Neural networks

CISC 5800
Professor Daniel Leeds

Two breeds of deep networks
Discriminative: \( \text{unit}^k(x) = (w^k)^T x + b = [0, 1] \)

**Neural networks / Convolutional neural networks**

Generative: \( \text{unit}^k(x) = P(x; \theta^k) = [0, 1] \)

**Bayes Nets / Deep Belief Nets**

Network architecture
Input layer:
- Compute based on initial features

“Hidden” layers
- Compute based on new features

“Output” layer
- Output final class or high-level features

Each unit takes inputs from past layer, outputs to next layer

Neural network building blocks
Individual unit “perceptron”:
- Typically logistic function \( \text{unit}(x) = g(h = w^T x + b) = \frac{1}{1 + e^{-h}} \)

Inter-layer computations
- Output \( r_{\text{unit level}}^{m-1} : \) \( r_k^m = g(\sum_j w_{kj}^m r_j^{m-1} + b_k^m) : \) \( r_k^m = g(W^m r_{\text{unit level}}^{m-1}) \)
- Parameters \( w_{\text{unit level}, \text{input level}}^{m} : W^m \)

Each unit takes inputs from past layer, outputs to next layer
Flow of calculation

Calculate output of each unit at layer 1 (based on input)
Calculate output of each unit at layer 2 (based on layer 1)
⋮
Calculate output of each unit at layer out (based on layer out - 1)

Parameters: \( w_{k,i}^m \): \( W^m \) - weights for every unit

Hyper-parameters:
- number of layers
- number of units per layer
- (sigmoid alternatives \( g(...) \) with hyper-parameters)
- learning step weight

Top layer units

\( r_{\text{classY}}^{\text{top}} \): Find the unit with \( r_{\text{top}} = 1 \) – that is your class
\( r_{\text{newFeatK}}^{\text{top}} \): Use outputs of all \( r_{\text{top}} \) for new classifier (e.g., SVM)

Neural Network units dividing feature space

Layer 1 unit

Layer 2 unit
Simple feedforward practice

Find $r_1$, $r_2$, $r_3$ Assume $b=0$

$x_1 = 0.1 \quad x_2 = 0.9$

Layer 1

Layer 2

What is $w_{1,2}^2$?

$w_{2,1}^1$?

Simple feedforward practice

Learning parameters: back-propagation

Training data: input $x$ and class $y$

Compute $x$’s output for all units, from layer 1 to layer out

Adjust weights in reverse order

$\Delta w$ for each unit at layer out (based on $y$)

$\Delta w$ for each unit at layer out-1 (based on layer out)

$\vdots$

$\Delta w$ for each unit at layer 1 (based on layer 2)

Parameters: $w_{kj}^m : W^m$ - weights for every unit

Hyper-parameters:

• number of layers

• number of units per layer

• (sigmoid alternatives $g(...)$ with hyper-parameters)

• $\epsilon$ learning step weight
Learning parameters: back-propagation

First: Change \( w \) in layer top for each unit

Want to minimize the error, as measured \( \sum_i (r_{top,i}^{top} - y_i)^2 \)

\[
\begin{align*}
\cdot r_k^{top} &= g(W_{top}^{top}r_{top-1}^{top-1}) \\
 &= \frac{1}{1 + \exp(-(W_{top}^{top}r_{top-1}^{top-1}))}
\end{align*}
\]