Aula16

DATA SCIENCE IPT

TURMA 02

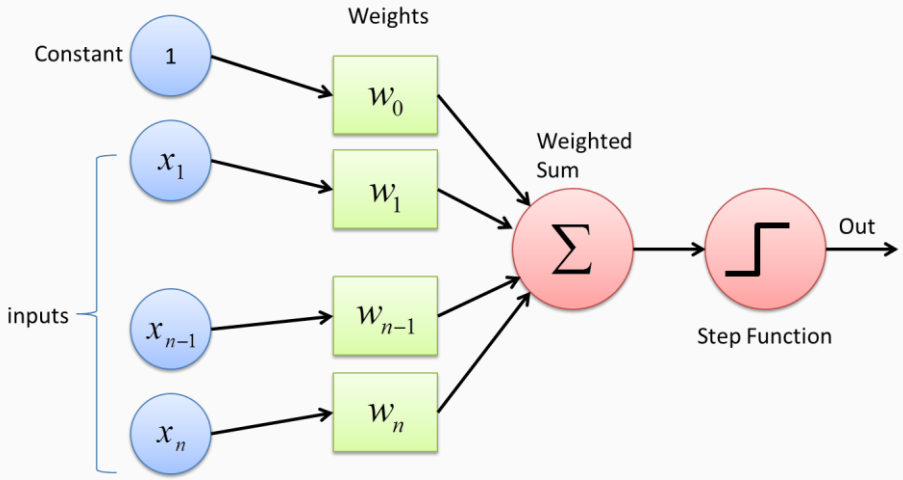
LINKED EDUCATION

V2019

Artificial Neural Netwok

Conteúdo

Vamos, diretamente apresentar o “neurônio artificial”, o **PERCEPTRON**

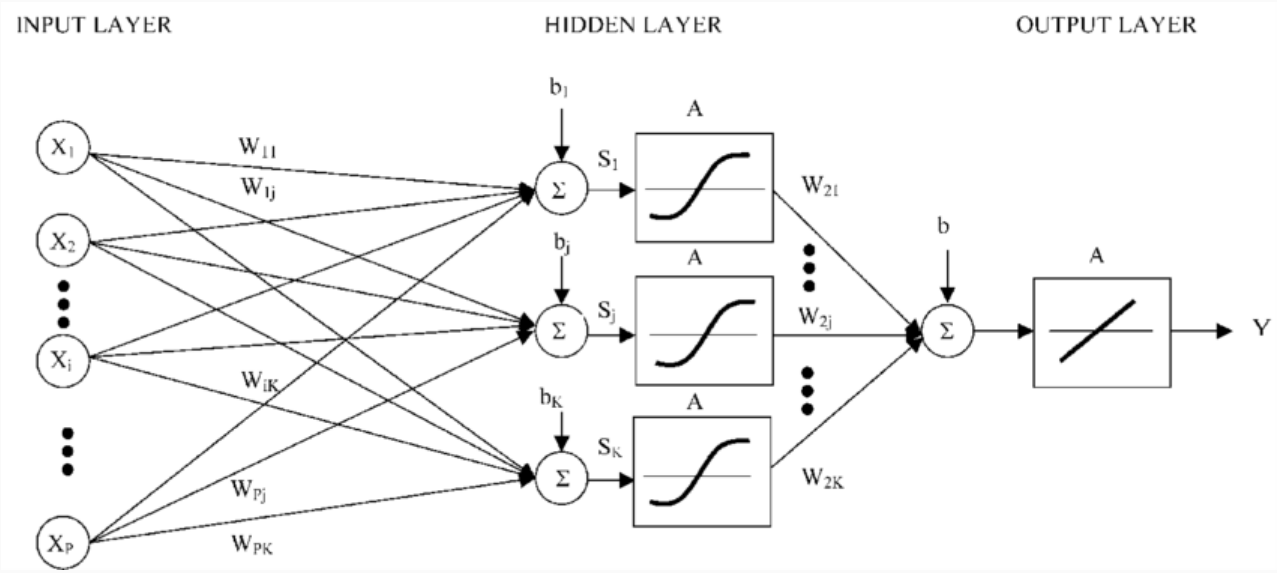


O perceptron recebe entradas 1,x_1,x_2..... Faz uma soma ponderada com os pesos (w0, w_1..) e depois aplica uma “função de ativação”, que “empurra” a saída para uma faixa determinada. Por exemplo, se usarmos como função de ativação a sigmoid, “empurraremos” a saída para a faixa (0,1)...isso parece regressão logística? **sim!**

Artificial Neural Netwok

Conteúdo

Podemos “injetar” a saída de um perceptron na entrada de outro, formando uma **REDE NEURAL.** Na ANN da figura abaixo, há 4 perceptrons, 3 na hidden layer e um na output layer. Como há mais de uma camada de perceptrons, temos uma MLP (multi layer perceptron)

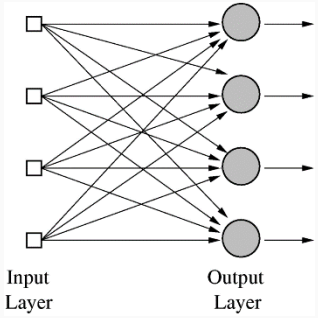


Artificial Neural Netwok

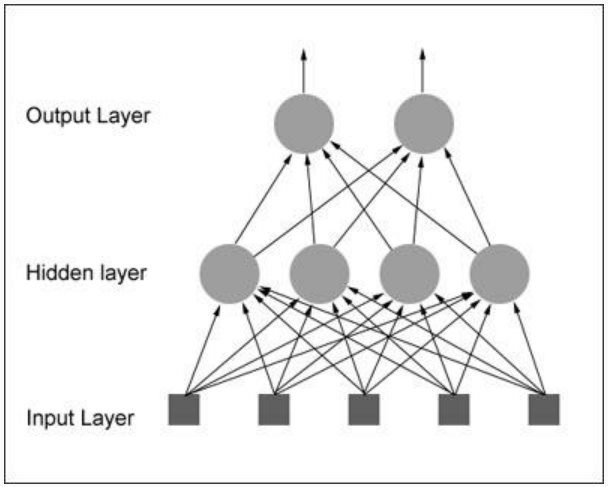
Conteúdo

Na combinação dos perceptrons em uma rede neural, o arranjo mais clássico é a Feed-Forward :

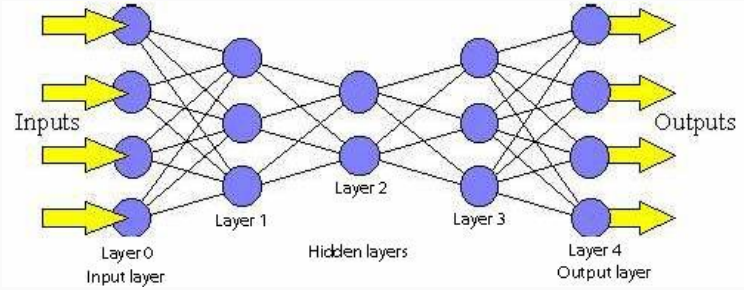
Feed-Forward Neural Networks



Single layer



Multi Layer Perceptron
MLP (Clássica : 1 hidden layer)



Multi Layer Perceptron
MLP (“deep” : >1 hidden layer)

Modelo mais usual de rede neural
Treinável via backpropagation
Pode ser ou não fully connected (todos os perceptrons de camadas sucessivas são ligados entre si)

Artificial Neural Netwok

Conteúdo

Partindo do código ann1.ipynb, obtenha os modelos de ann (classifier, com sklearn) que acertem 100% as funções AND, OR e 75% para XOR (por que?)

Depois monte uma ann (ann2) com 3 perceptrons na hidden layer para XOR (poderiam ser 2)

Finalmente, obtenha os pesos de ann2 e implemente-a diretamente (sem predict, com threshold 0.5)

Como o sklearn obteve os pesos da rede neural ann2 do slide anterior? Vamos entender o algoritmo backpropagation.

Backpropagation

O Backpropagation foi inventado nos anos 70. Porém, só em 1986 Rumelhart, Hinton Williams publicaram “*Learning Representations by Back-Propagating Errors," que a área de Redes Neurais percebeu a importância dele, que continua até hoje.

* https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf

Backpropagation

Conteúdo

Parte 1 : preparando a notação

w_{ij}^{k} = peso do perceptron j da camada k, correspondente à entrada do nó i

a_{j}^{k} = somatória de pesos*inputs +bias do perceptron j da camada k.

o_{j}^{k} = saída do perceptron j da camada k.

g(z) = função de ativação dos perceptrons da hidden layer

$g_{o}(z)$ = função de ativação dos perceptrons da output layer

n_{k}=número de perceptrons da camada k

Backpropagation

Conteúdo

Parte 2 : usando a notação em definições

$a_j^k = \sum_{i=0}^{n_{k-1}} w_{ij}^k \, o_i^{k-1}$

$o_j^k = g(a_j^k)$

E=Erro médio quadrático = $(1/2m)\sum_{i=1}^{m}(ye^i - y^i)^2$

w_{0j}^k= bias (b)
m=número de amostras
ye = y estimado
y = y da amostra (real)

Backpropagation

Parte 3 : Queremos o Gradiente de E em função de cada w_{ij}^{k} !

$$\frac{dE}{dw_{ij}^{k}} = (1/m)\sum_{n=1}^{m} \frac{dE^{n}}{dw_{ij}^{k}}$$

A derivada parcial de E em relação a um peso w, é a “média” das derivadas parciais do erro de cada amostra em relação ao peso w.

Backpropagation

Conteúdo

Parte 4 : O velho truque da regra da cadeia

$$\frac{dE}{dw_{ij}^k} = \frac{dE}{da_j^k}\frac{da_j^k}{w_{ij}^k}$$

O termo $\frac{dE}{da_j^k}$ é usualmente chamado erro e tem o símbolo : δ_j^k

$$\frac{da_j^k}{w_{ij}^k} = \frac{d(\sum_{i=0}^{n_{k-1}} w_{ij}^k o_i^{k-1})}{dw_{ij}^k} = o_i^{k-1}$$

Assim, $\frac{dE}{dw_{ij}^k} = \delta_j^k \; o_i^{k-1}$

Omitimos o “i” de cada amostra...

Backpropagation

Conteúdo

Parte 5 : Calculando $\frac{dE}{dw_{ij}^k}$ na output layer

$\frac{dE}{dw_{ij}^k} = \delta_j^k \; o_i^{k-1}$ $\quad \delta_j^k = \frac{dE}{da_j^k}$

Na output layer,

$\delta_j^k = \frac{dE}{da_j^k} = \frac{d(1/2\ (go(ajk)-y)2)}{da_j^k}$

Regra da cadeia novamente

$\delta_j^k = (go(ajk)-y).\ g_o'(ajk)$

$\frac{dE}{dw_{ij}^k} = (go(ajk)-y).\ g_o'(ajk).\ o_i^{k-1}$

Backpropagation

Conteúdo

Parte 6 : Calculando $\frac{dE}{dw_{ij}^{k}}$ na hidden layer

$\frac{dE}{dw_{ij}^{k}} = \delta_j^k \; o_i^{k-1} \quad \delta_j^k = \frac{dE}{da_j^k}$

Na hidden layer, para fugirmos de fazer diretamente

$\delta_j^k = \frac{dE}{da_j^k}$..... vamos apelar mais uma vez para a regra da cadeia e utilizar a camada abaixo...

$$\frac{dE}{da_j^k} = \sum_{l=1}^{n_k+1} \frac{dE}{da_l^{k+1}} \frac{da_l^{k+1}}{da_j^k} = \sum_{l=1}^{n_k+1} \delta_l^{k+1} \cdot \frac{da_l^{k+1}}{da_j^k}$$

Backpropagation

Conteúdo

Ainda a Parte 6 : Calculando $\frac{dE}{dw_{ij}^{k}}$ na hidden layer

$$\frac{dE}{da_j^k}=\sum_{l=1}^{n_k+1}\frac{dE}{da_l^{k+1}}\,\frac{da_l^{k+1}}{da_j^k}=\sum_{l=1}^{n_k+1}\delta_l^{k+1}.\frac{da_l^{k+1}}{da_j^k}$$

$a_l^{k+1} = \sum_{j=0}^{n_k} w_{jl}^{k+1}\, g(a_j^k)$ .... regra da cadeia ...

$$\frac{da_l^{k+1}}{da_j^k}= w_{jl}^{k+1}\, g'(a_j^k)$$

Assim, para a hidden layer :

$$\delta_j^k = g'(a_j^k)\,.\,\sum_{l=1}^{n_k+1}\delta_l^{k+1}.\,w_{jl}^{k+1} \quad \text{e}$$

$$\frac{dE}{dw_{ij}^{k}}= g'(a_j^k)\,.\,o_i^{k-1}\ \sum_{l=1}^{n_k+1}\delta_l^{k+1}.\,w_{jl}^{k+1}$$

Backpropagation

Conteúdo

Parte 7 :

Observe que, para a output layer :

$\frac{dE}{dw_{ij}^{k}}$=(go(a_j^k)−y). $g_o'(a_j^k)$. o_i^{k-1}= δ_j^k . o_i^{k-1}

E, para as hidden layers...

$\frac{dE}{dw_{ij}^{k}}= g'(a_j^k)$. o_i^{k-1} $\sum_{l=1}^{n_k+1} \delta_l^{k+1}$. w_{jl}^{k+1}

Ou seja, as hidden layers dependem de δ_l^{k+1}...começamos calculando as derivadas parciais pela output layer e, com esse resultado, avançamos (voltando) para as hidden layers..daí o nome backpropagation..os erros...vão sendo propagados da saída para a entrada.

Backpropagation

Conteúdo

Parte 8 :

Para o caso de função **ativação sigmoid nas hidden layers** e um **único perceptron de ativação linear (identidade)** na output layer :

Com função ativação linear, $g_o'(a_j^k)=1$

$\delta_j^k = (go(a_j^k) - y) = (a_j^k - y)$

$\frac{dE}{dw_{ij}^k} = (a_j^k - y).\, o_i^{k-1} = \delta_j^k \,.\, o_i^{k-1}$

E, para as hidden layers... $g'(a_j^k) = g(a_j^k)\,.(1- g(a_j^k))$

$\frac{dE}{dw_{ij}^k} = g(a_j^k)\,.(1- g(a_j^k)).\, o_i^{k-1} \sum_{l=1}^{n_k+1} \delta_l^{k+1}.\, w_{jl}^{k+1}$

$\frac{dE}{dw_{ij}^k} = o_j^k\,.(1- o_j^k).\, o_i^{k-1} \sum_{l=1}^{n_k+1} \delta_l^{k+1}.\, w_{jl}^{k+1}$

Backpropagation

Conteúdo

Resumo : Como treinar a rede neural com backpropagation

1) Inicializar os pesos
2) Para todos os elementos da amostra:
a) Calcular outputs (forward)
b) Calcular erros da camada output (δ_j^k) e usá-los no cálculo das derivadas parciais em relação aos pesos da camada output
c) Usando os valores dos erros da camada output, propagá-los (backward) para as hidden layers e usá-los nos cálculos das derivadas parciais em relação aos pesos das hidden layers
3)Fazer a média dos valores das derivadas obtidos com cada amostra
4)Usar Gradient Descent para atualizar os pesos :

w_{ij}^k (novo)= w_{ij}^k (antigo)- alpha*$\frac{dE}{dw_{ij}^k}$

Mais sobre backpropagation

Utilizando a planilha ANN-backprop.xlsm, vamos acompanhar o algoritmo backpropagation para uma ANN com 2 perceptrons sigmoid na Camada hidden e 1 linear na output. A ideia é a rede aprender a função lógica XOR.

Ann na avaliação do consumo do carro

Vamos voltar ao dataset de carros e criar um modelo de redes neurais feed-forward para avaliar o consumo (regressão).

Parta de ann-carro-custo_res.ipynb

Defina a hidden layer

Evolua o modelo

Cursos com Alta Performance de Aprendizado