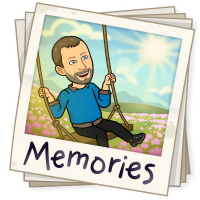


Machine Learning

Textbook: Machine Learning Refined, by Watt, Borhani, and Katsaggelos, 2nd edition:
MLRefined.com

Previous Lecture

- ◆ Gradient descent repeatedly moves opposite the gradient.
- ◆ $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla g(\mathbf{w}_k)$.
- ◆ The local linear model predicts a decrease of about $\alpha \|\nabla g\|^2$ for a sufficiently small step.
- ◆ The gradient must be recomputed after every step.
- ◆ The learning rate α controls how far we move.
- ◆ A Python `for` loop lets us carry out repeated gradient-descent updates.



§3.5, 3.6 - Step Size and Convergence

Last time:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla g(\mathbf{w}_k).$$

Gradient descent tells us which way to move.

But α tells us how far.

Today: when does the iteration actually converge?

Return to a Simple Quadratic

Let

$$g(w) = (w - 3)^2.$$

Then

$$g'(w) = 2(w - 3).$$

Gradient descent gives

$$w_{k+1} = w_k - 2\alpha(w_k - 3).$$

The minimum is

$$w^* = 3.$$

Question:

for which values of α do the iterates approach 3?

Track the Error Instead

Define the error

$$e_k = w_k - 3.$$

Then

$$e_{k+1} = w_{k+1} - 3.$$

Using the gradient-descent update:

$$e_{k+1} = w_k - 2\alpha(w_k - 3) - 3.$$

So

$$e_{k+1} = (1 - 2\alpha)e_k.$$

Every step multiplies the previous error by the same number.

After Many Steps

Repeatedly applying

$$e_{k+1} = (1 - 2\alpha)e_k$$

gives

$$e_k = (1 - 2\alpha)^k e_0.$$

For convergence we need

$$e_k \rightarrow 0.$$

Therefore

$$|1 - 2\alpha| < 1.$$

Solve:

$$-1 < 1 - 2\alpha < 1,$$

$$-2 < -2\alpha < 0,$$

$$0 < \alpha < 1.$$

For this cost function, gradient descent converges exactly when

$$0 < \alpha < 1.$$

Small Step: Slow but Steady

Take

$$\alpha = 0.1.$$

Then

$$1 - 2\alpha = 0.8,$$

$$e_k = 0.8^k e_0.$$

Same sign each time.

The iterates approach the minimum from one side.

k	0	1	2	3	4
$ e_k / e_0 $	1	0.8	0.64	0.512	0.4096

Stable — but not especially fast.

A Special Step Size

Take

$$\alpha = 0.5.$$

Then

$$1 - 2\alpha = 0.$$

Thus

$$e_1 = 0.$$

For this quadratic:

one step lands exactly at the minimum.

This is special to the shape of this particular function.

Larger Step: Oscillation

Take

$$\alpha = 0.75.$$

Then

$$1 - 2\alpha = -0.5.$$

So

$$e_k = (-0.5)^k e_0.$$

Magnitude shrinks.

Sign alternates.

The iterates jump across the minimum at every step.

But

$$|e_k| \rightarrow 0.$$

Oscillation does not necessarily mean failure.

At the Boundary

Take

$$\alpha = 1.$$

Then

$$1 - 2\alpha = -1.$$

Thus

$$e_k = (-1)^k e_0.$$

The iterates alternate forever.

The error does not shrink.

No convergence.

Too Large

Take

$$\alpha = 1.1.$$

Then

$$1 - 2\alpha = -1.2.$$

So

$$|e_k| = 1.2^k |e_0|.$$

The iterates alternate across the minimum and move farther away.

This is **divergence**.

What the Multiplier Tells Us

For

$$g(w) = (w - 3)^2,$$

the multiplier is

$$r = 1 - 2\alpha.$$

Multiplier	Behavior
$0 < r < 1$	monotone convergence
$-1 < r < 0$	oscillatory convergence
$r = 0$	exact arrival in one step
$ r = 1$	no decay of error
$ r > 1$	divergence

Convergence is a statement about repeated multiplication.

A General One-Variable Quadratic

Let

$$g(w) = a(w - w^*)^2,$$

where $a > 0$.

Then

$$g'(w) = 2a(w - w^*).$$

Gradient descent:

$$w_{k+1} = w_k - 2a\alpha(w_k - w^*).$$

With $e_k = w_k - w^*$:

$$e_{k+1} = (1 - 2a\alpha)e_k.$$

Therefore convergence requires

$$|1 - 2a\alpha| < 1,$$

so

$$0 < \alpha < \frac{1}{a}.$$

Larger curvature $a \Rightarrow$ smaller stable step size.

Back to Two Parameters

Use our familiar cost:

$$g(w_1, w_2) = (w_1 - 2)^2 + 4(w_2 + 1)^2.$$

Minimum:

$$(w_1^*, w_2^*) = (2, -1).$$

Gradient:

$$\nabla g = \begin{bmatrix} 2(w_1 - 2) \\ 8(w_2 + 1) \end{bmatrix}.$$

Define coordinate errors

$$e_{1,k} = w_{1,k} - 2,$$

$$e_{2,k} = w_{2,k} + 1.$$

Then gradient descent gives

$$e_{1,k+1} = (1 - 2\alpha)e_{1,k},$$

$$e_{2,k+1} = (1 - 8\alpha)e_{2,k}.$$

Each direction has its own multiplier.

One Step Size Must Work in Every Direction

For the first coordinate:

$$|1 - 2\alpha| < 1$$

requires

$$0 < \alpha < 1.$$

For the second coordinate:

$$|1 - 8\alpha| < 1$$

requires

$$0 < \alpha < \frac{1}{4}.$$

Both must hold.

Therefore

$$0 < \alpha < \frac{1}{4}.$$

The steeper direction controls the largest stable step.

Why Our Earlier Choice Worked

In Lesson 1.5 we used

$$\alpha = 0.1.$$

Then

$$1 - 2\alpha = 0.8,$$

$$1 - 8\alpha = 0.2.$$

So

$$e_{1,k+1} = 0.8e_{1,k},$$

$$e_{2,k+1} = 0.2e_{2,k}.$$

The second coordinate converges much faster.

This explains the behavior we saw in Python.

A Better Fixed Step for This Quadratic

Try

$$\alpha = 0.2.$$

Then

$$1 - 2\alpha = 0.6,$$

$$1 - 8\alpha = -0.6.$$

Both errors shrink by a factor of 0.6 in magnitude.

The w_2 coordinate oscillates.

But both coordinates converge at the same rate.

Compare with $\alpha = 0.1$:

first coordinate shrinks only by 0.8 each step.

Step size affects both stability and speed.

Poor Conditioning

Now consider

$$g(w_1, w_2) = w_1^2 + 100w_2^2.$$

Gradient:

$$\nabla g = \begin{bmatrix} 2w_1 \\ 200w_2 \end{bmatrix}.$$

Error multipliers:

$$1 - 2\alpha,$$

and

$$1 - 200\alpha.$$

The steep direction requires

$$0 < \alpha < 0.01.$$

But then the shallow direction has multiplier close to $1 - 2(0.01) = 0.98$.

Very slow.

The Geometry Behind Slow Convergence

For

$$g(w_1, w_2) = w_1^2 + 100w_2^2,$$

the level curves are long narrow ellipses.

One direction is shallow.

One direction is steep.

The steep direction forces a small α .

The shallow direction then moves slowly.

A cost surface with very different curvature in different directions is **poorly conditioned**.

Gradient descent may zigzag through a narrow valley.

Do We Need One Fixed Step Size?

Not necessarily.

Instead of using one constant α ,
choose a step size α_k at each iteration:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla g(\mathbf{w}_k).$$

One idea:
search along the downhill line.

Choose
$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \quad g(\mathbf{w}_k - \alpha \nabla g(\mathbf{w}_k)).$$

This is a **line search**.

Exact Versus Practical Line Search

Exact line search:
find the best α_k along the chosen line.

Often expensive.

Practical alternative:
start with a proposed step size;

if the cost does not decrease enough, reduce it;

try again.

This is the basic idea of **backtracking**.

Direction from the gradient.

Distance chosen adaptively.

Compare Step Sizes in Python

Use the one-variable quadratic:

```
def g1(w):  
    return (w - 3)**2  
  
def grad_g1(w):  
    return 2*(w - 3)
```

Run gradient descent:

```
def run_gd(alpha, steps=20):  
    w = 0.0  
    history = []  
  
    for k in range(steps):  
        history.append(g1(w))  
        w = w - alpha*grad_g1(w)  
  
    return history
```

New Python idea:

a function can return an entire list of values.

Plot the Cost History

Try several learning rates:

```
for alpha in [0.1, 0.5, 0.75, 1.1]:  
    history = run_gd(alpha)  
    plt.plot(history, label=str(alpha))  
  
plt.xlabel("Iteration")  
plt.ylabel("Cost")  
plt.legend()  
plt.show()
```

Expected behavior:

- $\alpha = 0.1$: steady decrease;
- $\alpha = 0.5$: reaches zero immediately;
- $\alpha = 0.75$: converges while oscillating in w ;
- $\alpha = 1.1$: diverges.

New Python idea:

label and plt identify several curves.

Stop When the Gradient Is Small

A fixed number of iterations is simple.

But we can stop when the algorithm is close to stationary.

```
w = 0.0
alpha = 0.1

for k in range(1000):
    if abs(grad_g1(w)) < 1e-6:
        break

    w = w - alpha*grad_g1(w)
```

New Python ideas:

if checks a condition;

break exits the loop.

Here we stop when

$|g'(w)| < 10^{-6}$.

Convergence Is Not Just “Cost Went Down Once”

A successful first step does not guarantee long-term convergence.

We care about the sequence

$w_0, w_1, w_2, \dots$

and ask whether

$w_k \rightarrow w^*$.

For quadratic functions we can often answer this exactly.

For more complicated cost functions:

local curvature changes as we move.

The safe step size may change too.

What did we learn?

- ◆ For a quadratic, gradient descent can be analyzed by tracking the error from the minimum.
- ◆ Convergence requires the error multiplier to have magnitude less than 1.
- ◆ Small learning rates can converge slowly; larger stable rates may oscillate but converge faster.
- ◆ If the learning rate is too large, gradient descent can diverge.
- ◆ Different curvature in different directions can force a small step size and slow convergence.
- ◆ A line search chooses the step size adaptively rather than fixing one value forever.



Next

- First-order information gives us gradient descent.

- Curvature determines much of its convergence behavior.
 - Next we describe curvature explicitly with second derivatives.
 - **1.7 - Second-Order Approximation**
-

Prepared by Dr. Jodin Morey.

Materials for Other Courses Found at MathTalker.org