CMPUT 367

Roderick Lan

Contents

Lecture 11 - Feb 15

1 Nonlinear Problems

(last lecture) Stack linear classifiers; use another linear classifier to classify obj based on results of stack
Weights/learnable params are w and b for each linear (sigmoid) classifier

Explain 1.0.1: How to get weights?

Manually "program" the weights
or
Use principled ML method to discover weights (ML training method)

Problem is NON-CONVEX (big problem in Deep Learning)
Symmetry of weights to prove non-convexity
Use Gradient Descent to find local minimas; improve minimas by performing Gradient Descent multiple times

Algorithm 1: GD update

Loop:
$w^{(new)} \leftarrow w^{(old)} - \alpha \frac{\partial J}{\partial w}\Big|_{w=w^{(old)}}$

$$\begin{aligned} h = \sigma(w^\top c + b) &= \frac{1}{1 + e^{-(w_1 c_1 + w_2 c_2 + b)}} \\ &= \frac{1}{1 + e^{-\left(w_1 \frac{1}{1+e^{-(w_{11}x_1 + w_{12}x_2 + b_1)}} + w_2 \frac{1}{1+e^{-(w_{21}x_1 + w_{22}x_2 + b_2)}} + b\right)}} \end{aligned}$$

Differentiable

$$\frac{\partial J}{\partial w_{11}} = \frac{\partial J}{\partial h} \cdot \frac{\partial h}{\partial w_{11}}$$

Need a way to systematically organize the derivatives in a neat way to gain **insight**

2 Neural Network

Building block is a Neuron; takes $d-$dimensional input

$$\mathbf{x} \to z \to y = f(z)$$

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$, f is the activation function (ie. sigmoid, ReLU, tanh, etc.)

ReLU most common

Arbitrary stacking of neurons is NOT convenient for both mathematically calculating derivatives

Laywise fully connected NN

Output of neuron is continuous
Balance width and height of neural net