Neural Networks: Optimization Variants

CS 760: Machine Learning

Spring 2019

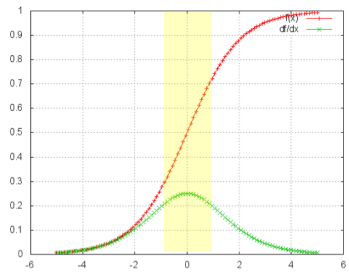
Mark Craven and David Page

Goals for the Lecture

- You should understand the following concepts:
  - momentum
  - Nesterov momentum
  - adaptive step size
  - AdaGrad
  - RMSProp
  - Adam
- Thanks to Yujia Bao for contributions to these lecture notes

Initializing weights

- Weights should be initialized to
  - small values so that e.g. with sigmoids, activations are in the range where the derivative is large (learning will be quicker)

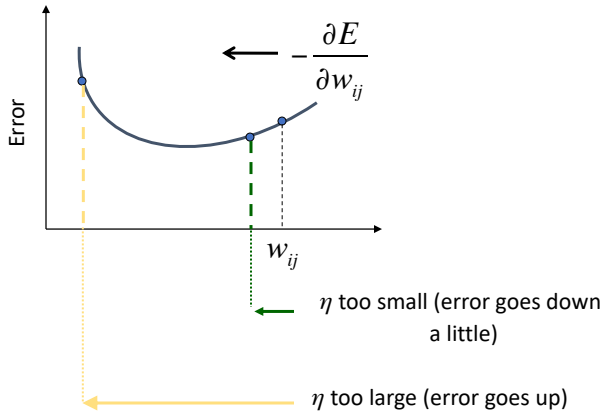


  - random values to ensure symmetry breaking (i.e. if all weights are the same, the hidden units will all represent the same thing)
- typical initial weight range [-0.01, 0.01]

3

Setting the learning rate

convergence depends on having an appropriate learning rate



4

Steepest Descent (a.k.a Gradient Descent)

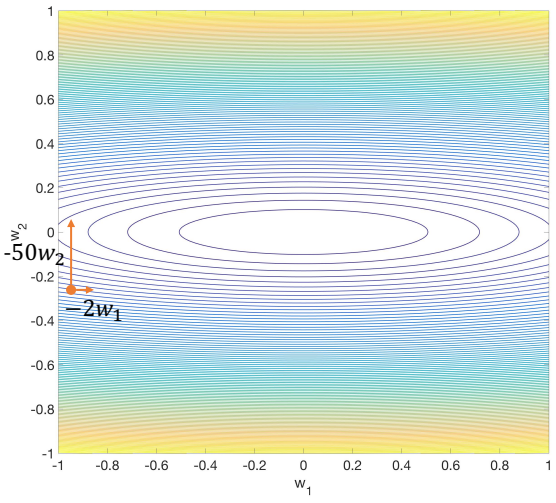
Algorithm:

Step 1: $w^{(t+1)} = w^{(t)} - \eta \cdot \frac{\partial E}{\partial w}\Big|_{w^{(t)}}$

- weight on iteration $t+1$
- weight on iteration t
- gradient computed at weight from iteration t

- Very simple
- Poor convergence if the function surface is steeper in one dimension than in another. Let's see an example!

Steepest Descent



Objective:

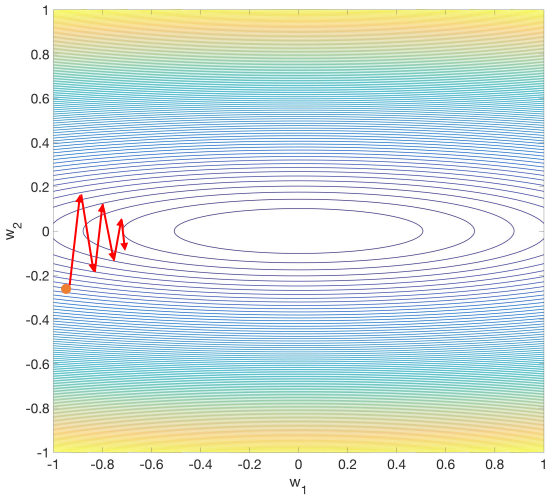
$$E(\boldsymbol{w}) = w_1^2 + 25w_2^2$$

Gradient:

$$\nabla E = \begin{bmatrix} 2w_1 \\ 50w_2 \end{bmatrix}$$

Small over w_1 direction, large over w_2 direction!

Steepest Descent



Zigzag behavior!

Move very slowly on the flat direction, oscillate on the steep direction.

Momentum

Algorithm:

Step 1: $v^{(t+1)} = \mu v^{(t)} - \eta \cdot \left.\frac{\partial E}{\partial w}\right|_{w^{(t)}}$

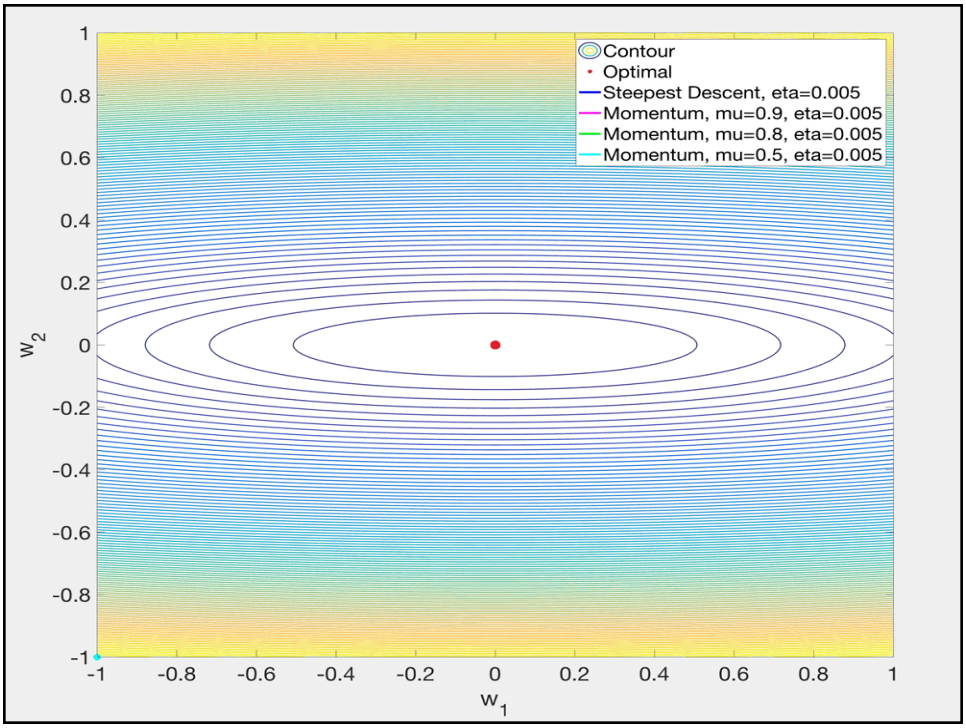
Interpret v as velocity, $1-\mu$ as friction. v stores the moving average of the gradients.

Step 2: $w^{(t+1)} = w^{(t)} + v^{(t+1)}$

- Initialize $v^{(0)} = 0$
- Common settings for μ range between 0.5 and 0.99

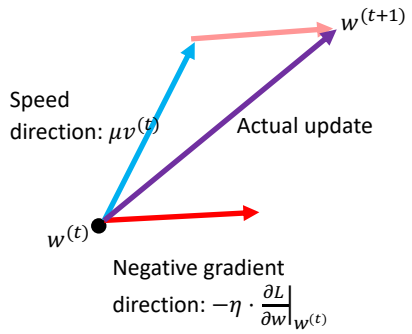
Momentum

- Accelerate convergence on the direction that the gradient keeps the same sign over the past few iterations.
- If μ is large, momentum will overshoot the target, but the overall convergence is still faster than steepest descent in practice.
- As μ is made smaller, momentum behaves more like steepest descent.

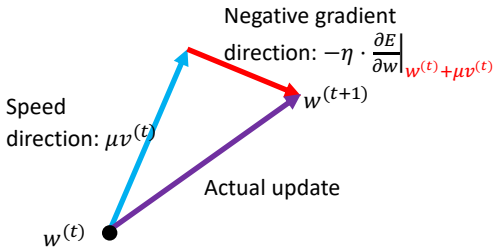


Nesterov Accelerated Gradient

Momentum



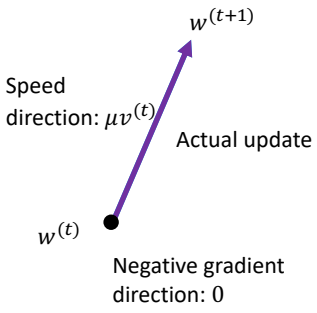
Nesterov Accelerated Gradient



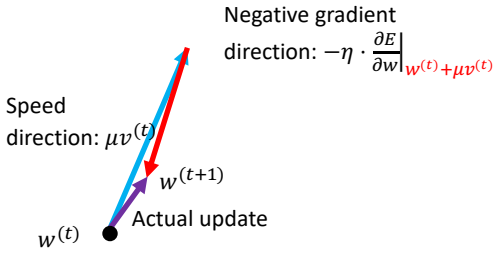
An example

Suppose $w^{(t)}$ is already optimal, but $v^{(t)} \neq 0$.

Momentum



Nesterov Accelerated Gradient



A look-ahead approach

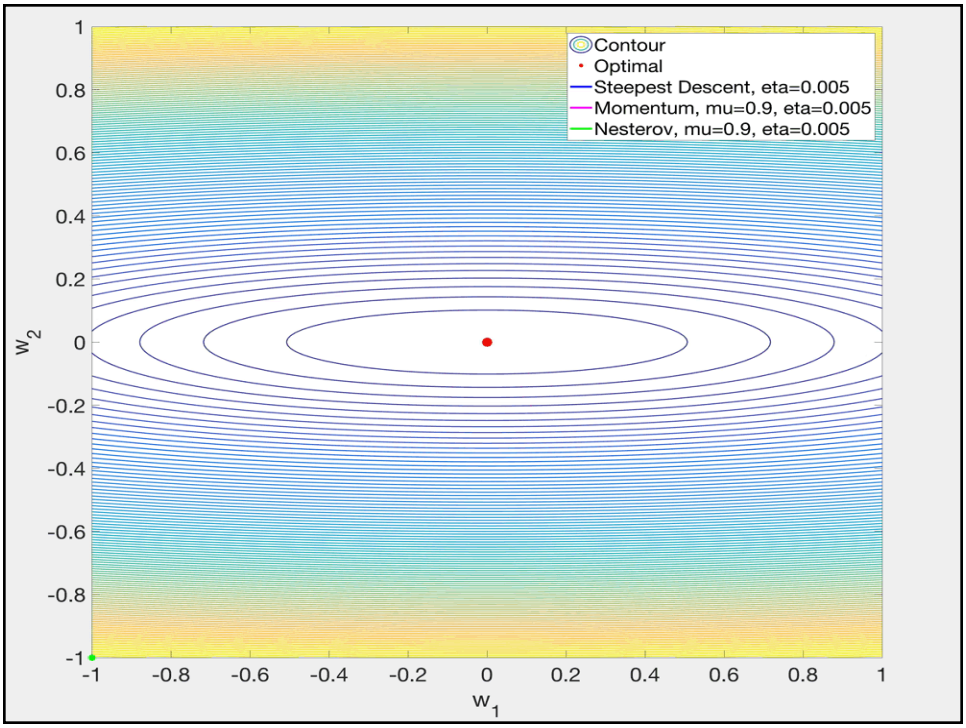
Nesterov Accelerated Gradient

Algorithm:

Step 1: $v^{(t+1)} = \mu v^{(t)} - \eta \cdot \frac{\partial E}{\partial w}\Big|_{w^{(t)}+\mu v^{(t)}}$

Step 2: $w^{(t+1)} = w^{(t)} + v^{(t+1)}$

Nesterov Accelerated Gradient has better performance than momentum [Ilya et al., 2013]



Some notes on notation in this lecture

- For a scalar η and vector $\boldsymbol{v}$, $\eta\boldsymbol{v}$ is a vector
- Likewise $\frac{\eta}{\boldsymbol{v}}$ is a vector, just as $\eta\frac{1}{\boldsymbol{v}}$ is a vector
- By $\odot$ between two vectors, we mean element-wise multiplication rather than dot product

AdaGrad (Adaptive Gradient)

[Duchi et al., 2011]

Algorithm:

Step 1: $G^{(t+1)} = G^{(t)} + \left(\frac{\partial E}{\partial w}\Big|_{w^{(t)}}\right)^2$

G is the sum of the square gradient

Step 2: $w^{(t+1)} = w^{(t)} - \frac{\eta}{\sqrt{G^{(t+1)}+10^{-7}}} \odot \frac{\partial E}{\partial w}\Big|_{w^{(t)}}$

Scale the learning rate by $1/\sqrt{G}$ and follow the negative gradient direction

Common setting for η is 0.01.

Different learning rate for each w (each w has its own G)

- Increase the learning rate on the flat direction
- Decrease the learning rate on the steep direction

AdaGrad (Adaptive Gradient)

Consider the previous example:

$$E(w) = w_1^2 + 25w_2^2$$

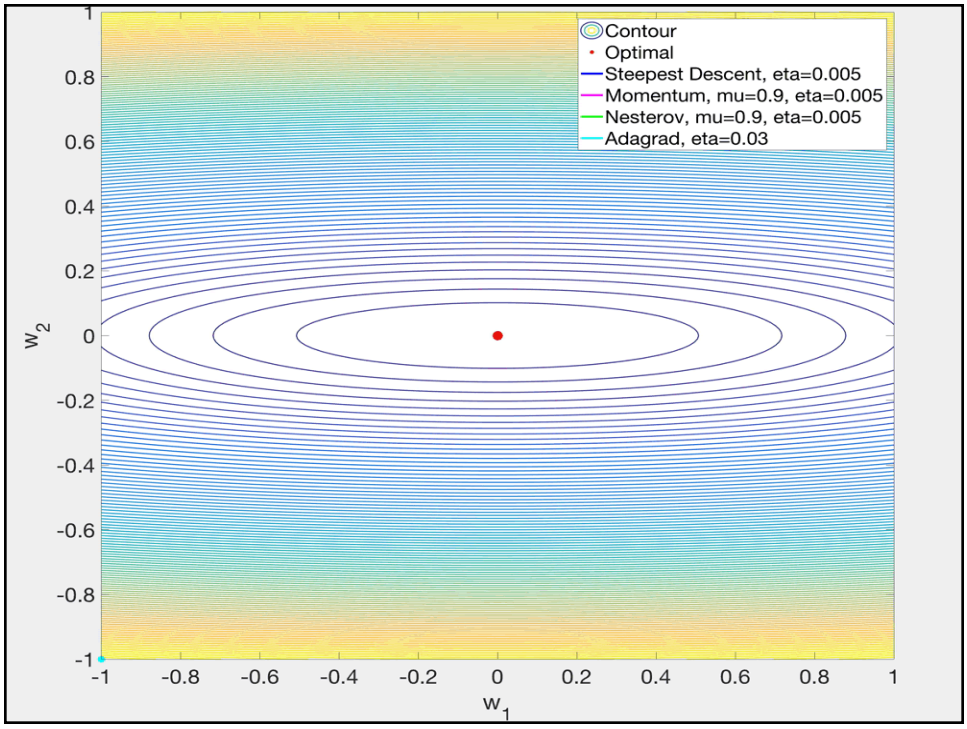
Gradient:

$$\nabla E = \begin{bmatrix} 2w_1 \\ 50w_2 \end{bmatrix}$$

Will follow the diagonal direction!

Initial $w^{(0)}$ is $(-1, -1)$.

$$w^{(1)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} \frac{\eta}{\sqrt{(-2)^2 + 10^{-7}}} \cdot (-2) \\ \frac{\eta}{\sqrt{(-50)^2 + 10^{-7}}} \cdot (-50) \end{bmatrix} \approx \begin{bmatrix} -1 \\ -1 \end{bmatrix} + \begin{bmatrix} \eta \\ \eta \end{bmatrix}$$



AdaGrad (Adaptive Gradient)

Algorithm:

Step 1: $G^{(t+1)} = G^{(t)} + \left(\frac{\partial E}{\partial w}\Big|_{w^{(t)}}\right)^2$

Step 2: $w^{(t+1)} = w^{(t)} - \frac{\eta}{\sqrt{G^{(t+1)}+10^{-7}}} \odot \frac{\partial E}{\partial w}\Big|_{w^{(t)}}$

Why is convergence that slow?

Learning rate goes down monotonically as t increases.

RMSProp (Root Mean Square Propagation)

[Tieleman & Hinton, 2012]

Algorithm:

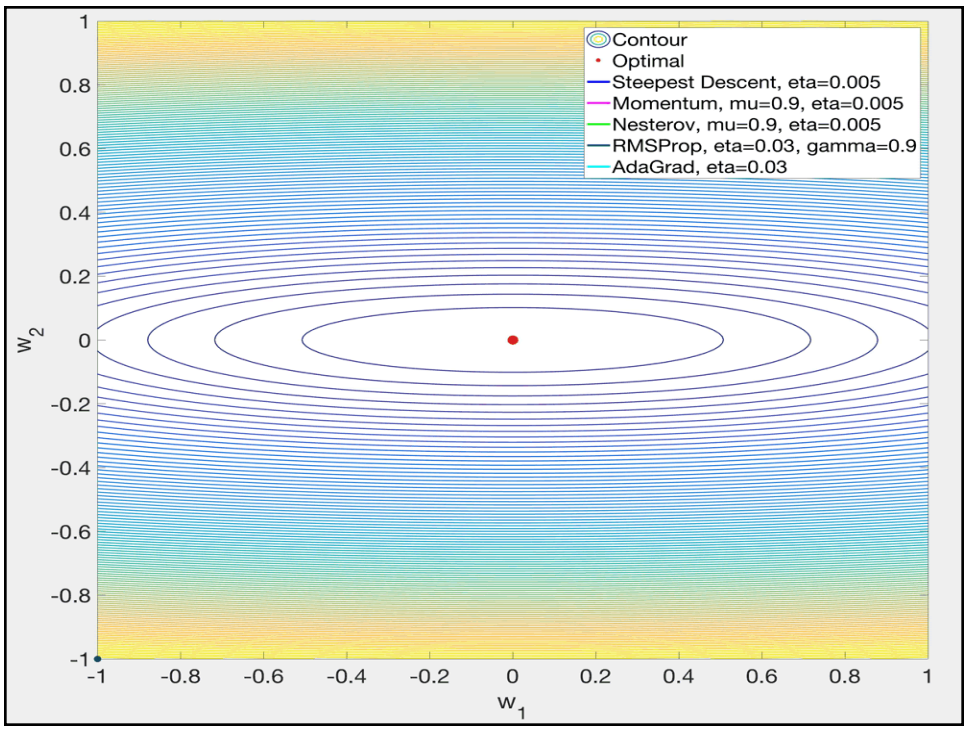
Step 1: $G^{(t+1)} = \gamma G^{(t)} + (1-\gamma) \cdot \left(\frac{\partial E}{\partial w}\Big|_{w^{(t)}}\right)^2$

G is a moving exponential average of the square gradient.

Step 2: $w^{(t+1)} = w^{(t)} - \frac{\eta}{\sqrt{G^{(t+1)}+10^{-7}}} \odot \frac{\partial E}{\partial w}\Big|_{w^{(t)}}$

Scale the learning rate by $1/\sqrt{G}$ and follow the negative gradient direction.

Default settings: $\gamma = 0.9, \eta = 0.001$



Another example (saddle point)

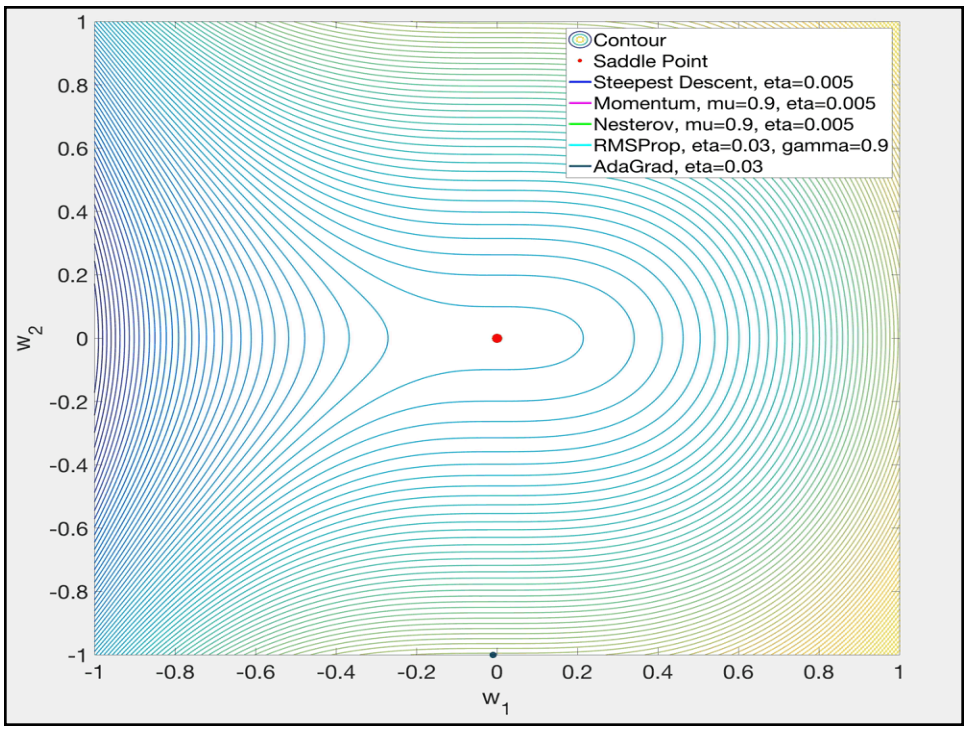
Now consider minimizing

$$L(w) = w_1^3 + w_2^2$$

The problem is unbounded and $(0,0)$ is a saddle point.

At $(-0.01, -1)$, we have $\nabla E = \begin{bmatrix} 0.0003 \\ -2 \end{bmatrix}$

How will these methods behave if we start at this point?



Adam (Adaptive Moment Estimation)

[Kingma & Ba, 2015]

Algorithm [Simplified Version]:

Step 1: $m^{(t+1)} = \beta_1 m^{(t)} + (1-\beta_1) \cdot \left.\frac{\partial E}{\partial w}\right|_{w^{(t)}}$

m is the exponential moving average of the gradients (similar to speed in the momentum method)

Step 2: $v^{(t+1)} = \beta_2 v^{(t)} + (1-\beta_2) \cdot \left(\left.\frac{\partial E}{\partial w}\right|_{w^{(t)}}\right)^2$

v is the exponential moving average of the square gradients

Step 3: $w^{(t+1)} = w^{(t)} - \frac{\eta}{\sqrt{v^{(t+1)}}+10^{-8}} \odot m^{(t+1)}$

Default settings: $\beta_1 = 0.9,\ \beta_2 = 0.999,\ \eta = 0.001$

Adam (Adaptive Moment Estimation)

Algorithm [Complete Version]:

Step 1: $m^{(t+1)} = \beta_1 m^{(t)} + (1-\beta_1) \cdot \left.\frac{\partial E}{\partial w}\right|_{w^{(t)}}$

Step 2: $v^{(t+1)} = \beta_2 v^{(t)} + (1-\beta_2) \cdot \left(\left.\frac{\partial E}{\partial w}\right|_{w^{(t)}}\right)^2$

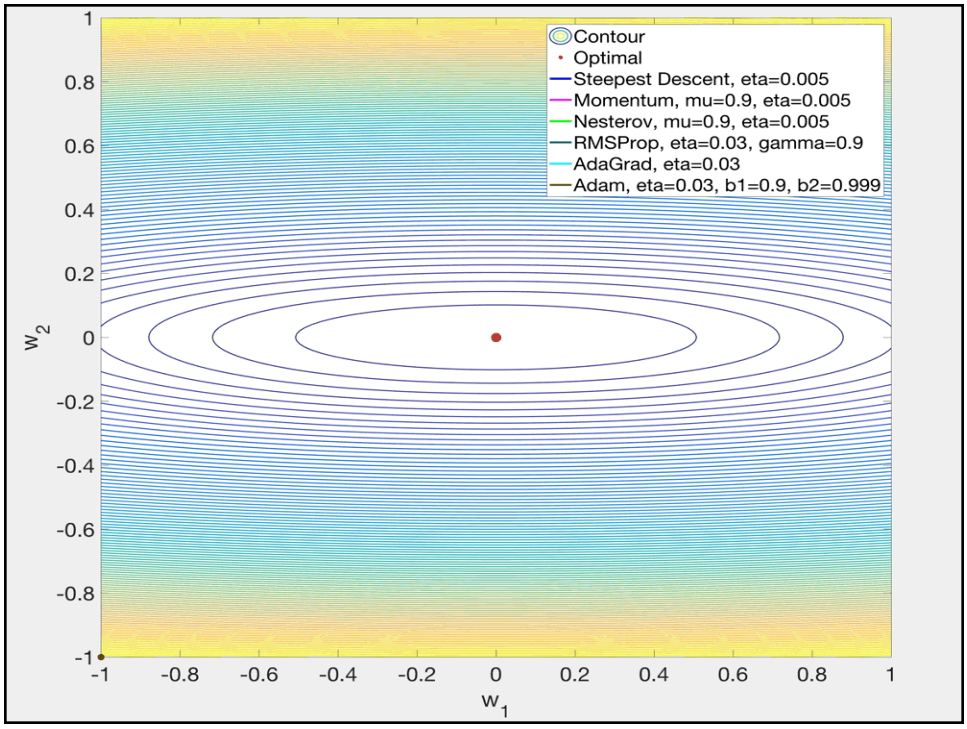
Step 3: $\hat{m}^{(t+1)} = \frac{1}{1-\beta_1^t} \cdot m^{(t+1)}$

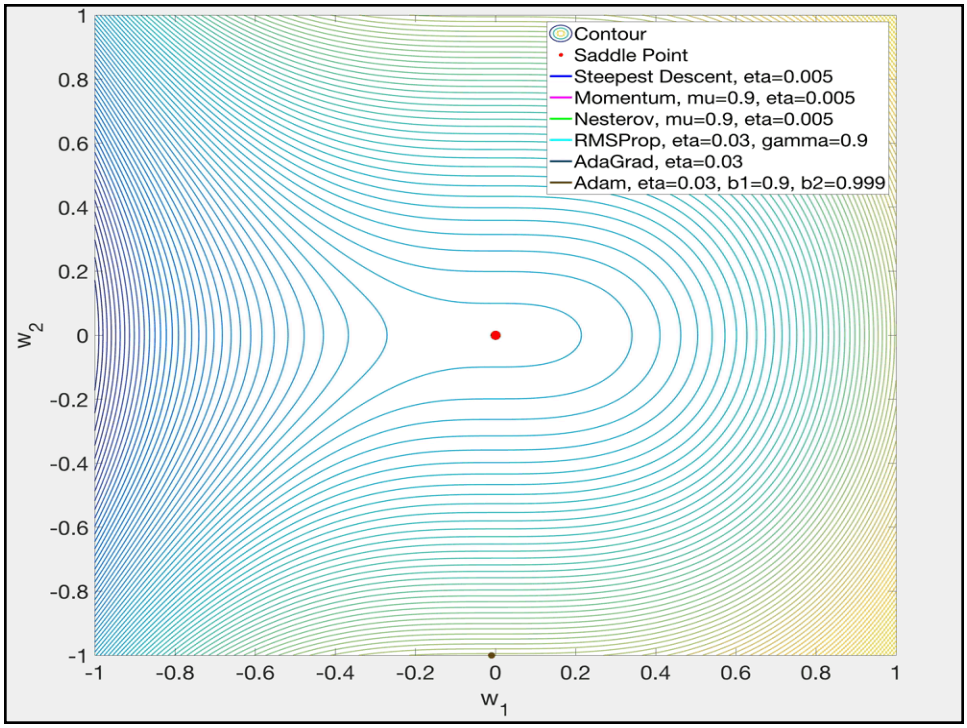
Step 4: $\hat{v}^{(t+1)} = \frac{1}{1-\beta_2^t} \cdot v^{(t+1)}$

Bias Correction

(since we set $m^{(0)}, v^{(0)}$ to 0)

Step 5: $w^{(t+1)} = w^{(t)} - \frac{\eta}{\sqrt{\hat{v}^{(t+1)}}+10^{-8}} \cdot \hat{m}^{(t+1)}$





Adam (Adaptive Moment Estimation)

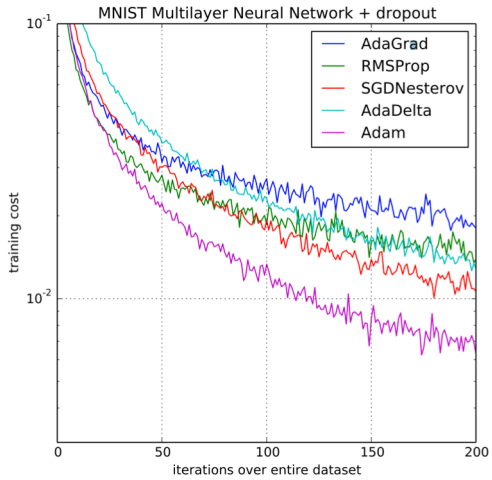


Figure from Kingma & Ba, 2015

Network configuration:

- Input layer: 28x28
- 2 hidden layers (1000 ReLUs each)
- Output: 10 SoftMax
- Cross-entropy loss