# MultiNomial Regression

Neda Keivandarian

## Train the MNR

- With No Regularization
- Regularized MNR

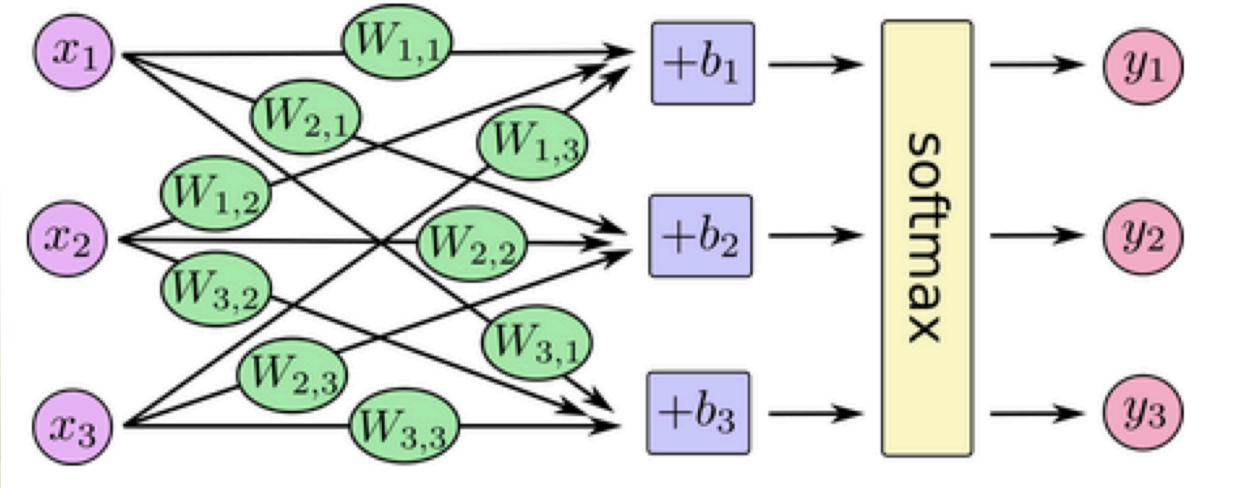
## Behavior of MNR with

- Gradient Descent
- Stochastic Gradient Descent

# What is MNR and How it works?

The multinomial logistic regression algorithm is an extension to the logistic regression model that involves changing the loss function to cross-entropy loss and predict probability distribution to a multinomial probability distribution to support multi-class classification problems.

Below is the procedure of MNR:



## 6 important steps to model MNR

- **Step-1**: Initialize the weight matrix and bias values with zero or a small random values.

- **Step-2**: For each class of K we compute a linear combination of the input features and the weight vector of class K, which is for each training example compute a score for each class. For class K and input vector $x_{(i)}$ we have:

$$s_k(x_{(i)}) = w^T_k \cdot x_{(i)} + b_k$$

In this equation $\cdot$ is the dot product and $w_{(k)}$ the weight vector of class K. So we can find and compute the s which is scores for all classes and training examples in parallel, using vectorization and broadcasting:

$$S = X \cdot W^T + b$$

which X is a matrix of data that has $n_{samples}$ and $n_{features}$ that holds all training examples, and W is a matrix in shape of $n_{classes}$ and $n_{features}$ that holds the weight vector for each class.

- **Step-3**: Apply the softmax function for activation to transform the scores into probabilities.
  The probability that an input vector $x_{(i)}$ belongs to class k is given by:

$$\hat{p}_k(\boldsymbol{x}_{(i)}) = \frac{\exp(s_k(\boldsymbol{x}_{(i)}))}{\sum_{j=1}^{K} \exp(s_j(\boldsymbol{x}_{(i)}))}$$

  In this step we have to perform all the formulas for all classes and our training examples when we using vectorization. We can see the class predicted by this model for $x_{(i)}$ is then simply the class with the highest probability.

- **Step-4**: In this step we should calculate the cost over the whole training set. The result that we expected from this step is our model predict a high probability for our targeted class and the lowest probability for other classes. So it can use the cross entropy loss function:

$$h(\boldsymbol{W}, b) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left[ y_k^{(i)} \log(\hat{p}_k^{(i)}) \right]$$

  Here we use one-hot encoding, because if we used numerical categories 1,2,3,... we would impute ordinal. So $y^{(i)}$ is 1 for the targeted class and for the other $x^{(i)}$ for k classes $y^{(i)}$ should be 0.

- **Step-5**: In this step we need to compute gradient of the cost function for each weight vector and bias: for class k we have:

$$\nabla_{\boldsymbol{w}_k} h(\boldsymbol{W}, b) = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}_{(i)} \left[ \hat{p}_k^{(i)} - y_k^{(i)} \right]$$

- **Step-6**: Here we just need to update biases and weights for all the classes of k an η is my learning rate or step length:

$$\boldsymbol{w}_k = \boldsymbol{w}_k - \eta \nabla_{\boldsymbol{w}_k} h$$
$$b_k = b_k - \eta \nabla_{b_k} h$$

## What are the Experimental Results ?

- Model MNR with No Regularization
- Regularized MNR
- MNR with Gradient Descent Vs Stochastic Gradient Descent

## Model MNR with No Regularization :

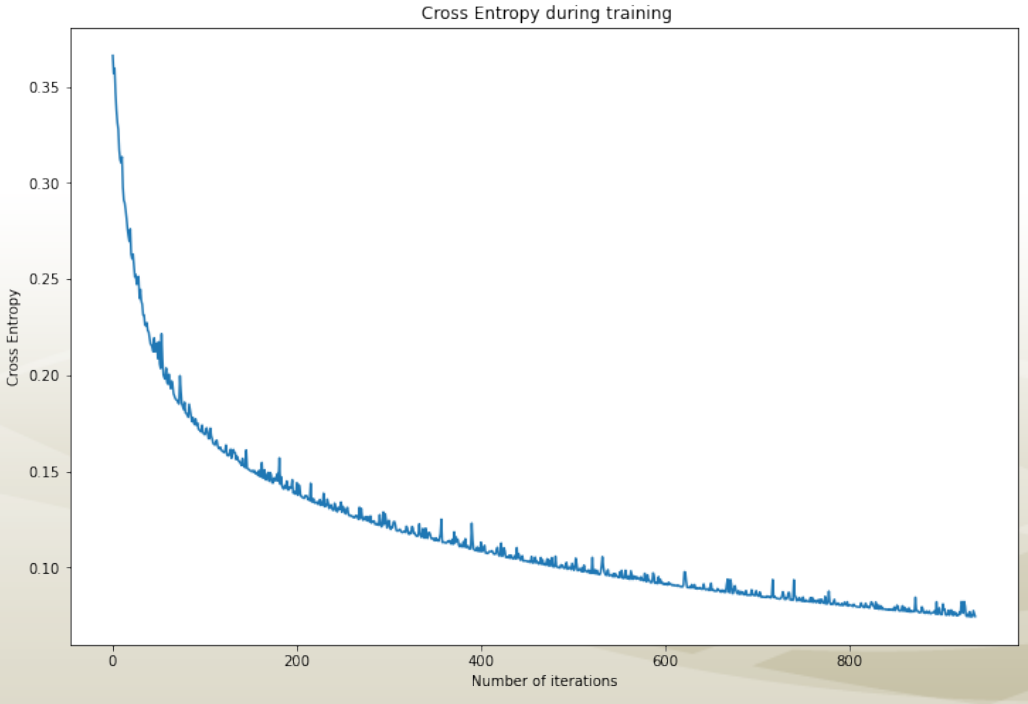
Following the above steps and train MNR with $\lambda = 0$ without regularization and show the obtained decision regions

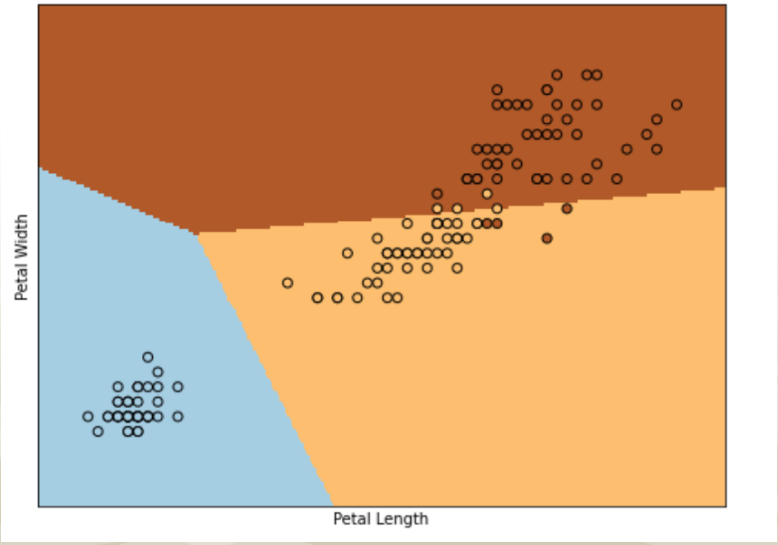
To implement gradient descent with backtracking, the maximum learning rate is 300 and stopping the condition when $d_{ACE}/d_w < 0.01$.

- Training error for 120 samples is 0.03

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$



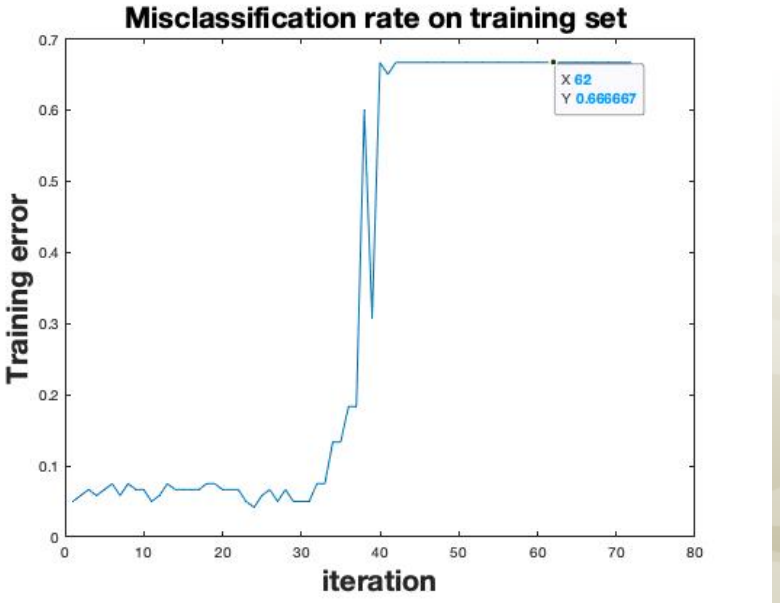



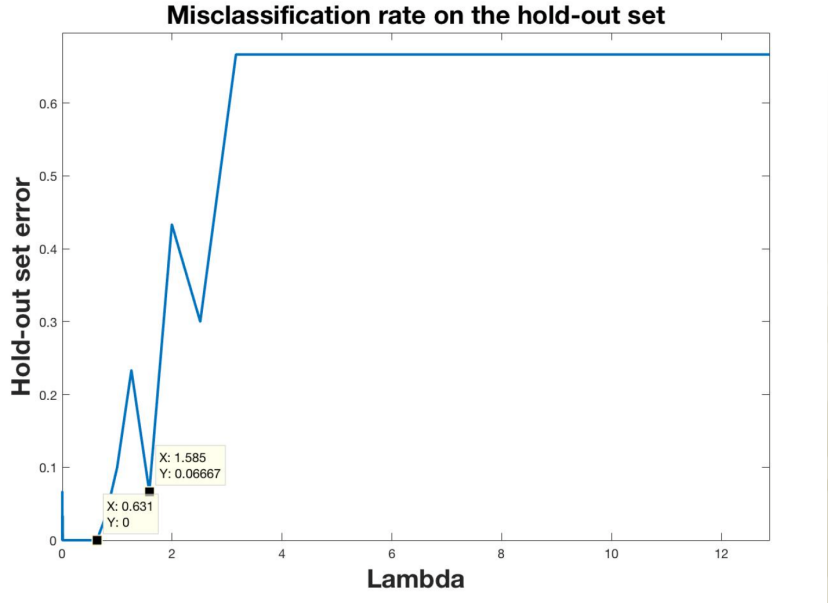
# Regularized MNR

First figure depicts misclassification rate for training set. As we expect this rate is going up as the value of λ increases, because small values for all parameters correspond to simpler model and less prone to overfitting. The maximum rate would be 0.6667.

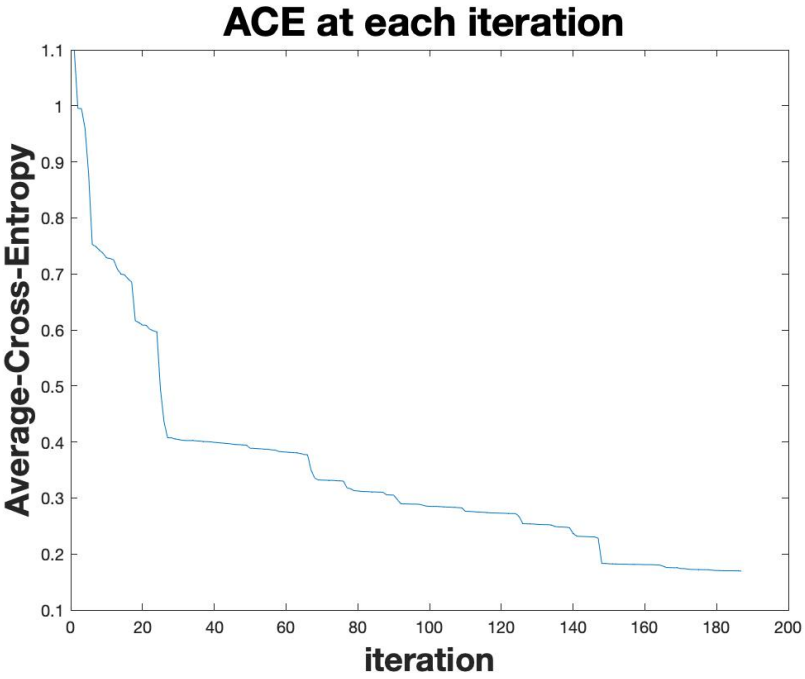
By using non-linear transformation, we can use MNR with transformed data and by picking values of $\lambda = 10^k$, where k = -7:0.2:7, decision boundary is not linear anymore.

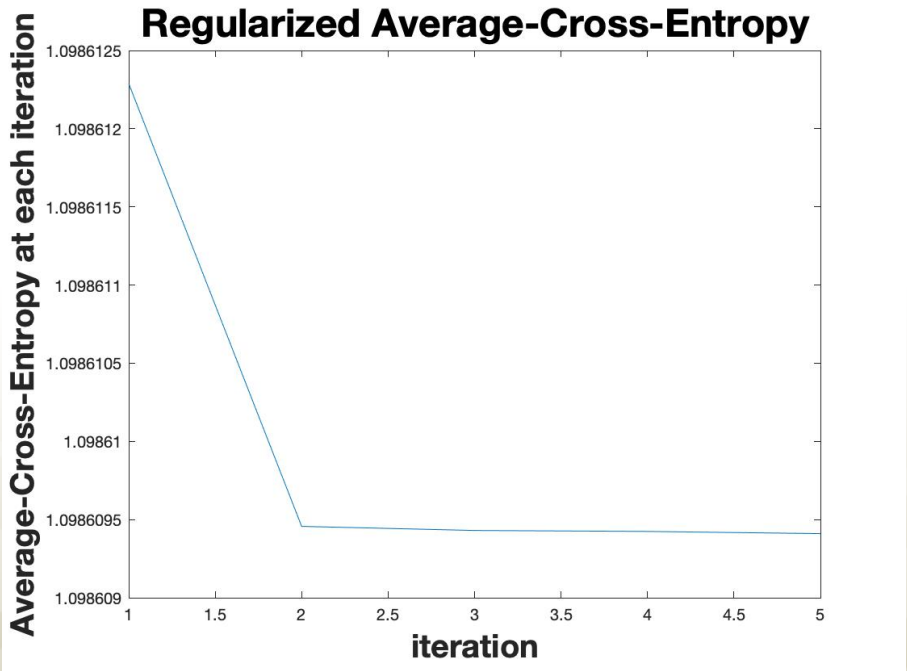
In our example since we use the same portion of each class we will get constant error of 0.66 Misclassification rate of the resulting model on the hold-out set represented as following, the model exhibiting the least error as the champion MNR model was gained for λ between 0 to 0.631. In these cases, the least error is equal to zero.



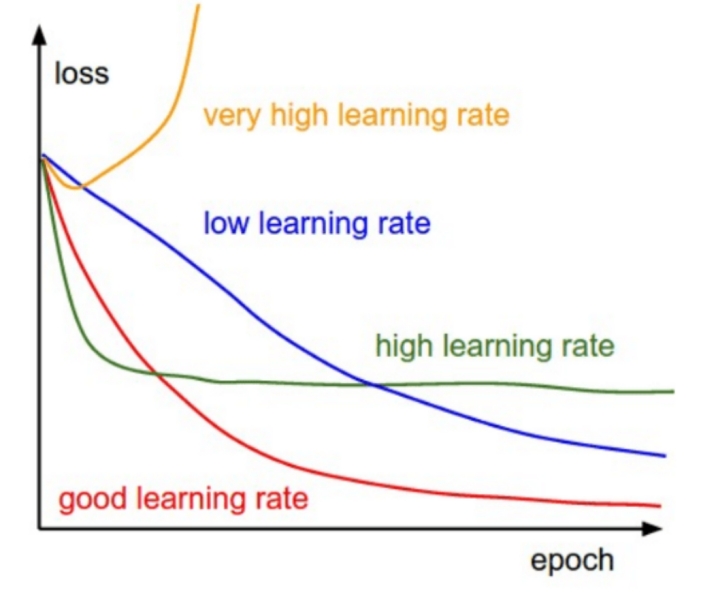
Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.
The second figure shows average cross entropy function is monotonically decreasing by non-linear transformation.



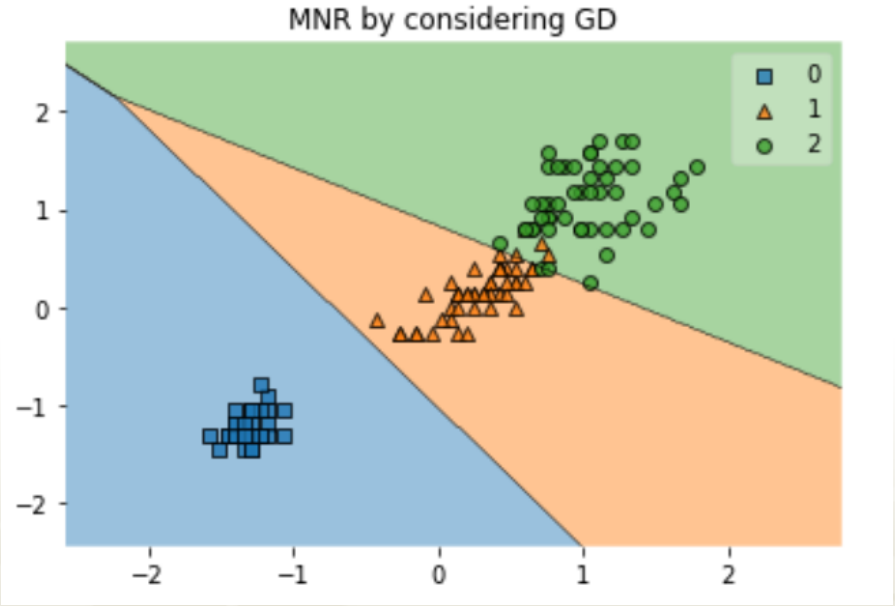



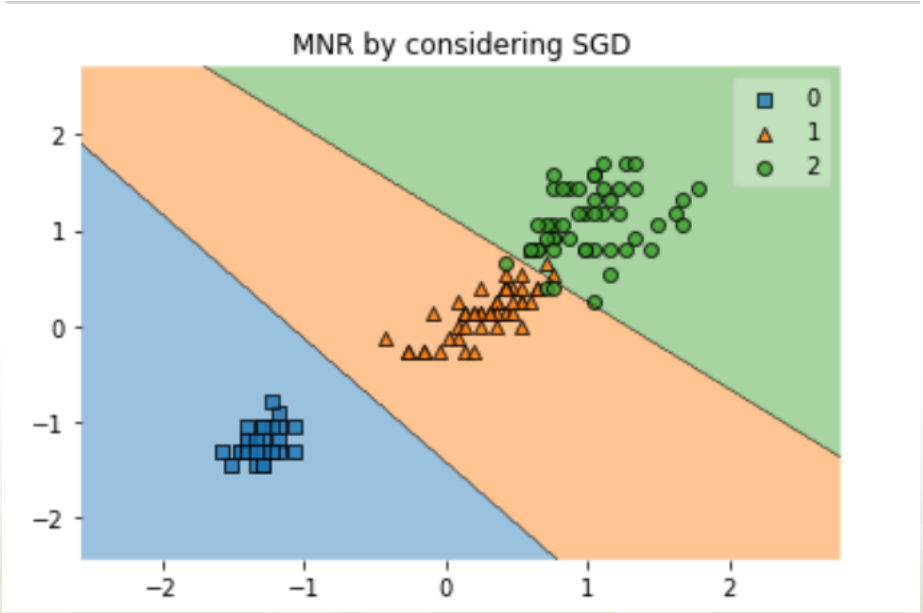
## Gradient Descent Vs Stochastic Gradient Descent

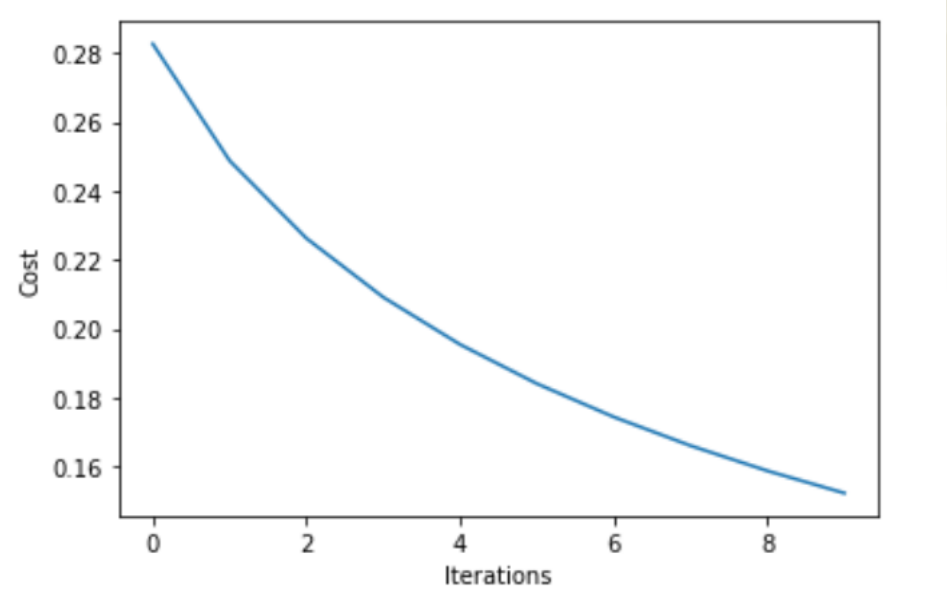
Both algorithms are quite similar. The only difference comes while iterating. In Gradient Descent, we consider all the points in calculating loss and derivative, while in Stochastic gradient descent, we use single point in loss function and its derivative randomly.



# Gradient Descent Vs Stochastic Gradient Descent

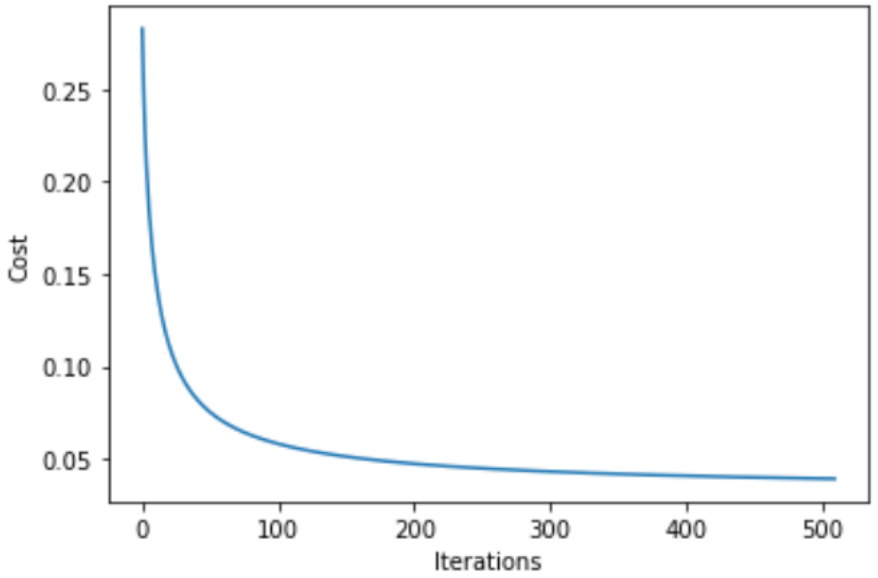





Cross-Entropy by having L2 regularization and considering Gradient Descent

Cross-Entropy by having L2 regularization and considering Stochastic Gradient Descent
for 500 epochs==> which has smoother softmax



## Future Work

- Adversarial Example:

  Adversarial examples are specialized inputs created with the purpose of confusing a neural network, resulting in the misclassification of a given input. These notorious inputs are indistinguishable to the human eye, but cause the network to fail to identify the contents of the image.

- Projected Gradient Descent or PGD:

  The PGD attack is a white-box attack which means the attacker has access to the model gradients i.e. the attacker has a copy of your model's weights.
  PGD attempts to find the perturbation that maximizes the loss of a model on a particular input while keeping the size of the perturbation smaller than a specified amount referred to as epsilon.

- So I want to use MNR with PGD attack for adversarial example.

# Thank You
# Any Questions ?