log_region_sp

May 26, 2025

[1]:
```python
import numpy as np
import random
import math
def process_check_node(mu):
    def gamma(x):
        ex=np.exp(x)
        return math.log((ex+1)/(ex-1))

    v_pass=dict()
    for i in range(6):
        filtered_mu = {key: val for key, val in mu.items() if key[0] == i}
        # print(filtered_mu)

        mu_tilde = {}
        mu_sgn = {}
        for j,v in filtered_mu.items():
            mu_tilde[j] = gamma(abs(filtered_mu[j]))
            # print(f"filtered_mu[{j}]={filtered_mu[j]}")
            # print(f"mu_tilde[{j}]={mu_tilde[j]}")
            mu_sgn[j] = np.sign(v)

        for j,v in filtered_mu.items():
            index_mu=filtered_mu.copy()
            index_mu.pop(j)
            sum_product=1
            sum_mu_tilde=0
            for k,v in index_mu.items():
                sum_product *= mu_sgn[k]
                sum_mu_tilde += mu_tilde[k]

            v_pass[j] = sum_product * gamma(sum_mu_tilde)


    return v_pass

def process_judge(v_factor,v_pass,H):
    kai=np.zeros(6)
```

```
    for j in range(6):
        filtered_vpass = {key: val for key, val in v_pass.items() if key[1] ==␣
↪j}.copy()
        # print(filtered_vpass)
        dot_all=v_factor[j].copy()
        for v in filtered_vpass.values():
            dot_all=dot_all*v
        kai[j] = dot_all

    if np.all(kai==0):
        print("x_pred is not estimated")
    else:
        x_pred=(0<kai).astype(int)
        print(f"x_pred={x_pred}")
        if np.all(np.dot(H,x_pred) == np.zeros(6)):
            print("x_pred is correct")
        else:
            print("x_pred is wrong")

def process_variable_node(v_factor,v_pass):
    mu=dict()
    for j in range(6):
        filtered_vpass = {key: val for key, val in v_pass.items() if key[1] ==␣
↪j}.copy()
        # print(f"filtered_vpass{filtered_vpass}")
        for i,v in filtered_vpass.items():
            dot_target_dic=filtered_vpass.copy()
            dot_target_dic.pop(i)
            # print(dot_target_dic)
            mu[i] = 1
            # print(mu[k])
            for _,conv_target in dot_target_dic.items():
                mu[i] = mu[i]*conv_target
            mu[i] = mu[i] + v_factor[j]
    return mu
```

[2]:
```
def log_region_sp_decoder(v_factor):
    def print_msg(msg):
        for i in range(6):
            for j in range(6):
                if (i,j) in msg:
                    print(f"msg[{i},{j}]={msg[(i,j)]}",end=" ")
            print("")

    H=np.array(
        [
            [1,1,0,0,0,0],
```

```
            [0,1,1,0,0,0],
            [0,0,1,1,0,0],
            [0,0,0,1,1,0],
            [0,0,0,0,1,1],
            [1,0,0,0,0,1]
        ]
    )

    v_pass=dict()
    for i in range(H.shape[0]):
        for j in range(H.shape[1]):
            if H[i,j] == 1:
                v_pass[(i,j)] = 0

    v_pass=v_pass.copy()
    for t in range(3):
        print(f"t={t}")
        mu=process_variable_node(v_factor,v_pass)
        print("mu")
        print_msg(mu)
        v_pass=process_check_node(mu)
        print("v_pass")
        print_msg(v_pass)
        process_judge(v_factor,v_pass,H)

msg=[math.log(3),math.log(1/3)]
init_v=np.array([msg[1],msg[1],msg[0],msg[0],msg[0],msg[0]])
print(init_v)
log_region_sp_decoder(init_v)
```

```
[-1.09861229 -1.09861229  1.09861229  1.09861229  1.09861229  1.09861229]
t=0
mu
msg[0,0]=-1.0986122886681098 msg[0,1]=-1.0986122886681098
msg[1,1]=-1.0986122886681098 msg[1,2]=1.0986122886681098
msg[2,2]=1.0986122886681098 msg[2,3]=1.0986122886681098
msg[3,3]=1.0986122886681098 msg[3,4]=1.0986122886681098
msg[4,4]=1.0986122886681098 msg[4,5]=1.0986122886681098
msg[5,0]=-1.0986122886681098 msg[5,5]=1.0986122886681098
v_pass
msg[0,0]=-1.09861228866811 msg[0,1]=-1.09861228866811
msg[1,1]=1.09861228866811 msg[1,2]=-1.09861228866811
msg[2,2]=1.09861228866811 msg[2,3]=1.09861228866811
msg[3,3]=1.09861228866811 msg[3,4]=1.09861228866811
msg[4,4]=1.09861228866811 msg[4,5]=1.09861228866811
msg[5,0]=1.09861228866811 msg[5,5]=-1.09861228866811
x_pred=[1 1 0 1 1 0]
```

```
x_pred is wrong
t=1
mu
msg[0,0]=2.220446049250313e-16 msg[0,1]=2.220446049250313e-16
msg[1,1]=-2.19722457733622 msg[1,2]=2.19722457733622
msg[2,2]=-2.220446049250313e-16 msg[2,3]=2.19722457733622
msg[3,3]=2.19722457733622 msg[3,4]=2.19722457733622
msg[4,4]=2.19722457733622 msg[4,5]=-2.220446049250313e-16
msg[5,0]=-2.19722457733622 msg[5,5]=2.19722457733622
v_pass
msg[0,0]=2.2204460492503128e-16 msg[0,1]=2.2204460492503128e-16
msg[1,1]=2.1972245773362205 msg[1,2]=-2.1972245773362205
msg[2,2]=2.1972245773362205 msg[2,3]=-2.2204460492503128e-16
msg[3,3]=2.1972245773362205 msg[3,4]=2.1972245773362205
msg[4,4]=-2.2204460492503128e-16 msg[4,5]=2.1972245773362205
msg[5,0]=2.1972245773362205 msg[5,5]=-2.1972245773362205
x_pred=[0 0 0 0 0 0]
x_pred is correct
t=2
mu
msg[0,0]=1.0986122886681107 msg[0,1]=1.0986122886681107
msg[1,1]=-1.0986122886681096 msg[1,2]=3.29583686600433
msg[2,2]=-1.0986122886681107 msg[2,3]=3.29583686600433
msg[3,3]=1.0986122886681096 msg[3,4]=1.0986122886681096
msg[4,4]=3.29583686600433 msg[4,5]=-1.0986122886681107
msg[5,0]=-1.0986122886681096 msg[5,5]=3.29583686600433
v_pass
msg[0,0]=1.0986122886681107 msg[0,1]=1.0986122886681107
msg[1,1]=3.2958368660043296 msg[1,2]=-1.0986122886681093
msg[2,2]=3.2958368660043296 msg[2,3]=-1.0986122886681107
msg[3,3]=1.0986122886681093 msg[3,4]=1.0986122886681093
msg[4,4]=-1.0986122886681107 msg[4,5]=3.2958368660043296
msg[5,0]=3.2958368660043296 msg[5,5]=-1.0986122886681093
x_pred=[0 0 0 0 0 0]
x_pred is correct
```