Neural Networks: II. Model (Part 1)

Jan Bauer

jan.bauer@dhbw-mannheim.de

14.05.19

Neuron

A Neuron is a function "f $(= y)$" of an input "X" weighted by a vector of connection weights "W" completed by a bias "b" and associated to an activation function "ϕ".

$$\rightsquigarrow f(X, W, b) = \phi(XW + b)$$

- Several binary inputs produces a binary output.
- Question: Going to the concert tonight? Yes/No?
  - x_1: Will there be cheap beer?
  - x_2: WIll the sun beam?

$$
\begin{aligned}
y = \phi(xW + b) \quad &= \quad \begin{cases} 0 & xW + b \leq \text{threshold} \\ 1 & xW + b > \text{threshold} \end{cases} \\
&= \quad \begin{cases} 0 & x_1 w_1 + x_2 w_2 + b \leq \text{threshold} \\ 1 & x_1 w_1 + x_2 w_2 + b > \text{threshold} \end{cases}
\end{aligned}
$$

Limitatons

Logical operations:

- AND?
- OR?
- XOR? (Exclusive Or) **EXERCISE TIME**

Solution:

- Nonlinear functions
- More layer (Multilayer Perception)

Neural Network Architecture [Sketch]

A NN associates to an input X an output $y \equiv f(X, W)$,

- $f : \mathbb{R}^{D \times N} \to \mathbb{R}^D$
- $X = \begin{pmatrix} x^{(11)} & x^{(12)} & \dots & x^{(1N)} \\ x^{(21)} & x^{(22)} & \dots & x^{(2N)} \\ \vdots & \vdots & \ddots & \vdots \\ x^{(D1)} & x^{(D2)} & \dots & x^{(DN)} \end{pmatrix} = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(D)} \end{pmatrix}$ is $D \times N$
- $W = \begin{pmatrix} w^{(1)} & w^{(2)} & \dots & w^{(N)} \end{pmatrix}'$ is $N \times 1$
- $D = \#$data (i.e. training data)

Neuron [Sketch]

The j-th Neuron in layer i associates to an input $Z_{i,j}$ from Neuron k in layer $i-1$ an output $y_{i,j} \equiv f_{i,j}(f_{i-1,k}, W_{i,j})$, also called scores, s.t.

- $f_{i,j} : \mathbb{R}^{D \times N_{i-1}} \to \mathbb{R}^{D \times N_i}$
- $Z_{i,j} = f_{i-1,k} W_{i,j} + b_{i,j}$
- $f_{i-1,k}$ is $D \times N_{i-1}$
- $W_{i,j}$ is $N_{i-1} \times N_i$

Loss Functions

- Multiclass Support Vector Machine loss (**SVM loss**)

$$L_d \equiv \sum_{k \neq k^*} \max\left(0, y_{[d]}^{(k)} - y_{[d]}^{(k^*)} + \Delta\right)$$

  - we want the model to perform better than by a margin of Δ
  - in pratice: $\Delta = 1$ (will be clear in Part 4 - Regularization)
  - squared loss might perform better, i.e. $\sum \max(0, \cdot)^2$

Neural Network Architecture

Output f

- f gives us a score to which class the data fits best
  - Example: Linear function

- sloppy notation: $f \overset{?}{=} \phi$

- sloppy notation: Bias "trick" **QUESTION TIME**

Activation Functions

Common activation functions:

- identity:

$$\phi(x) = x$$

- Sigmoid function:

$$\phi(x) = \frac{1}{1 + \exp(-x)}$$

- Hyperbolic Tangent function (tanh):

$$\frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1}$$

- Rectified Linear Unit (ReLU):

$$\phi(x) = \max(0, x)$$

- Softmax

$$\phi(y_{[d]}^{(\hat{k})}) = \frac{\exp(y_{[d]}^{(\hat{k})})}{\sum_k \exp(y_{[d]}^{(k)})}$$

Softmax

- Sigmoid probabilities might appear counterintuitive (E.g. $xW = (0.9, 0.8, 0.4)'$)

- Often times used for the output layer

- Yields the predicted probability for the $\hat{k}$-th class, given a sample x and a weighting W

$$P(y_{[d]} = \hat{k} | x) = \frac{\exp(y_{[d]}^{(\hat{k})})}{\sum_{k=1}^{K} \exp(y_{[d]}^{(k)})}$$

- Hardmax: Assigns probability 1 or 0

Process of "feeding" a NN

training data vs test data etc.