beta)b_{x-1} \qquad \text{trend}$$
$$\hat{y}_{x+1} = \ell_x + b_x \qquad \text{forecast}$$

```python
# given a series and alpha, return series of smoothed points
def double_exponential_smoothing(series, alpha, beta):
    result = [series[0]]
    for n in range(1, len(series)+1):
        if n == 1:
            level, trend = series[0], series[1] - series[0]
        if n >= len(series): # we are forecasting
          value = result[-1]
        else:
          value = series[n]
        last_level, level = level, alpha*value + (1-alpha)*(level+trend)
        trend = beta*(level-last_level) + (1-beta)*trend
        result.append(level+trend)
    return result

# >>> double_exponential_smoothing(series, alpha=0.9, beta=0.9)
# [3, 17.0, 15.45, 14.210500000000001, 11.396044999999999, 8.183803049999998, 12.753698384
```

Gráfica


alpha 0.9 beta 0.9
avg(last 4)
alpha 0.9
weighted avg
avg(last 3)
avg
alpha 0.1
0
2
4
6
8
10
12
14
16
18
0
1
2
3
4
5
6
7
8
9

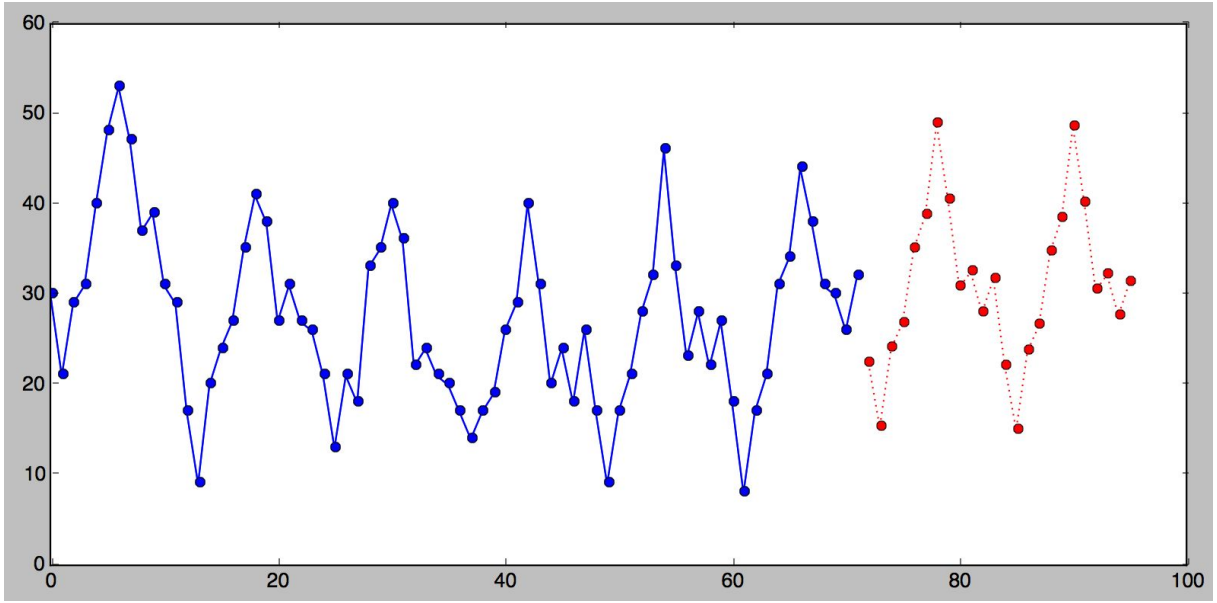
Suavizamiento exponencial triple

$$\ell_x = \alpha(y_x - s_{x-L}) + (1-\alpha)(\ell_{x-1} + b_{x-1}) \quad \text{level}$$

$$b_x = \beta(\ell_x - \ell_{x-1}) + (1-\beta)b_{x-1} \quad \text{trend}$$

$$s_x = \gamma(y_x - \ell_x) + (1-\gamma)s_{x-L} \quad \text{seasonal}$$

$$\hat{y}_{x+m} = \ell_x + mb_x + s_{x-L+1+(m-1) mod L} \quad \text{forecast}$$



Implementación

```python
def triple_exponential_smoothing(series, slen, alpha, beta, gamma, n_preds):
    result = []
    seasonals = initial_seasonal_components(series, slen)
    for i in range(len(series)+n_preds):
        if i == 0: # initial values
            smooth = series[0]
            trend = initial_trend(series, slen)
            result.append(series[0])
            continue
        if i >= len(series): # we are forecasting
            m = i - len(series) + 1
            result.append((smooth + m*trend) + seasonals[i%slen])
        else:
            val = series[i]
            last_smooth, smooth = smooth, alpha*(val-seasonals[i%slen]) + (1-alpha)*(smoot
            trend = beta * (smooth-last_smooth) + (1-beta)*trend
            seasonals[i%slen] = gamma*(val-smooth) + (1-gamma)*seasonals[i%slen]
            result.append(smooth+trend+seasonals[i%slen])
    return result

# # forecast 24 points (i.e. two seasons)
# >>> triple_exponential_smoothing(series, 12, 0.716, 0.029, 0.993, 24)
# [30, 20.34449316666667, 28.410051892109554, 30.438122252647577, 39.466817731253066, ...
```