Numerical Optimization

Nan Ye

School of Mathematics and Physics
The University of Queensland

Schedule

A tentative schedule is available on BlackBoard

- Week 1-2: machine learning basics
- Week 3-4: neural network basics
- Week 5-6: deep architectures
- Week 7-8: optimization
- Week 8-10: improving generalization
- Week 10-11: unsupervised learning
- Week 12: reinforcement learning

Recall: Learning as Optimization

- Many learning problems are directly formulated as an optimization problem.
  - e.g. learning a Bernoulli/Gaussian distribution
  - e.g. OLS, ridge regression
  - e.g. naive Bayes classifier, logistic regression, SVM
- Some of the optimization problems have closed-form solutions, but many do not.
  - e.g. logistic regression and SVM do not have closed-form solutions
- We often need to use numerical methods to solve optimization problems in machine learning.

Numerical Optimization

$$\begin{aligned} \min \quad & f(\mathbf{w}) \\ \text{s.t.} \quad & h_i(\mathbf{w}) \leq b_i, i = 1, \ldots, m, \end{aligned}$$

- $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbf{R}^d$: optimization variable.
- $f : \mathbf{R}^d \to \mathbf{R}$: the objective function.
- $h_i : \mathbf{R}^d \to \mathbf{R}$: (inequality) constraint functions.
- Feasible set: the set of points satisfying all constraints.

Numerical Optimization in Learning

Recall: Learning is...

- Collect some data, e.g. coin flips.
- Choose a hypothesis class, e.g. Bernoulli distribution.
- Choose a loss function, e.g. negative log-likelihood.
- Choose an optimization procedure, e.g. set derivative to 0.
- Regularization may be used to encode prior knowledge.

The optimization problem is often a numerical optimization problem.

- Some examples

OLS
$$\min_{\beta} \sum_i (\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2$$

Logistic regression
$$\min_{\mathbf{w}} \sum_i -\ln \frac{e^{\mathbf{w}_{y_i}^\top \mathbf{x}_i}}{\sum_{y'} e^{\mathbf{w}_{y'}^\top \mathbf{x}_i}}$$

Ridge regression
$$\sum_i (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$$

SVM
$$\min_{\mathbf{w}, w_0} \frac{1}{2} ||\mathbf{w}||_2^2$$
$$\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1, \quad i = 1, \ldots, n.$$

- When we train a neural network $f(\mathbf{x}; \mathbf{w})$ parametrized by $\mathbf{w}$, we often need to solve a numerical optimization problem.
- For regression, we often solve a problem of the form

$$\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i; \mathbf{w}) - y_i)^2.$$

- For multiclass classification, we often minimize the cross entropy loss

$$\min_{\mathbf{w}} \sum_i -\mathbf{y}_i^\top \ln\left(\text{softmax}(f(\mathbf{x}_i; \mathbf{w}))\right),$$

where $f(\mathbf{x}_i; \mathbf{w})$ is the vector of class scores, and $\mathbf{y}_i$ is the one-hot vector for the true class of $\mathbf{x}_i$.

Machine Learning != Numerical Optimization

- While learning generally involves solving an optimization problem on the training set, the objective is actually to optimize the *unknown* expected risk.
- In numerical optimization, we try very hard to obtain an optimizer of the objective function, because the objective function is usually exactly what we want to optimize.
- In machine learning, we often just try hard enough to obtain a near-optimal solution of the objective function, because it is usually just a proxy to what we want to optimize (the expected risk).

Optimization is Hard in General

- Many numerical optimization problems are provably hard – there are no efficient algorithms for finding the optimizers
- Classical machine learning algorithms often formulate learning as "easy" numerical optimization problems.
- Deep learning uses very complex functions, which are generally considered to be "hard" in numerical optimization literature.

"Easy" problems

- For quite some time, people thought that only linear optimization problems (i.e. linear objective, linear constraints) are efficiently solvable.
- Later, people realised that the larger class of convex optimization problems (i.e. convex objective, conex feasible set) are efficiently solvable.
- In addition, the class of smooth functions often has efficient algorithms as well.

"Hard" problems

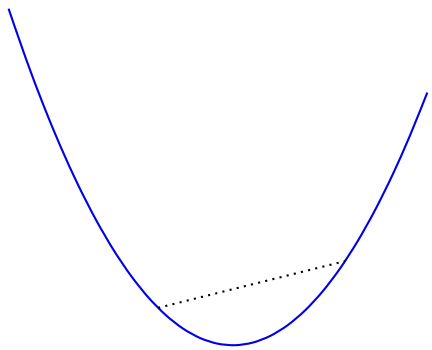
- Nonconvex and nonsmooth functions are often considered to be hard to optimize.
- In deep learning, we often need to deal with such functions.
- We discuss some difficulties associated with optimizing such functions and how to deal with them.

Convexity

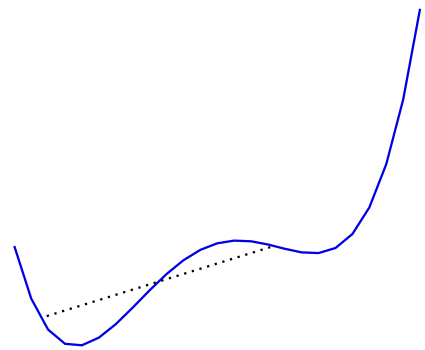
- A function f is convex if

$$f(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') \leq \lambda f(\mathbf{w}) + (1-\lambda) f(\mathbf{w}')$$

for all $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$.



Convex



Nonconvex

- Convexity $\Rightarrow$ a local minimizer is globally optimal, thus it is good enough to find a local minimizer.
- Nonconvexity $\Rightarrow$ local minima may not be globally optimal, but this is hard to detect.
- A common method to deal with local minima is to run an algorithm from different initial values, and then pick the solution with minimum value.

Smoothness

- A function f is smooth if its gradient is Lipschitz, i.e.

$$|| \nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')||_2 \leq \beta ||\mathbf{w} - \mathbf{w}'||_2.$$

- This means f has a quadratic upper bound around any given $\mathbf{w}$

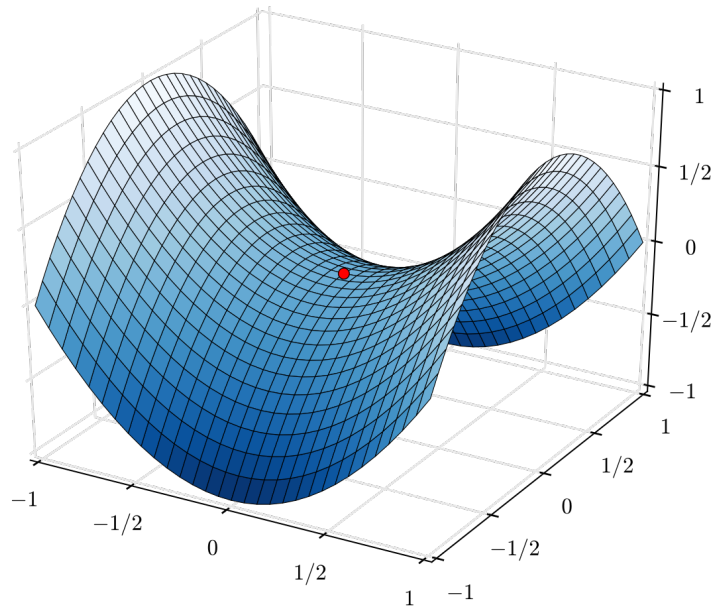
$$f(\mathbf{w}') \leq f(\mathbf{w}) + \nabla f(\mathbf{w})^T(\mathbf{w}' - \mathbf{w}) + \frac{\beta}{2}||\mathbf{w}' - \mathbf{w}||_2^2.$$

- Quadratic functions are easy to minimize, and we can use the quadratic upper bound to guide the minimization process.

- A smooth function need to have continuous gradients.
- A commonly used nonsmooth function is the positive part function $(u)_+$ (the ReLU activation).
- We can generalize gradient-based method to sub-gradient method.
  - g is a subgradient of f at $\mathbf{w}$ if $f(\mathbf{w}') \geq f(\mathbf{w}) + g^\top(\mathbf{w}' - \mathbf{w})$.
  - Example. The subgradients of $(u)_+ = \begin{cases} \{1\} & u > 0, \\ [0,1] & u = 0, \\ \{0\}, & u < 0. \end{cases}$

Saddle Points

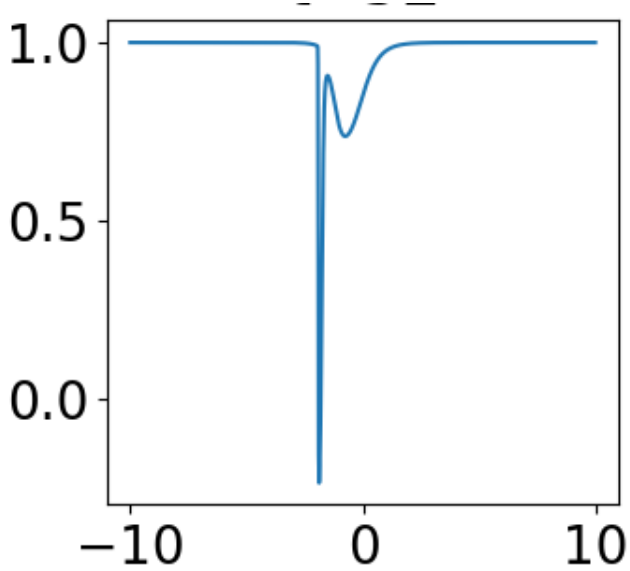
- A nonconvex function may have saddle points, which have zero gradient, but are neither minimizers nor maximizers.



- These points are not very useful, but for high-dimensional problems, it is hard to differentiate them from local optima.

Plateaux

- Both convex and nonconvex functions may have very flat regions.



- Gradients are very small at these regions, thus numerically, points in these regions really look like critical points.
- Plateaux are quite common in deep neural networks using sigmoid activations.

- We need to start with reasonable initial values to avoid plateaux.
- Small random values often work well in practice.
- For certain networks, there are some guidelines about how to set the range of these initial values (next lecture).

Your Turn

Which of the following statement is correct? (Multiple choice)

(a) $f(x) = \tanh(x)$ is a convex function.

(b) $f(x, y) = x^2 + y^2$ is a convex function.

(c) $f(x) = \sigma(x)$ have two plateux.

Overview of Methods

- An optimization method can be exact or stochastic (e.g. full gradient descent vs. stochastic gradient descent).
  - Exact methods require a full pass through the dataset in each iteration is not practical for deep neural networks (dataset is too large).
  - Stochastic method are often used instead — this is particularly useful when the dataset is highly redundant.

- Depending on the order of the derivatives used, an optimization algorithm can be a
  - zeroth order method: only use function evaluations
  - first-order method: requires gradient evaluations
  - second order method: requires Hessian evaluations
- 0th order method often requires large number of function evaluations.
- 2nd order method is computationally expensive due to the need to evaluate Hessians.

Gradient Descent

- In gradient descent, we start with some initial $\mathbf{w}_1 \in \mathbf{R}^d$. At step $s \geq 1$, we update $\mathbf{w}_s$ as follows

$$\mathbf{w}_{s+1} = \mathbf{w}_s - \eta_s \nabla f(\mathbf{w}_s),$$

  where $\eta_s > 0$ is a step size.
- Stochastic gradient descent (SGD) uses stochastic gradients in place of full gradients.
- For convex functions, there are learning rates that can be shown to guarantee convergence.
- However, such learning rates are often hard, if not impossible, to compute.

Momentum

Let's keep the momentum going!

- Gradient descent and SGD can be very slow in practice.
- A *momentum* term is often added to speed up convergence

$$\mathbf{w}_2 = \mathbf{w}_1 - \eta g_1,$$

$$\mathbf{w}_{s+1} = \mathbf{w}_s \overbrace{-\eta g_s}^{\textit{steepest descent}} \overbrace{+\alpha(\mathbf{w}_s - \mathbf{w}_{s-1})}^{\textit{momentum}}, \quad s \geq 2.$$

  where g_s is the full gradient or a stochastic gradient at $\mathbf{w}_s$.
- The momentum term keeps going along the previous descent direction.
- If a direction has consistent gradients, its velocity will build up.
- In general, η and α can change. Typically $\alpha = 0.9$.

- We can also write down w_{s+1} as

$$\mathbf{w}_{s+1} = \mathbf{w}_s - \eta(g_s + \alpha g_{s-1} + \ldots + \alpha^{s-1} g_1).$$

- Thus the momentum method moves along the direction of a weighted average of $g_s, \ldots, g_1$, with larger weights for more recent gradients.

Rolling ball interpretation

- We have a ball at a point $\mathbf{w}$.
- Objective: minimize the ball's potential energy $U(\mathbf{w})$ ("loss function") by rolling it down a hill.
- Assume that for a small time step δ, $\mathbf{w}$ changes to $\mathbf{w}'$ when the ball is rolling downhill.
- We show that the update is essentially gradient descent with momentum.

- From the definition of potential energy and Newton's 2nd law of motion, $U(\mathbf{w})$, the force F acting on the ball, and the ball's acceleration a satisfy

$$\left.\begin{aligned} F &= -\nabla U \\ F &= ma \end{aligned}\right\} \Rightarrow a = -\frac{1}{m}\nabla U.$$

- Velocity is updated from v to

$$v' = \beta v + a\delta,$$

  where β is due to friction.

- Position is updated from $\mathbf{w}$ to (approximately)

$$\mathbf{w}' = \mathbf{w} + \frac{v + v'}{2}\delta = \mathbf{w} - \frac{\delta^2}{2m}\nabla U + \beta\delta v,$$

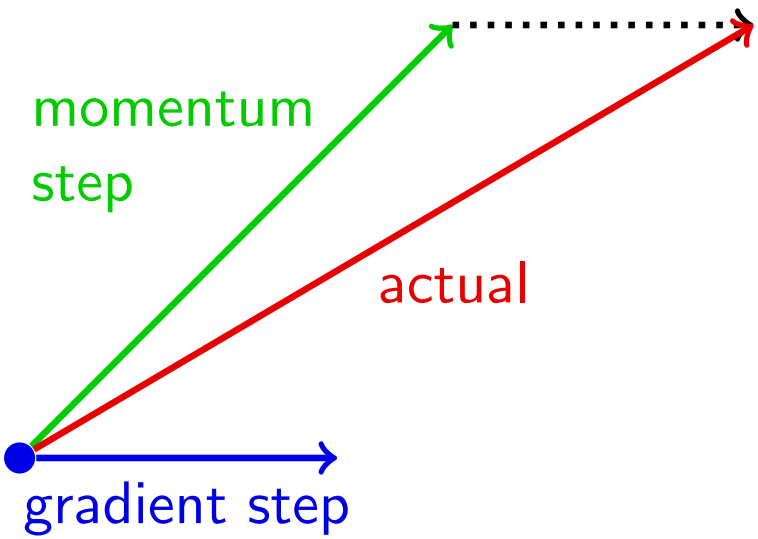
  which is exactly gradient descent with momentum.

Nesterov's Method

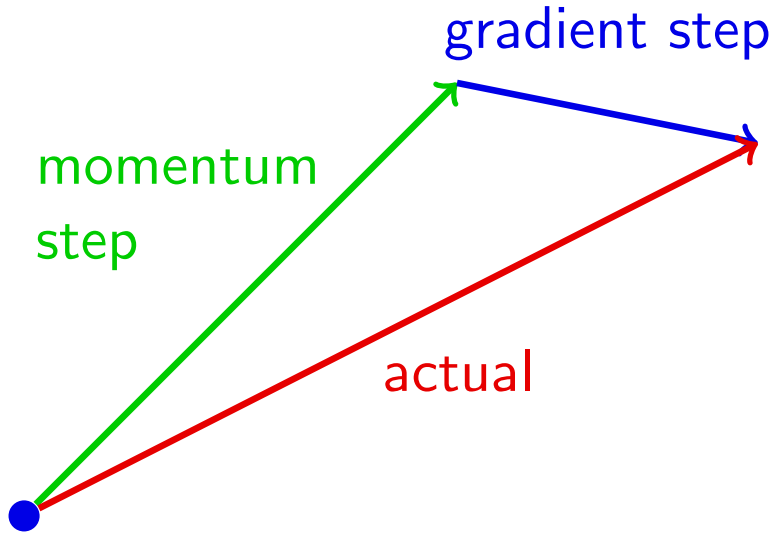
- Nesterov's method is very similar to the standard momentum method, but it applies momentum first, followed by gradient descent.

$$\mathbf{v}_s = \mathbf{w}_s + \alpha_s(\mathbf{w}_s - \mathbf{w}_{s-1}),$$
$$\mathbf{w}_{s+1} = \mathbf{v}_s - \eta_s \nabla f(\mathbf{v}_s).$$

momentum



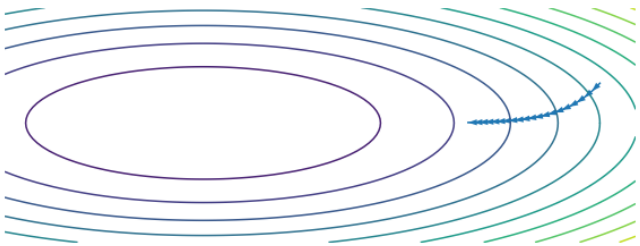
Nesterov



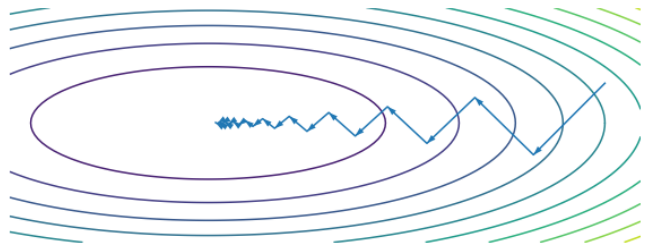
- Nesterov's method is a provably optimal first-order method for convex functions.
- In practice, it often works better than standard momentum method.

Example

- Minimize $f(x, y) = 0.1x^2 + y^2$, starting from $(10, 1)$.
  - Is f convex or non-convex? What is its minimizer?
- Vanilla GD



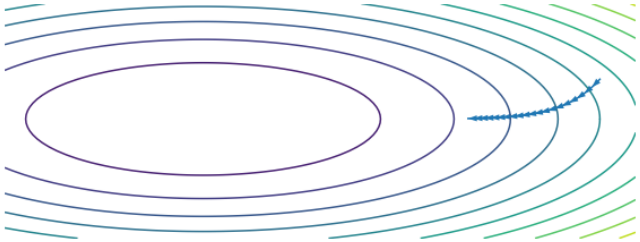
small learning rate



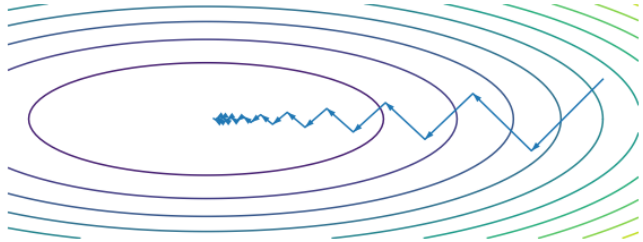
large learning rate

- How do learning rate η and momentum weight α affect the performance of momentum methods?

- Vanilla GD

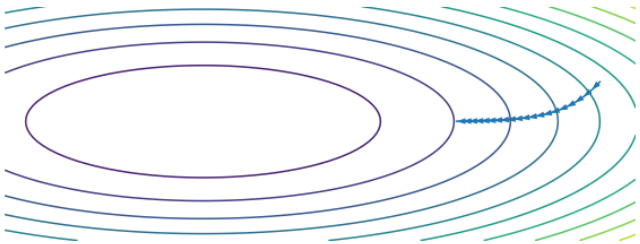


small learning rate

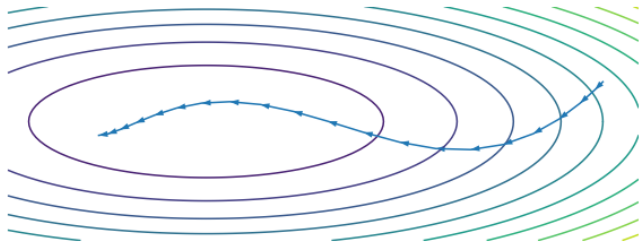


large learning rate

- Momentum with small learning rate

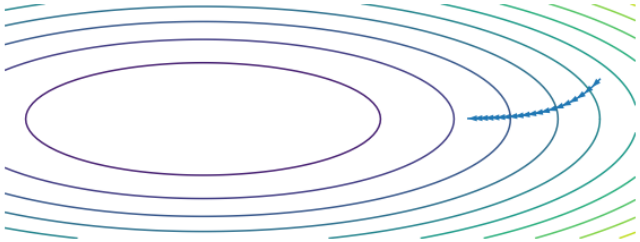


small momentum

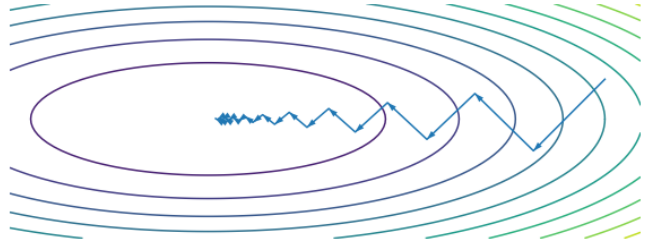


large momentum

- Vanilla GD

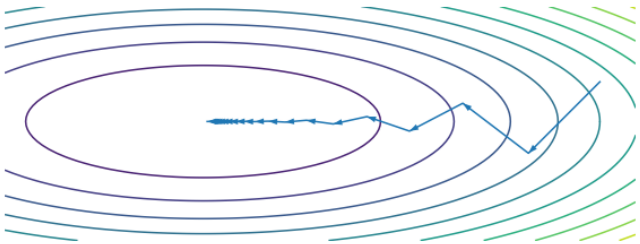


small learning rate

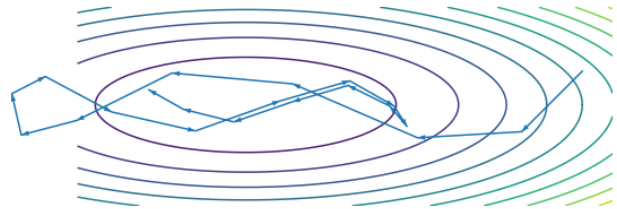


large learning rate

- Momentum with large learning rate

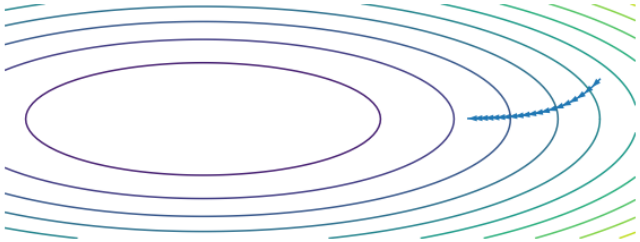


small momentum

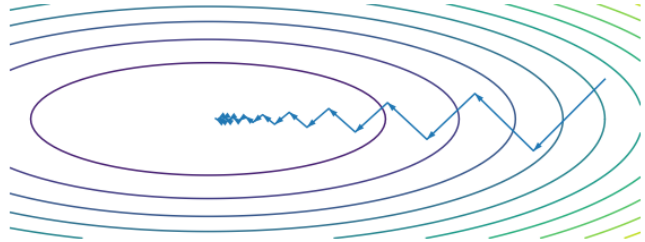


large momentum

- Vanilla GD

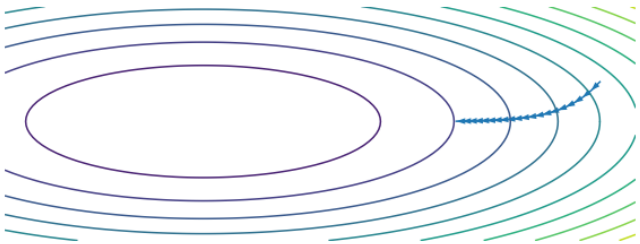


small learning rate

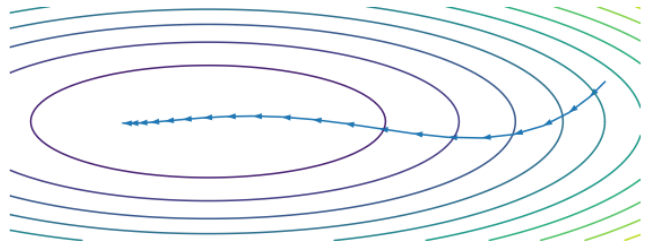


large learning rate

- Nesterov with small learning rate

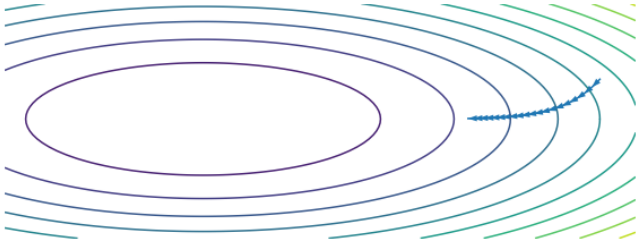


small momentum

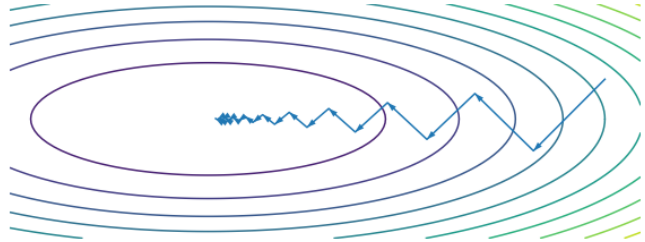


large momentum

- Vanilla GD

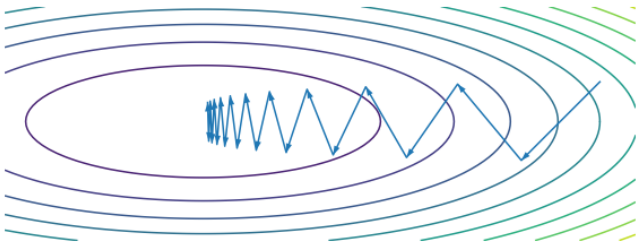


small learning rate

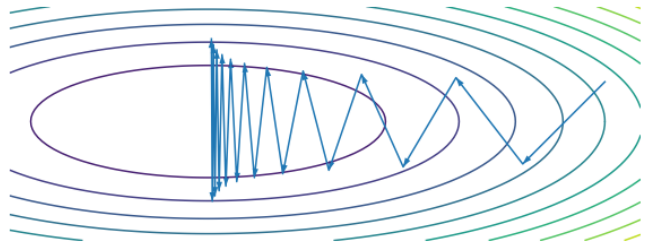


large learning rate

- Nesterov with large learning rate



small momentum



large momentum

- For all SGD variants, a small learning rate causes slow convergence, a large learning rate causes jumpy or divergent behavior.
- For standard momentum and Nesterov, a suitable momentum accelerates convergence, but a large momentum causes jumpy or divergent behavior.

What You Need to Know...

- Many learning problems involve numerical optimization
- Numerical optimizations are hard in general
- Easy problems: convexity and smoothness
- Hard problems: saddle points, plateaux
- Taxonomy of methods
  - Exact vs stochastic
  - Zero-th order, first order, second order
- Gradient descent, SGD, and acceleration using momentum or Nesterov's method