1.3

O método de *gradient descent* é usado para minimizar o risco empírico que consiste em , de forma a adaptar os pesos ao conjunto de treino. Neste caso será usado o método de gradient descest com fixed step size, momentum term e o treino será feito em modo *batch*, por isso as equações que expressam a evolução dos pesos são as seguintes:

Onde corresponde ao peso do ramo com nó inical i e final j, (i=0 para DC)

Para calular o nas equações acima é necessário primeiro fazer na rede uma propagação da rede para a frente de forma a obtermos os e posteriormente uma propagação para trás (backpropagation) obtendo os dando nos todos os dados necessários para adaptar os pesos consoante o conjunto de treino.

StepSize = 0.9

Momentum term = 0.5

Mediana dos epoch =235

1.4)

A expressão permanece igual à representada na alínea 1.3 com uma pequena adição, o é agora modificado a cada iteração segundo as seguintes equações:

Isto fará com que o valor inicial de seja menos relevante pois este é modificado a cada iteração. O uso de adaptive step sizes permite então adaptação do segundo características locais permitindo geralmente uma convergência mais rápida.

Recalcular epochs <-params do prof? 87

Os valores encontrados que minimizam o número de épocas são 0.9 para o learning rate e 0.7 para o momentum term que correspondem a XXX epochs.

Para o mesmo momentum term () devido à robustez deste algoritmo, ao variar o learning rate o número de epoch é praticamente inalterado.

No caso em que não é usado passo adaptativo ocorrem grandes alteração o número de epochs ao variar minimamente os parâmetros, isto deve-se simplemeste ao facto de o não se alterar consoante a região da função de custo que está a operar.

Assim concluí-se que ao usar passo adaptativo, o valor inicial de tem pouca importância e que o algoritmo vai diminuir o erro quadrático mínimo mais rapidamente.

Ao variar o momentum term () DIZER O QUE ACONTECE BOY PARA FIXED E ADAPTIVE

Ao variar o eta DIZER O QUE ACONTECE BOY PARA FIXED

Matriz confusao

A percentagem geral com que a rede identifica corretamente os dígitos de entrada é cerca de 73% o que é satisfatório.

É importante referir que para dígitos diferentes obtêm-se percentagens diferentes, sendo que a causa destes erros advém das similaridades entre dígitos de números diferentes, por exemplo o dígito 8 é facilmente confundido com o 9 devido às suas partes superiores serem semelhantes ficando com taxas de erro da à volta de 60% e o 1 com o 7 devido a ambos terem regiões retilíneas.

É também relevante apontar que as percentagens de accuracy variam ao correr outro treino com os mesmos conjunto de treino e de teste devido a os pesos iniciais serem escolhidos aleatoriamente.

Training error <- Corresponde ao MSE final? Probably not= recalcular 0.05492

Test accuraccy 73.3%

Ambos os parâmetros são relevantes no entanto é mais fácil e intuitivo de averiguar a qualidade da rede através do test set accuracy, pois nos é dito logo qual a percentagem de sucesso que é mais simples de compreender do que o *Mean squared error*.

ISTO FAZ SEQUER SENTIDO?:

Além disso pode ocorrer overfitting, se fizer quisermos aumentar a accurancy pendindo um MSE mais baixo poderá causar overfitting, que poderá causar que a rede seja menos flexível e porudza um test set accuracy mais baixo.

Os pesos são escolhidos de forma a obter um *MSE* mínimo para o grupo de validação, para fazer tal coisa é avaliado em cada *epoch*, qual o *MSE* do conjunto de validação escolhendo os pesos para o qual este é menor. O objetivo deste processo é verificar se os pesos escolhidos para a rede são válidos para um conjunto de validação que corresponde em padrões que a rede não tinha visto, de forma a entender-se qual o comportamente da rede com dados novos.

Através deste passo de validação é também possível prevenir que haja overfitting pois os pesos não serão escolhidos segundo MSE do conjunto de dados com que a rede treina mas sim com os dados com que esta é validada.

No plot é obervado como evolui a cada iteração o erro quadrado médio do conjunto de treino, de validação e de teste.

O *MSE* do conjunto de treino converge para o goal *MSE* definido, o validation error de cada época indica qual o pesos que devemos utilizar para a posterior utilização da rede que não corresponderão aos pesos da iteração onde se chega ao goal.

Finalmente pode se observar que para as últimas épocas erro do conjunto de treino aumenta o que se deve à existencia de overfitting.

O QUE ISTO QUER DIZER (VERIFICAR NO PLOT SE MSE DO TRAINING SET AUMENTA NO FIM )

, podemos ver que para as ultimas época dos training set os MSE correspondentes do test e validation estão a aumentar, o que é indicio de overfitting da rede.

SECALHAR POR O PLOT

Como se pode observar no gráfico à esquerda a rede faz um bom fitting do conjunto de treino, e também ao conjunto de teste o que demonstra que os pesos escolhidps foram adequados. É possível observar que o efeito de overfitting não é notável pois a função não tenta passar por todos os pontos de treino e por isso não existem picos desnecessários ao longo da função.

Por fim compreende-se que apesar de o fitting ser bom a função continua a afastar-se da função original em algumas zonas.

Não usando o validation set os pesos que a rede vais escolher no final serão os que minimizam o erro de treino o que vai forçar a função a passar pelos pontos do conjunto de treino ficando muito adaptadas a estes e gerando overfitting desviando-se da função que queremos efetivamente aproximar . Assim a põr outros inputs na rede como por exemplos as coordenada x do conjunto de teste, o output da rede afaste se bastante da coordenada y para esses pontos ou seja há um MSE maior para este conjunto. Assim concluí-se que é importante ter um *validation set* para impedir que situações destas ocorram.

SE O GOAL DO ERRO FOSSE MENOR HAVERIA MENOS OVERFITTING?

Sim existe overfitting pois a função tenta passar por todos os pontos de treino, causando que existam picos que se afastam muito da função inicial que se queria obter.