APRENDIZADO DE MÁQUINA

NEURAL NETWORKS

PROF. RONALDO CRISTIANO PRATI

RONALDO.PRATI@UFABC.EDU.BR

BLOCO A, SALA 513-2

MODELOS COMPLEXOS

- Suponha que tenhamos um problema de classificação complexo, que a regressão logística não funciona bem
- Como vimos, uma alternativa é criar termos polinomiais
- Se tivermos um número pequeno de atributos, a estratégia é adequada, mas e se tivermos, digamos, 100?
  - Se adicionarmos termos quadráticos, o número de novos atributos cresce na ordem de $O(n^2)$!
  - Se adicionarmos termos cúbicos, o número de novos atributos cresce na ordem de $O(n^3)$!
- Precisamos de outras maneiras de criar modelos complexos.

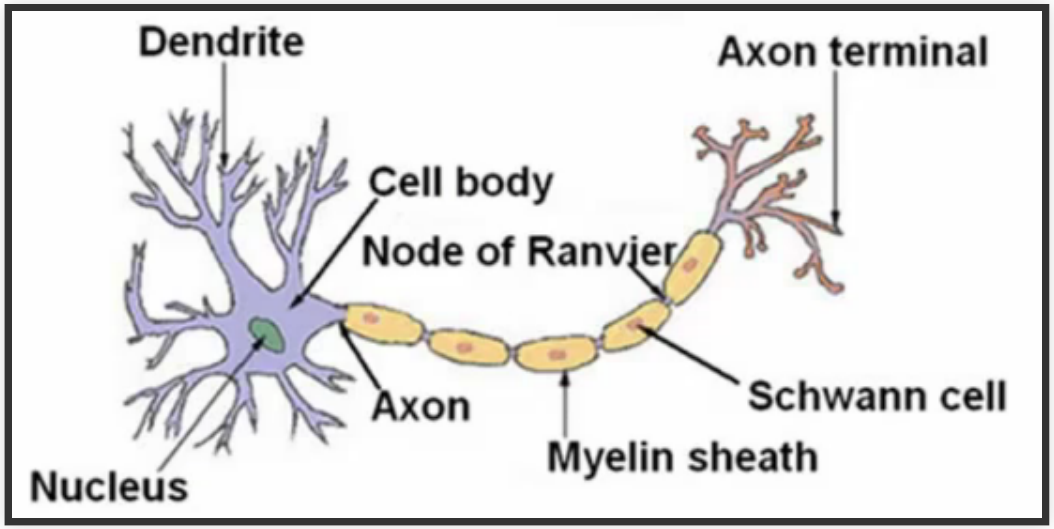
REDES NEURAIS

- Redes neurais (NNs) foram originalmente motivadas como modelos para replicar a funcionalidade do cérebro
- Já que queremos constuir um modelo de aprendizado, porque não imitar o cérebro?
  - Nosso enfoque aqui no curso é como um técnica de aprendizado de máquina

HISTÓRICO

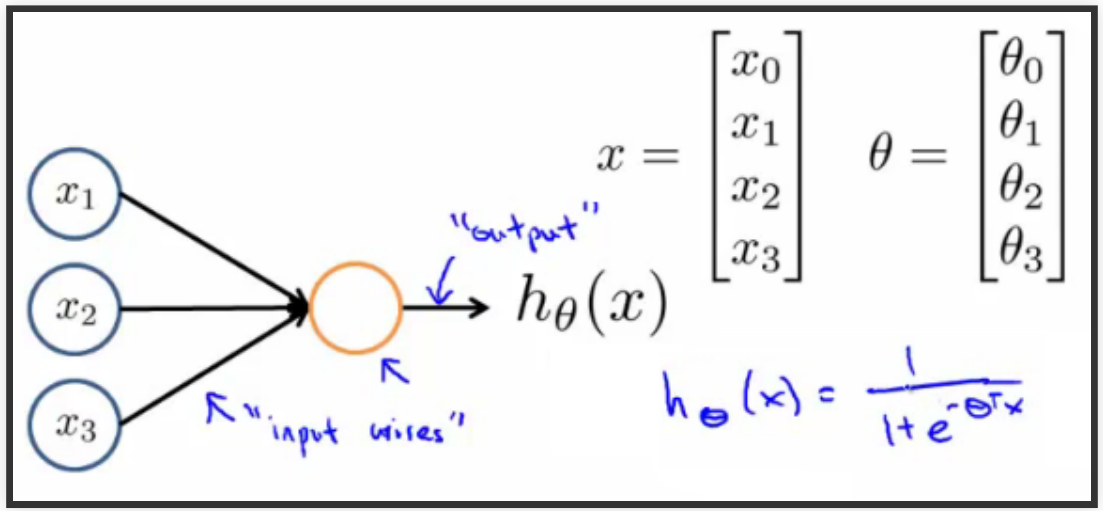
- Primeiro modelo nos anos 40 (McCulloch & Pitts)
- Muita euforia, mas limitações levaram a um período de pouco avanço
- Novos avanços entre o fim dos anos 70 e início dos anos 80 provocaram um ressurgimento
- Muito usadas nos anos 80 e 90
- Popularidade diminuiu no final dos anos 90 (falta de recursos computacionais)
- Ressurgimento nos últimos anos com redes profundas

NEURÔNIO



- O neurônio recebe uma ou mais entradas pelos dendritos
- Faz o processamento
- Envia a saída para o axio
- A comunicação é feita por meio de "spikes" elétricos (pulso elétrico pelo axônio para outros neurônios)

NEURÔNIO ARTIFICIAL



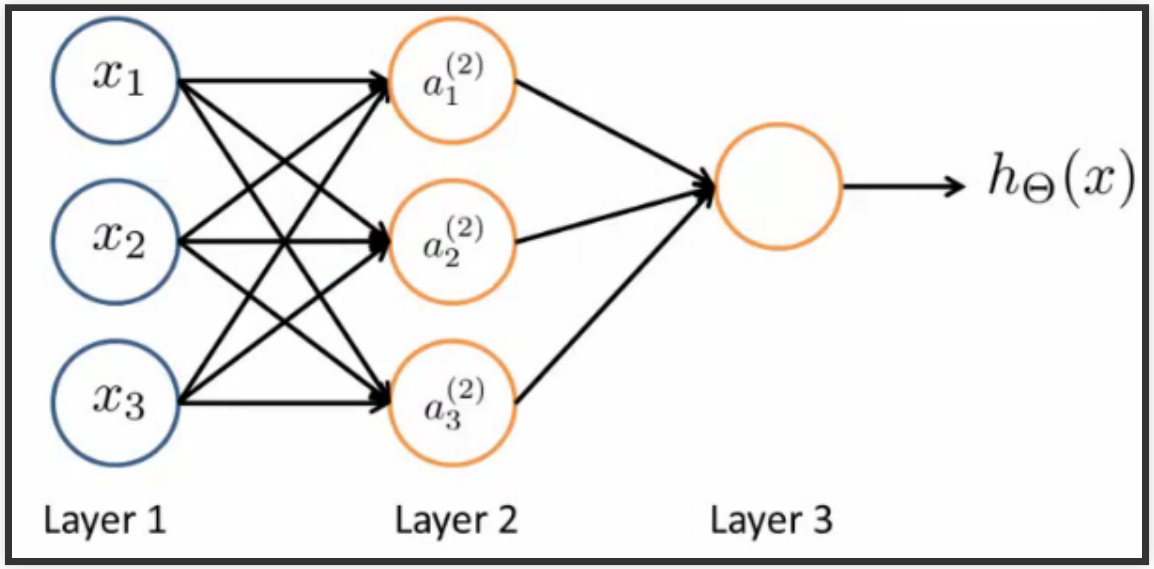
- Em uma rede neural artificial, um neurônio é um unidade logística
  - Alimenta a rede pela conexão de entrada
  - A unidade logística faz a computação
  - Envia a saída pela conexão de saída

NEURÔNIO ARTIFICIAL

- Um modelo muito simplificado da computação de um neurônio real
- Normalmente inclue uma entrada x_0, cujo valor é igual a 1
  - Essa entrada é chamada de **bias** (não confundir com o bias-variância)
- Esse é um neurônio com uma função de ativação sigmoide (logistica).
  - Outras funções de ativação pode ser usadas
- O vetor θ também é chamado de **vetor de pesos** do modelo.

REDES NEURAIS

- Neurônios aritificiais podem ser combinados em redes neurais



REDES NEURAIS

- Primeira camada é a **camada de entrada**
- Camada finald é a **camada de saída**
  - Ela produz o valor computado pela hipótese
- A(s) camada(s) do meio são chamadas de **camadas ocultas**
  - Não observamos os valores computados nas camadas intermediárias
  - Podemos ter mais de uma camada ocultas

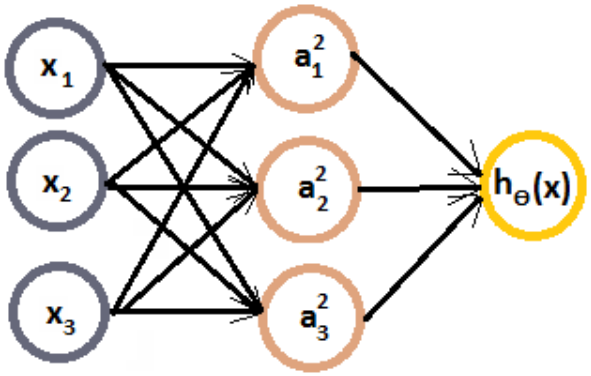
NOTAÇÃO

- a_i^j ativação da unidade i da camada j
  - a_1^2 é a ativação da 1a. unidade da segunda camada
  - Por ativação, nos referimos ao valor que é calculado como saída daquele unidade
- θ^j matriz de parâmetros controlando a função de mapeamento da camada j para a $j+1$
  - Se a rede tem s_j nós na camada j e s_{j+1} nós na camada $j+1$, então θ_j tem dimensões $[s_{j+1}\times s_j+1]$
  - s_{j+1} nós na camada $(j+1)$ e s_j na camada j mais o bias

COMPUTAÇÃO

- Quais são as computações que ocorrem?
  - Temos que calcular a ativação para cada nó
  - A ativação depende de
    - A(s) entrada(s) de cada nó
    - Os parâmetros associados com cada nó (do vetor θ associado a cada camada)

COMPUTAÇÃO



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

COMPUTAÇÃO (VETORIZAÇÃO)

- A implementação (e o entendimento) de uma rede neural pode ser facilitada utilizando vetorização. Vamos definir z_1^2 como

$$z_1^2 = \Theta_{10}^1 x_0 + \Theta_{11}^1 x_1 + \Theta_{12}^1 x_2 + \Theta_{13}^1 x_3$$

- Que significa $a_1^2 = g(z_1^2)$. Similarmente, podemos definir z_2^2 e z_3^2. Podemos definir os vetores:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^2 = \begin{bmatrix} z_1^2 \\ z_2^2 \\ z_3^2 \end{bmatrix}$$

COMPUTAÇÃO (VETORIZAÇÃO)

- Podemos vetorizar a computação em 2 etapas:
  - $z^2 = \theta^1 x$, em que θ^1 é a matriz definida anteriormente e x o vetor de entrada
  - $a^2 = g(z^2)$, ou seja, aplicamos $g()$ ao vetor z^2
- Tendo calculado a^2, podemos definir z^3 e $h_\theta(x) = a^3$ como:
  - $z^3 = \theta^2 a^2$, em que θ^1 é a matriz definida anteriormente e x o vetor de entrada
  - $a^3 = g(z^3)$, ou seja, aplicamos $g()$ ao vetor z^3

COMPUTAÇÃO (VETORIZAÇÃO)

- Este processo é chamado de **propagação forward**
- Começa com a ativação da camada de entrada, i.e., o vetor x como entrada
- Propaga para frente e calcula a ativação de cada camada sequencialmente
- Essa é uma versão vetorizada da implementação

REDE NEURAL "APRENDE" SUEUS PRÓPRIOS ATRIBUTOS

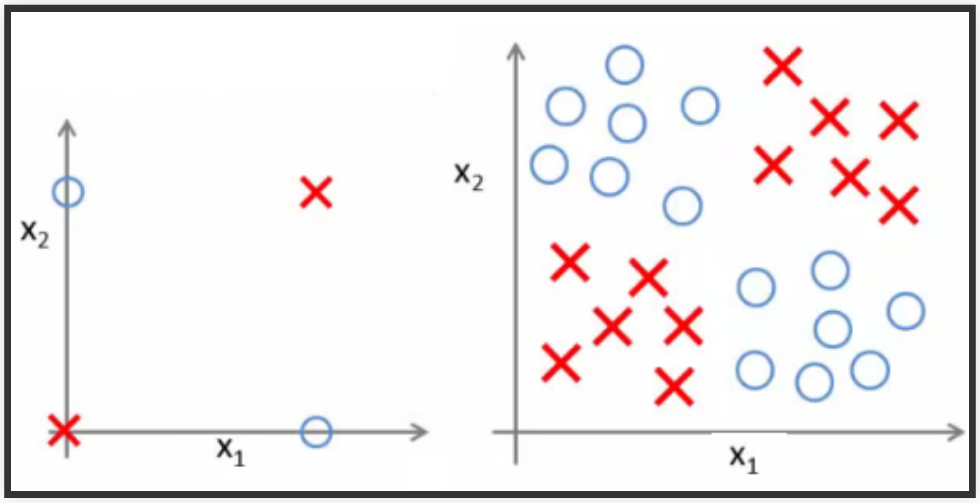
- A última camada é uma regressão logistica
- Mas ao invés de ser aplicada aos atributos originais, ela é aplicada à saída a^2 da camada anterior. O mesmo acontece com a camada anterior
- Dessa maneira, ao invés de estar restrita aos atributos de entrada, a rede neural pode "aprender" seus próprios atirbutos para a regressão logística
- Dependendo dos parâmetros de cada camada, podemos aprender combinações interessantes!
- As camadas oculta fazem o papel de criar novos atributos, ao invés de criamos manualmente como fazíamos anteriormente

OUTRAS CONFIGURAÇÕES

- Além da rede vista aqui, outras arquiteturas (topologias) são possíveis
  - Mais/menos nós por camada
  - Mais chamadas
  - Outras funções de ativação

XOR

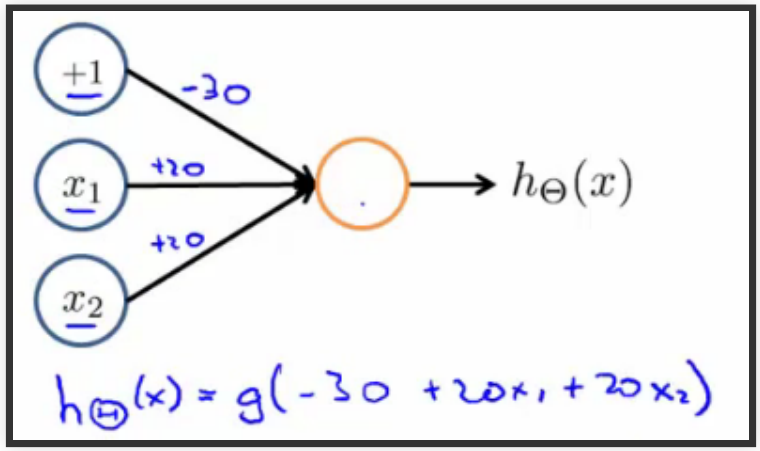
- Um exemplo simples de um problema de classificação não linar é o ou exclusivo (xor)



- Esta função não pode ser computada utilizando somente um neurônio

AND

- Vamos começar pelo exemplo de predizer x_1 AND x_2, que é o operador e lógico e só é verdade se ambos x_1 e x_2 são 1.



  - Depois do treinado, os pesos convergem para

$$\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \end{bmatrix}$$

AND

$$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$$

$$x_1 = 0 \ \ and \ \ x_2 = 0 \ \ then \ \ g(-30) \approx 0$$

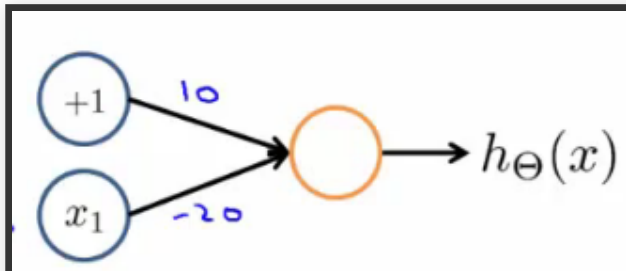
$$x_1 = 0 \ \ and \ \ x_2 = 1 \ \ then \ \ g(-10) \approx 0$$

$$x_1 = 1 \ \ and \ \ x_2 = 0 \ \ then \ \ g(-10) \approx 0$$

$$x_1 = 1 \ \ and \ \ x_2 = 1 \ \ then \ \ g(10) \approx 1$$

NOT

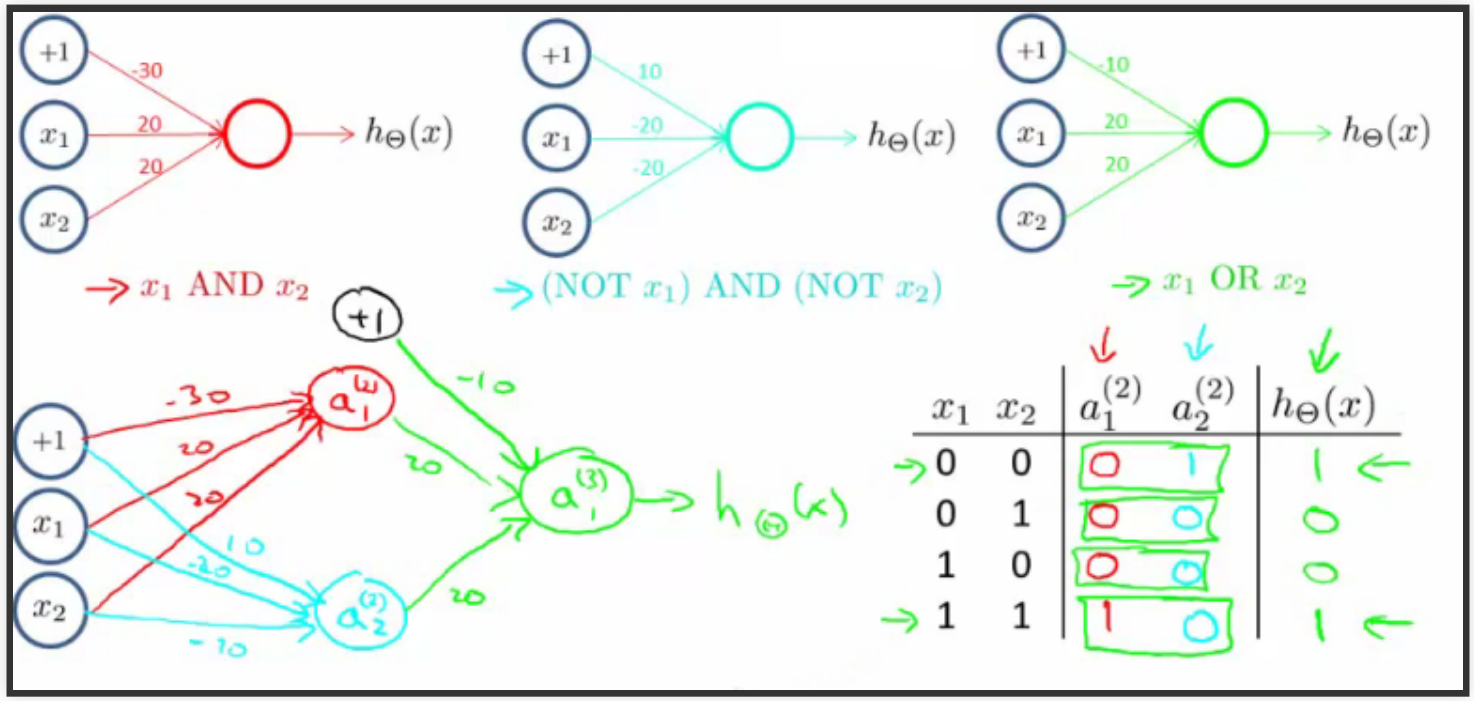
- Similarmente, podemos criar uma rede para calcular o não



x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

XNOR

- Podemos combinar as duas para criar o XNOR



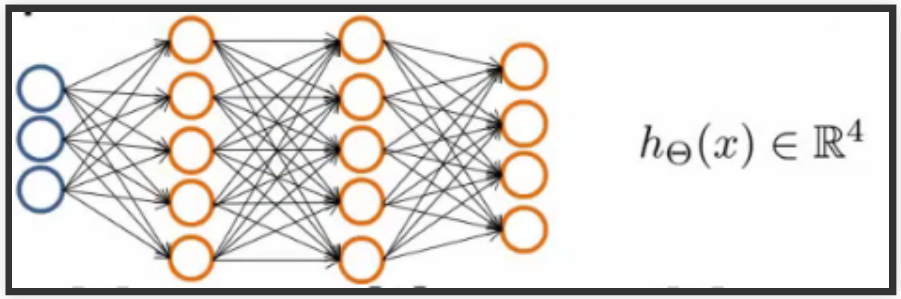
VIDEO

Video 1

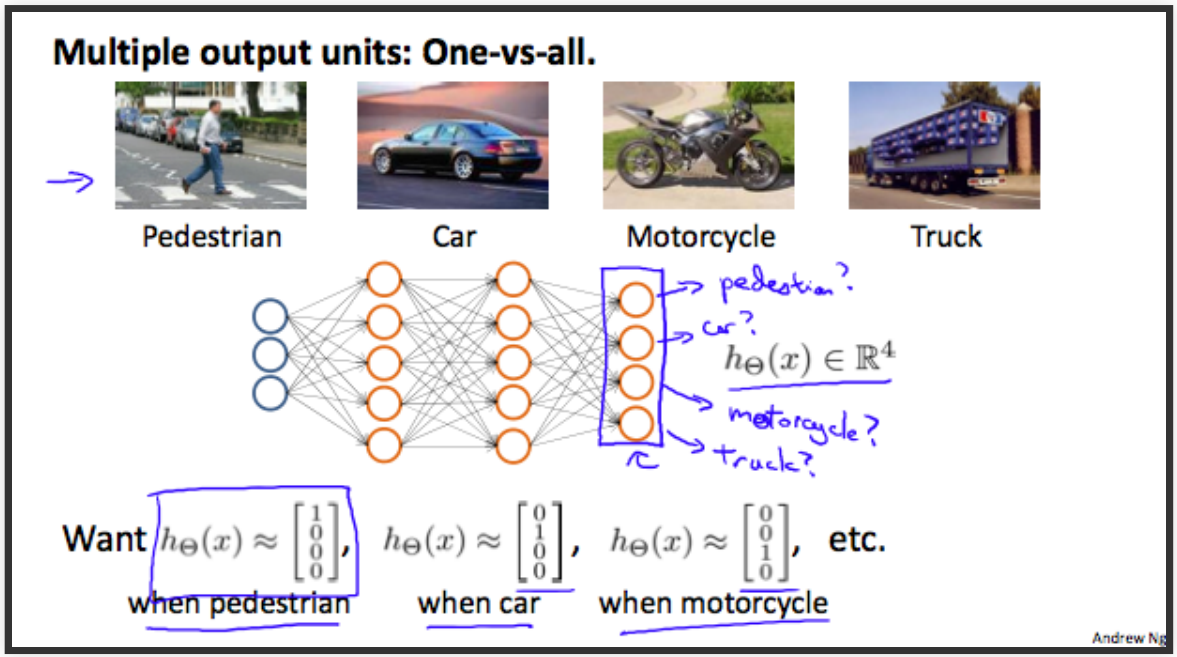
Video 2

CLASSIFICAÇÃO MULTICLASSE

- Uma extensão do "um-contra-todos"
- A saída da rede é um vetor um nó para cada classe



CLASSIFICAÇÃO MULTICLASSE



CLASSIFICAÇÃO MULTICLASSE

- Redefinimos nossas classes y como um vetor que representa uma classe diferente, com somente um 1 indicando qual é a classe, e 0 nas demais posições

$$y^{(i)} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix},$$

  - As camadas internas podem ser representadas como

$$\begin{bmatrix}x_0\\x_1\\x_2\\\cdots\\x_n\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(2)}\\a_1^{(2)}\\a_2^{(2)}\\\cdots\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(3)}\\a_1^{(3)}\\a_2^{(3)}\\\cdots\end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix}h_\Theta(x)_1\\h_\Theta(x)_2\\h_\Theta(x)_3\\h_\Theta(x)_4\end{bmatrix}$$

CLASSIFICAÇÃO MULTICLASSE

- A hipótese dá um vetor como saída

$$h_\Theta(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- Que no nosso exemplo é a terceira classe, ou $h_\Theta(x) = 3$, que representa a moto.