Introduction to Deep Learning for Computer Vision
Assignment 2: Simple Linear Classifier

Tyson Battistella

February 10, 2018

Abstract

This report details the process of creating a simple linear classifier from scratch without the help of deep learning libraries. We derive our own multinomial logistic regression gradients and compare it with a multiclass SVM. We also compare the performance of three feature types: raw RGB data, HSV color histograms, and oriented gradient histograms. We find that the most performant model is one using a histogram of oriented gradients and an SVM without L2 regularization.

1 Multinomial Logistic Regression

As part of this project, we implement our own softmax/cross-entropy layer.

1.1 Softmax

The softmax function outputs a probability distribution whose sum is equal to 1. This property makes it suitable for probabilistic imterpretation in classification tasks. To make the softmax numerically stable, the values in the vector are normalized by multiplying the numerator and denominator with a constant C. C is the value that shifts all of the elements of the vector from negative to zero, and negatives with large exponents saturate to zero rather than infinity, avoiding overflowing.

The softmax is defined as follows:

$$\begin{aligned} p_j &= \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} \\ &= \frac{Ce^{a_i}}{C\sum_{k=1}^{N} e^{a_k}} \\ &= \frac{e^{a_i + \log C}}{\sum_{k=1}^{N} e^{a_k + \log C}} \end{aligned} \tag{1}$$

1.1.1 Derivative of Softmax

For back propogation, we need the derivative of softmax. This is calculated below:

$$\frac{\partial p_j}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} \tag{2}$$

Note that if $i = j$, then $\frac{\partial}{\partial e^{a_j}}$ will be e^{a_j}, otherwise it is 0.
So, if $i = j$ and using the quotient rule, we have:

$$\begin{aligned}\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} &= \frac{e^{a_i}\sum_{k=1}^{N} e^{a_k} - e^{a_j}e^{a_i}}{(\sum_{k=1}^{N} a^{a_k})^2} \\ &= \frac{e^{a_i}(\sum_{k=1}^{N} e^{a_k} - e^{a_j})}{(\sum_{k=1}^{N} a^{a_k})^2} \\ &= \frac{e^{a_j}}{\sum_{k=1}^{N} a^{a_k}} \times \frac{\sum_{k=1}^{N} e^{a_k} - e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \\ &= p_i(1-p_i)\end{aligned} \tag{3}$$

For $i \neq j$ we have $-p_j \cdot p_i$

1.2 Cross Entropy Loss

Cross entropy gives us the distance between what the model thinks the output distribution should be, and what the distribution actually is. It is defined as $H(y,p) = -\sum_i y_i \log(p_i)$.

1.2.1 Derivative of Cross Entropy with Softmax

Using the derivative of the softmax defined above, we can derive the derivative of the cross entropy loss function:

$$\begin{aligned}L &= -\sum_i y_i \log(p_i) \\ \frac{\partial L}{\partial o_i} &= -\sum_k y_k \frac{\partial \log(p_k)}{\partial o_i} \\ &= -\sum_k y_k \frac{\partial \log(p_k)}{\partial p_k} \times \frac{\partial p_k}{\partial o_i} \\ &= -\sum y_k \frac{1}{p_k} \times \frac{\partial p_k}{\partial o_i}\end{aligned}$$

$$= -y_i(1-p_i) - \sum_{k \neq i} y_k \frac{1}{p_k}(-p_k \cdot p_i) \tag{4}$$

$$\begin{aligned}&= -y_i(1-p_i) + \sum_{k \neq 1} y_k \cdot p_i \\ &= -y_i + y_i p_i + \sum_{k \neq 1} y_k \cdot p_i \\ &= p_i(y_i + \sum_{k \neq 1} y_k) - y_i \\ &= p_i(y_i + \sum_{k \neq 1} y_k) - y_i\end{aligned}$$

Since y is a one hot encoded vector for the labels, $\sum_k y_k = 1$ and $y_i + \sum_{k \neq 1} y_k = 1$. So we have,

$$\frac{\partial L}{\partial o_i} = p_i - y_i \tag{5}$$

Table 1: Hyper parameter results

Feature Type	**Model Type**	**Regularization**	**Learning Rate**	**Test Accuracy**	**Avg Validatio**
HOG	SVM	1	.0001	32.02%	35.70%
HOG	Regression	1	.0001	30.44%	35.83%
HOG	SVM	0	.0001	35.67%	36.14%
HOG	Regression	0	.0001	31.39%	35.60%
HOG	SVM	1	0.00001	35.35%	34.95%
HOG	Regression	1	0.00001	22.01%	20.42%
HOG	SVM	0	0.00001	35.44%	35.12%
HOG	Regression	0	0.00001	20.06%	19.04%
Hue	SVM	1	.0001	16.76%	20.20%
Hue	Regression	1	.0001	20.08%	20.87%
Hue	SVM	0	.0001	17.88%	21.09%
Hue	Regression	0	.0001	19.72%	20.78%
Hue	SVM	1	0.000001	21.95%	20.85%
Hue	Regression	1	0.000001	17.58%	16.24%
Hue	SVM	0	0.000001	21.11%	20.93%
Hue	Regression	0	0.000001	17.71%	16.75%

2 Results

After creating a normalization method, HSV histogram extraction, and a linear regression model, the data was ran with a few different hyper parameter configurations. Namely, we tried with and without L2 regularization, and with learning rates of 0.0001 and 0.000001. The results are listed in table 1. The models were validated with 5-fold cross validation.

In conclusion, removing L2 regularization resulted in more accurate results when using a features of histogram of oriented gradients, whereas the histogram of hue values performed better with regularization included. It also appeared that models using linear regression performed better with a larger learning rate, while models using SVM were able to achieve a higher accuracy with a lower learning rate. Interestingly, when using a lower learning rate, the validation accuracy was consistently lower than the testing accuracy. In general, the most accurate model used a histogram of oriented gradients for features, with an SVM and no regularization.