

Neural networks

CISC 5800
Professor Daniel Leeds

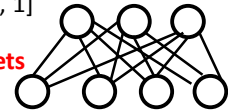
Two breeds of deep networks

Discriminative: $\text{unit}^k(\mathbf{x}) = (\mathbf{w}^k)^T \mathbf{x} + b = [0, 1]$

Neural networks / Convolutional neural networks

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

Bayes Nets / Deep Belief Nets



2

Network architecture

Input layer:

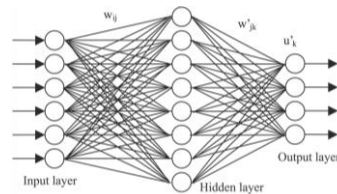
- Compute based on initial features

“Hidden” layers

- Compute based on new features

“Output” layer

- Output final class or high-level features



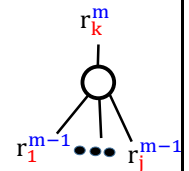
Each unit takes inputs from past layer, outputs to next layer

3

Neural network building blocks

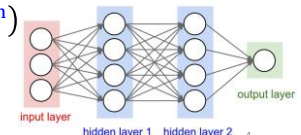
Individual unit “perceptron”:

- Typically logistic function $\text{unit}(\mathbf{x}) = g(h = \mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-h}}$



Inter-layer computations

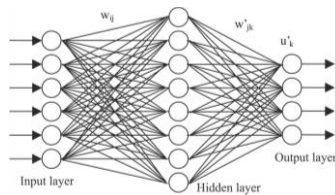
- Output $r_{\text{unit}\#}^{\text{level}}$: $r_k^m = g(\sum_j w_{k,j}^m r_j^{m-1} + b_k^m)$
- Parameters $w_{\text{unit}\#, \text{input}\#}^{\text{level}}$: $w_{k,j}^m$



4

Flow of calculation

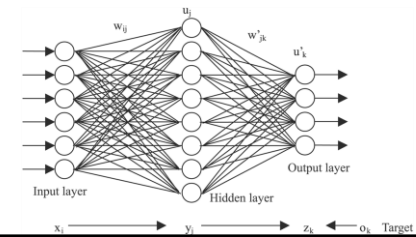
- Calculate output of each unit at layer 1 (based on input)
 Calculate output of each unit at layer 2 (based on layer 1)
 ⋮
 Calculate output of each unit at layer **out** (based on layer out-1)



5

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^{top} for new classifier (e.g., SVM)



6

Parameters: $w_{k,i}^m$ - weights for every unit

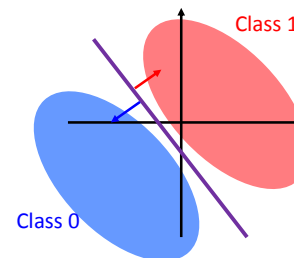
Hyper-parameters:

- number of layers
- number of units per layer
- (sigmoid alternatives $g(\dots)$ with hyper-parameters)
- learning step weight

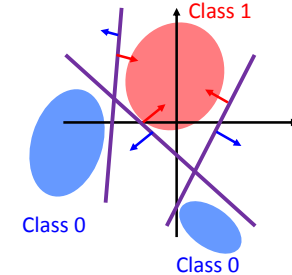
7

Neural Network units dividing feature space

Layer 1 unit



Layer 2 unit



8

Simple feedforward practice

Find r_1^1, r_2^1, r_1^2 Assume $b=0$

$x_1=0.1 \quad x_2=0.9$

$r_1^1 = \text{sigmoid}(0.1 \times 2 + 0.9 \times -1.5) = \text{sigmoid}(0.2 - 1.35) = \text{sigmoid}(-1.15) = 0.2$

$r_2^1 = \text{sigmoid}(0.1 \times 0 + 0.9 \times 2) = \text{sigmoid}(0 + 1.8) = \text{sigmoid}(1.8) = 0.85$

$r_1^2 = \text{sigmoid}(0.2 \times 1 + 0.85 \times -1) = \text{sigmoid}(0.2 - 0.85) = \text{sigmoid}(-0.65) = 0.3$

Simple feedforward practice

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

$w_{2,1}^1$? It is 0

Learning parameters: back-propagation

Training data: input x^i and class y^i

Compute x^i 's output for all units, from layer 1 to layer out

Adjust weights in reverse order

- Δw for each unit at layer out (based on y^i)
- Δw for each unit at layer out-1 (based on layer out)
- ⋮
- Δw for each unit at layer 1 (based on layer 2)

Parameters: $w_{k,i}^m$ - weights for every unit

Hyper-parameters:

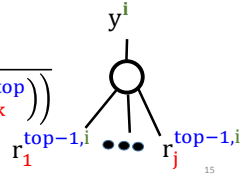
- number of layers
- number of units per layer
- (sigmoid alternatives $g(\dots)$ with hyper-parameters)
- ϵ learning step weight

Learning parameters: back-propagation

First: Change w in layer top for each unit

Want to minimize the error, as measured $\sum_i (r_k^{top,i} - y_k^i)^2$

$$r_k^{top} = g\left(\sum_j w_{kj}^{top, top-1} r_j^{top-1} + b_k^{top}\right)$$

$$= \frac{1}{1 + \exp\left(-\left(\sum_i w_{kj}^{top, top-1} r_j^{top-1} + b_k^{top}\right)\right)}$$


Δw at each layer

Calculate change to w 's at layer top

$$\Delta w_{kj}^{top} = \epsilon \underbrace{(1 - r_k^{top,i})}_{\text{Error correction}} \underbrace{(y^i - r_k^{top,i})}_{\text{input j effect}} r_j^{top,i} r_j^{top-1,i}$$

Define error signal δ : $\delta_k^{top,i} = (1 - r_k^{top,i})(y^i - r_k^{top,i}) r_k^{top,i}$

Send error signal back to layer m-1

16

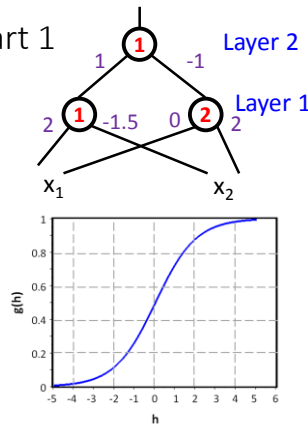
Simple feedback practice, part 1

Inputs: $x_1=0.1$ $x_2=0.9$
 Outputs: $r_1^1=0.3$ $r_2^1=0.8$
 $r_1^2=0.4$

$y=1$ $\epsilon=0.1$

Update layer 2 unit:

$$\Delta w_{kj}^{top} = \epsilon (1 - r_k^{top,i})(y^i - r_k^{top,i}) r_k^{top,i} r_j^{top-1,i}$$



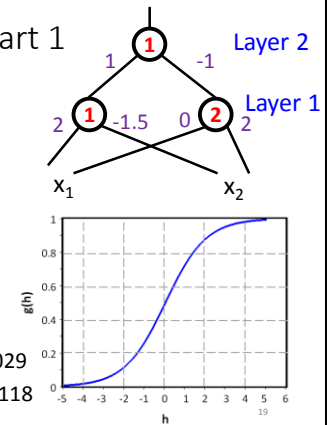
Simple feedback practice, part 1

• $x_1=0.1$ $x_2=0.9$
 • $r_1^1=0.2$ $r_2^1=0.8$
 • $r_1^2=0.3$

• $y=1$ $\epsilon=0.1$

$$\Delta w_{kj}^{top} = \epsilon (1 - r_k^{top,i})(y^i - r_k^{top,i}) r_k^{top,i} r_j^{top-1,i}$$

- $.1 \times (1 - 0.3) \times (1 - 0.3) \times 0.3 \times 0.2 = .1 \times .7 \times .7 \times .3 \times .2 = 0.0029$
- $.1 \times (1 - 0.3) \times (1 - 0.3) \times 0.3 \times 0.8 = .1 \times .7 \times .7 \times .3 \times .8 = 0.0118$



Δw at non-top layer

Top layer error signal δ : $\delta_k^{top,i} = (1 - r_k^{top,i})(y^i - r_k^{top,i})r_k^{top,i}$

Calculate change to w 's at layer $m < top$

$$\Delta w_{k,j}^m = \epsilon (1 - r_k^{m,i}) \underbrace{(\sum_n w_{n,k}^{m+1,i} \delta_n^{m+1,i})}_{\text{Error correction}} r_k^{m,i} \underbrace{r_j^{m-1,i}}_{\text{input j effect}}$$

Define error signal δ : $\delta_k^m = (1 - r_k^{m,i}) \sum_n (w_{n,k}^{m+1,i} \delta_n^{m+1,i}) r_k^{m,i}$

21

Simple feedback practice, part 2

Inputs: $x_1=0.1$ $x_2=0.9$
 Outputs: $r_1^1=0.2$ $r_2^1=0.8$
 $r_1^2=0.3$

$y=1$ $\epsilon=0.1$

Update layer 1 unit:

$$\delta_k^{top,i} = (1 - r_k^{top,i})(y^i - r_k^{top,i})r_k^{top,i}$$

$$\delta_k^{top,i} = (1-0.3) \times (1-0.3) \times 0.3 = .7 \times .7 \times .3 = .147$$

Unit 2:

$$\Delta w_{2,1}^1 = 0.1 \times (1-0.8) \times [-1 \times 0.147] \times 0.8 \times 0.1 = .1 \times .2 \times (-.147) \times .8 \times .1 = \mathbf{-0.00024}$$

New $w_{2,1}^1$: 0-0.00046 -> -0.00024

23

Alternative input transformations

- Traditional: Sum+sigmoid $g(\mathbf{w}^T \mathbf{x} + b)$
- Straight sum $\mathbf{w}^T \mathbf{x} + b$
- Sum+rectify
- Max (no weights!)

24

Alternative weights

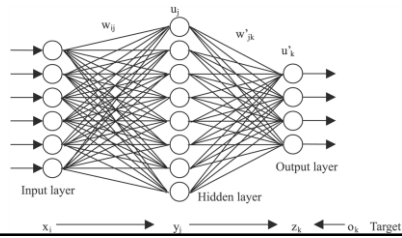
- Traditional: weight inputs from layer m-1
- Competition/Normalization: weight inputs from layer m
- Feedback: weight inputs from layer m+1

25

Top layer units

$r_{\text{class}Y}^{\text{top}}$ Find the unit with $r^{\text{top}}=1$ – that is your class

$r_{\text{newFeat}K}^{\text{top}}$ Use outputs of all r^{top} for new classifier (e.g., SVM)



26

Autoencoder

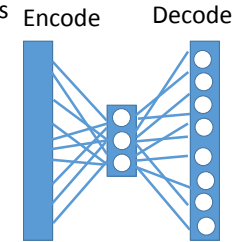
- Input data in large feature space (e.g., 1,000,000 features)

- Intermediate layer has small number of units (e.g., 100 units)

- Output layer has same number of units as input features (e.g., 1,000,000 units)

- Optimize network so output units produce same values as inputs

- Middle units are reduced feature space!



27

Convolutional neural networks

- Wait until end of semester!

28