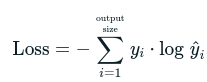
# Introduction to Loss Functions

The loss-function is used by the machines for learning. It's the way of determining how good the certain algorithms models data. If the forecasts are too far off from the actual findings, the loss-function will return the extremely high number. The loss-function tends to minimize prediction-error over time by providing the optimization-function. In a machine-learning, there is no such thing as the one-size-fits-all loss-functions. The kind of a machine-learning method used, a simplicity of computing derivatives, including, to the certain extent, a percentage of the outliers in a data-set all play a role in selecting the loss-function for the certain task.

Depends on a type of the learning tasks we're concerned with, loss-function can be divided into the two main categories: regression-loss as well as classification-loss. We strive to anticipate outcome from the set of a limited categorical-values in the classification. For example, we have given the big dataset of photographs of a handwritten number, categorize these into each of 0–9 numbers. As we have discussed that we have different types of loss functions, let’s have a look some of them here.

## 1 Categorical cross entropy

In a multi-class classification problem, the categorical cross-entropy is loss-function. These are problems whereby the example merely can fit into one of numerous different categories, and a model must choose one. Its formal purpose is to calculate a difference in between the two probabilistic distributions. A categorical cross-entropy loss-function computes the example's losses by adding a following values:



Where in [hat y] is a model’s output [ith] scalars value, [y i] is a corresponding targeted value, and outcome size is a number of scalars values in a model’s output. It is useful for the multi-class classification problems.

## 2 Sparse categorical cross entropy

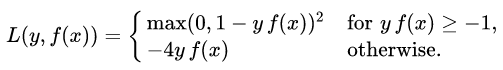
The structure of the true-labels is an only distinction in between the sparse-categorical-cross-entropy versus categorical-cross-entropy. While dealing with the single-label, multi-classes classification tasks, labels for every data input are essentially exclusionary, implying that every data-entry can merely belong to single class.

## 3-Huber loss

A Huber-loss is the loss-function in statistics which is not highly sensitive to the outliers in a data than squared errors loss. It is utilized in the robust regression. A categorization variation is also employed on occasion. Let's start with Huber loss. Mean squared error is more susceptible to anomalies in a data versus Huber loss. The Huber’s-loss equation is shown below.



**a-Variant of classification:** A changed Huber-loss is version of Huber-loss that is occasionally utilized for categorization applications. A-n altered Huber-loss is described as[6] given the prediction displayed style f(x) (the real-valued classification rating) and the true binary-classes label displayed style y belongs to {+1,-1}. A hinge-loss utilized by the support-vector-machines is known as displayed style max (0,1-y, f(x)); the quadratic function smoothing the hinge-loss is a generalization of L. This is best in classifications statistical robustness, M estimation and additive modeling.



# Optimizers

Most of peoples may be employing different optimizers when start training their neural networks without realizing that they are doing so. These optimizers are techniques or approaches that adjust the characteristics of our neural networks, for example weights as well as learning-rate, to minimize losses.

An optimizer we utilize determine how we should alter our neural networks training weights and learning rates to minimization losses. Optimization techniques or methods are in charge of lowering losses and delivering a most precise finding. Let’s have a look on different types of optimizers in the machine learning.

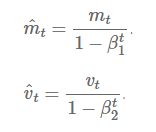
## stochastic gradient decent

The Gradient-Descent is a version of this game. It attempts to modify the parameters of model quite frequently. After every training example loss has been computed, a model’s parameters are changed. As a result, if a dataset has 1000 entries, SGD would upgrade a model’s parameter 1000 times in single dataset iteration, rather than once as with the Gradient-Descent.



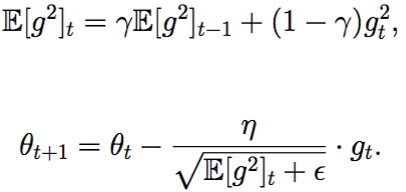
## 2 ADAM

The Adam (Adaptive Moments Estimation) operates both first-order and second-orders momentums. The Adams instinct is which we would not like to move too quickly only to hop over a minimum; instead, we like to slow down the little to allow for a more attentive exploration. Adam preserves the exponentially decaying averaged of past-gradient M in additional to the exponentially decaying-averaged of past square-gradients like a AdaDelta (t).



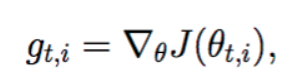
## 3 AdaDalta

It's the AdaGrads extension that aims to solve declining the learning-rate difficulty. AdaDelta confines a window of collected pasts gradients to a defined dimension w, rather than aggregating every previously accumulate gradients. Rather than the total of each gradient, the exponential moving averaged is employed in this case.



## 4 ADAGRAD

A learning-rate is a constant for complete parameters as well as for every cycle, which is single of downsides of each optimizers discussed. The optimizer alters the rate of a learning. It alters learning rate " for every parameter and the time step 't'. It's the second-order optimization technique of the type. It is based on the error functions derivative.



# Experiment

## Dataset

We used the Fashion MNIST dataset in this report for the analysis of learning algorithm. The Fashion MNIST dataset was downloaded from Kaggle in CSV format. The download dataset contains the two files, one for training of the model and second for the testing of the model. The training and testing files (set) contain the 60k and 10k samples respectively. Basically, the samples are based on the pixel value of different fashion images. Each sample in both sets have the 784-pixel values or features that were extracted by flatting the matrix values of image. The Fashion MNIST dataset images belong to ten different categories/ classes including the T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot.

## Data Exploration and Preprocessing

For the understanding of the dataset, firstly the class distribution of the dataset was evaluated. The bar chart of training and testing data revealed that the selected dataset is class balanced dataset in which each class contained the six thousand images. The bar chart of class samples in training set in shown in Figure 1. Next, the NAN test was conducted to find the null values in the dataset but the dataset didn’t find any null value.

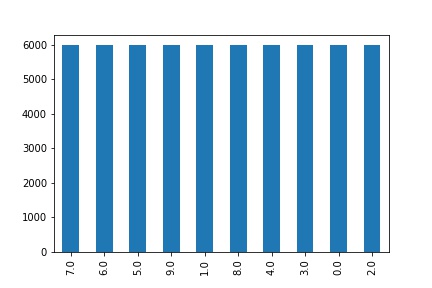


Figure 1: Sample distribution in training set.

In the preprocessing phase, we convert the sample vector into 2D matrix. As each sample contain the 784 feature values that are completely divisible in 28 vectors, that’s why we convert each sample into 28x28 size matrix/image. To feasibly pass the images in training model, all the images were reshaped with dept=1. After the preprocessing of the dataset, all the images were in shape of 28x28x1.

## Model

A convolutional 2D model was created for the classification of Fashion images. The convolutional model was based on the two convolutional layers with the 32 and 64 filter of shape 3x3 and ReLU activation function. Both layers followed by the MaxPooling, Batch Normalization and dropout layer. A 2x2 filter size was used for MaxPooling layer with stride=1 and 0.1 dropout rate for dropout layer. After the convolutional layers, a flatten layer was used to convert the matrix data into vector for classification. Lastly a dense layer with the ten neurons was used as output layer for the classification of images. The complete architecture of convolutional model is showed in Figure 2.

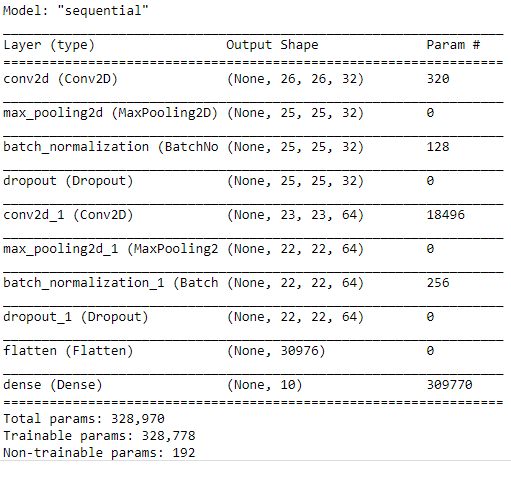


Figure 2: CNN model architecture.

## Experiment 1

We began our experiments with the training of the developed model. We used the training set of Fashion MNIST dataset for the training of the model. The model was trained with the 0.001 learning rate and 50 batch size. Training was performed with the Adam, SGD, AdaGrads, and AdaDelta optimizers. ALL the optimizers were used with the loss functions of Categorical-cross-entropy, sparse-categorical-cross-entropy, and Huber loss.

In the combination of optimizer and loss functions, total 12 combinations were used for the training of the model. Categorical-cross-entropy did not provide the satisfactory results with all the optimizers as shown in Figure 3. While the Huber lose showed the satisfactory results with all the optimizers. The Figure 3 showed that the Categorical-cross-entropy loss functions is not suitable to train the model as the accuracy and the loss of the model with this loss function remains same or increase during the training. On the other hand, Figure 3 illustrate that the Categorical cross entropy showed the greedy behavior to achieve the goal. The training and validation accuracy of the models with Categorical Cross Entropy continuously fluctuate with all the optimizers. The continues increase of accuracy and fall of loss with Huber loss function showed that it perfectly calculates the lose and help the optimizer to optimize the weights of the models successfully. The figure 3 showed that Huber loss function continuously try to achieve the accuracy by minimizing the loss with optimizer and trying to best fit the model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Loss/optimizer | AdaGrads | SGD | Adam | AdaDelta |
| Sparse categorical cross entropy |  |  |  |  |
| categorical cross entropy |  |  |  |  |
| Huber loss |  |  |  |  |

Figure 3: Accuracy and Loss plot of trained models.

As the experiment 1 showed the different behaviors with selected optimizers and loss functions, the resultant performance of each combination is also different. We calculate the accuracy of each model to evaluate the performance of each model. The Table 1 showed that the Huber loss function truly calculate the loss and help all the optimizers to optimize the weights of the CNN. Ultimately, the accuracy of all models with Huber loss function is satisfactory for deployment.

Table 1: Accuracy Score of all trained models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Loss/optimizer | AdaGrads | SGD | Adam | AdaDelta |
| Sparse categorical cross entropy | 0.4140 | 0.1027 | 0.2537 | 0.6416 |
| categorical cross entropy | 0.2738 | 0.1000 | 0.2250 | 0.0715 |
| Huber loss | 0.8522 | 0.8920 | 0.9157 | 0.6826 |

The same behavior was showed by the convergence rate of the trained models in Table 2. The Categorical and sparse categorical cross entropy showed the negative convergence rate while the Huber loss showed the even better positive convergence rate. The convergence rate of all the trained models is shown in Table 2.

Table 2: Convergence rate of trained models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Loss/optimizer | AdaGrads | SGD | Adam | AdaDelta |
| Sparse categorical cross entropy | 0.0168 | 0.3102 | 0.0729 | 0.4121 |
| categorical cross entropy | 0.0026 | 0.1331 | 0.1381 | 0.0272 |
| Huber loss | 0.0145 | 0.0368 | 0.0329 | 0.0233 |

# Discussion

In the proposed study, we use the CNN model for the classification of fashion images. For the training of the CNN, three different loss functions and four optimizers were used. All the combinations of optimizers and loss functions were used for the training of the model. Training results showed the three types of behavior: static, fluctuated and continuous. With all the optimizers, Sparse Categorical Cross entropy showed the static behavior that means that this loss function unable to provide the true loss to optimizers. Secondly, the Categorical Cross Entropy showed the fluctuated behavior that means that the optimizers with this loss functions try to best fit the model greedily. Lastly Huber loss function showed the continuous behavior that means that it truly calculates the loss and help the optimizers to truly optimize the performance of the model. The convergence rate is also positive only for Huber loss function. By seeing the behavior and accuracy performance of all optimizers and loss functions, we hypothesize that the Huber loss function with AdaGrads, SGD and Adam optimizer id ideal for the classification of Fashion images.