CSE344: Computer Vision

Assignment-1

Divyajeet Singh (2021529)

February 17, 2024

Theory

Question 1.

(a) The given problem of classifying papaya images into the classes 'sweet' and 'not sweet' is a binary classification problem. The Mean Squared Error function is non-convex for binary classifcation. Thus if a binary classifier is trained with MSE Loss, it is not guaranteed to converege to the minimum of the loss. Secondly, using MSE assumes an underlying Gaussian distribution[1], which is not valid in case of binary classification, since it is modeled with a Bernoulli distribution.

(b) For a single training example, the Binary Cross Entropy loss function is given by (assuming $\hat{y}$ is a valid probability in $[0, 1]$)

$$\mathcal{L}(\hat{y}) = -y \log_2(\hat{y}) - (1-y)\log_2(1-\hat{y})$$

(c) On a negative example (where $y = 0$), if the predicted value is $\hat{y} = 0.9$, the value of loss is given by

$$\mathcal{L}(0.9) = -0\log_2(0.9) - (1-0)\log_2(1-0.9) = -\log_2(0.1) \approx 3.322$$

(d) Now, we calculate the mean loss for 3 examples for the given ground truth and predicted values

$$\begin{aligned}\mathcal{L}(\hat{Y}) &= -\frac{1}{3}\sum_{i=1}^{3} y_i \log_2 \hat{y}_i + (1-y_i)\log_2(1-\hat{y}_i) \\ &= -\frac{1}{3}\big(\log_2(0.1) + \log_2(0.8) + \log_2(0.3)\big) \\ &\approx \frac{1}{3}\big(3.222 + 3.222 + 1.737\big) = \frac{8.181}{3} = 2.727\end{aligned}$$

(e) For a learning rate α and a traininable weight W, the update formula while using L_2-regularization (hyperparameter λ) is given by

$$W_{\textbf{new}} \leftarrow W_{\textbf{old}} - \alpha\left(\frac{\partial L_{BCE}}{\partial W} + 2\lambda W_{\textbf{old}}\right)$$

The L_2-regularization applied in model A penalizes very large weights by adding some proportion of the weights themselves to the gradient update. Hence, we expect model A's weights to be relatively smaller than model B's final trained weights. However, this technique (usually) reduces overfitting, as model A will be encouraged to learn simpler patterns than model B.

(f) KL-divergence is a measure of '*distance*' or difference between two probability distributions. KL-divergence measures the relative entropy of two distributions. On the other hand, cross entropy can be thought of as the total entropy between them. The formula for KL-divergence and cross-entropy are as follows

$$\begin{aligned}D_{\text{KL}}(P||Q) &= \mathbb{E}_{X\sim P}\left[\log\frac{P(X)}{Q(X)}\right] = \mathbb{E}_{X\sim P}[\log P(X) - \log Q(X)] \\ H(P,Q) &= -\mathbb{E}_{X\sim P}[\log Q(X)]\end{aligned}$$

[1] In fact, mean squared error is the KL divergence between the empirical distribution of the data and a Gaussian model.

These quantities are closely related - minimizing the cross-entropy $H(P,Q)$ with respect to Q is equivalent to minimize the KL-divergence $D_{\text{KL}}(P||Q)$. Their relationship is given in the following equations

$$\begin{aligned}
D_{\text{KL}}(P||Q) &= \mathbb{E}_{X\sim P}[\log P(X)] - \mathbb{E}_{X\sim P}[\log Q(X)] \\
&= H(P,Q) + \mathbb{E}_{X\sim P}[\log P(X)] \\
&= H(P,Q) - H(P) \\
\implies H(P,Q) &= H(P) + D_{\text{KL}}(P||Q)
\end{aligned}$$

where Shannon entropy of a distribution P is defined as in class,

$$H(P) = -\mathbb{E}_{X\sim P}[\log P(X)]$$

Question 2.

We are given the following information about the 2-layer neural network for K-class classification problem.

$$\begin{aligned}
z^{[i]} &= W^{[i]}a^{[i-1]} + b^{[i]}, \quad i = 1,2 \quad a^{[0]} = x \\
a^{[1]} &= \textsc{LeakyReLU}\left(z^{[1]}, \alpha = 10^{-2}\right) \\
\hat{y} &= \textsc{Softmax}\left(z^{[2]}\right) = \sigma\left(z^{[2]}\right) \quad \text{(say)} \\
\mathcal{L}(\hat{y}) &= -\sum_{i=1}^{K} y_i \log\left(\hat{y}_i\right)
\end{aligned}$$

(a) Let us assume $z^{[2]}$ is of size $K \times 1$ (since it is a K-class classification problem). Since $z^{[1]}$ is of size $D_a \times 1$, $a^{[1]}$ must be of the same size, and hence, the $W^{[2]}$ must of be size $K \times D_a$. Following this, $b^{[2]}$ must be of size $K \times 1$.

$$z^{[2]}_{(K\times 1)} = W^{[2]}_{(K\times D_a)} a^{[1]}_{(D_a\times 1)} + b^{[2]}_{(K\times 1)}$$

(b) Let us denote the sum of exponentials (normalizing term) of the entries in $z^{[2]}$ by

$$\mathbf{z} = \sum_{j=1}^{K} \exp z_j^{[2]}, \text{ which means } \hat{y} = \sigma\left(z^{[2]}\right) = \frac{\exp z^{[2]}}{\mathbf{z}}$$

We are required to find the partial derivative of the k^{th} entry in $\hat{y}$ with respect to the k^{th} entry in $z^{[2]}$.

$$\begin{aligned}
\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} &= \frac{\partial}{\partial z_k^{[2]}}\left(\sigma\left(z_k^{[2]}\right)\right) = \frac{\partial}{\partial z_k^{[2]}}\left(\frac{\exp z_k^{[2]}}{\mathbf{z}}\right) \\
&= \frac{1}{\mathbf{z}^2}\left(\mathbf{z}\,\frac{\partial}{\partial z_k^{[2]}}\left(\exp z_k^{[2]}\right) - \exp z_k^{[2]}\,\frac{\partial}{\partial z_k^{[2]}}\left(\mathbf{z}\right)\right) \\
&= \frac{1}{\mathbf{z}^2}\left(\mathbf{z}\,\exp z_k^{[2]} - \left(\exp z_k^{[2]}\right)^2\right) = \frac{\exp z_k^{[2]}}{\mathbf{z}} - \left(\frac{\exp z_k^{[2]}}{\mathbf{z}}\right)^2 \\
&= \hat{y}_k - \hat{y}_k^2 = \hat{y}_k\left(1 - \hat{y}_k\right)
\end{aligned}$$

(c) Next, we find the partial derivative of the k^{th} entry in $\hat{y}$ with respect to the i^{th} entry in $z^{[2]}$, given $i \neq k$.

$$\begin{aligned}
\frac{\partial \hat{y}_k}{\partial z_i^{[2]}} &= \frac{\partial}{\partial z_i^{[2]}}\left(\sigma\left(z_k^{[2]}\right)\right) = \frac{\partial}{\partial z_i^{[2]}}\left(\frac{\exp z_k^{[2]}}{\mathbf{z}}\right) \\
&= \frac{1}{\mathbf{z}^2}\left(\mathbf{z}\,\frac{\partial}{\partial z_i^{[2]}}\left(\exp z_k^{[2]}\right) - \exp z_k^{[2]}\,\frac{\partial}{\partial z_i^{[2]}}\left(\mathbf{z}\right)\right) \\
&= \frac{1}{\mathbf{z}^2}\left(0 - \exp z_k^{[2]} \exp z_i^{[2]}\right) = -\frac{\exp z_k^{[2]}}{\mathbf{z}} \cdot \frac{\exp z_i^{[2]}}{\mathbf{z}} \\
&= -\hat{y}_k\hat{y}_i
\end{aligned}$$

(d) We are interested in the partial derivative of the loss with respect to entries in $z^{[2]}$. Using the chain rule, we have

$$\frac{\partial \mathcal{L}}{\partial z_i^{[2]}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i^{[2]}}$$

With the above (given) loss function, we can find

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}\left(-\sum_{j=1}^{K} y_j \log(\hat{y}_j)\right) = -\sum_{j=1}^{K} y_j \frac{\partial}{\partial \hat{y}_i}(\log(\hat{y}_j)) = -\frac{y_i}{\hat{y}_i}$$

Note that as a result from **Question 2.** (b),

$$\frac{\partial \hat{y}_i}{\partial z_i^{[2]}} = \hat{y}_i(1-\hat{y}_i)$$

So, we have

$$\frac{\partial \mathcal{L}}{\partial z_i^{[2]}} = -\frac{y_i}{\hat{y}_i} \cdot \hat{y}_i(1-\hat{y}_i) = -y_i(1-\hat{y}_i)$$

Now, we are given that $y_k = 1$ and $y_i = 0, i \neq k$. So, we get (where $\mathbb{I}$ is an indicator variable)

$$\frac{\partial \mathcal{L}}{\partial z_i^{[2]}} = \mathbb{I}_{\{i=k\}}(\hat{y}_i - 1) = \begin{cases} \hat{y}_k - 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

(e) While implementing the softmax function, we run into the problem of numerical overflow and underflow. Overflow occurs while exponentiating large numbers, when the result exceeds the computer's largest representable float value. In such cases, the result of e^z is estimated to be infinity (`torch.inf` in PyTorch). Conversely, underflow occurs while exponentiating small numbers, when extremely small values are represented by zeros. Rarely, it is also possible that the denominator is extremely small, which makes division unstable.
A common modification to resolve the issue is normalizing the input by max-shifting. In this technique, we subtract the maximum of the vector from all its entries, making all of them non-positive, and at least one entry 0. The output probabilities do not change because the relative difference between the entries in z does not change, but the numberical operations become more stable. So, for a vector z of size m, we get

$$z^* = z - \max_{k=1,2,\ldots,m} z_k$$

$$\sigma(z) = \sigma(z^*) = \frac{\exp z^*}{\sum_{j=1}^{m} \exp z_j^*}$$