Loss functions &
Model Selection and Evaluation

- Neural networks are trained using stochastic gradient descent and require that you choose a loss function when designing and configuring your model.
- We need to discover the role of loss and loss functions in training deep learning neural networks and how to choose the right loss function for your predictive modeling problems.

say you are on the top of a hill and need to climb down. How do you decide where to walk towards

Look around to see all the possible paths
Reject the ones going up. This is because these paths would actually cost more energy and make task even more difficult
Finally, take the path that I ***think*** has the most slope downhill

A loss function maps decisions to their associated costs.

In supervised machine learning algorithms, we want to minimize the error for each training example during the learning process. This is done using some optimization strategies like gradient descent. *And this error comes from the loss function*

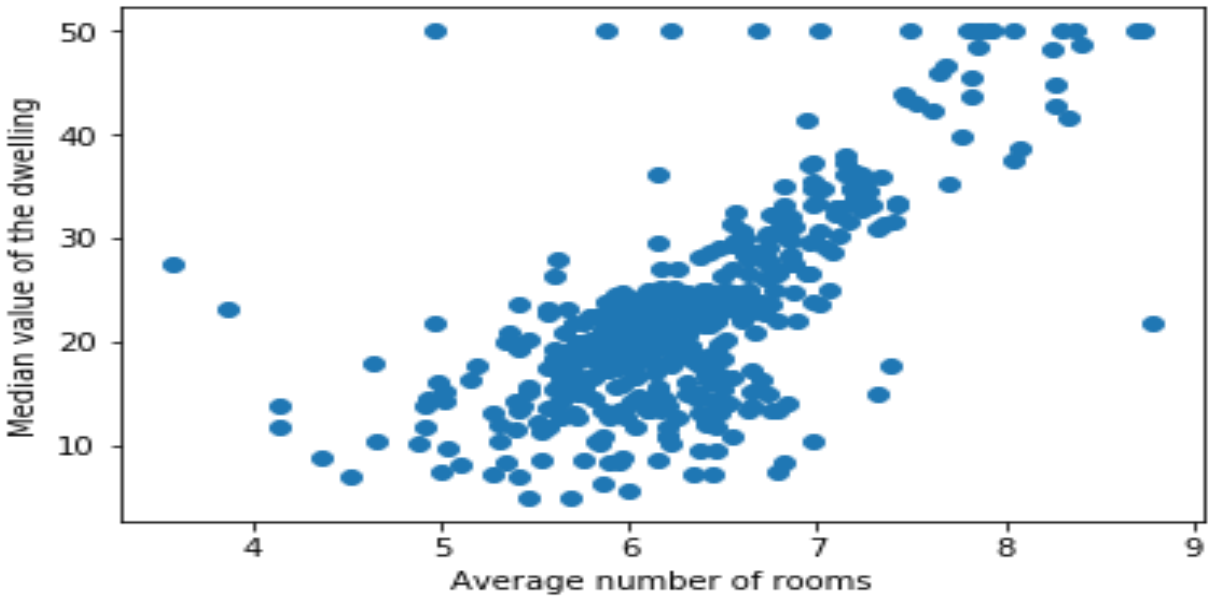
Although *cost function* and *loss function* are synonymous and used interchangeably, they are different.
A loss function is for a single training example. It is also sometimes called an error function.
A cost function, on the other hand, is the average loss over the entire training dataset.

The optimization strategies aim at minimizing the cost function.

Gradient descent approach applied to linear regression with weight (coefficient update strategy)

The steps that will be followed for each loss function below:

1. Write the expression for our predictor function, f(X) and identify the parameters that we need to find
2. Identify the loss to use for each training example
3. Find the expression for the Cost Function – the average loss on all examples
4. Find the gradient of the Cost Function with respect to each unknown parameter
5. Decide on the learning rate and run the weight update rule for a fixed number of iterations



Lets say we use the famous Boston Housing Dataset for understanding loss functions.

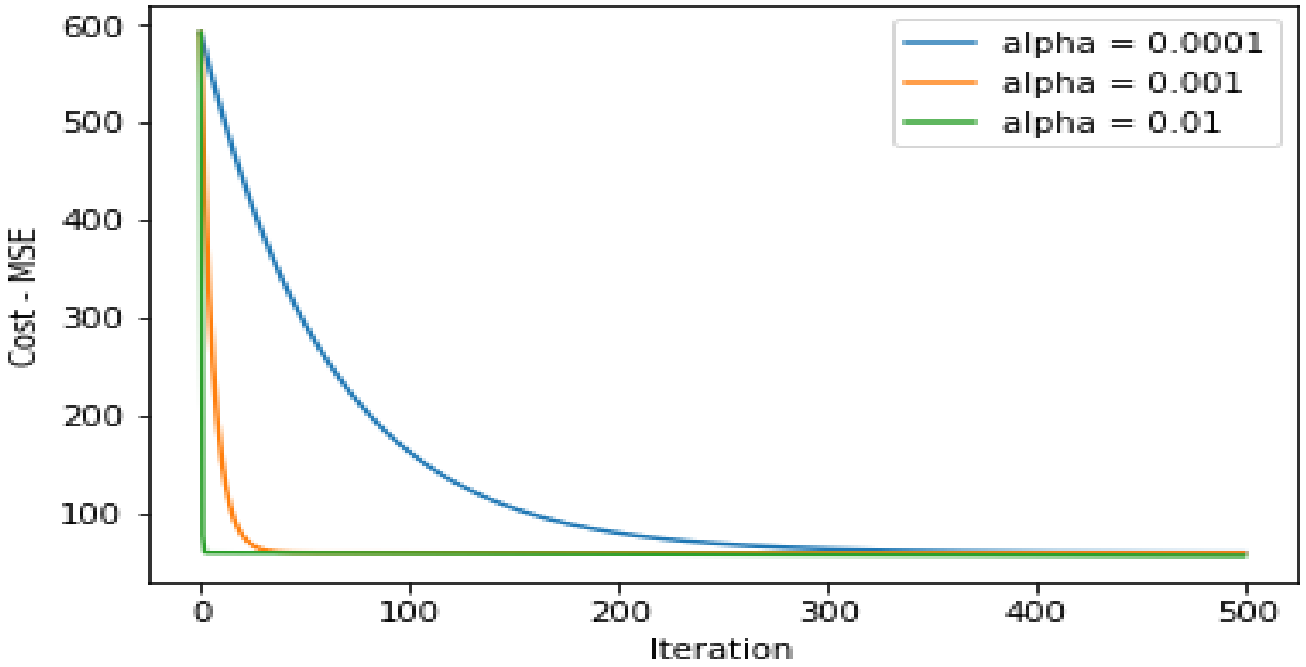
And to keep things simple, we will use only one feature – the *Average number of rooms per house* (X) – to predict the dependent variable – *Median Value* (Y) of houses in $1000' s.

1. Squared Error Loss

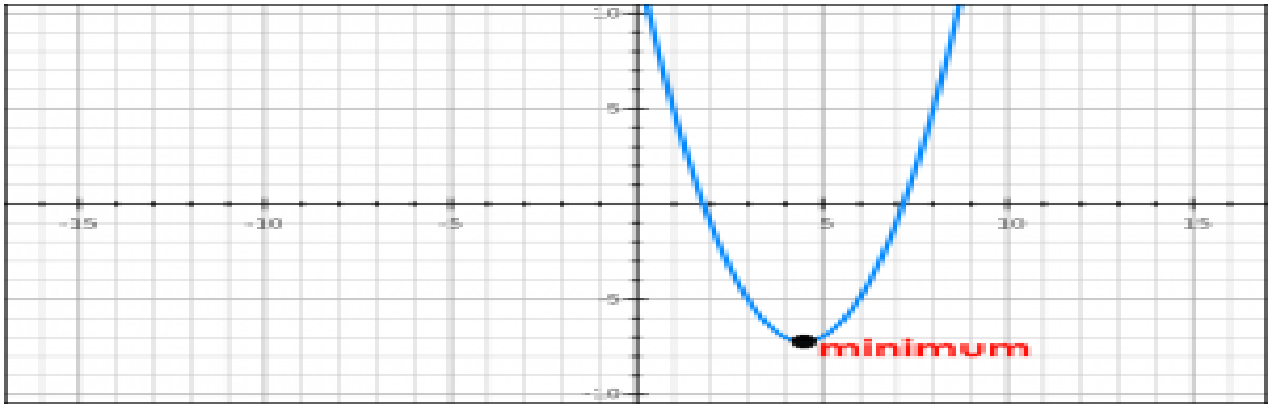
$$L = \left(y - f(x)\right)^2$$

The corresponding cost function is the **Mean** of these **Squared Errors (MSE)**.

Applied on Boston dataset for different values of the learning rate for 500 iterations each



a bit more about the MSE loss function. It is a positive quadratic function (of the form ax^2 + bx + c where a > 0). how it looks graphically?



A quadratic function only has a global minimum. Since there are no local minima, we will never get stuck in one.
Hence, it is always guaranteed that Gradient Descent will converge (*if it converges at all*) to the global minimum.

The MSE loss function penalizes the model for making large errors by squaring them. Squaring a large quantity makes it even larger, right? But there's a caveat.
This property makes the MSE cost function less robust to outliers.

Therefore, ***it should not be used if our data is prone to many outliers.***

2. Absolute Error Loss

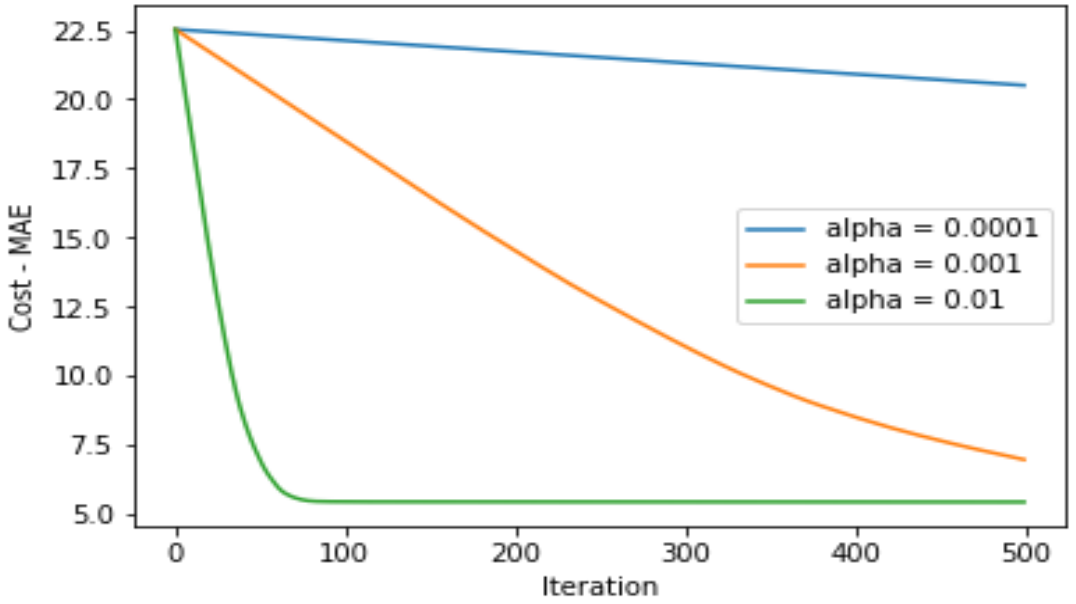
Absolute Error for each training example is the distance between the predicted and the actual values, irrespective of the sign. Absolute Error is also known as the **L1 loss**

$$L = |y - f(x)|$$

the cost is the **Mean** of these **Absolute Errors (MAE)**.

***The MAE cost is more robust to outliers as compared to MSE*.**
However, handling the absolute or modulus operator in mathematical equations is not easy.
We can consider this as a disadvantage of MAE.

plot after running the code for 500 iterations with different learning rates:



3. Huber Loss

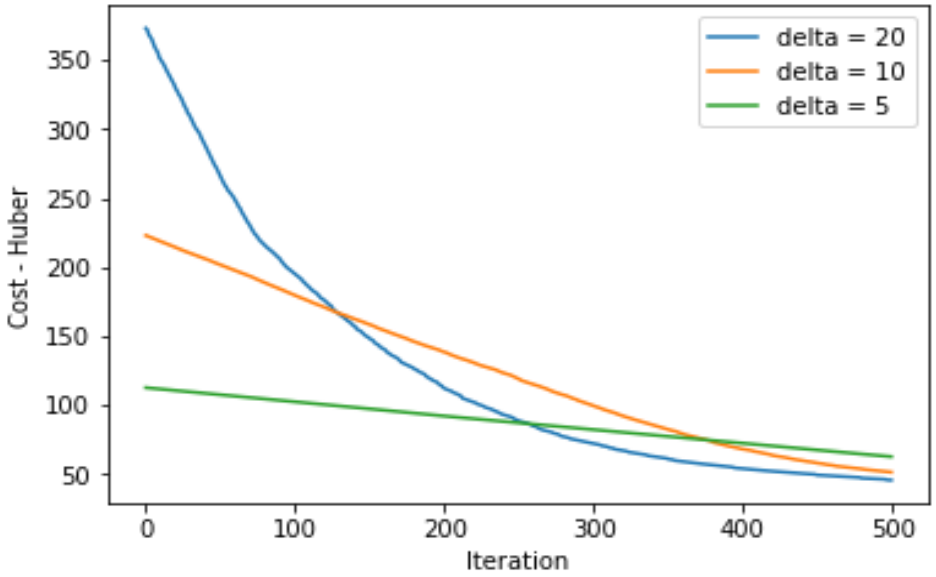
The Huber loss combines the best properties of MSE and MAE.
It is quadratic for smaller errors and is linear otherwise (and similarly for its gradient).
It is identified by its *delta* parameter:

$$L_\delta = \begin{cases} \frac{1}{2}\left(y - f(x)\right)^2, & if\ |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & otherwise \end{cases}$$

Quadratic

Linear

plot for 500 iterations of weight update at a learning rate of 0.0001 for different values of *the delta* parameter:



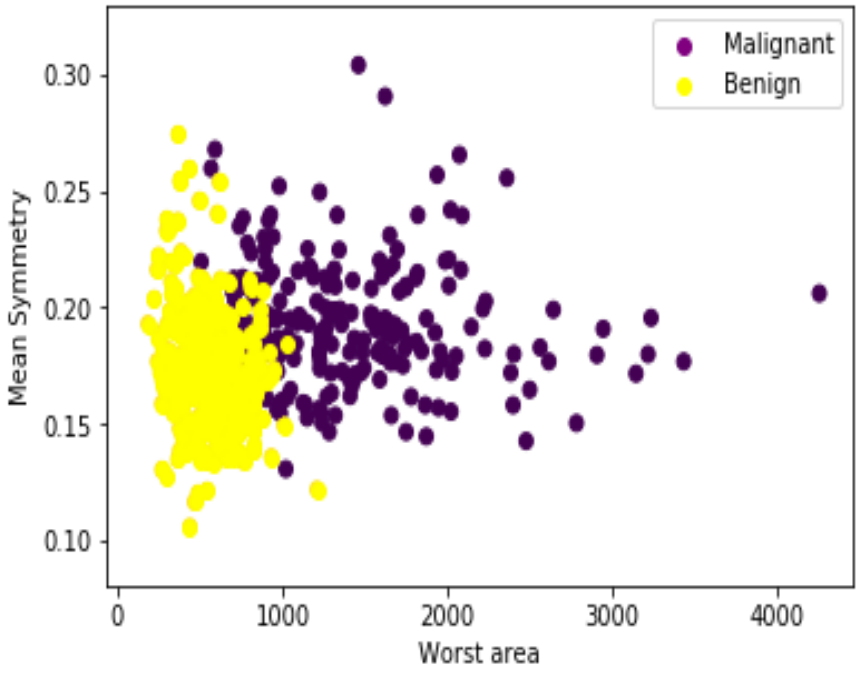
Huber loss is more robust to outliers than MSE. ***It is used in Robust Regression, M-estimation and Additive Modelling. A variant of Huber Loss is also used in classification.***

Binary Classification Loss Functions

Binary Classification refers to assigning an object into one of two classes.
This classification is based on a rule applied to the input feature vector.
For example, classifying an email as *spam* or *not spam* based on, say its subject line

##Cancer dataset

*To classify a tumor as **'Malignant'** or **'Benign'** based on features like average radius, area, perimeter, etc.*
*For simplification, we will use only two input features (X_1 and X_2) namely **'worst area'** and **'mean symmetry'** for classification.*
The target value Y can be 0 (Malignant) or 1 (Benign).



1. Binary Cross Entropy Loss or log loss

entropy to indicate disorder or uncertainty.
It is measured for a random variable X with probability distribution p(X):

$$S = \begin{cases} -\int p(x).\log p(x).dx, & \text{if } x \text{ is continuous} \\ -\sum_{x} p(x).\log p(x), & \text{if } x \text{ is discrete} \end{cases}$$

A greater value of entropy for a probability distribution indicates a greater uncertainty in the distribution. Likewise, a smaller value indicates a more certain distribution.

This makes binary cross-entropy suitable as a loss function – **you want to minimize its value.**
We use **binary cross-entropy** loss for classification models which output a probability *p*.

Probability that the element belongs to class 1 (or positive class) = p
Then, the probability that the element belongs to class 0 (or negative class) = 1 - p

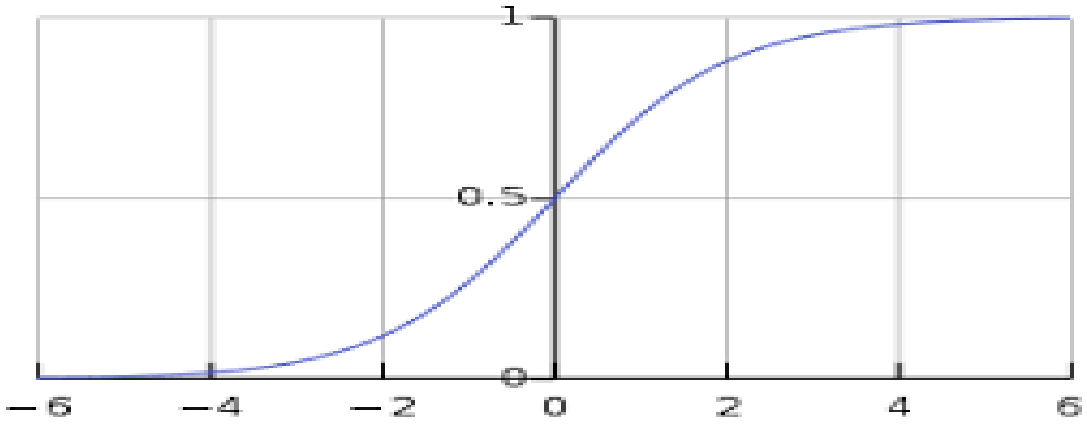
Then, the cross-entropy loss for output label y (can take values 0 and 1) and predicted probability p is defined as:

$$L = -y * \log(p) - (1-y) * \log(1-p) = \begin{cases} -\log(1-p), & if\ y = 0 \\ -\log(p), & if\ y = 1 \end{cases}$$

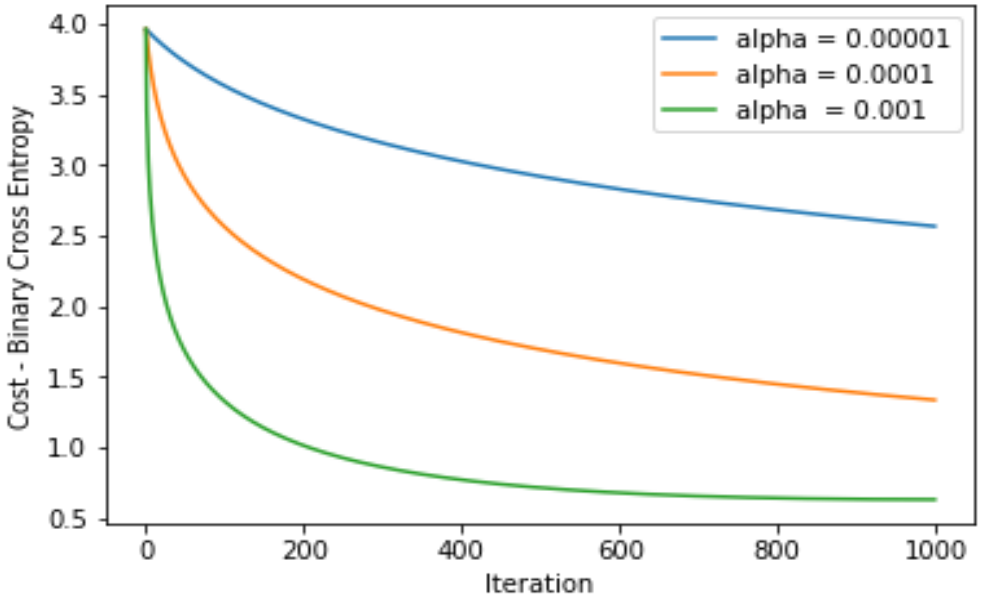
This is also called Log-Loss. To calculate the probability p, we can use the sigmoid function. Here, z is a function of our input features:

$$S(z) = \frac{1}{1 + e^{-z}}$$

The range of the sigmoid function is [0, 1] which makes it suitable for calculating probability.



plot on using the weight update rule for 1000 iterations with different values of alpha:



2. Hinge Loss

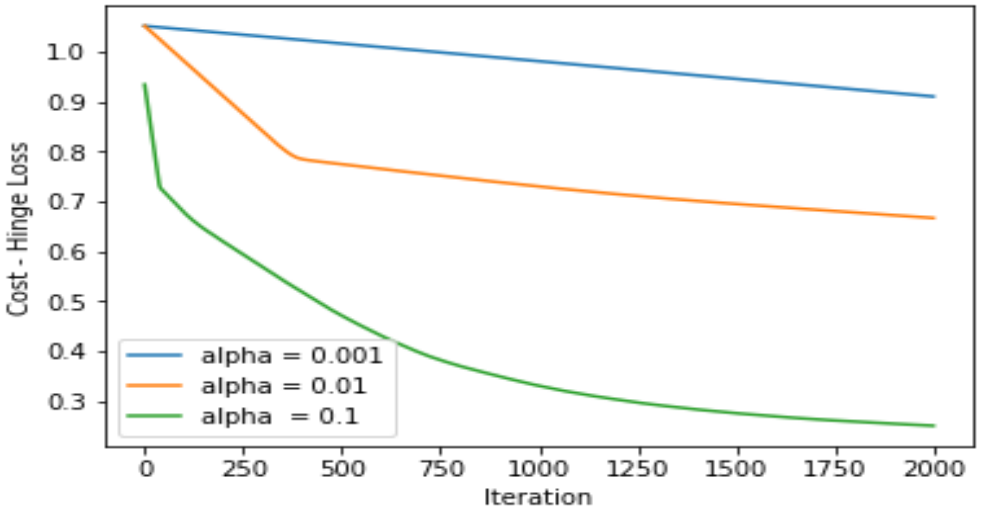
Hinge loss is mainly used with Support Vector Machine classifiers with class labels -1 and 1.

Hinge Loss not only penalizes wrong predictions but also right predictions that are not confident.

Hinge loss for an input-output pair (x, y) is :

$$L = \max\left(0, 1 - y * f(x)\right)$$

After running the update function for 2000 iterations with three different values of alpha, we obtain this plot

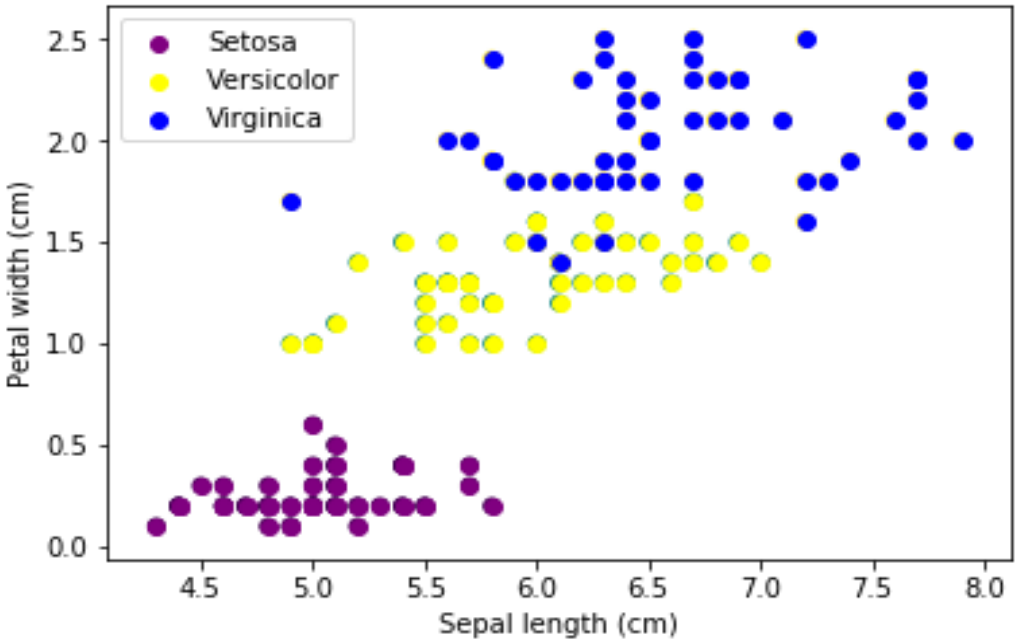


used when we want to make real-time decisions with not a laser-sharp focus on accuracy

Multi-Class Classification Loss Functions

use the Iris Dataset for understanding

use 2 features X_1, **Sepal length** and feature X_2, **Petal width,** to predict the class (Y) of the Iris flower – **Setosa, Versicolor or Virginica**



Multi-Class Cross Entropy Loss

The multi-class cross-entropy loss is a generalization of the Binary Cross Entropy loss. The loss for input vector Xi and the corresponding one-hot encoded target vector Yi :

$$L(X_i, Y_i) = -\sum_{j=1}^{c} y_{ij} * \log(p_{ij})$$

where Y_i is one − hot encoded target vector $(y_{i1}, y_{i2}, \ldots, y_{ic})$,

$$y_{ij} = \begin{cases} 1, & \text{if } i_{th} \text{ element is in class } j \\ 0, & \text{otherwise} \end{cases}$$

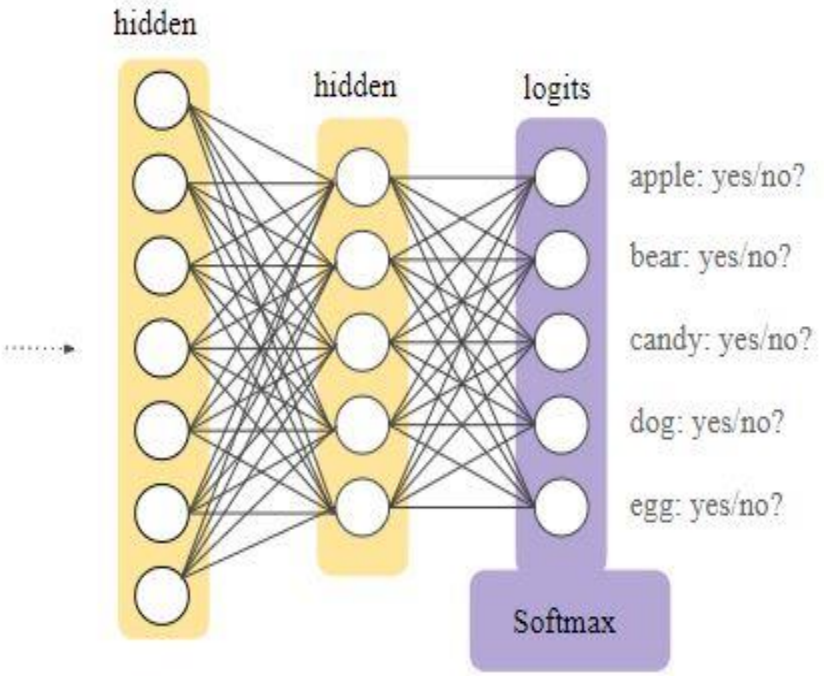
$p_{ij} = f(X_i) = \text{Probability that } i_{th} \text{ element is in class } j$

use the softmax function to find the probabilities p_ij:

Softmax is implemented through a neural network layer just before the output layer.

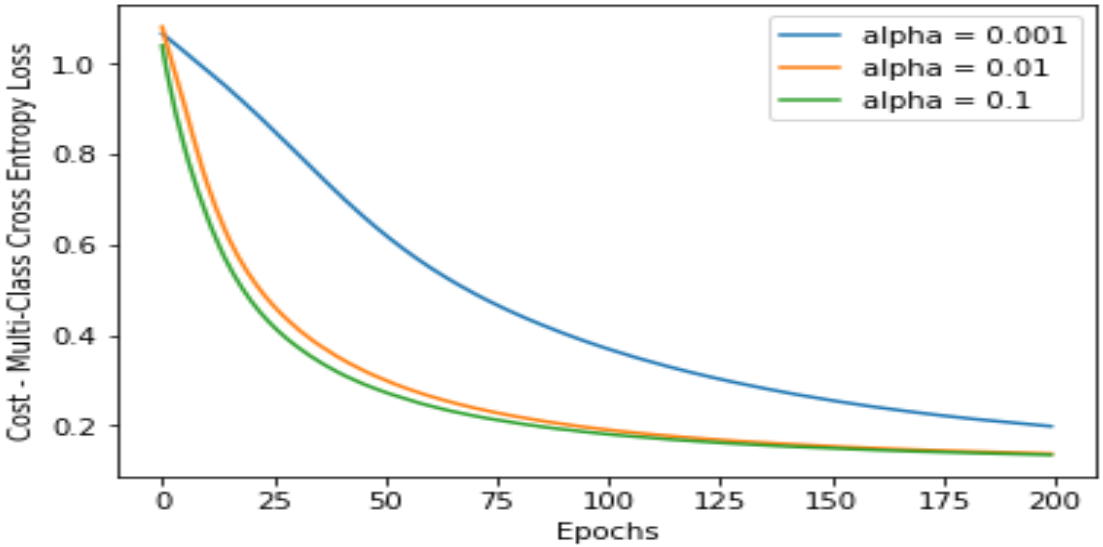
The Softmax layer must have the same number of nodes as the output layer."

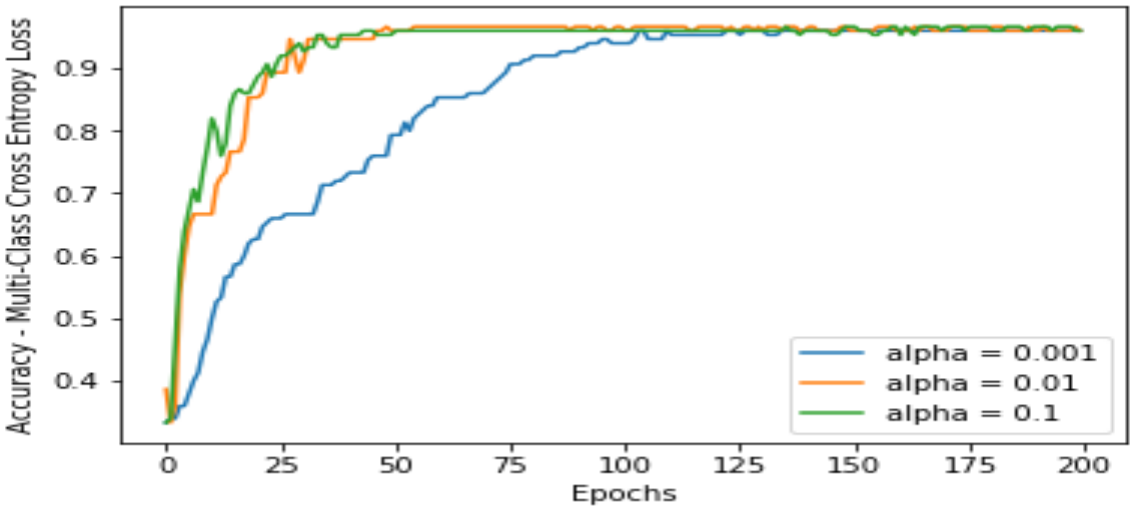
output is the class with the maximum probability for the given input.



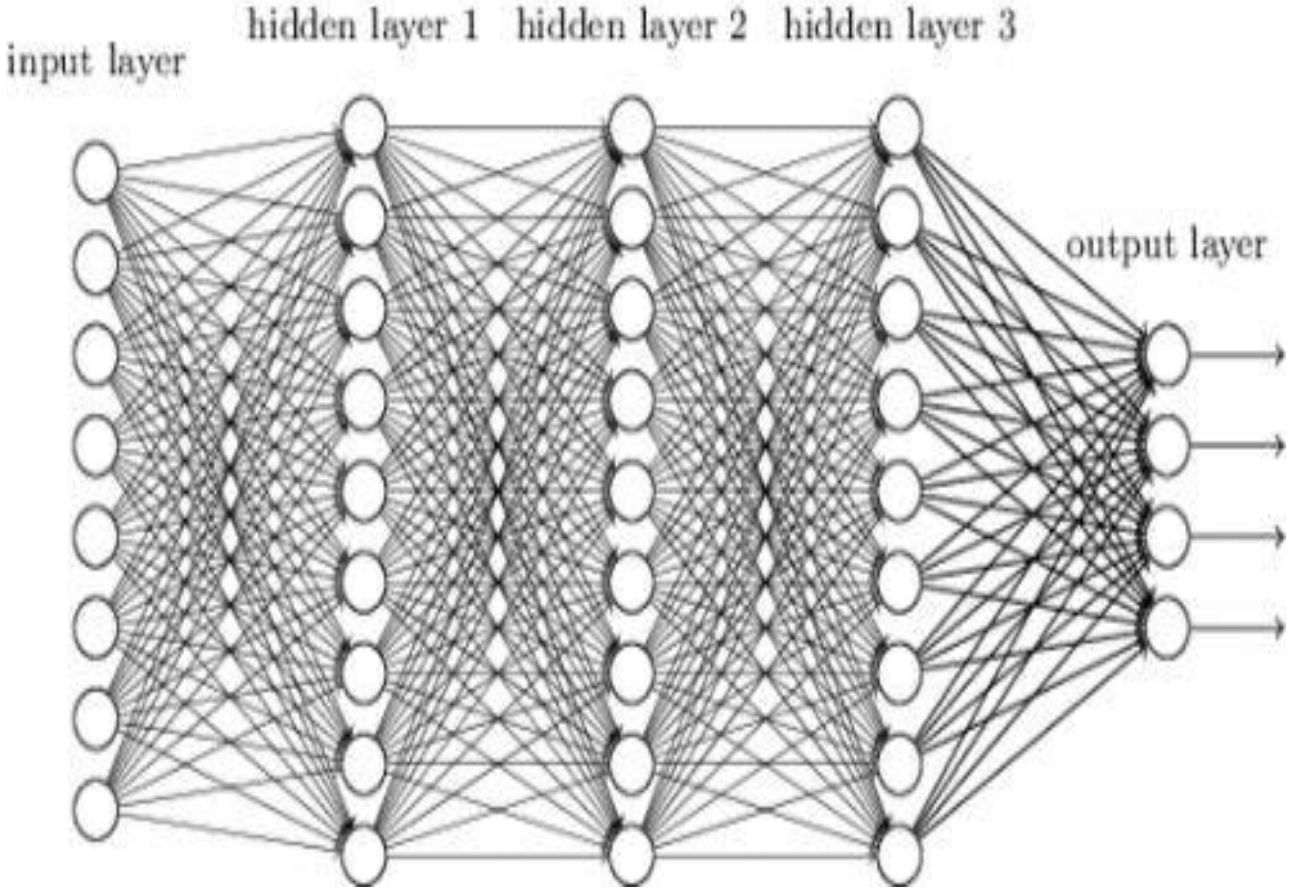
a model using an input layer and an output layer and compile it with different learning rates.

the plots for cost and accuracy respectively after training for 200 epochs





Deep neural network



Neural Network Learning as Optimization

- A deep learning neural network learns to map a set of inputs to a set of outputs from training data.
- cannot calculate the perfect weights for a neural network; there are too many unknowns. Instead, the problem of learning is cast as a search or optimization problem and an algorithm is used to navigate the space of possible sets of weights the model may use in order to make good or good enough predictions.
- a neural network model is trained using the stochastic gradient descent optimization algorithm and weights are updated using the backpropagation of error algorithm

- "*gradient*" in gradient descent refers to an error gradient. The model with a given set of weights is used to make predictions and the error for those predictions is calculated
- The gradient descent algorithm seeks to change the weights so that the next evaluation reduces the error, meaning the optimization algorithm is navigating down the gradient (or slope) of error.

- In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function.
- We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively.
- Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "*loss*."

- The cost or loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model.
- In calculating the error of the model during the optimization process, a loss function must be chosen.
- This can be a challenging problem as the function must capture the properties of the problem and be motivated by concerns that are important to the project and stakeholders.

Model Selection

Model selection is a technique for selecting the best model after the individual models are evaluated based on the required criteria.

Resampling methods, are simple techniques of rearranging data samples to inspect if the model performs well on data samples that it has not been trained on.

Random Splits are used to randomly sample a percentage of data into training, testing, and preferably validation sets

time-wise split :The training set can have data for the last three years and 10 months of the present year. The last two months can be reserved for the testing or validation set.

K-Fold Cross-Validation: The cross-validation technique works by **randomly shuffling the dataset and then splitting it into k groups.** Then on iterating over each group, the group needs to be considered as a test set while all other groups are clubbed together into the training set.

BootStrap: The first step is to select a sample size (which is usually equal to the size of the original dataset). Thereafter, a sample data point must be randomly selected from the original dataset and added to the bootstrap sample. After the addition, the sample needs to be put back into the original sample. This process needs to be repeated for N times, where N is the sample size.

Model Evaluation

For every classification model prediction, a matrix called the ***confusion matrix*** can be constructed which demonstrates number of test cases correctly and incorrectly classified.

	Actual 0	**Actual 1**
Predicted 0	True Negatives (TN)	False Negatives (FN)
Predicted 1	False Positives (FP)	True Positives (TP)

Accuracy is the simplest metric and can be defined as the number of test cases correctly classified divided by the total number of test cases.

$$Accuracy = (TP+TN)/(TP+TN+FP+FN)$$

Precision is the metric used to identify the correctness of classification.

$$Precision = TP/(TP + FP)$$

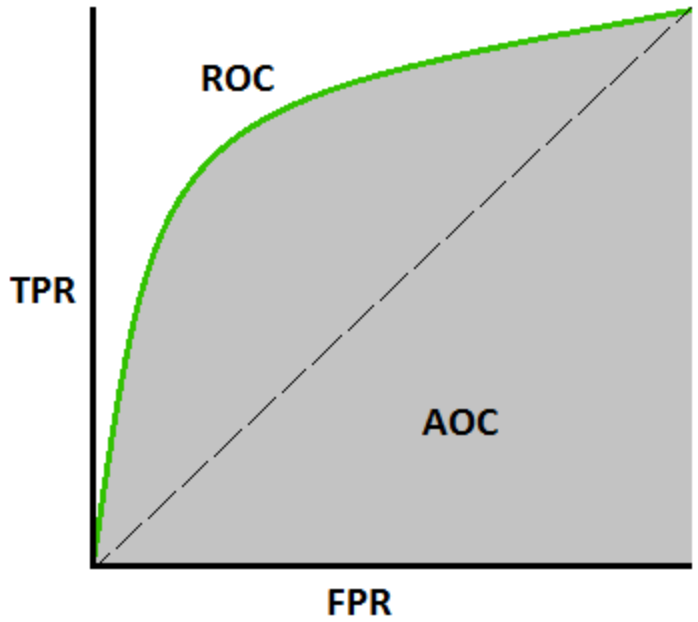
Recall tells us the number of positive cases correctly identified out of the total number of positive cases.

$$Recall = TP/(TP + FN)$$

F1 score is the harmonic mean of Recall and Precision and therefore, balances out the strengths of each.

$$F1Score = 2 * ((precision * recall)/(precision + recall))$$

AUC-ROC: ROC curve is a plot of **true positive rate** (recall) against **false positive rate** (TN / (TN+FP)). AUC-ROC stands for Area Under the Receiver Operating Characteristics and the higher the area, the better is the model performance.



AUC - ROC Curve [Image 2] (Image courtesy: My Photoshopped Collection)

If the curve is somewhere near the 50% diagonal line, it suggests that the model randomly predicts the output variable.

Bias , Variance

Bias occurs when a model is strictly ruled by assumptions – like the linear regression model assumes that the relationship of the output variable with the independent variables is a straight line. This leads to ***underfitting*** when the actual values are non-linearly related to the independent variables.

Variance is high when a model focuses on the training set too much and learns the variations very closely, compromising on generalization. This leads to ***overfitting***.

An optimal model is one that has the **lowest bias and variance** and since these two attributes are indirectly proportional, the only way to achieve this is through a tradeoff between the two.
Therefore, the model selection should be such that the bias and variance intersect like in the image