Diferença entre um lote e uma época em uma rede neural

por **Jason Brownle**

A descida gradiente estocástica é um algoritmo de aprendizagem que possui vários hiperparâmetros.

Dois hiperparâmetros que muitas vezes confundem os iniciantes são o tamanho do lote e o número de épocas. Ambos são valores inteiros e parecem fazer a mesma coisa.

Descida Gradiente Estocástica

Stochastic Gradient Descent, ou SGD, para abreviar, é um algoritmo de otimização usado para treinar algoritmos de aprendizado de máquina, principalmente redes neurais artific ais usadas em aprendizado profundo.

A tarefa do algoritmo é encontrar um conjunto de parâmetros internos do modelo que tenham um bom desempenho em relação a alguma medida de desempenho, como perda logarítmica ou erro quadrático médio.

A otim zação é um tipo de processo de busca e você pode pensar nessa busca como um aprendizado. O algoritmo de otimização é denominado " *gradiente descendente* ", onde " *gradiente* " se refere ao cálculo de um gradiente de erro ou inclinação do erro e "descida" se refere ao mov mento descendente ao longo dessa inclinação em direção a algum nível mínimo de erro.

O algoritmo é iterativo. Isso sign f ca que o processo de pesqu sa ocorre em várias etapas discretas, cada etapa melhorando ligeiramente os parâmetros do modelo.

Cada etapa envolve usar o modelo com o conjunto atual de parâmetros internos para fazer previsões em algumas amostras, comparar as previsões com os resultados reais esperados, calcular o erro e usar o erro para atualizar os parâmetros internos do modelo.

Este proced mento de atualização é diferente para algoritmos diferentes, mas no caso de redes neurais artificiais, o algoritmo de atualização de retropropagação é usado.

Antes de nos aprofundarmos em lotes e épocas, vamos dar uma olhada no que queremos d zer com amostra.

Saiba mais sobre descida gradiente aqui:

- Descida grad ente para aprendizado de máquina

Ele contém entradas que são inseridas no algoritmo e uma saída que é usada para comparar com a previsão e calcular um erro.

Um conjunto de dados de treinamento é composto por muitas linhas de dados, por exemplo, muitas amostras. Uma amostra também pode ser chamada de instância, observação, vetor de entrada ou vetor de características.

Agora que sabemos o que é uma amostra, vamos definir um lote.

O que é um lote?

O tamanho do lote é um hiperparâmetro que define o número de amostras a serem trabalhadas antes de atualizar os parâmetros internos do modelo.

Pense em um lote como um loop for iterando uma ou mais amostras e fazendo previsões. No final do lote, as previsões são comparadas com as variáveis de saída esperadas e um erro é calculado. A partir deste erro, o algoritmo de atualização é usado para melhorar o modelo, por exemplo, descer ao longo do gradiente de erro.

Um conjunto de dados de treinamento pode ser dividido em um ou mais lotes.

Quando todas as amostras de treinamento são usadas para criar um lote, o algoritmo de aprendizado é chamado de gradiente descendente em lote. Quando o lote tem o tamanho de uma amostra, o algoritmo de aprendizagem é chamado de gradiente descendente estocástico. Quando o tamanho do lote é maior que uma amostra e menor que o tamanho do conjunto de dados de treinamento, o algoritmo de aprendizagem é chamado de descida gradiente de minilote.

- **Descida gradiente em lote** . Tamanho do lote = tamanho do conjunto de treinamento
- **Descida Gradiente Estocástica** . Tamanho do lote = 1
- **Descida gradiente de minilote** . 1 <Tamanho do lote <Tamanho do conjunto de treinamento

No caso de descida gradiente de minilote, os tamanhos de lote populares incluem 32, 64 e 128 amostras. Você pode ver esses valores usados em modelos na literatura e em tutoriais.

E se o conjunto de dados não for dividido igualmente pelo tamanho do lote?

Isso pode acontecer e acontece frequentemente ao treinar um modelo. Significa simplesmente que o lote final tem menos amostras do que os outros lotes.

Como alternativa, você pode remover algumas amostras do conjunto de dados ou alterar o tamanho do lote de forma que o número de amostras no conjunto de dados seja dividido igualmente pelo tamanho do lote.

Para saber mais sobre as diferenças entre essas variações de descida gradiente, veja a postagem:

- Uma introdução suave ao gradiente descendente de minilotes e como configurar o tamanho do lote

Para saber mais sobre o efeito do tamanho do lote no processo de aprendizagem, veja a postagem:

- Como controlar a velocidade e a estabilidade do tamanho do lote de redes neurais de treinamento

Um lote envolve uma atualização do modelo usando amostras; a seguir, vejamos uma época.

O número de épocas é um hiperparâmetro que define o número de vezes que o algoritmo de aprendizagem funcionará em todo o conjunto de dados de treinamento.

Uma época significa que cada amostra no conjunto de dados de treinamento teve a oportunidade de atualizar os parâmetros internos do modelo. Uma época é composta por um ou mais lotes. Por exemplo, como acima, uma época que possui um lote é chamada de algoritmo de aprendizado de gradiente descendente em lote.

Você pode pensar em um loop for sobre o número de épocas em que cada loop prossegue no conjunto de dados de treinamento. Dentro deste loop for há outro loop for aninhado que itera sobre cada lote de amostras, onde um lote tem o número de amostras de "tamanho de lote" especificado.

O número de épocas é tradicionalmente grande, muitas vezes centenas ou milhares, permitindo que o algoritmo de aprendizagem seja executado até que o erro do modelo seja suficientemente minimizado. Você pode ver exemplos do número de épocas na literatura e em tutoriais definidos como 10, 100, 500, 1000 e maiores.

É comum criar gráficos de linhas que mostram épocas ao longo do eixo x como tempo e o erro ou habilidade do modelo no eixo y. Esses gráficos às vezes são chamados de curvas de aprendizado. Esses gráficos podem ajudar a diagnosticar se o modelo aprendeu demais, aprendeu pouco ou se está adequadamente ajustado ao conjunto de dados de treinamento.

Para mais informações sobre diagnósticos via curvas de aprendizado com redes LSTM, veja o post:

- Uma introdução suave às curvas de aprendizagem para diagnosticar o desempenho do modelo

Caso ainda não esteja claro, vejamos as diferenças entre lotes e épocas.

Qual é a diferença entre lote e época?

O tamanho do lote é um número de amostras processadas antes da atualização do modelo.

O número de épocas é o número de passagens completas pelo conjunto de dados de treinamento.

O tamanho de um lote deve ser maior ou igual a um e menor ou igual ao número de amostras no conjunto de dados de treinamento.

O número de épocas pode ser definido como um valor inteiro entre um e infinito. Você pode executar o algoritmo pelo tempo que desejar e até mesmo interrompê-lo usando outros critérios além de um número fixo de épocas, como uma alteração (ou falta de alteração) no erro do modelo ao longo do tempo.

Ambos são valores inteiros e hiperparâmetros para o algoritmo de aprendizagem, por exemplo, parâmetros para o processo de aprendizagem, e não parâmetros do modelo interno encontrados pelo processo de aprendizagem.

Você deve especificar o tamanho do lote e o número de épocas para um algoritmo de aprendizado.

Não existem regras mágicas sobre como configurar esses parâmetros. Você deve tentar valores diferentes e ver o que funciona melhor para o seu problema.

Assuming I have a dataset of 50,000 points.

The following parameters are set in Python/Keras as.
batch_size = 64
iterations = 50
epoch = 35

So, my assumption on what the code is doing is as follows:

50,000 samples will be divided by the batch size (=781.25 =~ 781).
So now I have 64 blocks (batches) of the whole dataset, with each containing 781 samples.

For iteration 1:
All of the blocks from 1 to 64 will be passed through the model. Each block/batch resulting in its own accuracy metric, resulting in 64 accuracy numbers that will be averaged at the end.
This above process will be repeated 35 times (the number of Epochs) resulting in 35 averaged accuracies, and as the Epoch increases along the iteration, the accuracy is (theoretically) going to be better than the previous accuracy.
After the iteration is done, the weights of the nodes will be updated and be used for iteration 2.

The above process will be repeated 50 times, as I have 50 iterations.