Aula17

DATA SCIENCE IPT

LINKED EDUCATION

V2019

Revisão do 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

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

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

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

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

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

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

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

Conteúdo

Obtivemos (planilha ANN-backprop.xlsm, da aula 16), **para a primeira amostra**, para o peso 1 (o primeiro depois do bias) do perceptron único do output

$\frac{dE}{dw^3_{11}}$ = 0,573030328 ...o que isso significa?

0,573030328 ...o que isso significa?

$\lim_{\Delta x \to 0} \frac{E(w^3{}_{11}+\Delta x)-E(w^3{}_{11})}{\Delta x}$ = 0,573030328

Ou seja, é como o erro varia com as variações de $w^3{}_{11}$

Na planilha ANN-backprop , $w^3{}_{11}$ começa com 0.9 , vamos aumentá-lo em 0.1 (uma aproximação para Δx => 0) e observar a variação do erro. Faremos isso com a primeira amostra (0,0)..próximo slide

Mais sobre backpropagation

Conteúdo

Vamos calcular o erro médio quadrático na primeiro “forward” do backpropagation...

ye	y	(ye-y)^2
1,09153	0	1,191438
1,276786	1	0,07661
1,253328	1	0,064175
1,41948	0	2,014923

Para essas 4 amostras, a média é 0.418 (com w_{11}^3=0.9)

Quando passamos w_{11}^3 para 1.0:

ye	y	(ye-y)^2
1,14	0	1,2996
1,336	1	0,112896
1,32	1	0,1024
1,49	0	2,2201

A média do erro passa para 0.4668

Mais sobre backpropagation

Conteúdo

Como $\frac{dE}{dw_{11}^3}$ é 0.486 e sabemos que com w_{11}^3=0.9 o erro médio é 0.418, podemos estimar o erro médio com w_{11}^3 =1.0

$$\frac{dE}{dw_{11}^3} = \lim_{\Delta x \to 0} \frac{E({w^3}_{11} + \Delta x) - E({w^3}_{11})}{\Delta x}$$

Para pequenos valores de Δx , podemos aproximar

$E({w^3}_{11} + \Delta x)$= $\frac{dE}{dw_{11}^3}$* Δx+ $E({w^3}_{11})$.....vamos testar?

0.486*0.1+0.418=0.466≃0.4668....Bingo!

Mais sobre backpropagation

No slide anterior sabíamos o erro com $w_{11}^3=0.9$ e conseguimos Estimar o erro com $w_{11}^3=1.0$. Desta forma, sabemos o que fazer (aumentar ou diminuir um peso) para minimizar o erro aos poucos.

Assim, o backpropagation nos dá $\frac{dE}{dw_{ij}^k}$ e com essas derivadas, vamos calibrando os pesos via Gradient Descent...

Mais sobre backpropagation

Conteúdo

Fazendo uma analogia, cada peso de cada perceptron terá um “botão de ajuste”

$\frac{dE}{dw_{ij}^{k}}$ Indica como variará o custo quanto alterarmos cada peso.

Tendo, para cada peso, $\frac{dE}{dw_{ij}^{k}}$ poderemos mexer em cada um deles (mexer nos botões...gradiente descent) para reduzirmos o custo.

Para obtermos $\frac{dE}{dw_{ij}^{k}}$ de cada peso, temos que “alimentar” a rede com um input, obter os outputs e, depois, efetuar o backpropagation. Os $\frac{dE}{dw_{ij}^{k}}$ são obtidos do output para o input (backpropagation).

A atualização de cada peso é feito com Gradient Descent, por exemplo, já que teremos o gradiente (todas as derivadas parciais em relação aos pesos).

Mais sobre backpropagation

Conteúdo

Com base em backpropag.ipynb :

Atividade 1 : analisar código

Atividade 2: simular planilha (iterações=1, pesos originais) e verificar resultados

Atividade 3: gerar pesos randômicos, 20000 iterações, learning Rate=0.2...obter output final

Atividade 4 : criar função custo (erro médio quadrático)
e traçar gráfico : iterações x custo

ANN com imagens

Conteúdo

Com base em
ANN_Digitos.ipynb :

Classificação de Imagem de dígitos usando rede MLP

Atividade 1 : analisar o código

Atividade 2 : mostrar a quantidade de amostras por classe

Atividade 3 : Obter a acurácia na amostra toda

Atividade 4 : Fazer split 70 30 e calcular acurácia na
amostra de testes
mostre as imagens (a imagem mesmo) com predição errada no split

Atividade 5 : Alterar parâmetros da MLP visando melhorar
acurácia na amostra de testes

Cursos com Alta Performance de Aprendizado