Variations on Stochastic Gradient Descent

Computational Statistics

Johan Larsson

Department of Mathematical Sciences, University of Copenhagen

October 22, 2024

Errata

- There is rationale for step size differences in least squared loss and log-likelihood loss: gradients are larger in former.

Last Time

SGD
Introduced stochastic gradient descent (SGD) and mini-batch version thereof.

Last Time

SGD
Introduced stochastic gradient descent (SGD) and mini-batch version thereof.

Problems
We indicated that there were problems with vanilla SGD: poor convergence, erratic behavior.

Algorithm 1: Mini-Batch SGD

Data: $\gamma_0 > 0$
for $k \leftarrow 1, 2, \ldots$ **do**
  $A_k \leftarrow$ random mini-batch of m samples;
  $x_k \leftarrow x_{k-1} - \frac{\gamma_k}{|A_k|} \sum_{i \in A_k} \nabla f_i(x_{k-1})$;

Today

How can we improve stochastic gradient descent?

Momentum
Base update on combination of gradient step and previous point.

Two versions: Polyak and Nesterov momentum

Today

How can we improve stochastic gradient descent?

Momentum

Base update on combination of gradient step and previous point.

Two versions: Polyak and Nesterov momentum

Adaptive Gradients

Adapt learning rate to particular feature.

Momentum

Basic Idea
Give the particle **momentum**: like a heavy ball

Not specific to stochastic GD!

Momentum

Basic Idea

Give the particle **momentum**: like a heavy ball

Not specific to stochastic GD!

Polyak Momentum

Classical version (Polyak 1964)

$\mu \in [0, 1)$ decides strength of momentum; $\mu = 0$ gives standard gradient descent

Guaranteed convergence for quadratic functions

Algorithm 2: GD with Polyak Momentum

Data: $\gamma > 0$, $\mu \in [0, 1)$, $x_{-1} = x_0$

for $k \leftarrow 1, 2, \ldots$ **do**

  $x_k \leftarrow x_{k-1} - \gamma \nabla f(x_{k-1}) + \mu(x_{k-1} - x_{k-2})$;

Polyak Momentum in Practice

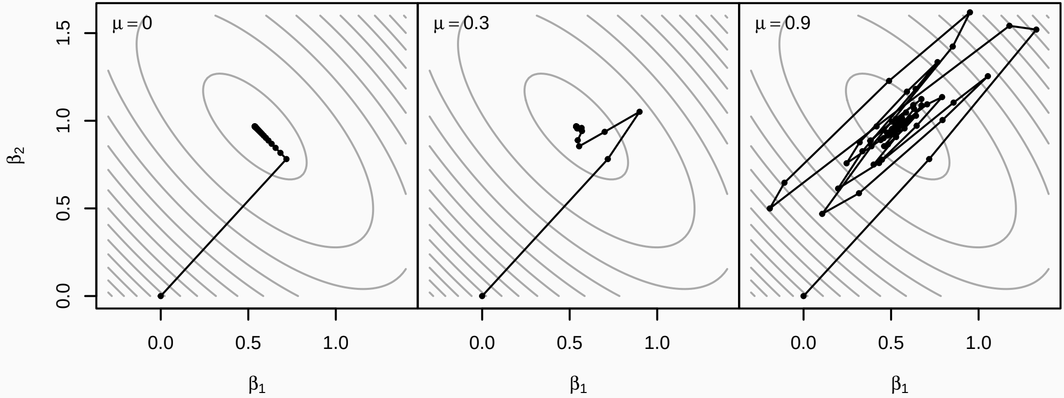

Figure 1: Trajectories of GD for different momentum values for a least-squares problem

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

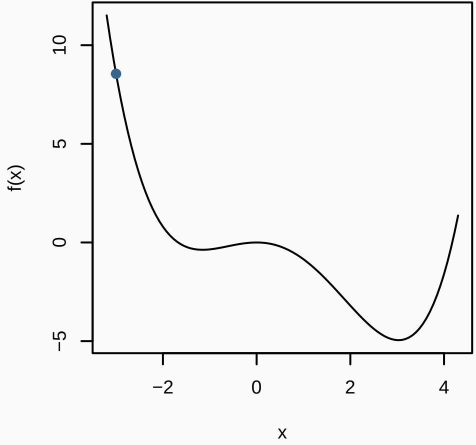

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

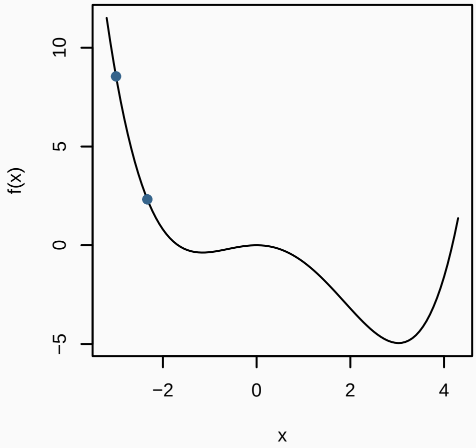

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

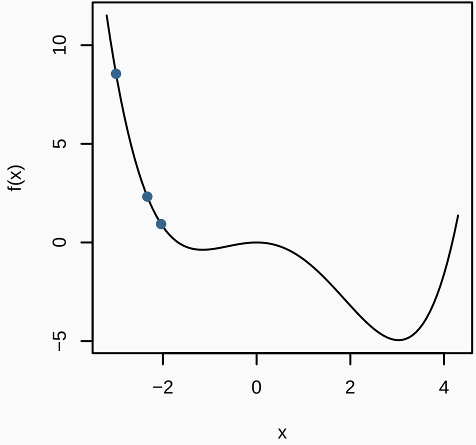

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

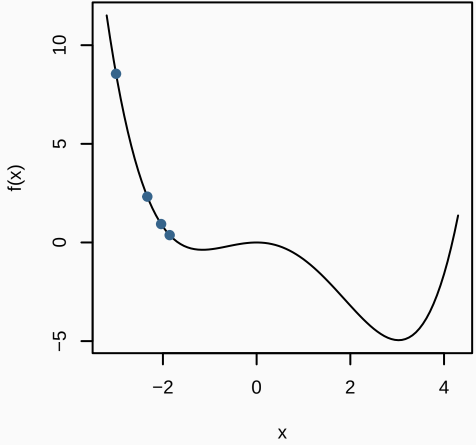

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

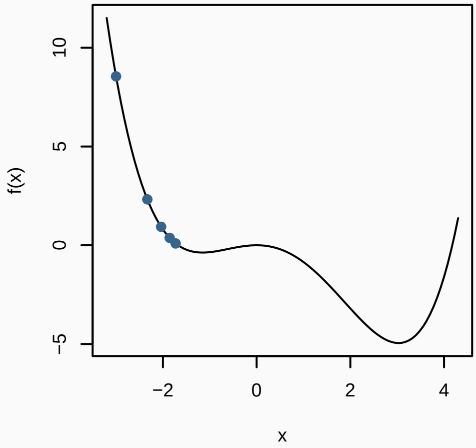

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

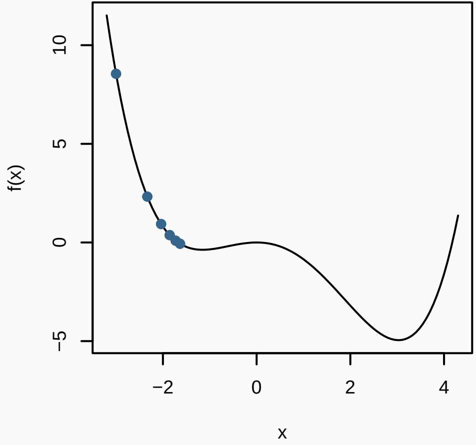

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

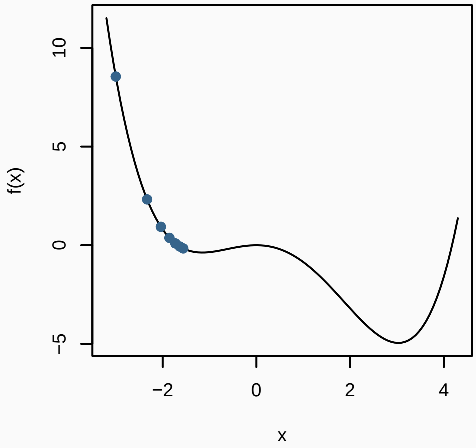

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

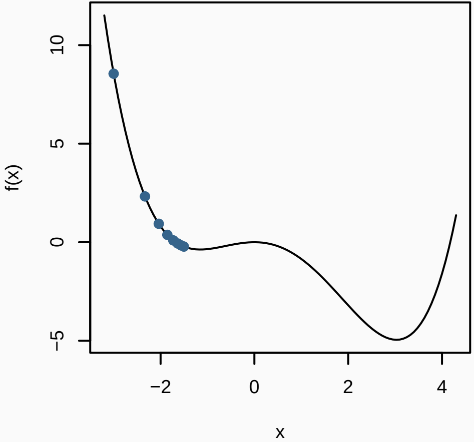

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

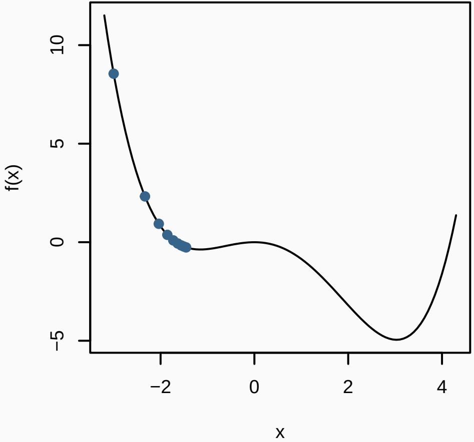

Figure 2: $\mu = 0$

Local Minima

For nonconvex f, gradient descent may get stuck at a local minimum.

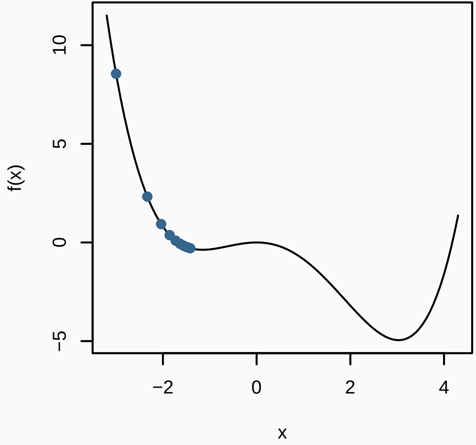

Figure 2: $\mu = 0$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

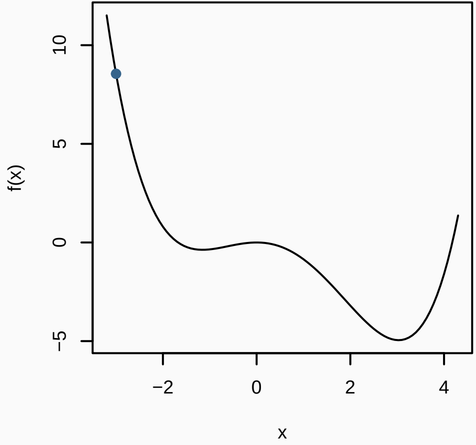

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

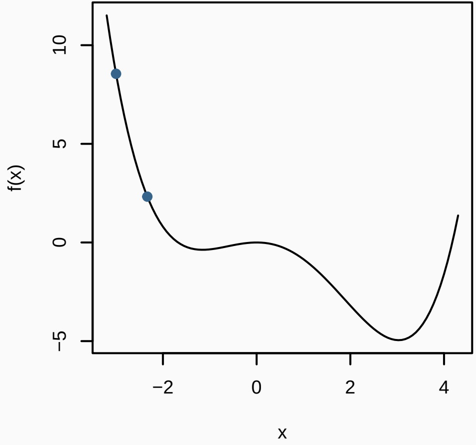

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

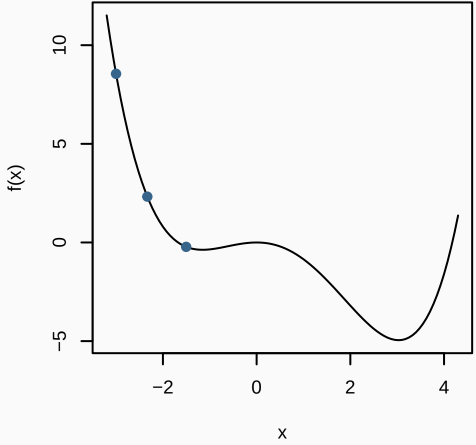

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

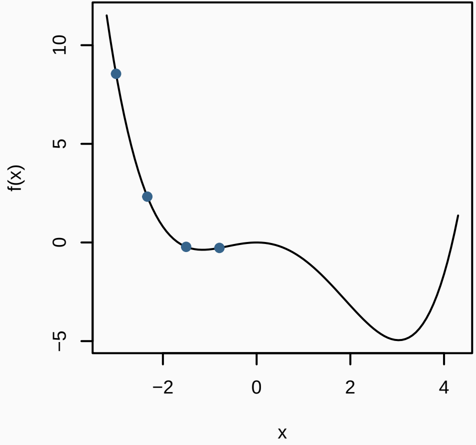

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

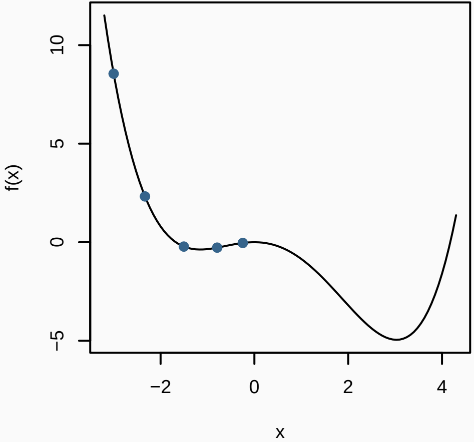

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

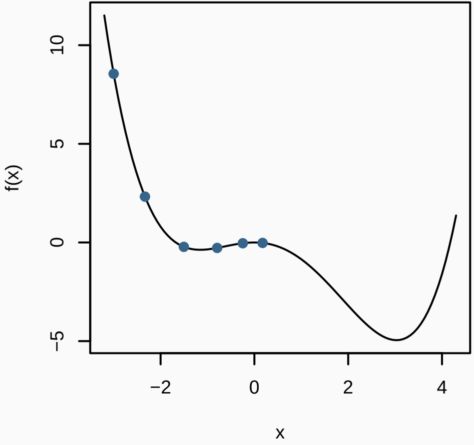

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

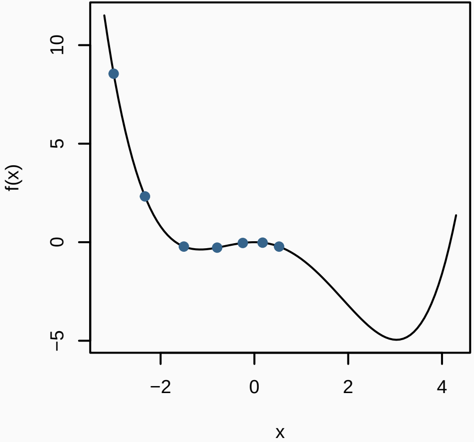

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

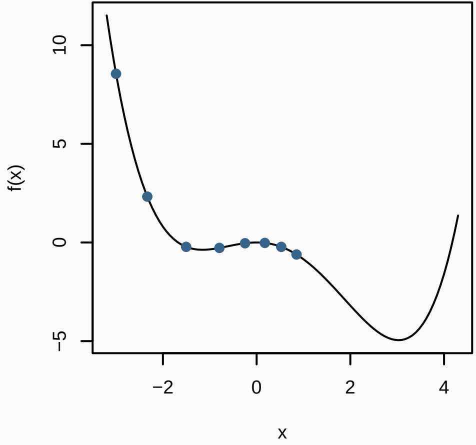

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

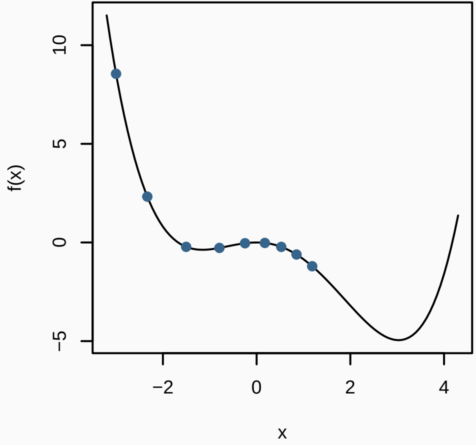

Figure 3: $\mu = 0.8$

Escaping Local Minima

With momentum, we can (sometimes) remedy this problem.

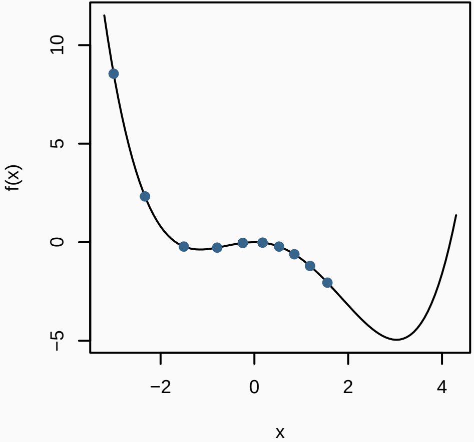

Figure 3: $\mu = 0.8$

Convergence Failure

For some problems, the momentum method may fail to converge (Lessard, Recht, and Packard 2016).

Convergence Failure

For some problems, the momentum method may fail to converge (Lessard, Recht, and Packard 2016).

Consider

$$f(x) = \begin{cases} \frac{25x^2}{2} & \text{if } x < 1, \\ \frac{x^2}{2} + 24x - 12 & \text{if } 1 \leq x < 2, \\ \frac{25x^2}{2} - 24x + 36 & \text{if } x \geq 2. \end{cases}$$

Convergence Failure

For some problems, the momentum method may fail to converge (Lessard, Recht, and Packard 2016).

Consider

$$f(x) = \begin{cases} \frac{25x^2}{2} & \text{if } x < 1, \\ \frac{x^2}{2} + 24x - 12 & \text{if } 1 \leq x < 2, \\ \frac{25x^2}{2} - 24x + 36 & \text{if } x \geq 2. \end{cases}$$

For an "optimal" step size $1 = 1/L$ with $L = 25$, GD momentum steps converge to three limit points.

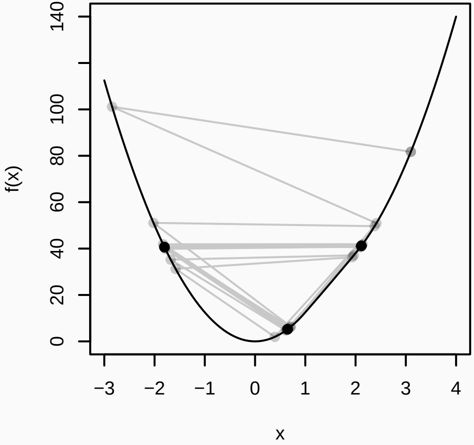

Figure 4: Initialized at $x_0 = 3.2$, the algorithm fails to converge.

Nesterov Momentum

Algorithm 3: GD with Nesterov Momentum

Data: $\gamma > 0$, $\mu \in [0, 1)$

for $i \leftarrow 1, 2, \dots$ **do**

$\quad v_k \leftarrow x_{k-1} - \gamma \nabla f(x_{k-1})$;

$\quad x_k \leftarrow v_k + \mu(v_k - v_{k-1})$;

Overcomes convergence problem of classical (Polyak) momentum.

Nesterov: Sutskever Perspective

Consider two iterations of Nesterov algorithm:

$$
\begin{aligned}
v_k &= x_{k-1} - \gamma \nabla f\left(x_{k-1}\right) \\
x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\
v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right) \\
x_{k+1} &= v_{k+1} + \mu\left(v_{k+1} - v_k\right)
\end{aligned}
$$

Nesterov: Sutskever Perspective

Consider two iterations of Nesterov algorithm:

$$
\begin{aligned}
v_k &= x_{k-1} - \gamma \nabla f\left(x_{k-1}\right) \\
x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\
v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right) \\
x_{k+1} &= v_{k+1} + \mu\left(v_{k+1} - v_k\right)
\end{aligned}
$$

Idea: focus on interim step:

$$
\begin{aligned}
x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\
v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right)
\end{aligned}
$$

Nesterov: Sutskever Perspective

Consider two iterations of Nesterov algorithm:

$$
\begin{aligned}
v_k &= x_{k-1} - \gamma \nabla f\left(x_{k-1}\right) \\
x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\
v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right) \\
x_{k+1} &= v_{k+1} + \mu\left(v_{k+1} - v_k\right)
\end{aligned}
$$

Idea: focus on interim step:

$$
\begin{aligned}
x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\
v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right)
\end{aligned}
$$

Reindex:

$$
\begin{aligned}
x_k &= v_{k-1} + \mu\left(v_{k-1} - v_{k-2}\right) \\
v_k &= x_k - \gamma \nabla f\left(x_k\right)
\end{aligned}
$$

Nesterov: Sutskever Perspective

Consider two iterations of Nesterov algorithm:

$$\begin{aligned} v_k &= x_{k-1} - \gamma \nabla f\left(x_{k-1}\right) \\ x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\ v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right) \\ x_{k+1} &= v_{k+1} + \mu\left(v_{k+1} - v_k\right) \end{aligned}$$

Idea: focus on interim step:

$$\begin{aligned} x_k &= v_k + \mu\left(v_k - v_{k-1}\right) \\ v_{k+1} &= x_k - \gamma \nabla f\left(x_k\right) \end{aligned}$$

Reindex:

$$\begin{aligned} x_k &= v_{k-1} + \mu\left(v_{k-1} - v_{k-2}\right) \\ v_k &= x_k - \gamma \nabla f\left(x_k\right) \end{aligned}$$

But since $x_k = v_k$ for $k = 1$ by construction, we can swap x_k for v_k and get the update

$$x_k = x_{k-1} + \mu(x_{k-1} - x_{k-2}) - \gamma \nabla f(x_k + \mu(x_{k-1} - x_{k-2})).$$

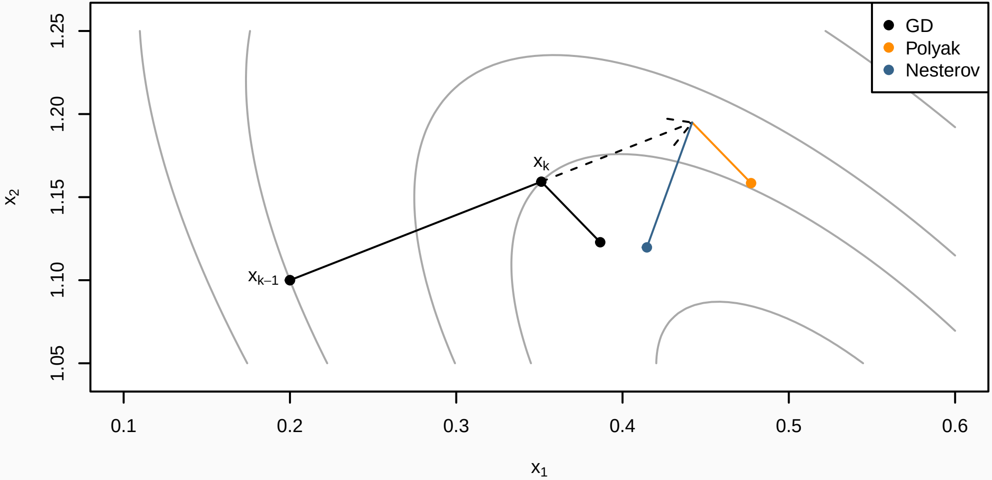

Figure 5: Illustration of Nesterov and Polyak momentum

Optimal Momentum

For gradient descent with $\gamma = 1/L$, the optimal choice of μ_k for general convex and smooth f is

$$\mu_k = \frac{a_{k-1} - 1}{a_k}$$

for a series of

$$a_k = \frac{1 + \sqrt{4a_{k-1}^2 + 1}}{2}$$

with $a_0 = 1$ (and hence $\mu_1 = 0$).

Optimal Momentum

For gradient descent with $\gamma = 1/L$, the optimal choice of μ_k for general convex and smooth f is

$$\mu_k = \frac{a_{k-1} - 1}{a_k}$$

for a series of

$$a_k = \frac{1 + \sqrt{4a_{k-1}^2 + 1}}{2}$$

with $a_0 = 1$ (and hence $\mu_1 = 0$).

First step ($k = 1$) is just standard gradient descent

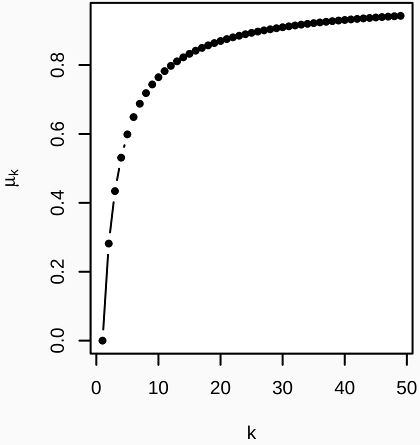

Figure 6: Optimal momentum for Nesterov acceleration (for GD).

Convergence

Convergence rate with Nesterov acceleration goes from $O(1/k)$ to $O(1/k^2)$

Convergence

Convergence rate with Nesterov acceleration goes from $O(1/k)$ to $O(1/k^2)$

This is optimal for a first-order method.

Convergence

Convergence rate with Nesterov acceleration goes from $O(1/k)$ to $O(1/k^2)$

This is optimal for a first-order method.

Convergence improves further for quadratic and strongly convex!

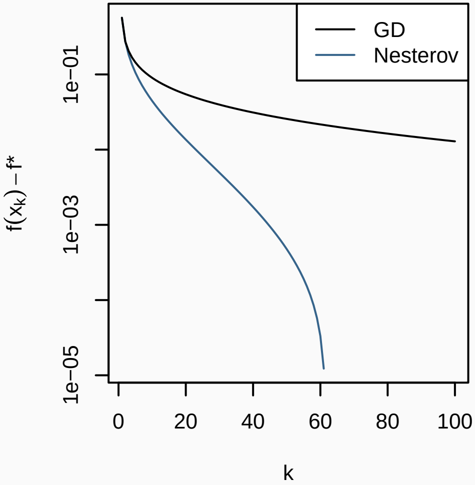

Figure 7: Suboptimality plot for a logistic regression problem with $n = 1\,000$, $p = 100$.

Exercise: Rosenbrock's Banana

Steps

1. Minimize
$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2$$
with $a = 1$ and $b = 100$ using GD with Polyak momentum. Optimum is (a, a^2).
2. Implement gradient descent.
3. Add Polyak momentum.

Exercise: Rosenbrock's Banana

Steps

1. Minimize
$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2$$
with $a = 1$ and $b = 100$ using GD with Polyak momentum. Optimum is (a, a^2).
2. Implement gradient descent.
3. Add Polyak momentum.

Algorithm 4: GD with Polyak Momentum

Data: $\gamma > 0$, $\mu \in [0, 1)$
for $k \leftarrow 1, 2, \ldots$ **do**
  $x_k \leftarrow x_{k-1} - \gamma \nabla f(x_{k-1}) + \mu(x_{k-1} - x_{k-2})$;

Exercise: Rosenbrock's Banana

Steps

1. Minimize
$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2$$
with $a = 1$ and $b = 100$ using GD with Polyak momentum. Optimum is (a, a^2).
2. Implement gradient descent.
3. Add Polyak momentum.

Algorithm 4: GD with Polyak Momentum

Data: $\gamma > 0$, $\mu \in [0, 1)$

for $k \leftarrow 1, 2, \dots$ **do**

$\quad x_k \leftarrow x_{k-1} - \gamma \nabla f(x_{k-1}) + \mu(x_{k-1} - x_{k-2})$;

Plot Contours

```
x1 <- seq(-2, 2, length.out =
    100)
x2 <- seq(-1, 3, length.out =
    100)
z <- outer(x1, x2, f)
contour(x1, x2, z, nlevels =
    20)
```

Exercise: Rosenbrock's Banana

Steps

1. Minimize
$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2$$
with $a = 1$ and $b = 100$ using GD with Polyak momentum. Optimum is (a, a^2).
2. Implement gradient descent.
3. Add Polyak momentum.

Algorithm 4: GD with Polyak Momentum

Data: $\gamma > 0$, $\mu \in [0, 1)$

for $k \leftarrow 1, 2, \ldots$ **do**

$\quad x_k \leftarrow x_{k-1} - \gamma \nabla f(x_{k-1}) + \mu(x_{k-1} - x_{k-2})$;

Plot Contours

```
x1 <- seq(-2, 2, length.out =
    100)
x2 <- seq(-1, 3, length.out =
    100)
z <- outer(x1, x2, f)
contour(x1, x2, z, nlevels =
    20)
```

What About SGD?

So far, we have mostly talked about standard GD, but we can use momentum (Polyak or Nesterov) for SGD as well.

For standard GD, Nesterov is the dominating method for achieving acceleration; for SGD, Polyak momentum is actually quite common.

In term of convergence, all bets are now off.

No optimal rates anymore, just heuristics.

Adaptive Gradients

General Idea

Some directions may be important, but feature information is sparse.

General Idea

Some directions may be important, but feature information is sparse.

AdaGrad (Duchi, Hazan, and Singer 2011)

Store matrix of gradient history,

$$G_k = \sum_{i=1}^{k} \nabla f(x_k) \nabla f(x_k)^\intercal,$$

and update by multiplying gradient with $G_k^{-1/2}$.

Algorithm 5: AdaGrad

Data: $\gamma > 0$, $G = 0$

for $k \leftarrow 1, 2, \ldots$ **do**

$G_k \leftarrow G_{k-1} + \nabla f(x_{k-1}) \nabla f(x_{k-1})^\intercal$;

$x_k \leftarrow x_{k-1} - \gamma G_k^{-1/2} \nabla f(x_{k-1})$;

Adaptive Gradients

General Idea

Some directions may be important, but feature information is sparse.

AdaGrad (Duchi, Hazan, and Singer 2011)

Store matrix of gradient history,

$$G_k = \sum_{i=1}^{k} \nabla f(x_k) \nabla f(x_k)^\intercal,$$

and update by multiplying gradient with $G_k^{-1/2}$.

Algorithm 5: AdaGrad

Data: $\gamma > 0$, $G = 0$
for $k \leftarrow 1, 2, \ldots$ **do**
  $G_k \leftarrow G_{k-1} + \nabla f(x_{k-1}) \nabla f(x_{k-1})^\intercal$;
  $x_k \leftarrow x_{k-1} - \gamma G_k^{-1/2} \nabla f(x_{k-1})$;

Effects

Larger learning rates for sparse features

Step-sizes adapt to curvature.

AdaGrad In Practice

Simplified Version

Computing $\nabla f \nabla f^\intercal$ is $O(p^2)$; expensive!

Replace $G_k^{-1/2}$ with $\mathrm{diag}(G_k)^{-1/2}$

AdaGrad In Practice

Simplified Version

Computing $\nabla f \nabla f^\intercal$ is $O(p^2)$; expensive!

Replace $G_k^{-1/2}$ with $\mathrm{diag}(G_k)^{-1/2}$

Avoid Singularities

Add a small ϵ to diagonal.

AdaGrad In Practice

Simplified Version

Computing $\nabla f \nabla f^\intercal$ is $O(p^2)$; expensive!

Replace $G_k^{-1/2}$ with $\mathrm{diag}(G_k)^{-1/2}$

Avoid Singularities

Add a small ϵ to diagonal.

Algorithm 6: Simplified AdaGrad

Data: $\gamma > 0$, $\epsilon > 0$

for $k \leftarrow 1, 2, \ldots$ **do**

$\quad G_k \leftarrow G_{k-1} + \mathrm{diag}\left(\nabla f(x_{k-1})^2\right)$;

$\quad x_k \leftarrow x_{k-1} - \gamma\, \mathrm{diag}\left(\epsilon I_p + G_k\right)^{-1/2} \nabla f(x_{k-1})$;

RMSProp

Acronym for **R**oot **M**ean **S**quare **Prop**agation (Hinton 2018)

Idea
Divide learning rate by running average of magnitude of recent gradients:

$$v(x,k) = \xi v(x,k-1) + (1-\xi)\nabla f(x_k)^2$$

where ξ is the forgetting factor.

Similar to AdaGrad, but uses forgetting to gradually decrease influence of old data.

Algorithm 7: RMSProp

Data: $\gamma > 0$, $\xi > 0$
for $k \leftarrow 1, 2, \ldots, \xi \in [0, 1)$ **do**
  $v_k = \xi v_{k-1} + (1-\xi)\nabla f(x_{k-1})$;
  $x_k \leftarrow x_{k-1} - \frac{\gamma}{\sqrt{v_k}} \odot \nabla f(x_{k-1})$;

Adam

Acronym for **Ada**ptive **m**oment estimation (Kingma and Ba 2015)

Basically RMSProp + *momentum* (for both gradients and second moments theorof)

Popular and still in much use today.

Implementation Aspects of SGD

Loops

Any language (e.g. R) that imposes overhead for loops, will have a difficult time with SGDS

Implementation Aspects of SGD

Loops

Any language (e.g. R) that imposes overhead for loops, will have a difficult time with SGDS

Storage Order

In a regression setting, when indexing a single obervation at a time, slicing rows is not efficient n is large.

We can either transpose first or use a row-major storage order (not possible in R).

Example: Nonlinear Least Squares

Let's assume we're trying to solve a least-squares type of problem:

$$f(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - g(\theta; x_i, y_i))^2$$

with $\theta = (\alpha, \beta)$ and

$$g(\theta; x, y) = \alpha \cos(\beta x).$$

Example: Nonlinear Least Squares

Let's assume we're trying to solve a least-squares type of problem:

$$f(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - g(\theta; x_i, y_i))^2$$

with $\theta = (\alpha, \beta)$ and

$$g(\theta; x, y) = \alpha \cos(\beta x).$$

Then

$$\nabla_\theta f(\theta) = \begin{bmatrix} \cos(\beta x) \\ -\alpha x \sin(\beta x) \end{bmatrix}.$$

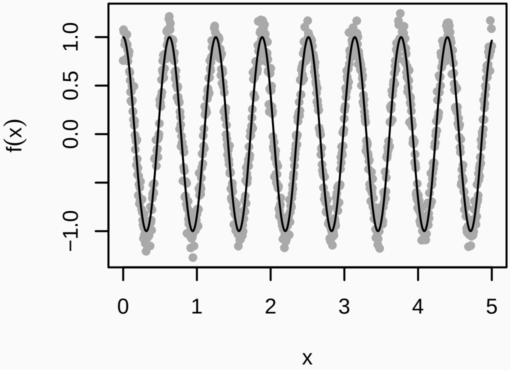

Figure 8: Simulation from problem

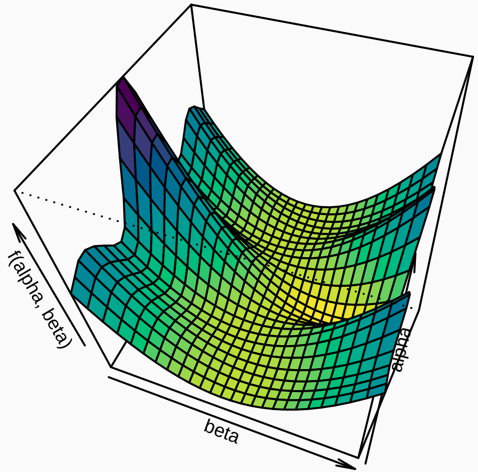

Figure 9: Perspective plot of function

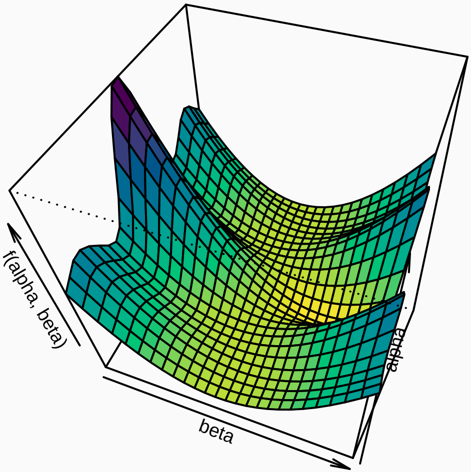

Figure 9: Perspective plot of function

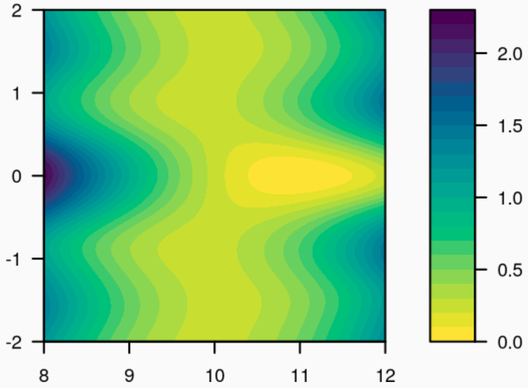

Figure 10: Contour plot of f

Variants

We will consider three variants:

- Batch gradient descent
- Batch gradient descent with momentum
- Adam

In each case, we'll use a batch of size $m = 50$.

We initialize at different starting values and see how well the algorithm converges.

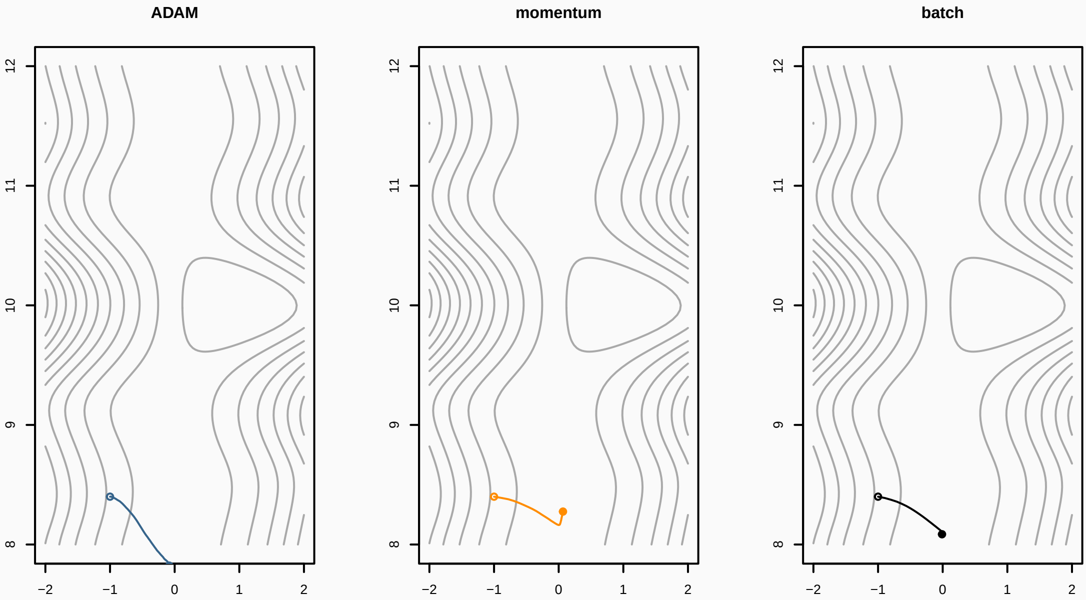

Figure 11:

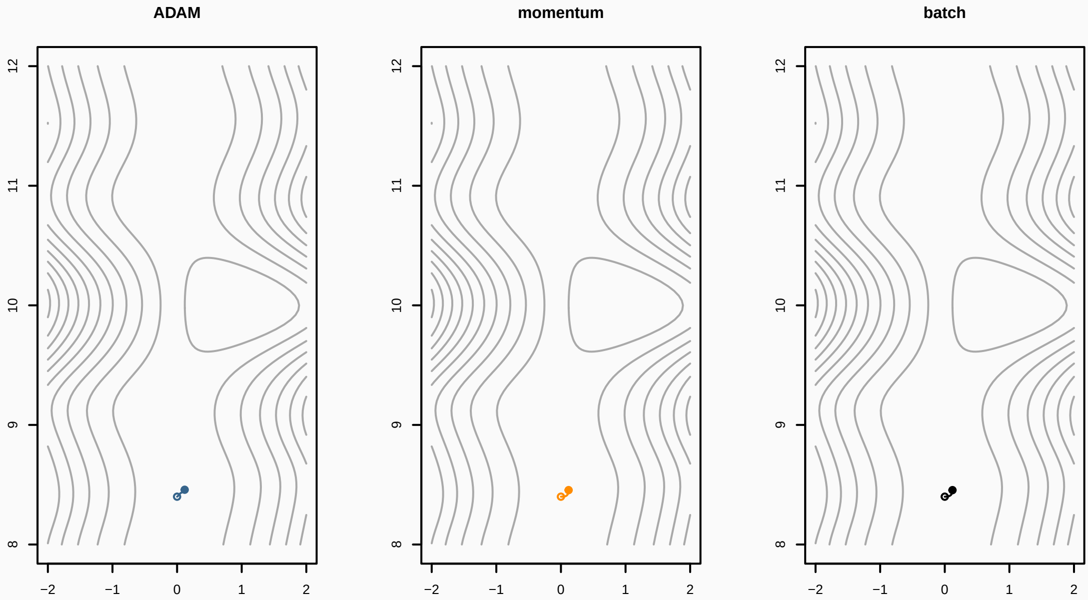

Figure 11:

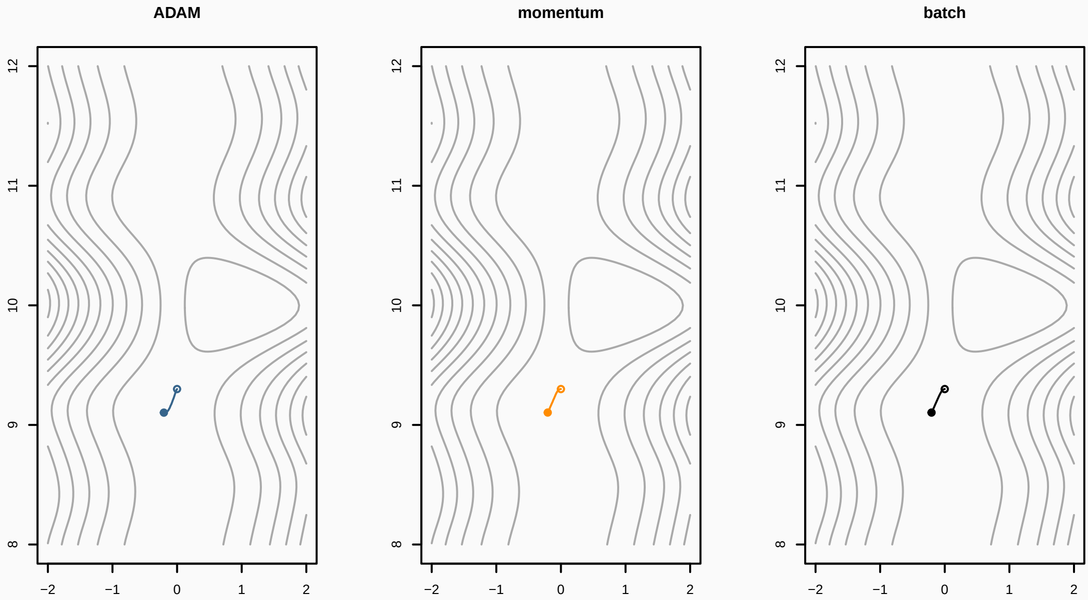

Figure 11:

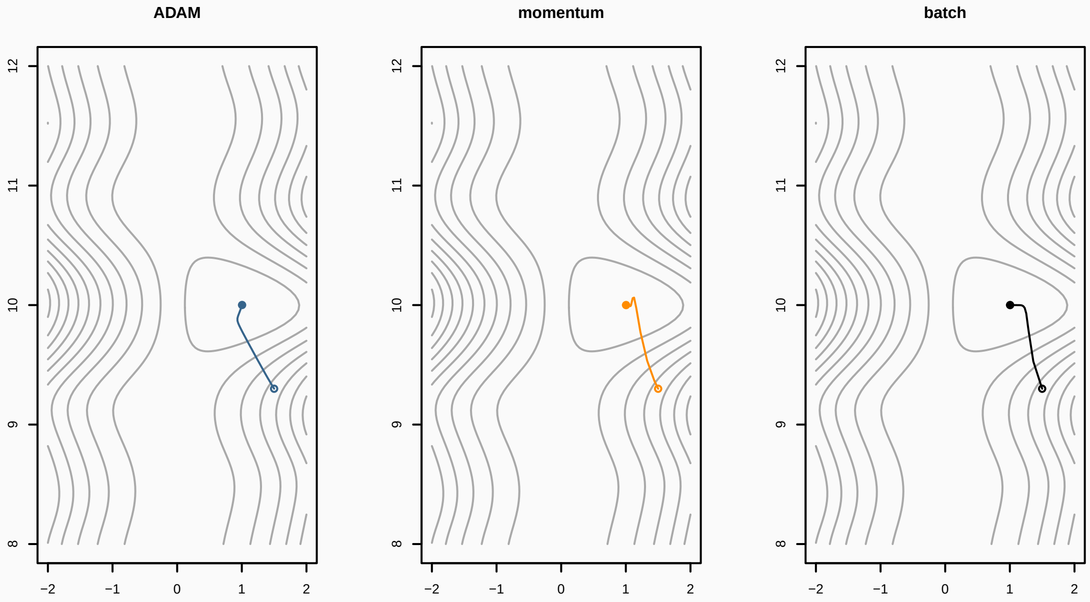

Figure 11:

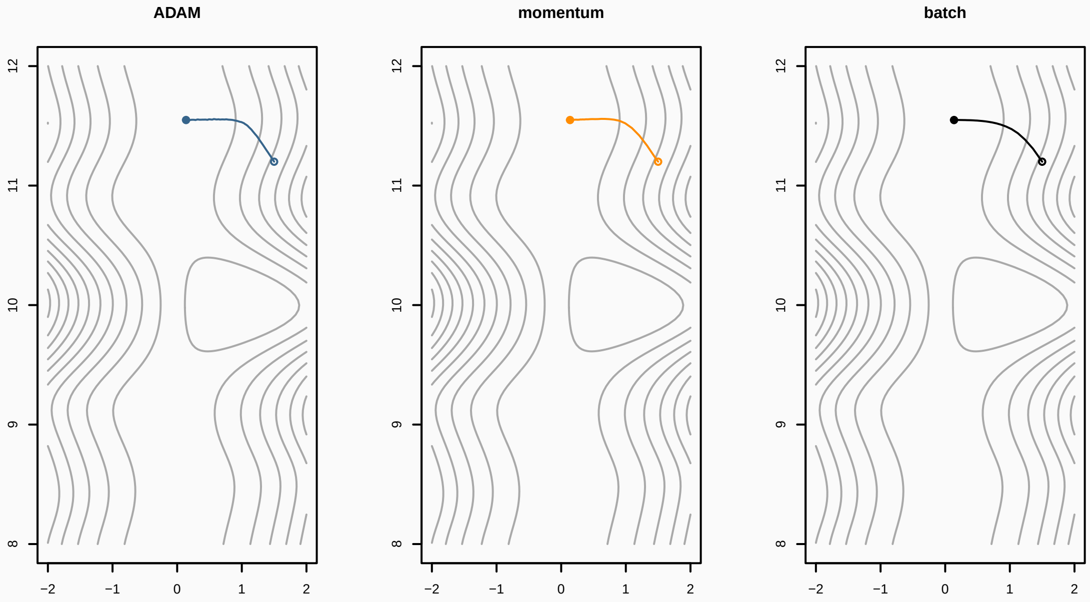

Figure 11:

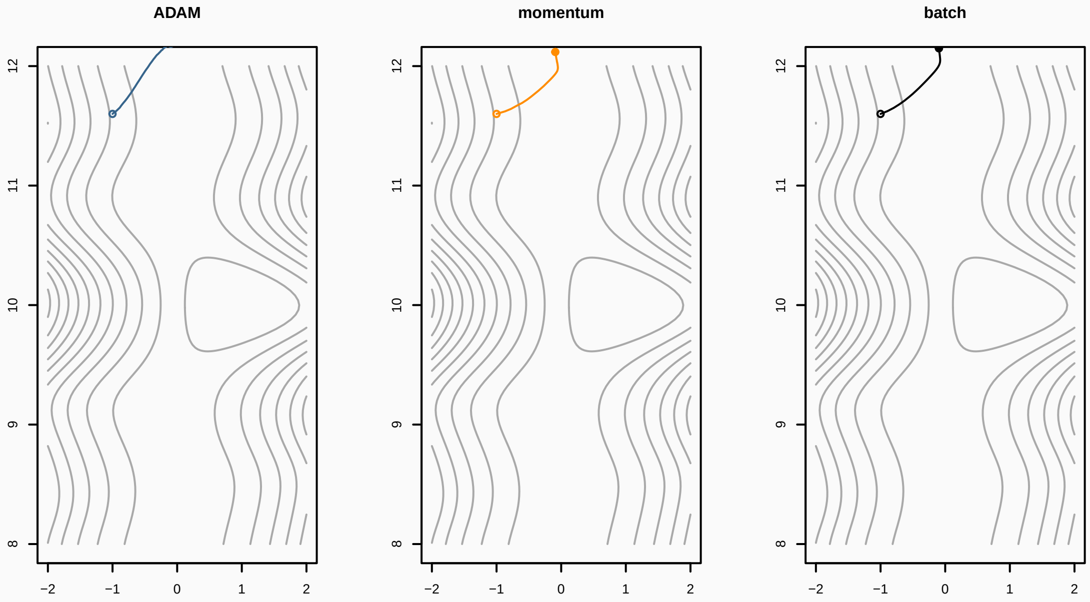

Figure 11:

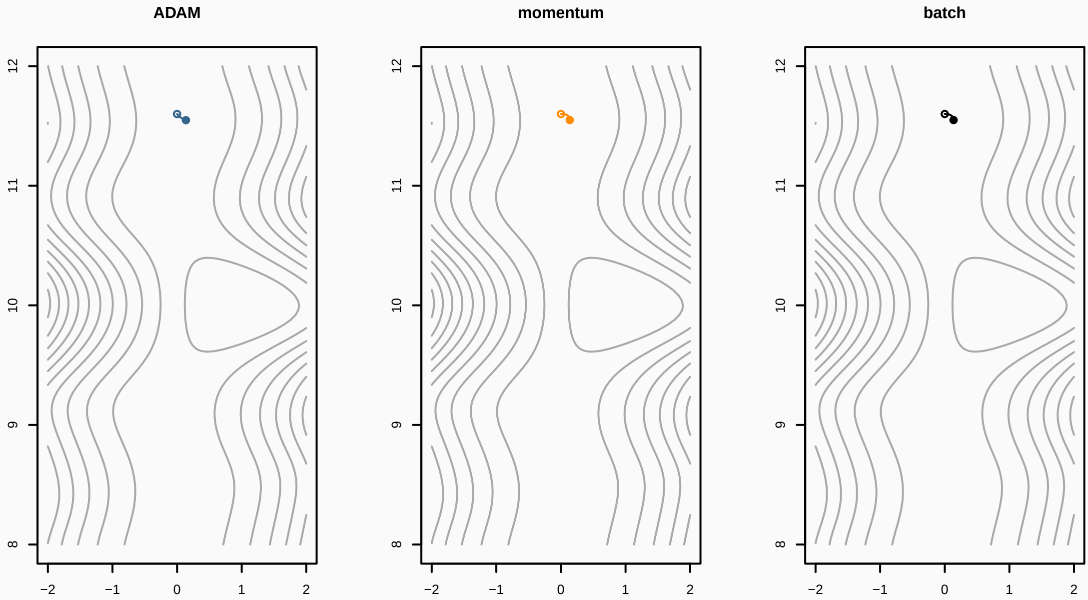

Figure 11:

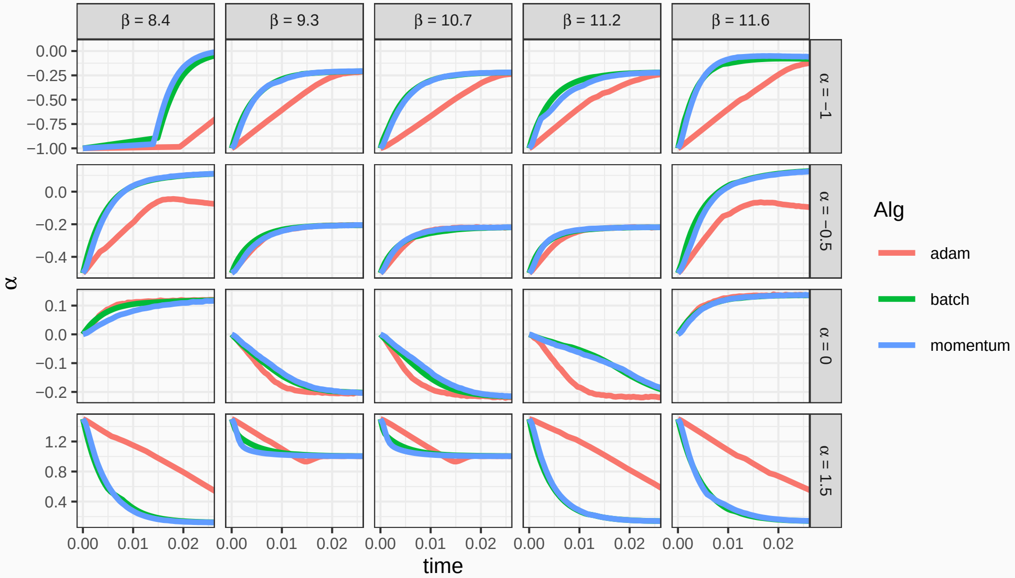

Figure 12: Updates of α parameter over time for the different algorithms over different starting values.

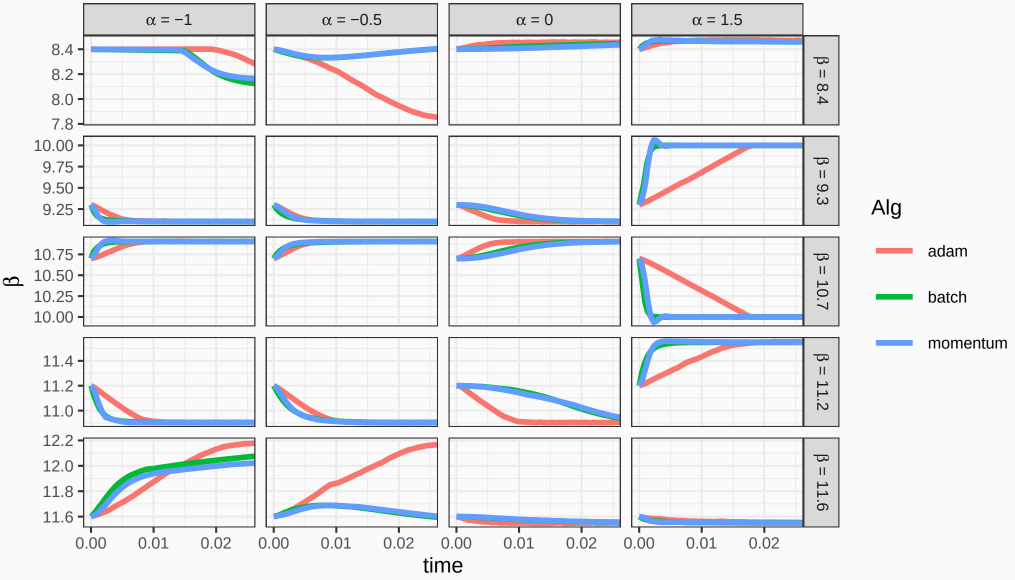

Figure 13: Updates of β parameter over time for the different algorithms over different starting values.

Rcpp

Very attractive for stochastic methods due to all the loop constructs and slicing.

However, Rcpp lacks linear algebra functions.

Approaches

- Still use only Rcpp (but then you need to write your own linear algebra functions[1]
- Use RcppEigen or RcppArmadillo.

[1] Not recommended!

Exercise: Rosenbrock Revisited

Steps

1. Convert your gradient descent algorithm to C++ through Rcpp.
2. Modify it to be a stochastic gradient descent algorithm instead.

Exercise: Rosenbrock Revisited

Steps

1. Convert your gradient descent algorithm to C++ through Rcpp.
2. Modify it to be a stochastic gradient descent algorithm instead.

Hints

- Use the Rcpp function `Rcpp::sugar()` to sample indices.
- Don't bother with a stopping criterion to begin with; just set a maximum number of iterations.
- You can return a list by calling `Rcpp::List::create(Rcpp::named("name") = x)`.
- Use a pure Rcpp implementation.

Summary

We introduced several new concepts:

- Polyak momentum,
- Nesterov acceleration (momentum), and
- adapative gradients (AdaGrad),

We practically implemented versions of gradient descent and stochastic gradient descent with momentum.

Summary

We introduced several new concepts:

- Polyak momentum,
- Nesterov acceleration (momentum), and
- adapative gradients (AdaGrad),

We practically implemented versions of gradient descent and stochastic gradient descent with momentum.

Additional Resources

- Goh (2017) is a article on momentum in gradient descent with lots of interactive visualizations.

Next Time

Reproducibility

How to make your code reproducible

We build an R package.

Next Time

Reproducibility

How to make your code reproducible

We build an R package.

Summary

We summarize the course.

Next Time

Reproducibility

How to make your code reproducible

We build an R package.

Summary

We summarize the course.

Exam Advice

We talk about the upcoming oral examinations.

Thank you!

References i

- Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61, pp. 2121–2159. ISSN: 1533-7928. URL: http://jmlr.org/papers/v12/duchi11a.html (visited on 10/22/2024).
- Goh, Gabriel (2017). "Why Momentum Really Works". In: *Distill*. DOI: 10.23915/distill.00006. URL: http://distill.pub/2017/momentum.
- Hinton, Geoffrey (2018). "Lecture 6". Oral Presentation. Coursera Course: Neural Networks for Machine Learning (Virtual). URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980 (visited on 10/22/2024).

References ii

Lessard, Laurent, Benjamin Recht, and Andrew Packard (Jan. 2016). "Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints". In: *SIAM Journal on Optimization* 26.1, pp. 57–95. ISSN: 1052-6234. DOI: 10.1137/15M1009597. URL: https://epubs.siam.org/doi/abs/10.1137/15M1009597 (visited on 10/17/2024).

Polyak, B. T. (Jan. 1, 1964). "Some Methods of Speeding up the Convergence of Iteration Methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5, pp. 1–17. ISSN: 0041-5553. DOI: 10.1016/0041-5553(64)90137-5. URL: https://www.sciencedirect.com/science/article/pii/0041555364901375 (visited on 10/22/2024).