Homework 5: November 30, 2016

Machine Learning

Susan Cherry

Question 1

Part A:

First, for a single i, $\frac{d\mathcal{L}_i}{dW_3} = \frac{d\mathcal{L}_i}{f(x_i)} * \frac{df(x_i)}{dW_3} = (y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * h_2$.

$\frac{d\mathcal{L}_i}{dW_2} = \frac{d\mathcal{L}_i}{f(x_i)} * \frac{df(x_i)}{dh2} * \frac{dh_2}{dW2} = (y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * W_3 * h_2(1-h_2)$.

Putting all of this together, I get: $\frac{d\mathcal{L}_i}{dW_1} = \frac{d\mathcal{L}_i}{f(x_i)} * \frac{df(x_i)}{dh2} * \frac{dh_2}{dh1}\frac{dh1}{dW_1} = (y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * W_3 * h_2(1-h_2)W_2h_1(1-h_1)x_i$

Next, I solve $\frac{d\mathcal{L}_i}{db_1}$, which is nearly identical to the process that I went through about. Now the only change is $\frac{dh1}{db_1} = 1$. This gives me:
$\frac{d\mathcal{L}_i}{db_1} = (y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * W_3 * h_2(1-h_2)W_2h_1(1-h_1)$.

Putting in the summation, my final answer is:
$\frac{d\mathcal{L}}{dW_1} = \sum_i((y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * W_3 * h_2(1-h_2)W_2h_1(1-h_1)x_i)$
$\frac{d\mathcal{L}}{db_1} = \sum_i((y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * W_3 * h_2(1-h_2)W_2h_1(1-h_1))$

Part B: Next, assume there are $L-1$ hidden layers. First, notice that for a single i, $\frac{d\mathcal{L}_i}{dW_L} = \frac{d\mathcal{L}_i}{df(x_i)} * \frac{df(x_i)}{d(W_L^T h_{L-1}+b_L)} * \frac{d(W_L^T h_{L-1}+b_L))}{dW_L}$. We also have:
$\frac{d\mathcal{L}_i}{df(x_i)} = y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}$

$\frac{df(x_i)}{d(W_L^T h_{L-1}+b_L)} = (h_{L-1})(1-(h_{L-1})) = f(x_i)(1-f(x_i))$

$\frac{d(W_L^T h_{L-1}+b_L))}{dW_L} = h_{L-1}$

This means that $\frac{d\mathcal{L}_i}{dW_L} = (y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)}) * f(x_i)(1-f(x_i)) * h_{L-1}$.

Define $\delta_L = y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)} * f(x_i)(1-f(x_i))$

Go to the next level: $\frac{d\mathcal{L}_i}{dW_{L-1}} = \frac{d\mathcal{L}_i}{df(x_i)} * \frac{df(x_i)}{d(W_{L-1}^T h_{L-2}+b_{L-1})} * \frac{d(W_{L-1}^T h_{L-2}+b_{L-1}))}{dW_{L-1}}$.

This gives us:
$\frac{d\mathcal{L}_i}{dh_{L-1}} = \frac{d\mathcal{L}_i}{d(W_{L-1}^T h_{L-2}+b_{L-1})} * \frac{d(W_{L-1}^T h_{L-2}+b_{L-1})}{dh_{L-1}} = \delta_L W_L$

$\frac{dh_{L-1}}{d(W_{L-1}^T h_{L-2}+b_{L-1})} = h_{L-1}(1-h_{L-1})$

$\frac{d(W_{L-1}^T h_{L-2}+b_{L-1}))}{dW_{L-1}} = h_{L-2}$

Putting this altogether, we have: $\frac{d\mathcal{L}_i}{dW_{L-1}} = \delta_L W_L * h_{L-1}(1-h_{L-1}) * h_{L-2}$
Now define $\delta_{L-1} = \delta_L W_L h_{L-1}(1-h_{L-1})$
It is clear that the general rule is: $\frac{d\mathcal{L}_i}{dW_{i-1}} = \delta_i h_{i-1}(1-h_{i-1})h_{i-2}$ where $\delta_i = \delta_{i+1} W_{i+1} h_i(1-h_i)$

This means $\frac{d\mathcal{L}_i}{dW_1} = \delta_2 W_2(h_1)(1-h_1)x_i = \delta_3 W_3 h_2(1-h_2)W_2(h_1)(1-h_1)x_i$

For $\frac{d\mathcal{L}_i}{db_1}$, I follow the same process.

$\frac{d\mathcal{L}_i}{dW_L} = \frac{d\mathcal{L}}{df(x_i)} * \frac{df(x_i)}{d(W_L^T h_{L-1}+b_L)} * \frac{d(W_L^T h_{L-1}+b_L))}{db_L}$.

$\frac{d(W_L^T h_{L-1}+b_L))}{db_L}$ is the only term that is different. $\frac{d(W_L^T h_{L-1}+b_L))}{db_L} = 1$.

This gives me: $\frac{d\mathcal{L}_i}{db_L} = \sum_i y_i * \frac{1}{f(x_i)} - (1-y_i)\frac{1}{1-f(x_i)} * f(x_i)(1-f(x_i) = \delta_L$ and $\frac{d\mathcal{L}_i}{db_{L-1}} = delta_L W_L * h_{L-1}(1-h_{L-1})$.

Thus, $\frac{d\mathcal{L}_i}{db_1} = \delta_2 W_2(h_1)(1-h_1) = \delta_3 W_3 h_2(1-h_2)W_2(h_1)(1-h_1)$.

Putting everything together and including the summation gives me the final answer:
$\frac{d\mathcal{L}}{dW_1} = \sum_i(\delta_2 W_2(h_1)(1-h_1)x_i) = \sum_i(\delta_3 W_3 h_2(1-h_2)W_2(h_1)(1-h_1)x_i)$
$\frac{d\mathcal{L}}{db_1} = \sum_i(\delta_2 W_2(h_1)(1-h_1)) = \sum_i(\delta_3 W_3 h_2(1-h_2)W_2(h_1)(1-h_1))$

Question 2

Part A: First, I need to use the EM algorithm to derive the estimation. First, notice that we have i samples with $i = 1....m$. For each sample i, I will introduce a "latent variable" Z_i which refers to the coin that was used for that sample. So if A was the coin actually used for sample i, $Z_i = A$. Notice that we only have two clusters, A and B. Note that I assume that the probability of choosing each coin is 0.5, as is specified on Piazza. Also let h_i denote the number of heads in sample i. This means that the number of tails is $t_i = n - h_i$

E Step: The E-Step of the EM algorithm is computing the cluster assignments probabilistically. In the notes this is written as compute $P(Z_i = k|x_i, \theta_t)$ for each i, k. In this case, we compute $P(Z_i = A|x_i, \theta_t) = \frac{\theta_A^{h_i}(1-\theta_A)^{t_i}}{\theta_A^{h_i}(1-\theta_A)^{t_i}+\theta_B^{h_i}(1-\theta_B)^{t_i}}$

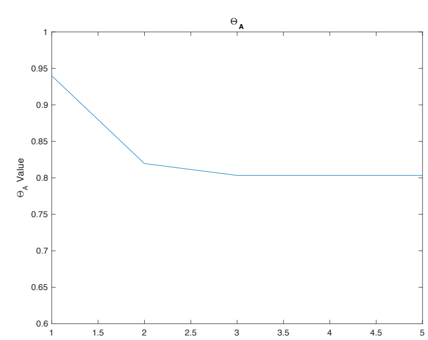
and $P(Z_i = B|x_i, \theta_t) = \frac{\theta_B^{h_i}(1-\theta_B)^{t_i}}{\theta_A^{h_i}(1-\theta_A)^{t_i}+\theta_B^{h_i}(1-\theta_B)^{t_i}}$ for all $i = 1...m$

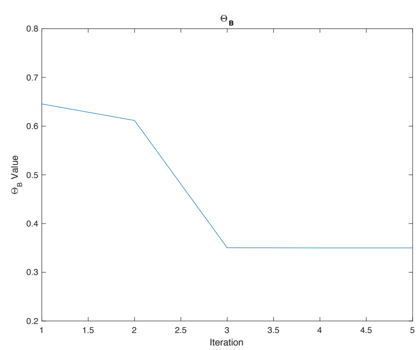
M Step: Next, the M-Step updates θ_t. According to the notes, it is calculated by maximizing $A(\theta, \theta_t) \propto \sum_i \sum_k P(Z_i = k|x_i, \theta_i) logP(X_i = xi, Z_i = k|\theta)$ by taking the gradient and setting to 0. In this case, I have:
$A(\theta, \theta_t) \propto \sum_{j=1}^{n}[P(Z_i = A|x_i, \theta_i)(log(0.5) + h_i log\theta_A + t_i log(1-\theta_A)) + P(Z_i = B|x_i, \theta_t)(log(0.5) + h_i log\theta_B + t_i log(1-\theta_B))]$.
By taking the derivative with respect to θ_A and θ_B and setting to zero, I find the following:
$\frac{\partial A(\theta,\theta_t)}{\partial \theta_A} = \sum_i [P(Z_i = A|x_i, \theta_i)\frac{h_i}{\theta_A} + P(Z_i = A|x_i, \theta_i)\frac{t_i}{1-\theta_A}]$.
Setting equal to zero gives: $\sum_i [P(Z_i = A|x_i, \theta_i)\frac{h_i}{\theta_A(1-\theta_A)} - P(Z_i = A|x_i, \theta_i)\frac{h_i\theta_A}{\theta_A(1-\theta_A)} - P(Z_i = A|x_i, \theta_i)\frac{t_i\theta_A}{\theta_A(1-\theta_A)}] = 0$
Rearranging gives us: $\theta_A = \frac{\sum_i P(Z_i=A|x_i,\theta_t)*h_i}{\sum_i P(Z_i=A|x_i,\theta_t)*(h_i+t_i)}$
The case for θ_B is exactly analogous. The final result is as follows:
$\theta_A^{t+1} = \frac{\sum_i P(Z_i=A|x_i,\theta_t)*h_i}{\sum_i P(Z_i=A|x_i,\theta_t)*n}$.
$\theta_B^{t+1} = \frac{\sum_i P(Z_i=B|x_i,\theta_t)*h_i}{\sum_i P(Z_i=B|x_i,\theta_t)*n}$.

The EM Algorithm is as follows: First, randomly initialize θ_A^0 and θ_B^0.
1) Perform the E Step.
2) Update θ_A^t and θ_A^t using the M Step.
Once $\theta^{t+1} = \theta^t$, the algorithm has converged. Stop and output θ^{t+1}.

Part B: Next I implement the EM algorithm to estimate θ. I set $m = 5$, $n = 100$, $\theta_A = 0.8$, and $\theta_B = 0.35$. I randomly initialize θ_A^0 and θ_B^0. I randomly generate the 5 m samples by randomly choosing which coin generates each sample and then using that coin to randomly determine the proportion of heads/tails. See my Matlab code for exact implementation. After coding up and running the algorithm, I find $\hat{\theta_A} = 0.7933$ and $\hat{\theta_B} = 0.3450$. These values are nearly identical to the true values of θ_A and θ_B, so it seems that the EM algorithm has worked well in this case.

Here are the plots of how my θ_A and θ_B change over time.





Here is the code from my EM algorithm. See the attached Matlab file for exactly how I simulated the coin flips.

```
  \%Continue until converges
while prev_theta_A~=theta_A || prev_theta_B~=theta_B
    prev_theta_A=theta_A;
    prev_theta_B=theta_B;

    \%E Step
    p_1=theta_A^sum(m1)*(1-theta_A)^(100-sum(m1))/(theta_A^sum(m1)*(1-theta_A)^
    (100-sum(m1))+theta_B^sum(m1)*(1-theta_B)^(100-sum(m1)));
    p_2=theta_A^sum(m2)*(1-theta_A)^(100-sum(m2))/(theta_A^sum(m2)*(1-theta_A)^
    (100-sum(m2))+theta_B^sum(m2)*(1-theta_B)^(100-sum(m2)));
    p_3=theta_A^sum(m3)*(1-theta_A)^(100-sum(m3))/(theta_A^sum(m3)*(1-theta_A)^
    (100-sum(m3))+theta_B^sum(m3)*(1-theta_B)^(100-sum(m3)));
    p_4=theta_A^sum(m4)*(1-theta_A)^(100-sum(m4))/(theta_A^sum(m4)*(1-theta_A)^
    (100-sum(m4))+theta_B^sum(m4)*(1-theta_B)^(100-sum(m4)));
    p_5=theta_A^sum(m5)*(1-theta_A)^(100-sum(m5))/(theta_A^sum(m5)*(1-theta_A)^
    (100-sum(m5))+theta_B^sum(m5)*(1-theta_B)^(100-sum(m5)));

    d_1=theta_B^sum(m1)*(1-theta_B)^(100-sum(m1))/(theta_A^sum(m1)*(1-theta_A)^
    (100-sum(m1))+theta_B^sum(m1)*(1-theta_B)^(100-sum(m1)));
    d_2=theta_B^sum(m2)*(1-theta_B)^(100-sum(m2))/(theta_A^sum(m2)*(1-theta_A)^
    (100-sum(m2))+theta_B^sum(m2)*(1-theta_B)^(100-sum(m2)));
    d_3=theta_B^sum(m3)*(1-theta_B)^(100-sum(m3))/(theta_A^sum(m3)*(1-theta_A)^
    (100-sum(m3))+theta_B^sum(m3)*(1-theta_B)^(100-sum(m3)));
    d_4=theta_B^sum(m4)*(1-theta_B)^(100-sum(m4))/(theta_A^sum(m4)*(1-theta_A)^
    (100-sum(m4))+theta_B^sum(m4)*(1-theta_B)^(100-sum(m4)));
    d_5=theta_B^sum(m5)*(1-theta_B)^(100-sum(m5))/(theta_A^sum(m5)*(1-theta_A)^
    (100-sum(m5))+theta_B^sum(m5)*(1-theta_B)^(100-sum(m5)));

    \%M Step
    theta_A=(p_1*sum(m1)+p_2*sum(m2)+p_3*sum(m3)+p_4*sum(m4)+p_5*sum(m5))/
   ((p_1+p_2+p_3+p_4+p_5)*100);
    theta_B=(d_1*sum(m1)+d_2*sum(m2)+d_3*sum(m3)+d_4*sum(m4)+d_5*sum(m5))/
   ((d_1+d_2+d_3+d_4+d_5)*100);
    blist=[blist theta_B];
    alist=[alist theta_A];

end
```

Question 3

Part A: See my uploaded code for my implementation of the k-means algorithm.

```
  function [new, k_means] = K_Means_Code(image,num_k)
\%K means function

unique_pixels=unique(image);

\%initialze cluster centers randomly
new=zeros(size(image));
k_means=randsample(unique_pixels,num_k);
tracker=0;
    while tracker<=2

      old_k=k_means;

      \%calculate distance
      for i=1:size(image,1)
            for j=1:size(image,2)
                values=abs(double(k_means)-double(image(i,j)));
                new(i,j)= find(values==min(values),1);
            end
      end

      \%update cluster means
        for i=1:length(k_means)
            matrix=new==i;
            k_means(i)=round(mean(image(matrix)));
        end

        \%end the algorithm if it has converged
        if k_means==old_k
            tracker=tracker+1;
        else
            tracker=0;
        end
    end

end
```

Part B: First, see my image for k=2. I can tell that the image is of a raccoon, but the quality is pretty poor.

Next, see my my image for k=4. You can see that even with just 4 clusters, the quality is dramatically improved. It's still not as good as the original, though.

Finally, I try k=10. Here the quality of the image is even better and looks pretty similar to the original to me.