

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

서 상우

목차

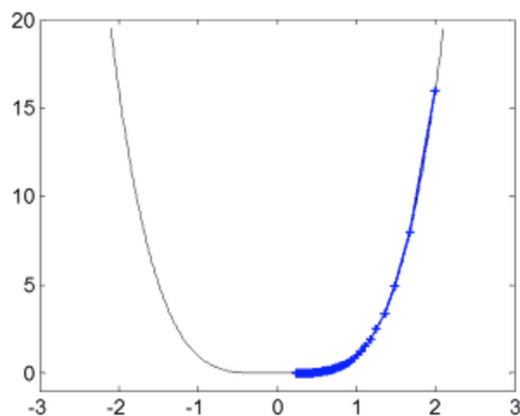
1. optimizer
2. Stochastic Gradient Descent
 1. Momentum
 2. Nesterov Accelerated Gradient (NAG)
 3. Adagrad
 4. RMSProp
3. Adam

1. optimizer

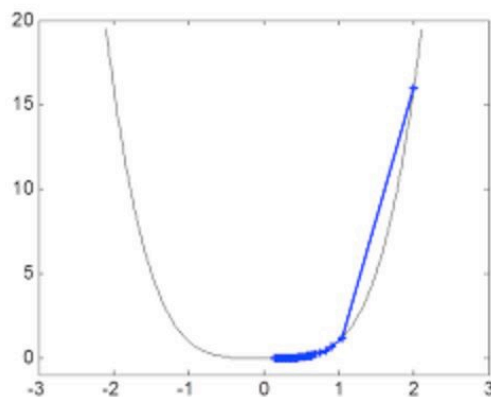
- 신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것
- 이는 곧 최적 매개변수를 찾는 문제
 - 최적화
 - 매개변수 공간은 매우 넓고 복잡하여, 최적 매개변수를 찾는 것은 쉽지 않은 문제
- Gradient descent는 cost function을 최소화하기 위해 이용할 수 있는 방법
- cost function 말고도 각종 optimization에 이용

1. optimizer

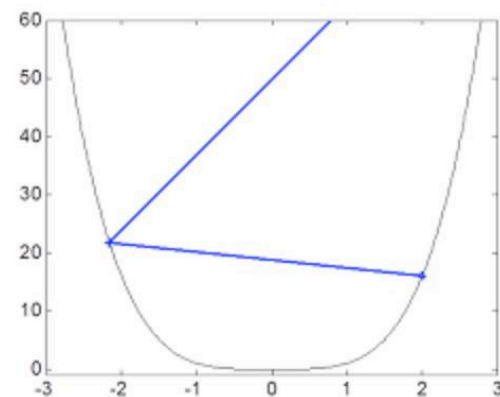
$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta)$$



$$\eta = 0.01$$



$$\eta = 0.03$$



$$\eta = 0.13$$

2.1 Momentum

- Momentum 방식은 말 그대로 Gradient Descent를 통해 이동하는 과정에 일종의 '관성'을 주는 것
- 현재 Gradient를 통해 이동하는 방향과는 별개로, 과거에 이동했던 방식을 기억하면서 그 방향으로 일정 정도를 추가적으로 이동하는 방식

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$$

$$\theta = \theta - v_t$$

v_t 는 time step t에서의 이동 벡터

γ 는 얼마나 momentum을 줄 것인지에 대한 momentum term (0.9 정도의 값)

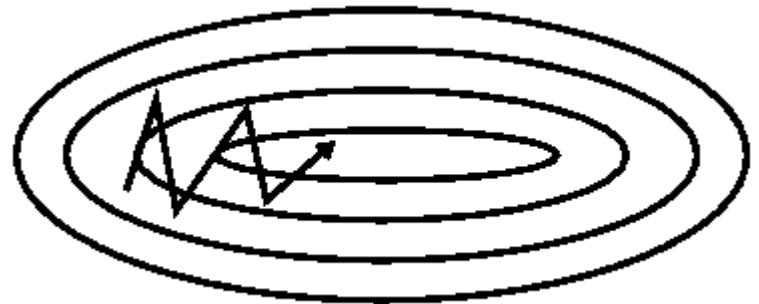
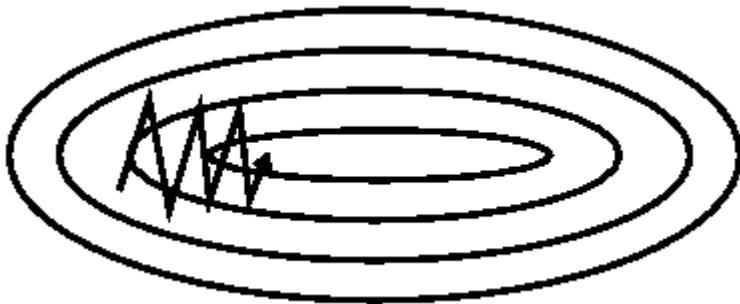
2.1 Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$$

- 식을 살펴보면 과거에 얼마나 이동했는지에 대한 이동 항 v 를 기억한다.
- 새로운 이동항을 구할 경우 과거에 이동했던 정도에 관성항만큼 곱해준 후 Gradient을 이용한 이동 step 항을 더해준다.
- 즉, Gradient들의 지수평균을 이용하여 이동한다고도 해석 가능
- $v_t = \eta \nabla_{\theta} J(\theta)_t + \gamma \eta \nabla_{\theta} J(\theta)_{t-1} + \gamma^2 \eta \nabla_{\theta} J(\theta)_{t-2} + \dots$

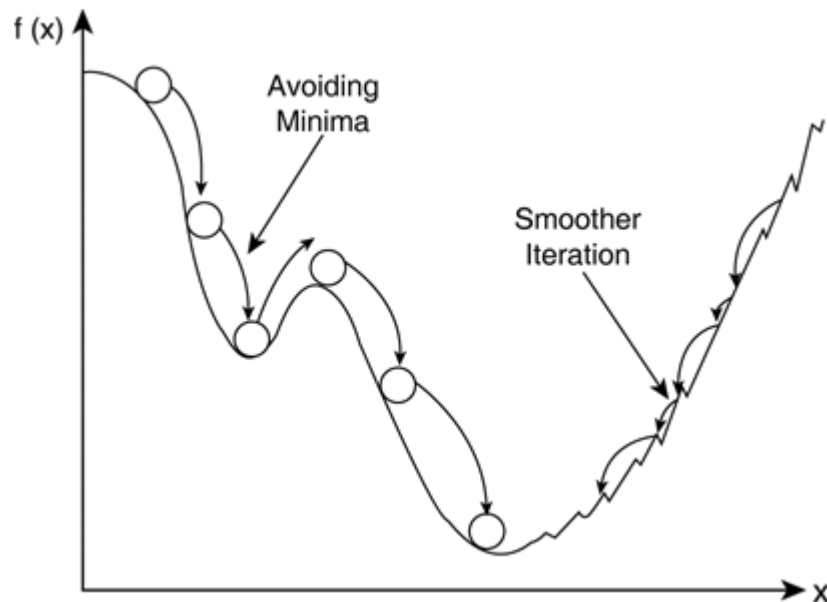
2.1 Momentum

- Momentum 방식은 SGD가 Oscilation 현상을 겪는 상황을 해결.
- SGD는 한번의 step에서 움직일 수 있는 step size는 한계가 있으므로 이러한 oscilation 현상이 일어날 때는 좌우로 계속 진동하면서 이동에 난항을 겪음.



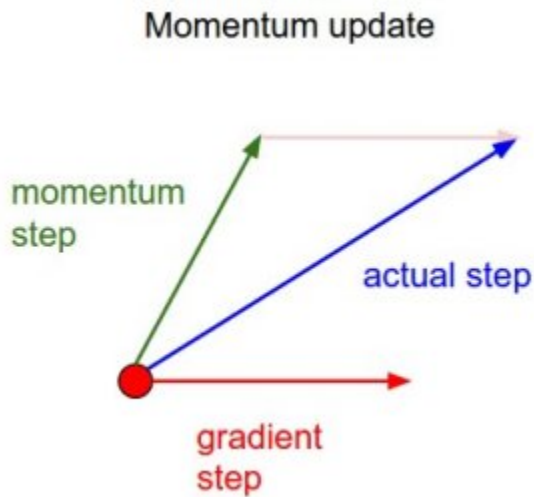
2.1 Momentum

- Momentum 방식을 사용할 경우 다음과 같이 자주 이동하는 방향에 관성이 걸리게 되고, 진동을 하더라도 중앙으로 가는 방향에 힘을 얻기 때문에 SGD에 비해 상대적으로 빠르게 이동 가능.



2.2 Nesterov Accelerated Gradient

- Nesterov Accelerated Gradient (NAG)
 - 1. 관성 방향으로 움직인 후,
 - 2. 움직인 자리에 스텝을 계산해보니 더 빠름



2.3 Adaptive gradient (Adagrad)

- 변수들을 update할 때 각각의 변수마다 step size를 다르게 설정해서 이동하는 방식
- 지금까지 많이 변화하지 않은 변수들은 step size를 크게 하고, 지금까지 많이 변화했던 변수들은 step size를 작게 하자!
- 자주 등장하거나 변화를 많이 한 변수들의 경우 optimum에 가까이 있을 확률이 높기 때문에 작은 크기로 이동하면서 세밀한 값을 조정
- 적게 변화한 변수들은 optimum 값에 도달하기 위해서는 많이 이동해야할 확률이 높기 때문에 먼저 빠르게 loss 값을 줄이는 방향으로 이동

2.3 Adaptive gradient (Adagrad)

- word2vec이나 GloVe 같이 word representation을 학습시킬 경우 단어의 등장 확률에 따라 variable의 사용 비율이 확연하게 차이나기 때문에 Adagrad와 같은 학습 방식을 이용하면 훨씬 더 좋은 성능을 거둘 수 있을 것이다.
- $G_t = G_{t-1} + (\nabla_{\theta} J(\theta_t))^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(G_t + \epsilon)}} \cdot \nabla_{\theta} J(\theta_t)$
- adagrad를 사용하면 학습을 진행하면서 굳이 step size decay등을 신경써주지 않아도 된다는 장점
- G에는 계속 제공한 값을 넣어주기 때문에 G의 값들은 계속해서 증가하기 때문에, 학습이 오래 진행될 경우 step size가 너무 작아져서 결국 거의 움직이지 않게 된다.



2.4 RMSprop


- 앞에서 처럼 보폭을 줄이는 건 좋은데 이전 맥락을 보가며 하자!

- $G_t = G_{t-1} + (\nabla_{\theta} J(\theta_t))^2$  $G_t = \gamma G_{t-1} + (1 - \gamma) \nabla_{\theta} J(\theta_t)^2$

- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla_{\theta} J(\theta_t)$

3. Adam

- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$  Momentum method
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$  RMSprop의 average squared gradient

- $\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}$
 - $\widehat{v}_t = \frac{v_t}{1 - \beta_2^t}$
- 
- m과 v가 처음에 0으로 초기화되어 있기 때문에
학습의 초반부에서는 m_t, v_t 가 0에 가깝게 bias
되어있을 것이라고 판단

=> 바이어스를 줄인다.

- $\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \cdot \widehat{m}_t$

3. Adam

$$E[v_t] = E[g_t^2]$$

$$E[m_t] = E[g_t]$$

$$v_t = (1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} \cdot g_i^2$$

$$\mathbb{E}[v_t] = \mathbb{E} \left[(1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} \cdot g_i^2 \right] \tag{2}$$

$$= \mathbb{E}[g_t^2] \cdot (1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} + \zeta \tag{3}$$

$$= \mathbb{E}[g_t^2] \cdot (1 - \beta_2^t) + \zeta \tag{4}$$

EXPERIMENT: LOGISTIC REGRESSION

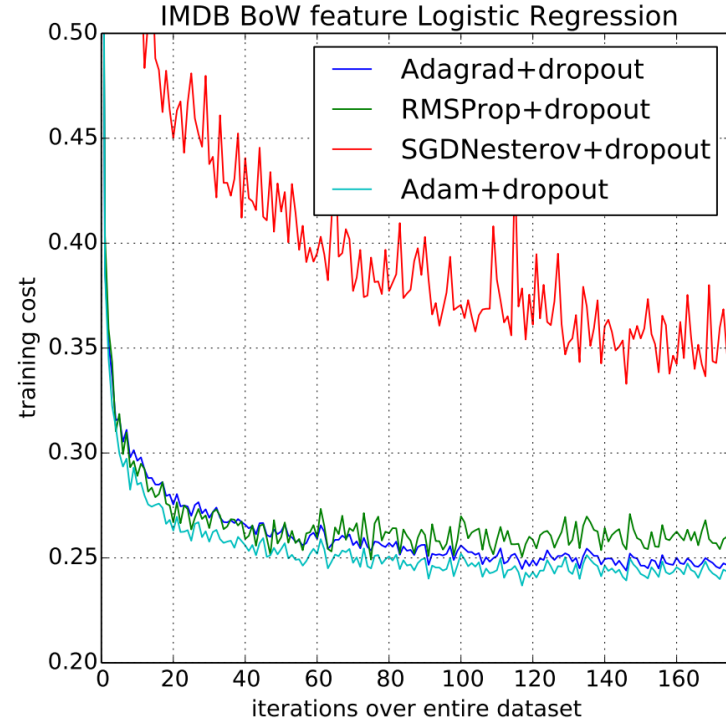
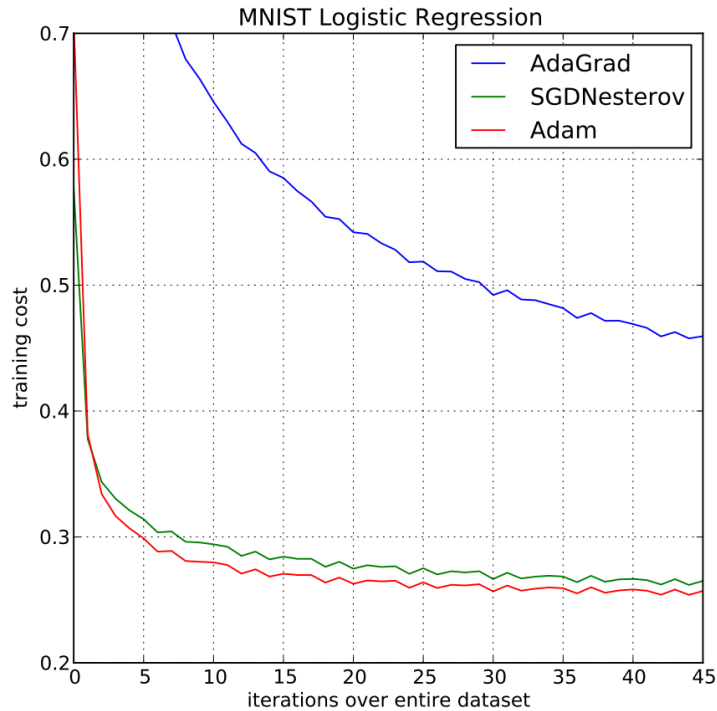
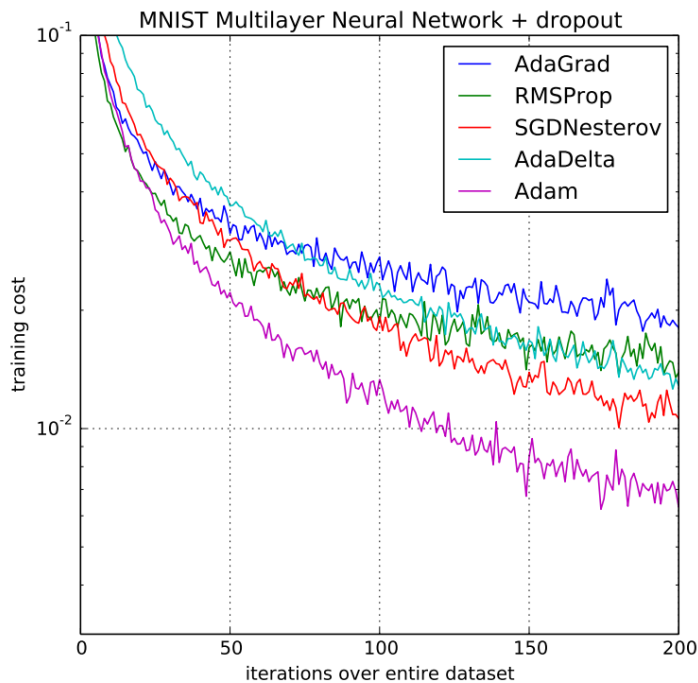
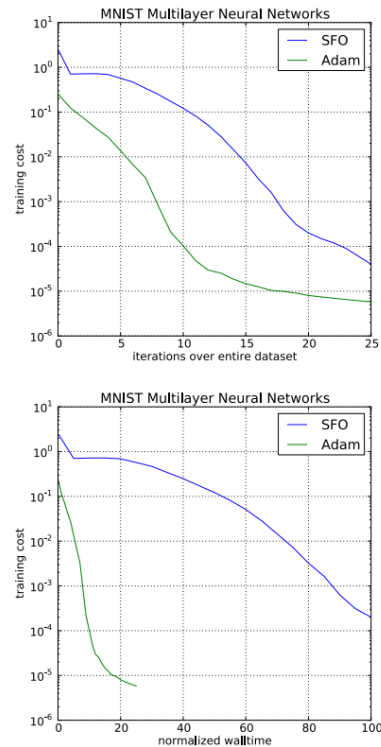


Figure 1: Logistic regression training negative log likelihood on MNIST images and IMDB movie reviews with 10,000 bag-of-words (BoW) feature vectors.

EXPERIMENT: MULTI-LAYER NEURAL NETWORKS



(a)



(b)

Figure 2: Training of multilayer neural networks on MNIST images. (a) Neural networks using dropout stochastic regularization. (b) Neural networks with deterministic cost function. We compare with the sum-of-functions (SFO) optimizer (Sohl-Dickstein et al., 2014)

EXPERIMENT: CONVOLUTIONAL NEURAL NETWORKS

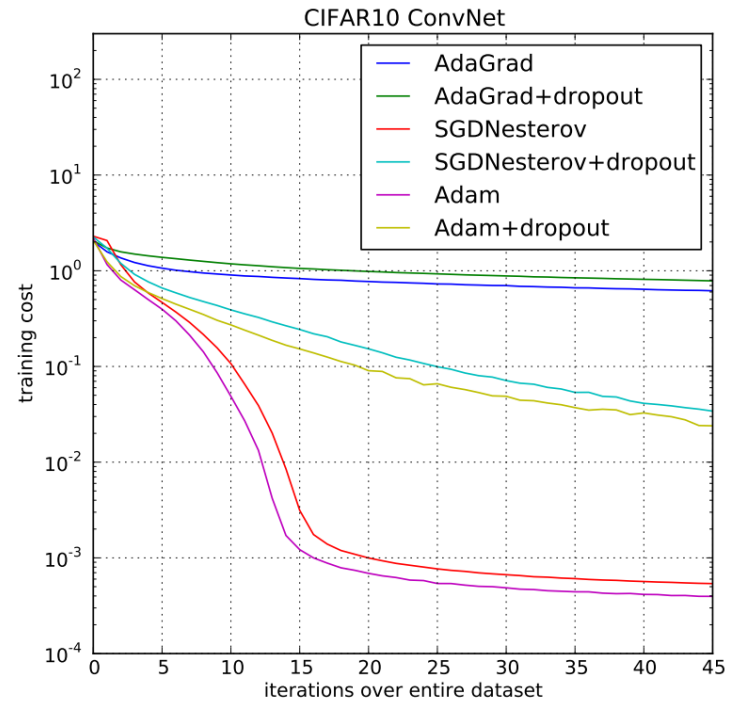
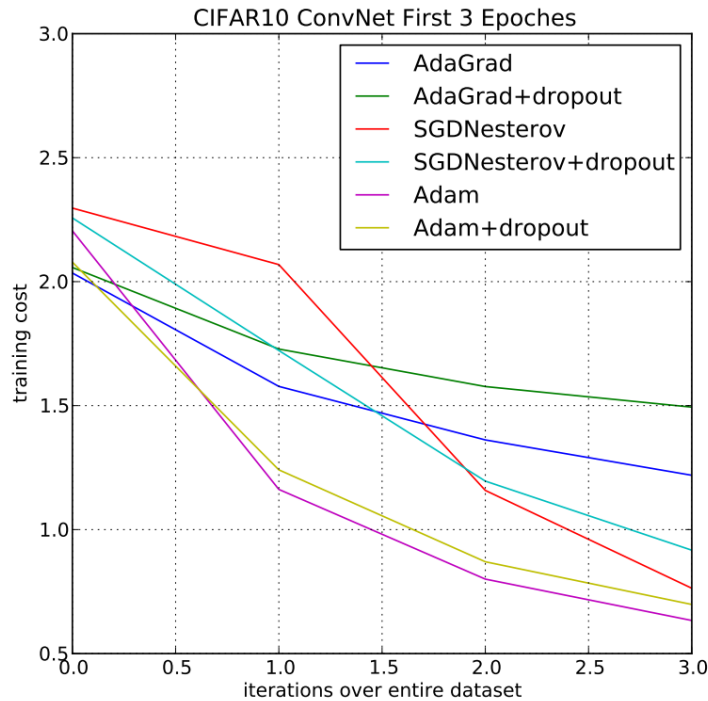


Figure 3: Convolutional neural networks training cost. (left) Training cost for the first three epochs. (right) Training cost over 45 epochs. CIFAR-10 with c64-c64-c128-1000 architecture.

EXPERIMENT: Bias-correction terms

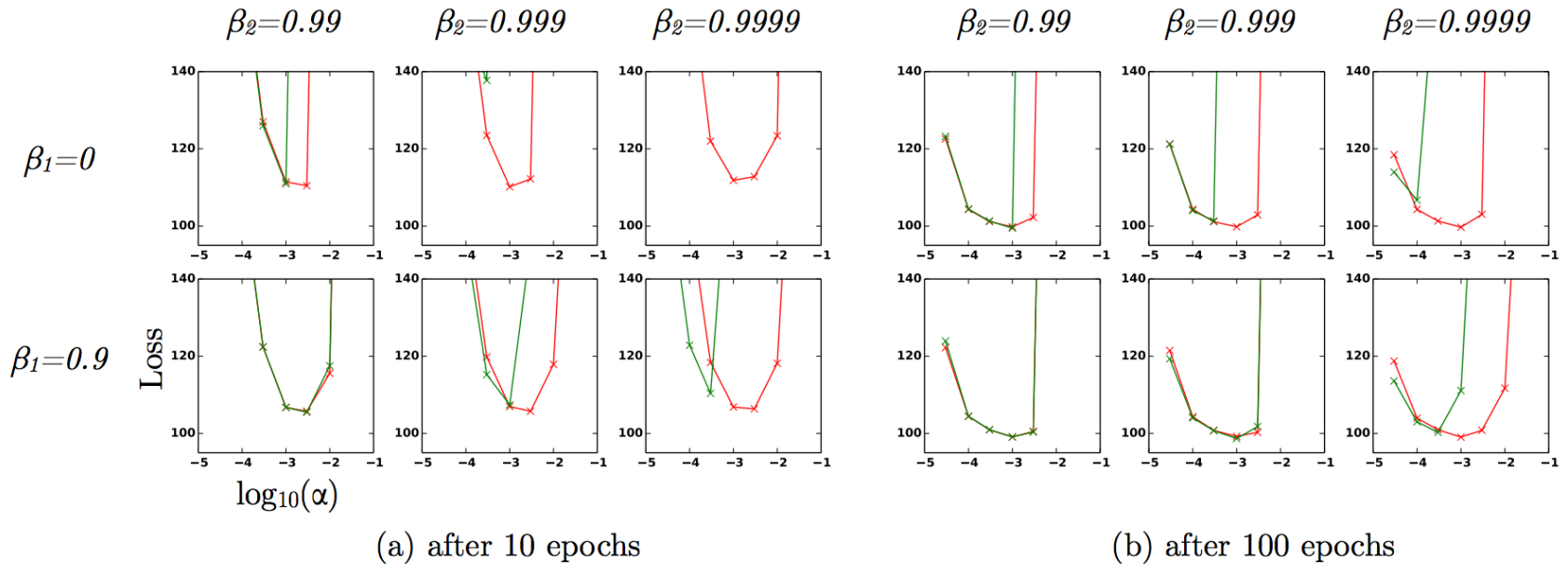


Figure 4: Effect of bias-correction terms (red line) versus no bias correction terms (green line) after 10 epochs (left) and 100 epochs (right) on the loss (y-axes) when learning a Variational Auto-Encoder (VAE) (Kingma & Welling, 2013), for different settings of stepsize α (x-axes) and hyper-parameters β_1 and β_2 .

Conclusion

- Adam Optimizer
- Simple and computationally efficient algorithm
- Combination of two popular optimization methods
 - AdaGrad : to deal with sparse gradients
 - RMSProp : to deal with non-stationary objectives
- Straightforward to implement
- Requirement little memory
- Robust and well-suited to a wide range of non-convex optimization problems in the field machine learning
- Cf.) Nadam : adding NAG instead of Momentum in Adam method