Classifying Handwritten Mathematical Symbols

Yavuz Bakman

June 3, 2020

1 Introduction

Classifiying handwritten numbers is one of the oldest problem in machine learning. Especially, MNIST dataset is such a popular that most of the introduction to classification lectures start with that example. However, classifiying numbers is not enough for the converting mathematical expression into for instance LATEX format. To do that, we also need convert mathematical symbols such as $!, \times, -, +$ into their ground truth.

2 Problem

In the project, we want to solve the first and crucial step of the converting handwritten mathematical expression into LATEX format which is classifiying handwritten mathematical symbols and numbers. We apply various machine learning model with different hyperparameters. Also, we explained the main idea of the Adam optimizer. It only explains the theoritical results of the main Adam paper and does not provide any new theoritical results.

3 Dataset

We used the Kaggle dataset [1] which consists of 100 000 handwritten mathematical symbol images. The images are grayscale and 45x45 jpg format. There are 82 different symbols including all math operators, set operators, basic pre-defined math functions like: log, lim, cos, sin, tan, math symbols like: $\int, \sum, \sqrt{}, \delta$ and more. Original source, that was parsed, extracted and modified is CROHME [2] dataset.

4 Methods

We implemented 4 different methods with various hyperparameters. The best hyperparameter set is selected by the 5-fold validation. Every algorithm will be explained in the corresponding subsections.

4.1 Softmax Regression

The first algorithm we apply is softmax regression. It is the linear model for the multiclass classification model. We flatten the image matrix into vector and multiply each pixel with random weights. Then we sum the result of the multiplication after that we add bias term. Now, all we have to do is minimize the cost function which Cross Entropy Loss. There are 82 classes and minibatch size = 50 so:

$$-\sum_{i=1}^{50}\sum_{l=1}^{82} 1_{y^{(i)}=l} \frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{82} e^{\theta_j^T x^{(i)}}} \tag{1}$$

We use pytorch for the implementation.

4.2 Support Vector Machines (SVM)

In support vector machines, the algorithm try to seperates classes by best margin. Here is the equation that SVM wants optimize:

$$min_{\gamma,w,b}\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

s.t

$$y^i(w^T x^i + b) \geq 1 - \xi_i, i = 1, ..., n$$

$$\xi_i \geq 0, i = 1, ..., n$$

There are 82 classes. We use one vs rest classification technique. Whole

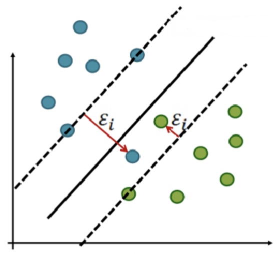

Figure 1: SVM Seperator

implementation is done by the sckit-learn [5].

4.3 Multilayer Perceptron (MLP)

We apply multilayer perceptron model. We use Relu as activation function. Cross Entropy cost function is used. Here is the model: We implemented the model in pytorch [3].

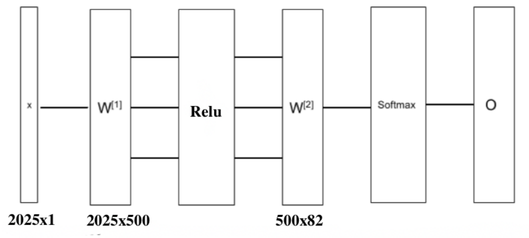

Figure 2: MLP model

4.4 Convolutional Neural Network (CNN)

The most modern tecnique we used is convolutional neural networks. In that model, convolutional layers extract feature from the image by considering neighbour pixels. Also, maxpool layers are used to decrease dimension and number of features that makes model easily trainable. Also, it makes rotational/position invariance feature extraction. We use Cross entropy loss function. We try to use Adam and SGD optimizer . The mathematics behind the Adam optimizer will be explained in section 5. The whole implementation is done by pytorch [3] framework. Here is the model:

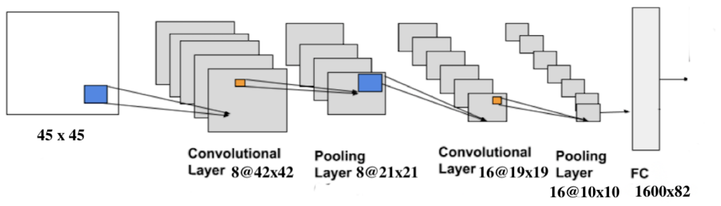

Figure 3: CNN model

5 Mathematics Behind the Algorithm

5.1 Adam Optimizer

Adam suggests an efficient stochastic optimization method that only needs first order gradient. It is computationally fast and convergence of the algorithm is proven. We show the algorithm and explain the main idea behind the algoritm and also we provide a theoritical analysis of Adam's convergence in online convex programming. Whole proof and the algorithm comes from the original Adam paper [6].

5.1.1 The Algorithm

Algorithm: Adam proposed algorithm for stochastic optimization. $g_t 2$ indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise.

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while $\theta_t$ not converged do
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t 2$ (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$(Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return $\theta_t$ (Resulting parameters)
```

Here is the pseudo-code for the algorithm. $f(\theta)$ is the objective function differantiable with respect to θ. Our main goal is to minimize $E(f(\theta))$. $\nabla_\theta f_t(\theta_t)$ denotes for the gradient which is partial derivatives of objective function evaluated at timestep t. m_t and v_t correspond first moment (the mean) and second raw moment (the uncentered variance) of the gradient respectively. These are updated with each iteration. After the rescaling of the gradient, the update happens with learning rate parameter.

5.1.2 The Convergence Analysis of Adam

We analyze the converge of the algorithm in online learning framework.We have unknown sequence of convex cost functions $f_1(\theta), f_2(\theta), ..., f_T(\theta)$. We analyse the algorithm by regret. Here is the definition of regret:

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)] \tag{2}$$

where $\theta^* = argmin_\theta \sum_{t=1}^{T} f_t(\theta_t)$. We show that Adam has $O(\sqrt{T})$ regret bound.For simplicity, we use some definitions: $g_t = \nabla f_t(\theta_t)$, $g_{t,i}$ as the ith element. We define $g_{1:t,i} = [g_{1,i}, g_{2,i} ..., g_{t,i}]$ and also $\gamma = \beta_1^2 / \sqrt{\beta_2}$. The following theorem holds when the learning is decaying at a rate of $t{-}1/2$ and first

moment running average coefficient $\beta_{1,t}$ decay exponentially with λ, that is typically close to 1.
Theorem: Assume that the function f_t has bounded gradients, $||\nabla f_t(\theta)||_2 \leq G, f_t(\theta)||_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded $||\theta_n - \theta_m||_2 \leq D, ||\theta_n - \theta_m||_\infty \leq D_\infty$ for any $m, n \in 1, ..., T$ and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\gamma < 1$. Let $\alpha_t = \alpha/\sqrt{t}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$ Adam achives the following:

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Proof: We put some lemmas for the main proof but we skip the proof of the lemmas because of the space issues of that paper. You can find the proof of the lemmas in the main paper [6].
Lemma 1.: If a function $f : R^d \to R$ is convex, then for all $x, y \in R^d$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Lemma 2.: Let $g_t = \nabla f_t(\theta_t)$ and $g_{1:t}$ be defined as above and bounded, $||g_t||_2 \leq G, ||g||_\infty \leq G_\infty$ then,

$$\sum_{t=1}^{T} \sqrt{g_{t,i}^2/t} \leq 2G_\infty ||g_{1:T,i}||_2$$

Lemma 3.: Let $\gamma = \beta_1^2/\sqrt{\beta_2}$. For $\beta_1, \beta_2 \in [0, 1)$ that satisfy $\beta_1^2/\sqrt{\beta_2} < 1$ and bounded $g_t, ||g_t||_2 \leq G, ||g_t||_\infty \leq G_\infty$ the following inequality holds:

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} \leq \frac{2}{1-\gamma} \frac{1}{\sqrt{1-\beta_2}} ||g_{1:T,i}||_2$$

Proof of the Theorem.: By first lemma:

$$f_t(\theta_t) - f_t(\theta^*) \leq g_t^T(\theta_t - \theta^*) \leq \sum_{i=1}^{d} g_{t,i}(\theta_{i,d} - \theta_i^*)$$

By the update rule from the Algorithm:

$$\begin{aligned} \theta_{t+1} &= \theta t - \alpha_t \hat{m}_t/\sqrt{\hat{v}_t} \\ &= \theta_t - \frac{\alpha_t}{1-\beta_1^t} \left(\frac{\beta_{1,t}}{\sqrt{\hat{v}_t}} m_{t-1} + \frac{(1-\beta_{1,t})}{\sqrt{\hat{v}_t}}\right) g_t \end{aligned}$$

We focus on the ith dimension of the parameter vector $\theta \in R^d$. Subtract the scalar θ^* and square both sides of the above update rule, we have:

$$(\theta_{t+1} - \theta_i^*)^2 = (\theta_t - \theta_i^*)^2 - \frac{2\alpha_t}{1-\beta_1^t} \left(\frac{\beta_{1,t}}{\sqrt{\hat{v}_{t,i}}} m_{t-1,i} + (1 - \frac{\beta_{1,t}}{\sqrt{\hat{v}_{t,i}}} g_{t,i}\right)(\theta_t - \theta_i^*) + \alpha_t^2 \left(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}}\right)^2$$

We can rearrange the equation by Young's inequality, $ab \leq a^2/2 + b^2/2$. Also, it can be shown that $\sqrt{\hat{v}_{t,i}} = \sqrt{\sum_{j=1}^{t}(1-\beta_2)\beta_2^{t-j}g_{j,i}^2}/\sqrt{1-\beta_2^t} \leq ||g_{1:t,i}||$ and $\beta_{1,t} \leq \beta_1$ Then:

$$
\begin{aligned}
g_{t,i}(\theta_{t,i} - \theta_i^*) &= \frac{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2\alpha_t(1-\beta_{1,t})}((\theta_{t,i} - \theta_t^*)^2 - (\theta_{t+1,i} - \theta_i^*)^2) \\
&+ \frac{\beta_{1,t}\hat{v}_{t-1,i}^{1/4}}{(1-\beta_{1,t})\sqrt{\alpha_{t-1}}}(\theta *_i - \theta_{t,i})\sqrt{\alpha_{t-1}}\frac{m_{t-1,i}}{\hat{v}_{t-1,i}^{1/4}} + \frac{\alpha_t(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2(1-\beta_{1,t})}(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}})^2 \\
&\leq \frac{1}{2\alpha_t(1-\beta_1)}((\theta_{t,i} - \theta_t^*)^2 - (\theta_{t+1,i} - \theta_i^*)^2)\sqrt{\hat{v}_{t,i}} + \frac{\beta_{1,t}}{2\alpha_{t-1}(1-\beta_{1,t})}(\theta_{t,i} - \theta_t^*)^2\sqrt{\hat{v}_{t-1,i}} \\
&+ \frac{\beta_1\alpha_{t-1}}{2(1-\beta_1)}\frac{m_{t-1,i}^2}{\sqrt{\hat{v}_{t-1,i}}} + \frac{\alpha_t\hat{m}_{t,i}^2}{2(1-\beta_1)\sqrt{\hat{v}_{t,i}}}
\end{aligned}
$$

We apply lemma 3 to the above inequality and derive the regret bound by summing across all the dimensions for $i \in 1,..,d$ in the upper bound of $f_t(\theta_t) - f_t(\theta^*)$ adn the sequence of convex functions for $t \in 1, ...T$:

$$
\begin{aligned}
R(T) &\leq \sum_{i=1}^{d}\frac{1}{2\alpha_1(1-\beta_1)}(\theta_{1,i} - \theta_i^*)^2\sqrt{\hat{v}_{1,i}} + \sum_{i=1}^{d}\sum_{t=2}^{T}\frac{1}{2(1-\beta_1)}(\theta_{t,i} - \theta_i^*)^2(\frac{\sqrt{\hat{v}_{t,i}}}{\alpha_t} - \frac{\sqrt{\hat{v}_{t-1,i}}}{\alpha_{t-1}}) \\
&+ \frac{\beta_1\alpha G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}||g_{1:T,}||_2 + \frac{\alpha G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}||g_{1:T,}||_2 \\
&+ \sum_{i=1}^{d}\sum_{t=1}^{T}\frac{\beta_{1,t}}{2\alpha_t(1-\beta_{1,t})}(\theta_i^* - \theta_{t,i})^2\sqrt{\hat{v}_{t,i}}
\end{aligned}
$$

From the assumption, $||\theta_t - \theta^*|| \leq D||\theta_m - \theta_n|| \leq D_\infty$ we have:

$$
\begin{aligned}
R(T) &\leq \frac{D^2}{2\alpha(1-\beta_1)}\sum_{i=1}^{d}\sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}||g_{1:T,i}||_2 + \frac{D_\infty^2}{2\alpha}\sum_{i=1}^{d}\sum_{t=1}^{t}\frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t\hat{v}_{t,i}} \\
&\leq \frac{D^2}{2\alpha(1-\beta_1)}\sum_{i=1}^{d}\sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}||g_{1:T,i}||_2 \\
&+ \frac{D_\infty^2 G_\infty\sqrt{1-\beta_2}}{2\alpha}\sum_{i=1}^{d}\sum_{t=1}^{t}\frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t}
\end{aligned}
$$

For the last term, we apply geometric series upper bound:

$$\begin{aligned}\sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t} &\leq \sum_{t=1}^{t} \frac{\lambda^{t-1}\sqrt{t}}{(1-\beta_1)}\sqrt{t} \\ &\leq \sum_{t=1}^{t} \frac{\lambda^{t-1}\sqrt{t}}{(1-\beta_1)} t \\ &\leq \sum_{t=1}^{t} \frac{1}{(1-\beta_1)(1-\lambda)^2}\end{aligned}$$

Now we reach the following inequation and we are done:

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)}\sum_{i=1}^{d}\sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}||g_{1:T,i}||_2 + \sum_{i=1}^{d}\frac{D_\infty^2 G_\infty\sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

6 Results

We share the results of the algorithms in the dataset. 80 percent of the data is used as training and remaining is test. We plot the test accuracy, test loss and training loss vs epoch number.

6.1 Graphs

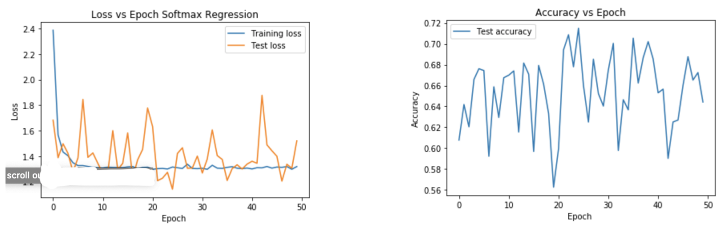

Figure 4: Softmax Regression Graphs

Algorithms	Best Accuracy
Softmax Regression	0.655
SVM	0.813
MLP	0.864
CNN	0.943

Table 1: Accuracy Table

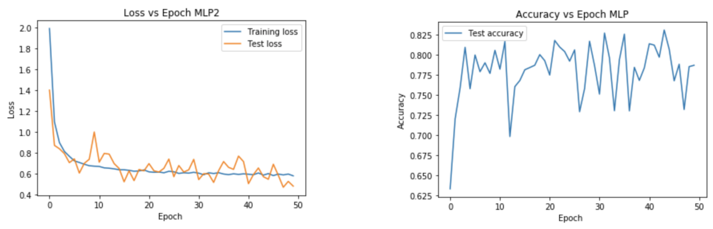

Figure 5: MLP Graphs

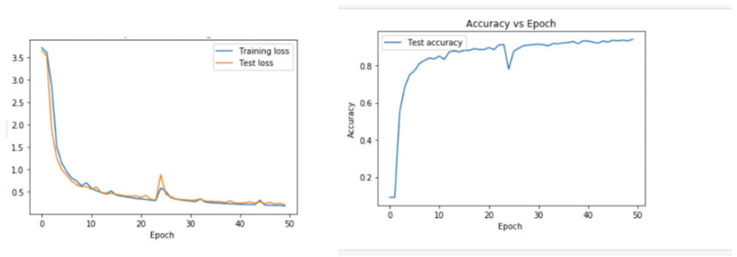

Figure 6: CNN Graphs

6.2 Discussion

The results of the algorithms are expected. As softmax regression is only a linear model and does not try to make best seperation like it has minimum accuracy. SVM makes the linear seperation better than softmax regression then it has better performance. However, it is still a linear mode. The iideal classification function is propably more complicated than a linear function. MLP has non-linearity by non-linear activation function which is Relu. Therefore, it can classify more than a linear model. However, it stills ignore the neighbour relationship of each pixel. It decreases the performance obviously. CNN handles this problem and it considers other near pixels while extraction feauture from the image. Therefore, it achieves the best performance among all models.

7 Conclusion

In the project, we implemented the classification of handwritten symbols and explains the Adam optimizer algorithm. Whole implementation can be found in our github page [4].

References

[1] https://www.kaggle.com/xainano/handwrittenmathsymbols

[2] http://www.isical.ac.in/ crohme/index.html

[3] https://pytorch.org

[4] https://github.com/Ybakman/Classifying-Handwritten-Mathematical-Symbols

[5] https://scikit-learn.org

[6] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).