22.neuro-evolution-novelty-search-agent

September 29, 2021

[1]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
import seaborn as sns
sns.set()
```

[2]:

```
df = pd.read_csv('../dataset/GOOG-year.csv')
df.head()
```

[2]:

```
         Date        Open        High         Low       Close   Adj Close  \
0  2016-11-02  778.200012  781.650024  763.450012  768.700012  768.700012
1  2016-11-03  767.250000  769.950012  759.030029  762.130005  762.130005
2  2016-11-04  750.659973  770.359985  750.560974  762.020020  762.020020
3  2016-11-07  774.500000  785.190002  772.549988  782.520020  782.520020
4  2016-11-08  783.400024  795.632996  780.190002  790.510010  790.510010

    Volume
0  1872400
1  1943200
2  2134800
3  1585100
4  1350800
```

[3]:

```
close = df.Close.values.tolist()
initial_money = 10000
window_size = 30
skip = 1

novelty_search_threshold = 6
novelty_log_maxlen = 1000
backlog_maxsize = 500
novelty_log_add_amount = 3
```

[4]:

```
class neuralnetwork:
    def __init__(self, id_, hidden_size = 128):
        self.W1 = np.random.randn(window_size, hidden_size) / np.
↪sqrt(window_size)
        self.W2 = np.random.randn(hidden_size, 3) / np.sqrt(hidden_size)
        self.fitness = 0
        self.last_features = None
        self.id = id_

def relu(X):
    return np.maximum(X, 0)

def softmax(X):
    e_x = np.exp(X - np.max(X, axis=-1, keepdims=True))
    return e_x / np.sum(e_x, axis=-1, keepdims=True)

def feed_forward(X, nets):
    a1 = np.dot(X, nets.W1)
    z1 = relu(a1)
    a2 = np.dot(z1, nets.W2)
    return softmax(a2)
```

[8]:

```
class NeuroEvolution:
    def __init__(self, population_size, mutation_rate, model_generator,
                state_size, window_size, trend, skip, initial_money):
        self.population_size = population_size
        self.mutation_rate = mutation_rate
        self.model_generator = model_generator
        self.state_size = state_size
        self.window_size = window_size
        self.half_window = window_size // 2
        self.trend = trend
        self.skip = skip
        self.initial_money = initial_money
        self.generation_backlog = []
        self.novel_backlog = []
        self.novel_pop = []

    def _initialize_population(self):
        self.population = []
        for i in range(self.population_size):
            self.population.append(self.model_generator(i))

    def _memorize(self, q, i, limit):
        q.append(i)
        if len(q) > limit:
            q.pop()

    def mutate(self, individual, scale=1.0):
        mutation_mask = np.random.binomial(1, p=self.mutation_rate,␣
↪size=individual.W1.shape)
        individual.W1 += np.random.normal(loc=0, scale=scale, size=individual.
↪W1.shape) * mutation_mask
        mutation_mask = np.random.binomial(1, p=self.mutation_rate,␣
↪size=individual.W2.shape)
        individual.W2 += np.random.normal(loc=0, scale=scale, size=individual.
↪W2.shape) * mutation_mask
        return individual

    def inherit_weights(self, parent, child):
        child.W1 = parent.W1.copy()
        child.W2 = parent.W2.copy()
        return child

    def crossover(self, parent1, parent2):
        child1 = self.model_generator((parent1.id+1)*10)
        child1 = self.inherit_weights(parent1, child1)
        child2 = self.model_generator((parent2.id+1)*10)
        child2 = self.inherit_weights(parent2, child2)
        # first W
        n_neurons = child1.W1.shape[1]
        cutoff = np.random.randint(0, n_neurons)
        child1.W1[:, cutoff:] = parent2.W1[:, cutoff:].copy()
        child2.W1[:, cutoff:] = parent1.W1[:, cutoff:].copy()
        # second W
        n_neurons = child1.W2.shape[1]
        cutoff = np.random.randint(0, n_neurons)
        child1.W2[:, cutoff:] = parent2.W2[:, cutoff:].copy()
        child2.W2[:, cutoff:] = parent1.W2[:, cutoff:].copy()
        return child1, child2

    def get_state(self, t):
        window_size = self.window_size + 1
        d = t - window_size + 1
        block = self.trend[d : t + 1] if d >= 0 else -d * [self.trend[0]] +␣
↪self.trend[0 : t + 1]
        res = []
        for i in range(window_size - 1):
            res.append(block[i + 1] - block[i])
        return np.array([res])

    def act(self, p, state):
        logits = feed_forward(state, p)
        return np.argmax(logits, 1)[0]

    def buy(self, individual):
        initial_money = self.initial_money
        starting_money = initial_money
        state = self.get_state(0)
        inventory = []
        states_sell = []
        states_buy = []

        for t in range(0, len(self.trend) - 1, self.skip):
            action = self.act(individual, state)
            next_state = self.get_state(t + 1)

            if action == 1 and starting_money >= self.trend[t]:
                inventory.append(self.trend[t])
                initial_money -= self.trend[t]
                states_buy.append(t)
                print('day %d: buy 1 unit at price %f, total balance %f'% (t, self.trend[t], initial_money))

            elif action == 2 and len(inventory):
                bought_price = inventory.pop(0)
                initial_money += self.trend[t]
                states_sell.append(t)
                try:
                    invest = ((self.trend[t] - bought_price) / bought_price) * 100
                except:
                    invest = 0
                print(
                    'day %d, sell 1 unit at price %f, investment %f %%, total balance %f,'
                    % (t, self.trend[t], invest, initial_money)
                )
            state = next_state

        invest = ((initial_money - starting_money) / starting_money) * 100
        total_gains = initial_money - starting_money
        return states_buy, states_sell, total_gains, invest

    def calculate_fitness(self):
        for i in range(self.population_size):
            initial_money = self.initial_money
            starting_money = initial_money
            state = self.get_state(0)
            inventory = []

            for t in range(0, len(self.trend) - 1, self.skip):
                action = self.act(self.population[i], state)
                next_state = self.get_state(t + 1)

                if action == 1 and starting_money >= self.trend[t]:
                    inventory.append(self.trend[t])
                    starting_money -= self.trend[t]

                elif action == 2 and len(inventory):
                    bought_price = inventory.pop(0)
                    starting_money += self.trend[t]

                state = next_state
            invest = ((starting_money - initial_money) / initial_money) * 100
            self.population[i].fitness = invest
            self.population[i].last_features = self.population[i].W2.flatten()

    def evaluate(self, individual, backlog, pop, k = 4):
        score = 0
        if len(backlog):
            x = np.array(backlog)
            nn = NearestNeighbors(n_neighbors = k, metric = 'euclidean').fit(np.
↪array(backlog))
            d, _ = nn.kneighbors([individual])
            score += np.mean(d)

        if len(pop):
            nn = NearestNeighbors(n_neighbors = k, metric = 'euclidean').fit(np.
↪array(pop))
            d, _ = nn.kneighbors([individual])
            score += np.mean(d)

        return score

    def evolve(self, generations=20, checkpoint= 5):
        self._initialize_population()
        n_winners = int(self.population_size * 0.4)
        n_parents = self.population_size - n_winners
        for epoch in range(generations):
            self.calculate_fitness()
            scores = [self.evaluate(p.last_features, self.novel_backlog, self.
↪novel_pop) for p in self.population]
            sort_fitness = np.argsort(scores)[::-1]
            self.population = [self.population[i] for i in sort_fitness]
            fittest_individual = self.population[0]
            if (epoch+1) % checkpoint == 0:
                print('epoch %d, fittest individual %d with accuracy␣
↪%f'%(epoch+1, sort_fitness[0],
                                                                                    ␣
↪fittest_individual.fitness))
            next_population = [self.population[i] for i in range(n_winners)]
            total_fitness = np.sum([np.abs(i.fitness) for i in self.population])
            parent_probabilities = [np.abs(i.fitness / total_fitness) for i in␣
↪self.population]
            parents = np.random.choice(self.population, size=n_parents,␣
↪p=parent_probabilities, replace=False)

            for p in next_population:
                if p.last_features is not None:
                    self._memorize(self.novel_pop, p.last_features,␣
↪backlog_maxsize)
                    if np.random.randint(0,10) < novelty_search_threshold:
                        self._memorize(self.novel_backlog, p.last_features,␣
↪novelty_log_maxlen)

            for i in np.arange(0, len(parents), 2):
                child1, child2 = self.crossover(parents[i], parents[i+1])
                next_population += [self.mutate(child1), self.mutate(child2)]
            self.population = next_population

            if np.random.randint(0,10) < novelty_search_threshold:
                pop_sorted = sorted(self.population, key=lambda p: p.fitness,␣
↪reverse=True)
                self.generation_backlog.append(pop_sorted[0])
                print('novel add fittest, score: %f, backlog size:␣
↪%d'%(pop_sorted[0].fitness,
                                                                                  ␣
↪len(self.generation_backlog)))
                generation_backlog_temp = self.generation_backlog
                if len(self.generation_backlog) > backlog_maxsize:
                    generation_backlog_temp = random.sample(generation_backlog,␣
↪backlog_maxsize)
                for p in generation_backlog_temp:
                    if p.last_features is not None:
                        self._memorize(self.novel_backlog, p.last_features,␣
↪novelty_log_maxlen)

        return fittest_individual
```

[9]:

```
population_size = 100
generations = 100
mutation_rate = 0.1
neural_evolve = NeuroEvolution(population_size, mutation_rate, neuralnetwork,
                                    window_size, window_size, close, skip,␣
↪initial_money)
```

[11]:
```
fittest_nets = neural_evolve.evolve(100)
```

```
novel add fittest, score: 5.960001, backlog size: 16
novel add fittest, score: 2.560349, backlog size: 17
epoch 5, fittest individual 86 with accuracy -99.353801
novel add fittest, score: 2.073401, backlog size: 18
epoch 10, fittest individual 53 with accuracy -99.622801
novel add fittest, score: 9.773855, backlog size: 19
novel add fittest, score: 1.068502, backlog size: 20
novel add fittest, score: 1.733602, backlog size: 21
epoch 15, fittest individual 49 with accuracy -94.018300
novel add fittest, score: 1.439049, backlog size: 22
novel add fittest, score: 0.000000, backlog size: 23
novel add fittest, score: 0.000000, backlog size: 24
novel add fittest, score: 3.052850, backlog size: 25
epoch 20, fittest individual 83 with accuracy -42.284500
novel add fittest, score: 3.284498, backlog size: 26
novel add fittest, score: 3.284498, backlog size: 27
novel add fittest, score: 0.000000, backlog size: 28
novel add fittest, score: 0.000000, backlog size: 29
epoch 25, fittest individual 43 with accuracy -99.809850
novel add fittest, score: 0.000000, backlog size: 30
novel add fittest, score: 0.000000, backlog size: 31
novel add fittest, score: 4.712602, backlog size: 32
novel add fittest, score: 4.712602, backlog size: 33
epoch 30, fittest individual 51 with accuracy -94.734501
novel add fittest, score: 4.712602, backlog size: 34
novel add fittest, score: 0.000000, backlog size: 35
novel add fittest, score: 0.000000, backlog size: 36
epoch 35, fittest individual 74 with accuracy -99.895853
novel add fittest, score: 0.000000, backlog size: 37
novel add fittest, score: 0.000000, backlog size: 38
novel add fittest, score: 0.000000, backlog size: 39
novel add fittest, score: 0.000000, backlog size: 40
epoch 40, fittest individual 50 with accuracy -99.900900
novel add fittest, score: 0.000000, backlog size: 41
novel add fittest, score: 0.000000, backlog size: 42
epoch 45, fittest individual 98 with accuracy -92.305952
novel add fittest, score: 0.000000, backlog size: 43
novel add fittest, score: 0.000000, backlog size: 44
novel add fittest, score: 0.000000, backlog size: 45
novel add fittest, score: 0.000000, backlog size: 46
epoch 50, fittest individual 55 with accuracy -99.841901
novel add fittest, score: 0.000000, backlog size: 47
novel add fittest, score: 0.000000, backlog size: 48
novel add fittest, score: 0.000000, backlog size: 49
epoch 55, fittest individual 0 with accuracy -99.351002
novel add fittest, score: 0.000000, backlog size: 50
novel add fittest, score: 0.000000, backlog size: 51
novel add fittest, score: 0.000000, backlog size: 52
epoch 60, fittest individual 56 with accuracy -91.532553
novel add fittest, score: 0.000000, backlog size: 53
novel add fittest, score: 0.000000, backlog size: 54
novel add fittest, score: 0.000000, backlog size: 55
epoch 65, fittest individual 0 with accuracy -99.389200
novel add fittest, score: 0.000000, backlog size: 56
novel add fittest, score: 0.000000, backlog size: 57
novel add fittest, score: 0.000000, backlog size: 58
epoch 70, fittest individual 68 with accuracy -90.999901
novel add fittest, score: 0.000000, backlog size: 59
novel add fittest, score: 0.000000, backlog size: 60
novel add fittest, score: 0.000000, backlog size: 61
novel add fittest, score: 0.000000, backlog size: 62
epoch 75, fittest individual 50 with accuracy -98.881400
novel add fittest, score: 0.000000, backlog size: 63
novel add fittest, score: 0.000000, backlog size: 64
novel add fittest, score: 0.000000, backlog size: 65
epoch 80, fittest individual 0 with accuracy -91.959200
novel add fittest, score: 0.000000, backlog size: 66
novel add fittest, score: 0.000000, backlog size: 67
novel add fittest, score: 0.000000, backlog size: 68
epoch 85, fittest individual 0 with accuracy -94.175699
novel add fittest, score: 0.000000, backlog size: 69
novel add fittest, score: 0.000000, backlog size: 70
novel add fittest, score: 0.000000, backlog size: 71
novel add fittest, score: 0.000000, backlog size: 72
novel add fittest, score: 0.000000, backlog size: 73
epoch 90, fittest individual 60 with accuracy -93.196199
novel add fittest, score: 0.000000, backlog size: 74
novel add fittest, score: 0.000000, backlog size: 75
novel add fittest, score: 0.000000, backlog size: 76
epoch 95, fittest individual 66 with accuracy -93.122201
novel add fittest, score: 0.000000, backlog size: 77
novel add fittest, score: 0.000000, backlog size: 78
novel add fittest, score: 0.000000, backlog size: 79
epoch 100, fittest individual 52 with accuracy -93.193801
novel add fittest, score: 0.000000, backlog size: 80
```

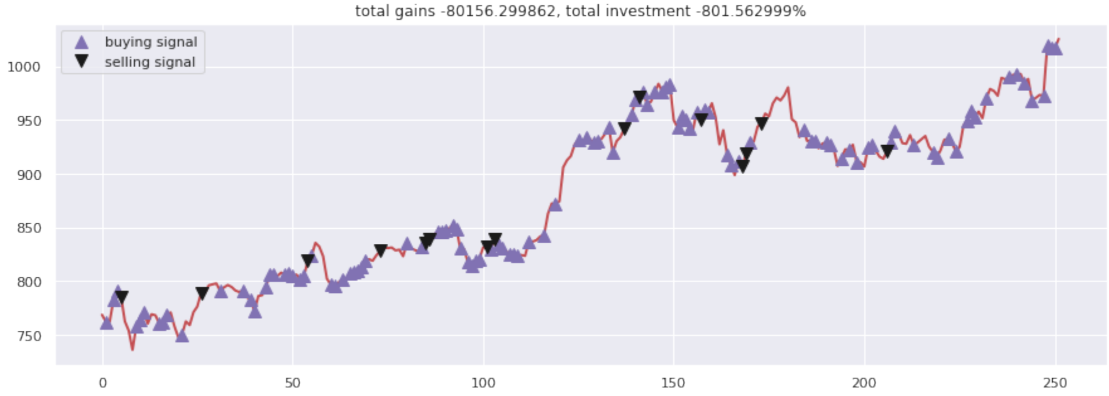
[12]:

```
states_buy, states_sell, total_gains, invest = neural_evolve.buy(fittest_nets)
```

```
day 1: buy 1 unit at price 762.130005, total balance 9237.869995
day 3: buy 1 unit at price 782.520020, total balance 8455.349975
day 4: buy 1 unit at price 790.510010, total balance 7664.839965
day 5, sell 1 unit at price 785.309998, investment 3.041475 %, total balance
8450.149963,
day 9: buy 1 unit at price 758.489990, total balance 7691.659973
day 10: buy 1 unit at price 764.479980, total balance 6927.179993
day 11: buy 1 unit at price 771.229980, total balance 6155.950013
day 15: buy 1 unit at price 760.989990, total balance 5394.960023
day 16: buy 1 unit at price 761.679993, total balance 4633.280030
day 17: buy 1 unit at price 768.239990, total balance 3865.040040
day 21: buy 1 unit at price 750.500000, total balance 3114.540040
day 26, sell 1 unit at price 789.289978, investment 0.865148 %, total balance
3903.830018,
day 31: buy 1 unit at price 790.799988, total balance 3113.030030
day 37: buy 1 unit at price 791.549988, total balance 2321.480042
day 39: buy 1 unit at price 782.789978, total balance 1538.690064
day 40: buy 1 unit at price 771.820007, total balance 766.870057
day 43: buy 1 unit at price 794.020020, total balance -27.149963
day 44: buy 1 unit at price 806.150024, total balance -833.299987
day 45: buy 1 unit at price 806.650024, total balance -1639.950011
day 48: buy 1 unit at price 806.359985, total balance -2446.309996
day 49: buy 1 unit at price 807.880005, total balance -3254.190001
day 50: buy 1 unit at price 804.609985, total balance -4058.799986
day 52: buy 1 unit at price 802.174988, total balance -4860.974974
day 53: buy 1 unit at price 805.020020, total balance -5665.994994
day 54, sell 1 unit at price 819.309998, investment 3.643216 %, total balance
-4846.684996,
day 55: buy 1 unit at price 823.869995, total balance -5670.554991
day 60: buy 1 unit at price 796.789978, total balance -6467.344969
day 61: buy 1 unit at price 795.695007, total balance -7263.039976
day 63: buy 1 unit at price 801.489990, total balance -8064.529966
day 65: buy 1 unit at price 806.969971, total balance -8871.499937
day 66: buy 1 unit at price 808.380005, total balance -9679.879942
day 67: buy 1 unit at price 809.559998, total balance -10489.439940
day 68: buy 1 unit at price 813.669983, total balance -11303.109923
day 69: buy 1 unit at price 819.239990, total balance -12122.349913
day 73, sell 1 unit at price 828.070007, investment 9.173492 %, total balance
-11294.279906,
day 80: buy 1 unit at price 835.239990, total balance -12129.519896
day 84: buy 1 unit at price 831.909973, total balance -12961.429869
day 85, sell 1 unit at price 835.369995, investment 9.272972 %, total balance
-12126.059874,
day 86, sell 1 unit at price 838.679993, investment 8.745772 %, total balance
-11287.379881,
day 88: buy 1 unit at price 845.539978, total balance -12132.919859
day 89: buy 1 unit at price 845.619995, total balance -12978.539854
day 90: buy 1 unit at price 847.200012, total balance -13825.739866
day 92: buy 1 unit at price 852.119995, total balance -14677.859861
day 93: buy 1 unit at price 848.400024, total balance -15526.259885
day 94: buy 1 unit at price 830.460022, total balance -16356.719907
day 96: buy 1 unit at price 817.580017, total balance -17174.299924
day 97: buy 1 unit at price 814.429993, total balance -17988.729917
day 98: buy 1 unit at price 819.510010, total balance -18808.239927
day 99: buy 1 unit at price 820.919983, total balance -19629.159910
day 101, sell 1 unit at price 831.500000, investment 9.265563 %, total balance
-18797.659910,
day 102: buy 1 unit at price 829.559998, total balance -19627.219908
day 103, sell 1 unit at price 838.549988, investment 10.092164 %, total balance
-18788.669920,
day 104: buy 1 unit at price 834.570007, total balance -19623.239927
day 105: buy 1 unit at price 831.409973, total balance -20454.649900
day 107: buy 1 unit at price 824.669983, total balance -21279.319883
day 108: buy 1 unit at price 824.729980, total balance -22104.049863
day 109: buy 1 unit at price 823.349976, total balance -22927.399839
day 112: buy 1 unit at price 837.169983, total balance -23764.569822
day 116: buy 1 unit at price 843.190002, total balance -24607.759824
day 119: buy 1 unit at price 871.729980, total balance -25479.489804
day 125: buy 1 unit at price 931.659973, total balance -26411.149777
day 127: buy 1 unit at price 934.299988, total balance -27345.449765
day 129: buy 1 unit at price 928.780029, total balance -28274.229794
day 130: buy 1 unit at price 930.599976, total balance -29204.829770
day 133: buy 1 unit at price 943.000000, total balance -30147.829770
day 134: buy 1 unit at price 919.619995, total balance -31067.449765
day 137, sell 1 unit at price 941.859985, investment 22.599708 %, total balance
-30125.589780,
day 139: buy 1 unit at price 954.960022, total balance -31080.549802
day 140: buy 1 unit at price 969.539978, total balance -32050.089780
day 141, sell 1 unit at price 971.469971, investment 29.443034 %, total balance
-31078.619809,
day 142: buy 1 unit at price 975.880005, total balance -32054.499814
day 143: buy 1 unit at price 964.859985, total balance -33019.359799
day 145: buy 1 unit at price 975.599976, total balance -33994.959775
day 147: buy 1 unit at price 976.570007, total balance -34971.529782
day 148: buy 1 unit at price 980.940002, total balance -35952.469784
day 149: buy 1 unit at price 983.409973, total balance -36935.879757
day 151: buy 1 unit at price 942.900024, total balance -37878.779781
day 152: buy 1 unit at price 953.400024, total balance -38832.179805
day 153: buy 1 unit at price 950.760010, total balance -39782.939815
day 154: buy 1 unit at price 942.309998, total balance -40725.249813
day 156: buy 1 unit at price 957.369995, total balance -41682.619808
day 157, sell 1 unit at price 950.630005, investment 20.211181 %, total balance
-40731.989803,
day 158: buy 1 unit at price 959.450012, total balance -41691.439815
day 159: buy 1 unit at price 957.090027, total balance -42648.529842
day 164: buy 1 unit at price 917.789978, total balance -43566.319820
day 165: buy 1 unit at price 908.729980, total balance -44475.049800
day 167: buy 1 unit at price 911.710022, total balance -45386.759822
day 168, sell 1 unit at price 906.690002, investment 14.546146 %, total balance
-44480.069820,
day 169, sell 1 unit at price 918.590027, investment 17.348210 %, total balance
-43561.479793,
day 170: buy 1 unit at price 928.799988, total balance -44490.279781
day 173, sell 1 unit at price 947.159973, investment 22.717728 %, total balance
-43543.119808,
day 184: buy 1 unit at price 941.530029, total balance -44484.649837
day 186: buy 1 unit at price 930.830017, total balance -45415.479854
day 187: buy 1 unit at price 930.390015, total balance -46345.869869
day 190: buy 1 unit at price 929.359985, total balance -47275.229854
day 191: buy 1 unit at price 926.789978, total balance -48202.019832
day 194: buy 1 unit at price 914.390015, total balance -49116.409847
day 196: buy 1 unit at price 922.219971, total balance -50038.629818
day 198: buy 1 unit at price 910.979980, total balance -50949.609798
day 201: buy 1 unit at price 924.690002, total balance -51874.299800
day 202: buy 1 unit at price 927.000000, total balance -52801.299800
day 206, sell 1 unit at price 921.289978, investment 16.028558 %, total balance
-51880.009822,
day 207: buy 1 unit at price 929.570007, total balance -52809.579829
day 208: buy 1 unit at price 939.330017, total balance -53748.909846
day 213: buy 1 unit at price 926.500000, total balance -54675.409846
day 218: buy 1 unit at price 920.289978, total balance -55595.699824
day 219: buy 1 unit at price 915.000000, total balance -56510.699824
day 222: buy 1 unit at price 932.450012, total balance -57443.149836
day 224: buy 1 unit at price 920.969971, total balance -58364.119807
day 227: buy 1 unit at price 949.500000, total balance -59313.619807
day 228: buy 1 unit at price 959.109985, total balance -60272.729792
day 229: buy 1 unit at price 953.270020, total balance -61225.999812
day 232: buy 1 unit at price 969.960022, total balance -62195.959834
day 238: buy 1 unit at price 989.679993, total balance -63185.639827
day 240: buy 1 unit at price 992.179993, total balance -64177.819820
day 242: buy 1 unit at price 984.450012, total balance -65162.269832
day 244: buy 1 unit at price 968.450012, total balance -66130.719844
day 247: buy 1 unit at price 972.559998, total balance -67103.279842
day 248: buy 1 unit at price 1019.270020, total balance -68122.549862
day 249: buy 1 unit at price 1017.109985, total balance -69139.659847
day 250: buy 1 unit at price 1016.640015, total balance -70156.299862
```

[13]:

```
fig = plt.figure(figsize = (15,5))
plt.plot(close, color='r', lw=2.)
plt.plot(close, '^', markersize=10, color='m', label = 'buying signal',␣
 ↪markevery = states_buy)
plt.plot(close, 'v', markersize=10, color='k', label = 'selling signal',␣
 ↪markevery = states_sell)
plt.title('total gains %f, total investment %f%%'%(total_gains, invest))
plt.legend()
plt.show()
```



total gains -80156.299862, total investment -801.562999%

[ ]: