Neural Network Lab Problems: Detailed Summary Tables and Descriptions

---

1. AND Function with Bipolar Inputs using Perceptron

Aspect	**Description**
Problem	Evaluate AND logic function using bipolar inputs and targets with a perceptron neural network.
Input	Bipolar inputs $x_i \in \{+1, -1\}$ representing logic levels, and bipolar targets $d \in \{+1, -1\}$.
Model	Single-layer perceptron with weighted sum and bias.
Activation Function	Bipolar step function (sign function).
Training Method	Delta learning rule with stochastic gradient descent updating weights per sample.
Learning Rate	Fixed scalar controlling weight update magnitude.
Weight Initialization	Random small values for weights and bias.
Training Process	Iterative weight update until convergence or max epochs reached.
Output	Predicted output matching AND logic behavior.
Visualization	Convergence curves (error vs epochs) and decision boundary lines plotted in input space.

Detailed Description:

The perceptron computes the net input as

$$net = \sum_i w_i x_i + b$$

where w_i are weights and b is bias. The output y is given by

$$y = \text{sign}(net) = \begin{cases} +1 & \text{if } net \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

The error is

$$e = d - y$$

and the weights are updated by the delta rule:

$$\Delta w_i = \eta \cdot e \cdot x_i$$

where η is the learning rate. The network is trained over multiple epochs until the error converges.

---

2. XOR Function Using McCulloch-Pitts Neuron

Aspect	**Description**
Problem	Generate XOR logic function using McCulloch-Pitts neuron model.
Input	Binary inputs $x_i \in \{0, 1\}$ and targets $d \in \{0, 1\}$.
Model	McCulloch-Pitts neuron with weighted sum and threshold activation.
Activation Function	Binary threshold function.
Training Method	Logic gate realization using fixed weights and threshold values.
Weight Initialization	Manually set to achieve XOR functionality (since XOR is not linearly separable).

Aspect	Description
Training Process	No learning; fixed weights chosen to simulate XOR using multiple neurons or layered approach.
Output	Correct XOR output for input pairs.
Visualization	Convergence and decision boundary plot demonstrating non-linear separation.

Detailed Description:

XOR is a non-linearly separable function, so it cannot be solved by a single McCulloch-Pitts neuron. Typically, two neurons are combined to realize XOR logic. The neuron calculates

$$net = \sum_i w_i x_i$$

and outputs 1 if $net \geq \theta$ (threshold), else 0. Appropriate thresholds and weights are assigned manually.

---

3. Implement SGD Using Delta Learning Rule

Aspect	Description
Problem	Implement Stochastic Gradient Descent (SGD) with Delta learning for specified input-target sets
Input	Input matrix X and target vector D.
Model	Single-layer perceptron.
Activation Function	Step or sign function.
Training Method	Weight updates per sample based on Delta rule with learning rate η.
Learning Rate	Fixed scalar controlling update magnitude.
Weight Initialization	Small random values.
Training Process	Iterate over each sample, calculate output, compute error, and update weights until convergence.
Output	Weight vector converged to minimize error on dataset.
Visualization	Convergence plot showing error decreasing over epochs.

Detailed Description:

Using inputs

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

weights w are updated per sample via

$$\Delta w = \eta(d - y)x$$

where y is the predicted output. The stochastic update helps faster convergence on small datasets.

---

4. Compare SGD and Batch Methods using Delta Learning Rule

Aspect	Description
Problem	Compare performance of Stochastic Gradient Descent (SGD) and Batch learning methods with Delta rule.

Aspect	Description
Input	Same input-target datasets for both methods.
Model	Single-layer perceptron network.
Training Method	SGD: updates weights per sample; Batch: updates weights after processing all samples.
Learning Rate	Same for both methods for fair comparison.
Weight Initialization	Identical for both.
Training Process	Run both algorithms for fixed epochs, measure convergence speed and error.
Output	Performance metrics such as training time, final error, and convergence curves.
Visualization	Plots comparing error reduction over epochs for both methods.

Detailed Description:

Batch learning computes total error gradient for the entire dataset before updating weights:

$$\Delta w = \eta \sum_{i=1}^{N} (d_i - y_i) x_i$$

while SGD updates weights incrementally after each sample. SGD often converges faster on noisy data but with more fluctuations.

---

5. Digit Recognition Using 5x5 Pixel Images

Aspect	Description
Problem	Recognize handwritten digits 1 to 5 from 5x5 pixel input images.
Input	5x5 binary pixel matrices representing digits.
Model	Single or multilayer neural network with fully connected layers.
Activation Function	Sigmoid or ReLU for hidden layers; Softmax for output layer.
Training Method	Backpropagation with gradient descent.
Dataset	Custom dataset of 5 digits represented as 5x5 pixel images.
Weight Initialization	Random initialization.
Training Process	Forward propagation of input through network, calculate error with cross-entropy, backpropagat to update weights.
Output	Classification of input into digit classes 1-5.
Visualization	Training loss and accuracy curves; confusion matrix for evaluation.

Detailed Description:

Each 5x5 pixel image is flattened into a 25-dimensional input vector x. The network learns to map these to one-hot encoded output classes. The loss function used is categorical cross-entropy:

$$L = -\sum_{i=1}^{C} d_i \log y_i$$

where $C = 5$ classes, d_i is true label and y_i predicted probability.

---

6. Image Classification using CNN

Aspect	Description
Problem	Classify images (e.g., face/fruit/bird) using Convolutional Neural Networks.
Input	Image datasets with labeled classes.
Model	CNN with convolutional, pooling, fully connected layers.
Activation Function	ReLU in hidden layers, Softmax in output layer.
Training Method	Backpropagation with Adam or SGD optimizer.
Dataset	Standard datasets like CIFAR, or custom image datasets.
Weight Initialization	Xavier or He initialization for deep networks.
Training Process	Multiple epochs of forward and backward passes to minimize classification error.
Output	Predicted class labels with probability scores.
Visualization	Accuracy and loss curves, confusion matrix, sample classification results.

Detailed Description:

CNNs extract hierarchical features from images by sliding filters (kernels) over the input to produce feature maps. Pooling layers reduce spatial dimensions. The output layer uses Softmax to produce class probabilities:

$$P(y = j|x) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

where z_j is the logit for class j.

---

7. ANN with Backpropagation for 3-Layer Network

Aspect	Description
Problem	Learn weights of a 3-layer artificial neural network using backpropagation.
Input	Input vectors and target outputs.
Model	3-layer ANN: input layer, one hidden layer, output layer.
Activation Function	Sigmoid or ReLU in hidden layers; sigmoid or softmax in output.
Training Method	Backpropagation algorithm with gradient descent.
Weight Initialization	Random small values.
Training Process	Forward pass, error calculation, backward pass to update weights.
Output	Trained network capable of generalizing input-output mapping.
Visualization	Error vs epoch curves, accuracy on training and validation sets.

Detailed Description:

The backpropagation algorithm computes gradients of error w.r.t weights using chain rule. Weight updates:

$$w^{new} = w^{old} - \eta \frac{\partial E}{\partial w}$$

where E is loss (e.g. mean squared error):

$$E = \frac{1}{2} \sum (d - y)^2$$

---

8. Speech Signal Number Recognition using ANN

Aspect	Description
Problem	Recognize spoken digits (1 to 4) using features from speech signals with an ANN.
Input	Speech signal feature vectors (e.g., MFCCs).
Model	Feedforward ANN with input, hidden, and output layers.
Activation Function	Sigmoid or ReLU in hidden layers; softmax in output.
Training Method	Backpropagation with gradient descent.
Dataset	Digit speech samples with extracted features.
Training Process	Train network to classify digit features into corresponding classes.
Output	Predicted digit label for given speech input.
Visualization	Training accuracy, confusion matrix.

Detailed Description:

Speech features (e.g., Mel Frequency Cepstral Coefficients) are extracted to represent audio. The ANN learns mapping from these features to digits, using softmax output for classification.

---

9. Training Neural Networks Using Perceptron Learning Rule

Aspect	Description
Problem	Train a perceptron using Perceptron Learning Rule to classify inputs into two classes.
Input	Binary or bipolar inputs with known targets.
Model	Single-layer perceptron.
Activation Function	Step function.
Training Method	Perceptron learning algorithm iteratively adjusting weights on misclassification.
Learning Rate	Fixed scalar for weight updates.
Training Process	Update weights if prediction is incorrect until all inputs classified correctly or max epochs reached.
Output	Final weight vector producing correct classification.
Visualization	Error reduction over epochs and decision boundary.

Detailed Description:

The perceptron learning rule updates weights only if the output is incorrect:

$$w^{new} = w^{old} + \eta(d - y)x$$

The algorithm guarantees convergence if data is linearly separable.

---

10. Implement Delta Learning Rule for Linearly Separable Problem

Aspect	Description
Problem	Implement Delta learning rule for training on a linearly separable dataset.
Input	Inputs and target outputs consistent with linear separability.
Model	Single-layer perceptron with linear or step activation.
Training Method	Delta rule with gradient descent to minimize squared error.
Learning Rate	Fixed scalar.
Training Process	Iterative updates until error minimized.
Output	Weights that correctly classify input data.
Visualization	Training error reduction over epochs.

Detailed Description:

The Delta rule minimizes mean squared error between network output and target:

$$\Delta w = \eta(d - y)x$$

where y can be a linear activation output.