

Deep Learning

Semana 01 - Aula 03

Renato Assunção - DCC - UFMG



Roteiro desta aula

- Retomando classificação com duas classes
 - Com várias classes: máxima verossimilhança e logística
 - Problema de regressão: aptos e características; máxima verossimilhança
 - Função de custo = - log-verossimilhança
- Redes neurais com múltiplas camadas
- Longa explicação da notação e conceitos
- Exemplos com redes rasas (poucas camadas e poucos neurônios)
- Um digressão sobre porque as redes neurais funcionam:
 - Teorema de Kolmogorov
 - Teorema de aproximação universal

Recapitulando...

- Vimos regressão logística e perceptron: algoritmos para classificação supervisionada em duas classes:
 - Temos dados estatísticos: coleção de exemplos ou casos
 - Duas classes rotuladas como $Y = 0$ ou 1
 - Para cada exemplo: inputs ou features num vetor \mathbf{x}
 - Os mesmos inputs devem ser medidos em cada exemplo
- Objetivo: usar os dados para obter uma boa representação de $\mathbb{P}(Y = 1|\mathbf{x})$
- Esta probabilidade condicional é uma função matemática dos inputs \mathbf{x} .
- Dados os inputs \mathbf{x} , obtemos $\mathbb{P}(Y = 1|\mathbf{x})$

Como usar esta probabilidade?

- De posse da função $\mathbb{P}(Y = 1|\mathbf{x})$, fazemos classificações de novos exemplos onde Y não é conhecido.
 - Recebemos x
 - Calculamos $\mathbb{P}(Y = 1|\mathbf{x})$
 - Se for aprox 1, classifique na categoria 1.
 - É uma predição que pode ou não se confirmar.
 - Se for aprox 0, classique na categoria 0.
 - Se for aprox $\frac{1}{2}$
 - neste exemplo, o input x não fornece informação suficiente para prever a resposta Y . É um caso em que a classificação tem grande incerteza.

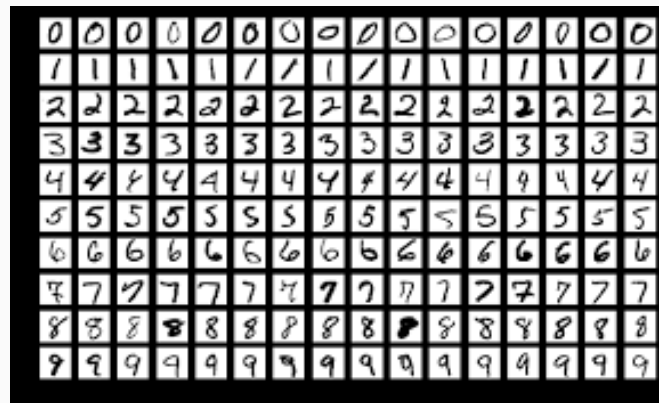
Expandindo um pouco os problemas

- Existem dois problemas adicionais intimamente relacionados com a classificação supervisionada em duas classes:
 - Classificação em $K > 2$ categorias
 - Regressão
- Estes dois problemas diferem do anterior pela estrutura da resposta Y
- Classificação em $K > 2$ categorias:
 - Y tem mais que duas classes: $Y = 0, 1, 2, \dots, K-1$
 - O resto é igual
- Regressão:
 - Y é uma variável contínua, o resto é igual

Classificação multi-classes

Classificação com k categorias

- Exemplo canônico: MNIST
- Dados são 70 mil imagens de dígitos manuscritos: cada imagem, um único dígito.
- $Y = 0, 1, 2, \dots, 9$ (resposta é o dígito exibido na imagem)
- Input: o tom de cinza (0-255) em cada pixel da imagem, empilhados como vetor



Criadores do MNIST

- Yann Le Cunn, Corinna Cortes, Christopher Burges
- Le Cunn é Silver Professor do Instituto Courant de Ciências Matemáticas da New York University e Chief AI Scientist no Facebook.
- Muito reconhecido por seu trabalho pioneiro em reconhecimento óptico de caracteres e visão computacional.
- Um dos principais criadores das redes neurais convolucionais (CNN), tópico da semana 3.
- Ganhador do Turing Award de 2019 com Hinton e Bengio

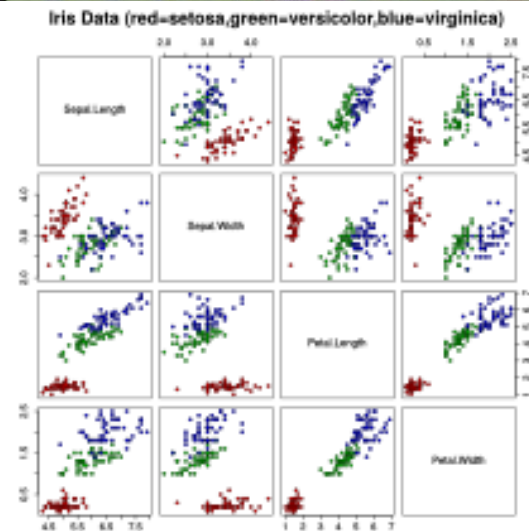


Segundo exemplo canônico

- Iris dataset
- Dados de 150 flores
- Três categorias (espécies) de flores:
 - Iris setosa, Iris virginica, Iris versicolor
- Inputs: 4 variáveis:
 - comprimento e largura da pétala
 - Comprimento e largura da sépala
- Objetivo: criar um modelo para distinguir as espécies umas das outras com base nos 4 inputs.

The Iris Dataset

Collected by Ronald
Fisher in 1936



Sir Ronald Aylmer Fisher

- Iris dataset: usado pelo estatístico britânico Sir Ronald Fisher em seu artigo de 1936, *The use of multiple measurements in taxonomic problems*, como um exemplo de análise discriminante linear.
- Fisher foi também um dos maiores geneticistas da história, responsável por unir Darwin e Mendel de forma coerente.



Log-verossimilhança com DUAS classes

- m exemplos, duas classes: LOG-verossimilhança:

$$\ell(\mathbf{w}) = \log(L(w_0, w_1, \dots, w_n))$$

$$= \log \left(\prod_{i=1}^m \mathbb{P}(Y_i = y_i) \right)$$

$$= \log \left(\prod_{i=1}^m \sigma(\mathbf{x}_i)^{y_i} (1 - \sigma(\mathbf{x}_i))^{1-y_i} \right)$$

$$= \sum_{i=1}^m \log(\sigma(\mathbf{x}_i)^{y_i} (1 - \sigma(\mathbf{x}_i))^{1-y_i})$$

$$= \sum_{i=1}^m (y_i \log(\sigma(\mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i)))$$

- sendo que $\sigma(x_i) = \mathbb{P}(Y_i = 1 | \mathbf{x}_i) = \frac{1}{1 + e^{-(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})}}$

Olhando um único exemplo com DUAS classes

- Log-verossimilhança com m exemplos: soma sobre exemplos individuais
- Vamos olhar um único exemplo, o exemplo i
- Temos $\sigma(x_i) = \mathbb{P}(Y_i = 1 | \mathbf{x}_i) = \frac{1}{1 + e^{-(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})}}$

- A log-verossimilhança é a soma de m termos:

$$\ell(\mathbf{w}) = \sum_{i=1}^m (y_i \log(\sigma(\mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i)))$$

$$y_i \log(\sigma(\mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i)) = \begin{cases} \log(\sigma(\mathbf{x}_i)), & \text{se } y_i = 1 \\ \log(1 - \sigma(\mathbf{x}_i)), & \text{se } y_i = 0 \end{cases}$$

Com K classes o natural seria ter cada termo somando o log(probabil) da classe realmente observada no exemplo i

A log-verossimilhança será exatamente isto. Vamos ver...

Estrutura estocástica para o caso multi-classe

- Em cada exemplo, a resposta Y é um rótulo indicando sua classe

$$Y_i = \begin{cases} 1, & \text{com probab } \sigma_1 \\ 2, & \text{com probab } \sigma_2 \\ \vdots & \\ K, & \text{com probab } \sigma_K \end{cases}$$

As probabilidades de cada classe

- Com duas classes, tínhamos apenas uma probabilidade de sucesso

$$\mathbb{P}(Y = 1|\mathbf{x}) = \sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}'\mathbf{x}}}$$

- A outra probabilidade era $1 - \sigma(\mathbf{x})$
- A probabilidade de sucesso é função dos inputs \mathbf{x} : diferentes \mathbf{x} , diferentes sigmas
- Precisamos especificar agora K probabilidades, todas dependendo dos inputs \mathbf{x} :

$$\mathbb{P}(Y = k|\mathbf{x}) = \sigma_k(\mathbf{x})$$

- tais que

$$\sigma_1(\mathbf{x}) + \dots + \sigma_K(\mathbf{x}) = 1$$

A verossimilhança

- Suponha que, de alguma forma, especificamos $\sigma_1(\mathbf{x}), \dots, \sigma_K(\mathbf{x})$
- Qual a chance de observar uma certa sequência de classes?
- Por exemplo, com $K=3$ classes, e 5 exemplos

$$\begin{aligned} L &= \mathbb{P}(Y_1 = 3|\mathbf{x}_1)\mathbb{P}(Y_2 = 1|\mathbf{x}_2)\mathbb{P}(Y_3 = 2|\mathbf{x}_3)\mathbb{P}(Y_4 = 1|\mathbf{x}_4)\mathbb{P}(Y_5 = 3|\mathbf{x}_5) \\ &= \sigma_3(\mathbf{x}_1)\sigma_1(\mathbf{x}_2)\sigma_2(\mathbf{x}_3)\sigma_1(\mathbf{x}_4)\sigma_3(\mathbf{x}_5) \\ &= \prod_{i=1}^m \prod_{k=1}^3 \sigma_k(\mathbf{x}_i)^{I[y_i=k]} \end{aligned}$$

A LOG-verossimilhança

- Basta tomar o log agora:
- Produtos viram somas:

$$\begin{aligned}\ell &= \log \left[\prod_{i=1}^m \prod_{k=1}^3 \sigma_k(\mathbf{x}_i)^{I[y_i=k]} \right] \\ &= \sum_{i=1}^m \sum_{k=1}^3 \left[\log \sigma_k(\mathbf{x}_i)^{I[y_i=k]} \right] \\ &= \sum_{i=1}^m \sum_{k=1}^3 \left[I[y_i = k] \log \sigma_k(\mathbf{x}_i) \right]\end{aligned}$$

- Nosso exemplo fica então

$$\ell = \log(\sigma_3(\mathbf{x}_1)) + \log(\sigma_1(\mathbf{x}_2)) + \log(\sigma_2(\mathbf{x}_3)) + \log(\sigma_1(\mathbf{x}_4)) + \log(\sigma_3(\mathbf{x}_5))$$

- Isto é, cada exemplo contribui com o log da probab da sua classe observada

As probabilidades das classes: de duas para K classes

- No caso de duas classes:

$$\sigma(\mathbf{x}_i) = \mathbb{P}(Y_i = 1 | \mathbf{x}_i) = \frac{1}{1 + e^{-(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})}}$$

- A probabilidade da classe 0 é obtida por subtração
- Podemos escrever

$$\frac{1}{1 + e^{-(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})}} = \frac{e^{(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})}}{1 + e^{(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})}} \propto e^{(w_0 + w_1 x_{1i} + \dots + w_n x_{ni})} = e^{\mathbf{w}' \mathbf{x}_i}$$

- Para o caso multi-classes, especificamos um vetor de pesos para cada uma das K classes: $\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_K$

Especificando as probabilidades das classes: softmax

- Para o caso multi-classes, especificamos um vetor de pesos para cada uma das K classes: $\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_K$
- As probabilidades $\sigma_k(\mathbf{x}_i) = \mathbb{P}(Y_i = k | \mathbf{x}_i) \propto e^{\mathbf{w}'_k \mathbf{x}_i}$
- Elas devem somar 1. Basta normalizarmos agora (modelo softmax):

$$\sigma_k(\mathbf{x}_i) = \mathbb{P}(Y_i = k | \mathbf{x}_i) = \frac{e^{\mathbf{w}'_k \mathbf{x}_i}}{\sum_{j=1}^K e^{\mathbf{w}'_j \mathbf{x}_i}}$$

Resultado final: log-verossimilhança para multi-classe

- Combinando a log-verossimilhança de antes com esta expressão para as probabilidades das classes, temos

$$\begin{aligned}\ell &= \ell(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K) \\ &= \sum_{i=1}^m \sum_{k=1}^K [I[y_i = k] \log \sigma_k(\mathbf{x}_i)] \\ &= \sum_{i=1}^m \sum_{k=1}^K \left[I[y_i = k] \log \left(\frac{e^{\mathbf{w}'_k \mathbf{x}_i}}{\sum_{j=1}^K e^{\mathbf{w}'_j \mathbf{x}_i}} \right) \right]\end{aligned}$$

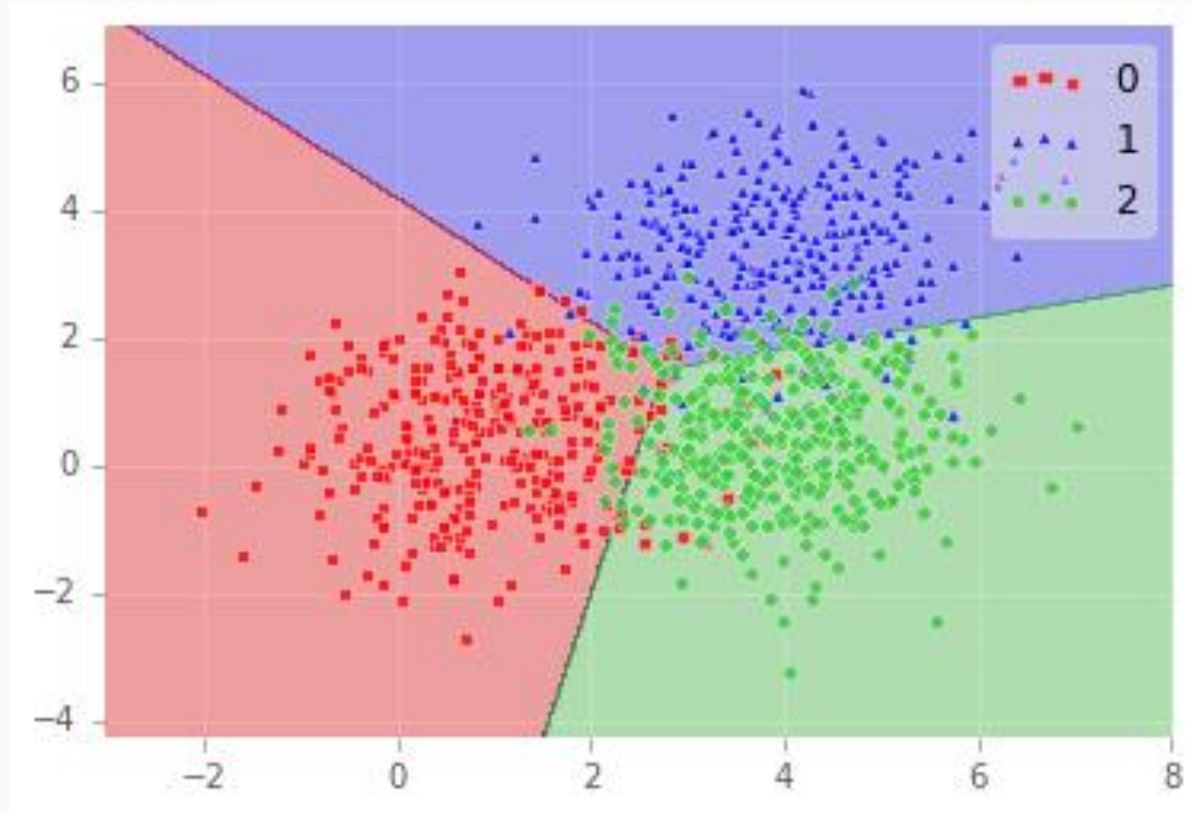
Como obter o estimador de máxima verossimilhança?

- Método numérico de Newton

$$\mathbf{w}^{t+1} = \begin{bmatrix} w_0^{t+1} \\ w_1^{t+1} \\ \vdots \\ w_n^{t+1} \end{bmatrix} = \begin{bmatrix} w_0^t \\ w_1^t \\ \vdots \\ w_n^t \end{bmatrix} - \left[\underbrace{J^2(\mathbf{w}^t)}_{\text{matriz der. parciais 2a ordem de J}} \right]^{-1} \underbrace{\nabla J(\mathbf{w}^t)}_{\text{vetor gradiente de J}}$$

- Vetor gradiente da log-verossimilhança.
- Se temos K classes e p inputs, teremos vetor gradiente de dimensão $K^*(p+1)$ -dim
- Matriz hessiana de derivadas parciais de segunda ordem: $K^*(p+1) \times K^*(p+1)$

A regra de decisão e os decision boundaries



Problema de regressão

O outro problema: regressão

- Exemplo canônico: preço de imóveis residenciais
- Variável resposta Y:
 - Preço de venda de um imóvel
 - Resposta Y não é uma variável categórica
 - É uma variável contínua: faz sentido somar e subtrair valores
- Coletamos preços de 1500 imóveis a venda no mercado de BH.
- Alguns são caros, outros são baratos.
- O que faz com que os preços dos imóveis variem?
- As três coisas mais importantes que afetam o valor de um imóvel...

Localização...

- Dividir a cidade em pequenas áreas.
- Outra abordagem mais simples: criar proxy
 - Localização é status socio-econômico;
 - Status é mensurado por renda.
 - Renda é medida pelo IBGE em 2000 pequenas áreas da cidade.
 - Renda do “chefe do domicílio”.
- Então: “localização” = renda média da região onde está o imóvel.

Outras características do imóvel

- Ano da construção
- Área total do imóvel
- Número de quartos
- Número de suítes
- Quantos aptos por andar?
- Possui salão de festas? 0 ou 1
- Possui piscina? 0 ou 1
- ETC...
- Ao todo, 30 características numéricas para cada um dos 1500 imóveis.

Resposta Y e inputs-features X

- Organizar os dados como vetores e matrizes.
- Preços: um vetor Y de dimensão 1500.
- As features: matriz 1500 x 30

Cada linha = um imóvel

1ª. coluna = renda média da região

2ª. coluna = ano da construção

3ª. coluna = área total

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{1499} \\ y_{1500} \end{pmatrix}$$

$$X = \begin{pmatrix} \text{renda}_1 & \text{área}_1 & \cdots & \text{salão}_1 \\ \text{renda}_2 & \text{área}_2 & \cdots & \text{salão}_2 \\ \vdots & \vdots & \vdots & \\ \text{renda}_{1499} & \text{área}_{1499} & \cdots & \text{salão}_{1499} \\ \text{renda}_{1500} & \text{área}_{1500} & \cdots & \text{salão}_{1500} \end{pmatrix}$$

Modelo de regressão linear

A partir das 30 características do imóvel, fornecer uma predição do preço

Forma simples: cada característica contribui com certo peso para o preço

Y é aproximadamente igual a

$$a + b * \text{área} + c * \text{idade} + d * \text{localização} + \text{ETC} \dots$$

O problema é:

como encontrar os valores de a , b , c , etc que tornem a aproximação a melhor possível?

Um modelo simples

Queremos que cada um desses 1500 valores seja aproximadamente igual a uma combinação linear das 30 características (mais a constante a)

$$y_1 \approx a + b * \text{área}_1 + c * \text{idade}_1 + \dots$$

$$y_2 \approx a + b * \text{área}_2 + c * \text{idade}_2 + \dots$$

$$\vdots$$

$$y_{1500} \approx a + b * \text{área}_{1500} + c * \text{idade}_{1500} + \dots$$

Podemos escrever isto de forma matricial.

O problema de forma matemática

Para facilitar a notação, vamos escrever os pesos que multiplicam cada característica como:

b_0 , para a constante

b_1 para área

b_2 para idade, etc

$$y_1 \approx b_0 + b_1 * \text{área}_1 + b_2 * \text{idade}_1 + \dots b_{30} * \text{salão}_1$$

$$y_2 \approx b_0 + b_1 * \text{área}_2 + b_2 * \text{idade}_2 + \dots b_{30} * \text{salão}_2$$

\vdots

$$y_{1500} \approx b_0 + b_1 * \text{área}_{1500} + b_2 * \text{idade}_{1500} + \dots b_{30} * \text{salão}_{1500}$$

O problema de forma matricial

$$y_1 \approx b_0 + b_1 * \text{área}_1 + b_2 * \text{idade}_1 + \dots b_{30} * \text{salão}_1$$

$$y_2 \approx b_0 + b_1 * \text{área}_2 + b_2 * \text{idade}_2 + \dots b_{30} * \text{salão}_2$$

\vdots

$$y_{1500} \approx \underbrace{b_0 + b_1 * \text{área}_{1500} + b_2 * \text{idade}_{1500} + \dots b_{30} * \text{salão}_{1500}}$$

$$b_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} + b_1 \begin{pmatrix} \text{área}_1 \\ \text{área}_2 \\ \vdots \\ \text{área}_{1499} \\ \text{área}_{1500} \end{pmatrix} + b_2 \begin{pmatrix} \text{idade}_1 \\ \text{idade}_2 \\ \vdots \\ \text{idade}_{1499} \\ \text{idade}_{1500} \end{pmatrix} + \dots + b_{30} \begin{pmatrix} \text{salão}_1 \\ \text{salão}_2 \\ \vdots \\ \text{salão}_{1499} \\ \text{salão}_{1500} \end{pmatrix}$$

Forma vetorial

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{1499} \\ y_{1500} \end{pmatrix} \approx b_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} + b_1 \begin{pmatrix} \text{área}_1 \\ \text{área}_2 \\ \vdots \\ \text{área}_{1499} \\ \text{área}_{1500} \end{pmatrix} + b_2 \begin{pmatrix} \text{idade}_1 \\ \text{idade}_2 \\ \vdots \\ \text{idade}_{1499} \\ \text{idade}_{1500} \end{pmatrix} + \dots + b_{30} \begin{pmatrix} \text{salão}_1 \\ \text{salão}_2 \\ \vdots \\ \text{salão}_{1499} \\ \text{salão}_{1500} \end{pmatrix}$$

A matriz X das features

Seja X a matriz 1500 x 31 abaixo (note que ela tem uma coluna composta apenas de 1's):

$$\mathbf{X} = \begin{pmatrix} 1 & \text{renda}_1 & \text{área}_1 & \cdots & \text{salão}_1 \\ 1 & \text{renda}_2 & \text{área}_2 & \cdots & \text{salão}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \text{renda}_{1499} & \text{área}_{1499} & \cdots & \text{salão}_{1499} \\ 1 & \text{renda}_{1500} & \text{área}_{1500} & \cdots & \text{salão}_{1500} \end{pmatrix}$$

Vetores próximos

- Nosso problema é encontrar os coeficientes b_0, b_1, \dots, b_{30} tais que

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{1499} \\ y_{1500} \end{pmatrix} \approx b_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} + b_1 \begin{pmatrix} \text{área}_1 \\ \text{área}_2 \\ \vdots \\ \text{área}_{1499} \\ \text{área}_{1500} \end{pmatrix} + b_2 \begin{pmatrix} \text{idade}_1 \\ \text{idade}_2 \\ \vdots \\ \text{idade}_{1499} \\ \text{idade}_{1500} \end{pmatrix} + \dots + b_{30} \begin{pmatrix} \text{salão}_1 \\ \text{salão}_2 \\ \vdots \\ \text{salão}_{1499} \\ \text{salão}_{1500} \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{1498} \\ y_{1499} \\ y_{1500} \end{pmatrix} \approx \begin{pmatrix} 1 & \text{renda}_1 & \text{área}_1 & \dots & \text{salão}_1 \\ 1 & \text{renda}_2 & \text{área}_2 & \dots & \text{salão}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \text{renda}_{1499} & \text{área}_{1499} & \dots & \text{salão}_{1499} \\ 1 & \text{renda}_{1500} & \text{área}_{1500} & \dots & \text{salão}_{1500} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{30} \end{pmatrix} = \mathbf{X} \mathbf{b}$$

Solução

X é uma matriz 1500 x 3. **Y** e **Xb** são vetores 1500-dim.
Além disso, **Xb** é uma combinação linear das colunas da matriz **X**.
Cada coluna é um vetor 1500-dim

Queremos encontrar **b** tal que o vetor **Xb** seja o mais próximo possível do vetor **Y**.

Queremos $\hat{b} = \arg \min_b ||\mathbf{Y} - \mathbf{Xb}||^2$

Solução: $\mathbf{X}\hat{\mathbf{b}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$

Uma roupagem probabilística

- Suponha que resposta Y siga uma distribuição gaussiana (normal)
- O valor esperado do preço Y é uma função das features do apto.
- Modelo: $Y_i \sim N(\mu(\mathbf{x}_i), \sigma^2)$

$$\mu(\mathbf{x}_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_{30} x_{i,30}$$

- Os preços são variáveis aleatórias independentes.
- Maximizar a log-verossimilhança é EQUIVALENTE a minimizar a distância ao quadrado entre os vetores $\|\mathbf{Y} - \mathbf{X}\mathbf{b}\|^2$

Função custo e gradiente descendente

Maximizar a log-verossimilhança ou minimizar o custo


- Em machine learning, é mais comum falar em minimizar uma função custo.
- A função custo é o NEGATIVO da log-verossimilhança

$$J(w_0, w_1, \dots, w_n) = -\ell(w_0, w_1, \dots, w_n) = -\log L(w_0, w_1, \dots, w_n)$$

- Usamos o método de Newton como antes.
- Newton acha mínimos e máximos.
- Nossos modelos de redes neurais terão muitos e muitos parâmetros.
- O cálculo do Hessiano (matriz de derivadas parciais de 2a ordem) será proibitivo.
- Vamos adotar outro método, mais simples e talvez com convergência mais lenta.

Esqueça o hessiano, use apenas o gradiente

- Ao invés de usar

$$\mathbf{w}^{k+1} = \begin{bmatrix} w_0^{k+1} \\ w_1^{k+1} \\ \vdots \\ w_n^{k+1} \end{bmatrix} = \begin{bmatrix} w_0^k \\ w_1^k \\ \vdots \\ w_n^k \end{bmatrix} - \left[\underbrace{J^2(\mathbf{w}^k)}_{\text{matriz derivadas parciais 2a ordem de J}} \right]^{-1} \underbrace{\nabla J(\mathbf{w}^k)}_{\text{vetor gradiente de J}}$$


- usamos

$$\mathbf{w}^{k+1} = \begin{bmatrix} w_0^{k+1} \\ w_1^{k+1} \\ \vdots \\ w_n^{k+1} \end{bmatrix} = \begin{bmatrix} w_0^k \\ w_1^k \\ \vdots \\ w_n^k \end{bmatrix} - \alpha \underbrace{\nabla J(\mathbf{w}^k)}_{\text{vetor gradiente de J}}$$

α é um escalar positivo e pequeno.

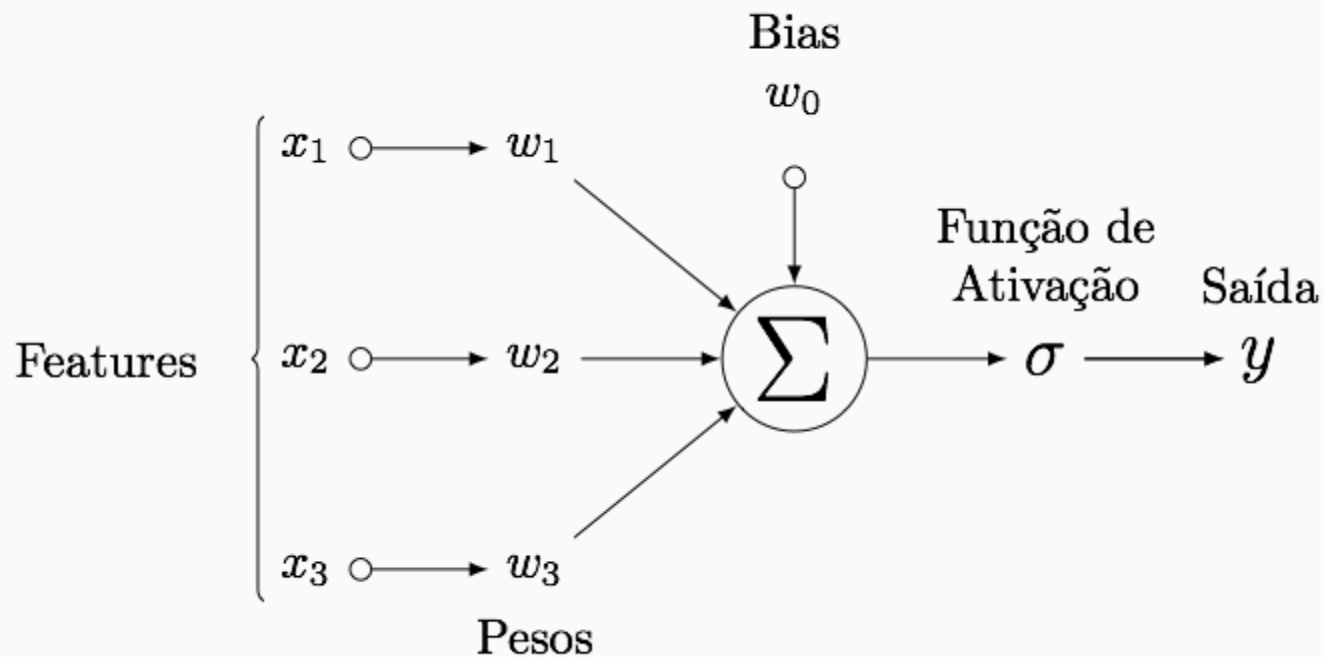
Por exemplo, é comum usar $\alpha = 0.01$

Resumo do problema supervisionado

- Amostra de m exemplos de dados:
 - Uma resposta Y
 - Inputs-features num vetor x
- Objetivo: representar o valor esperado de Y através de uma função dos inputs
 - Dado x , quanto é $\mathbb{E}(Y|x)$?
 - No caso binário: $\mathbb{E}(Y|x) = \mathbb{P}(Y = 1|x)$
 - No caso multi-classe: $\mathbb{P}(Y = k|x)$
 - No caso contínuo: $\mathbb{E}(Y|x) = \mu(x)$
- Esta função de x depende de parâmetros-pesos $\mathbf{w} = (w_0, w_1, \dots, w_n)$
- Obtenha a log-verossimilhança dos pesos: a probabilidade de gerar os dados da amostra para cada possível valor dos pesos.
- Função de custo: $J(w_0, w_1, \dots, w_n) = -\text{Log-verossimilhança} \rightarrow$ Minimize a função de custo com Newton

Redes Neurais

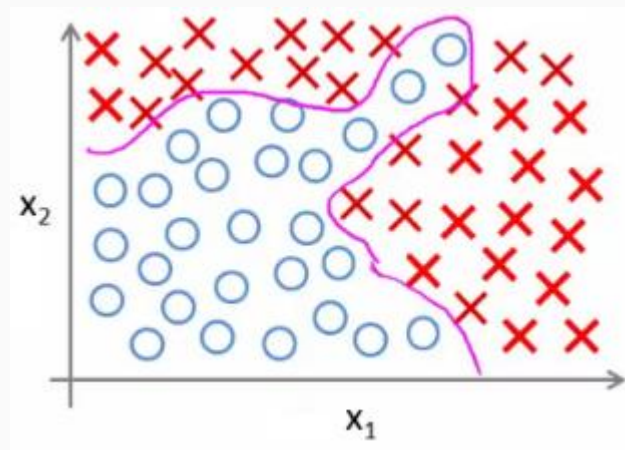
De volta a reg logística como rede neural



Por que usar redes neurais?

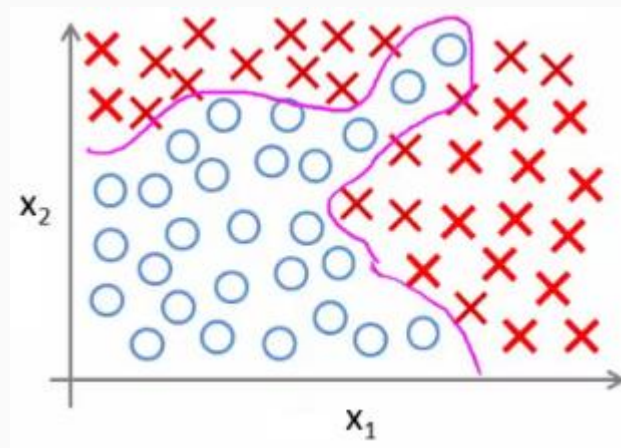
- Para aprender uma decision boundary não-linear com regressão logística → precisamos de muitos termos não lineares das features "básicas"
- Por exemplo, com duas features x_1 e x_2 , podemos buscar os pesos w com

$$\mathbb{P}(Y = 1|x_1, x_2) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_6 x_1^3 + w_7 x_1^2 x_2 + w_8 x_1 x_2^2)}}$$

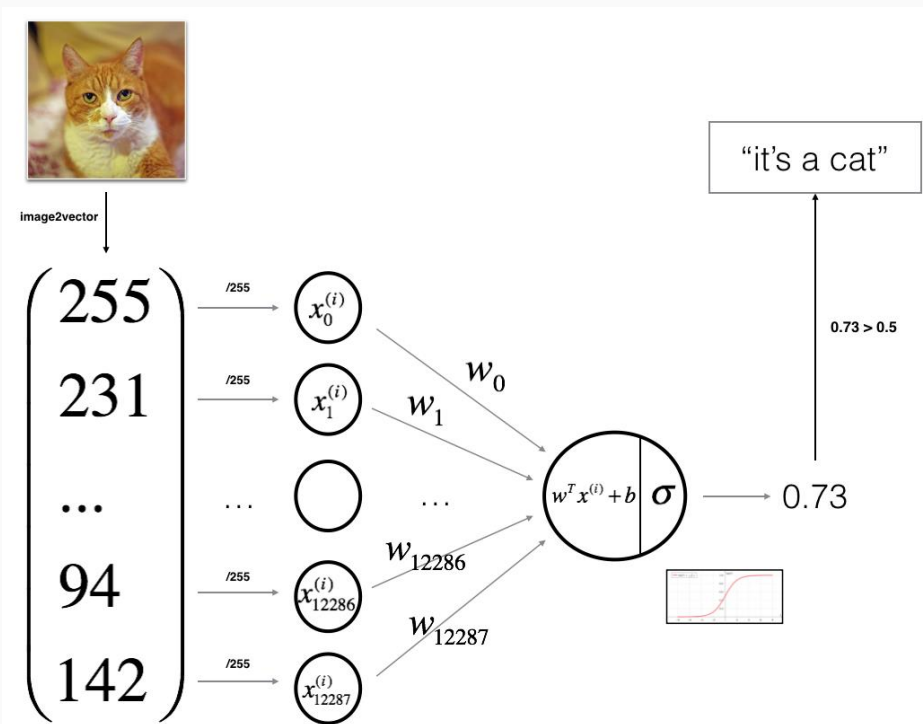


Por que usar redes neurais?

- Mas se temos muitas features no vetor x ...
- Para ter uma função flexível, vamos usar termos polinomiais de x
- Não vai funcionar bem se temos muitas features
- Exemplo: crianças, 10 features básicas+ 10 features $**2$ + 10 features $**3$ + 45 pares de features (produtos) + 120 triplas = 195 features
- Vai precisar de amostra com $\gg 195$ crianças

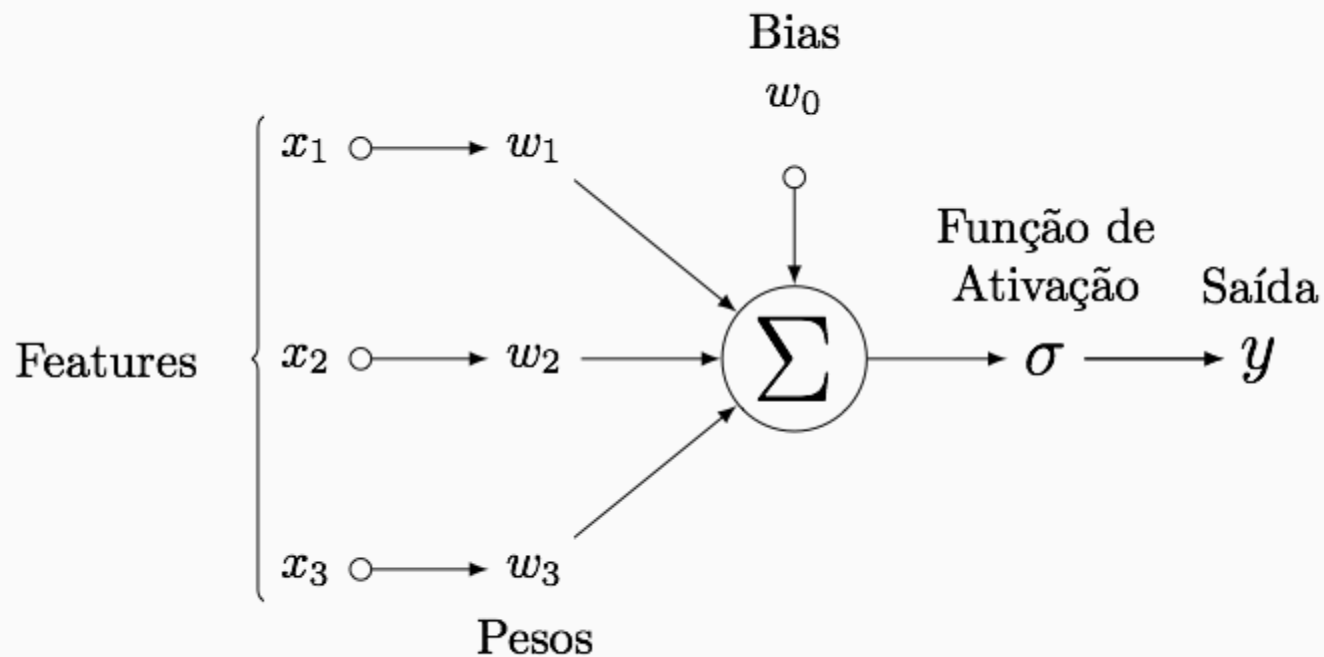


Logística para imagem?

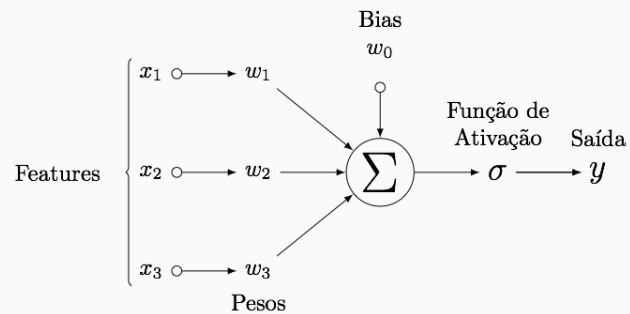


- Espaço de features
- 50 x 50 pixels em tons de cinza → 2500 pixels (ou features)
- Se RGB, então 7500 features
- Acrescentando os termos quadráticos → 3 milhões de features
- Regressão logística simples não é apropriada para modelos complexos
- Redes neurais são muito melhores para modelos não-lineares, mesmo quando o espaço de features é enorme

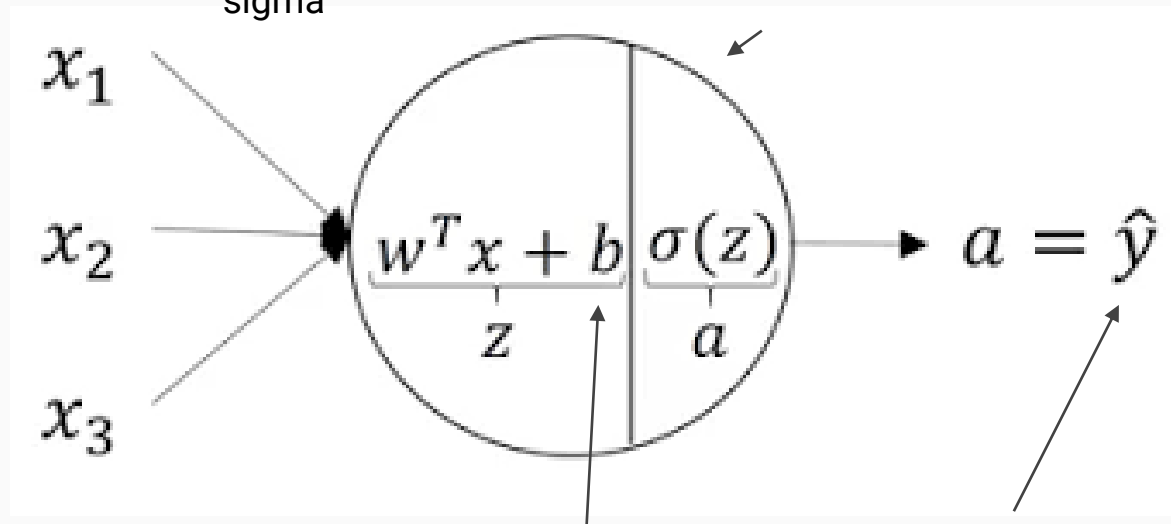
Introdução à terminologia e notação



Notação não é uniforme



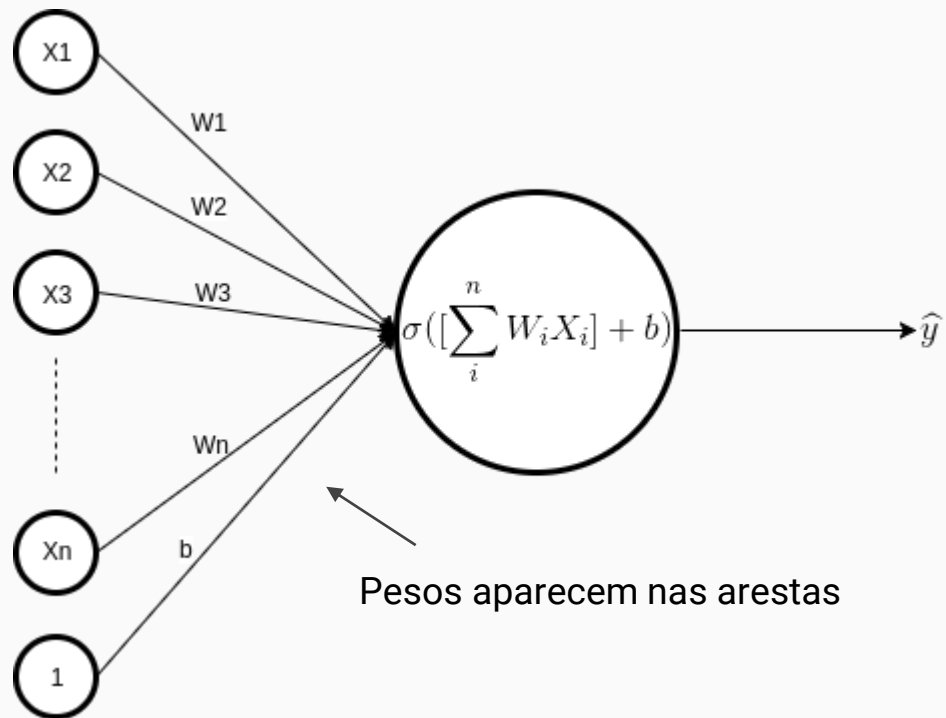
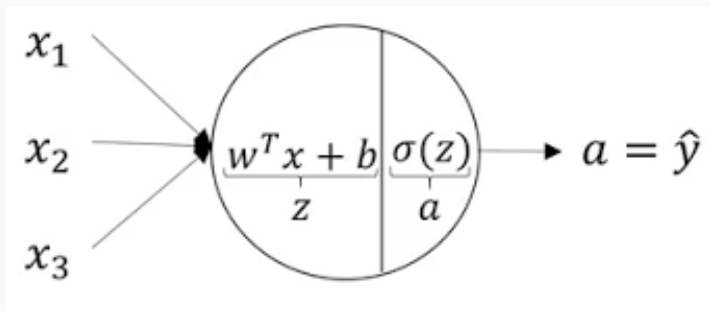
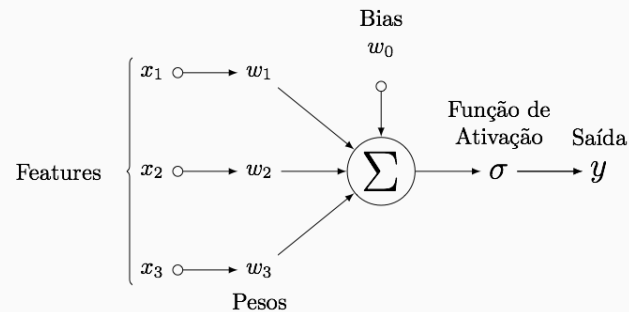
Representação dos dois passos no neurônio: a combinação linear das features e a ativação com sigma



Termo de bias é b ao invés de w_0

Valor ativado é $a = \hat{y}$

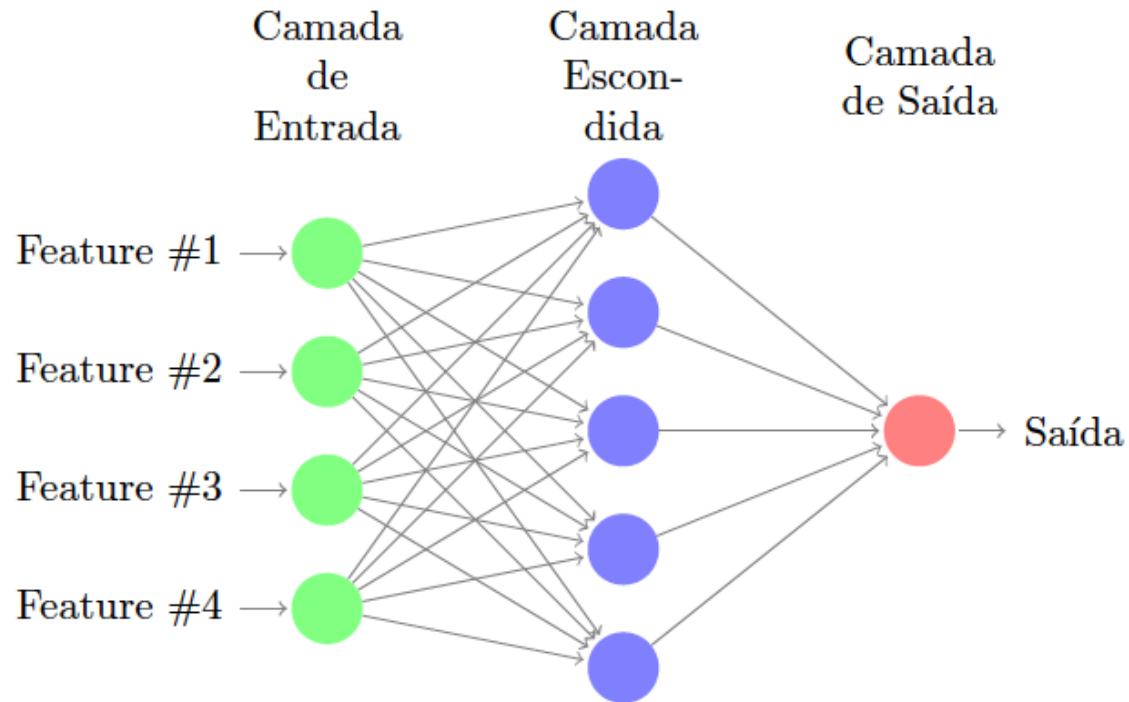
Notação varia entre autores



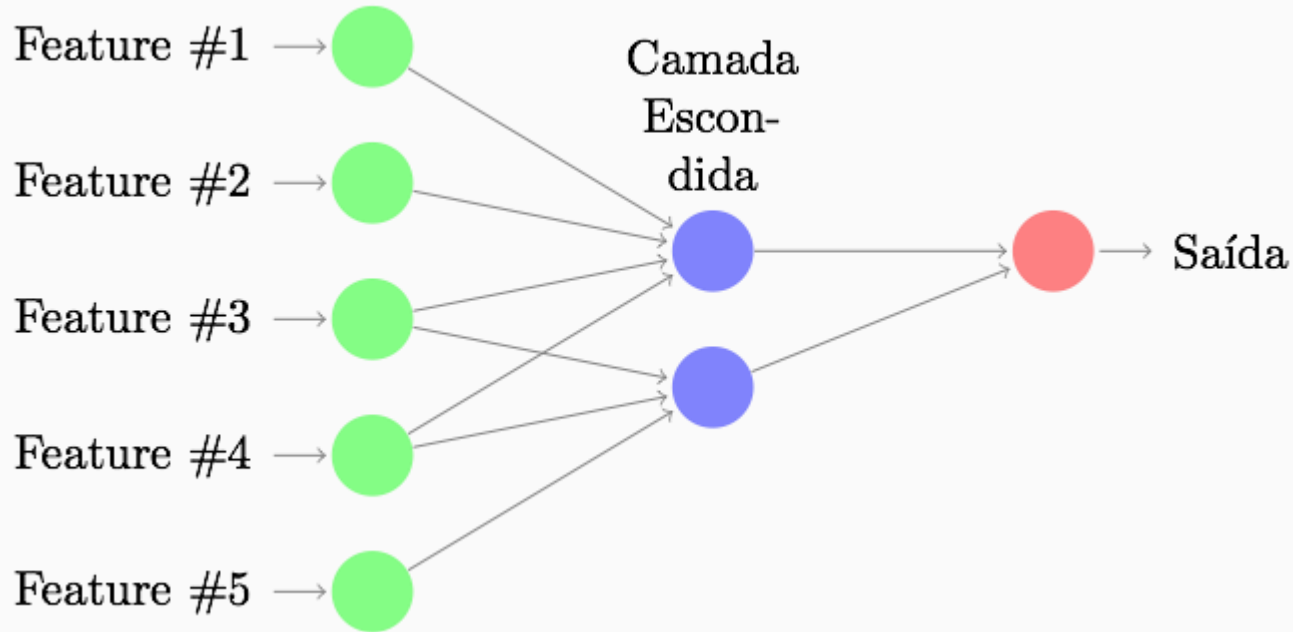
Redes neurais: interpondo camadas

- Este modelo é muito simples.
- Não vamos usar os inputs diretamente para modelar a probabilidade de sucesso
- Os inputs são agregados em fatores/dimensões (as unidades escondidas).
- Cada fator/dimensão mistura os inputs de forma linear num primeiro passo
 - Esta mistura usa pesos para cada input
 - Diferentes fatores escondidos usam pesos diferentes
- Num segundo passo, os fatores latentes são submetidos a uma função de ativação, tal como a logística para gerar NOVOS inputs (inputs transformados).
- Estes inputs transformados são então combinados em mais uma função de ativação (pode ser a logística de novo) para predizer a resposta: $\mathbb{E}(Y|\mathbf{x})$

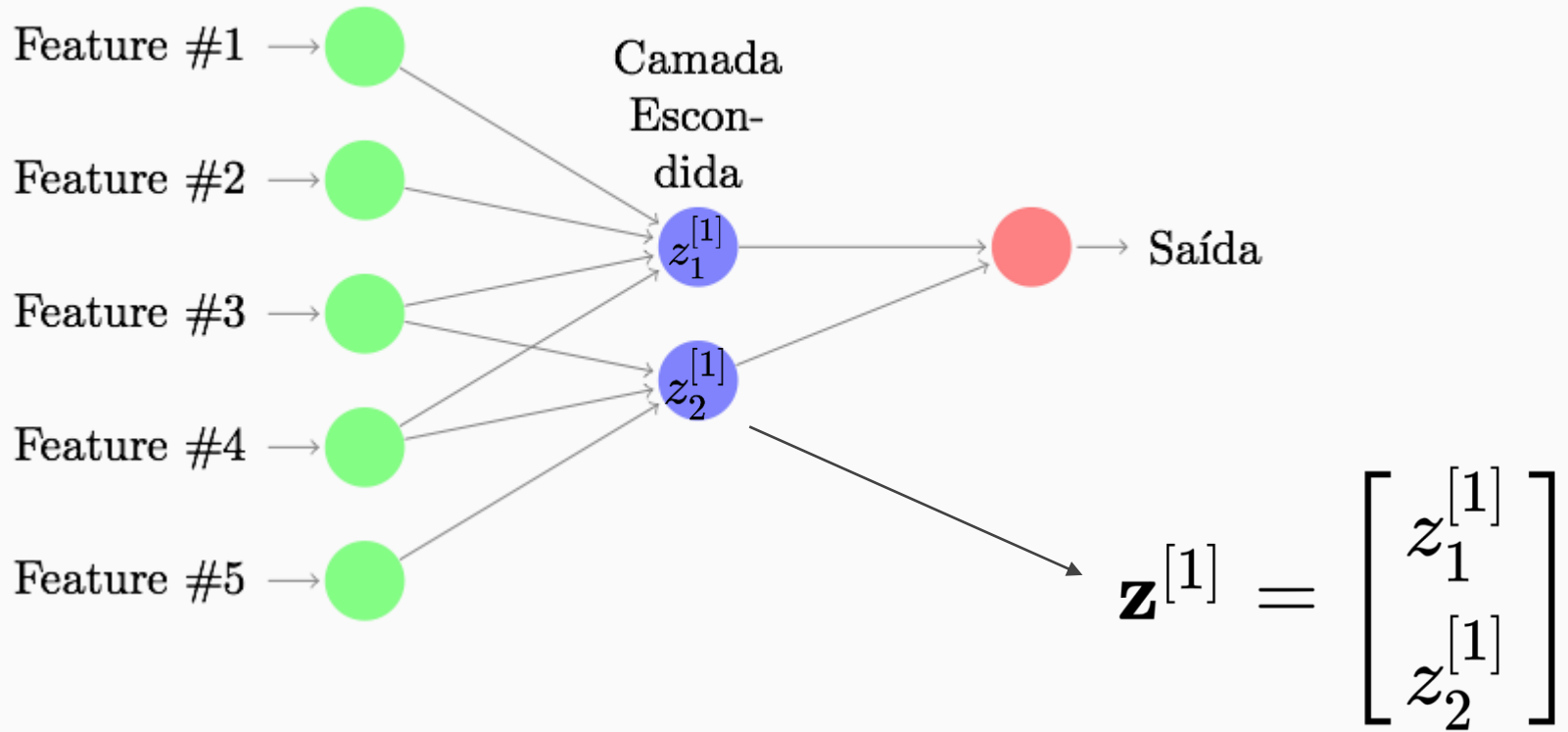
Sobrepondo uma camada escondida



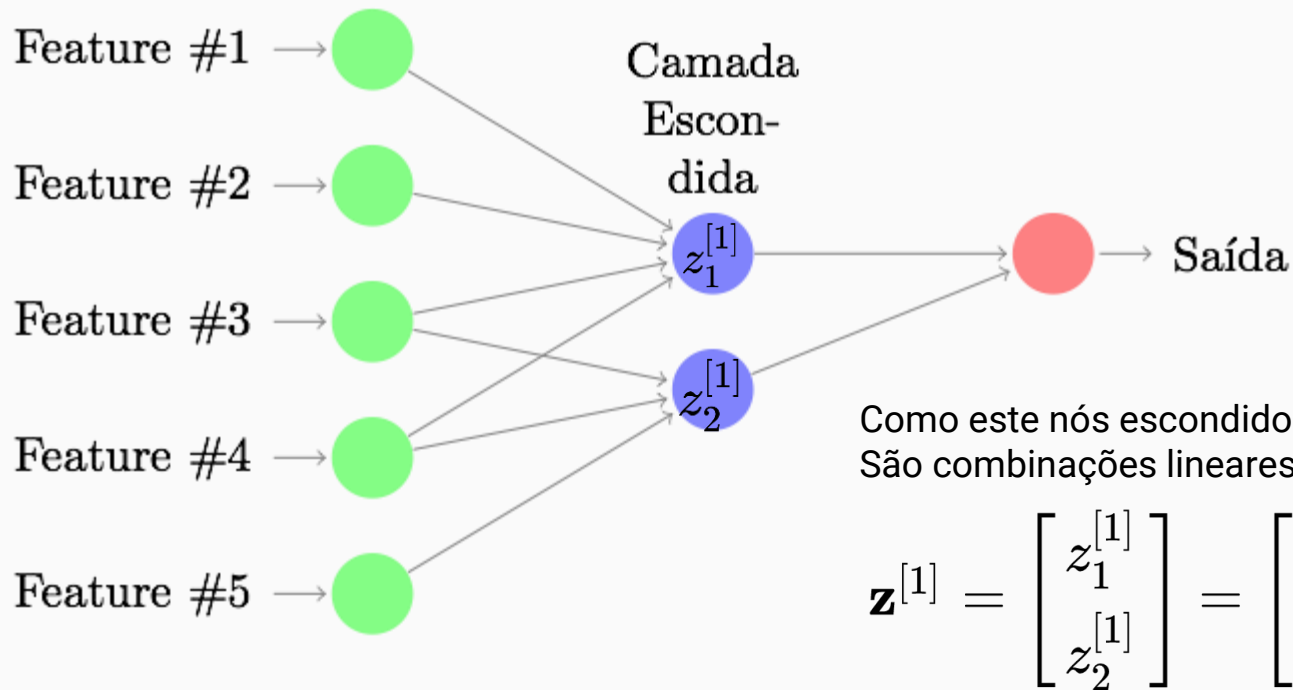
Uma rede com menos neurônios (mais esparsa)



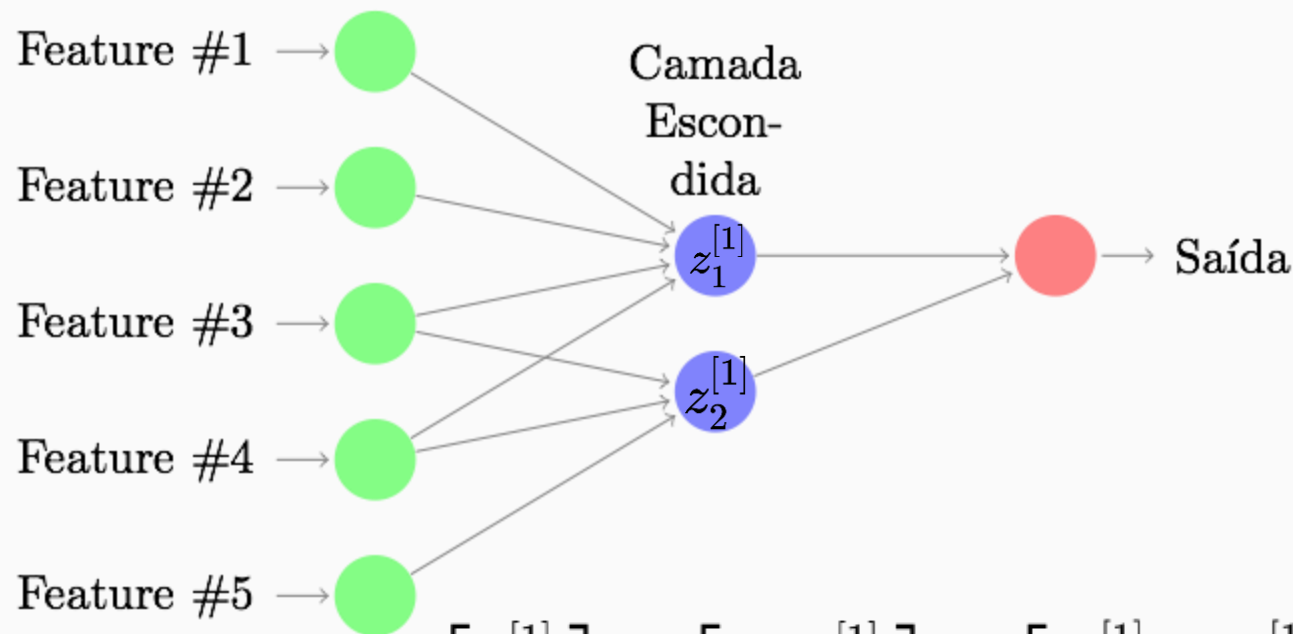
Uma rede com menos neurônios (mais esparsa)



Uma rede com menos neurônios (mais esparsa)

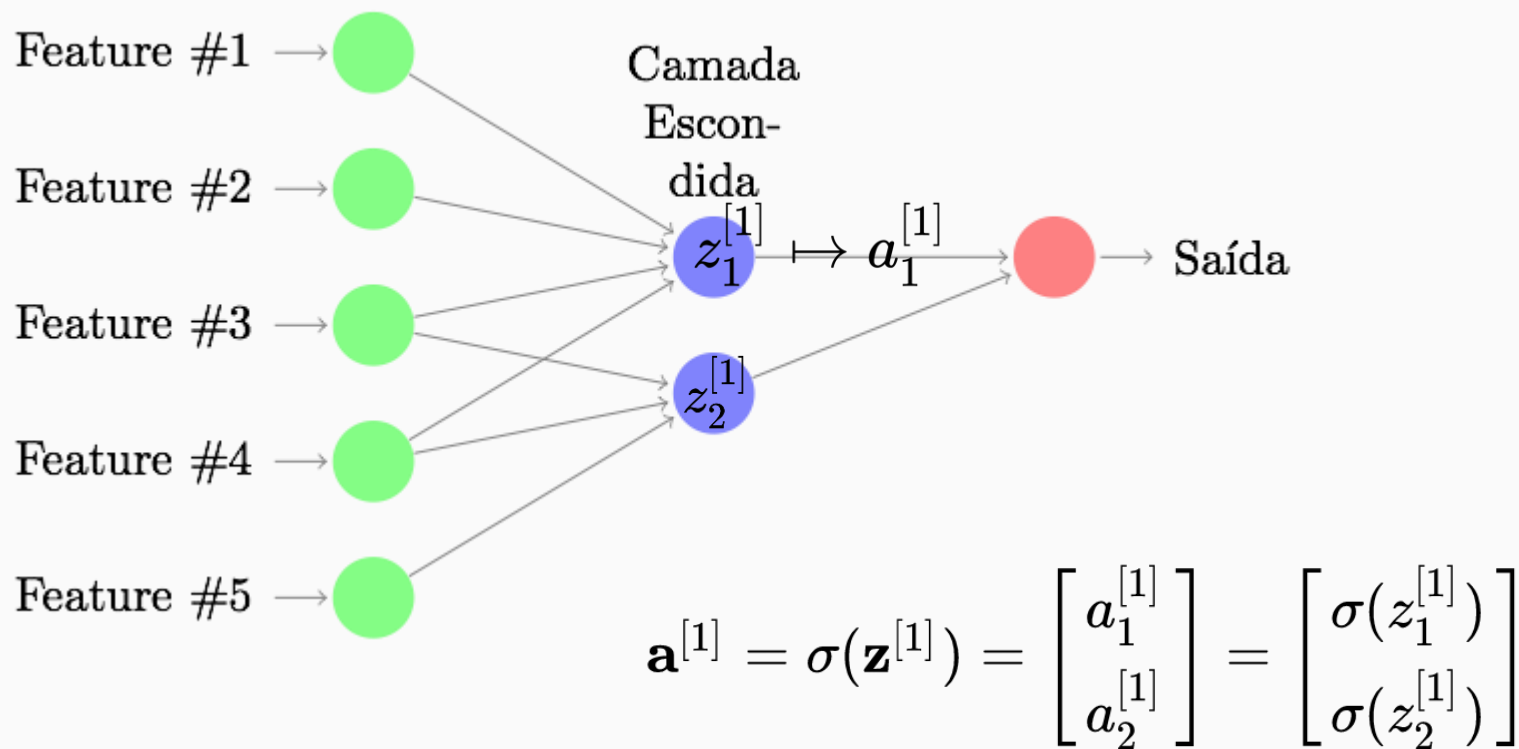


Uma rede com menos neurônios (mais esparsa)

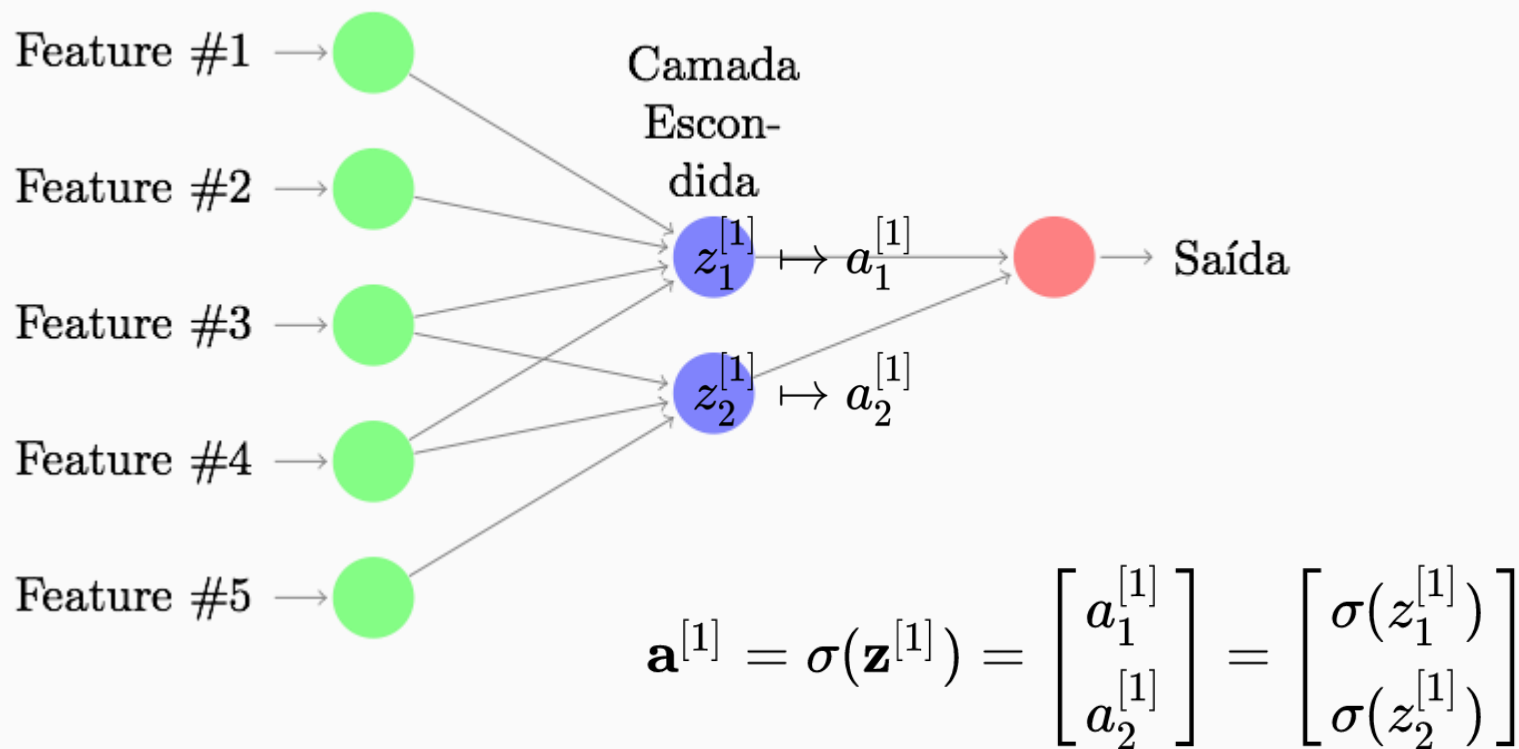


$$\mathbf{z}^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \mathbf{w}_1^{[1]} \\ \mathbf{x}' \mathbf{w}_2^{[1]} \end{bmatrix} = \begin{bmatrix} w_{01}^{[1]} + w_{11}^{[1]}x_1 + \dots + w_{n1}^{[1]}x_n \\ w_{02}^{[1]} + w_{12}^{[1]}x_1 + \dots + w_{n2}^{[1]}x_n \end{bmatrix}$$

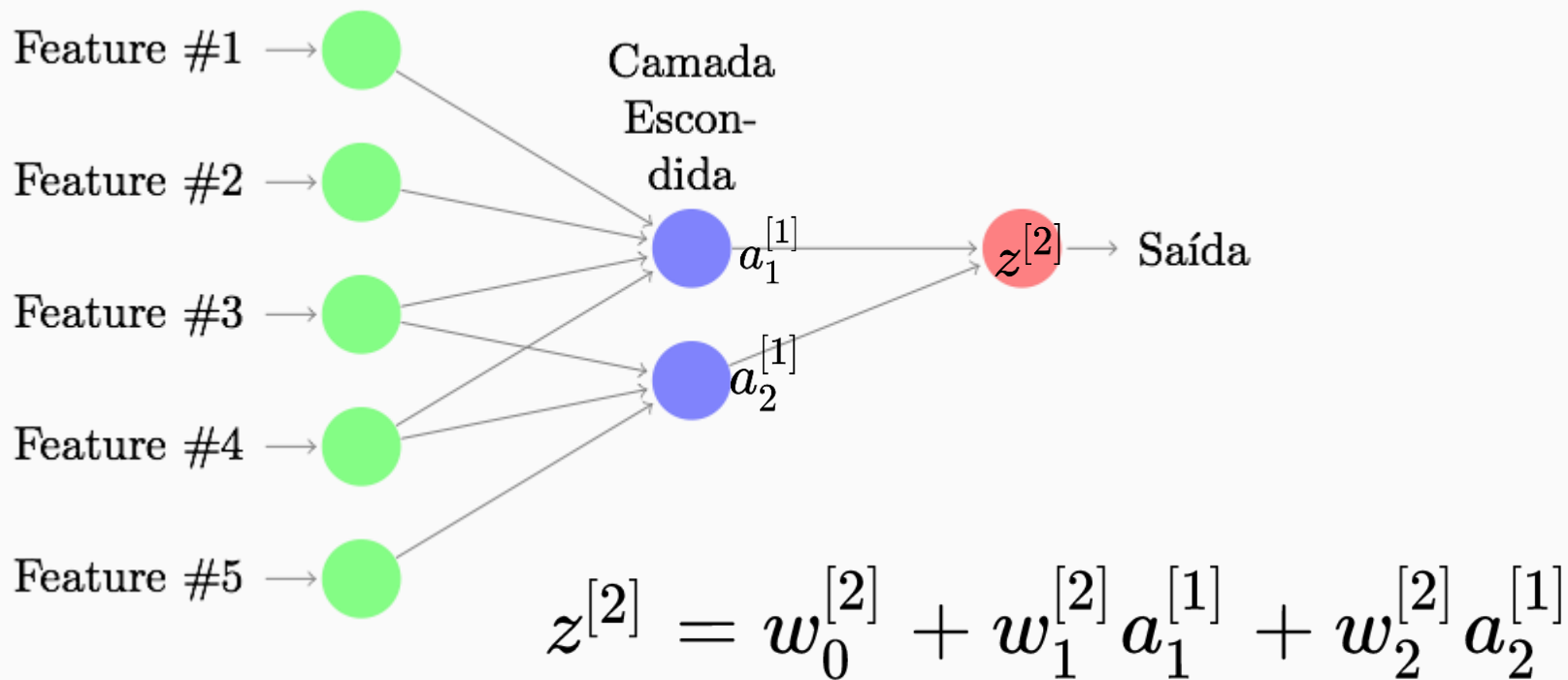
Uma rede com menos neurônios (mais esparsa)



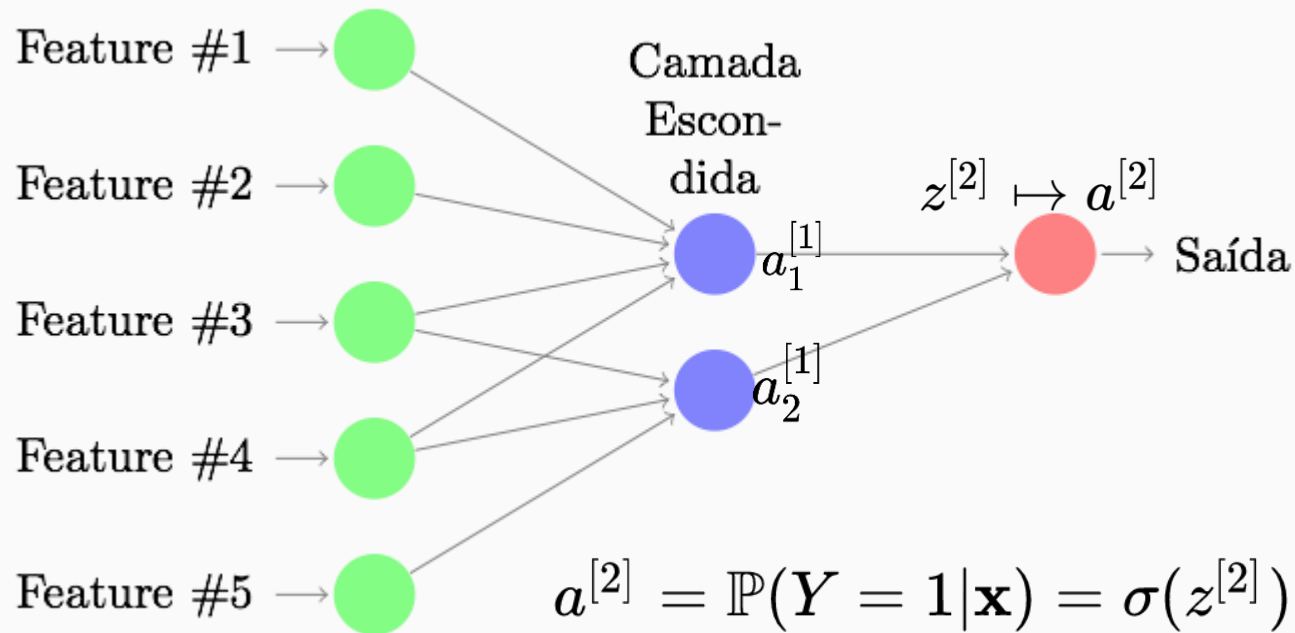
Uma rede com menos neurônios (mais esparsa)



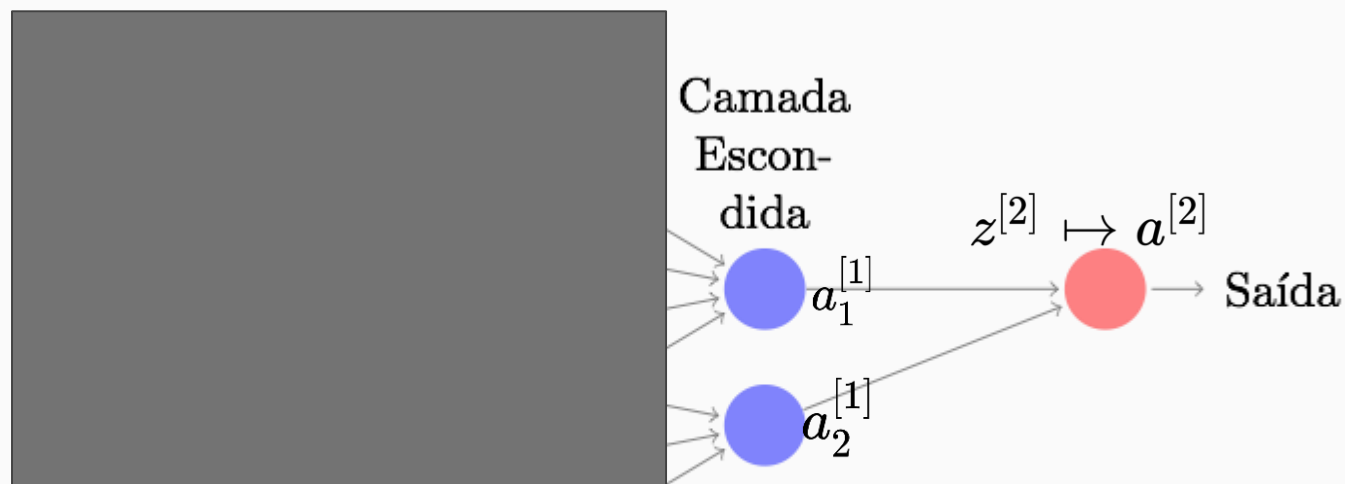
Crie um preditor linear baseado nos a's



A saída binária é uma logística com os a's como features



A saída binária é uma logística com os a's como features



$$z^{[2]} = w_0^{[2]} + w_1^{[2]} a_1^{[1]} + w_2^{[2]} a_2^{[1]}$$
$$a^{[2]} = \mathbb{P}(Y = 1 | \mathbf{x}) = \sigma(z^{[2]}) = \frac{1}{1 + e^{-z^{[2]}}}$$

Exemplos

- Preços de imóveis
- Imagine que todos os fatores relevantes podem ser agrupados em três índices:
 - Um deles agrega as informações sobre as características físicas do imóvel:
 - Tamanho, ano de construção, número de banheiros, vagas de garagem
 - Outro agrega informações sobre a vizinhança onde o imóvel está localizado
 - Um terceiro captura a presença ou não de uma vista exterior agradável
- Ao invés de prever o preço com base nas muitas features individuais, use os 3 índices para fazer isto.
- É como se os os três índices fossem as features de um modelo de regressão

Outra possibilidade para preços de imóveis

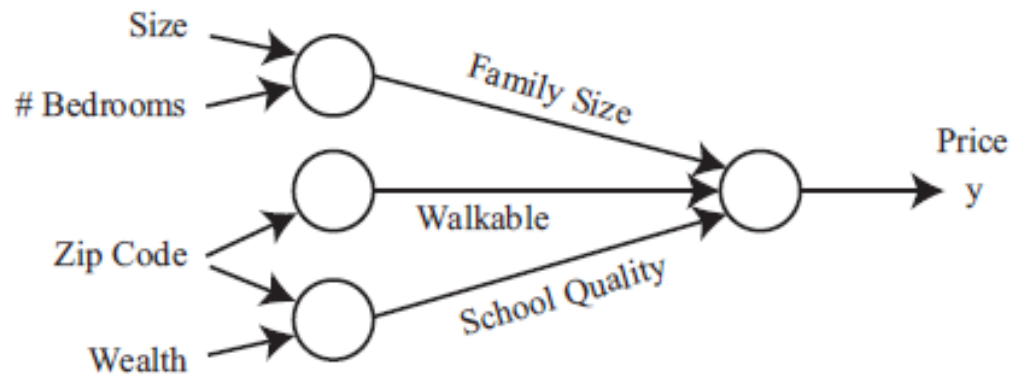


Figure 2: Diagram of a small neural network for predicting housing prices.

De curso de Andrew Ng

Exemplo

- Sucesso ou não num exame (resposta binária)
- Exame é composto de vários provas-testes de conhecimento
- Input é um vetor 15-dim com as notas de 15 testes:
 - X1 = matemática
 - X2 = física
 - X3 = química
 - X4 = gramática
 - X5 = redação
 - X6 = interpretação de texto
 - X7 = história
 - X8 = geografia
 - X9 = sociologia
 - X10 = filosofia
 - X11 = lógica
 - X12 = educação física
 - X13 = biologia
 - X14 = inglês
 - X15 = geometria

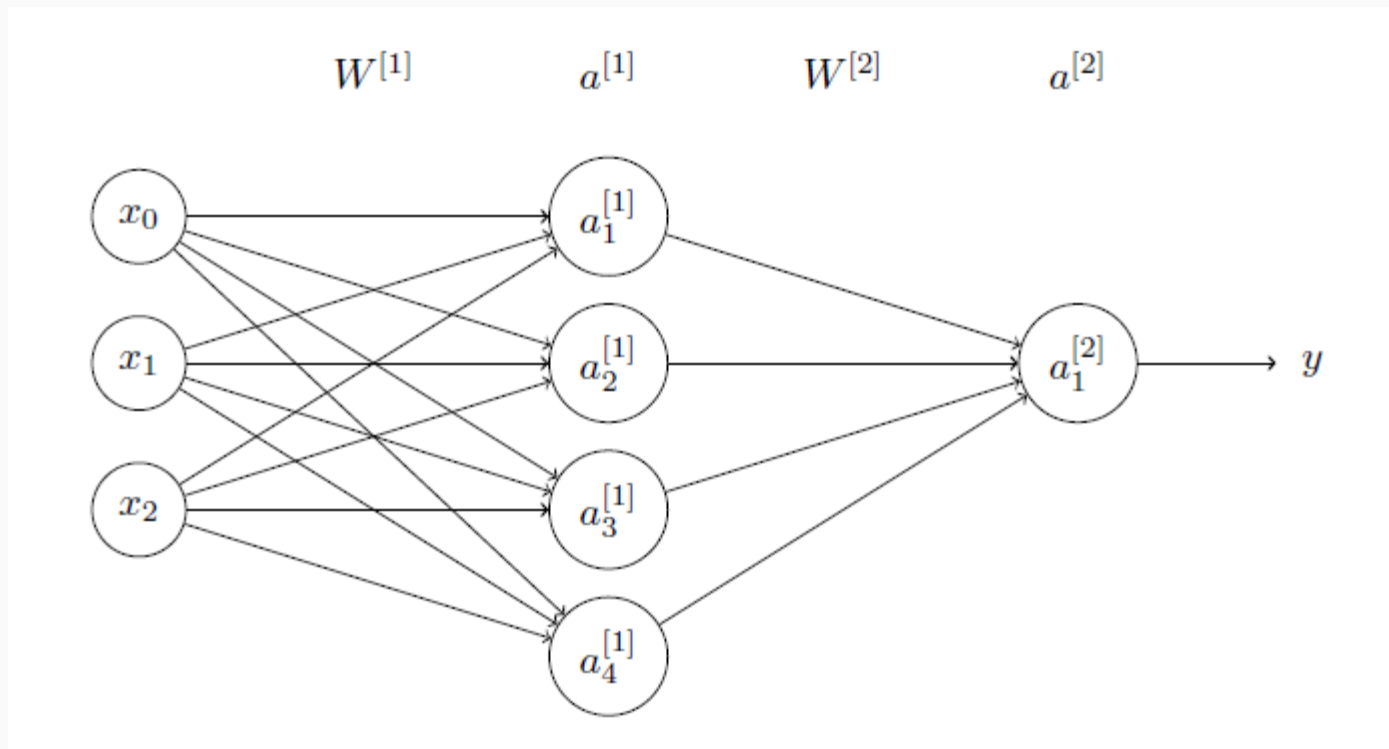
Outro exemplo como motivação

- Resultado binário final Y segue um modelo logístico:

$$\mathbb{P}(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(w_0 + w_1 FQ + w_2 FV)}}$$

- FQ = fator quantitativo
 - Crie um índice = uma soma ponderada das 15 notas, com pesos maiores nas notas de ciências exatas + gramática + lógica...
 - Transforme este índice com uma logística (efeito de saturação do índice: depois de certo valor alto, ele não acrescenta muito)
 - FV = fator verbal, índice baseado numa soma ponderadas das 15 notas, pesos bem diferentes do FQ

Notação completa com o modelo inicial básico:



Notação

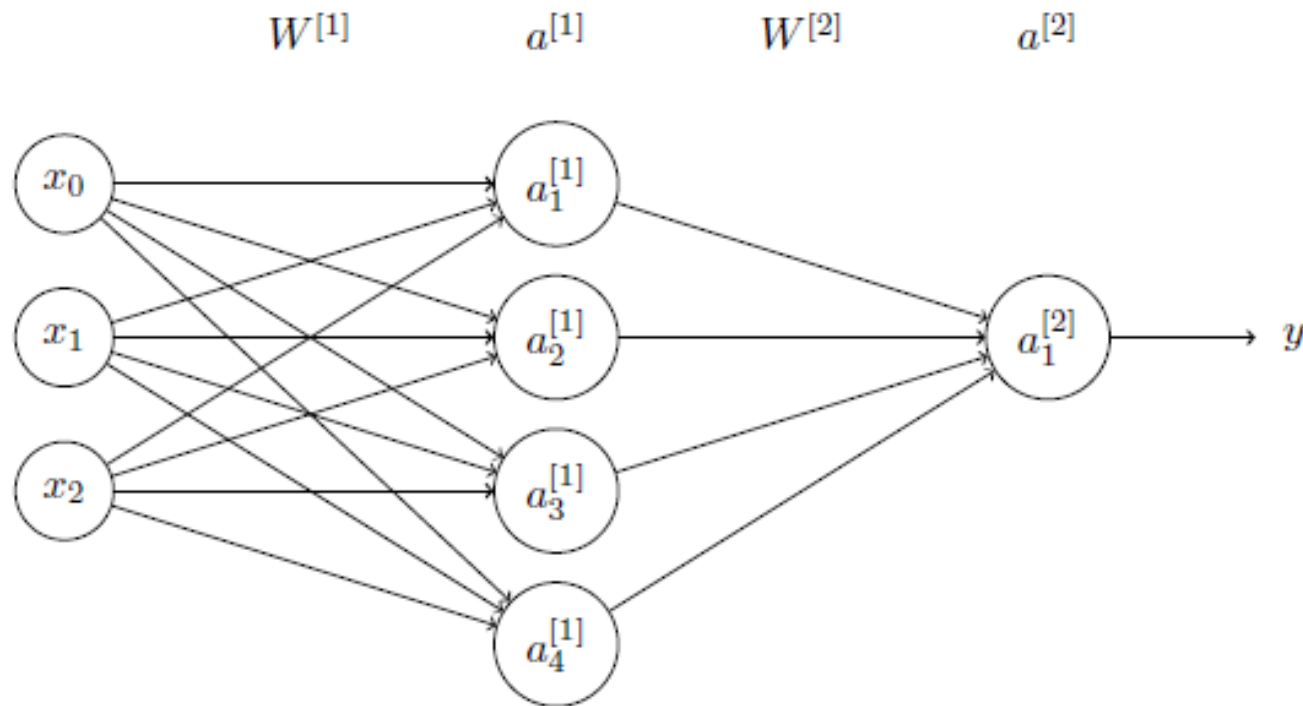
Uma camada escondida com 4 unidades (neurônios) e 3 features de entrada

Primeiro passo a frente:

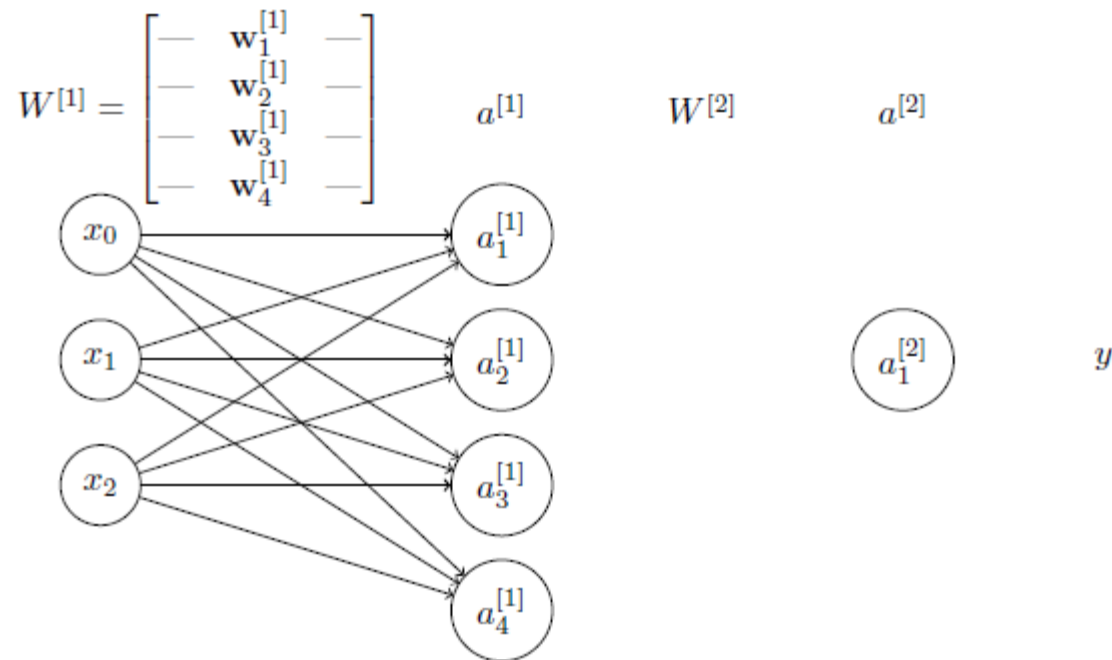
$$\begin{array}{ccccccc} z_1^{[1]} & = & \mathbf{w}_1^{[1]T} \mathbf{X} & \rightarrow & a_1^{[1]} & = & \sigma(z_1^{[1]}) \\ \vdots & & & & & & \vdots \\ z_4^{[1]} & = & \mathbf{w}_4^{[1]T} \mathbf{X} & \rightarrow & a_4^{[1]} & = & \sigma(z_4^{[1]}) \end{array}$$

Notação para o termo de bias: duas notações

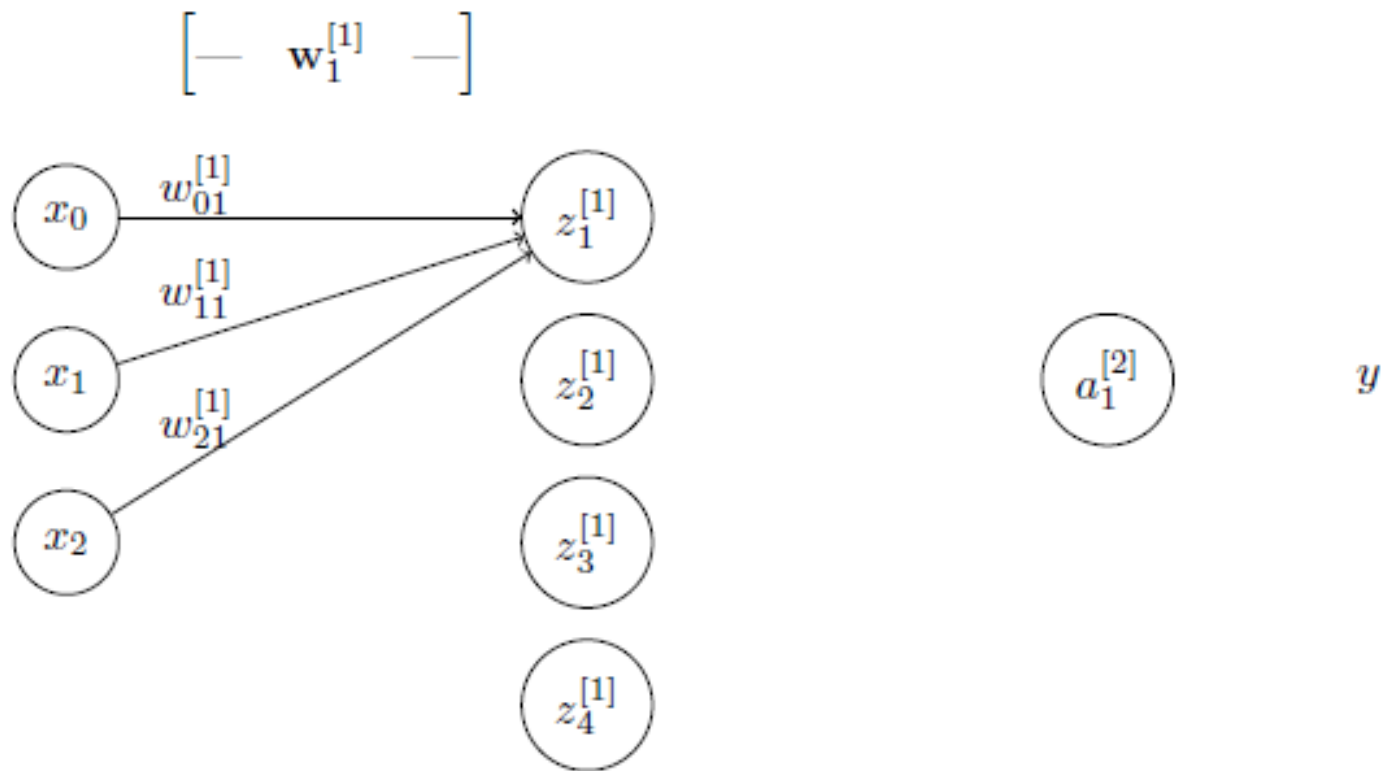
Notação e estrutura da rede neural com uma camada



Do input para a primeira camada escondida

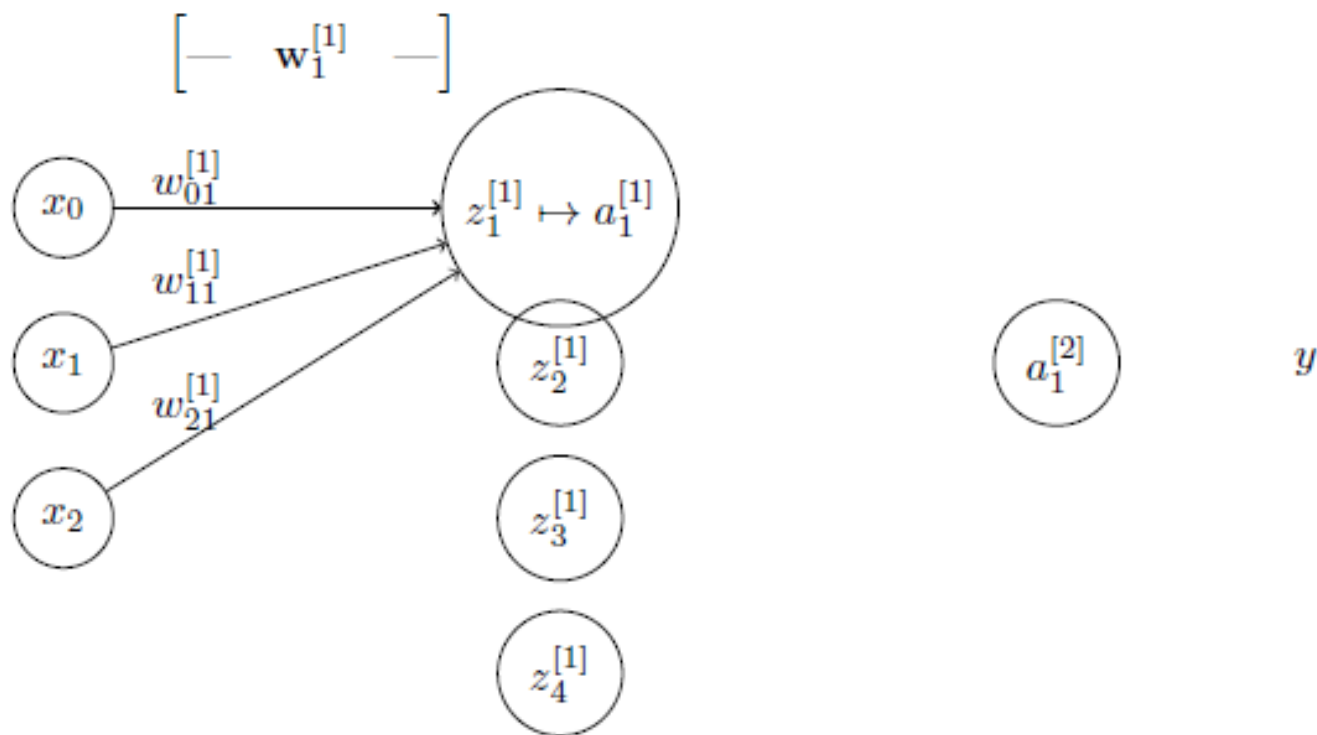


Apenas o 1o neurônio, primeiro passo

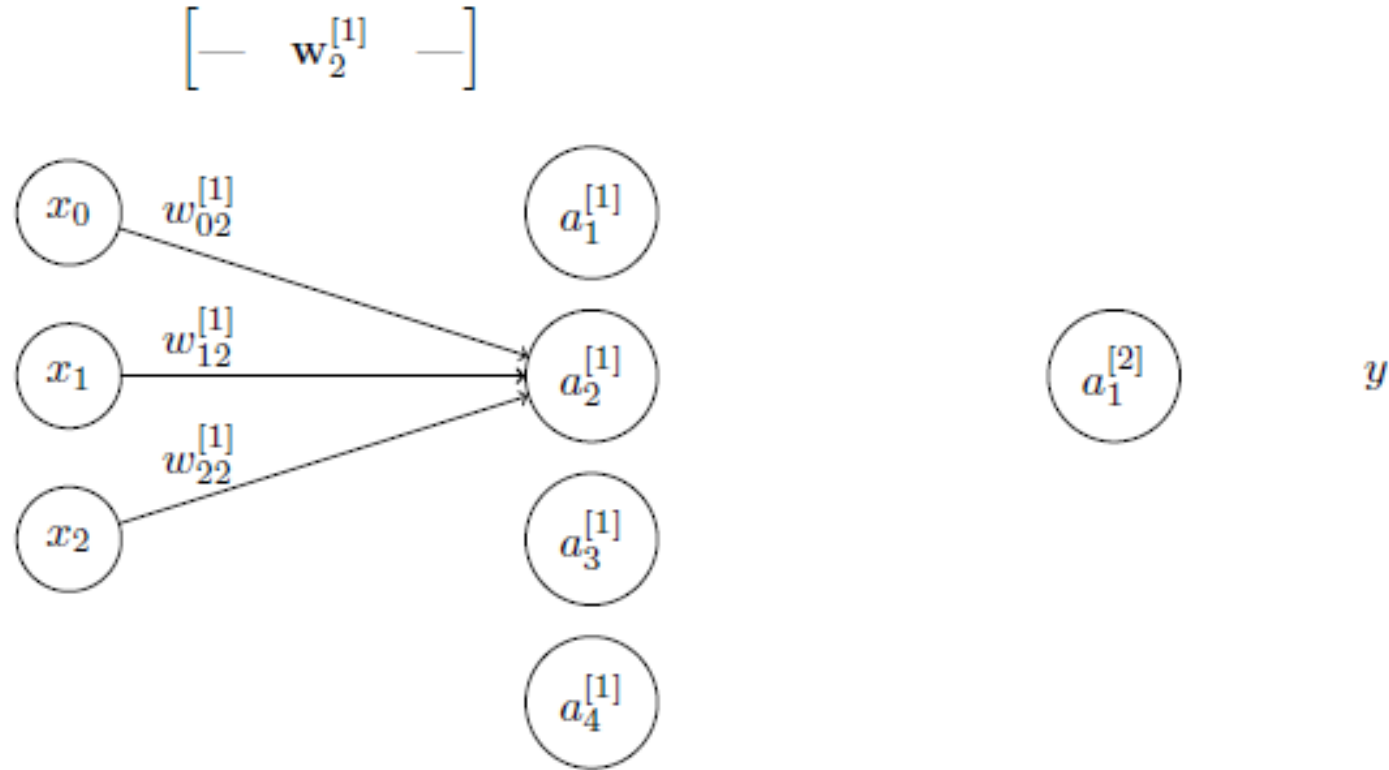


Apenas o 1o neurônio, segundo passo

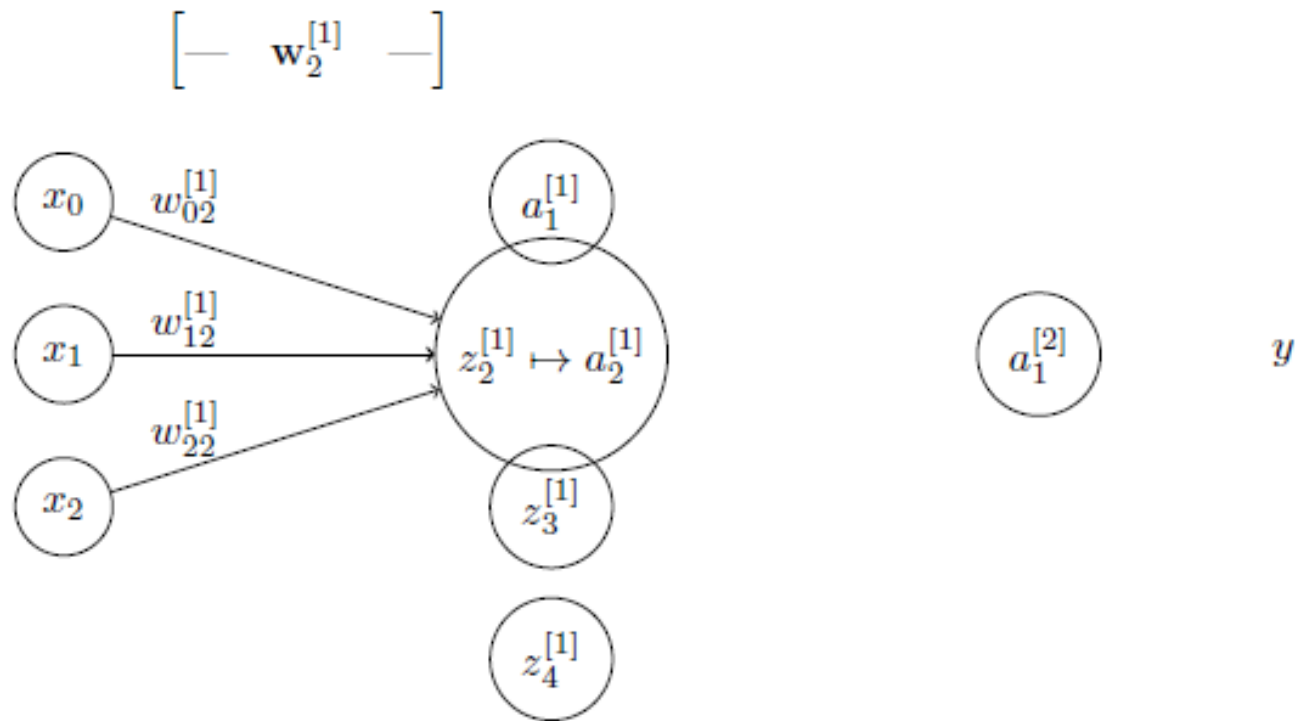
Primeiro neurônio: de z para a



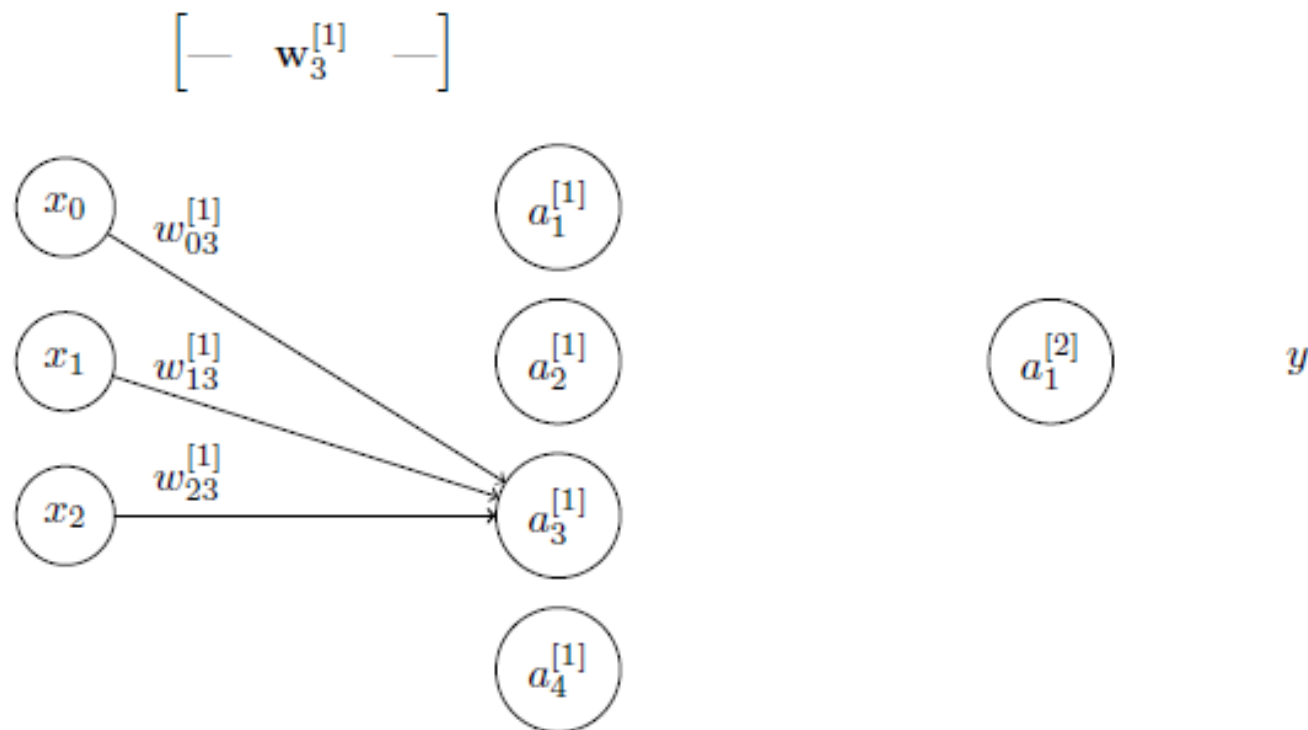
Apenas o 2o neurônio, primeiro passo



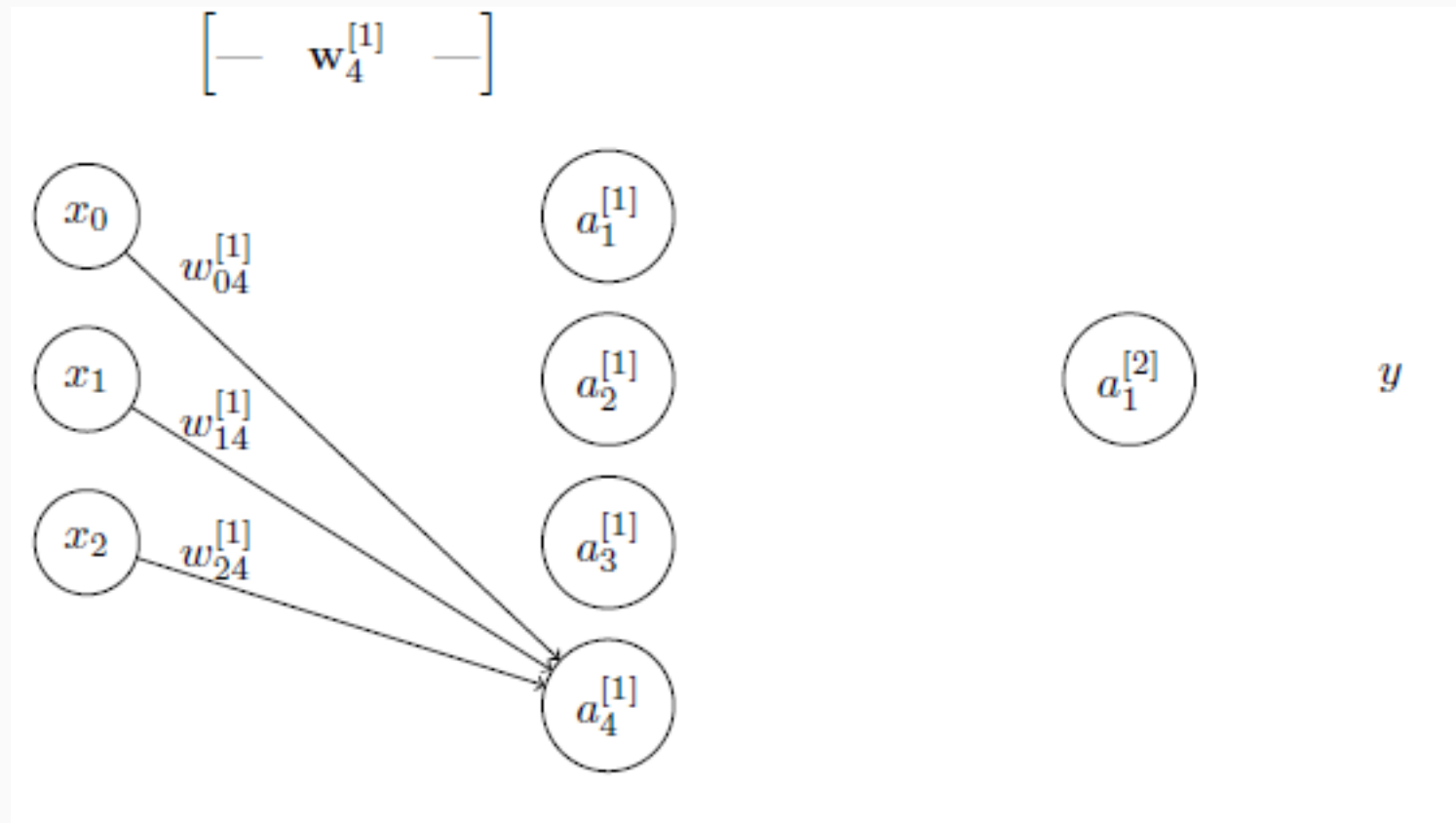
Apenas o 2o neurônio, segundo passo



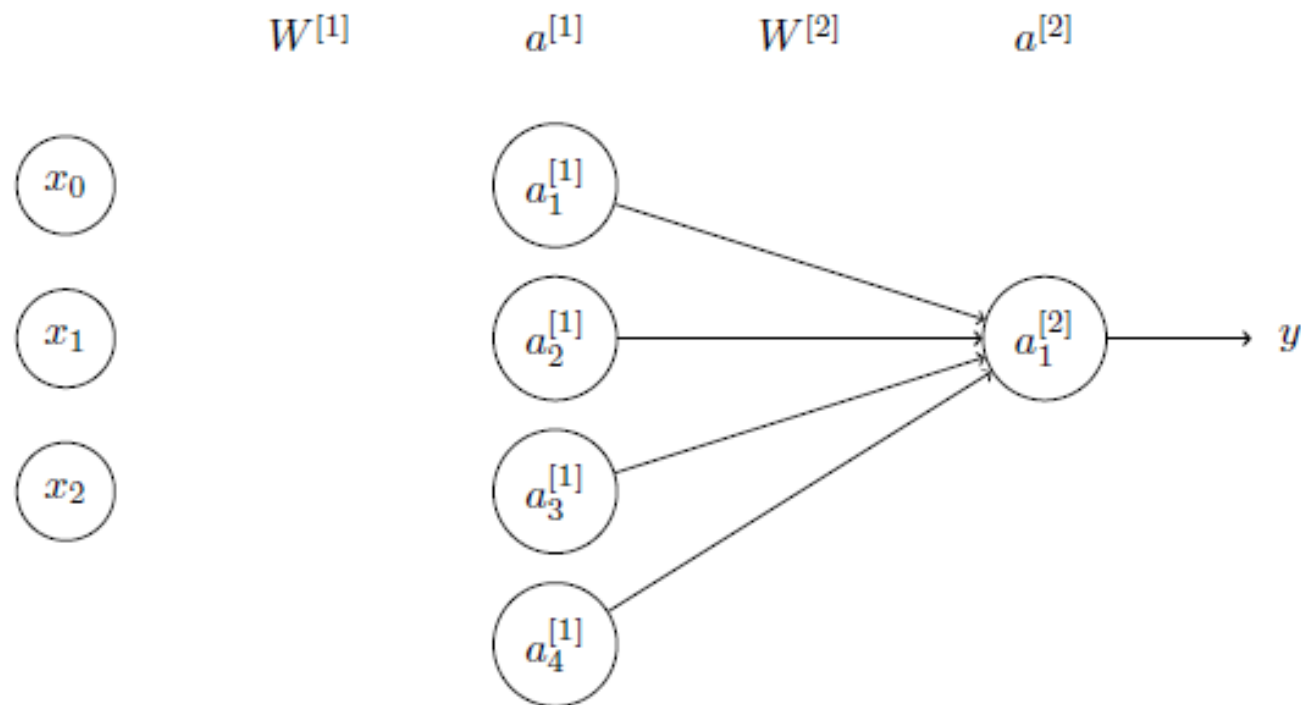
Apenas o 3o neurônio



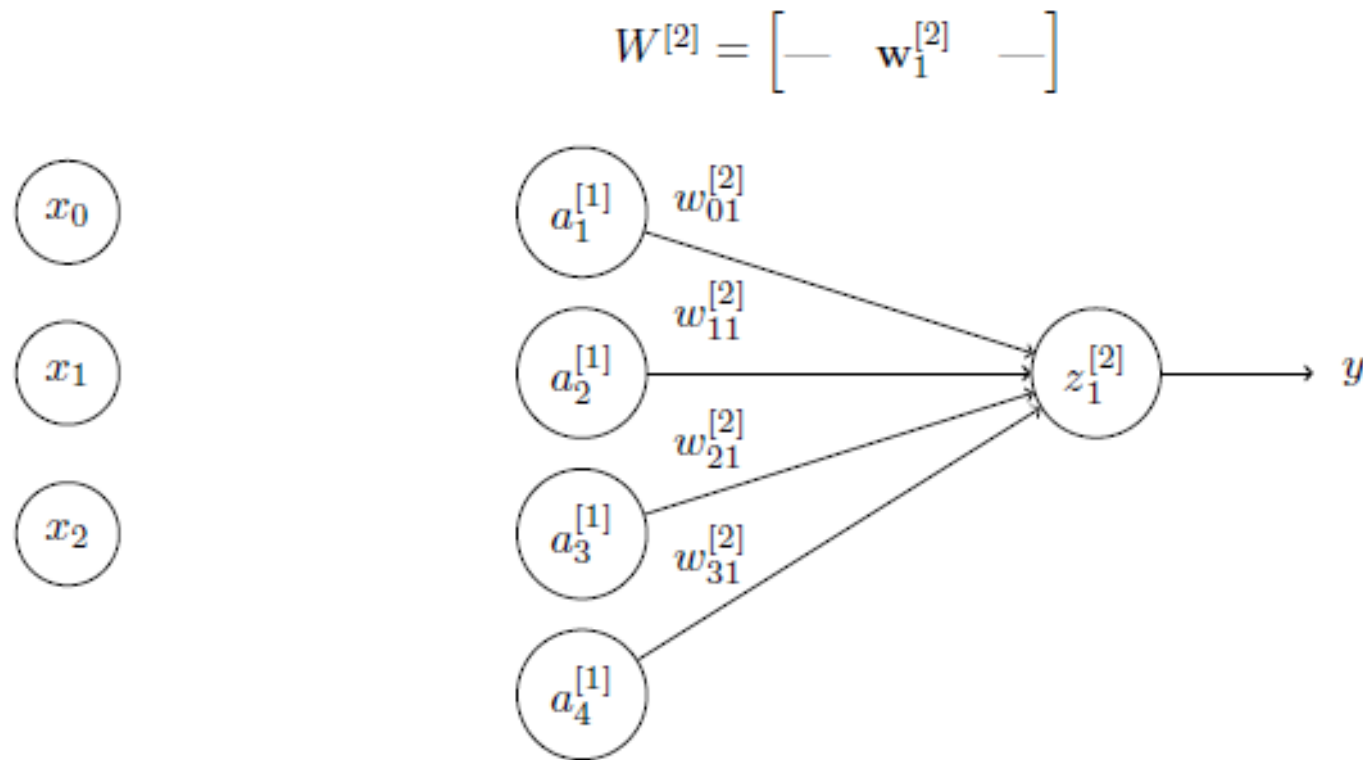
Apenas o 4o neurônio



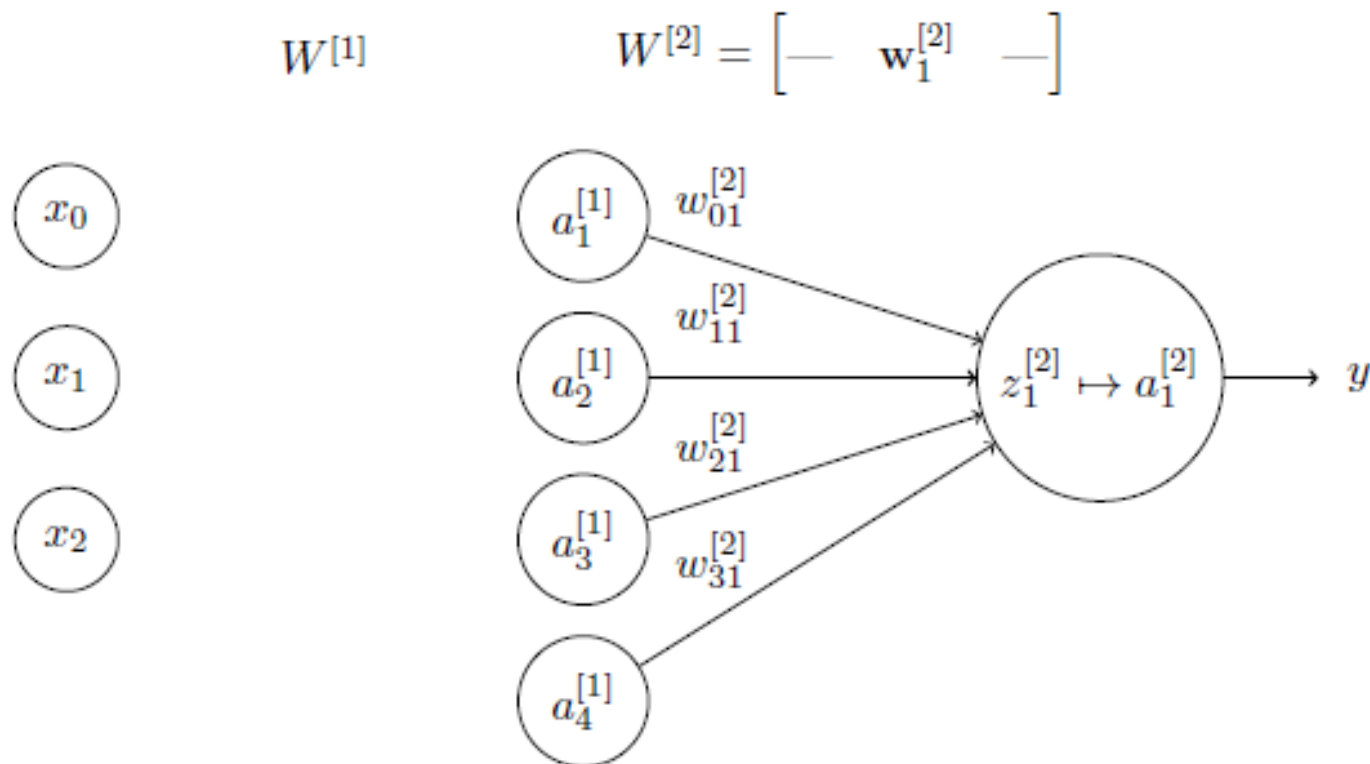
Da camada escondida para a saída



Pesos da camada escondida para a saída



De z para a



Em resumo...

Uma camada escondida com 4 unidades (neurônios) e 3 features de entrada

Primeiro e Segundo passo a frente(forward):

$$\begin{array}{rclcl} z_1^{[1]} & = & \mathbf{w}_1^{[1]T} \mathbf{x} & \rightarrow & a_1^{[1]} = \sigma(z_1^{[1]}) \\ \vdots & & & & \vdots \\ z_4^{[1]} & = & \mathbf{w}_4^{[1]T} \mathbf{x} & \rightarrow & a_4^{[1]} = \sigma(z_4^{[1]}) \end{array}$$

$$z^{[2]} = \mathbf{w}^{[2]T} \mathbf{a}^{[1]} = w_0^{[2]} + w_1^{[2]} a_1^{[1]} + w_2^{[2]} a_2^{[1]} + w_3^{[2]} a_3^{[1]} + w_4^{[2]} a_4^{[1]}$$

$$a^{[2]} = \mathbb{P}(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z^{[2]}}}$$

Vetorizando - notação matricial

Podemos colocar numa notação matricial:

$$\mathbf{z}^{[1]} = \underbrace{\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}}_{4 \times 1} = \underbrace{\begin{bmatrix} \text{---} & \mathbf{w}_1^{[1]} & \text{---} \\ \text{---} & \mathbf{w}_2^{[1]} & \text{---} \\ \text{---} & \mathbf{w}_3^{[1]} & \text{---} \\ \text{---} & \mathbf{w}_4^{[1]} & \text{---} \end{bmatrix}}_{4 \times 3} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}}_{3 \times 1} = \mathbf{W}^{[1]} \mathbf{x}$$

Vetorizando - notação matricial

Podemos colocar numa notação matricial:

$$\mathbf{a}^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \\ \sigma(z_3^{[1]}) \\ \sigma(z_4^{[1]}) \end{bmatrix} = \begin{bmatrix} 1/(1 + e^{-z_1^{[1]}}) \\ 1/(1 + e^{-z_2^{[1]}}) \\ 1/(1 + e^{-z_3^{[1]}}) \\ 1/(1 + e^{-z_4^{[1]}}) \end{bmatrix} = \sigma(\mathbf{z}^{[1]})$$

$$z^{[2]} = w_0^{[2]} + w_1^{[2]} a_1^{[1]} + w_2^{[2]} a_2^{[1]} + w_3^{[2]} a_3^{[1]} + w_4^{[2]} a_4^{[1]}$$

$$a^{[2]} = \mathbb{P}(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z^{[2]}}}$$

Notação de Andrew Ng - isolando o termos de bias

$$\underbrace{\begin{bmatrix} z_1^{[1]} \\ \vdots \\ \vdots \\ z_4^{[1]} \end{bmatrix}}_{z^{[1]} \in \mathbb{R}^{4 \times 1}} = \underbrace{\begin{bmatrix} - & W_1^{[1]T} & - \\ - & W_2^{[1]T} & - \\ & \vdots & \\ - & W_4^{[1]T} & - \end{bmatrix}}_{W^{[1]} \in \mathbb{R}^{4 \times 3}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x \in \mathbb{R}^{3 \times 1}} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_4^{[1]} \end{bmatrix}}_{b^{[1]} \in \mathbb{R}^{4 \times 1}}$$

Vetorizando sobre vários exemplos

Imagine que temos 3 exemplos, rede com 4 unidades:

[.] = camada

(.) = exemplo

$$z^{1} = W^{[1]}x^{(1)} + b^{[1]}$$

$$z^{[1](2)} = W^{[1]}x^{(2)} + b^{[1]}$$

$$z^{[1](3)} = W^{[1]}x^{(3)} + b^{[1]}$$

Defina

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & x^{(3)} \\ | & | & | \end{bmatrix}$$

Então:

$$Z^{[1]} = \begin{bmatrix} | & | & | \\ z^{1} & z^{[1](2)} & z^{[1](3)} \\ | & | & | \end{bmatrix}$$

$$= \overset{4 \times 3}{W^{[1]}} X + \cancel{\overset{4 \times 1}{b^{[1]}}}$$

$$\tilde{b}^{[1]} = \overset{4 \times 3, \text{broadcasting}}{\begin{bmatrix} | & | & | \\ b^{[1]} & b^{[1]} & b^{[1]} \\ | & | & | \end{bmatrix}}$$

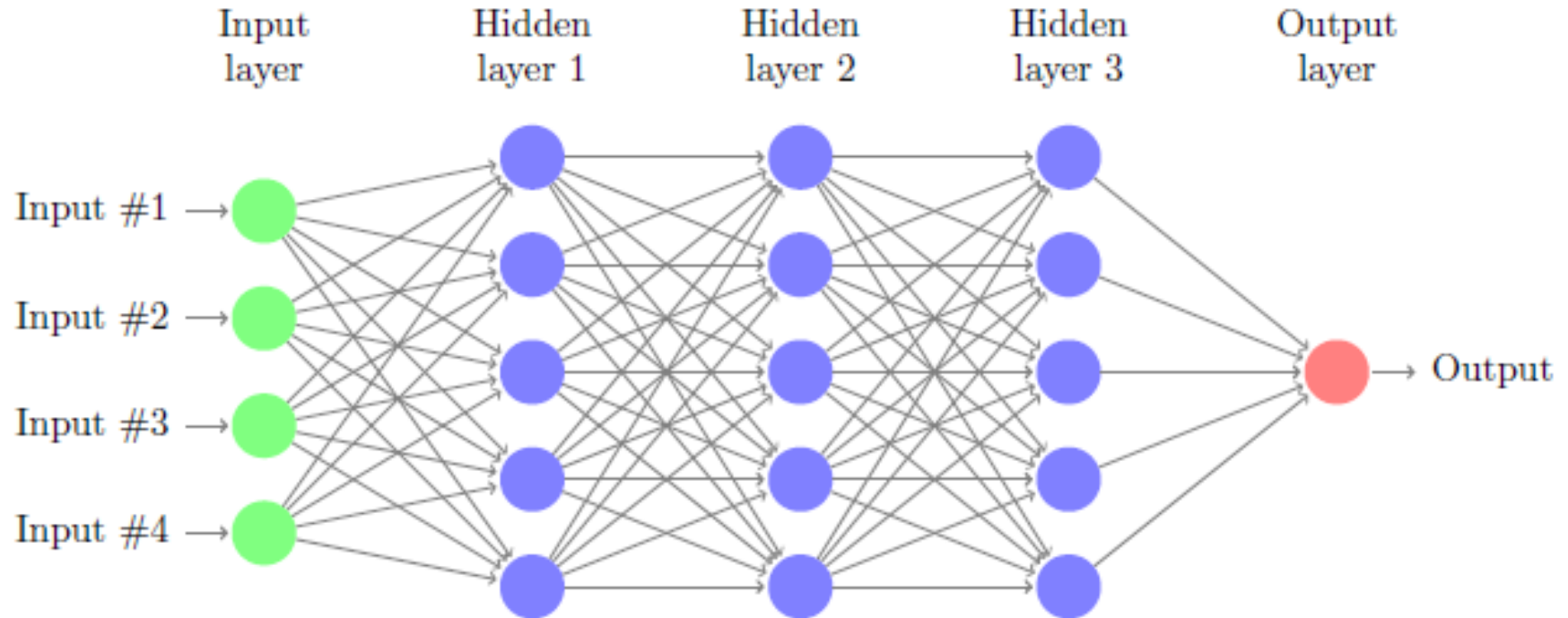
Forward propagation

- Modelo de redes neurais com uma camada:
 - Descreve como os dados observados podem ser gerados a partir dos inputs
 - Dados os inputs-features, a rede propaga os sinais do inputs até a saída.
 - Esta propagação é feita usando:
 - Os pesos w
 - A função de ativação
 - A função de ativação é fixa, escolhida pelo usuário (logística até agora)
 - Pesos são desconhecidos e devem ser aprendidos a partir dos dados
 - Propagação para a frente (saída): forward propagation

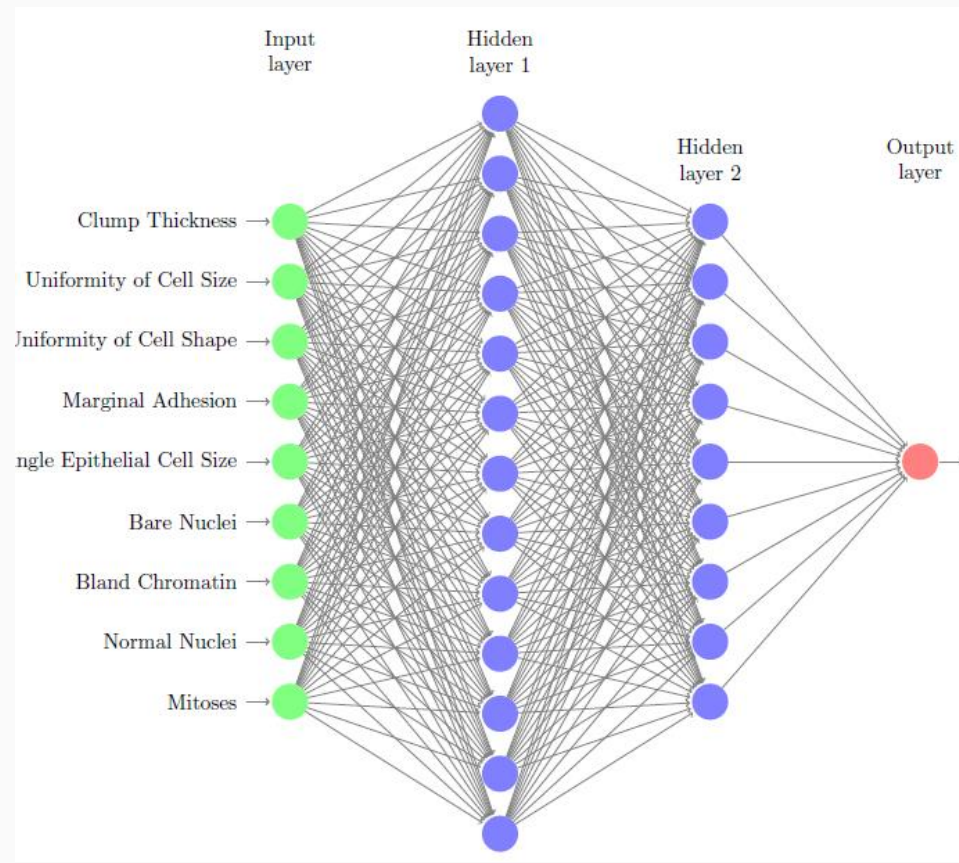
Ajustando e visualizando uma rede neural



<https://playground.tensorflow.org/>

Redes com mais de uma camada escondida



Redes com mais de uma camada escondida



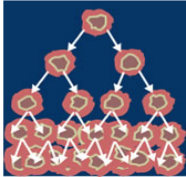


Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	Number of Instances:	569	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	32	Date Donated	1995-11-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	939316

Source:

Creators:

1. Dr. William H. Wolberg, General Surgery Dept.
University of Wisconsin, Clinical Sciences Center
Madison, WI 53792
wolberg '@' eagle.surgery.wisc.edu
2. W. Nick Street, Computer Sciences Dept.
University of Wisconsin, 1210 West Dayton St., Madison, WI 53706
street '@' cs.wisc.edu 608-262-6619
3. Olvi L. Mangasarian, Computer Sciences Dept.
University of Wisconsin, 1210 West Dayton St., Madison, WI 53706
olvi '@' cs.wisc.edu

Donor:

Nick Street

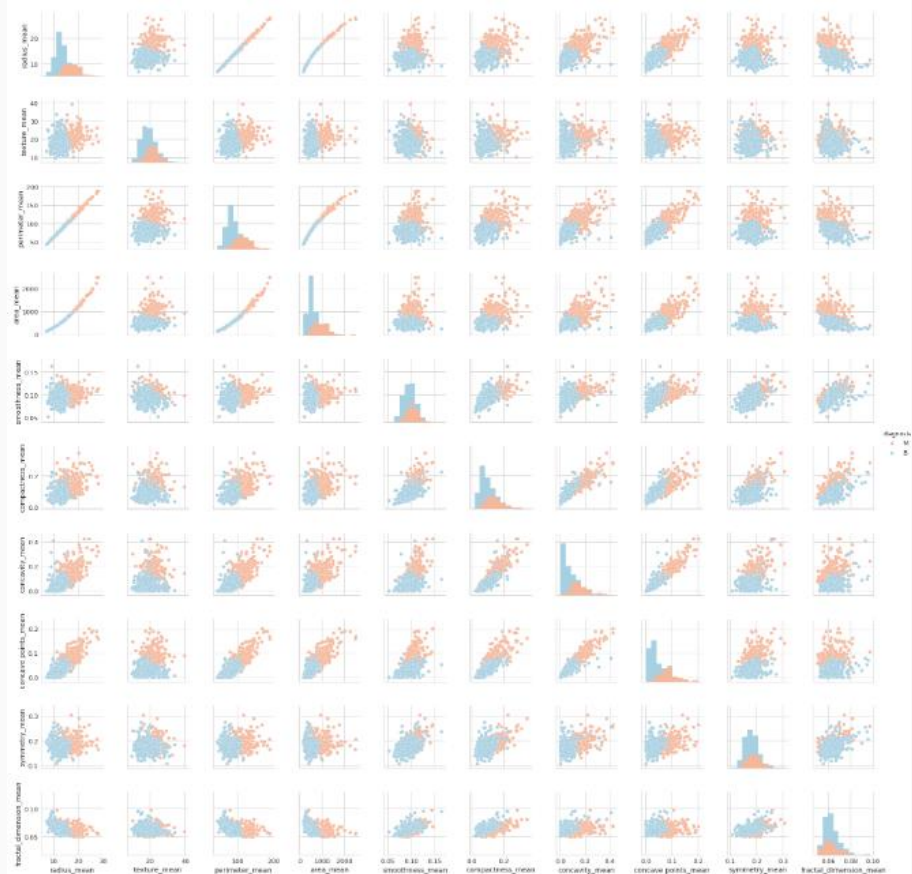
[https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

UCI: Breast Cancer Wisconsin (Diagnostic) Data Set

- 569 exemplos-pacientes
 - classe 1 = câncer de mama presente
 - ou classe 0 = sem câncer de mama
- Em cada imagem, o núcleo de algumas células foram observados.
- Foram medidas 10 variáveis em cada núcleo:
 - a) radius (mean of distances from center to points on the perimeter)
 - b) texture (standard deviation of gray-scale values)
 - c) perimeter
 - d) area
 - e) smoothness (local variation in radius lengths)
 - f) compactness ($\text{perimeter}^2 / \text{area} - 1.0$)
 - g) concavity (severity of concave portions of the contour)
 - h) concave points (number of concave portions of the contour)
 - i) symmetry
 - j) fractal dimension ("coastline approximation" - 1)

- 10 variáveis em cada núcleo/célula.
- Em cada célula, alguns núcleos.
- No final, 30 features em cada imagem = 10 variáveis * 3 resumos
- Exemplo:
 - uma das variáveis é a raio do núcleo (aprox esférico)
 - vários núcleos em cada imagem → vários raios
 - Deriva-se então 3 features:
 - o raio médio dos núcleos
 - o DP dos núcleos
 - a médias dos 3 "piores" (maiores) raios
- No final, 30 features em cada imagem.

Matriz de scatterplots com as 10 features de médias por imagem



Fonte: <https://www.kaggle.com/leemun1/predicting-breast-cancer-logistic-regression>

Wisconsin Breast Cancer Dataset

- Não temos certeza de qual é o melhor número de camadas
- E nem qual é o melhor número de unidades ocultas em cada camada.
- Vamos tentar algumas combinações diferentes:
 - Rede de uma camada escondida com 5 unidades, saída logística
 - Rede de uma camada escondida com 10 unidades, saída logística
 - Rede de uma camada escondida com 15 unidades, saída logística
 - Rede com duas camadas escondidas, 5 unidades em cada uma, saída logística
 - Duas camadas com 10 unidades cada, saída logística
 - Duas camadas com 15 unidades cada, saída logística

Comparação

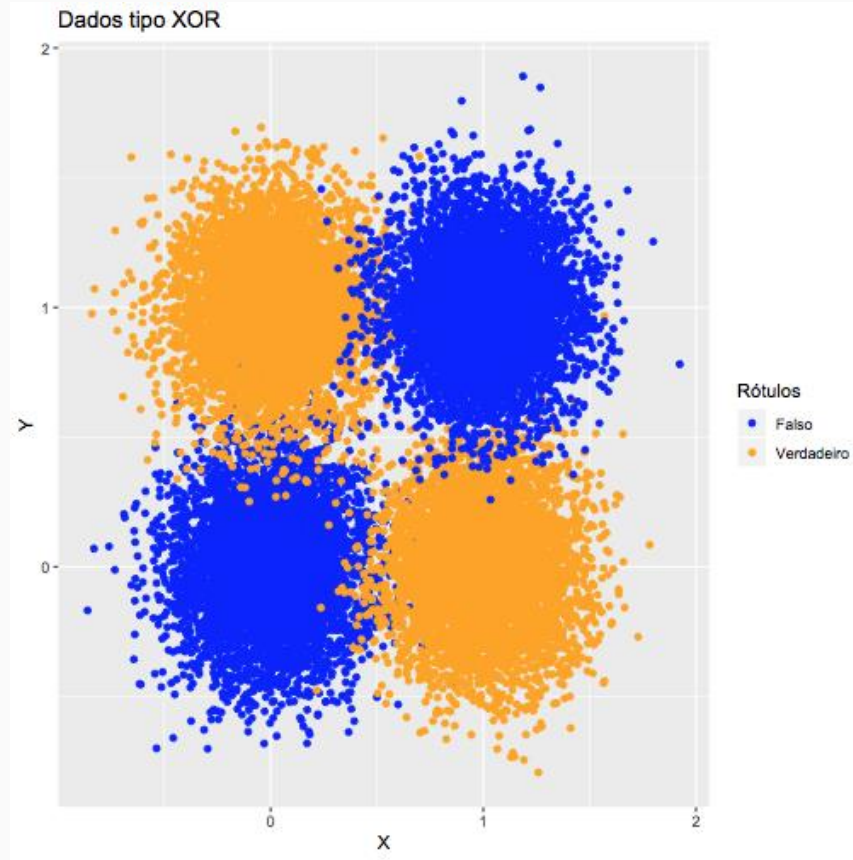
- Usei 80% dos dados (= 455) para treinamento das redes
- Deixei 20% dos dados (= 114) para teste
- Fiz uma análise 5-fold (média sobre 5 partições aleatórias dos dados)

Acurácia = % dos casos de teste corretamente classificada

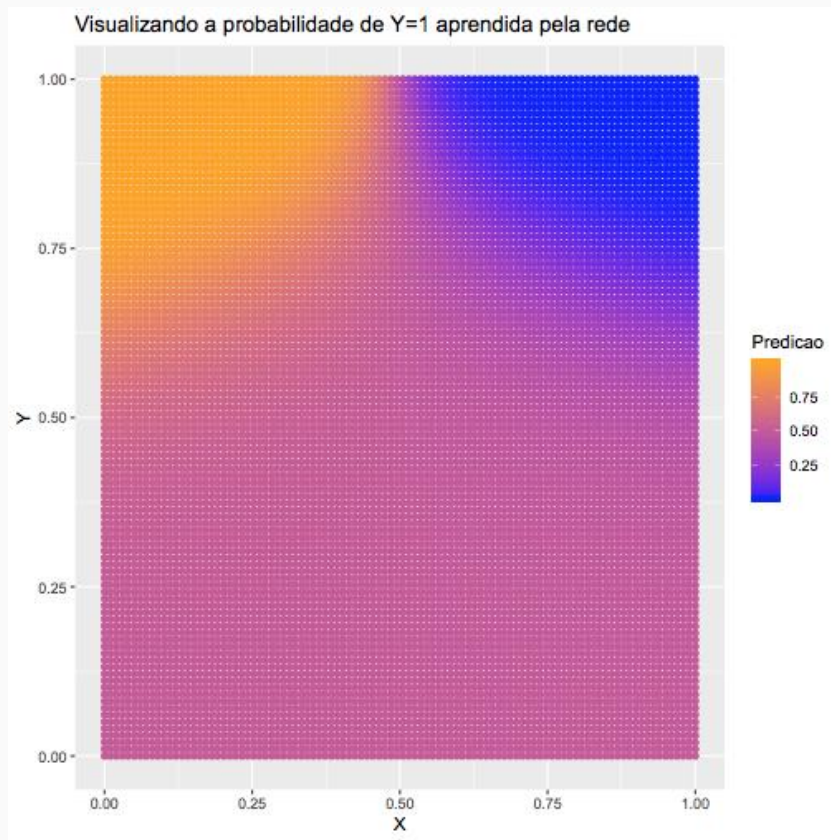
## 1 Hidden Layer, 5 Hidden Units:	0.965
## 1 Hidden Layer, 10 Hidden Units:	0.949
## 1 Hidden Layer, 15 Hidden Units:	0.947
## 2 Hidden Layers, 5 Hidden Units Each:	0.956
## 2 Hidden Layers, 10 Hidden Units Each:	0.895
## 2 Hidden Layers, 15 Hidden Units Each:	0.912

Exemplo inspirado pelo problema XOR

- Um exemplo de classificação baseado em dados tipo XOR
- $Y = 1$ (amarelo) se:
 - $X < 0$ and $Y > 0$... ou
 - $X > 0$ and $Y < 0$
- $Y = 0$ (blue) caso contrário
- Tentando uma rede com:
 - Input: x e y
 - Uma camada escondida
 - Duas unidades

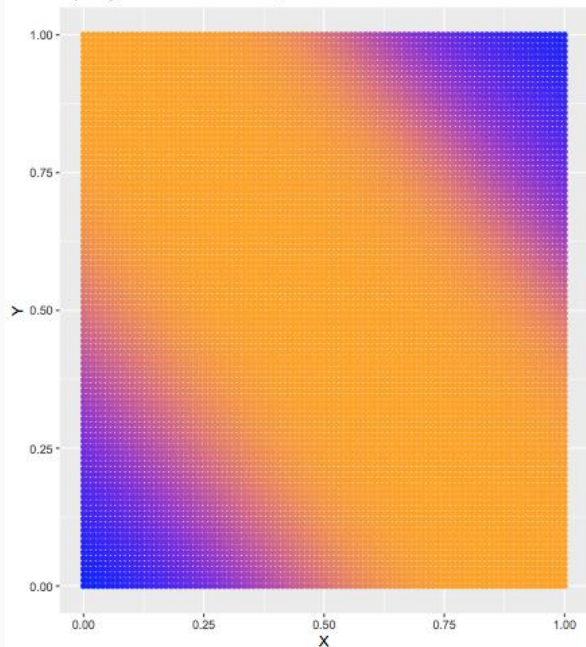


Resultado

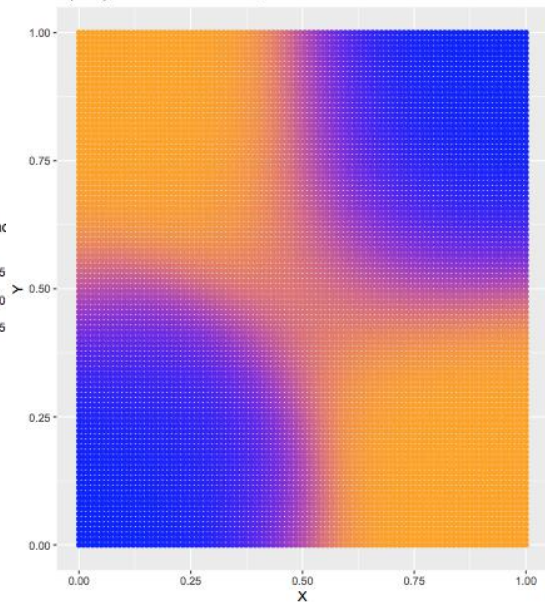


Variando a rede, ativação logística

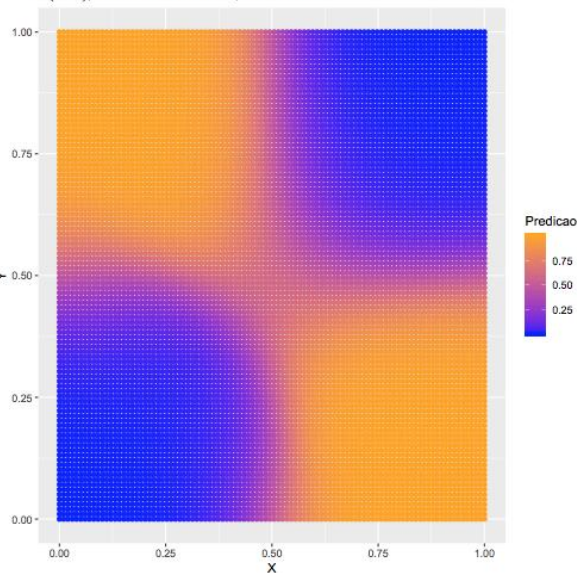
P(Y=1), rede com 1 camada, 4 unidades



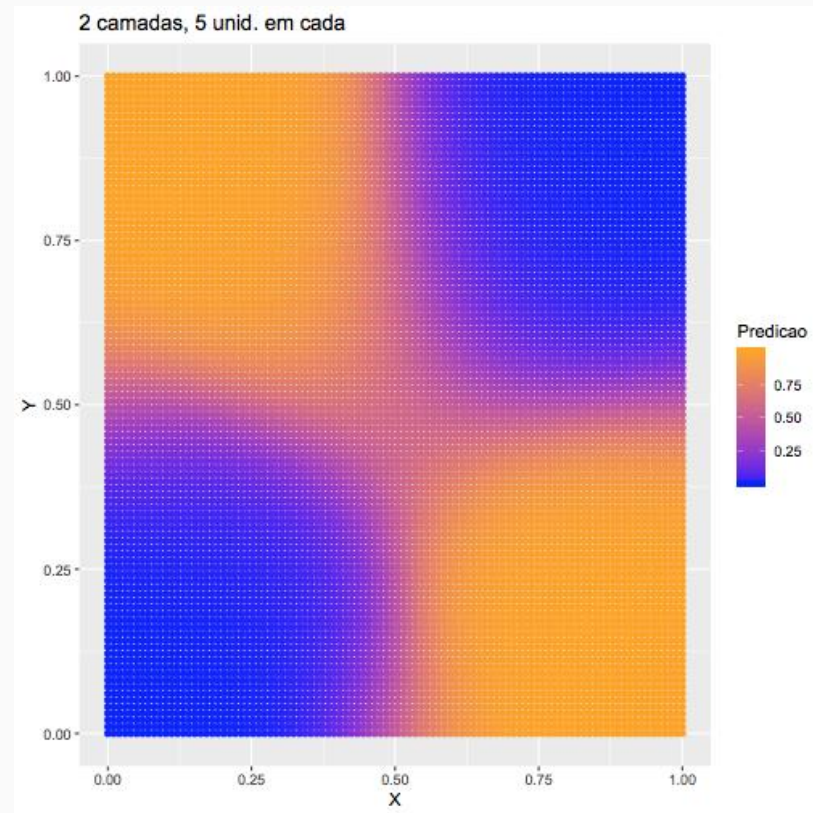
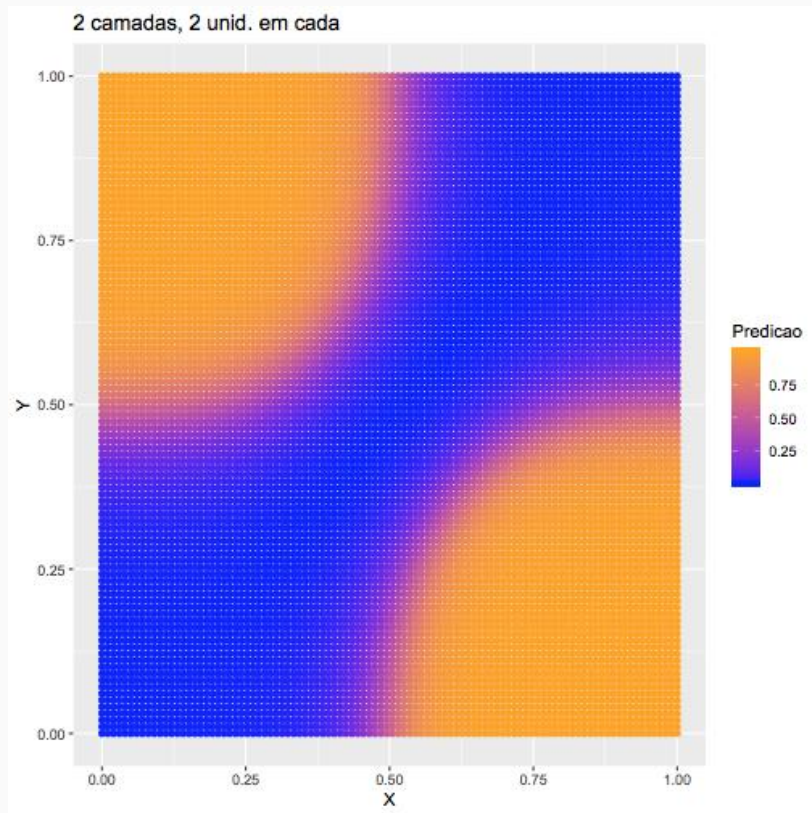
P(Y=1), rede com 1 camada, 5 unidades



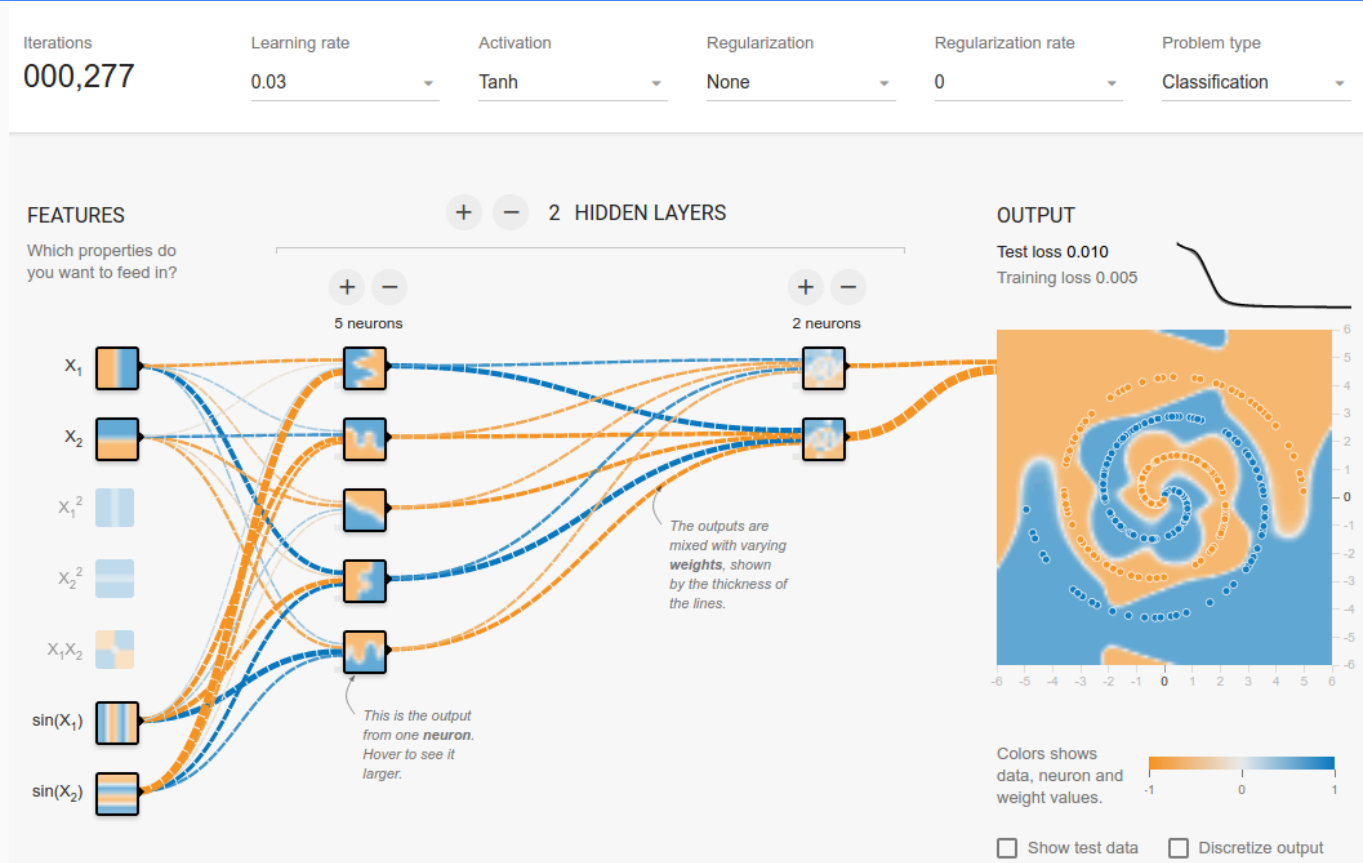
P(Y=1), rede com 1 camada, 10 unidades



Com duas camadas escondidas



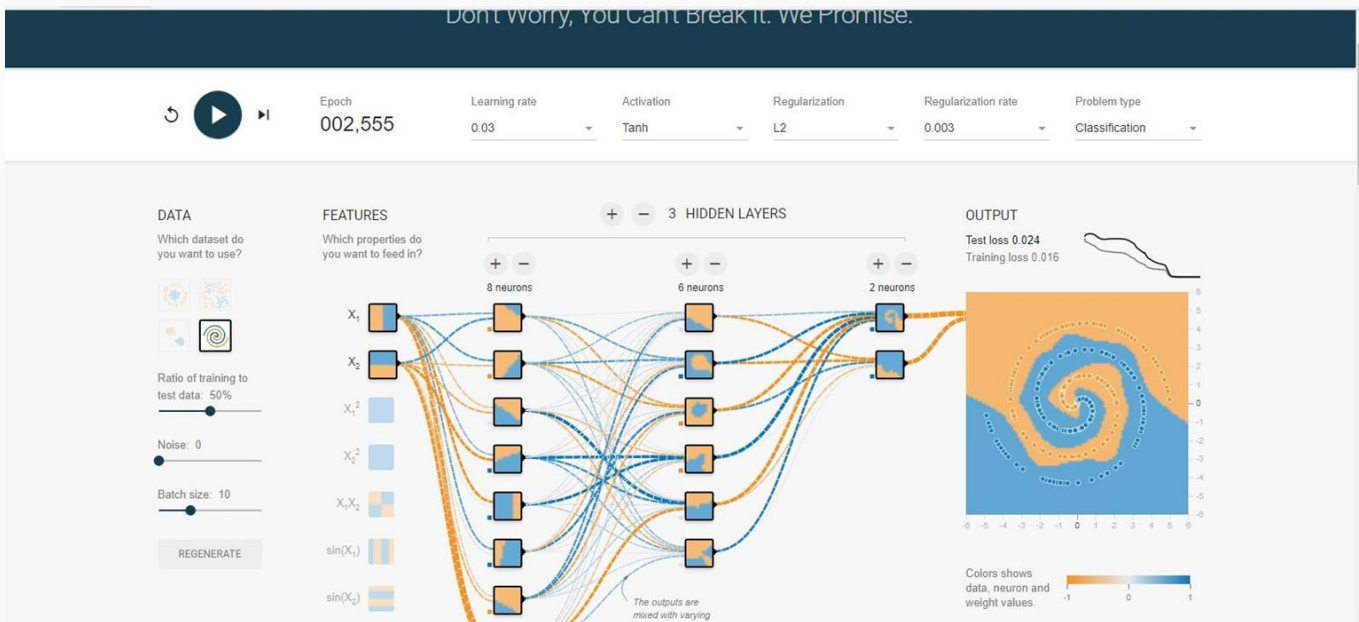
Spiral example - <https://playground.tensorflow.org>



From

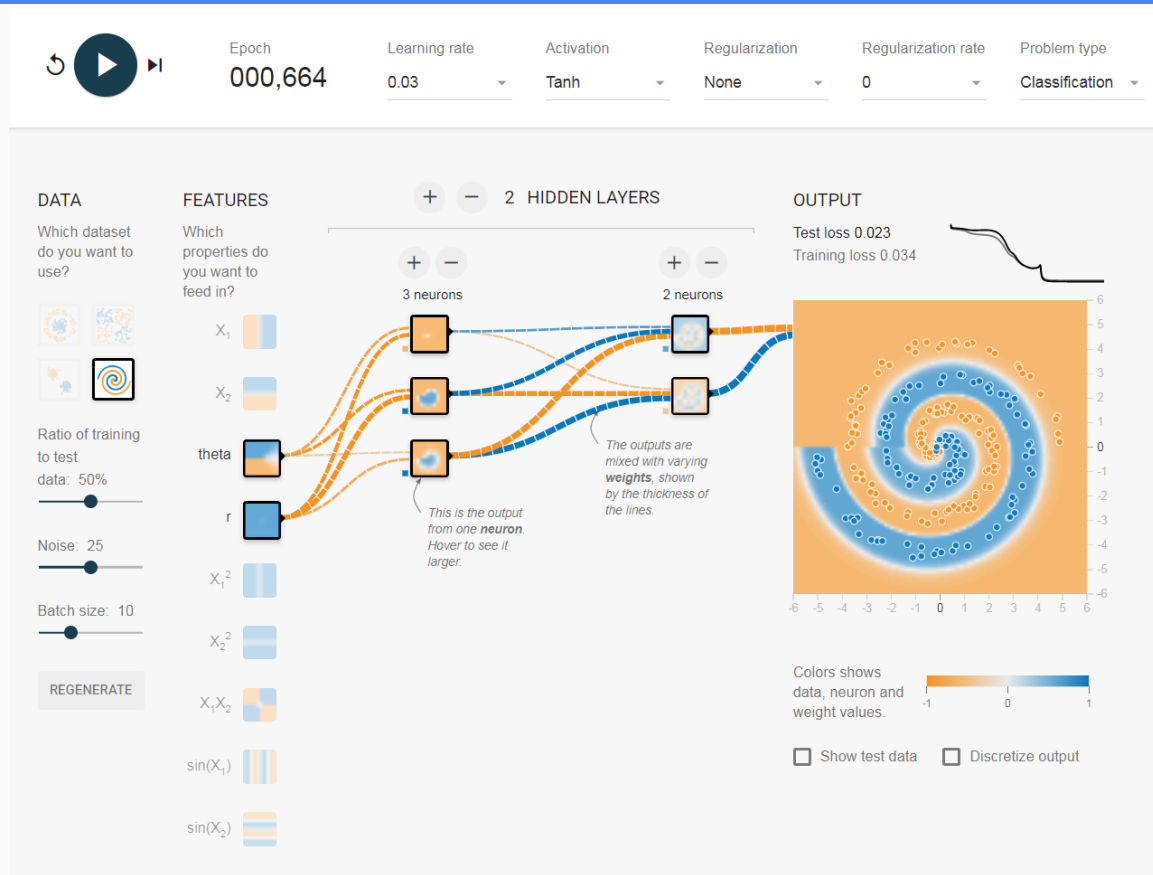
<https://ai.stackexchange.com/questions/1987/how-to-classify-data-which-is-spiral-in-shape>

Spiral example - <https://playground.tensorflow.org>



From <https://ai.stackexchange.com/questions/1987/how-to-classify-data-which-is-spiral-in-shape>

Spiral example - playground - TensorFlow



From

<https://ai.stackexchange.com/questions/1987/how-to-classify-data-which-is-spiral-in-shape>

By cheating... θ is

$\arctan(y, x)$

$r = \sqrt{x^2 + y^2}$.

Várias camadas

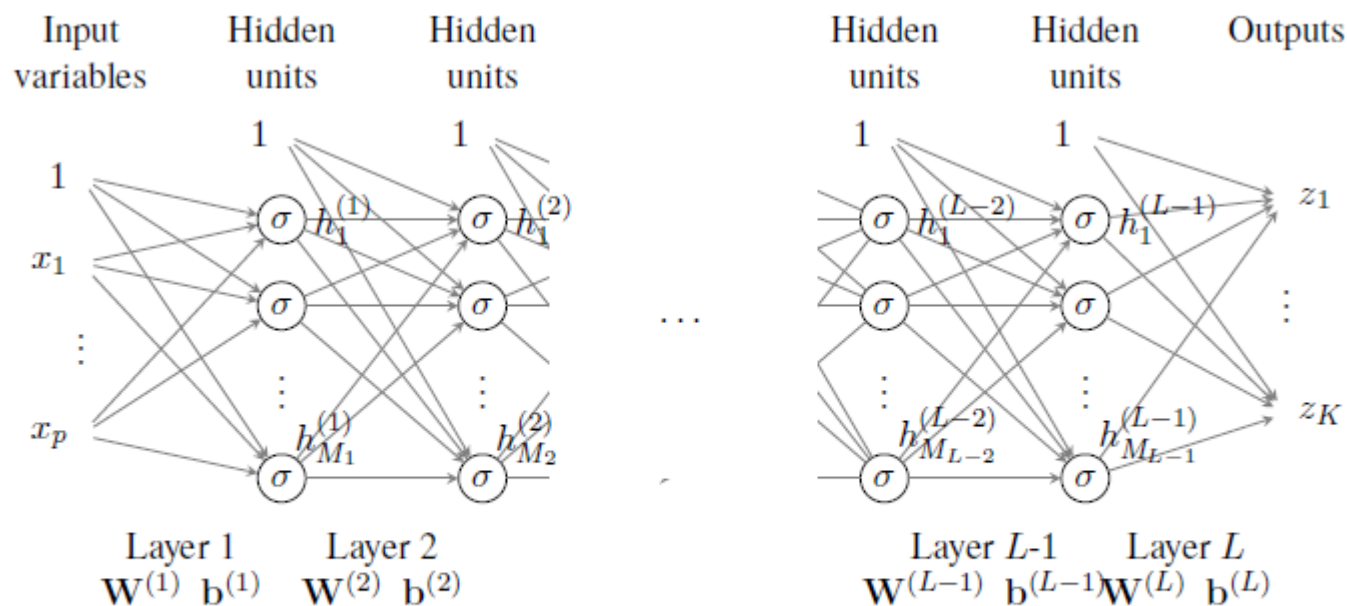


Figure 7.4: A deep neural network with L layers. Each layer is parameterized with $W^{(l)}$ and $b^{(l)}$.

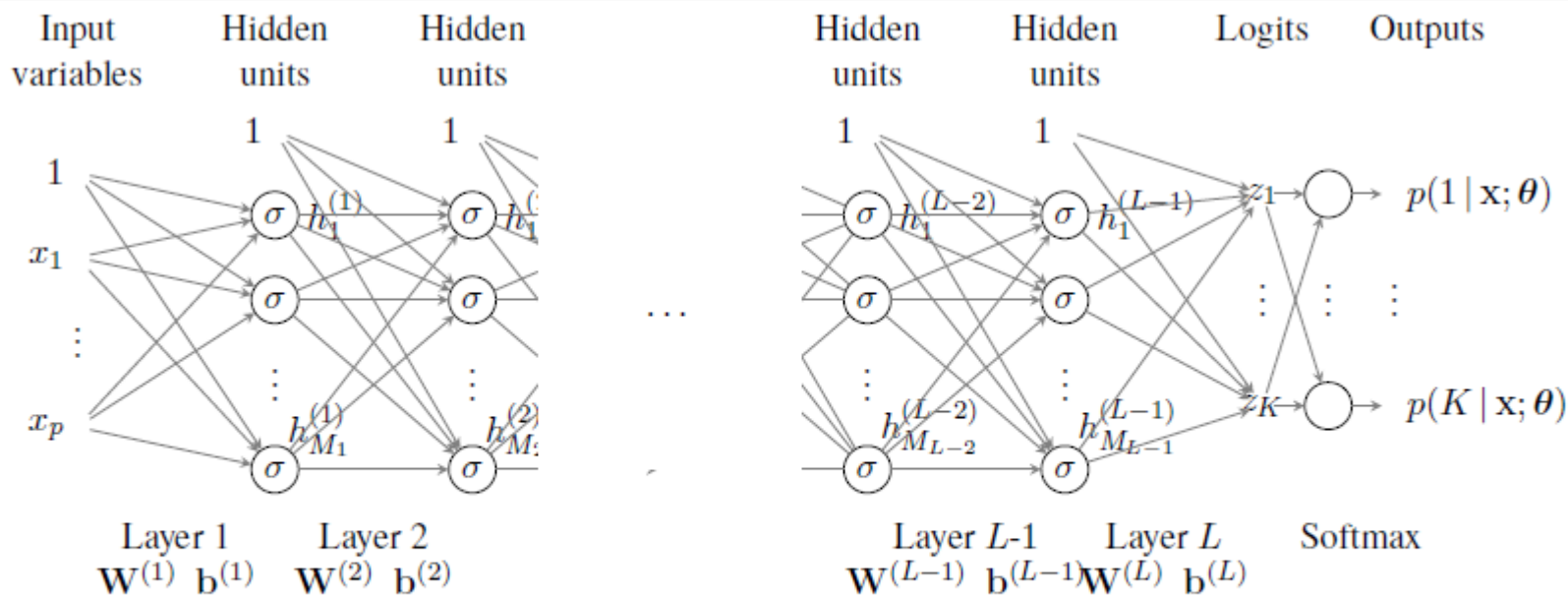
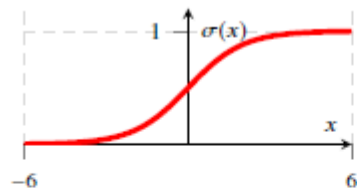


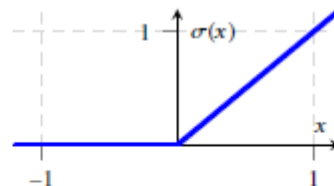
Figure 7.5: A deep neural network with L layers for classification. The only difference to regression (Figure 7.4) is the softmax transformation after layer L .

Funções de ativação



Logistic: $\sigma(x) = \frac{1}{1+e^{-x}}$

(a)



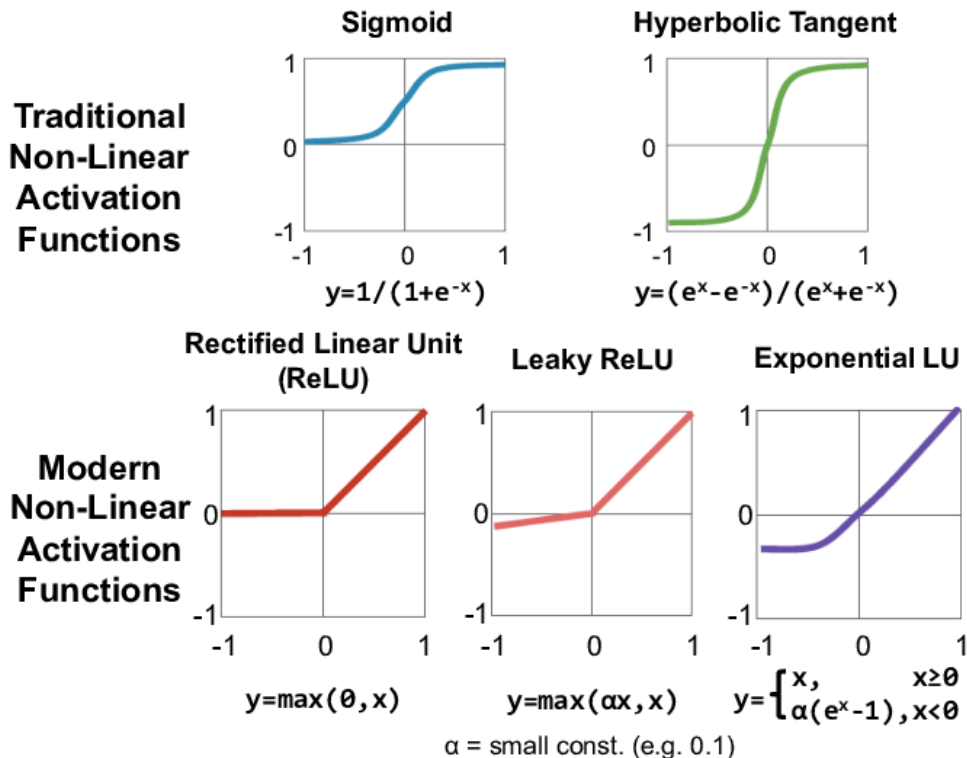
ReLU: $\sigma(x) = \max(0, x)$

(b)

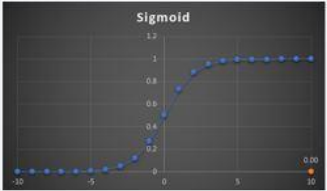
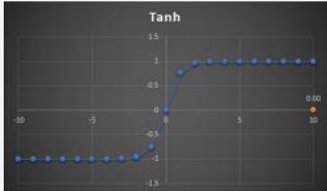
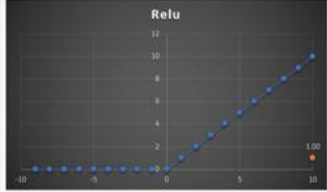
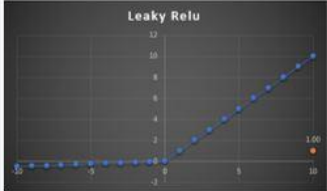
Figure 7.2: Two common activation functions used in neural networks. The logistic (or sigmoid) function (Figure 7.2a), and the rectified linear unit (Figure 7.2b).

Função de ativação diferentes da logística

- Até agora, estivemos vendo apenas uma função de ativação: a logística
- Deep Learning faz uso de outras funções de ativação além da logística.



As derivadas das funções de ativação

Name	Plot	Equation	Derivative
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$
Rectified Linear Unit (relu)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky Rectified Linear Unit (Leaky relu)		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

De volta a terminologia: bias term, beta ou w

Input variables Hidden units Output

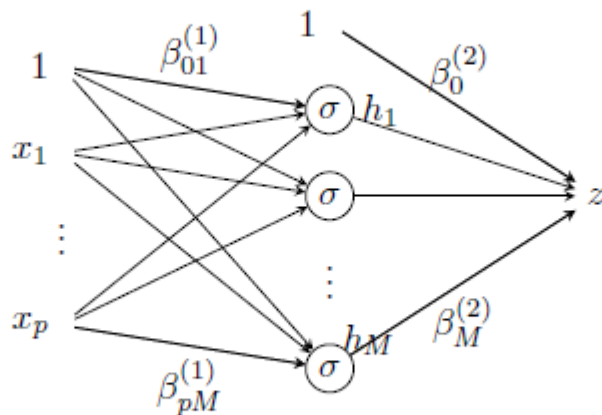


Figure 7.3: A two-layer neural network, or equivalently, a neural network with one intermediate layer of hidden units.

Mas funciona? Isto pode dar certo?

Teorema de Kolmogorov

- Em 1900, o matemático David Hilbert formulou 23 problemas desafiadores em que os matemáticos do século XX deveriam trabalhar.
- Em 1956, um desses problemas foi parcialmente resolvido por Kolmogorov e seu estudante Arnol'd
- Melhorias no teorema de Kolmogorov foram obtidas por Sprecher (1963) e Lorentz (1966).

Theorem 1 (Kolmogorov's Superposition Theorem). *For each $n \geq 2$ there exist continuous functions $\varphi_q : [0, 1] \rightarrow \mathbb{R}$, $q = 0, \dots, 2n$ and positive constants $\lambda_p \in \mathbb{R}$, $p = 1, \dots, n$ such that the following holds true: for each continuous function $f : [0, 1]^n \rightarrow \mathbb{R}$ there exists a continuous function $g : [0, 1] \rightarrow \mathbb{R}$ such that*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} g \left(\sum_{p=1}^n \lambda_p \varphi_q(x_p) \right).$$

Teorema de Kolmogorov - 2

Theorem 1 (Kolmogorov's Superposition Theorem). *For each $n \geq 2$ there exist continuous functions $\varphi_q : [0, 1] \rightarrow \mathbb{R}$, $q = 0, \dots, 2n$ and positive constants $\lambda_p \in \mathbb{R}$, $p = 1, \dots, n$ such that the following holds true: for each continuous function $f : [0, 1]^n \rightarrow \mathbb{R}$ there exists a continuous function $g : [0, 1] \rightarrow \mathbb{R}$ such that*

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} g \left(\sum_{p=1}^n \lambda_p \varphi_q(x_p) \right).$$

- Exemplo com $n=2$: Existem cinco funções de uma única variável $\varphi_0, \varphi_1, \dots, \varphi_4$ e constantes λ_1 e λ_2 tais que toda função contínua $f(x,y)$ pode ser escrita de forma exata como

$$f(x, y) = g(\lambda_1 \varphi_0(x) + \lambda_2 \varphi_0(y)) + g(\lambda_1 \varphi_1(x) + \lambda_2 \varphi_1(y)) + \dots + g(\lambda_1 \varphi_4(x) + \lambda_2 \varphi_4(y))$$

Teorema de Kolmogorov - 3

- Cinco funções de uma única variável \rightarrow para toda e qualquer $f(x,y)$
- Constantes λ_1 e λ_2 \rightarrow para toda e qualquer $f(x,y)$
- Com as $\varphi_0, \varphi_1, \dots, \varphi_4$ crie cinco features baseadas apenas em x e apenas em y :
 - $\varphi_0(x), \varphi_1(x)\varphi_2(x), \varphi_3(x), \varphi_4(x)$
 $\varphi_0(y), \varphi_1(y)\varphi_2(y), \varphi_3(y), \varphi_4(y)$

- Imagine, por exemplo, que $\varphi_0(x) = x, \varphi_1(x) = x^2, \dots, \varphi_4(x) = x^5$
- Forme combinações lineares duas a duas das cinco features (a MESMA combinação)
- Use uma função de ativação g e some os resultados: obtemos $f(x,y)$

$$f(x,y) = g(\lambda_1\varphi_0(x) + \lambda_2\varphi_0(y)) + g(\lambda_1\varphi_1(x) + \lambda_2\varphi_1(y)) + \dots + g(\lambda_1\varphi_4(x) + \lambda_2\varphi_4(y))$$

Teorema de Kolmogorov e

Em 1987, Hecht-Nielsen aplica o teorema de Kolmogorov para mostrar que toda e qualquer função contínua de n inputs pode ser escrita como uma rede neural de 3 camadas usando apenas duas funções (φ e g)

Hecht-Nielsen (1987) Proceedings IEEE Int Conf Neural Networks, vol II

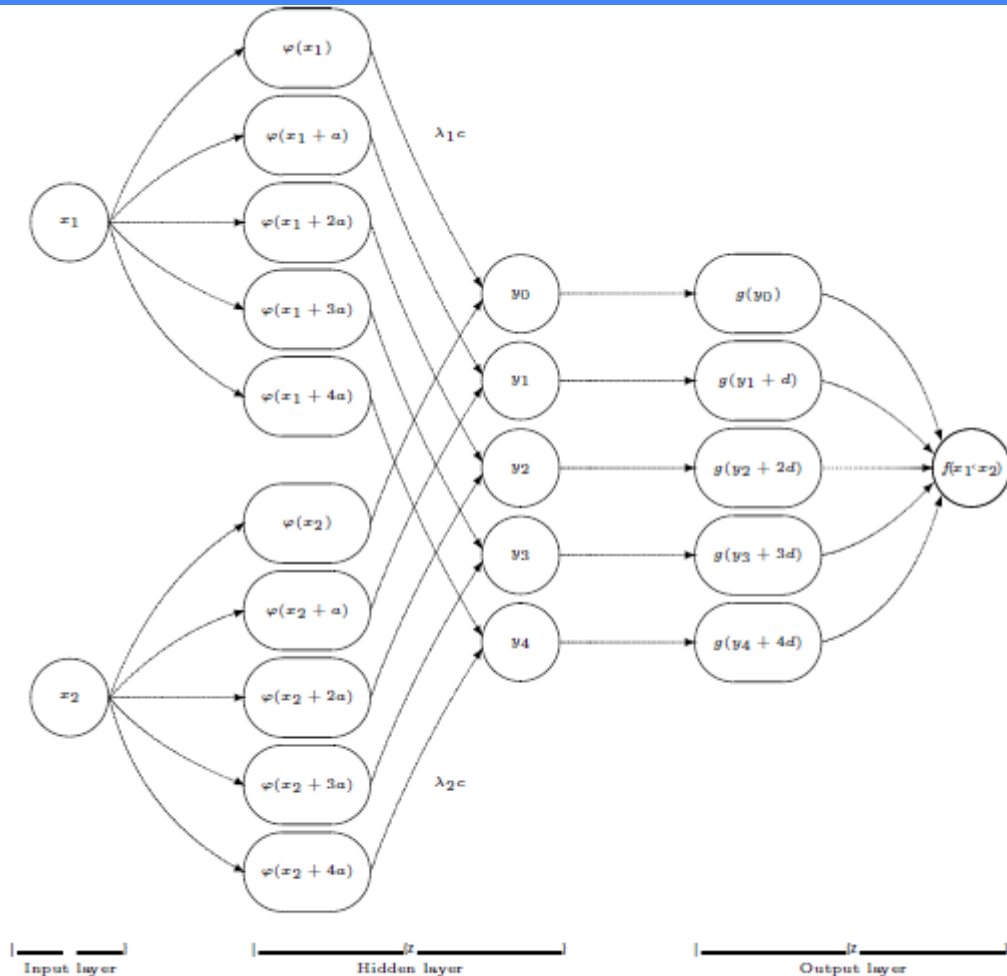


Fig. 13.6. Kolmogorov neural network for $n = 2$

Relevância do teorema de Kolmogorov

- Relevância prática: Limitada
 - A prova do teorema não é construtiva: não nos diz como obter as funções envolvidas
 - É um teorema de existência.
 - Ela nos diz que essa rede neural de três camadas deve existir, mas não nos diz como encontrá-la.
 - Infelizmente, não parece haver muita esperança de que um método para encontrar a rede Kolmogorov seja desenvolvido em breve.
- Relevância teórica: Grande:
 - Redes neurais de 3 camadas são capazes de representar toda e qualquer função de n inputs.
 - Apenas DUAS funções desconhecidas envolvidas na rede: φ e g
 - Isto dá a “esperança” de que nossas redes com ativações RELU ou logística consigam representar bem as funções $\mathbb{E}(Y|\mathbf{x})$

Neural Networks, Vol. 2, pp. 359–366, 1989
Printed in the USA. All rights reserved.

0893-6080/89 \$3.00 + .00
Copyright © 1989 Pergamon Press plc

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE

University of California, San Diego

(Received 16 September 1988; revised and accepted 9 March 1989)

Exemplo - Teorema universal com uma única variável

```
# Teorema universal no caso uni-dimensional
```

```
x = seq(-100, 100, by =0.1)
```

```
a1 = 1/(1+exp(-0.35*(x - 5)))
```

```
a2 = 1/(1+exp(+0.35*(x-25)))
```

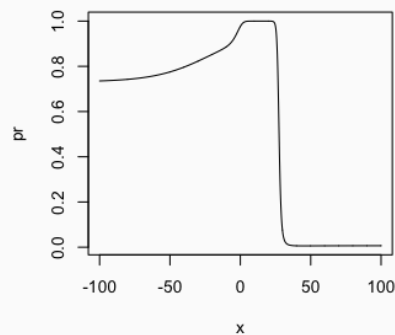
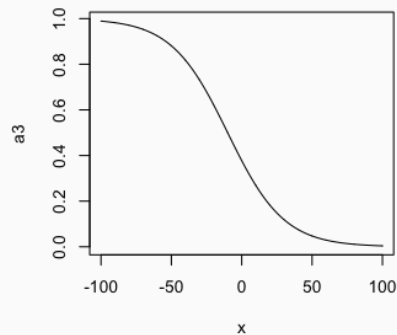
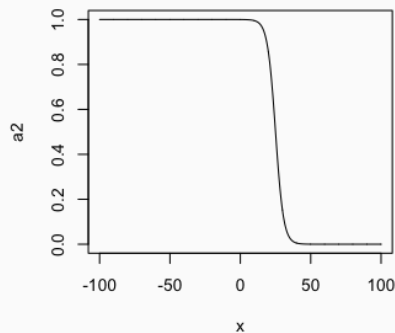
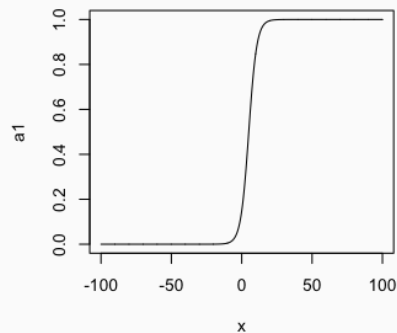
```
a3 = 1/(1 +exp(0.05*(x+10)))
```

```
pr = 1/(1+exp(-(-15 + 10*a1 + 18*a2 -2*a3 )))
```

```
par(mfrow=c(2,2))
```

```
plot(x, a1, type="l"); plot(x, a2, type="l"); plot(x, a3, type="l"); plot(x, pr, type="l")
```

```
quartz.save("TeoremaUniversal01.png")
```



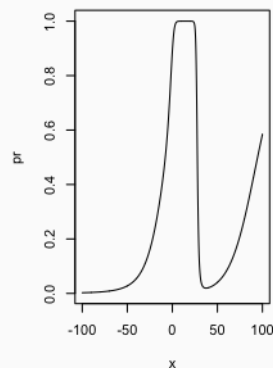
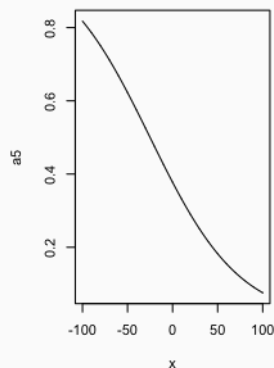
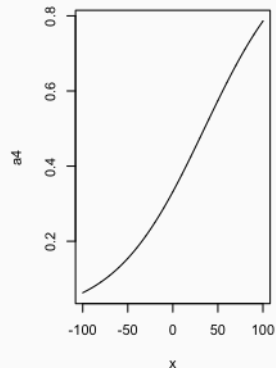
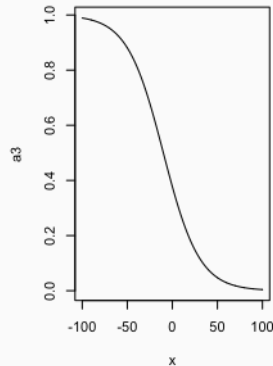
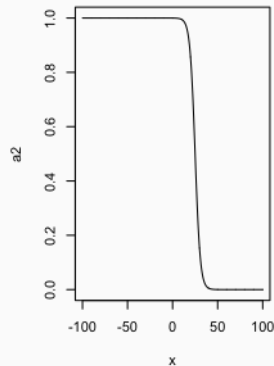
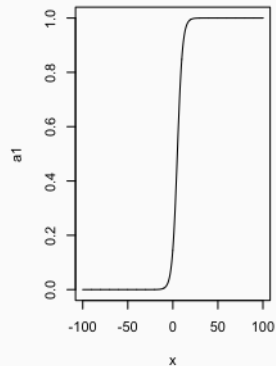
$$\mathbb{P}(Y = 1|x) = \frac{1}{1 + e^{-(-15 + 10a_1(x) + 18a_2(x) - 2a_3(x))}}$$

$$a_1(x) = \frac{1}{1 + e^{-(-1.75 + 0.35x)}}$$

$$a_2(x) = \frac{1}{1 + e^{-(8.75 - 0.35x)}}$$

$$a_3(x) = \frac{1}{1 + e^{-(-0.5 - 0.05x)}}$$

Outro exemplo - Teorema universal com uma única variável



$$a_1(x) = \frac{1}{1 + e^{-(-1.75 + 0.35x)}}$$

$$a_2(x) = \frac{1}{1 + e^{-(8.75 - 0.35x)}}$$

$$a_3(x) = \frac{1}{1 + e^{-(-0.5 - 0.05x)}}$$

$$a_4(x) = \frac{1}{1 + e^{-(-0.7 + 0.02x)}}$$

$$a_5(x) = \frac{1}{1 + e^{-(-0.5 - 0.02x)}}$$

$$\mathbb{P}(Y = 1|x) = \frac{1}{1 + e^{-(-15 + 10a_1(x) + 18a_2(x) - 2a_3(x) + 10a_4(x) - 10a_5(x))}}$$

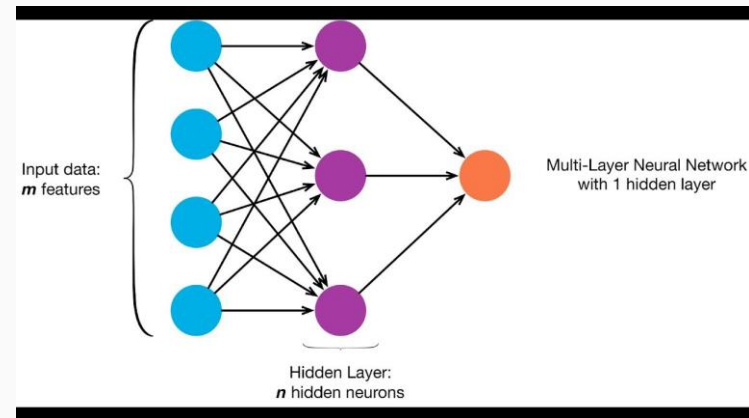
Teorema da aproximação universal

- Hornik, Stinchcombe e White (1989) provaram um teorema muito importante
- Conhecido como Teorema da Aproximação Universal
- **Abstract:**
- This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.
- ANY Squashing functions = ANY activation function

Teorema da aproximação universal - debulhando...

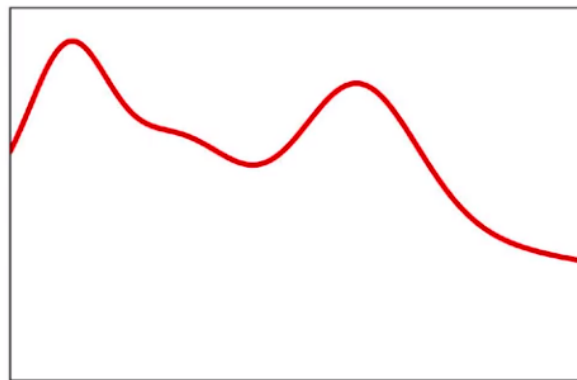
- Qualquer problema de classificação ou regressão pode ser resolvido com uma rede neural com uma única camada.
- Talvez a camada precise de muitos neurônios mas basta uma única camada!!
- Uma pouco mais formalmente: Seja $h(\mathbf{x}) = \mathbb{E}(Y|\mathbf{x})$ uma função desconhecida
- Existe uma rede neural com uma única camada tal que sua saída $\hat{y}(\mathbf{x})$ é muito próxima de $h(\mathbf{x})$ para todo \mathbf{x}
- Escolha ϵ bem pequeno (por exemplo, $\epsilon = 10^{-5}$)
- Então existe rede neural de uma camada tal que

$$|h(\mathbf{x}) - \hat{y}(\mathbf{x})| < \epsilon \quad \forall \mathbf{x}$$



Illustrative Proof of Universal Approximation Theorem

- Extraído de post de Niranjana Kumar
 - <https://hackernoon.com/illustrative-proof-of-universal-approximation-theorem-5845c02822f6>
- Suponha que $h(\mathbf{x}) = \mathbb{E}(Y|\mathbf{x})$ seja a função abaixo (é 1-dim)
- Nós não conhecemos (nem vemos) a “fórmula” $h(x)$
- Não temos uma representação $h(x)$

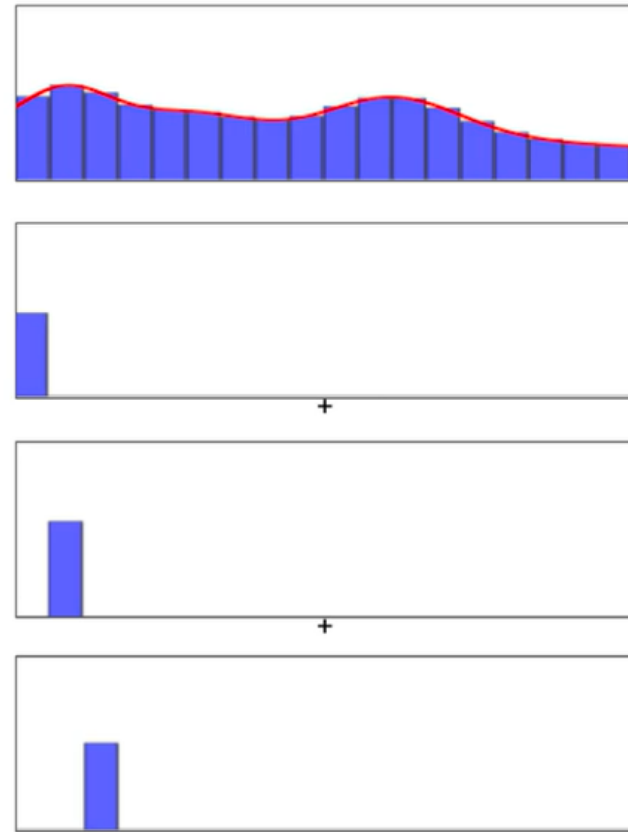


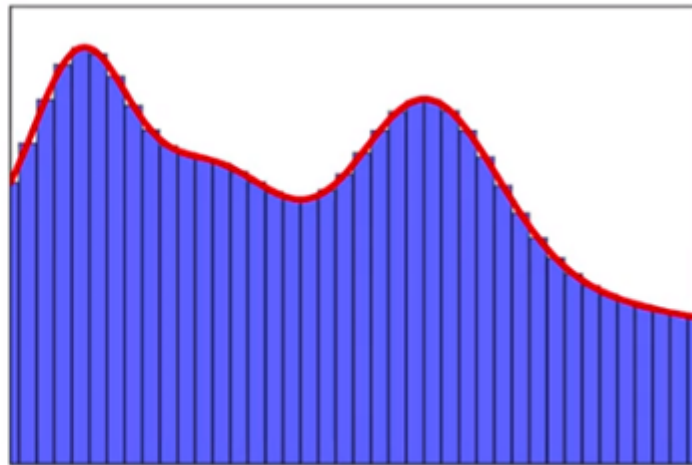
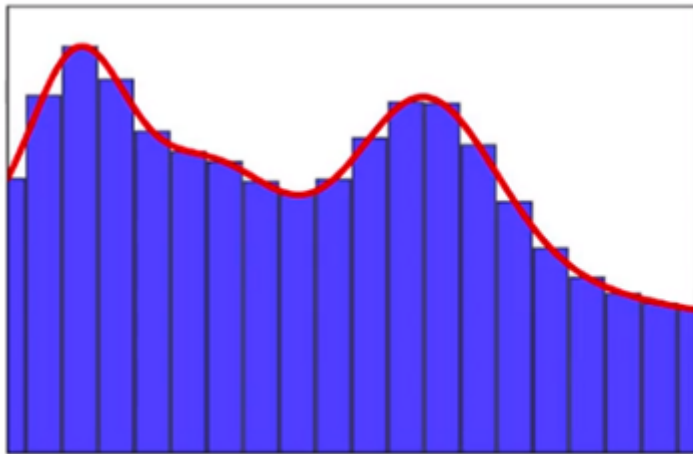
- Aproximamos a função $h(x)$ por funções escada (step functions)
- Cada step function é função diferente de zero apenas num dos intervalos-bins

$$c_i I[x \in (a_i, b_i)]$$

- $h(x)$ é aprox a soma das step functions

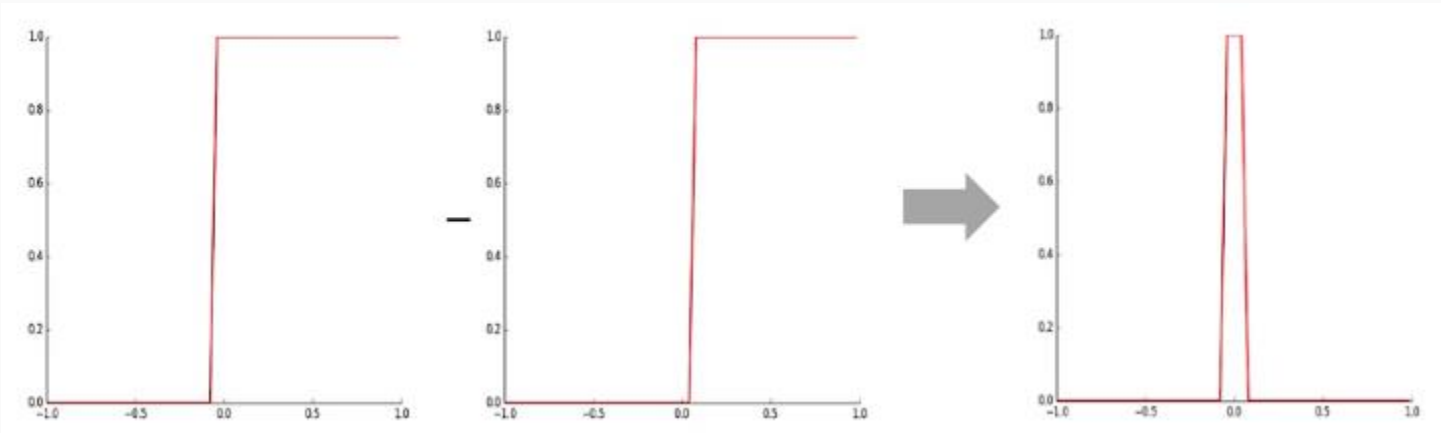
$$h(x) \approx \sum_i c_i I[x \in (a_i, b_i)]$$





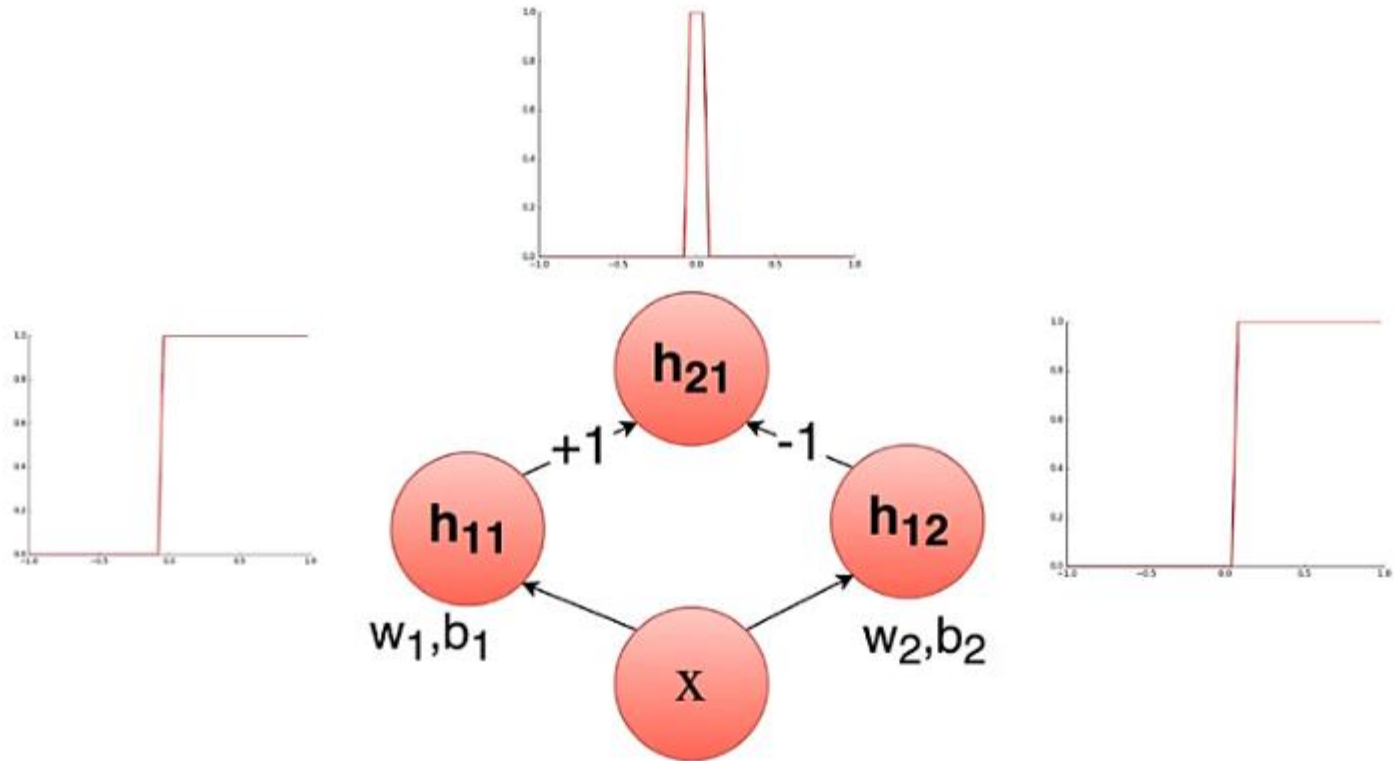
- Podemos fazer a aproximação ser tão boa quanto quisermos (menor que qualquer epsilon fixado)
- Basta tomar bins-intervalos cada vez menores
- Mas como esta função constante por partes pode ser escrita como soma de logísticas?

Step function é aprox a diferença de duas logísticas * c_i



- Tome duas logísticas muito íngremes e ligeiramente deslocadas no eixo horizontal
- Tome a sua diferença: teremos praticamente uma step function
- Multiplique o resultado por c_i
- Como estabelecer uma rede neural para representar essa operação de subtrair uma função logística de outra?

Representação de step function como rede neural



Relevância do teorema universal

- O teorema da aproximação universal é um teorema poderoso.
- Ele indica que uma rede neural é um aproximador universal de funções: qualquer função pode ser extremamente bem aproximada por uma rede neural de UMA camada escondida.
- No entanto, ele não diz nada sobre o tamanho da rede neural de camada oculta.
- Apenas que é finita.
- E que você pode tornar a rede neural arbitrariamente próxima da função original simplesmente lançando mais unidades ocultas nela.
- Isto é pouco ou muito?
- Serve como uma garantia psicológica de que redes neurais dão conta do recado.
- IMPORTANTE: rede simples de duas camadas pode ser muito melhor que rede complexa de uma única camada. Ver exemplo a seguir.

Modelo gerador dos dados é uma árvore com dois inputs

Rede com 6 unidades em uma camada escondida faz um serviço ok. Precisa de muito mais unidades com uma camada.

Rede de duas camadas com 3 + 4 unidades é perfeita.

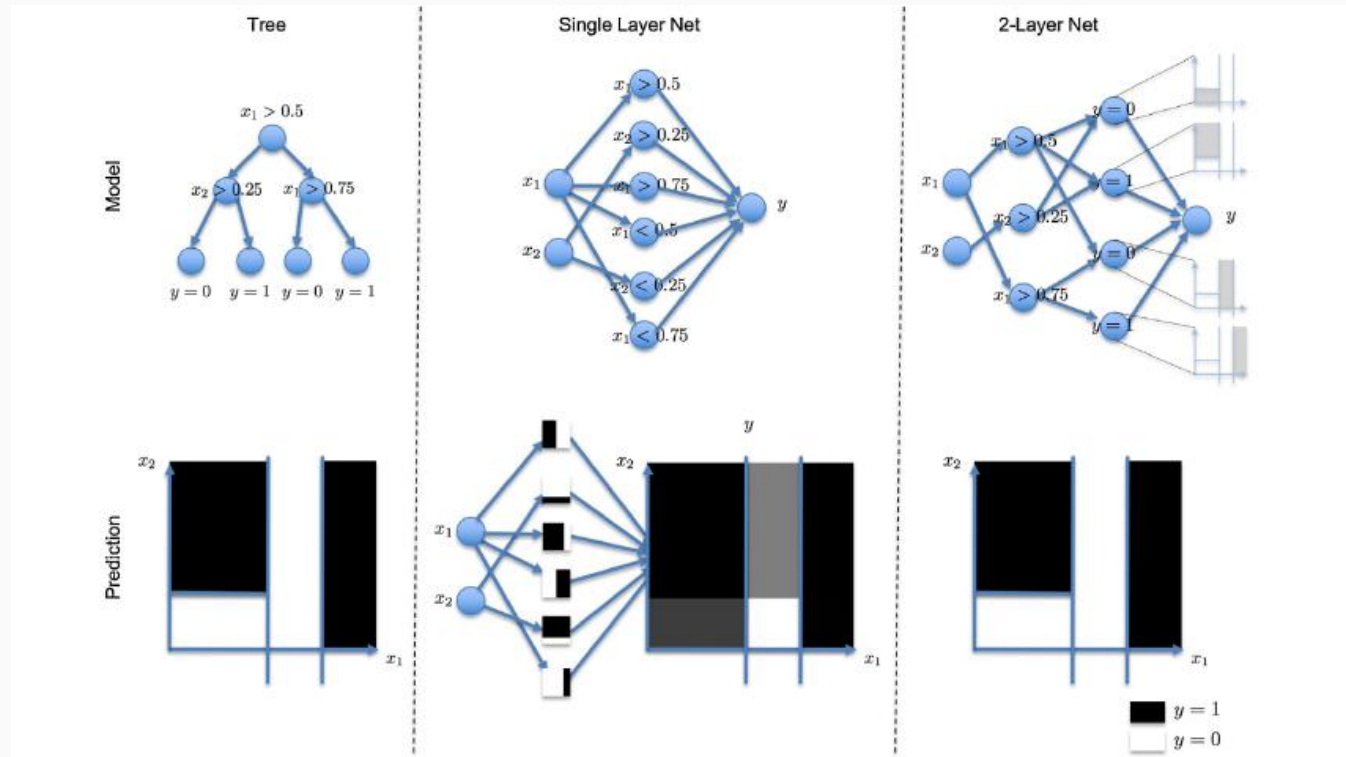


Figure 3. A decision tree allows to describe any partition of space and can thus model any decision boundary. Mapping the tree into a one-layer network is possible. Yet, there still is significant residual error in the resulting function. In the center example, $\epsilon \approx 0.7$. In order to reduce this error further, a higher number of neurons would be required. If we construct a network with one node for every inner node in the first layer and one node for every leaf node in the second layer, we are able to construct a network that results in $\epsilon = 0$.

Como escolher o tamanho de uma camada escondida?

- Podemos escolher a dimensionalidade (o número de nós) da camada oculta.
- Quanto mais nós colocarmos na camada oculta, mais complexas podem ser as funções aproximadas.
- Mas a maior dimensionalidade tem um custo.
 - Primeiro, mais computação é necessária para fazer previsões e aprender os parâmetros da rede.
 - Um número maior de parâmetros também significa que nos tornamos mais propensos a super-ajustar aos nossos dados.

Como escolher o tamanho de uma camada escondida?

- Como escolher o tamanho da camada oculta?
- O melhor é guiar-se pela prática.
- A escolha depende do seu problema específico e é mais uma arte do que uma ciência.
- Vamos explorar o número de nós na camada oculta nas aulas práticas e ver como isso afeta nossa saída.