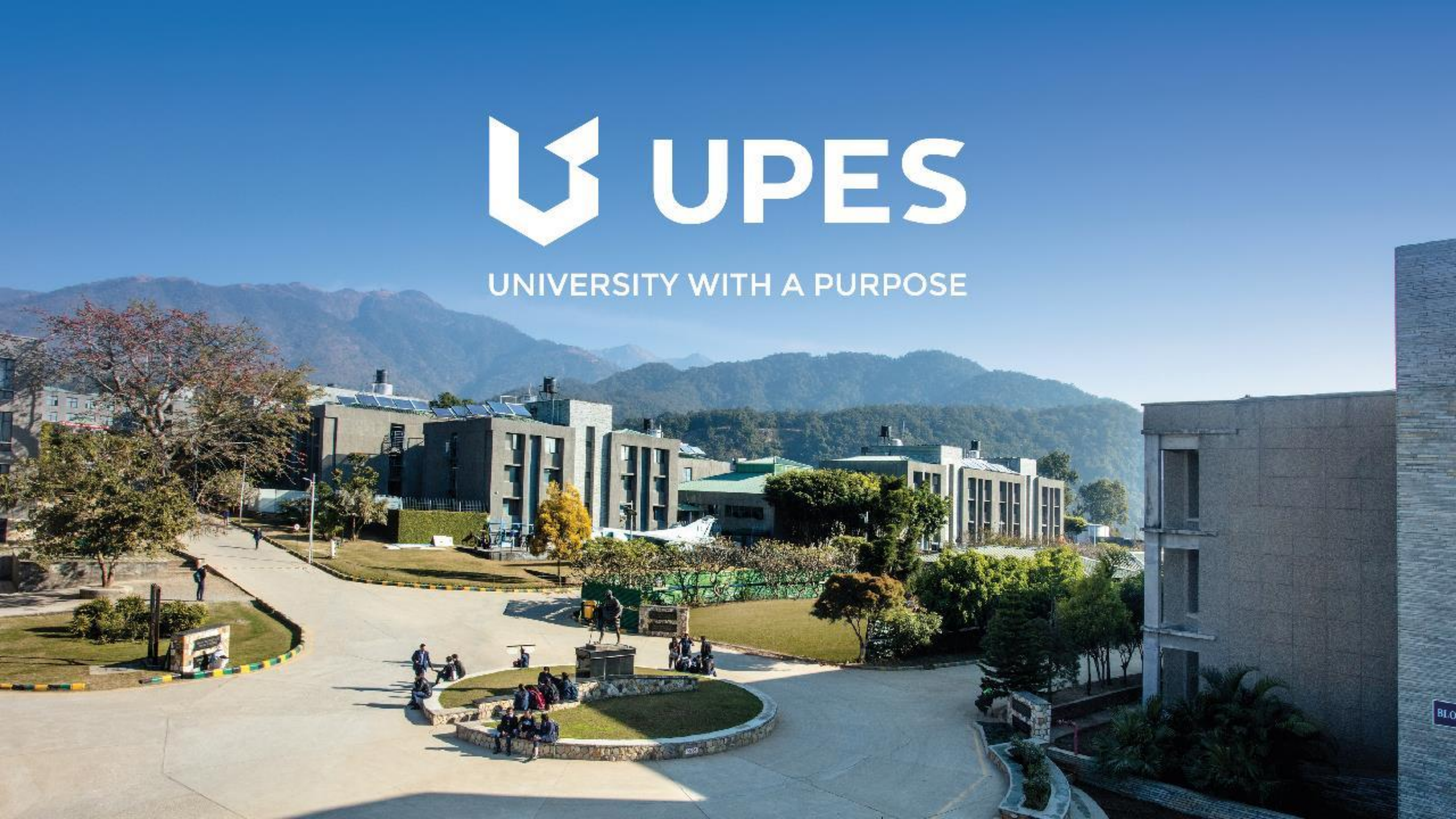
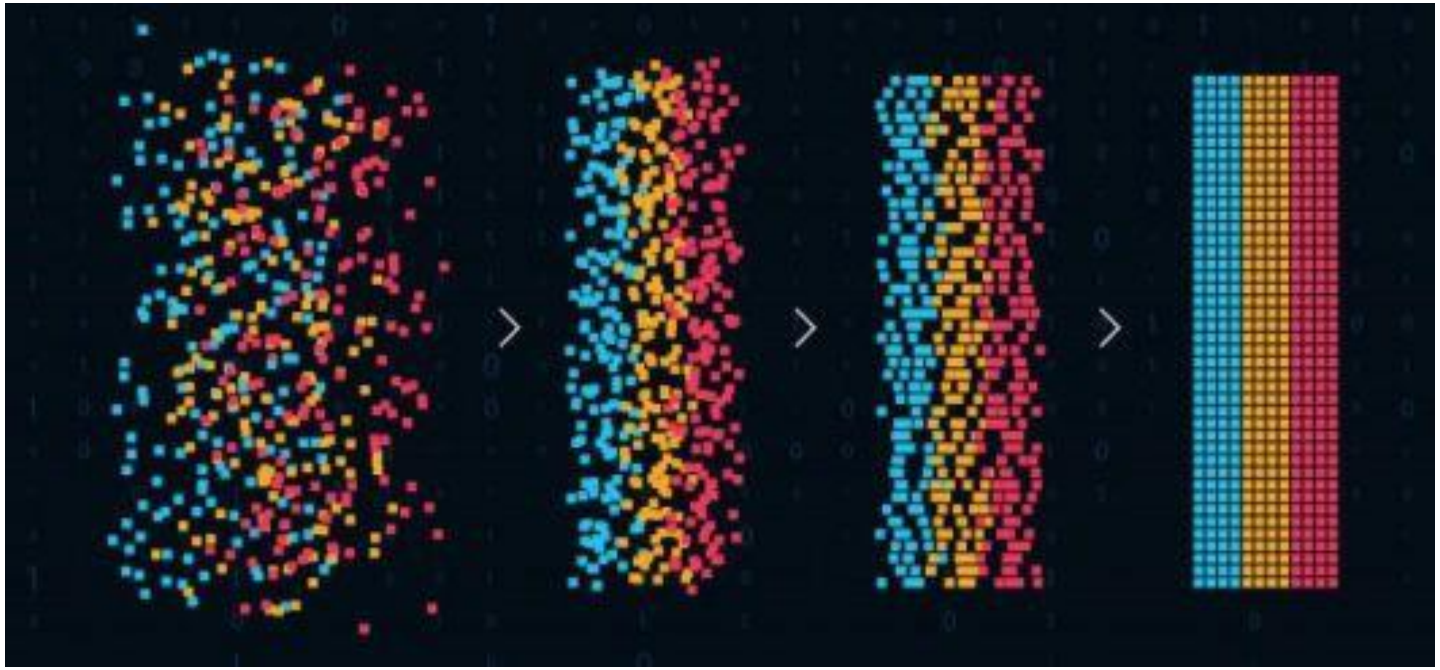

UPES
UNIVERSITY WITH A PURPOSE

Pattern and Anomaly Detection



Source: Edureka

B. Tech., CSE + AI/ML

Dr Gopal Singh Phartiyal

16/11/2021

Neural Networks: Network Training: Finding Weights

- Use of gradient information

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

- **Gradient descent** algorithm : change w in a way that a small step is taken in the direction of negative gradient.
  - At each step, the weight vector is moved in the direction of the greatest rate of decrease of the error function
  - After each step, the gradient is re-evaluated and it goes on again and again until termination criteria s met.
  - In batch type gradient descent methods: weights are updated only after the model has seen all training samples once and only once.
  - Conjugate gradients and quasi-Newton are variants of batch gradient descent.
  - Issue: Local minima, all points are required, computationaly expensive

Neural Networks: Network Training: Finding Weights

- **Stochastic GD:**
- **Motivation:-**Updating weight on all training samples in one go is equivalent to updating weights after training with one sample at a time.

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}).$$

- Advantages: Easier to implement
- Computationally cheap
- Randomization leads to generalization

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)}).$$

Neural Networks: Network Training: Finding Weights

- **Error Backpropagation**
- Method of evaluating the gradient of error function in a feedforward neural network
- Local message passing scheme (forward and backword propagation of information)
- To understand
  - Forward pass (output based on updated weights)
  - Backward pass (backpropagation of error)
  - Weight update
- In general, backpropagation can be used elsewhere.

Neural Networks: Network Training: Errors

- Error Functions in NN based models
- 1. Regression: When the output layer of the NN has linear or identity activation function

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

- 2. Classification
  - Binary: Two class

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

  - Multiple two-class

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \{t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln(1 - y_{nk})\}$$

  - Multi-class

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w}).$$

Neural Networks: Network Training: Finding Weights

- **Error Backpropagation:**
- For one sample

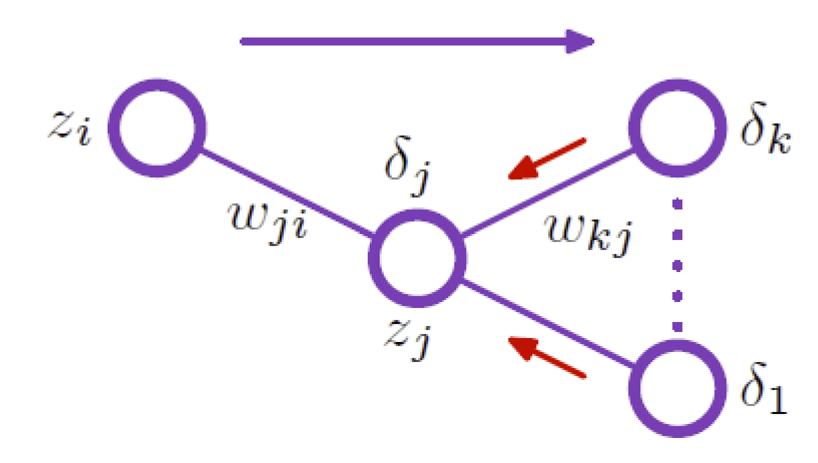
$$E_n = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2$$

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj})x_{ni}$$

$$a_j = \sum_i w_{ji} z_i$$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}.$$

$$\frac{\partial a_j}{\partial w_{ji}} = z_i.$$



$$\boxed{\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i.}$$

Neural Networks: Network Training: Finding Weights

- **Error Backpropagation**
- .

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

- For batch methods

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}.$$

Next time: Neural Networks

Thank You

UPES
UNIVERSITY WITH A PURPOSE