## 질문

마지막 예제mnist에서 처음 부분에 data import 하는 부분이 있습니다.

datat를 load하고 바로 x\_train를 print해보면 정수로된 array 입니다. 이를 0~1사이값으로 scale를 해야하는 이유가 무엇인가요? data을 변환하고 train 하는 이유가 있을것 같아서 문의드립니다. 어떤 모델에서든 항상 0~1값으로 변경해야하는것인지?

## 답변

이론적인 면에서는 Normalize는 Neural Network의 Weight, Bias가 조정되는 것으로 모두 잡힐 수 있어야 하지만, Practical 하게는 1. training more stable and faster 2. reduce the chances of getting stuck in local optima 3. 학습과 예측에 사용되는 Domain 데이터 사이의 샘플링 Population 일치 문제 (out of scale from one another)를 해결 하기 위한 것으로 알려져 있습니다.

1번, 2번의 경우를 보다 자세히 얘기드리자면, gradient descent를 할때는 gradient error vectors와 learning rate를 곱해서 weight를 조정해 주는데, input feature들 간의 distribution 차이가 나면, 한쪽 feature에 대해서는 잘 작동하는 learning rate가 다른 feature에 대해서는 under compensating하는 문제가 있을 수 있습니다. (since the learning rate in the update equation of Stochastic Gradient Descent is the same for every parameter) (Elkan, Charles. (2008) Log-linear models and conditional random fields)

이런 이유로 error surface에서 Global minimum을 찾기 위해서는 구모양에 가까운 input feature를 가지는 것이 타원형 구조 (한쪽 feature의 range가 훨씬 큰 경우)보다 빠르게 converge 하는 것으로 알려져 있습니다.

3번 같은 경우에 조금 더 쉬운 예를 들자면, 주식/코인 같은 경우에는 Price를 그대로 입력하는 것이 아니라, Return, Log Return을 Scaling의 목적으로 사용할 때가 있는데, 학습할때의 Price 값 (ex: KOSPI: 2000, BTC: $3000)의 기준과 예측할 때(ex: KOSPI: 2300, BTC: $7000)의 Price 값의 기준이 크게 차이가 나는 문제를 해결하기 위해서 사용합니다. (같은 input feature이지만, 시기가 달라서 학습에 관찰되지 않은 데이터가 입력 될 수 있음, out of scale from one another)

또한, 가격에 대한 입력과, Bid/Ask Volume에 대한 입력은 Scale 차이가 상당한데 이러할 경우에, 모든 feature에 기반하여 모델이 작동하기 보다는, Range가 보다 큰 particular한 feature에 모델이 종속되어 버리는 것 같습니다. (ex. 모델 내에서 distance metric이 계산될 경우 값의 범위가 큰 feature에 종속)

그리고, 마지막으로 0~1 값으로 변경해야 할지, 다른 옵션 (Mean Normalization, Standardization, Scaling to unit length)를 할지, 혹은 Scailing이 하지 않을지 등에 대한 Feature Scaling 선택 문제가 있을 수 있는데, 풀고자 하는 문제의 특성에 따라서 다른 Scaling 방법을 취해주는 것이 좋을 것 같습니다. 가량 Exact Boundary가 있는 문제의 경우에는 Standardization이 적절하지 않을 수도 있고, min-max scaling은 long tail 분포가 있는 문제에 대해서는 적절하지 않을 수 있습니다. 혹시 이 외에도 추가로 궁금하신 점이 있으시면 언제든지 질문 부탁드립니다.

## 질문

원본 data의 각 몇%로 x\_train, x\_test에 각각 할당되었는지요? 할당도 황금(?) 비율이 있는지요? 그리고 예를 들어 data가 충분하지 않을때 Train에 많이 할당하면 좋을텐데, 이런 경우를 위해서 할당량을 조절할 방법이 있나요?

질문 2번에 대해서는, 현재 x\_train.shape = (60000, 28, 28) / x\_test.shape = (10000, 28, 28) 으로 train은 60000 개 test는 10000개로 쪼개져 있습니다.

train / test split의 목적은 generalization error 를 최소화 하기 위함인데, 둘 사이의 샘플 개수 결정은 경쟁 관계에 있습니다. 만약 train 사이즈가 작으면 paramter estimate의 variance 가 높고, 반대로 test 사이즈가 작으면 performace statistic의 variance가 높아집니다. 따라서, 양쪽의 variance가 너무 높지 않도록 설정해야하는데, 이는 sample size에 따라서 그 크기를 결정하는 방식에 차이가 날 것 같습니다.

예를들어 100개의 sample에 대해서 train / test 를 쪼갤때는 어떤 방식을 취해도 parameter estimate variance를 줄이기가 쉽지가 않을 것이고, 100000개의 sample을 train / test 를 쪼갤때는 0.8:0.2 를 취하거나, 0.9:0.1 을 취하든 generalization에 문제가 없을 것으로 보입니다. 어떤 황금률을 제시하기는 어렵지만, variance를 줄이기 위한 방법은 아래와 같은 방법이 있습니다.

1. training 데이터를 또 다시 train / validation으로 쪼개는 방법

2. train을 비율 (90, 80, 70) 등을 바꿔가면서 각 비율당 여러차래 (ex. 20회 이상) random sampling 하여 mean performance를 비교하여 lower variance를 선택하는 방법

저는 약식으로 할때는 보통 70/30 혹은 80/20을 선호합니다.

이론적인 부분을 조금 더 살펴보고 싶으시면 아래 논문을 추천드립니다.

<https://pdfs.semanticscholar.org/452e/6c05d46e061290fefff8b46d0ff161998677.pdf>

## 질문

Build Model부분이 마치 black box 같아 보입니다.

옛날 코딩하던 습성이 남아서 그런지, 그때 그때 변수들의 변화되는 과정을 print 해서 이해도를 높이고 논리적 문제가 없는지 파악하곤 합니다. 하여 모델 학습에도 이런식의 중간 중간 파악해볼만한 방법이 있을까요?

없다면.. ㅠ.ㅠ. 학습 다 시키고 Evaluation의 accuracy까지 보고 낮으면 다시 학습하고 이렇게 하면 비 효율적일것 같은데...(Data셑이 많아서 모델 학습하는데 한시간 이상 걸리는경우.. ㅠ.ㅠ.) 어떤 노하우가 있으신지 궁금합니다.

## 답변

질문 3번에 대해서 답변드리자면, neural net의 디버깅은 원인이 너무나 다양할 수 있어서 (exploding gradients, dead relu, vanishing gradient etc) 항상 쉬운 일이 아닌것 같습니다. 저는 주로 model compile 이후에 "model.summary()" 를 통해 전체 shape 구조를 보거나, keras model의 특정 layer를 취득하여 해당 layer의 결과를 확인할 수 있는 function을 만들어 사용합니다. 해당 방법에 대해서는 아래 링크를 참조해 주시면 됩니다.

http://laid.delanover.com/debugging-a-keras-neural-network/

추가적으로 저는 사용 경험은 없지만, 조금 더 조사하여보니 tensorflow의 debugger (tfdbg)를 이용한 방법도 사용되는 것 같습니다 (keras의 자체 debugger는 없음). 해당 방법에 대해서는 공식 문서가 안보여서 아래 일본어로 되어 있는 문서를 첨부드립니다. 혹시 필요하시면 구글 번역을 통하여 따라해 보실 수 있을 것 같습니다.

https://qiita.com/keisuke-nakata/items/5d6918678e8099b565d0

추가적으로 neural network 모델의 일반적인 디버깅 방식은 아래의 링크를 참조해 주시면 좋을 것 같습니다.

https://medium.com/machine-learning-world/how-to-debug-neural-networks-manual-dc2a200f10f2

감사합니다. 그럼 좋은 하루 보내시고, 혹시 추가적으로 궁금하신점이 있으시면 언제든지 질문 바랍니다