Relatório 13 - Prática: Redes Neurais (II)

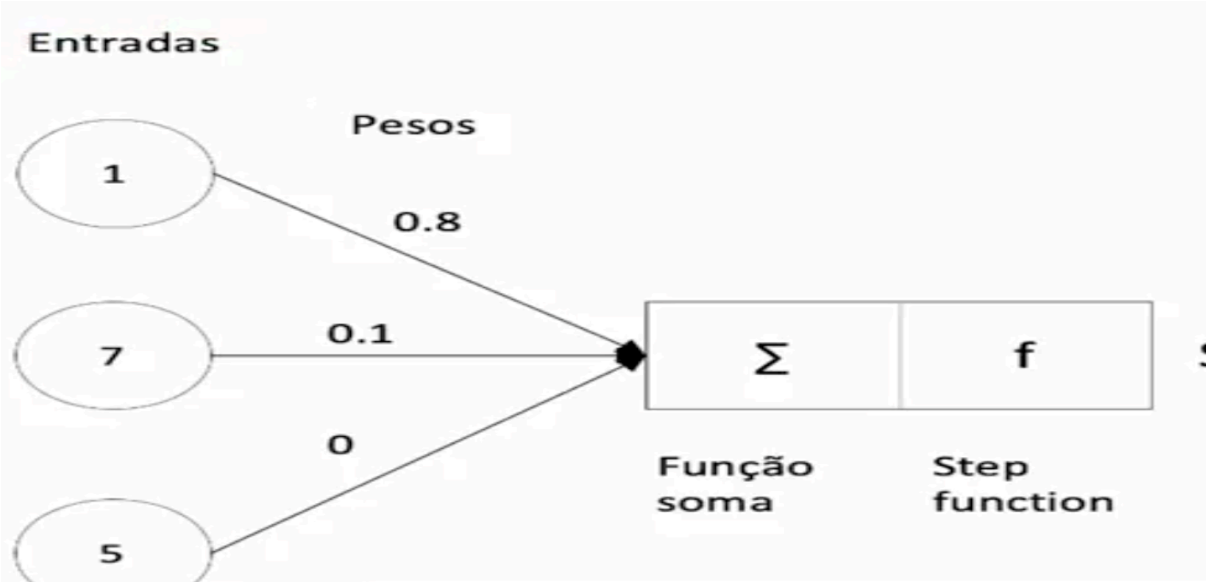
Higor Miller Grassi

Na segunda seção temos a introdução de qual conteúdo será mostrado no restante do curso sobre Deep learning.

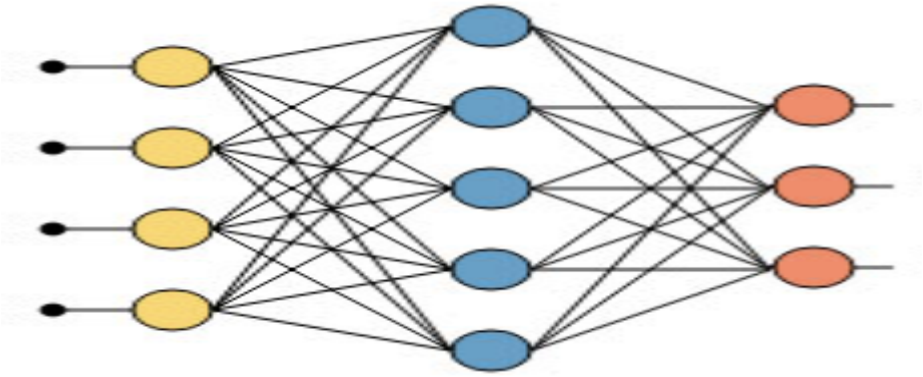
Terceira seção:

Temos a explicação dos cálculos envolvidos nos ajustes de pesos para realizar o Aprendizado. Tendo em vista as teorias apresentada a seguir:

- **Rede de uma Camada:** É Uma rede neural onde tem as entradas(sendo valores escolhidos para cada necessidade), os pesos onde vai ser multiplicado e por final temos a função soma onde realiza os cálculos anteriormente citados e o step function(retorna 1 ou 0) que de acordo com a especificação, agindo como uma aplicação de função de ativação.



- Redes multicamadas: Diferentemente da rede de uma camada, a multicamadas tem uma camada a mais(camada oculta) tendo também a função de soma e ativação juntamente a eles, sendo cada dado de entrada conectado em cada neurônio da camada oculta e os mesmos são cada um conectados a camada de saída.



-Tratamento de Erro: Depois de realizar o cálculo e obtivemos o resultado de ativação, em primeira instância os resultados reais vai ser diferente do resultado da rede neural, para tratarmos o ocorrido, fazemos a resposta esperada menos o resultado obtido, tendo o erro(apenas o valor absoluto, em módulo), e por fim, obtemos também a média absoluta considerando os pesos que realizamos nos cálculos, a fim de sempre tentar diminuir esta média para ser o mais preciso possível, mudando os pesos.

- Descida do Gradiente(atualização de pesos): Basicamente devemos calcular a derivada parcial para conseguir mover para a direção do gradiente, cada vez que a média de erro diminui após os ajustes de peso, mais chegamos na parte inferior do gráfico(melhor solução/erro menor/menor custo)
Na primeira fórmula, temos o sigmoid(formula utilizada como exemplo, sendo uma das mais usadas), já na segunda forma temos o cálculo da derivada para indicar a direção para onde os pesos deverão ser atualizados

$$y = \frac{1}{1 + e^{-x}} \quad \rightarrow \quad d = y * (1 - y)$$

-**Cálculo do Delta(obter o gradiente):** É feito para chegar no mínimo global, sendo obtido o delta para a camada de saída com a multiplicação do erro com a derivada da função sigmoid, tendo que ser repetido para cada um dos registros, já o delta da camada escondida e cálculo na multiplicação da derivada com o peso e o delta de saída, sendo aplicado também para cada registro

-Ajuste dos pesos(BackPropagation): Primeiramente, o cálculo é feito igual citado, juntamente com a soma e a camada de aplicação, após isso voltamos para a camada de entrada, calculando a direita para esquerda e depois da esquerda para direita.
Nesta seguinte forma primeiro cálculos a entrada*delta para cada peso em todos os registros igualmente e depois realizar o restante do cálculo, sendo o resultado qual deve ser a manipulação do ajuste de pesos, sendo realizado o processo novamente pela quantidade de época anteriormente definidas.

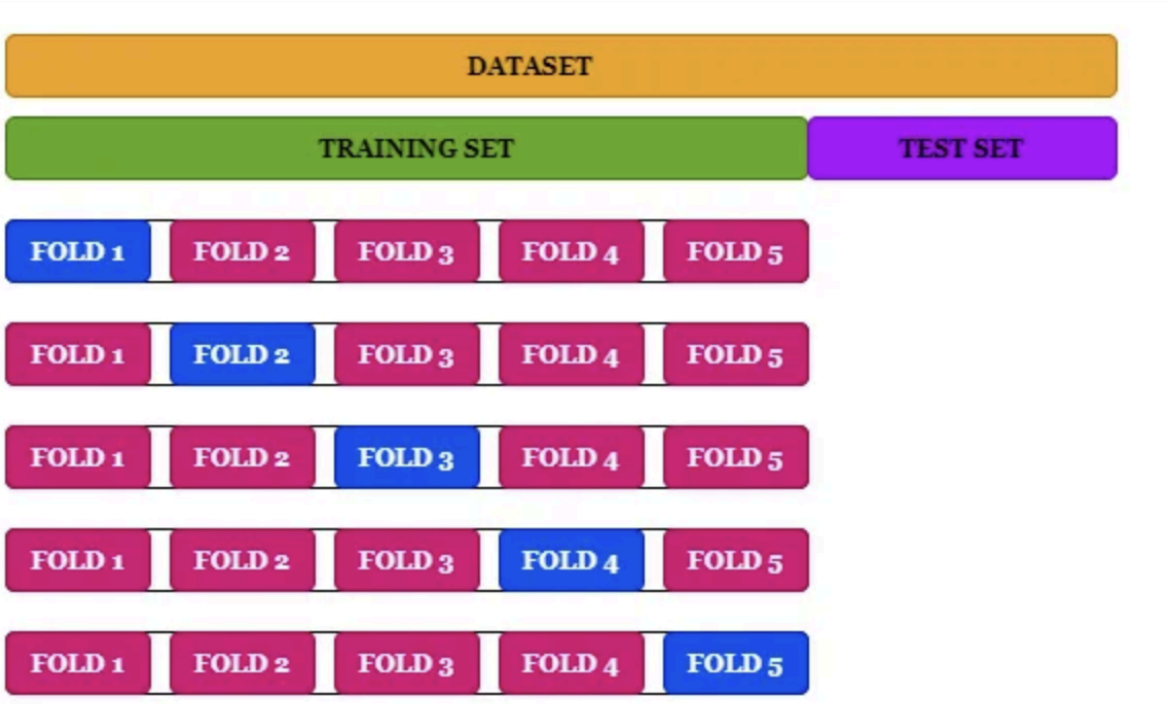
$$peso_{n+1} = (peso_n * momento) + (\boldsymbol{entrada} * \boldsymbol{delta} * taxa\ de\ aprendizagem)$$

-BIAS: Utilizado para melhorar os resultados de uma rede neural, permite ao modelo deslocar a função de ativação, possibilitando maior flexibilidade na modelagem de dados complexos, agindo como uma constante adicionada a soma ponderada das entradas antes de ser passada por uma função de ativação, sendo adicionada transparentemente através de bibliotecas, sem necessidade de configurar parametros especificos.

Quarta Seção

-**Mean Squared Error(MSE) e o Root Mean Squared Error(RMSE)**: realizando o cálculo do erro e elevando ao quadrado, assim facilitando encontrar onde está mais grave, e depois tirando a média do mesmo, a única diferença dos dois é que RMSE após o cálculo da média, ele faz a raiz quadrada, assim tendo um valor maior e penalizando mais os erros.

-**K-Fold(Validação Cruzada):** Divide o conjunto de dados em K partes (folds) e é treinado K vezes, utilizando K-1 folds para treinamento e o fold restante para validação, esse processo é repetido para cada fold, e a média dos resultados obtidos é calculada.
Por exemplo, caso seja escolhido o k como 5, ele usa as bolinhas preenchidas para treinamento e a restante para realizar o teste, para saber a accuracy é so fazer o somatório de cada divisão (normalmente usa-se o k valendo 10)



-Underfitting x Overfitting: Não adianta utilizarmos um complexo para resolver problemas simples, do mesmo jeito que algoritmos simples não resolvem problemas mais complexos. Assim, o overfitting se molda tanto aos dados de treinamento que quando surgem novos dados ele não consegue ser tão preciso, e o underfitting é modelo não consegue aprender adequadamente a relação entre os dados de entrada, tendo um resultado ruim aos dados de treinamento

Quinta Seção

Na quinta secao mexemos com o banco de dados de iris(flores), assim tendo que definir a camada de saida como 3, sendo uma para cada especie. Fizemos o treinamento comum e tambem com a validacao cruzada(k-fold).

Sexta Seção

Na sexta secao ja mexemos com um banco de dados diferente, o de games, onde tivemos que alterar e dropar tabelas que nao seriam necessarias para a rede neural, assim como antes, mexemos com varios dados e desta vez tivemos 3 saidas diferentes, cada uma sendo um local distinto do mundo e