Neural networks!

CISC 5800

Professor Daniel Leeds

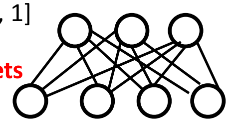
Two breeds of deep networks

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

Neural networks / Convolutional neural networks

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

Bayes Nets / Deep Belief Nets



Network architecture

Input layer:

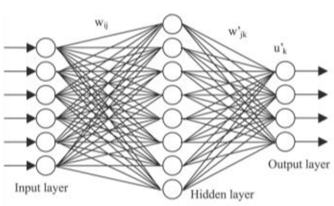
- Compute based on initial features

"Hidden" layers

- Compute based on new features

"Output" layer

- Output final class or high-level features

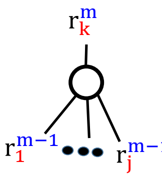


Each unit takes inputs from past layer, outputs to next layer

Neural network building blocks

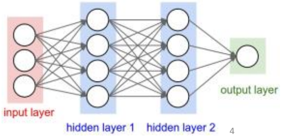
Individual unit "perceptron":

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



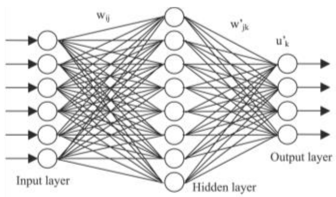
Inter-layer computations

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



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)

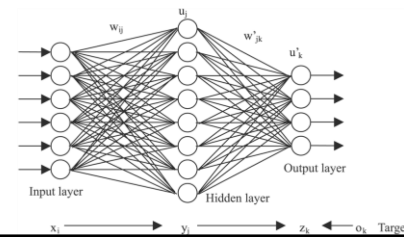


5

Top layer units

r^{top}_{classY} Find the unit with r^{top}=1 – that is your class

$r^{top}_{newFeatK}$ Use outputs of all r^{top} for new classifier (e.g., SVM)



6

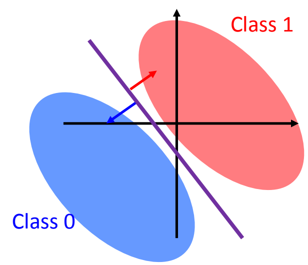
Parameters: $w^{m}_{k,i}$ - weights for every unit

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

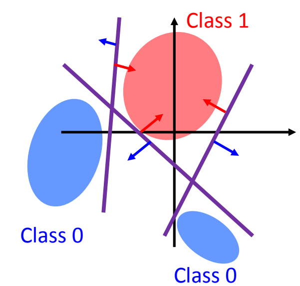
7

Neural Network units dividing feature space

Layer 1 unit



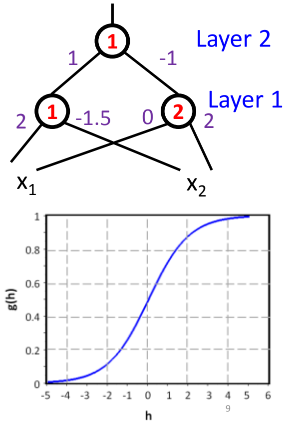
Layer 2 unit



8

Simple feedforward practice

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



x_1=0.1 x_2=0.9

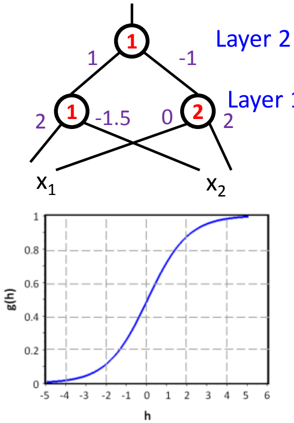
r_1^1=sigmoid(0.1x2+0.9x-1.5)=
sigmoid(0.2-1.35)=sigmoid(-1.15)=0.2

r_2^1=sigmoid(0.1x0+0.9x2)=
sigmoid(0+1.8)=sigmoid(1.8)=0.85

r_1^2=sigmoid(0.2x1+0.85x-1)=
sigmoid(0.2-0.85)=sigmoid(-0.65)=0.3

9

Simple feedforward practice



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

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

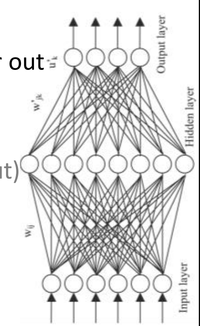
Learning parameters: back-propagation

Training data: input $\mathbf{x}^i$ and class $\mathbf{y}^i$

Compute $\mathbf{x}^i$'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(…) with hyper-parameters)
- ϵ learning step weight

12

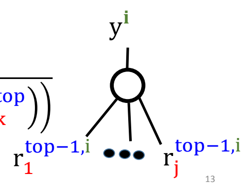
Learning parameters: back-propagation

First: Change **w** in layer top for each unit

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

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

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



13

Δw at each layer

Calculate change to w's at layer top

$$\Delta w_{k,j}^{top} = \epsilon\left(1 - r_k^{top,i}\right)\left(y^i - r_k^{top,i}\right) r_k^{top,i} r_j^{top-1,i}$$

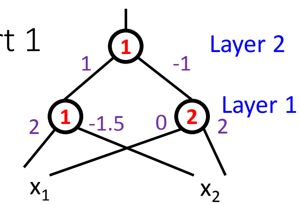
Error correction — input j effect

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

Send error signal back to layer m-1

14

Simple feedback practice, part 1



Inputs: x_1=0.2 x_2=0.8

Outputs: r_1^1=0.3 r_2^1=0.8

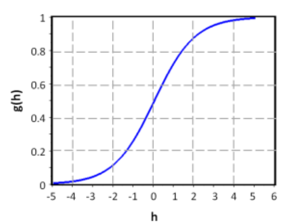
r_1^2=~~0.6~~ 0.4

y=1 ϵ=0.1

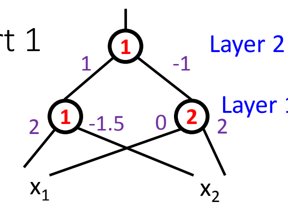
Update layer 2 unit:

$\Delta w_{k,j}^{top}$ =

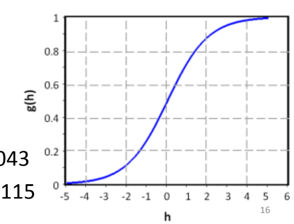
$$\epsilon\left(1 - r_k^{top,i}\right)\left(y^i - r_k^{top,i}\right) r_k^{top,i} r_j^{top-1,i}$$



Simple feedback practice, part 1



- x_1=0.2 x_2=0.8
- r_1^1=0.3 r_2^1=0.8
- r_1^2=0.4
- y=1 ϵ=0.1
- $\Delta w_{k,j}^{top}$ = $\epsilon\left(1 - r_k^{top,i}\right)\left(y^i - r_k^{top,i}\right) r_k^{top,i} r_j^{top-1,i}$
- .1x(1-0.4)x(1-0.4)x0.4x0.3=.1x.6x.6x.4x.3=0.0043
- .1x(1-0.4)x(1-0.4)x0.4x0.8=.1x.6.x.6x.4x.8=0.0115



16

Δw at non-top layer

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

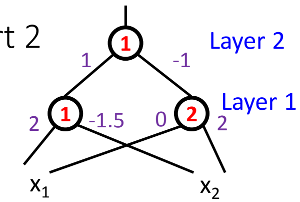
Calculate change to w's at layer $m<top$

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

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

17

Simple feedback practice, part 2

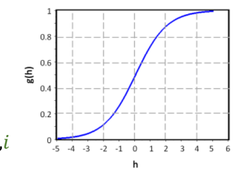


Inputs: x_1=0.2 x_2=0.8

Outputs: r_1^1=0.3 r_2^1=0.8

r_1^2=~~0.6~~ 0.4

y=1 ϵ=0.1



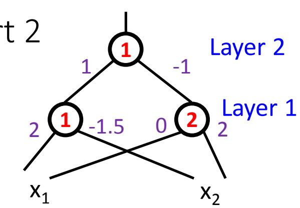
Update layer 1, unit 2:

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

$\Delta w_{k,j}^{m} = \epsilon\left(1 - r_k^{m,i}\right)\left(\sum_n w_{n,k}^{m+1,i}\delta_n^{m+1,i}\right) r_k^{m,i} r_j^{m-1,i}$

18

Simple feedback practice, part 2

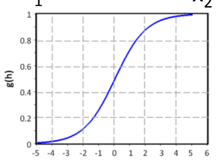


Inputs: x_1=0.2 x_2=0.8

Outputs: r_1^1=0.3 r_2^1=0.8

r_1^2=0.4

y=1 ϵ=0.1



Update layer 1 unit:

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

$\delta_k^{top,i}$=(1-0.4)x(1-0.4)x0.4=.6x.6x.4 = .144

Unit 2:

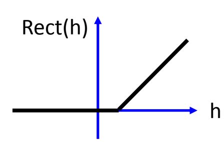
$\Delta w_{2,1}^{1}$=0.1x(1-0.8)x[-1x0.144]x0.8x0.2=.1x.2x(-.144)x.8x.2=**-0.00046**

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

19

Alternative input transformations

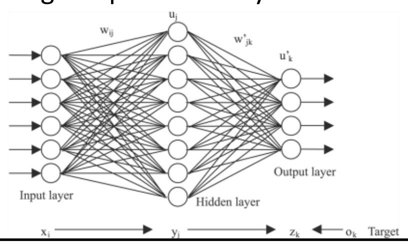
- Traditional: Sum+sigmoid g($\mathbf{w}^T\mathbf{x}$ + b)
- Straight sum $\mathbf{w}^T\mathbf{x}$ + b
- Sum+rectify
- Max (no weights!)



20

Alternative weights

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



Top layer units

r^{top}_{classY} Find the unit with r^{top}=1 – that is your class

$r^{top}_{newFeatK}$ Use outputs of all r^{top} for new classifier (e.g., SVM)