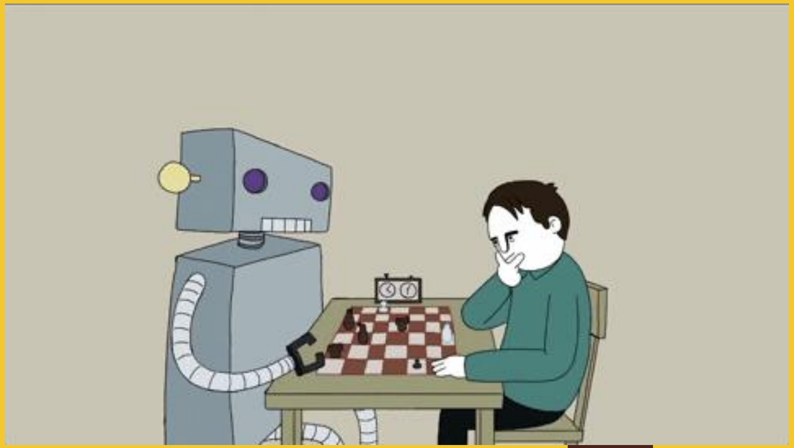


# CSE4006 DEEP LEARNING

**Dr K G Suma**
Associate Professor
School of Computer Science and Engineering

# Module No. 3
# Convolution Neural Networks
# 7 Hours

- Convolutional networks optimization
- Loss functions in classifiers
- Convolution layers
- Max pool layers
- VGG
- Google Net
- ResNet
- Dropout
- Normalization
- Rules update
- Data augmentation
- Transfer learning
- Analysis of pre trained models

- Convolutional networks optimization – Empirical risk optimization covered in Module 2

- Loss functions in classifiers –
  - *0-1 loss function*
  - *Surrogate loss function and need of surrogate loss function covered in Module 2*

- Dropout – Covered in regularization Module 2

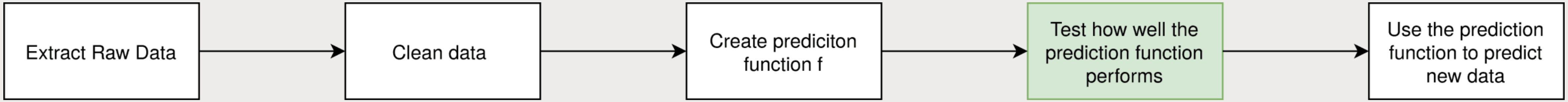
- Rules update – weight updation in Backpropagation, Gradient Descent covered in Module 2

# Loss function in classifiers – 0-1 loss function

- It is an important metric for the quality of Binary and Multiclass classification algorithms.
- Generally, the loss function plays a key role in deciding if a Machine learning algorithm is actually useful or not for a given data set.

# Loss functions  - Measuring Model Quality

- In the machine learning process, we have various steps including cleaning the data and creating a prediction function:




- To measure this quality of a data model, we can use three different metrics: **Loss, accuracy, and precision**. There are many more, like F1 score, recall, and AUC.

# Loss functions - Measuring Model Quality

- **The loss function takes the actual values and compares them with the predicted ones.** There are several ways to compare them. A loss of 0 significates perfect prediction. The interpretation of the amount of loss depends on the given dataset and model. A popular loss function used in machine learning is the squared loss:

$$\mathcal{L}_{sq} = \frac{1}{n}\sum_{i=1}^{n}(\tilde{y_i} - y_i)^2$$

In this formula, $\widehat{yi}$ is the correct result and $yi$ the predicted outcome. Squaring the differences actually gives us only positive results and magnifies large errors. As we can see, we also average our sum, dividing by n , to compensate for the size of our dataset.

# Loss functions - Measuring Model Quality

- **Another statistical metric is the accuracy that directly measures how many of our predictions are right or wrong.**
- In this case, it does not matter the size of the error, only if we have predicted false or correct. Accuracy can be calculated with the formula:

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{all classifications}}$$

# Loss functions - Measuring Model Quality

- **With our last metric, the precision, we calculate how close the predictions are.** Not to the original values, but to each other. The formula for precision is given by:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$$

- TP means True Positives and FP False Positives.

- True Positive - means a test correctly identifies a positive condition, i.e., A medical test shows a patient has a disease, and they actually do have the disease.

- False Positive - means a test indicates a positive condition when it is actually negative, The test incorrectly identifies a negative case as positive, i.e., A security system alerts about a potential intrusion, but there is no actual intruder present

- True Negative - means a test correctly identifies something as not having a certain condition, Test result matches the actual condition (negative result for a non-existent condition), i.e., A person takes a medical test and it comes back negative, and they are actually not have the disease.

- False Negative - means a test incorrectly indicates that something does not have a condition when it actually does, Test result does not match the actual condition (negative result when a condition is actually present). i.e., A security system not alerts about a potential intrusion, but there is actual intruder present, that is the test not indicating

# Example

- Let's have a look at an example to get a better understanding of this process. **Suppose we are working with a dataset of three pictures and we want to detect whether our pictures show dogs or not.**
- Our machine learning model gives us a probability of a dog being displayed for each picture. In this example, it gives us 80% for the first picture, 70% for the second, and 20% for the third picture. Our threshold for recognizing the picture as a dog picture is 70% and all of the pictures are actually dogs.

We thus have a loss of $\mathcal{L}_{sq} = \frac{1}{3}((1-0.8)^2 + (1-0.7)^2 + (1-0.2)^2) \approx 0.256$, an accuracy of $\frac{2}{3}$ and a precision of $\frac{2}{3}$, since we have 2 true positives and 1 false positive.

# The 0-1 Loss Function

- The 0-1 Loss function is actually synonymous with the accuracy as follows:

$$\mathcal{L}_{01}(\tilde{y}, y) = \frac{1}{n}\sum_{i=1}^{n} \delta_{\tilde{y}_i \neq y_i} \text{ with } \delta_{\tilde{y}_i = y_i} = \begin{cases} 0, & \text{if } \delta_{\tilde{y}_i \neq y_i} \\ 1, & \text{otherwise} \end{cases}$$

# The 0-1 Loss Function

- This allows a different weighing of our results. **For example, when working with disease recognition, we might want to have as few false negatives as possible, thus we could weigh them differently with the help of a loss matrix:**

$$A = \begin{bmatrix} 0 & 5 \\ 0.5 & 0 \end{bmatrix}$$

This loss matrix amplifies false negatives and weighs just half the loss of true positives.

The general loss matrix has the form:

$$A = \begin{bmatrix} TN & FN \\ FP & TP \end{bmatrix}$$

The problem with our 0-1 Loss function stays. It's not differentiable. Therefore it is not possible to apply methods such as gradient descent. Fortunately, we can still use a great palette of other classification algorithms, such as K-Means or Naive Bayes.

# The 0-1 Loss Function

- The simplest loss function is the zero-one loss. It literally counts how many mistakes an hypothesis function h makes on the training set. For every single example it suffers a loss of 1 if it is mispredicted, and 0 otherwise. The normalized zero-one loss returns the fraction of misclassified training samples, also often referred to as the training error. The zero-one loss is often used to evaluate classifiers in multi-class/binary classification settings but rarely useful to guide optimization procedures because the function is non-differentiable and non-continuous. Formally, the zero-one loss can be stated has:

$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n}\delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{o.w.} \end{cases}$$

- This loss function returns the error rate on this data set $D$. For every example that the classifier misclassifies (i.e. gets wrong) a loss of 1 is suffered, whereas correctly classified samples lead to 0 loss.

# 0-1 Loss Function - Advantages

**Intuitive interpretation:**

- It directly measures the classification error by assigning a loss of 1 for incorrect predictions and 0 for correct ones, making it easy to understand the performance of a model in terms of accuracy.

**Simplicity:**

- Due to its binary nature, the 0-1 loss is straightforward to calculate and interpret.

**Accurate evaluation metric:**

- When used to evaluate a trained model on a test set, it provides a clear measure of the overall classification accuracy.

# 0-1 Loss Function - Disadvantages

**Non-differentiable:**

- The biggest drawback is that the 0-1 loss function has sharp jumps at the decision boundary, resulting in a non-differentiable function, which means gradient-based optimization algorithms cannot be directly applied to minimize it.

**Not suitable for training:**

- Due to the non-differentiability, using the 0-1 loss directly for training a model would not allow for efficient parameter updates through gradient descent.

**Sensitivity to outliers:**

- A single misclassification can significantly impact the overall loss, making it potentially sensitive to outliers in the data.