UNIVERSITY OF CALIFORNIA, BERKELEY

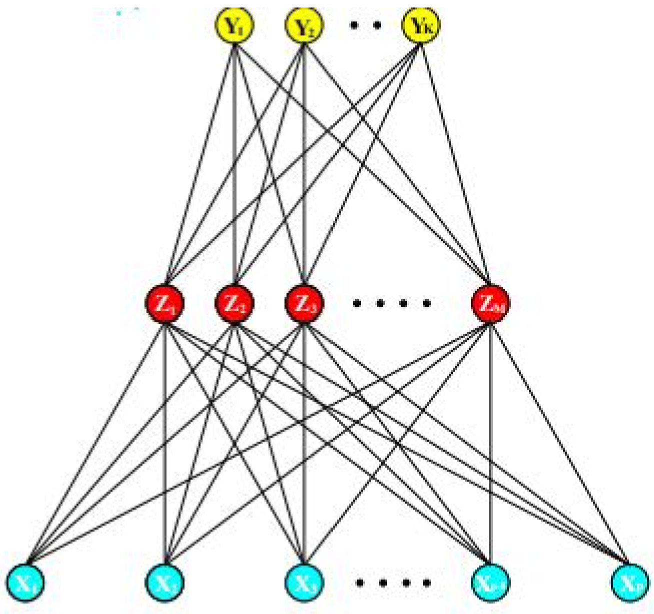
Lecture 17: CS-189

Oscar Ortega

July 16, 2021

1 ESL 11.3: NEURAL NETWORKS

A neural network is a two-stage regression or classification model, typically represented by a **network diagram** as in the figure below:



For K-class classification there would be K nodes on the top with the kth node modeling the probability of class K. There are K target measurements $Y_k : k = 1, \ldots, k$ each being coded as a 0-1 variable for the kth class. The derived features Z_i are created from linear combinations of the inputs, and then the Target values are modeled as functions of linear combinations of the Z variables. Can be modeled as follows:

$$Z_m = \sigma(\alpha_{0,m} + \alpha_m^T X) \tag{1.1}$$

$$T_k = B_{0,k} + B_k^T Z : k = 1, ..., k \tag{1.2}$$

$$f_k(X) = g_k(T) : k = 1, ..., k \tag{1.3}$$

Typically the activation function σ, will be the sigmoid function although, sometimes radial basis functions are used. Sometimes an additional bias term will be used. The softmax function is also commonly used.

$$\text{Softmax}(T) = \frac{e^{T_k}}{\sum_{i=1}^{K} e^{T_i}}$$

The unites in the middle of the network, computing the derived features, Z_m, are called **hidden units** because the values are not directly observed. There can also be more than one hidden layer.

2 Lecture

2.1 Neural Networks

Can be used for both classification and regression:
Perceptron research was halted in 1969, when Rosenblatt's perceptron algorithm was criticized for its inability to learn XOR, it is not a linearly separable problem.
However the XOR, problem can be solved if you add one new quadratic feature, x_1, x_2 XOR is linearly separable.
A linear combo of a linear combo is a linear combo ... only works for linearly separable points.

2.2 Network with 1 Hidden Layer

Input layer: $x_1, ..., x_d; x_{d+1} = 1$
Hidden Layer: $h_1, ...h_m; h_{m+1} = 1$
Output Layer: $z_1, ..., z_k$

Layer 1 weights: $m \times (d+1)$ matrix V, V_i^T is row i of V
Layer 2 weights: $k \times (m+1)$ matrix W, w_j^T is row j of W
Recall both definition of sigmoid $s(v) = \frac{1}{1+e^{-v}}$, however other nonlinear functions can be used. For vector v,

$$s(v) = \begin{bmatrix} s(v_1) \\ s(v_2) \\ . \\ . \\ . \\ s(v_n) \end{bmatrix}$$

2.3 TRAINING

Usually stochastic or batch gradient descent:
Pick loss fn $L(z, y)$
Cost fn is $J(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x_i, Y_i)$ Usually there are many local minima!
Let w be a vector containing all the weights in V and W for the sake of mathematical notation.
Batch gradient descent:
$w \rightarrow$ vector of random weights
repeat
$w - \epsilon J(w)$
You need to be careful with the values of the random weights, don't make them too big or too small.

Stochastic gradient descent:
The same way as before, we typically also want to shuffle the data and like before iterate through the training points of the given example.
Naive Gradient computation: O(units * edges) time
Backpropagation: O(edges) time

2.4 COMPUTING GRADIENTS FOR ARITHMETIC EXPRESSIONS

2.5 THE BACK-PROPAGATION ALGORITHM

Note that we did above is a dynamic programming algorithm, as we are solving the smaller sub-problems before we begin to solve the overall problem.

v_i^T is row i of weight matrix V and recall $s'(y) = s(y)(1 - s(y))$
$h_i = s(v_i^T x)$, so $\nabla_{v_i} h_i = s'(v_i^T x) = h_i(1 - h_i)x$
$z_j = s(W_j^T h)$, so $\nabla_{w_j} z_j = s'(W_j^T h)h = z_j(1 - z_j)h$ and $\nabla_h z_j = z_j(1 - z_j)w_j$