Project 4 recitation

Wednesday, November 14, 2018 3:01 PM

Q1: Feed Forward

- inActs -> Input actions
- Return outputs (list of lists)
- Loop over all the layers
- Create a new list of inputs -> list
- Loop over all the nodes
- List.append(run the sigmoid activation on previous layer for this node)
- outputs: [[set of inputs], [set of inputs], …]
- Outputs.append(list)

Q1.5: Math

- Sigmoid
    - Returns the equation
- Sigmoid Activation
    - Given inActs
    - Insert 1.0 to the front of inActs
    - Get weighted sum (inActs)
    - Return sigmoid(weighted sum)

Q2: All the actual math

- Sigmoid ActivationDeriv
    - Given inActs
    - Insert 1.0 to the front of inActs
    - Get weighted sum (inActs)
    - Return sigmoidDeriv(weighted sum)
- SigmoidDeriv
    - N = exp(value)
    - D = pow(exp(value) + 1, 2)
    - n/d
- Updating Weights
    - Totalmodifaction = 0.0
    - List of weights -> w

```
    ▪ inActs insert 1.0 at the beginning
    ▪ Loop over self.weights
        □ newWeight = weight + (alpha * inAct[i] + delta)
        □ w.append(newWeight)
        □ Totalmod += abs(newWeight - weight)
    ▪ Self.weights = w
    ▪ Return totalmod
```

Q3: BackProp

```
- Example -> ([input], [expected output])
- lastLayerOutput = allLayerOutput[-1]
- Loop over example[1] -> elem
    □ Gprime = outputlayer[elem].sAD(allLayerOutput[-2]
    □ Error = example[1][elem] - lastLayeroutput[elem]
    □ Delta = error * gprime
    □ outDelta.append(delta)
- Deltas.append(outDelta)
- Loop over every layer
    □ Loop over each neuron -> neuron
    □ Gprime = layer[neuron].sAD(allLayerOutput[layerNum]
    □ Error = 0.0
    □ For each neuron n  in the next layer
        ◆ Deltatemp = nextLayer[n].weight * deltas[0][n]
        ◆ Error += deltatemp
    □ Delta  = gprime * error
    □ hiddenDelta.append(delta)
    □ Deltas = hiddendelta + deltas
- Loop over the numberOfLayers
    □ For each neuron
        ◆ Weightmod =  neuron.updateWeights
        ◆ AverageWeightChange += weightmod
- Return averageError, averageWeightChange
```

Q4: BuildNeuralNet

```
- Loop while weightMod > weightChangeThreshold and iteration
```

```
    < maxIterations
      □ backPropLearning(exampleTrain)
  - TestError
  - Test correct
  - For each example in Example Test
      □ feedForward(example)
      □ For each outputNode in outputLayer
          ◆ If not outputNode == example[1][n]
      □ If allCorrect : testCorrect++
  - testAccuracy = testCorrect/Total
  - Return nnet, testAccuracy
```