Restricted Boltzmann Machine

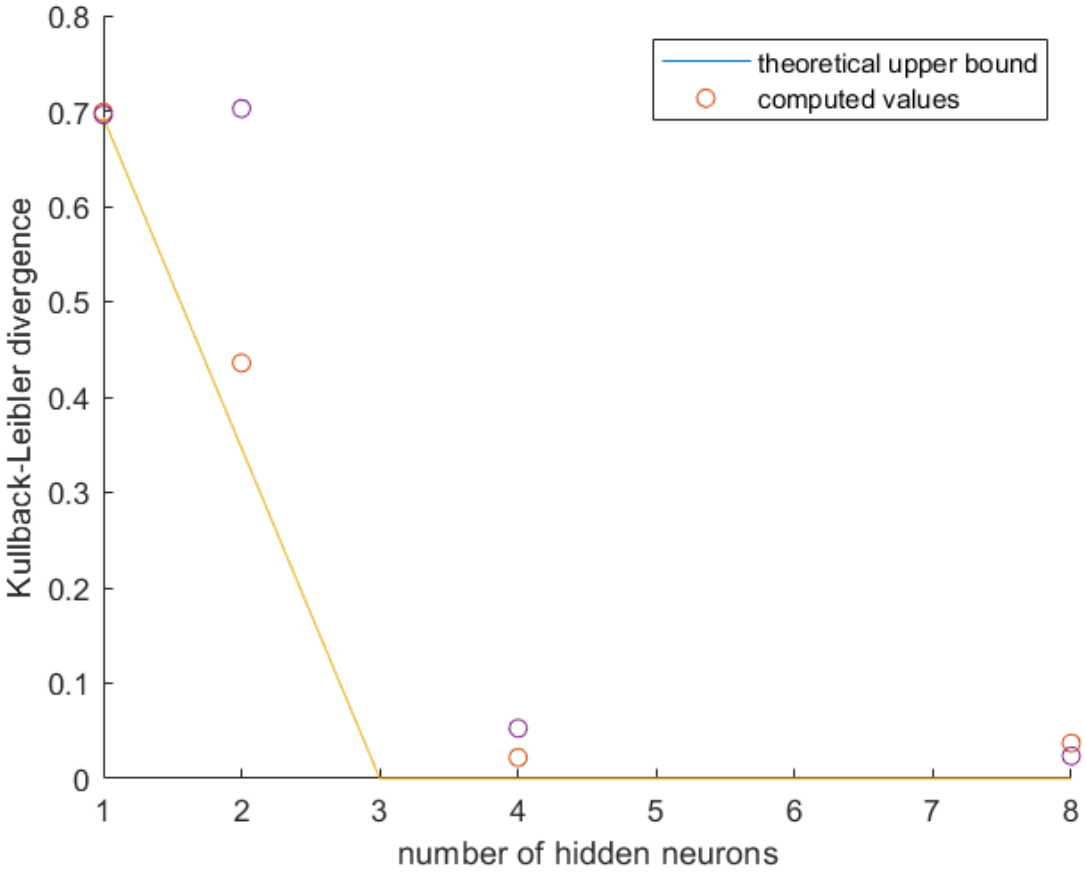
```
% Restricted Boltzmann Machine
 clear all;
 clc
  x1 = [-1,-1,-1];
  x2 = [1,-1,1];
  x3 = [-1,1,1];
  x4 = [1,1,-1];
  x5 = [1,1,1];
  x6 = [-1,1,-1];
  x7 = [1,-1,-1];
  x8 = [-1,-1,1];

  x = [x1; x2; x3; x4; x5; x6; x7; x8];

  neurons_visible = 3;
  neurons_hidden = [1, 2, 4, 8];
  nTrials = 1000;
  mini_batch = 20;
  eta = 0.005;
  dkl_final = [];
  for m = 1:length(neurons_hidden)
      v = zeros(3,1);
      h = zeros(neurons_hidden(m),1); %state of the hidden
layer
      theta_v = zeros(3,1);
      theta_h = zeros(neurons_hidden(m),1);
      weights = randn(neurons_hidden(m),3);

      for i = 1:nTrials
          total_delta_theta_v = zeros(3,1);
          total_delta_theta_h = zeros(neurons_hidden(m),1);
          total_delta_weights = zeros(neurons_hidden(m),3);

          for j = 1:mini_batch
              index_pattern = randi(4);
              v = transpose(x(index_pattern,:));
              v0 = v;
              local_field_h0 = weights*v0 - theta_h;

              local_field_h = weights*v - theta_h;
              prob_local_field_h = 1./(1+exp(-
2*local_field_h));
```

```
                for p = 1: neurons_hidden(m)
                    r = rand;
                    if r < prob_local_field_h(p)
                        h(p) = 1;
                    else
                        h(p) = -1;
                    end
                end

                for q = 1:200
                    local_field_v =
transpose(transpose(h)*weights) - theta_v;
                    prob_local_field_v = 1./(1+exp(-
2*local_field_v));

                    for t = 1: neurons_visible
                      r = rand;
                      if r < prob_local_field_v(t)
                          v(t) = 1;
                      else
                          v(t) = -1;
                      end
                    end

                    local_field_h = weights*v - theta_h;
                    prob_local_field_h = 1./(1+exp(-
2*local_field_h));

                    for p = 1: neurons_hidden(m)
                      r = rand;
                      if r < prob_local_field_h(p)
                          h(p) = 1;
                      else
                          h(p) = -1;
                      end
                    end
                end

                delta_weights = eta*(tanh(local_field_h0)*v0'
- tanh((local_field_h)*v'));
                total_delta_weights = total_delta_weights +
delta_weights;
                delta_theta_v = -eta*(v0 - v);
```

```
                total_delta_theta_v = total_delta_theta_v +
delta_theta_v;
                delta_theta_h = -eta*(tanh(local_field_h0) -
tanh(local_field_h));
                total_delta_theta_h = total_delta_theta_h +
delta_theta_h;
            end


            weights = weights + total_delta_weights;
            theta_v = theta_v + total_delta_theta_v;
            theta_h = theta_h + total_delta_theta_h;
        end
        dkl = 0;
        boltzmann_probability = [0;0;0;0;0;0;0;0];
        for outer_loop = 1: 3000
            random_pattern = randi(8);
            v = transpose(x(random_pattern,:));
            local_field_h = weights*v - theta_h;
            prob_local_field_h = 1./(1+exp(-
2*local_field_h));
            for p = 1: neurons_hidden(m)
              r = rand;
              if r < prob_local_field_h(p)
                  h(p) = 1;
              else
                  h(p) = -1;
              end
            end

            for inner_loop = 1 : 2000
              local_field_v = transpose(transpose(h)*weights)
- theta_v;
              prob_local_field_v = 1./(1+exp(-
2*local_field_v));

                  for t = 1: neurons_visible
                      r = rand;
                      if r < prob_local_field_v(t)
                          v(t) = 1;
                      else
                          v(t) = -1;
                      end
                  end
```

```
                local_field_h = weights*v - theta_h;
                prob_local_field_h = 1./(1+exp(-
2*local_field_h));

                for p = 1: neurons_hidden(m)
                    r = rand;
                    if r < prob_local_field_h(p)
                        h(p) = 1;
                    else
                        h(p) = -1;
                    end
                end

                for c = 1:8
                    if isequal(v, x(c, :)')
                        boltzmann_probability(c) =
boltzmann_probability(c)+1;
                    end
                end
            end
        end

        boltzmann_probability =
boltzmann_probability/(outer_loop*inner_loop);
        for b = 1:4
            if boltzmann_probability(b) == 0
                l_boltzmann_probability(b) = 0;
            else
                l_boltzmann_probability(b)=
log(boltzmann_probability(b));
            end
        end

        for z = 1:4
          dkl = dkl + 0.25*(log(0.25) -
l_boltzmann_probability(z));
        end
        dkl_final = [dkl_final,dkl];
        disp(dkl_final);
    end


neurons_hidden = [1, 2, 4, 8];
```

```
M1 = [1,2,3,4,8];
upperBounds = zeros(5, 1);
for i = 1:5
    neuronsHidden = M1(i);
    if neuronsHidden >= 2^(neurons_visible-1)-1
        continue
    end
    x = floor(log(neuronsHidden+1));
    upperBounds(i) = log(2) * (neurons_visible - x -
(neuronsHidden+1)/(2^x)); %theoretical dkl
end

hold on;
plot(M1 , upperBounds);
scatter(neurons_hidden, dkl_final);
legend('theoretical upper bound', 'computed values');
xlabel('number of hidden neurons');
ylabel('Kullback-Leibler divergence');
hold off;
```


theoretical upper bound
computed values
Kullback-Leibler divergence
number of hidden neurons
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8