Deep Learning II: Momentum & Adaptive Step Size

CS 760: Machine Learning

Spring 2018

Mark Craven and David Page

www.biostat.wisc.edu/~craven/cs760

Goals for the Lecture

- You should understand the following concepts:
  - momentum
  - Nesterov momentum
  - adaptive step size
  - AdaGrad
  - RMSProp
  - Adam
- Thanks Yujia Bao!

Background: Mini-batch gradient descent

Procedure:

- Pick a mini-batch of examples from the data
- For each example:
  - Forward propagation to get the output;
  - Backward propagation to get the gradient;
- Update the weights using the average gradient.

Common mini-batch size: 32, 64, 128.

Note: Forward/Backward propagation can both be done simultaneously for many examples using matrix multiplication, and this is fast on GPUs!

Mini-batch gradient descent

Recall the forward prop for a MLP:

$$z_{k+1} = f_{k+1}(W_k z_k + b_k)$$

where f_{k+1} is the activation function for the $(k+1)$st layer, $W_k[i][j]$ is the weights from $z_k[j]$ to $z_{k+1}[i]$

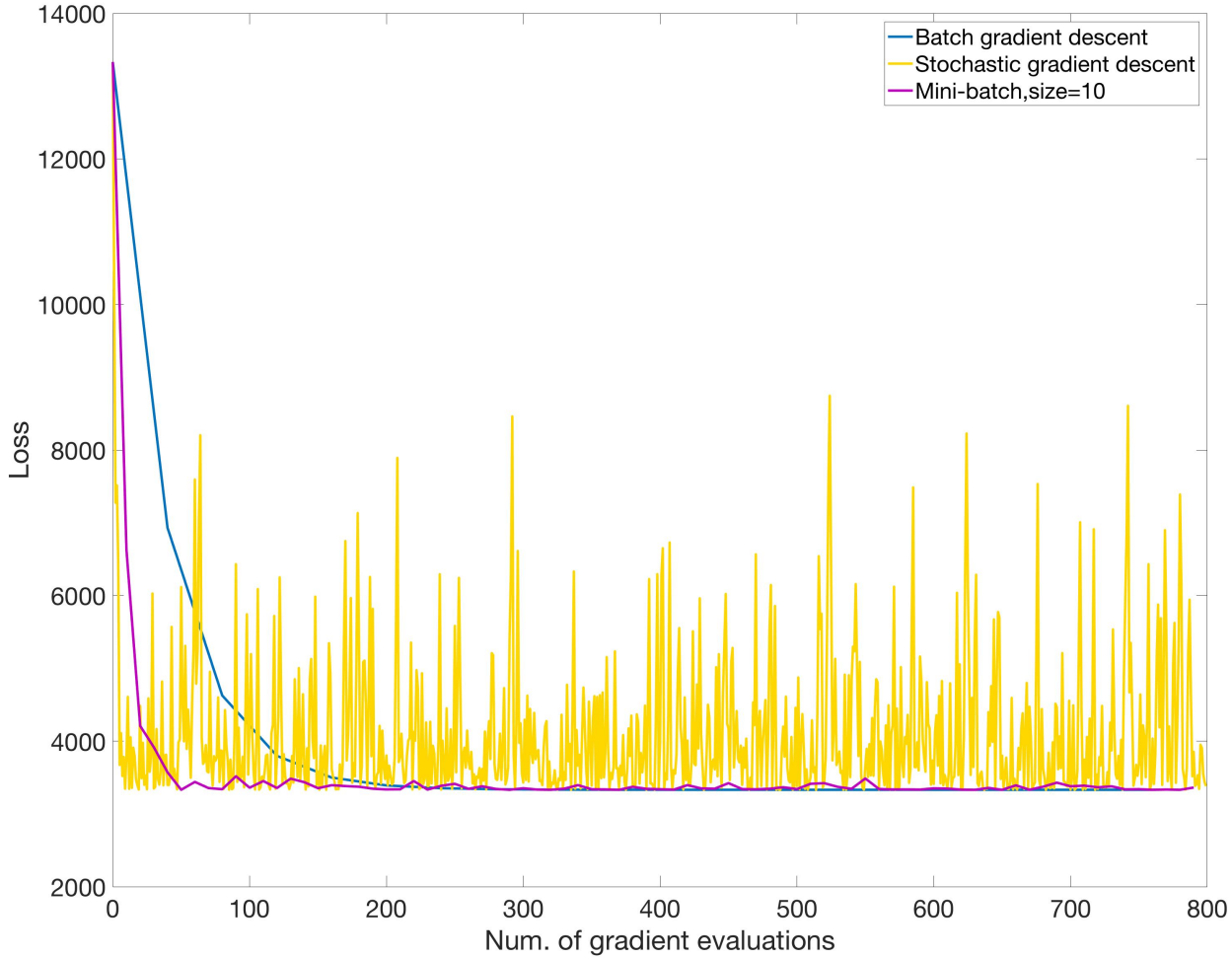
If we have n examples, we can define

$$Z_k = [z_k^1 \quad \cdots \quad z_k^n], \qquad B_k = [b_k \quad \cdots \quad b_k]$$

and express the forward prop as

$$Z_{k+1} = f_k(W_k Z_k + B_k)$$

An example


Batch gradient descent
Stochastic gradient descent
Mini-batch,size=10
Loss
Num. of gradient evaluations
14000
12000
10000
8000
6000
4000
2000
0
100
200
300
400
500
600
700
800

Weight Update Methods

- Steepest Descent (Review)
- Momentum (Review)
- Nesterov Accelerated Gradient

- AdaGrad [Duchi et al., 2011]
- RMSprop [Tieleman & Hinton, 2012]
- Adam [Kingma & Ba, 2015]

→ Element-wise adaptive learning rate

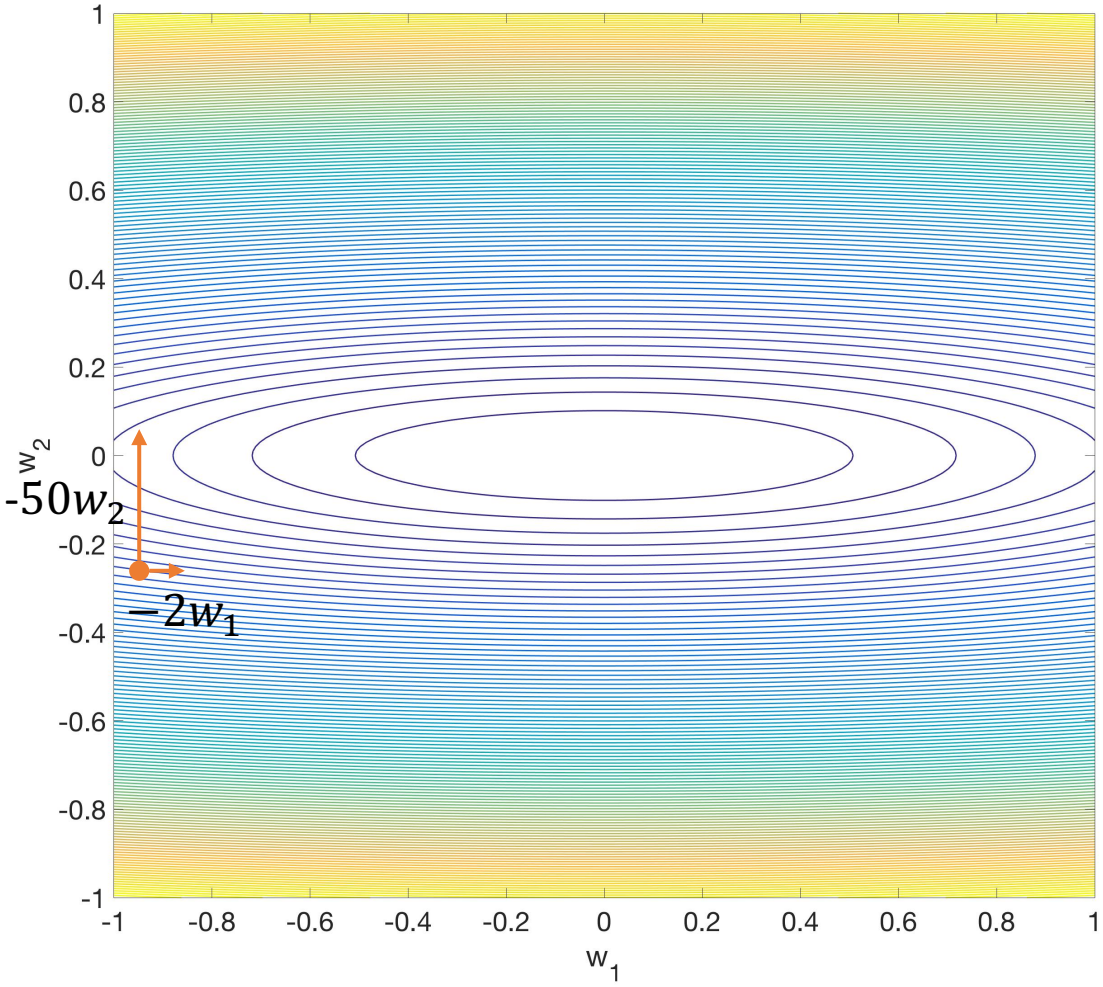
Steepest Descent

Algorithm:

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

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

Steepest Descent



Objective:

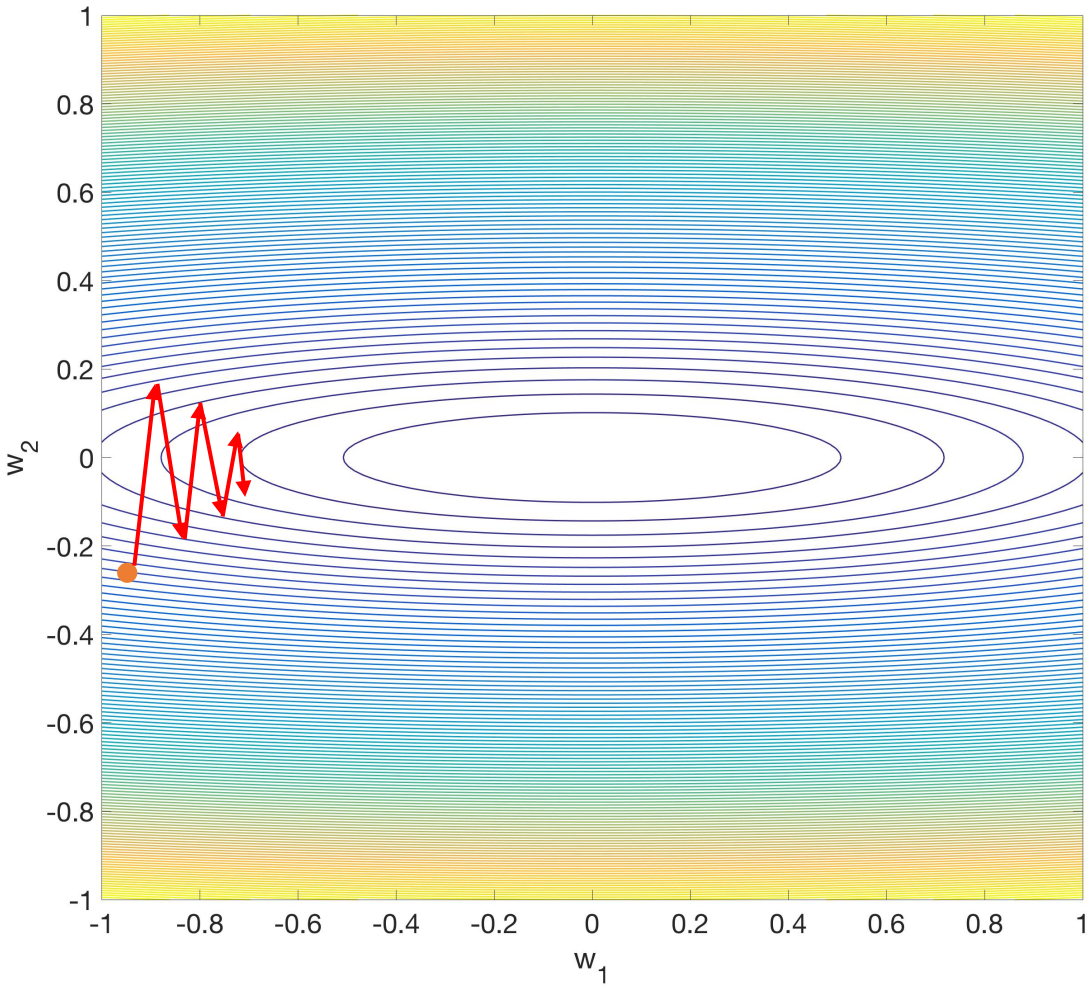
$$L(w) = w_1^2 + 25w_2^2$$

Gradient:

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

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.

Trick 7: Momentum

Algorithm:

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

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

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

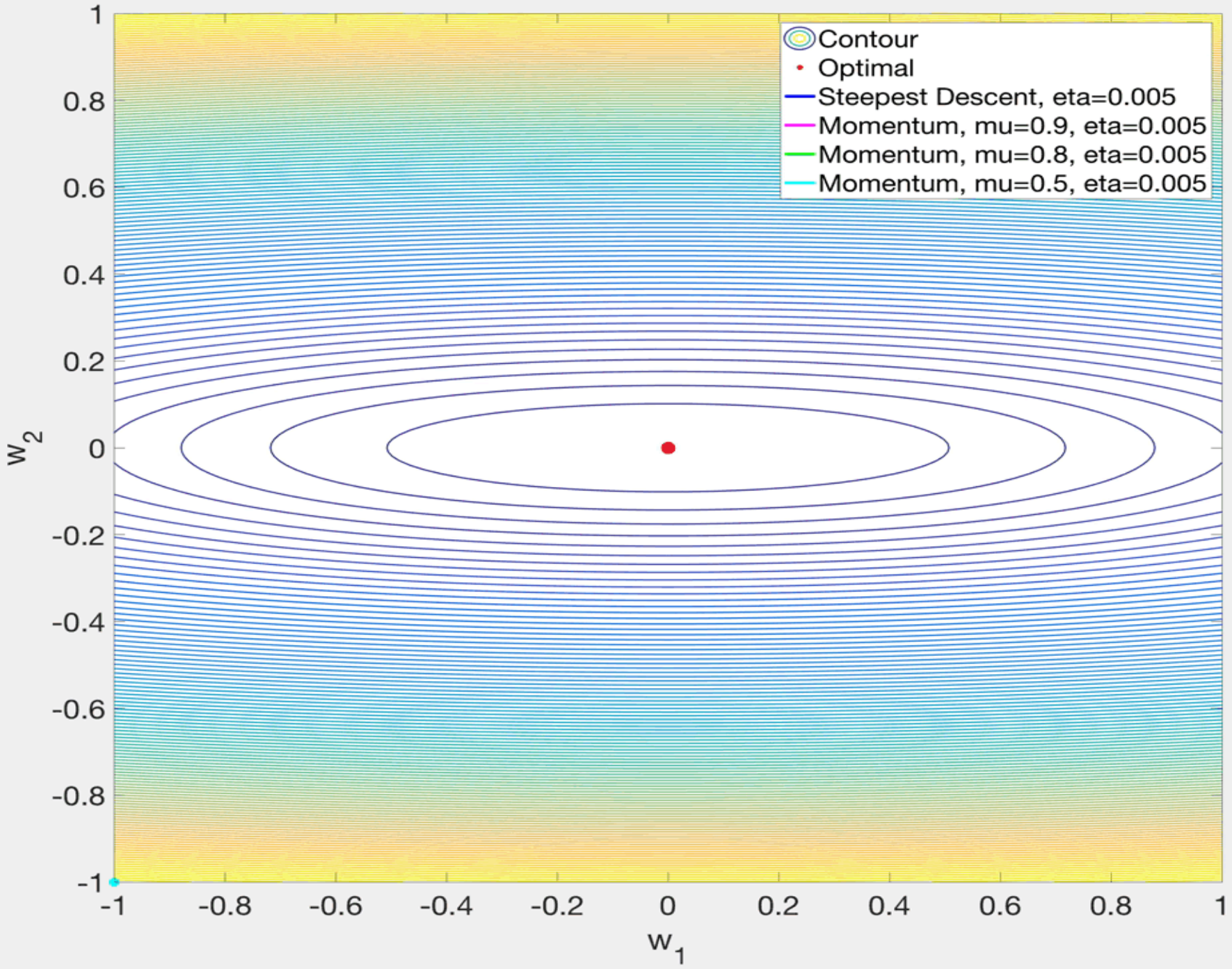
- A ball rolling down from the surface of the loss function.
- Initialize $v^{(0)}$ to zero.
- Common settings for μ ranges between 0.5 to 0.99.

Trick 7: Momentum

- Accelerate convergence on the direction that the gradient keeps the same sign over the past few iterations.

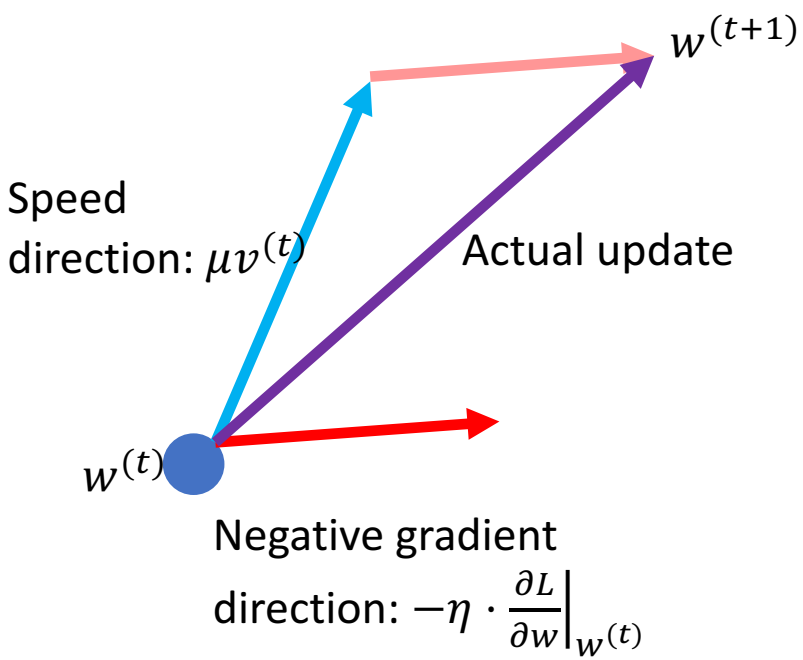
- If μ is large, momentum will overshoot the target, but the overall convergence is still faster than steepest descent in practice.

- As μ goes smaller, momentum behaves more similar to steepest descent.

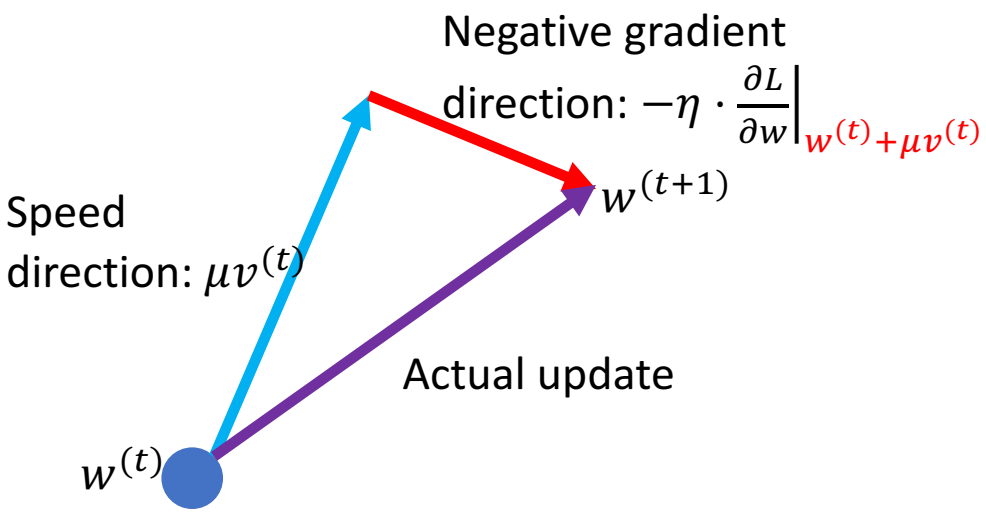

Contour
Optimal
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Momentum, mu=0.8, eta=0.005
Momentum, mu=0.5, eta=0.005
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1

Nesterov Accelerated Gradient

Momentum



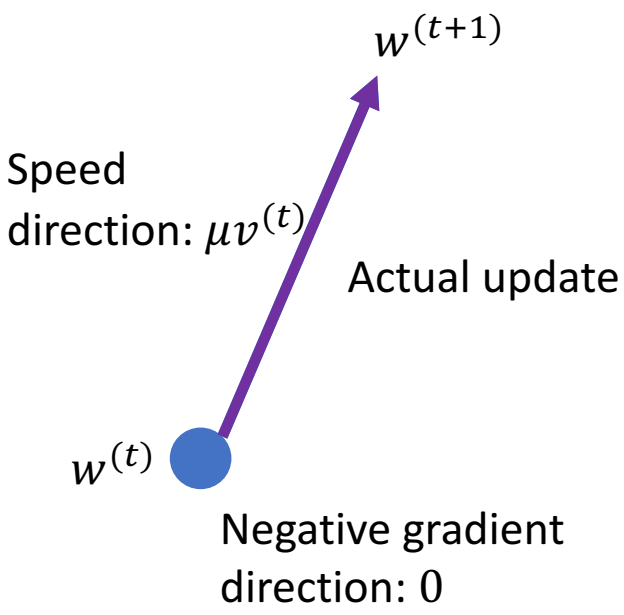
Nesterov Accelerated Gradient



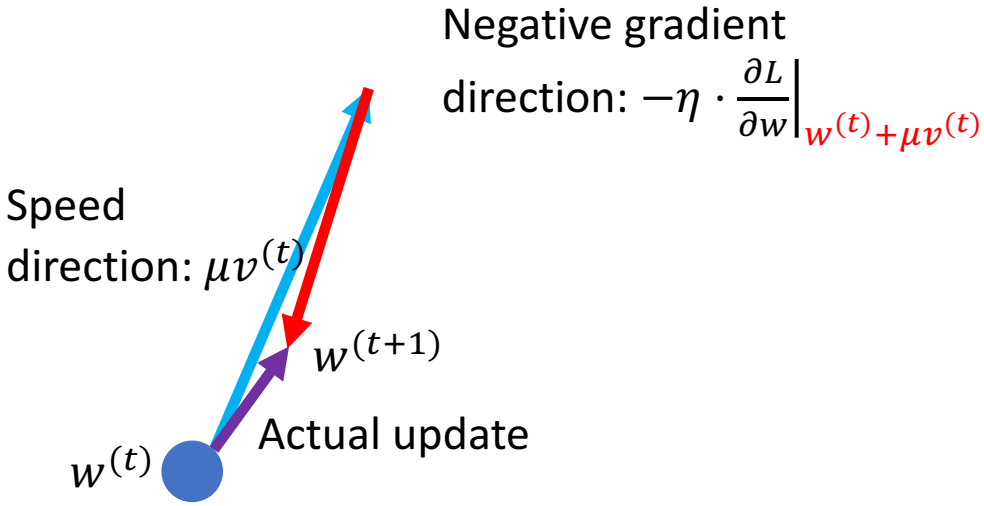
An example

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

Momentum



Nesterov Accelerated Gradient



Look ahead! The actual update is more reliable than Momentum.

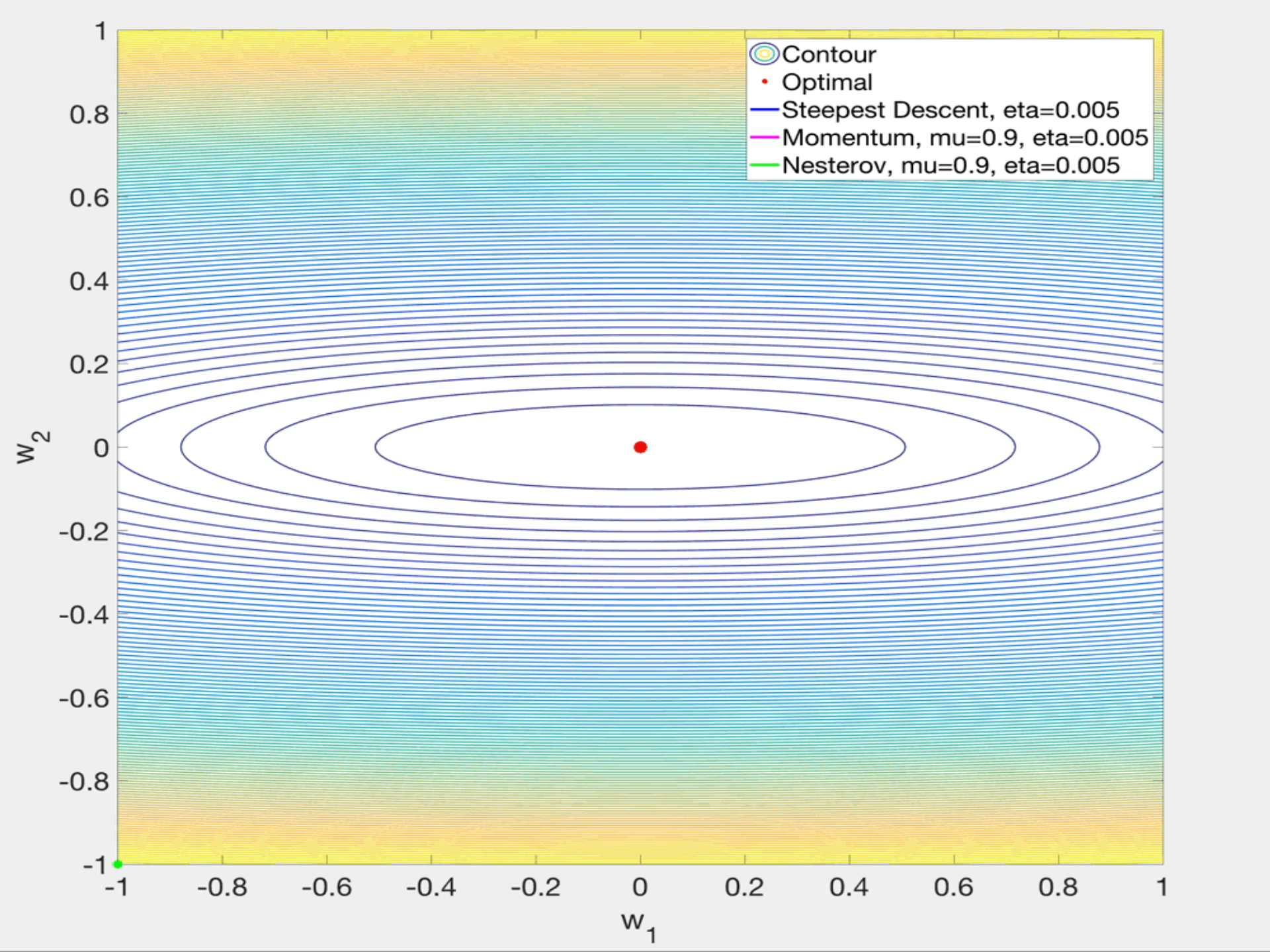
Nesterov Accelerated Gradient

Algorithm:

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

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

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


Contour
Optimal
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Nesterov, mu=0.9, eta=0.005
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1

Trick 8: Adaptive Learning Rate

Some Notes on our 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)

Algorithm:

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

G is the sum of the square gradient.

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

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:

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

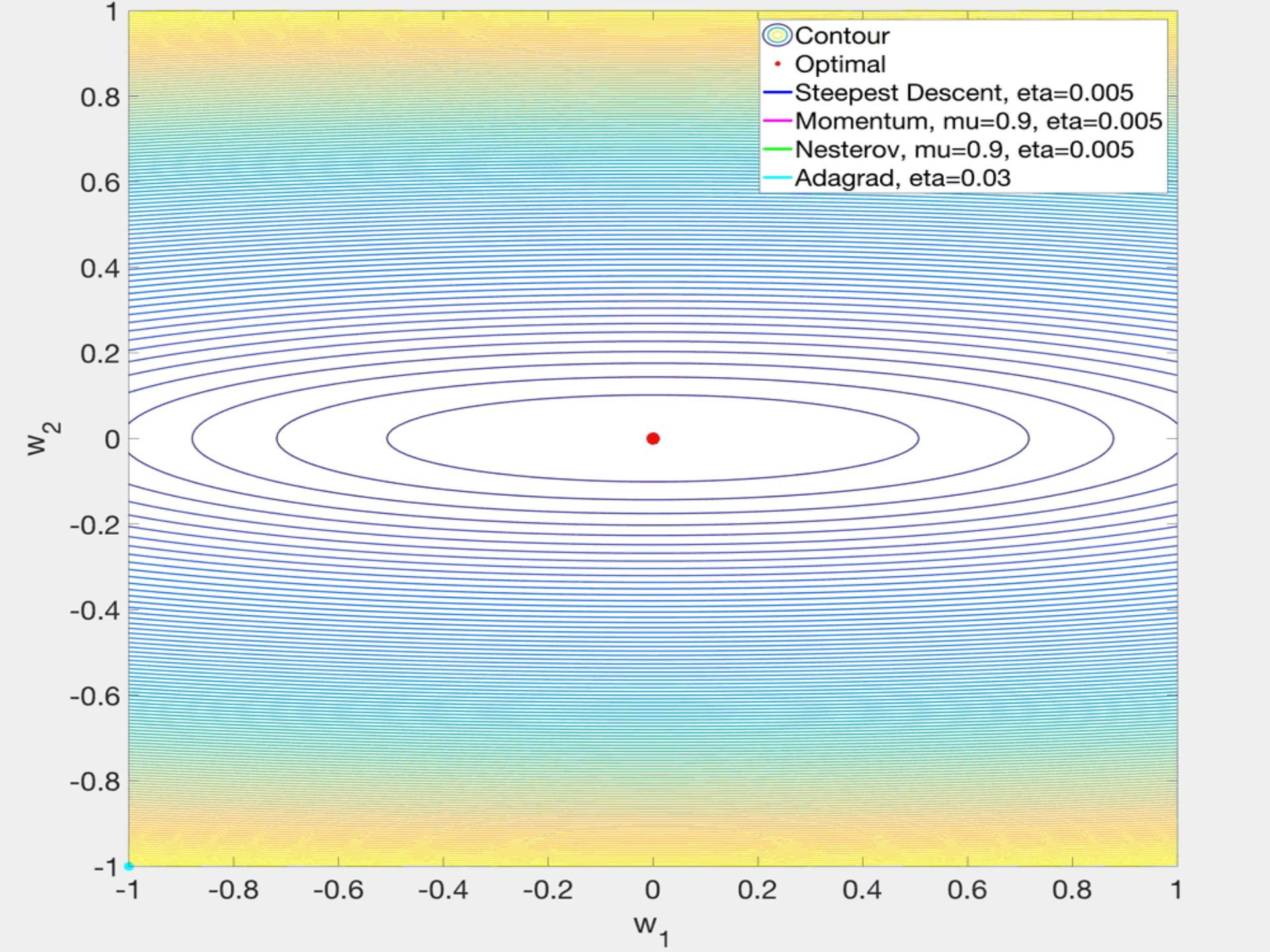
Gradient:

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

Will follow the diagonal direction!

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

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


Contour
Optimal
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Nesterov, mu=0.9, eta=0.005
Adagrad, eta=0.03
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1

AdaGrad (Adaptive Gradient)

Algorithm:

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

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

Why that slow?

Learning rate goes down monotonically as t increases.

Stop learning!!!

RMSProp (Root Mean Square Propagation)

Algorithm:

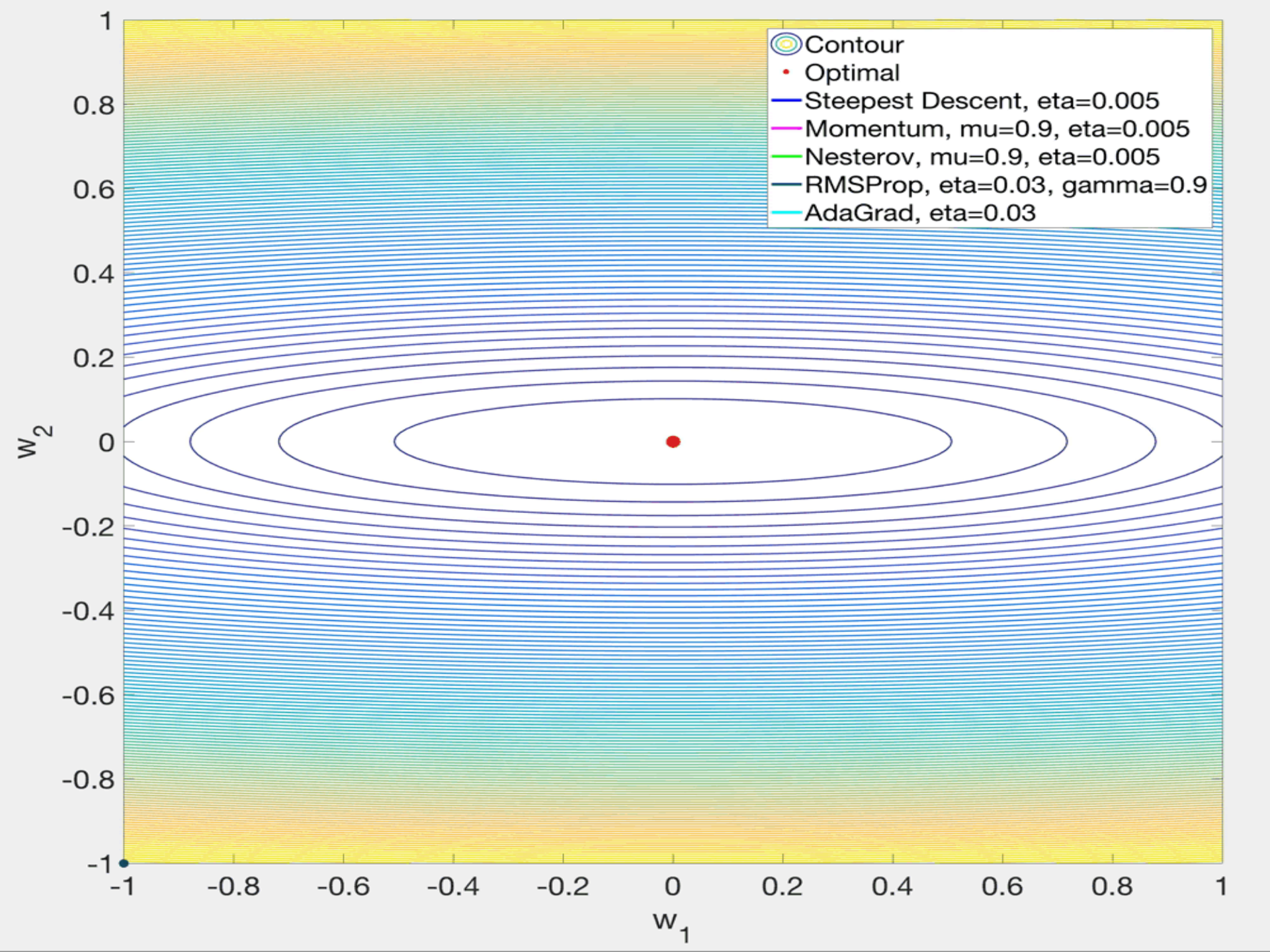
Step 1. $G^{(t+1)} = \gamma G^{(t)} + (1-\gamma) \cdot \left(\frac{\partial L}{\partial w}\Big|_{w^{(t)}} \right)^2$

G is a moving exponential average of the square gradient.

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

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

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


Contour
Optimal
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Nesterov, mu=0.9, eta=0.005
RMSProp, eta=0.03, gamma=0.9
AdaGrad, eta=0.03
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1

Another example (saddle point)

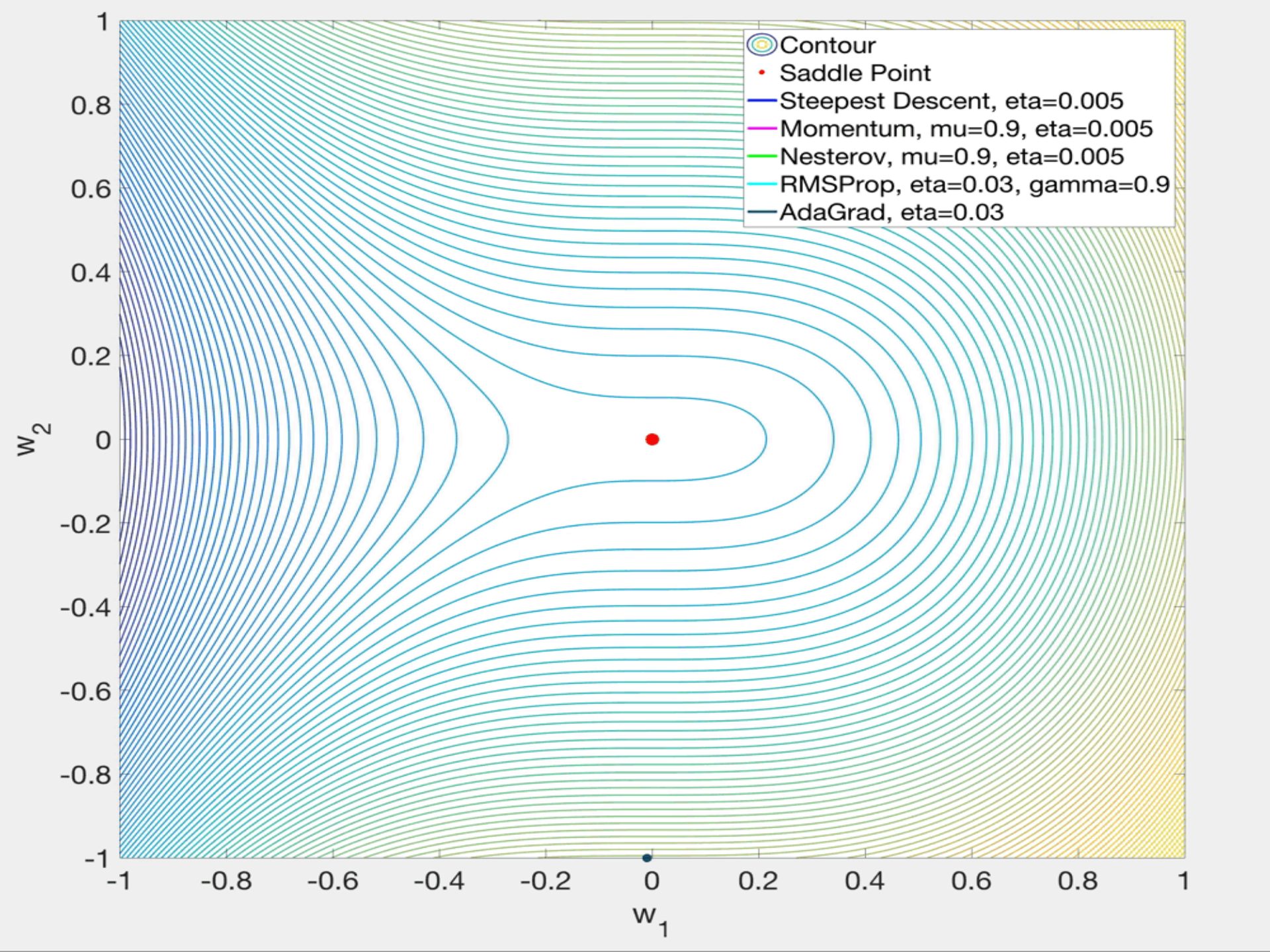
Now consider minimizing

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

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

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

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


Contour
Saddle Point
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Nesterov, mu=0.9, eta=0.005
RMSProp, eta=0.03, gamma=0.9
AdaGrad, eta=0.03
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1

Adam (Adaptive Moment Estimation)

Algorithm [Simplified Version]:

Step 1. $m^{(t+1)} = \beta_1 m^{(t)} + (1-\beta_1) \cdot \frac{\partial L}{\partial w}\Big|_{w^{(t)}}$

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

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

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

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

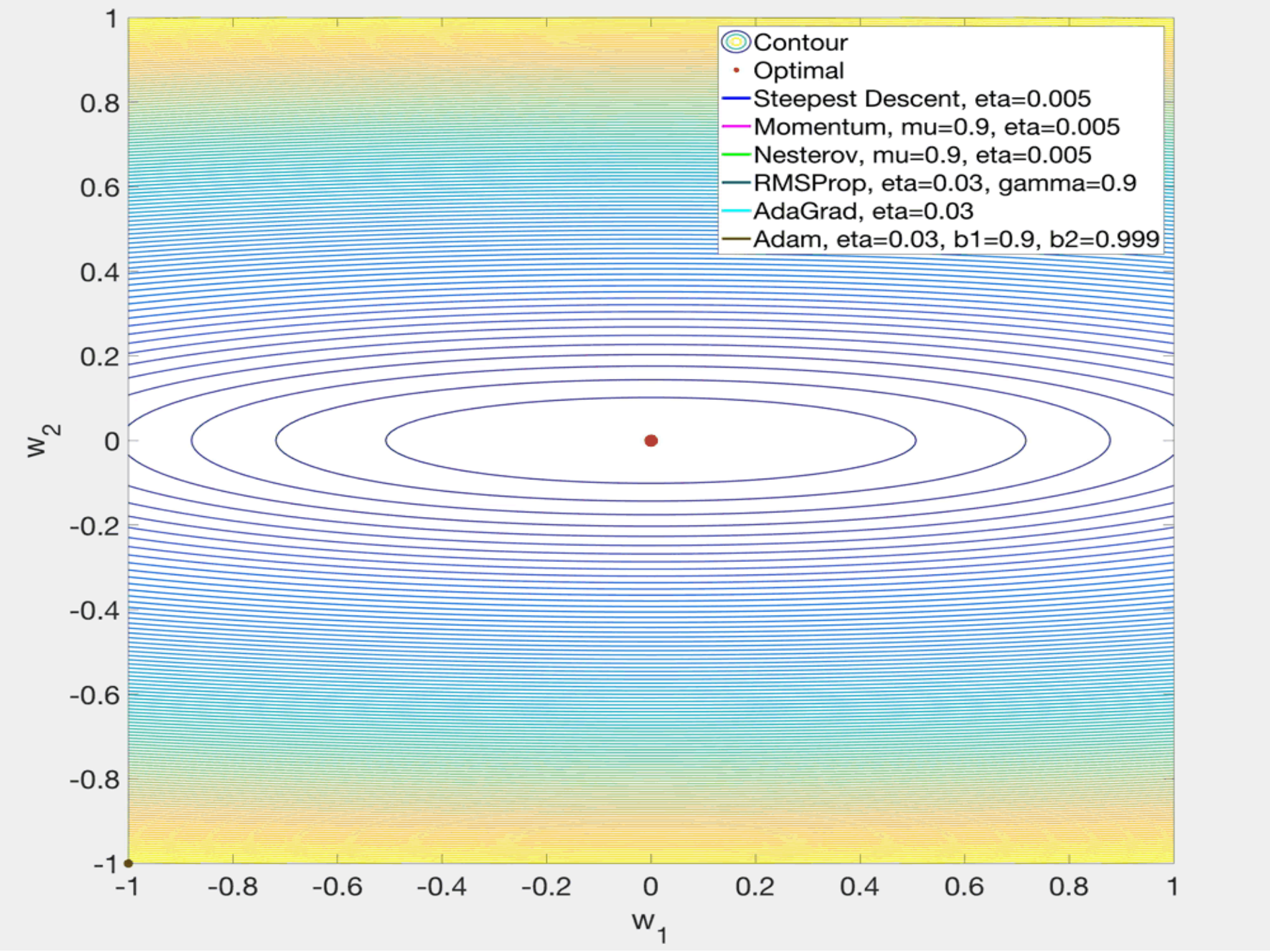
Step 3. $\hat{m}^{(t+1)} = \frac{1}{1-\beta_1^t} \cdot m^{(t+1)}$

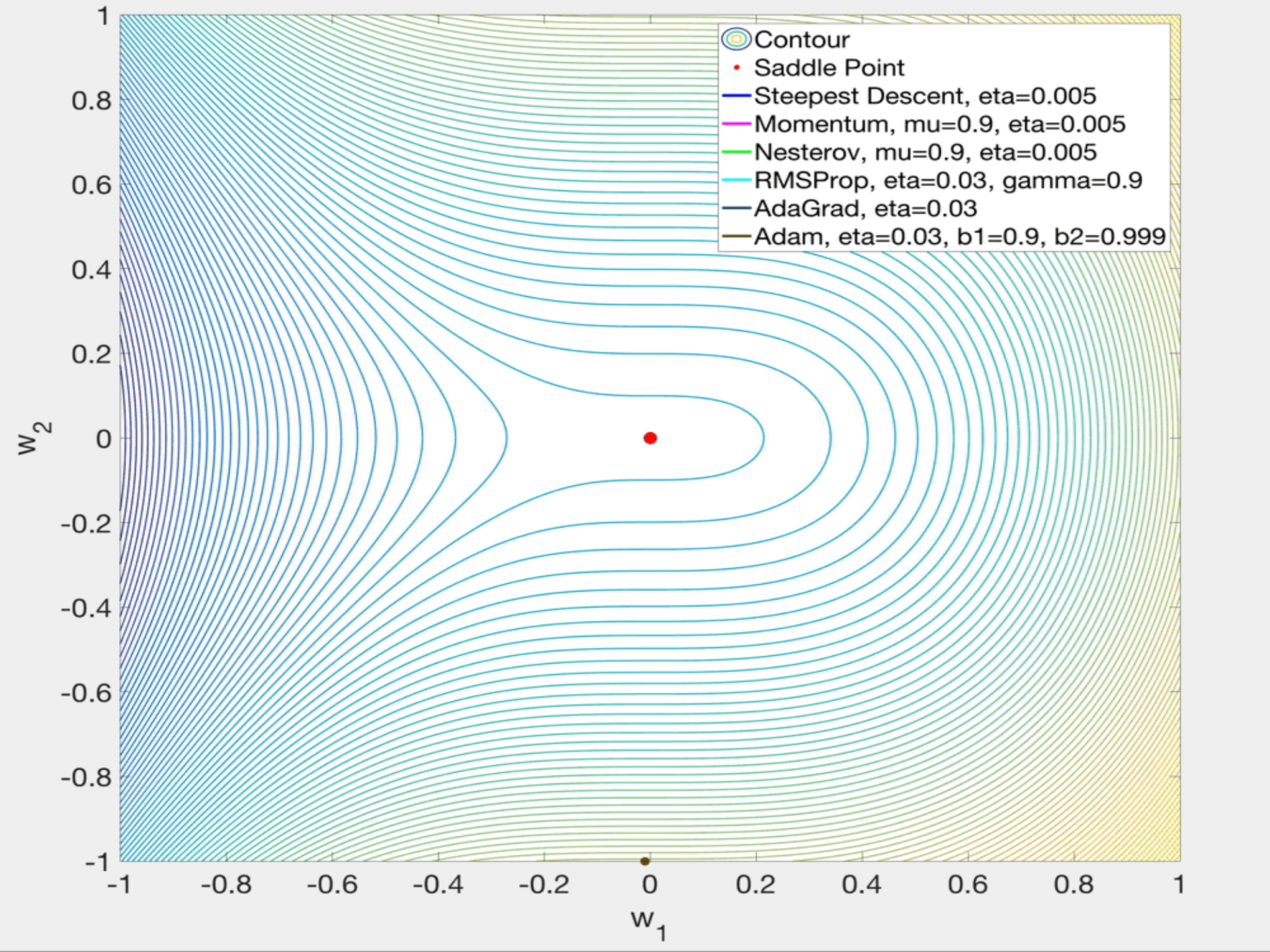
Step 4. $\hat{v}^{(t+1)} = \frac{1}{1-\beta_2^t} \cdot v^{(t+1)}$

Bias Correction

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

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


Contour
Optimal
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Nesterov, mu=0.9, eta=0.005
RMSProp, eta=0.03, gamma=0.9
AdaGrad, eta=0.03
Adam, eta=0.03, b1=0.9, b2=0.999
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1


Contour
Saddle Point
Steepest Descent, eta=0.005
Momentum, mu=0.9, eta=0.005
Nesterov, mu=0.9, eta=0.005
RMSProp, eta=0.03, gamma=0.9
AdaGrad, eta=0.03
Adam, eta=0.03, b1=0.9, b2=0.999
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
w2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
w1

Adam (Adaptive Moment Estimation)

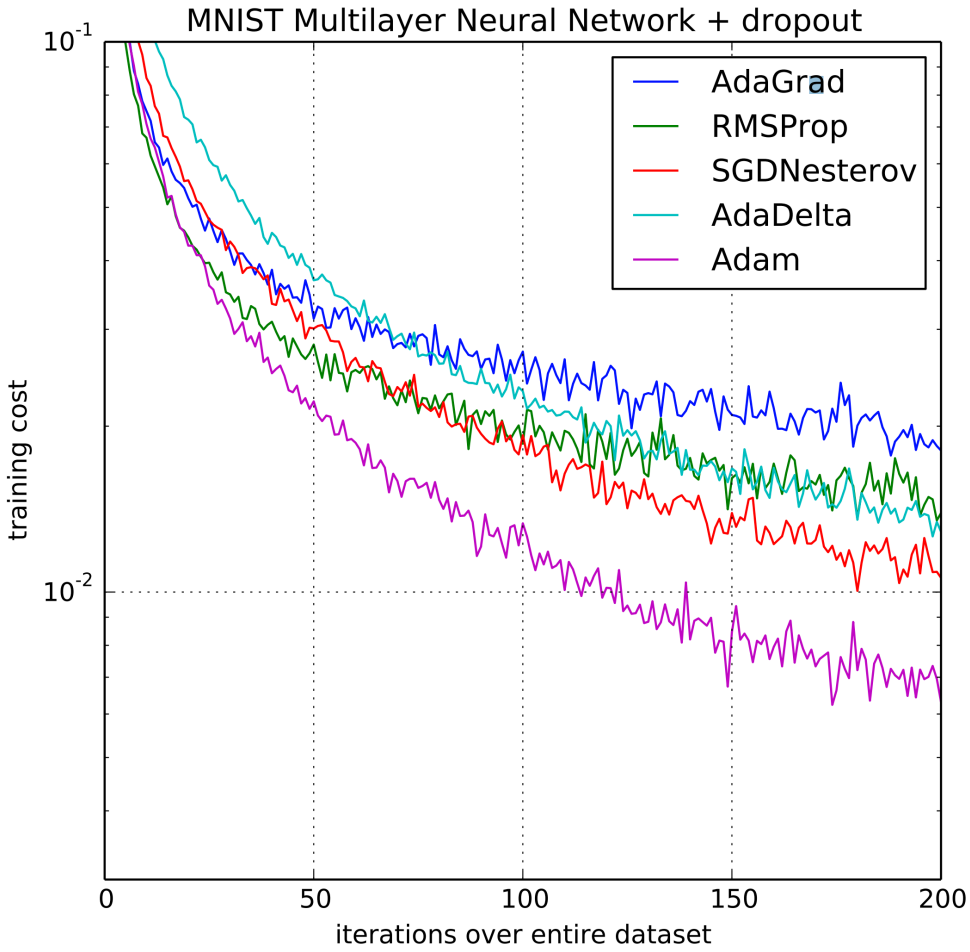


Image credit: Kingma & Ba, 2015

Network configurations:

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

In practice, Adam is the default choice to use. [Fei-Fei Li, Andrej Karpathy, Justin Johnson]