DD2424 - Assignment 1 (Bonus)

Summary

I have completed both parts of the bonus assignment exercises. For the first part, I have used the entirety of the training data, implemented training data augmentation, and used a decaying learning rate. I also experimented with a numer of different values for the regularization parameter. For the second part, I have derived the gradients for the multiple binary cross-entropy loss and compared the results when using the sigmoid activation with the corresponding results for the softmax.

Oskar Stigland
DD2424
Spring 2023

Part I - Improving Performance of the Network

For this part, I have experimented with different parameter setting while adding the improvements. I recorded the best results and the (seemingly) most stable training procedure with the following parameter settings:

- `batchN` $= 50$
- `epochsN` $= 50$
- $\lambda = 0.1$
- $\eta = 0.01$
- $\eta_n = 10$
- $p_{\text{flip}} = 0.2$

where the η_n corresponds to the number of epochs per decay of the learning rate. Per the assignment suggestion, I initially used a horizontal flip probability of $p = 0.5$ for the training data, performing a shuffle and flip for every epoch. However, it seems a lower flip probability works equally well and I thus settled for $p = 0.2$. As mentioned, I have also utilized all of the tranining data, reducing the validation data to a smaller subset of the training data with $N_{\text{val}} = 1000$. I also experimented with a number of different settings for the regularization parameter. It would seem $\lambda = 0.1$ is a fairly optimal value for training. For the above parameters, the model reaches an accuracy of 40.76% and across the different improvements it seems that expanding and augmenting the dataset yields the largest benefits in terms of accuracy, i.e. by using the entirety of the dataset with a non-zero flip probability.

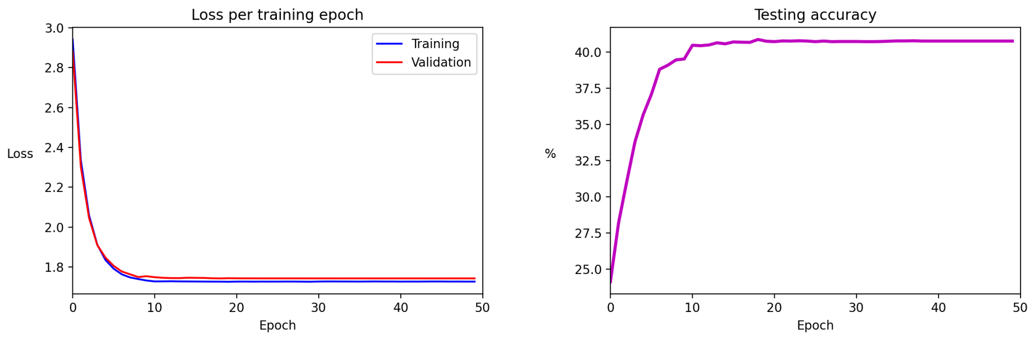


Figure 1: Loss and accuracy for optimal set of parameters

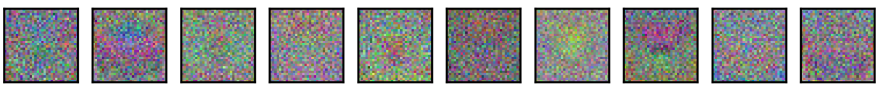


Figure 2: Class-specific weights for optimal set of parameters

Part II - Multiple Binary Cross-Entropy Losses

Obtaining the gradient

In order to obtain the expression for $\partial \ell / \partial \boldsymbol{s}$, we consider the chain rule, where

$$\frac{\partial \ell}{\partial \boldsymbol{s}} = \frac{\partial \ell}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}}$$

First, we considering differentiating the class-specific loss w.r.t. the correspoding probability, s.t. $\ell = -1/K \sum_{k=1}^{K} \ell_k$:

$$\frac{\partial \ell_k}{\partial p_k} = \frac{\partial}{\partial p_k} \left[(1-y_k)\log(1-p_k) + y_k \log(p_k)\right] = \frac{y_k - 1}{1-p_k} + \frac{y_k}{p_k}$$

Second, we recognize that $p_k = \sigma(s_k)$, wherer $\sigma : \mathbb{R} \to [0,1]$ is the logistic sigmoid function, for which we also have that

$$\sigma(s) = \frac{1}{1+e^{-s}} = \frac{e^s}{e^s + 1}$$

as well as $1 - \sigma(s) = \sigma(-s)$ and $\sigma'(s) = \sigma(s)(1-\sigma(s))$. Hence, we get that

$$\begin{aligned}
\frac{\partial \ell_k}{\partial s_k} &= \frac{\partial}{\partial s_k} \left[(1-y_k)\log(1-\sigma(s_k)) + y_k \log(\sigma(s_k))\right] \\
&= \frac{y_k - 1}{1-\sigma(s_k)} \sigma(s_k)(1-\sigma(s_k)) + \frac{y_k}{\sigma(s_k)} \sigma(s_k)(1-\sigma(s_k)) \\
&= (y_k - 1)\sigma(s_k) + y_k(1-\sigma(s_k) \\
&= y_k - \sigma(s_k)
\end{aligned}$$

Hence, the Jacobians we're looking for are given by

$$\begin{aligned}
\frac{\partial \ell}{\partial \boldsymbol{p}} &= -\frac{1}{K}\left[\boldsymbol{y}^T \mathrm{diag}(\boldsymbol{p})^{-1} - (\boldsymbol{1} - \boldsymbol{y})^T \mathrm{diag}(\boldsymbol{1}-\boldsymbol{p})^{-1}\right] \\
\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} &= \mathrm{diag}(\boldsymbol{p})\,\mathrm{diag}(\boldsymbol{1}-\boldsymbol{p})
\end{aligned}$$

such that

$$\begin{aligned}
\frac{\partial \ell}{\partial \boldsymbol{s}} &= -\frac{1}{K}\left[\boldsymbol{y}^T \underbrace{\mathrm{diag}(\boldsymbol{p})^{-1}\mathrm{diag}(\boldsymbol{p})\,\mathrm{diag}(\boldsymbol{1}-\boldsymbol{p})}_{=\mathrm{diag}(\boldsymbol{1}-\boldsymbol{p})} - (\boldsymbol{1}-\boldsymbol{y})^T \underbrace{\mathrm{diag}(\boldsymbol{1}-\boldsymbol{p})^{-1}\mathrm{diag}(\boldsymbol{p})\,\mathrm{diag}(\boldsymbol{1}-\boldsymbol{p})}_{=\mathrm{diag}(\boldsymbol{p})}\right] \\
&= -\frac{1}{K}\left[\boldsymbol{y}^T \mathrm{diag}(\boldsymbol{1}-\boldsymbol{p}) - (\boldsymbol{1}-\boldsymbol{y})^T \mathrm{diag}(\boldsymbol{p})\right] \\
&= -\frac{1}{K}\left[\boldsymbol{y} - \boldsymbol{y}^T \mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p} + \boldsymbol{y}^T \mathrm{diag}(\boldsymbol{p})\right] \\
&= -\frac{1}{K}(\boldsymbol{y} - \boldsymbol{p})
\end{aligned}$$

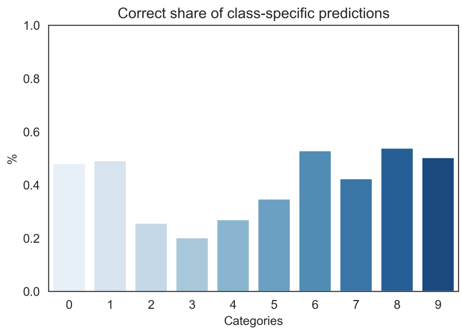
Hence, using the result from the previous loss, we should have that

$$\frac{\partial \ell}{\partial \boldsymbol{W}} = -\frac{1}{K}(\boldsymbol{y}-\boldsymbol{p})\,\boldsymbol{x}^T, \quad \frac{\partial \ell}{\partial \boldsymbol{b}} = -\frac{1}{K}(\boldsymbol{y}-\boldsymbol{p})$$

Results

I implemented the linear classifier with a `sigmoid` activation and the multiple binary cross-entropy loss for a number of parameter settings. The loss, accuracy and class-specific weights have been plotted on the following page. Due to the $1/K$ factor in the loss, I have used a slightly higher learning rate for this part, $\eta = 0.01$ or $\eta = 0.05$, with a slightly modified decay rate, e.g. reducing the learning rate by half every 10 steps. Overall, the results are comparable to those when using the `softmax` activation, but the achieved accuracy is actually lower across all tests compared to the highest accuracy achieved with the `softmax`. However, it does seem that similarly to using the `softmax`, the highest accuracy is achieved when using a moderate regularization parameter - again, in this case $\lambda = 0.1$. In fact, the performance is somewhat worse when using $\lambda = 1.0$ as compared to using $\lambda = 0.0$.

In order to compare the `softmax` and `sigmoid` activations, I trained two separate models with the optimal set of parameters found for each type. The class-specific prediction results and the parameter setting are shown in the plots and table below. The results are in fact very similar. Both models achieve a relatively low accuracy for the category 2 and 3, while they both achieve a relatively high accuracy for cateogories 0, 1, and 6. The `softmax` model also seems to achieve a slightly better result for category 7. The differences, however, are marginal. I wouldn't call this overfitting, since the class-specific accuracies are fairly well-distributed across the classes, with the exception of 2 to 4 - for which both models fail.



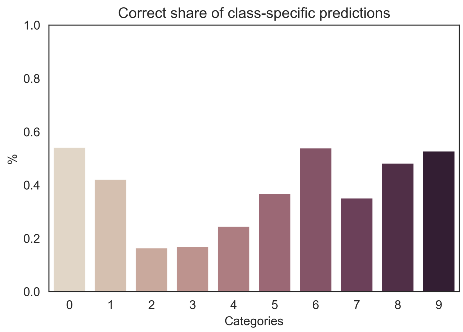


Figure 3: Class-specific accuracy for `softmax` (left) and `sigmoid` (right)

Activation	`batchN`	`epochsN`	η	η_n	λ	P_{flip}
`softmax`	100	50	0.01	10	0.1	0.2
`sigmoid`	50	50	0.01	0	0.1	0.5

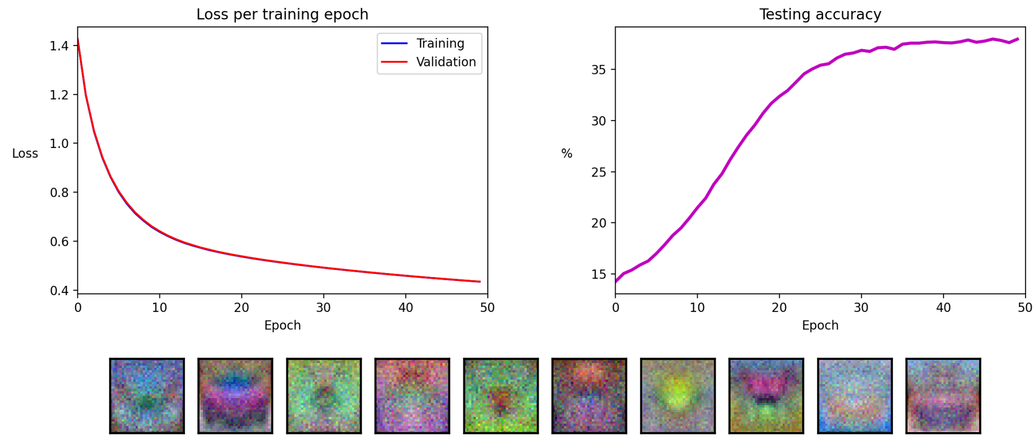


Figure 4: `sigmoid` with $\lambda = 0.1$, $\eta = 0.01$, `batchN` $= 100$, `epochsN` $= 50$, $P_{\text{flip}} = 0.5$

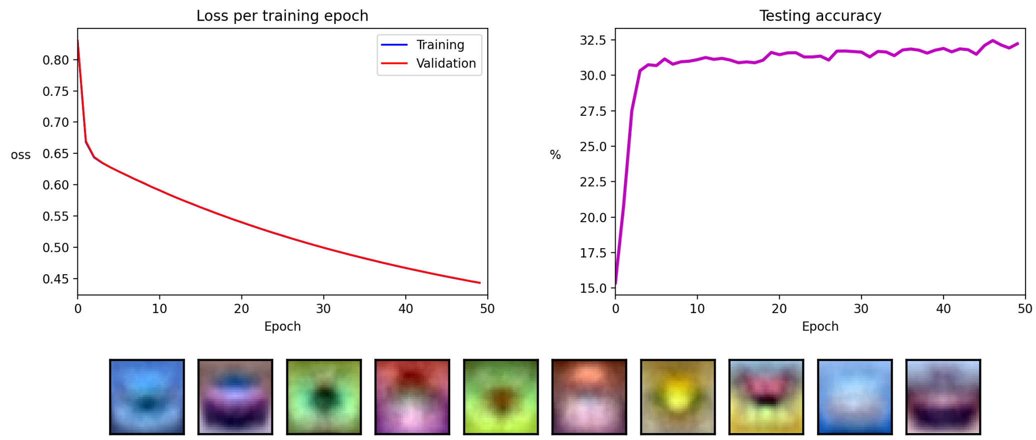


Figure 5: `sigmoid` with $\lambda = 1.0$, $\eta = 0.01$, `batchN` $= 100$, `epochsN` $= 50$, $P_{\text{flip}} = 0.5$

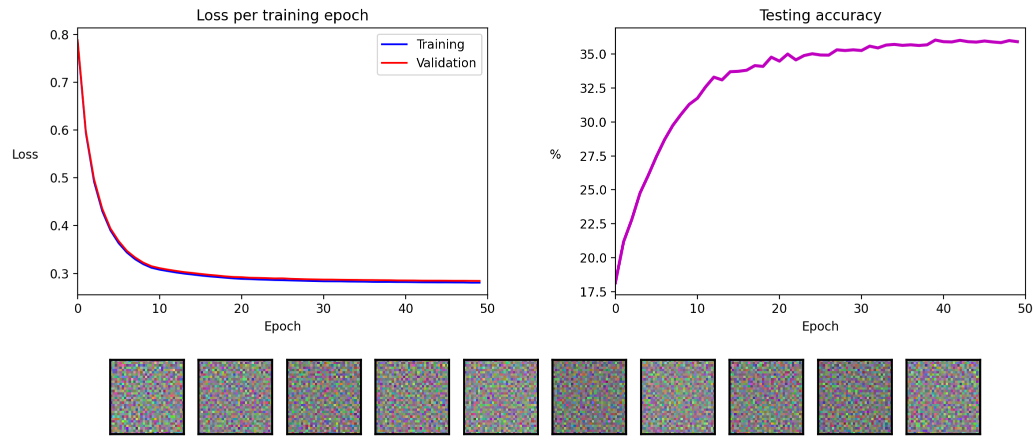


Figure 6: `sigmoid` with $\lambda = 0.0$, $\eta = 0.01$, `batchN` $= 100$, `epochsN` $= 50$, $P_{\text{flip}} = 0.1$