tensorflow library

추가 자료

mnist 데이터 셋 처리 순서

- mnist 데이터 셋은 다음과 같은 순서대로 다운로드 하면 된다.

1) input_data 모듈을 로딩한다.
from tensorflow.examples.tutorials.mnist import input_data

2) read_data_sets 함수를 이용하여 mnist 객체를 구한다.

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
  (4)                    (1)              (2)          (3)
```

번호	설명
(1)	하드 디스크에 자동으로 다운로드 해주는 함수이다.
(2)	다운로드 받을 하위 폴더 이름(자동으로 생성이 된다.)
(3)	True이면, y(정답)의 값을 자동으로 one hot 처리해준다.
(4)	mnist 객체의 이름이다.

mnist 데이터 셋 처리 순서

- mnist 데이터 셋

항목	설명
mnist.train	훈련용 데이터 셋
mnist.test	테스트용 데이터 셋
mnist.train.num_examples	훈련용 데이터 셋의 개수(55,000)
mnist.test.images	테스트용 데이터 이미지
mnist.test.labels	테스트용 데이터 라벨(정답)
mnist.train.next_batch(숫자)	전체 데이터 셋에서 지정된 숫자 만큼의 subset 가져 오기

```
# 훈련용/테스트용 모두 xs(이미지), ys(레이블)를 가지고 있다.
# next_batch : 전체 데이터 중에서 subset 가져 오기
batch_xs, batch_ys = mnist.train.next_batch( 1 )
```

붓꽃 데이터

- 머신 러닝 및 통계학의 전 분야에서 사용하는 전통적인 데이터 셋이다.
- 붓꽃의 종류(setosa, virginaca, versicolor)에 대한 꽃받침 길이, 꽃받침 너비, 꽃잎 길이, 꽃잎 너비에 대한 측정치가 들어 있다.
- 각 종류마다 50개의 행, 총 150개의 측정치가 들어 있다.
- 파이썬 패키지의 scikit learn 패키지의 데이터 셋 함수를 이용하여 데이터 셋을 로딩할 수 있다.
- 참조 파일 : iris 데이터 셋.xlsx

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
꽃받침 길이	꽃받침 너비	꽃잎 길이	꽃잎 너비	붓꽃의 종류
5.1	3.5	1.4	0.2	0
4.9	3	1.4	0.2	0
4.7	3.2	1.3	0.2	0
4.6	3.1	1.5	0.2	0
5	3.6	1.4	0.2	0
5.4	3.9	1.7	0.4	0
4.6	3.4	1.4	0.3	0
5	3.4	1.5	0.2	0
4.4	2.9	1.4	0.2	0
4.9	3.1	1.5	0.1	0

```
from sklearn import datasets
iris = datasets.load_iris()

print(len(iris.data))
# 150

print(len(iris.target))
# 150

print(len(iris.target[0]))# 꽃받침 길이, 꽃받침 너비, 꽃잎 길이, 꽃잎 너비
[5.1 3.5 1.4 0.2]

print(set(iris.target)) # 종류(setosa, virginaca, versicolor)
{0, 1, 2}
```

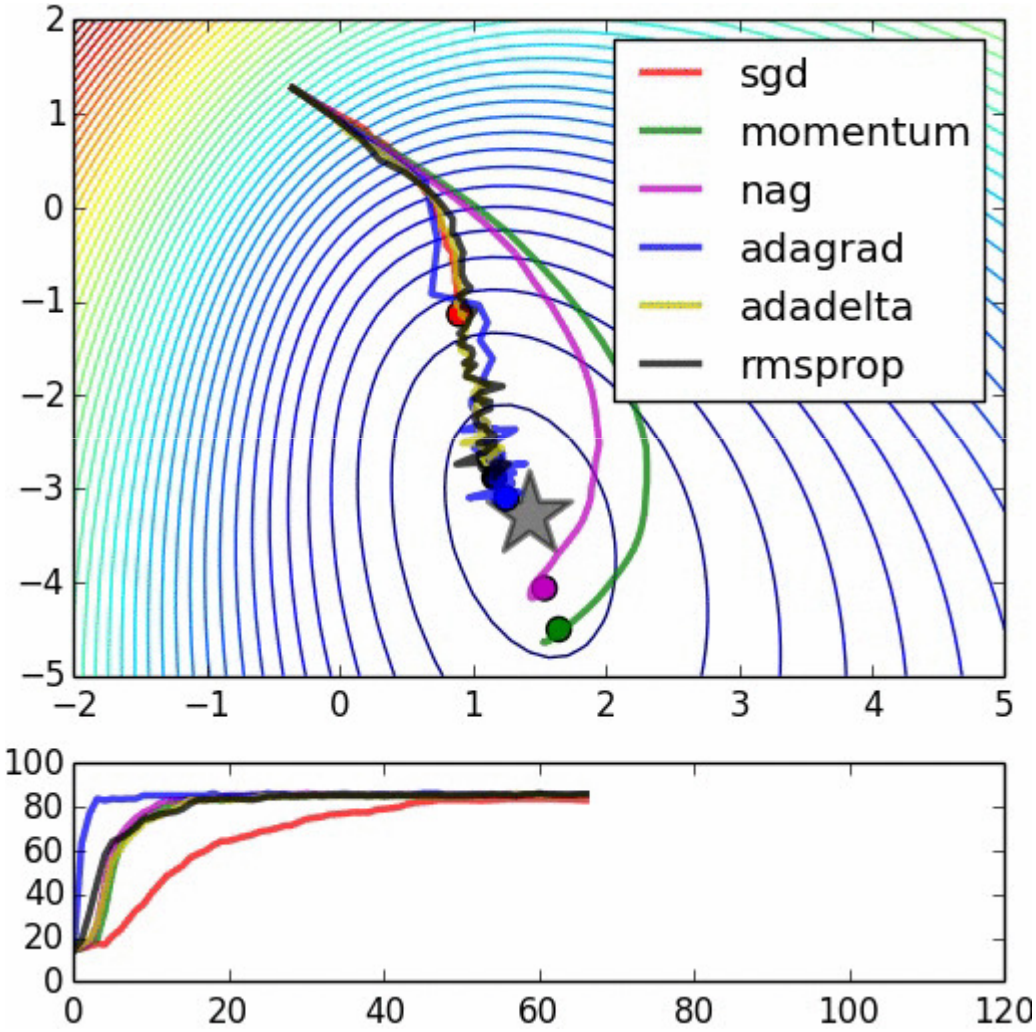
옵티마이저(optimizer)

- 옵티마이저란 최적화를 수행해주는 객체이다.
- 경사 하강법으로 부족해서 많은 옵티마이저를 만들어서 사용하고 있다.

- 참조 사이트
  - http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html
  - https://www.tensorflow.org/api_guides/python/train
  - https://github.com/MagmaTart/DeepLearningStudy/blob/master/Soomin/summarys/21_Optimizer2.md

옵티마이저(optimizer)

- 옵티마이저 비교

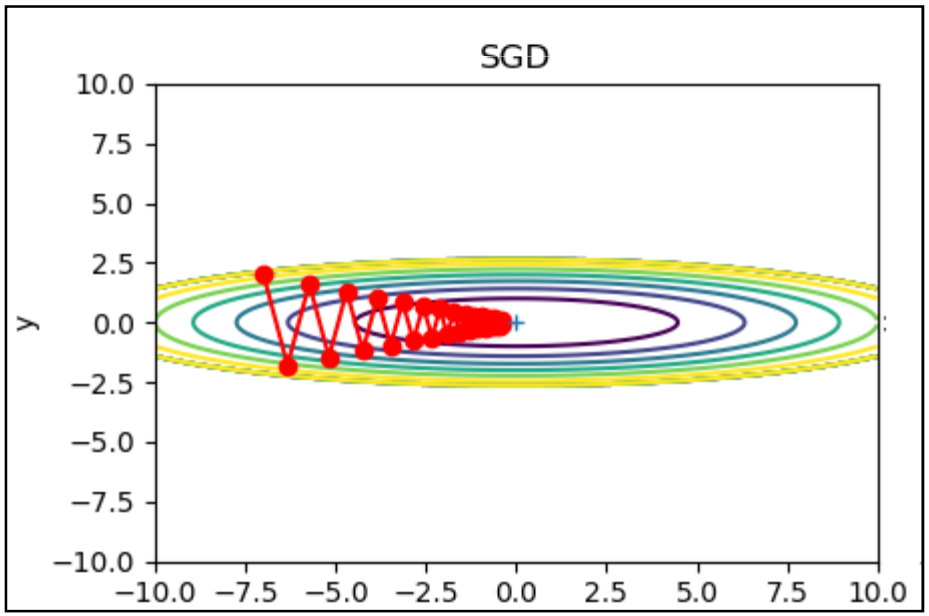


옵티마이저

- 확률적 경사 하강법(SGD)은 가장 크게 기울어진 방향으로 이동하자.
- 기울어진 방향으로 일정한 거리만큼 이동하겠다.는 전략이다.
- 학습율이 크면 많이 이동한다.
- 탐색 경로가 지그 재그(비효율적으로 이동)로 이동한다

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

α : 학습율, $\frac{\partial L}{\partial W}$: w에 대한 손실 함수의 기울기 (미분 계수)

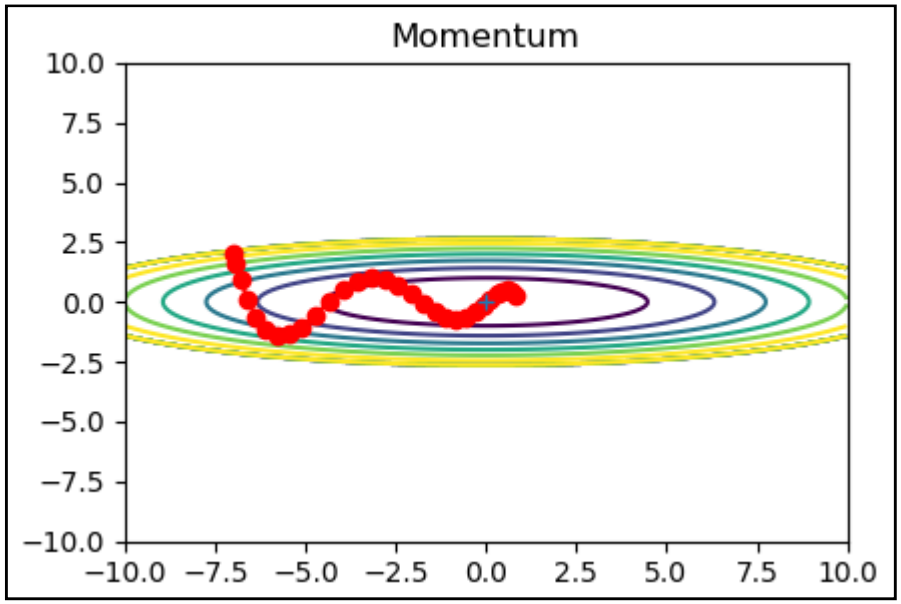


옵티마이저

- 모멘텀(Momemtum)은 물리에서 운동량이라고 한다.
- 기울기 방향으로 힘을 받아서 물체가 가속되는 특징이 있다.
- 관성을 유지하면 쭉 아래로 이동한다.
- 공이 그릇의 곡면을 따라서 굴러 가듯이 움직인다.
- SGD에 비하여 지그 재그가 좀 덜하다.

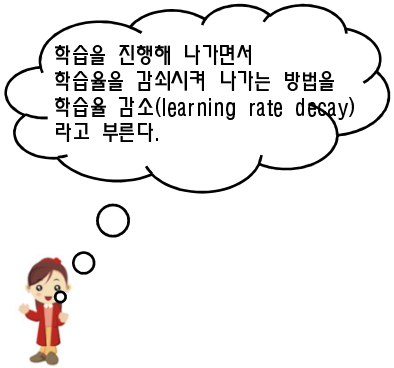
$$W \leftarrow W - \alpha \frac{\partial L}{\partial W} + av$$

av : a는 마찰 계수, v는 속도(velocity)



옵티마이저

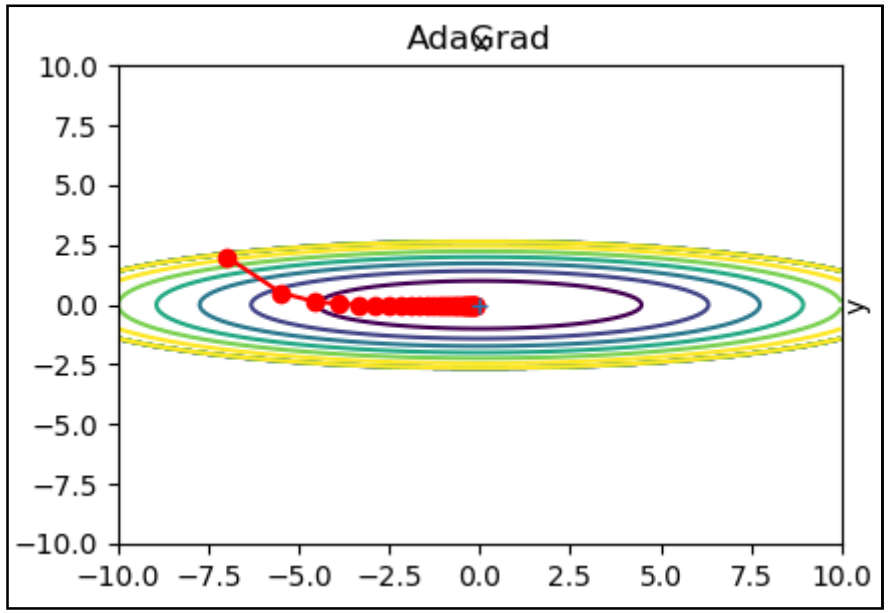
- AdaGrad은 학습을 진행해 나가면서 학습율을 줄여 나가면서 진행하는 방식이다.
- 과거의 기울기를 제곱하여 더해 나가는 방식이다.
- 진행할수록 갱신 강도(strength)는 약해진다.
- 무한 학습을 하게 되면 어느 순간 갱신되는 량이 0이 되는 문제가 있다.



$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$

⊙은 행렬의 원소별 곱셈을 의미한다.

$$W \leftarrow W + \alpha \frac{1}{\sqrt{h}} \cdot \frac{\partial L}{\partial W}$$

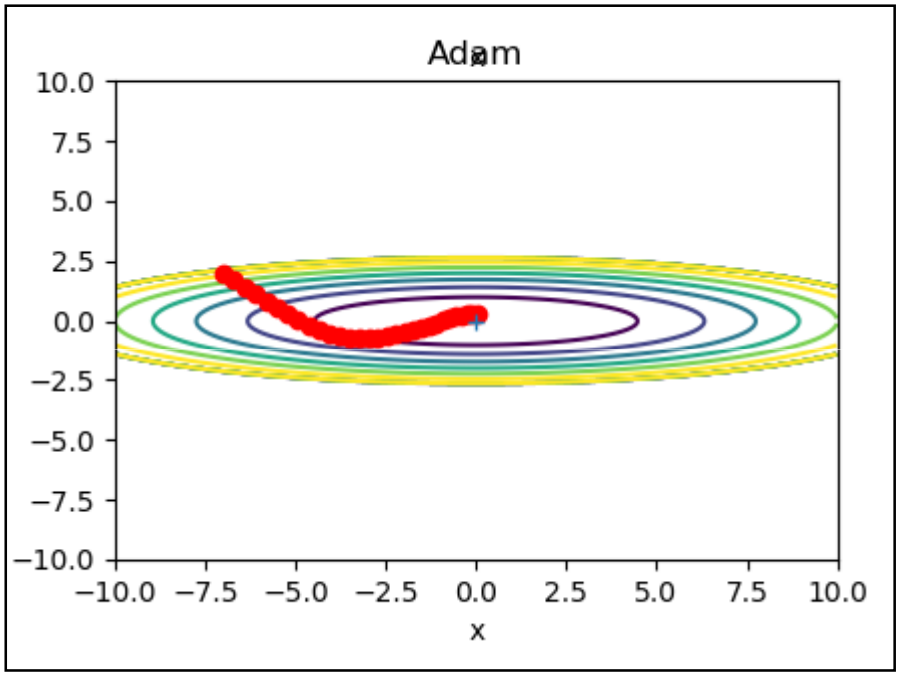


옵티마이저

- AdaGrad는 무한 학습시 갱신되는 량이 0이 되는 문제가 있었다.
- 이것을 보완한 방식이 RMSProp이다.
- 먼 과거의 기울기는 고려하지 않고, 최근의 새로운 기울기 정보만을 반영하는 방식이다.
- 지수 이동 평균(Exponential Moving Average) 방식이라고 한다.

옵티마이저

- Adam 방식은 경사 하강법을 보완한 방식이다.(2015년 제안)
- Adam = Momentum + AdaGrad



옵티마이저(optimizer)

- 어떠한 옵티마이저를 사용할 것인가 ?
- 애석하게도 모든 문제에서 항상 뛰어난 기법이란 것은 없다.
- 학습율, 함수의 수식 등 여러 개의 요소가 복합적이다.