Sunday, 25, 2016

- Since softmax turns evidence into prob dist it's used as the last layer even in complicated models.

- ~~Soft~~ i = class
  j = Summing index over all classes
  x = input image

- for this softmax is serving as a link func ~~out~~ shaping the output of our linear func into the form we want!!
- its tallying the evidence into prob for our input ing

$$\text{Softmax}(x) = \text{normalize}(\exp(x))$$

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

↳ exponentiating its inputs & then normalizing them

- No hypothesis has 0 or -ve weights
- The exp inc the weight for one unit of +ve evidence & reduce multiplicatively similarly for -ve evidence
- $y = \text{softmax}(Wx + b)$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

Regression

- Numpy ~~comp~~ does computations in C & thus cares about data types. There is overhead in switching b/w the two ~~though~~.
- Tensorflow does heavy lifting outside Python. You describe the graph in py & execute outsid of py.

Training

- Define what it means for the model to be bad?
  ↳ cost func, loss func etc
- minimize the error
- Cross-entropy

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

y = predicted prob ~~dits~~ dist

y' = true dist

measure ~~it~~ how inefficient our prediction is

→

Back Propagation

→ Reverse mode differentiation

→ get → ~~How to~~ gives the derivate of the output with respect to every single node

dMukherjee
25/09/16