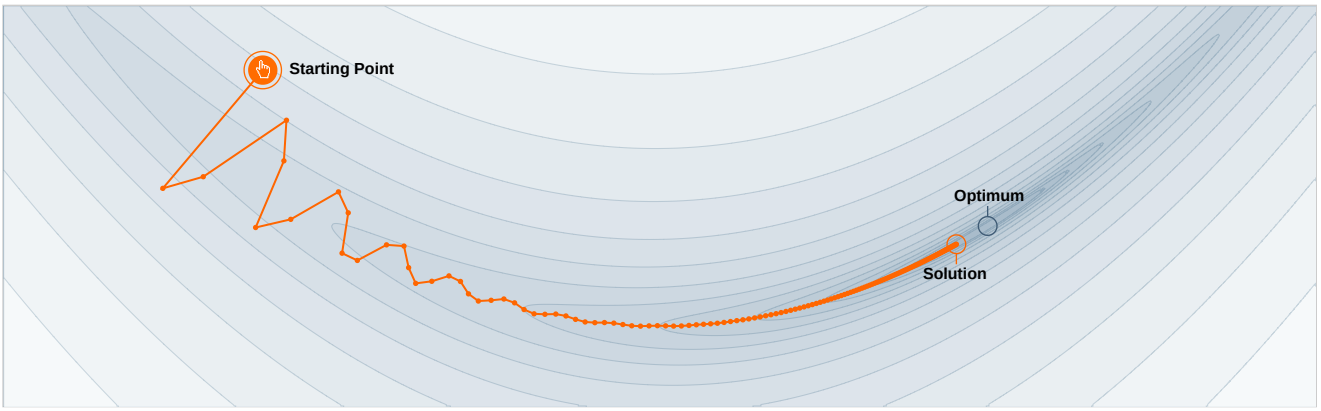
Why Momentum Really Works



Step-size α = 0.02

0 0.003 0.006

Momentum β = 0.99

0.00 0.500 0.990

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

GABRIEL GOH UC Davis
April. 4 2017
Citation: Goh, 2017

Here's a popular story about momentum [1, 2, 3]: gradient descent is a man walking down a hill. He follows the steepest path downwards; his progress is slow, but steady. Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima.

This standard story isn't wrong, but it fails to explain many important behaviors of momentum. In fact, momentum can be understood far more precisely if we study it on the right model.

One nice model is the convex quadratic. This model is rich enough to reproduce momentum's local dynamics in real problems, and yet simple enough to be understood in closed form. This balance gives us powerful traction for understanding this algorithm.

We begin with gradient descent. The algorithm has many virtues, but speed is not one of them. It is simple — when optimizing a smooth function f, we make a small step in the gradient

$$w^{k+1} = w^k - \alpha \nabla f(w^k).$$

For a step-size small enough, gradient descent makes a monotonic improvement at every iteration. It always converges, albeit to a local minimum. And under a few weak curvature conditions it can even get there at an exponential rate.

But the exponential decrease, though appealing in theory, can often be infuriatingly small. Things often begin quite well — with an impressive, almost immediate decrease in the loss. But as the iterations progress, things start to slow down. You start to get a nagging feeling you're not making as much progress as you should be. What has gone wrong?

The problem could be the optimizer's old nemesis, pathological curvature. Pathological curvature is, simply put, regions of f which aren't scaled properly. The landscapes are often described as valleys, trenches, canals and ravines. The iterates either jump between valleys, or approach the optimum in small, timid steps. Progress along certain directions grind to a halt. In these unfortunate regions, gradient descent fumbles.

Momentum proposes the following tweak to gradient descent. We give gradient descent a short-term memory:

$$z^{k+1} = \beta z^k + \nabla f(w^k)$$
$$w^{k+1} = w^k - \alpha z^{k+1}$$

The change is innocent, and costs almost nothing. When $\beta = 0$, we recover gradient descent. But for $\beta = 0.99$ (sometimes 0.999, if things are really bad), this appears to be the boost we need. Our iterations regain that speed and boldness it lost, speeding to the optimum with a renewed energy.

Optimizers call this minor miracle “acceleration”.

The new algorithm may seem at first glance like a cheap hack. A simple trick to get around gradient descent’s more aberrant behavior — a smoother for oscillations between steep canyons. But the truth, if anything, is the other way round. It is gradient descent which is the hack. First, momentum gives up to a quadratic speedup on many functions. [1] This is no small matter — this is similar to the speedup you get from the Fast Fourier Transform, Quicksort, and Grover’s Algorithm. When the universe gives you quadratic speedups, you should start to pay attention.

But there’s more. A lower bound, courtesy of Nesterov [5], states that momentum is, in a certain very narrow and technical sense, optimal. Now, this doesn’t mean it is the best algorithm for all functions in all circumstances. But it does satisfy some curiously beautiful mathematical properties which scratch a very human itch for perfection and closure. But more on that later. Let’s say this for now — momentum is an algorithm for the book.

First Steps: Gradient Descent

We begin by studying gradient descent on the simplest model possible which isn’t trivial — the convex quadratic,

$$f(w) = \tfrac{1}{2} w^T A w - b^T w, \qquad w \in \mathbf{R}^n.$$

Assume A is symmetric and invertible, then the optimal solution $w^\star$ occurs at

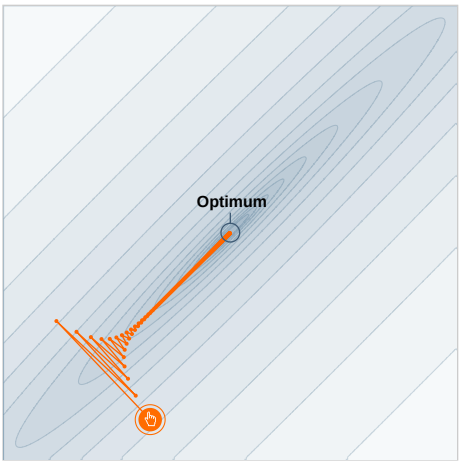
$$w^\star = A^{-1} b.$$

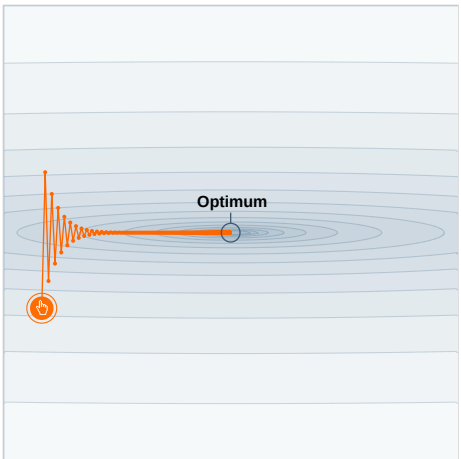
Simple as this model may be, it is rich enough to approximate many functions (think of A as your favorite model of curvature — the Hessian, Fisher Information Matrix [6], etc) and captures all the key features of pathological curvature. And more importantly, we can write an exact closed formula for gradient descent on this function.

This is how it goes. Since $\nabla f(w) = Aw - b$, the iterates are

$$w^{k+1} = w^k - \alpha(Aw^k - b).$$

Here’s the trick. There is a very natural space to view gradient descent where all the dimensions act independently — the eigenvectors of A.





Every symmetric matrix A has an eigenvalue decomposition

$$A = Q\,\mathrm{diag}(\lambda_1, \dots, \lambda_n)\,Q^T, \qquad Q = [q_1, \dots, q_n],$$

and, as per convention, we will assume that the λ_i's are sorted, from smallest λ_1 to biggest λ_n. If we perform a change of basis, $x^k = Q^T(w^k - w^\star)$, the iterations break apart, becoming:

$$\begin{aligned} x_i^{k+1} &= x_i^k - \alpha\lambda_i x_i^k \\ &= (1-\alpha\lambda_i)x_i^k = (1-\alpha\lambda_i)^{k+1}x_i^0 \end{aligned}$$

Moving back to our original space w, we can see that

$$w^k - w^\star = Qx^k = \sum_i^n x_i^0(1-\alpha\lambda_i)^k q_i$$

and there we have it — gradient descent in closed form.

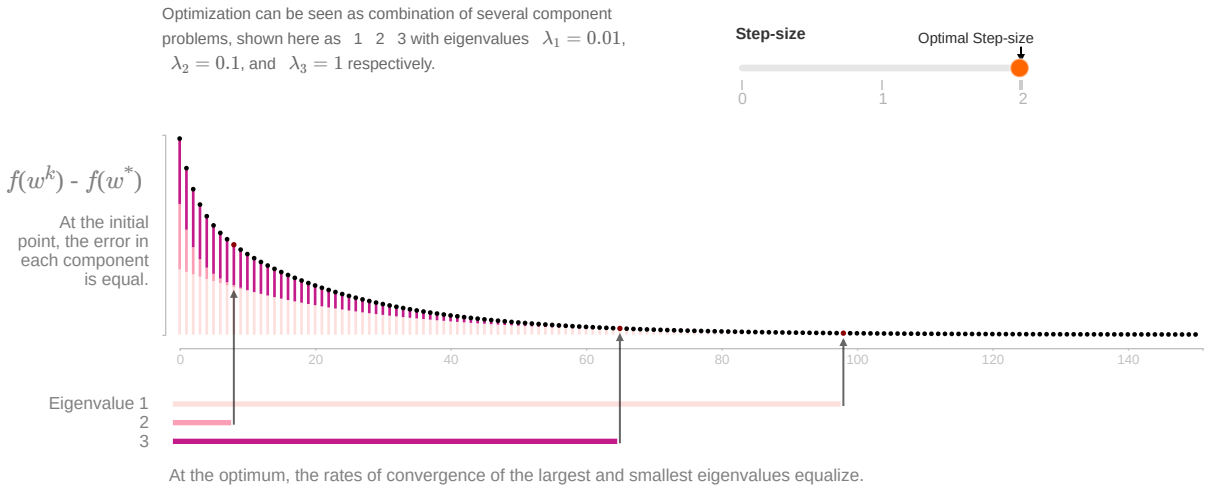
Decomposing the Error

The above equation admits a simple interpretation. Each element of x^0 is the component of the error in the initial guess in the Q-basis. There are n such errors, and each of these errors follows its own, solitary path to the minimum, decreasing exponentially with a compounding rate of $1-\alpha\lambda_i$. The closer that number is to 1, the slower it converges.

For most step-sizes, the eigenvectors with largest eigenvalues converge the fastest. This triggers an explosion of progress in the first few iterations, before things slow down as the smaller eigenvectors' struggles are revealed. By writing the contributions of each eigenspace's error to the loss

$$f(w^k) - f(w^\star) = \sum (1-\alpha\lambda_i)^{2k}\lambda_i[x_i^0]^2$$

we can visualize the contributions of each error component to the loss.



At the optimum, the rates of convergence of the largest and smallest eigenvalues equalize.

Choosing A Step-size

The above analysis gives us immediate guidance as to how to set a step-size α. In order to converge, each $|1-\alpha\lambda_i|$ must be strictly less than 1. All workable step-sizes, therefore, fall in the interval

$$0 < \alpha\lambda_i < 2.$$

The overall convergence rate is determined by the slowest error component, which must be either λ_1 or λ_n:

$$\begin{aligned} \mathrm{rate}(\alpha) &= \max_i |1-\alpha\lambda_i| \\ &= \max\{|1-\alpha\lambda_1|,\ |1-\alpha\lambda_n|\} \end{aligned}$$

This overall rate is minimized when the rates for λ_1 and λ_n are the same — this mirrors our informal observation in the previous section that the optimal step-size causes the first and last eigenvectors to converge at the same rate. If we work this through we get:

$$\text{optimal } \alpha \;=\; \operatorname*{argmin}_{\alpha} \text{rate}(\alpha) \;=\; \frac{2}{\lambda_1 + \lambda_n}$$

$$\text{optimal rate} \;=\; \min_{\alpha} \text{rate}(\alpha) \;=\; \frac{\lambda_n/\lambda_1 - 1}{\lambda_n/\lambda_1 + 1}$$

Notice the ratio λ_n/λ_1 determines the convergence rate of the problem. In fact, this ratio appears often enough that we give it a name, and a symbol — the condition number.

$$\text{condition number} := \kappa := \frac{\lambda_n}{\lambda_1}$$

The condition number means many things. It is a measure of how close to singular a matrix is. It is a measure of how robust $A^{-1}b$ is to perturbations in b. And, in this context, the condition number gives us a measure of how poorly gradient descent will perform. A ratio of $\kappa = 1$ is ideal, giving convergence in one step (of course, the function is trivial). Unfortunately the larger the ratio, the slower gradient descent will be. The condition number is therefore a direct measure of pathological curvature.

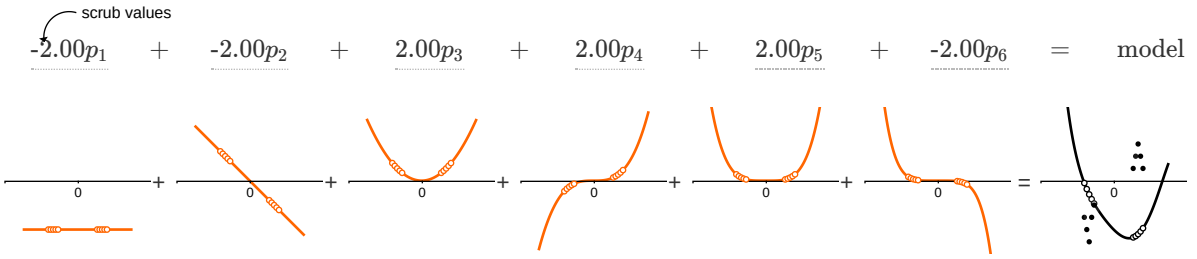
Example: Polynomial Regression

The above analysis reveals an insight: all errors are not made equal. Indeed, there are different kinds of errors, n to be exact, one for each of the eigenvectors of A. And gradient descent is better at correcting some kinds of errors than others. But what do the eigenvectors of A mean? Surprisingly, in many applications they admit a very concrete interpretation.

Lets see how this plays out in polynomial regression. Given 1D data, ξ_i, our problem is to fit the model

$$\text{model}(\xi) = w_1 p_1(\xi) + \cdots + w_n p_n(\xi) \qquad p_i = \xi \mapsto \xi^{i-1}$$

to our observations, d_i. This model, though nonlinear in the input ξ, is linear in the weights, and therefore we can write the model as a linear combination of monomials, like:



Because of the linearity, we can fit this model to our data ξ_i using linear regression on the model mismatch

$$\text{minimize}_w \qquad \tfrac{1}{2}\sum_i (\text{model}(\xi_i) - d_i)^2 \;=\; \tfrac{1}{2}\|Zw - d\|^2$$

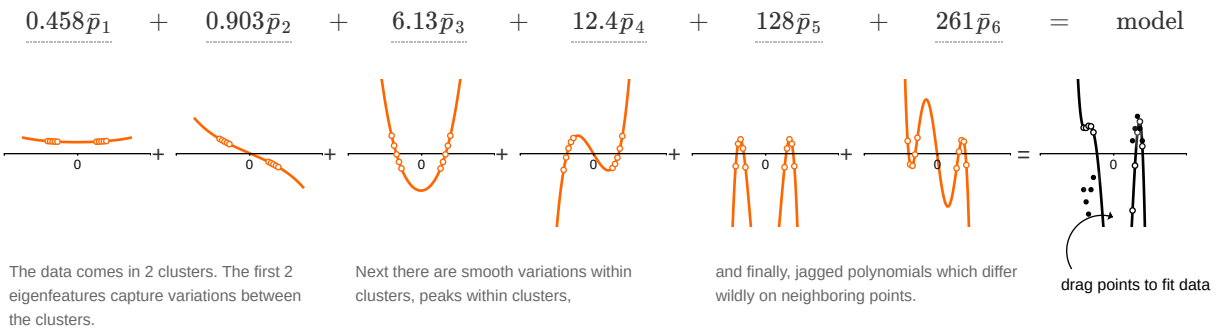
where

$$Z = \begin{pmatrix} 1 & \xi_1 & \xi_1^2 & \cdots & \xi_1^{n-1} \\ 1 & \xi_2 & \xi_2^2 & \cdots & \xi_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_m & \xi_m^2 & \cdots & \xi_m^{n-1} \end{pmatrix}.$$

The path of convergence, as we know, is elucidated when we view the iterates in the space of Q (the eigenvectors of $Z^T Z$). So let's recast our regression problem in the basis of Q. First, we do a change of basis, by rotating w into Qw, and counter-rotating our feature maps p into eigenspace, $\bar{p}$. We can now conceptualize the same regression as one over a different polynomial basis, with the model

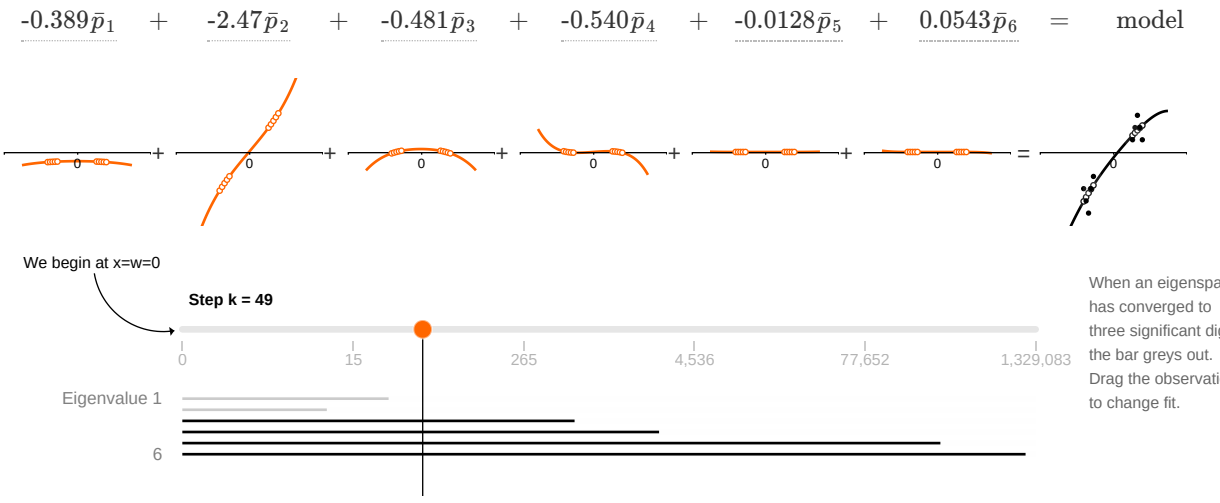
$$\text{model}(\xi) \;=\; x_1\bar{p}_1(\xi) \;+\; \cdots \;+\; x_n\bar{p}_n(\xi) \qquad \bar{p}_i = \sum q_{ij}p_j.$$

This model is identical to the old one. But these new features $\bar{p}$ (which I call "eigenfeatures") and weights have the pleasing property that each coordinate acts independently of the others. Now our optimization problem breaks down, really, into n small 1D optimization problems. And each coordinate can be optimized greedily and independently, one at a time in any order, to produce the final, global, optimum. The eigenfeatures are also much more informative:



The observations in the above diagram can be justified mathematically. From a statistical point of view, we would like a model which is, in some sense, robust to noise. Our model cannot possibly be meaningful if the slightest perturbation to the observations changes the entire model dramatically. And the eigenfeatures, the principal components of the data, give us exactly the decomposition we need to sort the features by its sensitivity to perturbations in d_i's. The most robust components appear in the front (with the largest eigenvalues), and the most sensitive components in the back (with the smallest eigenvalues).

This measure of robustness, by a rather convenient coincidence, is also a measure of how easily an eigenspace converges. And thus, the "pathological directions" — the eigenspaces which converge the slowest — are also those which are most sensitive to noise! So starting at a simple initial point like 0 (by a gross abuse of language, let's think of this as a prior), we track the iterates till a desired level of complexity is reached. Let's see how this plays out in gradient descent.



This effect is harnessed with the heuristic of early stopping : by stopping the optimization early, you can often get better generalizing results. Indeed, the effect of early stopping is very similar to that of more conventional methods of regularization, such as Tikhonov Regression. Both methods try to suppress the components of the smallest eigenvalues directly, though they employ different methods of spectral decay.[2] But early stopping has a distinct advantage. Once the step-size is chosen, there are no regularization parameters to fiddle with. Indeed, in the course of a single optimization, we have the entire family of models, from underfitted to overfitted, at our disposal. This gift, it seems, doesn't come at a price. A beautiful free lunch [7] indeed.

The Dynamics of Momentum

Let's turn our attention back to momentum. Recall that the momentum update is

$$z^{k+1} = \beta z^k + \nabla f(w^k)$$
$$w^{k+1} = w^k - \alpha z^{k+1}.$$

Since $\nabla f(w^k) = Aw^k - b$, the update on the quadratic is

$$z^{k+1} = \beta z^k + (Aw^k - b)$$
$$w^{k+1} = w^k - \alpha z^{k+1}.$$

Following [8], we go through the same motions, with the change of basis $x^k = Q(w^k - w^\star)$ and $y^k = Qz^k$, to yield the update rule

$$y_i^{k+1} = \beta y_i^k + \lambda_i x_i^k$$
$$x_i^{k+1} = x_i^k - \alpha y_i^{k+1}.$$

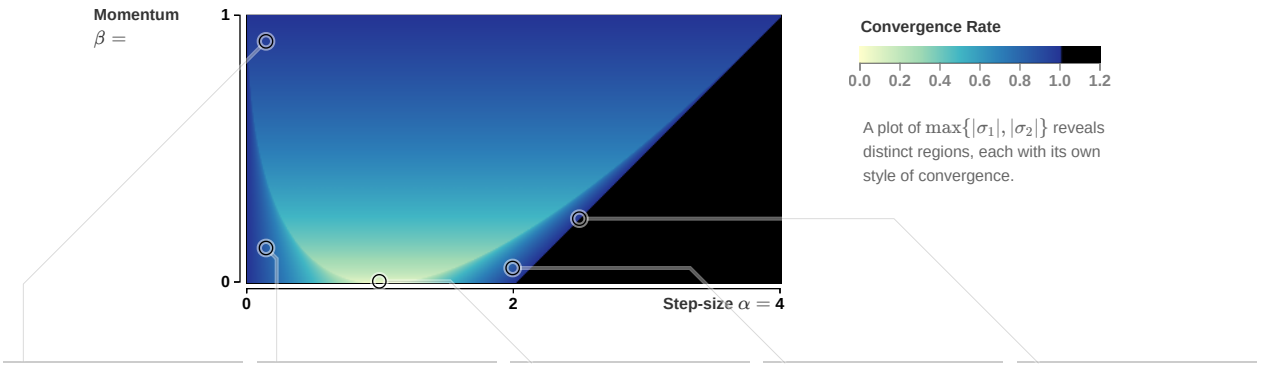
in which each component acts independently of the other components (though x_i^k and y_i^k are coupled). This lets us rewrite our iterates as [3]

$$\begin{pmatrix} y_i^k \\ x_i^k \end{pmatrix} = R^k \begin{pmatrix} y_i^0 \\ x_i^0 \end{pmatrix} \qquad R = \begin{pmatrix} \beta & \lambda_i \\ -\alpha\beta & 1 - \alpha\lambda_i \end{pmatrix}.$$

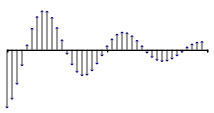
There are many ways of taking a matrix to the k^{th} power. But for the 2×2 case there is an elegant and little known formula [9] in terms of the eigenvalues of R, σ_1 and σ_2.

$$R^k = \begin{cases} \sigma_1^k R_1 - \sigma_2^k R_2 & \sigma_1 \neq \sigma_2 \\ \sigma_1^k (kR/\sigma_1 - (k-1)I) & \sigma_1 = \sigma_2 \end{cases}, \qquad R_j = \frac{R - \sigma_j I}{\sigma_1 - \sigma_2}$$

This formula is rather complicated, but the takeaway here is that it plays the exact same role the individual convergence rates, $1 - \alpha\lambda_i$ do in gradient descent. But instead of one geometric series, we have two coupled series, which may have real or complex values. The convergence rate is therefore the slowest of the two rates, $\max\{|\sigma_1|, |\sigma_2|\}$ [4]. By plotting this out, we see there are distinct regions of the parameter space which reveal a rich taxonomy of convergence behavior [10]:

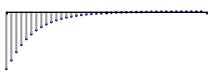


Ripples



R's eigenvalues are complex, and the iterates display low frequency ripples. Surprisingly, the convergence rate $2\sqrt{\beta}$ is independent of α and λ_i.

Monotonic Decrease

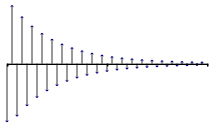


R's eigenvalues are both real, are positive, and have norm less than one. The behavior here resembles gradient descent.

1-Step Convergence

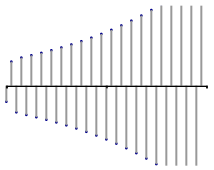
When $\alpha = 1/\lambda_i$, and $\beta = 0$, we converge in one step. This is a very special point, and kills the error in the eigenspace completely.

Monotonic Oscillations



When $\alpha > 1/\lambda_i$, the iterates flip between $+$ and $-$ at each iteration. These are often referred to as 'oscillations' in gradient descent.

Divergence



When $\max\{|\sigma_1|, |\sigma_2|\} > 1$, the iterates diverge.

For what values of α and β does momentum converge? Since we need both σ_1 and σ_2 to converge, our convergence criterion is now $\max\{|\sigma_1|, |\sigma_2|\} < 1$. The range of available step-sizes work out [5] to be

$$0 < \alpha\lambda_i < 2 + 2\beta \qquad \text{for} \qquad 0 \leq \beta < 1$$

We recover the previous result for gradient descent when $\beta = 0$. But notice an immediate boon we get. Momentum allows us to crank up the step-size up by a factor of 2 before diverging.

The Critical Damping Coefficient

The true magic happens, however, when we find the sweet spot of α and β. Let us try to first optimize over β.

Momentum admits an interesting physical interpretation when α is [11] small: it is a discretization of a damped harmonic oscillator. Consider a physical simulation operating in discrete time (like a video game).

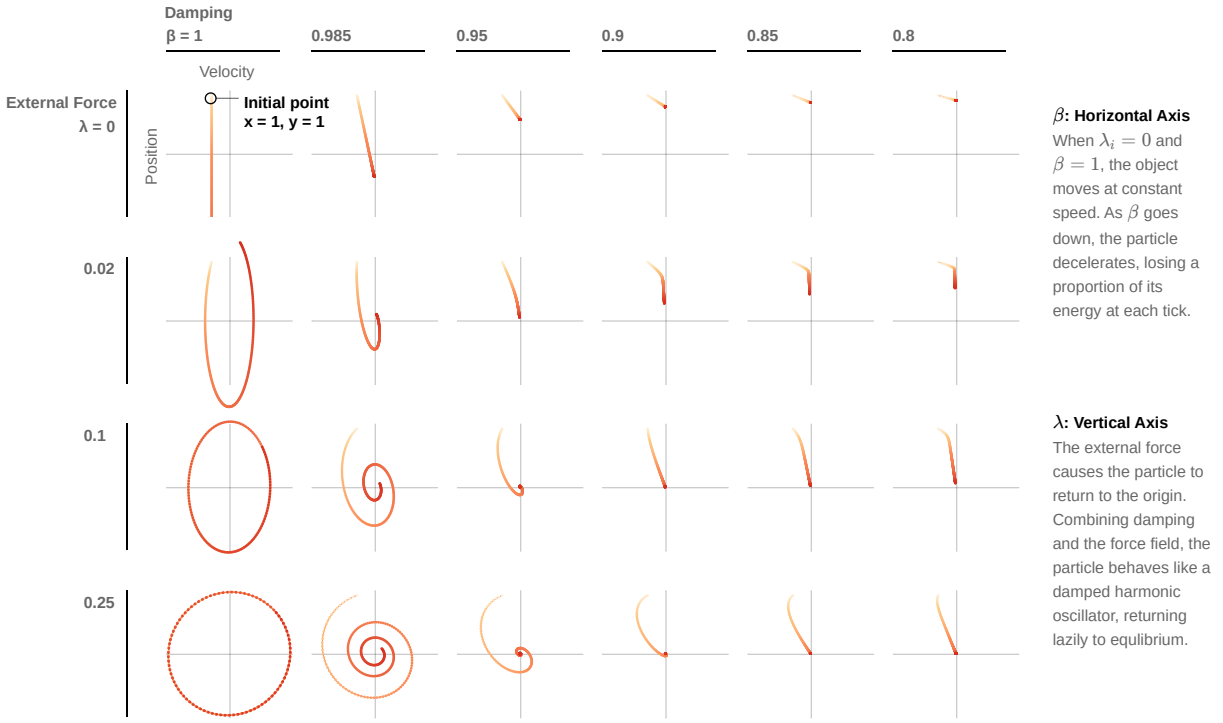
$$y_i^{k+1} = \beta y_i^k + \lambda_i x_i^k$$

We can think of $-y_i^k$ as **velocity** — which is dampened at each step — and perturbed by an external force field

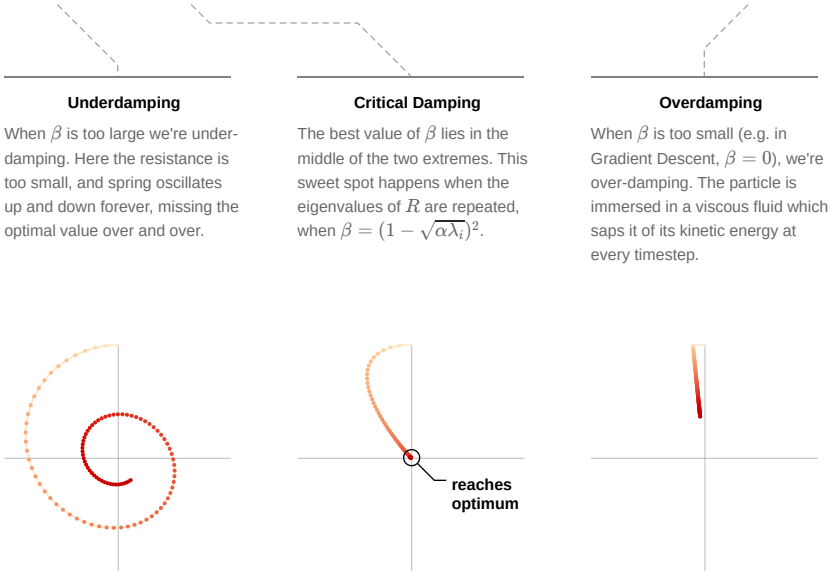
$$x_i^{k+1} = x_i^k - \alpha y_i^{k+1}$$

And x is our particle's **position** — which is moved at each step by a small amount in the direction of the velocity y_i^{k+1}.

We can break this equation apart to see how each component affects the dynamics of the system. Here we plot, for 150 iterates, the particle's velocity (the horizontal axis) against its position (the vertical axis), in a phase diagram.



This system is best imagined as a weight suspended on a spring. We pull the weight down by one unit, and we study the path it follows as it returns to equilibrium. In the analogy, the spring is the source of our external force $\lambda_i x_i^k$, and equilibrium is the state when both the position x_i^k and the speed y_i^k are 0. The choice of β crucially affects the rate of return to equilibrium.



The critical value of $\beta = (1 - \sqrt{\alpha\lambda_i})^2$ gives us a convergence rate (in eigenspace i) of $1 - \sqrt{\alpha\lambda_i}$. A square root improvement over gradient descent, $1 - \alpha\lambda_i$! Alas, this only applies to the error in the i^{th} eigenspace, with α fixed.

Optimal parameters

To get a global convergence rate, we must optimize over both α and β. This is a more complicated affair,[6] but they work out to be

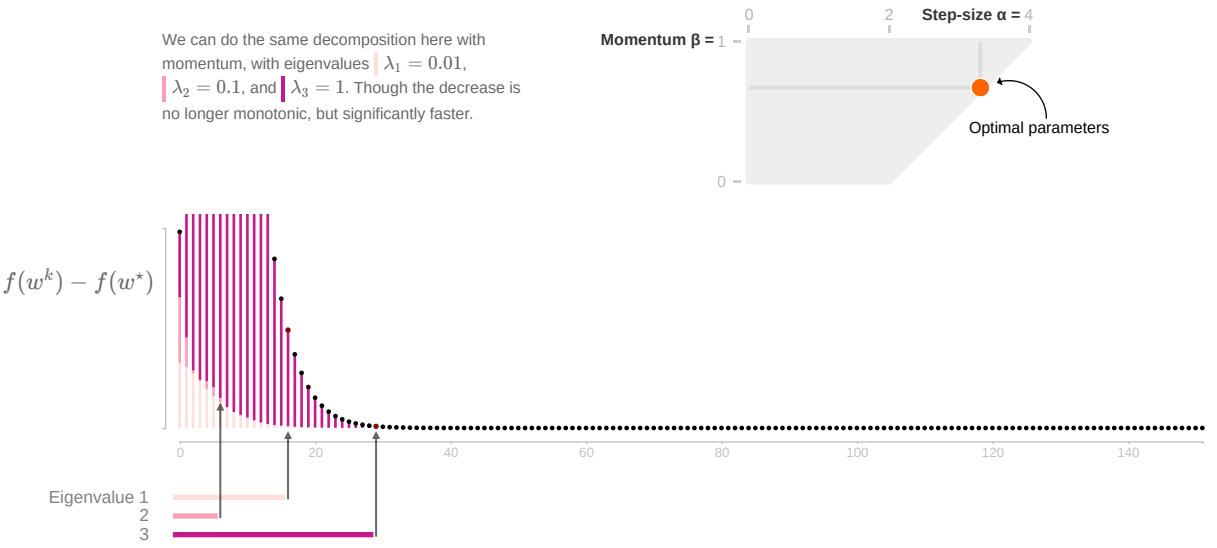
$$\alpha = \left(\frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_n}}\right)^2 \qquad \beta = \left(\frac{\sqrt{\lambda_n} - \sqrt{\lambda_1}}{\sqrt{\lambda_n} + \sqrt{\lambda_1}}\right)^2$$

Plug this into the convergence rate, and you get

$$\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$$ Convergence rate, **Momentum**

$$\frac{\kappa - 1}{\kappa + 1}$$ Convergence rate, **Gradient Descent**

With barely a modicum of extra effort, we have essentially square rooted the condition number! These gains, in principle, require explicit knowledge of λ_1 and λ_n. But the formulas reveal a simple guideline. When the problem's conditioning is poor, the optimal α is approximately twice that of gradient descent, and the momentum term is close to 1. So set β as close to 1 as you can, and then find the highest α which still converges. Being at the knife's edge of divergence, like in gradient descent, is a good place to be.



Note that the optimal parameters do not necessarily imply the fastest convergence, though, only the fastest asymptotic convergence rate.

While the loss function of gradient descent had a graceful, monotonic curve, optimization with momentum displays clear oscillations. These ripples are not restricted to quadratics, and occur in all kinds of functions in practice. They are not cause for alarm, but are an indication that extra tuning of the hyperparameters is required.

Example: The Colorization Problem

Let's look at how momentum accelerates convergence with a concrete example. On a grid of pixels let G be the graph with vertices as pixels, E be the set of edges connecting each pixel to its four neighboring pixels, and D be a small set of a few distinguished vertices. Consider the problem of minimizing

$$\text{minimize} \qquad \frac{1}{2}\sum_{i\in D}(w_i-1)^2 \qquad + \qquad \frac{1}{2}\sum_{i,j\in E}(w_i-w_j)^2.$$

The **colorizer** pulls distinguished pixels towards 1

The **smoother** spreads out the color

The optimal solution to this problem is a vector of all 1's [7]. An inspection of the gradient iteration reveals why we take a long time to get there. The gradient step, for each component, is some form of weighted average of the current value and its neighbors:

$$w_i^{k+1} = w_i^k - \alpha\sum_{j\in N}(w_i^k - w_j^k) - \begin{cases}\alpha(w_i^k-1) & i\in D\\ 0 & i\notin D\end{cases}$$

This kind of local averaging is effective at smoothing out local variations in the pixels, but poor at taking advantage of global structure. The updates are akin to a drop of ink, diffusing through water. Movement towards equilibrium is made only through local corrections and so, left undisturbed, its march towards the solution is slow and laborious. Fortunately, momentum speeds things up significantly.

The eigenvectors of the colorization problem form a generalized Fourier basis for R^n. The smallest eigenvalues have low frequencies, hence gradient descent corrects high frequency errors well but not low frequency ones.

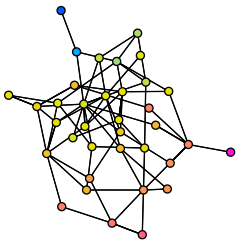
In vectorized form, the colorization problem is

$$\text{minimize} \quad \frac{1}{2}x^T L_G x \quad + \quad \frac{1}{2}\sum_{i \in D}\left(x^T e_i e_i^T x - e_i^T x\right)$$

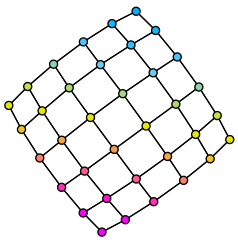
The **smoother**'s quadratic form is the **Graph Laplacian**

And the colorizer is a small low rank correction with a linear term. e_i is the i^{th} unit vector.

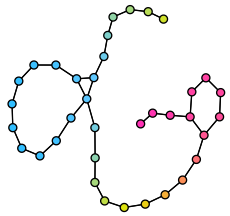
The Laplacian matrix, L_G [8], which dominates the behavior of the optimization problem, is a valuable bridge between linear algebra and graph theory. This is a rich field of study, but one fact is pertinent to our discussion here. The conditioning of L_G, here defined as the ratio of the second eigenvector to the last (the first eigenvalue is always 0, with eigenvector equal to the matrix of all 1′s), is directly connected to the connectivity of the graph.



Small world graphs, like expanders and dense graphs, have excellent conditioning



The conditioning of grids improves with its dimensionality.



And long, wiry graphs, like paths, condition poorly.

These observations carry through to the colorization problem, and the intuition behind it should be clear. Well connected graphs allow rapid diffusion of information through the edges, while graphs with poor connectivity do not. And this principle, taken to the extreme, furnishes a class of functions so hard to optimize they reveal the limits of first order optimization.

The Limits of Descent

Let's take a step back. We have, with a clever trick, improved the convergence of gradient descent by a quadratic factor with the introduction of a single auxiliary sequence. But is this the best we can do? Could we improve convergence even more with two sequences? Could one perhaps choose the α's and β's intelligently and adaptively? It is tempting to ride this wave of optimism - to the cube root and beyond!

Unfortunately, while improvements to the momentum algorithm do exist, they all run into a certain, critical, almost inescapable lower bound.

Adventures in Algorithmic Space

To understand the limits of what we can do, we must first formally define the algorithmic space in which we are searching. Here's one possible definition. The observation we will make is that both gradient descent and momentum can be "unrolled". Indeed, since

$$\begin{aligned} w^1 &= w^0 - \alpha\nabla f(w^0) \\ w^2 &= w^1 - \alpha\nabla f(w^1) \\ &= w^0 - \alpha\nabla f(w^0) - \alpha\nabla f(w^1) \\ &\vdots \\ w^{k+1} &= w^0 - \alpha\nabla f(w^0) - \cdots\cdots - \alpha\nabla f(w^k) \end{aligned}$$

we can write gradient descent as

$$w^{k+1} = w^0 - \alpha\sum_i^k \nabla f(w^i).$$

A similar trick can be done with momentum:

$$w^{k+1} = w^0 + \alpha\sum_i^k \frac{(1-\beta^{k+1-i})}{1-\beta}\nabla f(w^i).$$

In fact, all manner of first order algorithms, including the Conjugate Gradient algorithm, AdaMax, Averaged Gradient and more, can be written (though not quite so neatly) in this unrolled form. Therefore the class of algorithms for which

$$w^{k+1} = w^0 + \sum_i^k \gamma_i^k \nabla f(w^i) \qquad \text{for some } \gamma_i^k$$

contains momentum, gradient descent and a whole bunch of other algorithms you might dream up. This is what is assumed in Assumption 2.1.4 [5] of Nesterov. But let's push this even further, and expand this class to allow different step-sizes for different directions.

$$w^{k+1} = w^0 + \sum_i^k \Gamma_i^k \nabla f(w^i) \quad \text{for some diagonal matrix } \Gamma_i^k.$$

This class of methods covers most of the popular algorithms for training neural networks, including ADAM and AdaGrad. We shall refer to this class of methods as "Linear First Order Methods", and we will show a single function all these methods ultimately fail on.

The Resisting Oracle

Earlier, when we talked about the colorizer problem, we observed that wiry graphs cause bad conditioning in our optimization problem. Taking this to its extreme, we can look at a graph consisting of a single path — a function so badly conditioned that Nesterov called a variant of it "the worst function in the world". The function follows the same structure as the colorizer problem, and we shall call this the Convex Rosenbrock,

$$f^n(w) = \frac{1}{2}(w_1 - 1)^2 + \frac{1}{2}\sum_{i=1}^n (w_i - w_{i+1})^2 + \frac{2}{\kappa - 1}\|w\|^2.$$

with a colorizer of one node; strong couplings of adjacent nodes in the path; and a small regularization term.

The optimal solution of this problem is

$$w_i^\star = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i$$

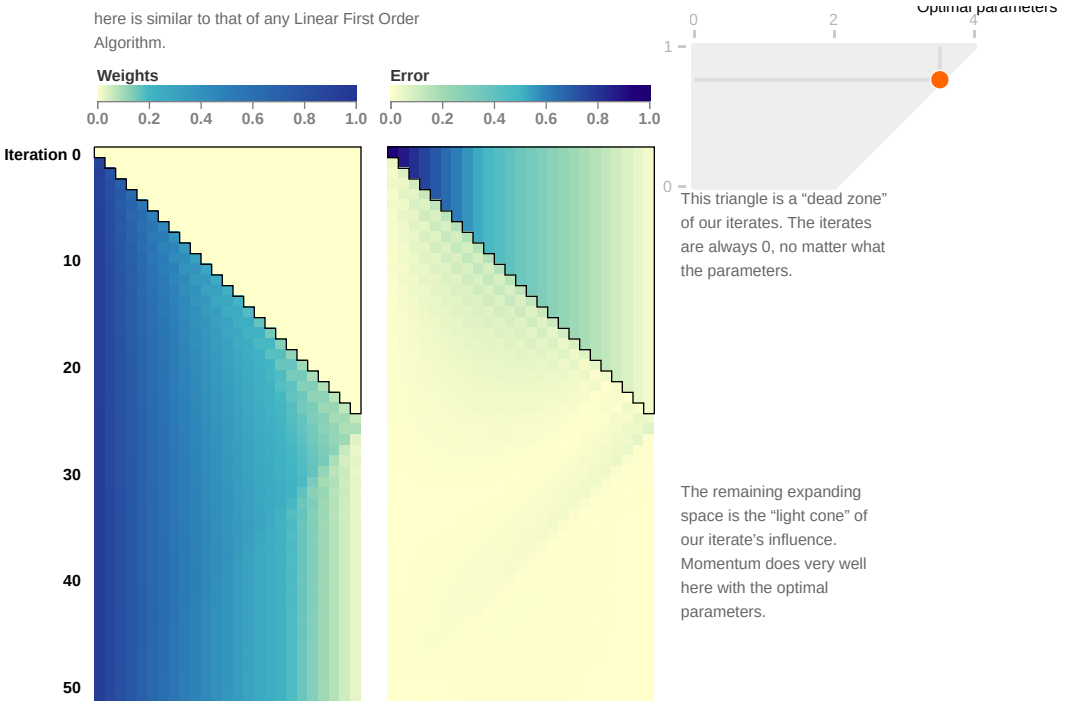
and the condition number of the problem f^n approaches κ as n goes to infinity. Now observe the behavior of the momentum algorithm on this function, starting from $w^0 = 0$.

Here we see the first 50 iterates of momentum on the Convex Rosenbrock for $n = 25$. The behavior

Momentum β =

Step-size α =

Optimal parameters



The observations made in the above diagram are true for any Linear First Order algorithm. Let us prove this. First observe that each component of the gradient depends only on the values directly before and after it:

$$\nabla f(x)_i = 2w_i - w_{i-1} - w_{i+1} + \frac{4}{\kappa - 1} w_i, \qquad i \neq 1.$$

Therefore the fact we start at 0 guarantees that that component must remain stoically there till an element either before or after it turns nonzero. And therefore, by induction, for any linear first order algorithm,

$$\begin{aligned}
w^0 &= [\,0, \quad 0, \quad 0, \quad \dots \quad 0, \quad 0, \quad \dots \quad 0\,] \\
w^1 &= [\,w_1^1, \quad 0, \quad 0, \quad \dots \quad 0, \quad 0, \quad \dots \quad 0\,] \\
w^2 &= [\,w_1^2, \quad w_2^2, \quad 0, \quad \dots \quad 0, \quad 0, \quad \dots \quad 0\,] \\
&\vdots \\
w^k &= [\,w_1^k, \quad w_2^k, \quad w_3^k, \quad \dots \quad w_k^k, \quad 0, \quad \dots \quad 0\,].
\end{aligned}$$

Think of this restriction as a "speed of light" of information transfer. Error signals will take at least k steps to move from w_0 to w_k. We can therefore sum up the errors which cannot have changed yet[9]:

$$\begin{aligned}
\|w^k - w^\star\|_\infty &\geq \max_{i \geq k+1} \{|w_i^\star|\} \\
&= \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{k+1} \\
&= \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{k} \|w^0 - w^\star\|_\infty.
\end{aligned}$$

As n gets large, the condition number of f^n approaches κ. And the gap therefore closes; the convergence rate that momentum promises matches the best any linear first order algorithm can do. And we arrive at the disappointing conclusion that on this problem, we cannot do better.

Like many such lower bounds, this result must not be taken literally, but spiritually. It, perhaps, gives a sense of closure and finality to our investigation. But this is not the final word on first order optimization. This lower bound does not preclude the possibility, for example, of reformulating the problem to change the condition number itself! There is still much room for speedups, if you understand the right places to look.

Momentum with Stochastic Gradients

There is a final point worth addressing. All the discussion above assumes access to the true gradient — a luxury seldom afforded in modern machine learning. Computing the exact gradient requires a full pass over all the data, the cost of which can be prohibitively expensive. Instead, randomized approximations of the gradient, like minibatch sampling, are often used as a plug-in replacement of $\nabla f(w)$. We can write the approximation in two parts,

$$\nabla f(w) \quad + \quad \text{error}(w).$$

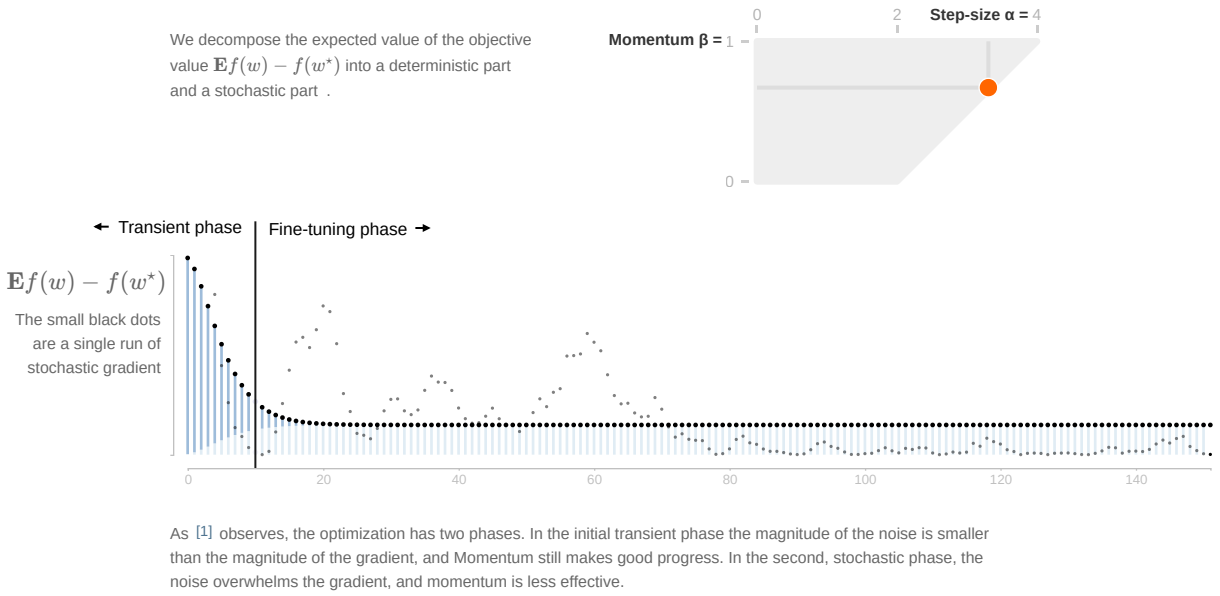
the true gradient — and an approximation error. If the estimator is unbiased e.g. $\mathbf{E}[\text{error}(w)] = 0$

It is helpful to think of our approximate gradient as the injection of a special kind of noise into our iteration. And using the machinery developed in the previous sections, we can deal with this extra term directly. On a quadratic, the error term cleaves cleanly into a separate term, where [10]

$$\begin{pmatrix} y_i^k \\ x_i^k \end{pmatrix} \quad = \quad R^k \begin{pmatrix} y_i^0 \\ x_i^0 \end{pmatrix} \quad + \quad \epsilon_i^k \sum_{j=1}^{k} R^{k-j} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}$$

the noisy iterates are a sum of — the noiseless, deterministic iterates and — a decaying sum of the errors, where $\epsilon^k = Q \cdot \text{error}(w^k)$.

The error term, ϵ^k, with its dependence on the w^k, is a fairly hairy object. Following [10], we model this as independent 0-mean Gaussian noise. In this simplified model, the objective also breaks into two separable components, a sum of a deterministic error and a stochastic error [11], visualized here.



As [1] observes, the optimization has two phases. In the initial transient phase the magnitude of the noise is smaller than the magnitude of the gradient, and Momentum still makes good progress. In the second, stochastic phase, the noise overwhelms the gradient, and momentum is less effective.

Note that there are a set of unfortunate tradeoffs which seem to pit the two components of error against each other. Lowering the step-size, for example, decreases the stochastic error, but also slows down the rate of convergence. And increasing momentum, contrary to popular belief, causes the errors to compound. Despite these undesirable properties, stochastic gradient descent with momentum has still been shown to have competitive performance on neural networks. As [1] has observed, the transient phase seems to matter more than the fine-tuning phase in machine learning. And in fact, it has been recently suggested [12] that this noise is a good thing — it acts as a implicit regularizer, which, like early stopping, prevents overfitting in the fine-tuning phase of optimization.

Onwards and Downwards

The study of acceleration is seeing a small revival within the optimization community. If the ideas in this article excite you, you may wish to read [13], which fully explores the idea of momentum as the discretization of a certain differential equation. But other, less physical, interpretations exist. There is an algebraic interpretation of momentum in terms of approximating polynomials [3, 14]. Geometric interpretations are emerging [15, 16], connecting momentum to older methods, like the Ellipsoid method. And finally, there are interpretations relating momentum to duality [17], perhaps providing a clue as how to accelerate second order methods and Quasi Newton (for a first step, see [18]). But like the proverbial blind men feeling an elephant, momentum seems like something bigger than the sum of its parts. One day, hopefully soon, the many perspectives will converge into a satisfying whole.

Acknowledgments

I am deeply indebted to the editorial contributions of Shan Carter and Chris Olah, without which this article would be greatly impoverished. Shan Carter provided complete redesigns of many of my original interactive widgets, a visual coherence for all the figures, and valuable optimizations to the page's performance. Chris Olah provided impeccable editorial feedback at all levels of detail and abstraction - from the structure of the content, to the alignment of equations.

I am also grateful to Michael Nielsen for providing the title of this article, which really tied the article together. Marcos Ginestra provided editorial input for the earliest drafts of this article, and spiritual encouragement when I needed it the most. And my gratitude extends to my reviewers, Matt Hoffman and Anonymous Reviewer B for their astute observations and criticism. I would like to thank Reviewer B, in particular, for pointing out two non-trivial errors in the original manuscript (discussion here). The contour plotting library for the hero visualization is the joint work of Ben Frederickson, Jeff Heer and Mike Bostock.

Many thanks to the numerous pull requests and issues filed on github. Thanks in particular, to Osemwaro Pedro for spotting an off by one error in one of the equations. And also to Dan Schmidt who did an editing pass over the whole project, correcting numerous typographical and grammatical errors.

Discussion and Review

Reviewer A - Matt Hoffman
Reviewer B - Anonymous
Discussion with User derifatives

Footnotes

1. It is possible, however, to construct very specific counterexamples where momentum does not converge, even on convex functions. See [4] for a counterexample.
2. In Tikhonov Regression we add a quadratic penalty to the regression, minimizing

$$\text{minimize} \qquad \tfrac{1}{2}\|Zw - d\|^2 + \frac{\eta}{2}\|w\|^2 = \tfrac{1}{2}w^T(Z^TZ + \eta I)w - (Zd)^Tw$$

Recall that $Z^TZ = Q\ \text{diag}(\Lambda_1, \dots, \Lambda_n)\ Q^T$. The solution to Tikhonov Regression is therefore

$$(Z^TZ + \eta I)^{-1}(Zd) = Q\ \text{diag}\left(\frac{1}{\lambda_1 + \eta}, \cdots, \frac{1}{\lambda_n + \eta}\right) Q^T(Zd)$$

We can think of regularization as a function which decays the largest eigenvalues, as follows:

$$\text{Tikhonov Regularized } \lambda_i = \frac{1}{\lambda_i + \eta} = \frac{1}{\lambda_i}\left(1 - (1 + \lambda_i/\eta)^{-1}\right).$$

Gradient descent can be seen as employing a similar decay, but with the decay rate

$$\text{Gradient Descent Regularized } \lambda_i = \frac{1}{\lambda_i}\left(1 - (1 - \alpha\lambda_i)^k\right)$$

instead. Note that this decay is dependent on the step-size.
3. This is true as we can write updates in matrix form as

$$\begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} y_i^{k+1} \\ x_i^{k+1} \end{pmatrix} = \begin{pmatrix} \beta & \lambda_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_i^k \\ x_i^k \end{pmatrix}$$

which implies, by inverting the matrix on the left,

$$\begin{pmatrix} y_i^{k+1} \\ x_i^{k+1} \end{pmatrix} = \begin{pmatrix} \beta & \lambda_i \\ -\alpha\beta & 1-\alpha\lambda_i \end{pmatrix} \begin{pmatrix} y_i^{k} \\ x_i^{k} \end{pmatrix} = R^{k+1} \begin{pmatrix} x_i^{0} \\ y_i^{0} \end{pmatrix}$$

4. We can write out the convergence rates explicitly. The eigenvalues are

$$\sigma_1 = \frac{1}{2}\left(1-\alpha\lambda+\beta+\sqrt{(-\alpha\lambda+\beta+1)^2-4\beta}\right)$$
$$\sigma_2 = \frac{1}{2}\left(1-\alpha\lambda+\beta-\sqrt{(-\alpha\lambda+\beta+1)^2-4\beta}\right)$$

   When the $(-\alpha\lambda+\beta+1)^2-4\beta < 0$ is less than zero, then the roots are complex and the convergence rate is

$$|\sigma_1| = |\sigma_2| = \sqrt{(1-\alpha\lambda+\beta)^2 + |(-\alpha\lambda+\beta+1)^2-4\beta|} = 2\sqrt{\beta}$$

   Which is, surprisingly, independent of the step-size or the eigenvalue $\alpha\lambda$. When the roots are real, the convergence rate is

$$\max\{|\sigma_1|,|\sigma_2|\} = \tfrac{1}{2}\max\left\{|1-\alpha\lambda_i+\beta \pm \sqrt{(1-\alpha\lambda_i+\beta)^2-4\beta}|\right\}$$

5. This can be derived by reducing the inequalities for all 4 + 1 cases in the explicit form of the convergence rate above.
6. We must optimize over

$$\min_{\alpha,\beta}\max\left\{\left\|\begin{pmatrix} \beta & \lambda_i \\ -\alpha\beta & 1-\alpha\lambda_i \end{pmatrix}\right\|, \ldots, \left\|\begin{pmatrix} \beta & \lambda_n \\ -\alpha\beta & 1-\alpha\lambda_n \end{pmatrix}\right\|\right\}.$$

   ($\|\cdot\|$ here denotes the magnitude of the maximum eigenvalue), and occurs when the roots of the characteristic polynomial are repeated for the matrices corresponding to the extremal eigenvalues.
7. The above optimization problem is bounded from below by 0, and vector of all 1's achieve this.
8. This can be written explicitly as

$$[L_G]_{ij} = \begin{cases} \text{degree of vertex } i & i = j \\ -1 & i \neq j, (i,j) \text{ or } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$$

9. We use the infinity norm to measure our error, similar results can be derived for the 1 and 2 norms.
10. The momentum iterations are

$$z^{k+1} = \beta z^k + Aw^k + \text{error}(w^k)$$
$$w^{k+1} = w^k - \alpha z^{k+1}.$$

    which, after a change of variables, become

$$\begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} y_i^{k+1} \\ x_i^{k+1} \end{pmatrix} = \begin{pmatrix} \beta & \lambda_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_i^{k} \\ x_i^{k} \end{pmatrix} + \begin{pmatrix} \epsilon_i^{k} \\ 0 \end{pmatrix}$$

    Inverting the 2×2 matrix on the left, and applying the formula recursively yields the final solution.
11. On the 1D function $f(x) = \frac{\lambda}{2}x^2$, the objective value is

$$\begin{aligned}
\mathbf{E}f(x^k) &= \frac{\lambda}{2}\mathbf{E}[(x^k)^2] \\
&= \frac{\lambda}{2}\mathbf{E}\left(e_2^T R^k \begin{pmatrix} y^0 \\ x^0 \end{pmatrix} + \epsilon^k e_2^T \sum_{i=1}^{k} R^{k-i} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}\right)^2 \\
&= \frac{\lambda}{2} e_2^T R^k \begin{pmatrix} y^0 \\ x^0 \end{pmatrix} + \frac{\lambda}{2}\mathbf{E}\left(\epsilon^k e_2^T \sum_{i=1}^{k} R^{k-i} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}\right)^2 \\
&= \frac{\lambda}{2} e_2^T R^k \begin{pmatrix} y^0 \\ x^0 \end{pmatrix} + \frac{\lambda}{2}\mathbf{E}[\epsilon^k] \cdot \sum_{i=1}^{k}\left(e_2^T R^{k-i} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}\right)^2 \\
&= \frac{\lambda}{2} e_2^T R^k \begin{pmatrix} y^0 \\ x^0 \end{pmatrix} + \frac{\lambda\mathbf{E}[\epsilon^k}{2} \cdot \sum_{i=1}^{k}\gamma_i^2, \qquad \gamma_i = e_2^T R^{k-i} \begin{pmatrix} 1 \\ -\alpha \end{pmatrix}
\end{aligned}$$

    The third inequality uses the fact that $\mathbf{E}\epsilon^k = 0$ and the fourth uses the fact they are uncorrelated.

References

1. **On the importance of initialization and momentum in deep learning.** [PDF]
   Sutskever, I., Martens, J., Dahl, G.E. and Hinton, G.E., 2013. ICML (3), Vol 28, pp. 1139—1147.
2. **Some methods of speeding up the convergence of iteration methods** [PDF]
   Polyak, B.T., 1964. USSR Computational Mathematics and Mathematical Physics, Vol 4(5), pp. 1—17. Elsevier. DOI: 10.1016/0041-5553(64)90137-5
3. **Theory of gradient methods**
   Rutishauser, H., 1959. Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems, pp. 24—49. Springer. DOI: 10.1007/978-3-0348-7224-9_2

4. **Analysis and design of optimization algorithms via integral quadratic constraints** [PDF]
   Lessard, L., Recht, B. and Packard, A., 2016. SIAM Journal on Optimization, Vol 26(1), pp. 57—95. SIAM.
5. **Introductory lectures on convex optimization: A basic course**
   Nesterov, Y., 2013. , Vol 87. Springer Science \& Business Media. DOI: 10.1007/978-1-4419-8853-9
6. **Natural gradient works efficiently in learning** [link]
   Amari, S., 1998. Neural computation, Vol 10(2), pp. 251—276. MIT Press. DOI: 10.1162/089976698300017746
7. **Deep Learning, NIPS'2015 Tutorial** [PDF]
   Hinton, G., Bengio, Y. and LeCun, Y., 2015.
8. **Adaptive restart for accelerated gradient schemes** [PDF]
   O'Donoghue, B. and Candes, E., 2015. Foundations of computational mathematics, Vol 15(3), pp. 715—732. Springer. DOI: 10.1007/s10208-013-9150-3
9. **The Nth Power of a 2x2 Matrix.** [PDF]
   Williams, K., 1992. Mathematics Magazine, Vol 65(5), pp. 336. MAA. DOI: 10.2307/2691246
10. **From Averaging to Acceleration, There is Only a Step-size.** [PDF]
    Flammarion, N. and Bach, F.R., 2015. COLT, pp. 658—695.
11. **On the momentum term in gradient descent learning algorithms** [PDF]
    Qian, N., 1999. Neural networks, Vol 12(1), pp. 145—151. Elsevier. DOI: 10.1016/s0893-6080(98)00116-6
12. **Understanding deep learning requires rethinking generalization** [PDF]
    Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O., 2016. arXiv preprint arXiv:1611.03530.
13. **A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights** [PDF]
    Su, W., Boyd, S. and Candes, E., 2014. Advances in Neural Information Processing Systems, pp. 2510—2518.
14. **The Zen of Gradient Descent** [HTML]
    Hardt, M., 2013.
15. **A geometric alternative to Nesterov's accelerated gradient descent** [PDF]
    Bubeck, S., Lee, Y.T. and Singh, M., 2015. arXiv preprint arXiv:1506.08187.
16. **An optimal first order method based on optimal quadratic averaging** [PDF]
    Drusvyatskiy, D., Fazel, M. and Roy, S., 2016. arXiv preprint arXiv:1604.06543.
17. **Linear coupling: An ultimate unification of gradient and mirror descent** [PDF]
    Allen-Zhu, Z. and Orecchia, L., 2014. arXiv preprint arXiv:1407.1537.
18. **Accelerating the cubic regularization of Newton's method on convex problems** [PDF]
    Nesterov, Y., 2008. Mathematical Programming, Vol 112(1), pp. 159—181. Springer. DOI: 10.1007/s10107-006-0089-x

Updates and Corrections

View all changes to this article since it was first published. If you see a mistake or want to suggest a change, please create an issue on GitHub.

Citations and Reuse

For attribution in academic contexts, please cite this work as

```
Goh, "Why Momentum Really Works", Distill, 2017. http://doi.org/10.23915/distill.00006
```

BibTeX citation

```
@article{goh2017why,
  author = {Goh, Gabriel},
  title = {Why Momentum Really Works},
  journal = {Distill},
  year = {2017},
  url = {http://distill.pub/2017/momentum},
  doi = {10.23915/distill.00006}
}
```