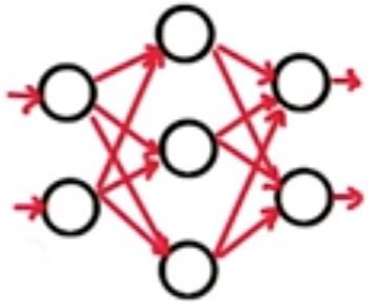
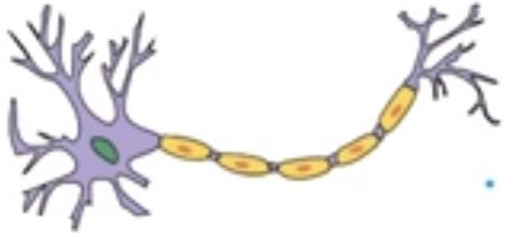
Redes Neurais Artificiais - 4



NEURONS?

HOW THE BRAIN WORKS?



NEUROMORPHIC ENGINEERING?

Perceptron Multicamadas - MPL

Backpropagation

Agora chegamos ao problema de como fazer uma rede neural de múltiplas camadas *aprender*. Antes, vimos como atualizar os pesos usando o gradiente descendente. O algoritmo de Retropropagação (backpropagation daqui em diante) é apenas uma extensão disso, usando a regra de cadeia para encontrar o erro respeitando os pesos conectando a camada de input para a camada oculta (numa rede de duas camadas).

Para atualizar os pesos das camadas ocultas usando o gradiente descendente, é necessário saber quanto contribuiu cada unidade oculta para a produção daquele erro no output final. Uma vez que o output de uma camada é determinado pelos pesos entre camadas, o erro resultante de unidades é proporcional aos pesos ao longo da rede. Uma vez que sabemos o erro no output, nós usamos os pesos para trazê-lo de volta às camadas ocultas.

Perceptron Multicamadas - MPL

Cont… Backpropagation

Por exemplo, na camada de outuput, temos erros δ_k^o atribuídos para cada unidade de output k. Então, o erro atribuído para a unidade oculta j é igual aos erros dos outputs, proporcional aos pesos entre as camadas oculta e de output (levando o gradiente em conta):

$$\delta_j^h = \sum W_{jk} \delta_k^o f'(h_j)$$

Então, o passo do gradiente descendente é o mesmo que antes, apenas com os novos erros:

$$\Delta w_{ij} = \eta \delta_j^h x_i$$

onde w_{ij} são os pesos entre o inputs e a camada oculta e x_i são os valores de input da unidade. Esse formato continua válido para qualquer quantidade de camadas. O passos de peso são iguais ao tamanho do passo vezes o erro de output da camada vezes os valores de input daquela camada

$$\Delta w_{pq} = \eta \delta_{output} V_{in}$$

Aqui, temos o erro de output, δ_{output} , ao propagar os erros retrospectivamente de camadas mais altas. E os valores de input, V_{in} são os inputs da camada, as ativações da camada oculta para camada de output, por exemplo.

Perceptron Multicamadas - MPL

Exemplo no material auxiliar...

Implementação com Numpy

Você já possui a maior parte das coisas necessárias para implementar a backpropagation com o Numpy.

No entanto, antes só devíamos lidar com erros de uma unidade. Agora, na atualização dos pesos, temos que considerar o erro de *cada unidade* da camada oculta, δ_j: $\Delta w_{ij} = \eta \delta_j x_i$

Tarefa 5 - Backpropagation

Exercício de Backpropagation

Abaixo, você implementará o código para calcular uma rodada de atualização com backpropagation para dois conjuntos de pesos. Escrevi o andamento para frente, o seu objetivo é escrever o andamento para trás.

Coisas a fazer

- Calcular o erro da rede.
- Calcular o gradiente de erro da camada de output.
- Usar a backpropagation para calcular o erro da camada oculta.
- Calcular o passo de atualização dos pesos.

Perceptron Multicamadas - MPL

Implementando backpropagation

Agra que vimos que o erro da camada de output é

$$\delta_k = (y_k - \hat{y}_k) f'(a_k)$$

e que o erro da camada oculta é

$$\delta_j = \sum [w_{jk} \delta_k] f'(h_j)$$

Por enquanto iremos contar com uma rede simples com uma camada oculta e uma unidade de output. Eis o algoritmo geral para atualizar os pesos com backpropagation:

- Defina os passos de atualização dos pesos para cada camada como zero
  - Os pesos do input para oculta $\Delta w_{ij} = 0$
  - Os pesos da oculta para o output $\Delta W_j = 0$
- Para cada observação dos dados de treinamento:
  - Faça um andamento adiante pela rede, calculando o output $\hat{y}$
  - Calcule o gradiente de erro da unidade de output, $\delta^o = (y - \hat{y}) f'(z)$ em que $z = \sum_j W_j a_j$, o input da unidade de output.
  - Propague os erros para a camada oculta $\delta_j^h = \delta^o W_j f'(h_j)$
  - Atualize os passos dos erros:
    - \Delta W_j = \Delta W_j + \delta^o a_j
    - \Delta w_{ij} = \Delta w_{ij} + \delta^h_j a_i
- Atualize os pesos, onde η é a taxa de aprendizado e m é o número de observações:
  - W_j = W_j + \eta \Delta W_j / m
  - w_{ij} = w_{ij} + \eta \Delta w_{ij} / m
- Repita por e épocas.

Tarefa 6 - Backpropagation_2

Implementando o Backpropagation

Agora você implementará o algoritmo de backpropagation em uma rede treinada com os dados de admissão em universidades. Você possui tudo o que precisa para resolver esse exercício nos exercícios anteriores.

Seus objetivos aqui:

- Implementar o andamento adiante.
- Implementar o algoritmo de backpropagation.
- Atualizar os pesos.