

50ETF看涨期权 Delta套期保值

任选一个我国ETF看涨期权，截取2026年4个不同时间段的delta、期权价格及对应标的的价格。假设最开始出售1份该看涨期权（1份=10000份标的），最后一个时间点平仓，对该期权做套期保值，计算套保效果。

数据来源：tushare，无风险利率取2%连续复利。

```
In [1]: import tushare as ts
import pandas as pd
import numpy as np
from scipy.stats import norm
from scipy.optimize import brentq

pd.set_option('display.float_format', lambda x: f'{x:.4f}')

# tushare
ts.set_token('d4ade4dc2337bb63733c3ec90deb49bfe013ac6b81452bdba73641e7')
pro = ts.pro_api()

# BS公式，用来算delta
def bs_d1(S, K, T, r, sigma):
    return (np.log(S / K) + (r + sigma**2 / 2) * T) / (sigma * np.sqrt(T))

def bs_call_price(S, K, T, r, sigma):
    d1 = bs_d1(S, K, T, r, sigma)
    d2 = d1 - sigma * np.sqrt(T)
    return S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)

def call_delta(S, K, T, r, sigma):
    # 看涨期权的delta就是N(d1)
    return norm.cdf(bs_d1(S, K, T, r, sigma))

def calc_iv(S, K, T, r, mkt_price):
    # 从市场价格反推隐含波动率
    try:
        return brentq(lambda sig: bs_call_price(S, K, T, r, sig) - mkt_price, 0.01,
            except:
                return 0.25

r_f = 0.02 # 无风险利率
unit = 10000 # 一份合约对应10000份标的
```

选取期权合约

从上交所50ETF看涨期权里，选一个到期日在2026年6-9月、执行价接近当前50ETF价格的平值合约。

```
In [2]: # 拿到上交所所有看涨期权
opts = pro.opt_basic(exchange='SSE', call_put='C',
                     fields='ts_code,name,exercise_price,list_date,delist_date,per_unit')

# 只要50ETF的, 到期日在2026年6-9月
opts = opts[opts['per_unit'] == 10000].copy()
opts['exp_dt'] = pd.to_datetime(opts['delist_date'])
opts = opts[(opts['exp_dt'] >= '2026-06-01') & (opts['exp_dt'] <= '2026-09-30')]
opts = opts.sort_values('exercise_price').reset_index(drop=True)
print(f'筛选出{len(opts)}个合约')

# 看一下50ETF现在什么价格
etf_latest = pro.fund_daily(ts_code='510050.SH', start_date='20260401', end_date='20260401')
etf_latest = etf_latest.sort_values('trade_date', ascending=False)
S_now = etf_latest.iloc[0]['close']
print(f'50ETF最新价: {S_now}')

# 选执行价最接近现价的
opts['diff'] = abs(opts['exercise_price'] - S_now)
sel = opts.sort_values('diff').iloc[0]

opt_code = sel['ts_code']
opt_name = sel['name']
K = sel['exercise_price']
exp_date = sel['delist_date']

print(f'\n选定合约: {opt_name}')
print(f'代码: {opt_code}')
print(f'执行价K = {K}')
print(f'到期日: {exp_date}')
```

筛选出144个合约

50ETF最新价: 2.958

选定合约: 华夏上证50ETF期权2609认购2.95

代码: 10010973.SH

执行价K = 2.95

到期日: 20260923

第一步: 获取4个时间点的Delta、期权价格、标的价格

从2026年1月到4月, 每月选一个交易日, 分别获取当天的:

- 50ETF收盘价 (标的价格S)
- 期权收盘价 (c)
- 通过BS公式反推隐含波动率, 再算出Delta

```
In [3]: # 选4个日期, 1月到4月各一个
dates = ['20260106', '20260205', '20260306', '20260408']

# 拉期权和ETF的日线数据
opt_daily = pro.opt_daily(ts_code=opt_code, start_date='20260101', end_date='20260408')
```

```

opt_daily['trade_date'] = opt_daily['trade_date'].astype(str)
opt_daily = opt_daily.sort_values('trade_date').reset_index(drop=True)

etf_daily = pro.fund_daily(ts_code='510050.SH', start_date='20260101', end_date='20
etf_daily['trade_date'] = etf_daily['trade_date'].astype(str)
etf_daily = etf_daily.sort_values('trade_date').reset_index(drop=True)

print(f'获取到期权行情 {len(opt_daily)} 条, ETF行情 {len(etf_daily)} 条')

# 对每个目标日期, 找最近的交易日, 算隐含波动率和delta
rows = []
for d in dates:
    tmp = opt_daily[opt_daily['trade_date'] >= d]
    if len(tmp) == 0:
        continue
    orow = tmp.iloc[0]
    tdate = orow['trade_date']
    c = orow['close']

    etmp = etf_daily[etf_daily['trade_date'] == tdate]
    if len(etmp) == 0:
        continue
    S = etmp.iloc[0]['close']

    T = (pd.to_datetime(exp_date) - pd.to_datetime(tdate)).days / 365.0
    iv = calc_iv(S, K, T, r_f, c)
    delta = call_delta(S, K, T, r_f, iv)

    rows.append({'日期': tdate, '标的价格S': round(S, 4), '期权价格c': round(c, 4),
                'T(年)': round(T, 4), '隐含波动率': round(iv, 4), 'Delta': round(de

df = pd.DataFrame(rows)

print(f'\n合约: {opt_name} ({opt_code})')
print(f'执行价 K = {K}, 到期日 = {exp_date}')
display(df)

```

获取到期权行情 44 条, ETF行情 62 条

合约: 华夏上证50ETF期权2609认购2.95 (10010973.SH)

执行价 K = 2.95, 到期日 = 20260923

	日期	标的价格S	期权价格c	T(年)	隐含波动率	Delta
0	20260129	3.1960	0.3636	0.6493	0.1923	0.7511
1	20260205	3.1370	0.3002	0.6301	0.1720	0.7293
2	20260306	3.0680	0.2371	0.5507	0.1696	0.6780
3	20260408	2.9780	0.1506	0.4603	0.1517	0.5920

第二步: 计算每个时间点应买入/卖出的股票数量

我们卖出了1份看涨期权，期权的delta是正的，所以我们的delta暴露是负的。为了对冲，需要买入 $\Delta \times 10000$ 股标的ETF，让组合的delta接近0。

每次调仓时，根据新的delta调整持股数。最后一个时间点全部平仓（卖掉所有股票，买回期权）。

```
In [4]: held = 0 # 当前持股
recs = []

for i, row in df.iterrows():
    S = row['标的价格S']
    c = row['期权价格c']
    delta = row['Delta']
    last = (i == len(df) - 1)

    # 最后一天平仓，目标持股为0
    target = 0 if last else round(delta * unit)
    chg = target - held

    if chg > 0:
        op = f'买入{int(chg)}股'
    elif chg < 0:
        op = f'卖出{int(-chg)}股'
    else:
        op = '不动'

    recs.append({'日期': row['日期'], 'S': S, 'c': c, 'Delta': delta,
                '目标持股': int(target), '变动': int(chg), '操作': op})
    held = target

df_hedge = pd.DataFrame(recs)

print('每个时间点的操作（目标持股 = Delta x 10000）:')
display(df_hedge)

for _, r in df_hedge.iterrows():
    print(f" {r['日期']}: Delta={r['Delta']:.4f}, 应持{r['目标持股']}股, {r['操作']}")
```

每个时间点的操作（目标持股 = $\Delta \times 10000$ ）：

	日期	S	c	Delta	目标持股	变动	操作
0	20260129	3.1960	0.3636	0.7511	7511	7511	买入7511股
1	20260205	3.1370	0.3002	0.7293	7293	-218	卖出218股
2	20260306	3.0680	0.2371	0.6780	6780	-513	卖出513股
3	20260408	2.9780	0.1506	0.5920	0	-6780	卖出6780股

20260129: Delta=0.7511, 应持7511股, 买入7511股
20260205: Delta=0.7293, 应持7293股, 卖出218股
20260306: Delta=0.6780, 应持6780股, 卖出513股
20260408: Delta=0.5920, 应持0股, 卖出6780股

第三步：计算每个时间点的操作成本和累计成本

每次买卖股票都有现金流出/流入，同时因为买股票的钱一部分来自卖期权收入、一部分需要借款，所以要算借款产生的利息。利息用连续复利公式：利息 = 借款 $\times (e^{(r \times \text{天数} / 365)} - 1)$

```
In [5]: cost_rows = []
cum_interest = 0
borrow = 0
prev_dt = None

for i, row in df_hedge.iterrows():
    S = row['S']
    c = row['c']
    chg = row['变动']
    cur_dt = pd.to_datetime(df.iloc[i]['日期'])
    first = (i == 0)
    last = (i == len(df_hedge) - 1)

    # 算这段时间的利息
    interest = 0
    if prev_dt is not None:
        days = (cur_dt - prev_dt).days
        interest = borrow * (np.exp(r_f * days / 365) - 1)
        cum_interest += interest
        borrow = borrow * np.exp(r_f * days / 365)

    # 股票买卖的钱
    stock_cost = chg * S

    # 期权现金流
    if first:
        opt_cf = c * unit # 卖期权收到的钱
        borrow = stock_cost - opt_cf # 净借款 = 买股票花的 - 卖期权收的
    elif last:
        opt_cf = -c * unit # 买回期权
        borrow += stock_cost
    else:
        opt_cf = 0
        borrow += stock_cost

    cost_rows.append({
        '日期': df.iloc[i]['日期'], 'S': S, 'c': c,
        'Delta': row['Delta'], '持股': row['目标持股'], '变动': int(chg),
        '股票交易额': round(stock_cost, 2), '期权现金流': round(opt_cf, 2),
        '利息': round(interest, 2), '累计利息': round(cum_interest, 2),
        '净借款': round(borrow, 2)
    })
    prev_dt = cur_dt

df_cost = pd.DataFrame(cost_rows)

print('成本明细表:')
display(df_cost)
```

```
print(f'累计借款利息: {cum_interest:.2f}元')
```

成本明细表:

	日期	S	c	Delta	持股	变动	股票交易额	期权现金流	利息	累
0	20260129	3.1960	0.3636	0.7511	7511	7511	24005.1600	3636.0000	0.0000	0.
1	20260205	3.1370	0.3002	0.7293	7293	-218	-683.8700	0.0000	7.8100	7.
2	20260306	3.0680	0.2371	0.6780	6780	-513	-1573.8800	0.0000	31.3200	39.
3	20260408	2.9780	0.1506	0.5920	0	-6780	-20190.8400	-1506.0000	32.8500	71.

累计借款利息: 71.98元

第四步: 对比套保和不套保的效果

分三种情况来看:

1. **不套保**: 只看卖出期权的损益, 即 (卖出价 - 买回价) x 10000
2. **套保, 不算利息**: 期权损益 + 股票损益
3. **套保, 算利息**: 期权损益 + 股票损益 - 累计利息

```
In [6]: c0 = df.iloc[0]['期权价格c']
cT = df.iloc[-1]['期权价格c']
S0 = df.iloc[0]['标的价格S']
ST = df.iloc[-1]['标的价格S']

# 期权损益: 卖出价-买回价
opt_pnl = (c0 - cT) * unit

# 股票损益: 每段持仓 * 价格变动
stk_pnl = 0
for i in range(len(df) - 1):
    n = cost_rows[i]['持股']
    ds = df.iloc[i+1]['标的价格S'] - df.iloc[i]['标的价格S']
    stk_pnl += n * ds
    print(f' 第{i+1}段: 持{n}股, 标的从{df.iloc[i]["标的价格S"]}变到{df.iloc[i+1]["标的价格S"]}')

print(f'\n期权损益 = ({c0} - {cT}) x {unit} = {opt_pnl:.2f}元')
print(f'股票总损益 = {stk_pnl:.2f}元')

print()
print('='*50)
print(' 情况1: 不套保')
print('='*50)
print(f' 只有期权损益: {opt_pnl:.2f}元')

print()
print('='*50)
print(' 情况2: 套保 (不考虑利息)')
print('='*50)
total1 = opt_pnl + stk_pnl
```

```

print(f'  期权损益:  {opt_pnl:.2f}')
print(f'  股票损益:  {stk_pnl:.2f}')
print(f'  合计:      {total1:.2f}元')

print()
print('='*50)
print('  情况3: 套保 (考虑利息, r=2%)')
print('='*50)
total2 = total1 - cum_interest
print(f'  期权损益:  {opt_pnl:.2f}')
print(f'  股票损益:  {stk_pnl:.2f}')
print(f'  利息支出:  -{cum_interest:.2f}')
print(f'  合计:      {total2:.2f}元')

print()
print('='*50)
print('  对比汇总')
print('='*50)
print(f'  不套保损益:      {opt_pnl:.2f}元')
print(f'  套保损益(不含利息): {total1:.2f}元')
print(f'  套保损益(含利息):  {total2:.2f}元')

```

第1段: 持7511股, 标的从3.196变到3.137, 损益=-443.15

第2段: 持7293股, 标的从3.137变到3.068, 损益=-503.22

第3段: 持6780股, 标的从3.068变到2.978, 损益=-610.20

期权损益 = $(0.3636 - 0.1506) \times 10000 = 2130.00$ 元

股票总损益 = -1556.57元

=====
情况1: 不套保
=====

只有期权损益: 2130.00元

=====
情况2: 套保 (不考虑利息)
=====

期权损益: 2130.00
股票损益: -1556.57
合计: 573.43元

=====
情况3: 套保 (考虑利息, r=2%)
=====

期权损益: 2130.00
股票损益: -1556.57
利息支出: -71.98
合计: 501.45元

=====
对比汇总
=====

不套保损益: 2130.00元
套保损益(不含利息): 573.43元
套保损益(含利息): 501.45元

小结

这次用50ETF的看涨期权做了delta套保的练习，选了4个时间点来看。

从数据上看，这段时间标的价格整体是往下走的(从3.196降到2.978)，因为我们卖的是看涨期权，标的跌了期权价格也跌了，所以卖期权这边是赚钱的。但是做delta套保买了股票，股票跟着跌了就亏了。

不过delta套保的意义不在于多赚钱，而是降低风险。如果标的价格大幅上涨，不套保的话期权空头会亏很多，但套保后股票端能对冲掉大部分损失。本次标的恰好是下跌的，所以看起来不套保赚得更多，但这只是事后来看，事前我们并不知道方向。

利息方面，买股票需要借钱，总共利息才几十块，相对总金额来说很小，基本可以忽略。

另外因为只在4个时间点调仓，中间delta变化了没有及时调整，存在gamma带来的误差，如果调仓更频繁，套保效果会更好。