Quiz 7 Solutions

CSE 4/574

Fall, 2024

Question 1

We consider the points: - positive examples: $(1, 1)$, $(7, -2)$, $(3, -2)$. - negative examples: $(1, 3)$, $(3, 4)$, $(3, 5)$, $(5, 5)$, $(7, 6)$.
We run a support vector machine (SVM) (with linear kernel) on this tiny set of points. Remember that a SVM tries to maximize the margin between positive and negative examples. Find the equation of the decision boundary and compute the number of support vectors SV, as well as the margin (recall that the margin is defined as the distance from the decision boundary to any support vector).

Correct Choice

$SV = 2$ and the decision boundary is $y - 2 = 0$

Problem Explanation:
$SV = 2$
- positive examples: $(1, 1)$ [SV], $(7, -2)$, $(3, -2)$.
- negative examples: $(1, 3)$ [SV], $(3, 4)$, $(3, 5)$, $(5, 5)$, $(7, 6)$.
The decision boundary is $y - 2 = 0$. The margin is 1.

Question 2

You will be required to run Expectation Maximization for estimating parameters of a Gaussian mixture model in this example. Refer to lecture slides for the exact formulation. You can use python or do the computations by hand. Recall that you need to use the pdf formulation for computing $P(x|\theta_k)$.

Consider the following one dimensional data set:

$[2.3, 3.2, 3.1, 1.6, 1.9, 11.5, 10.2, 12.3, 8.6, 10.9]$
Assume that we are interested in learning a mixture model with two components (k = 2). Let P_{ik} denote the probability $P(z_i = k)$ for any i. Let $(\mu_1, \sigma 1)$ be the parameters for the first Gaussian component of the mixture and $(\mu_2, \sigma 2)$ be the parameters for the second Gaussian

component of the mixture.

Given the following initialization:

$$p_{i1} = p_{i2} = 0.5$$
$$\mu_1 = \mu_2 = 0$$
$$\sigma_1 = \sigma_2 = 1$$

Correct Choice

The estimates of p_{i2} and p_{i2} are equal after the first M steps.
Problem Explanation:

```
s = np.array([2.3,3.2,3.1,1.6,1.9,11.5,10.2,12.3,8.6,10.9])

# sample
max_iter = 20
N = 10

# Initial guess of parameters and initializations
theta0 = np.array([1,1,10,1,0.5])
mu1, sig1, mu2, sig2, pi_1 = theta0
mu = np.array([mu1, mu2])
sig = np.array([sig1, sig2])
pi_ = np.array([pi_1, 1-pi_1])

gamma = np.zeros((2, s.size))
N_ = np.zeros(2)
theta_new = theta0
print(gamma.size)

# EM loop
counter = 0
converged = False
while not converged:
    print(counter)
    # Compute the responsibility func. and new parameters
    for k in [0,1]:
        # E Step
        gamma[k,:] = pi_[k]*norm.pdf(s, mu[k], sig[k])/pdf_model(s, theta_new)
        # M Step
        N_[k] = 1.*gamma[k].sum()
        mu[k] = sum(gamma[k]*s)/N_[k]
        sig[k] = np.sqrt( sum(gamma[k]*(s-mu[k])**2)/N_[k] )
        pi_[k] = N_[k]/s.size
    theta_new = [mu[0], sig[0], mu[1], sig[1], pi_[0]]
    assert abs(N_.sum() - N)/float(N) < 1e-6
    assert abs(pi_.sum() - 1) < 1e-6
    print("iter")
    print(theta_new)
    # Convergence check
    counter += 1
    converged = counter >= max_iter
```

Output:
20

0
iter

2.5499999997776315, 0.6500000000149414, 10.899999998145324, 1.2315302191338837, 0.3999999998560775

1
iter

2.549999999010176, 0.6500000000523068, 10.89999999166535, 1.2315302389819789, 0.3999999935368635

2
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

3
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

4
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

5
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

6
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

7
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

8
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

9
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

10
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

11
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

12
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

13
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

14
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

15
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

16
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

17
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

18
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859

19
iter

2.5499999990101756, 0.650000000052307, 10.899999991665345, 1.2315302389819958, 0.3999999993536859