

# CS545: Gradient Descent

Chuck Anderson

Department of Computer Science  
Colorado State University

Fall, 2009

# Outline

CS545: Gradient  
Descent

Chuck Anderson

Gradient Descent

Parabola

Examples in R

Gradient Descent

Parabola

Examples in R

## Finding Minimum of Parabola

- Find  $x$  that is minimum of  $f(x) = 1.2(x - 2)^2 + 3.2$  or, said another way, find  $\operatorname{argmax}_x f(x)$ . How?

## Finding Minimum of Parabola

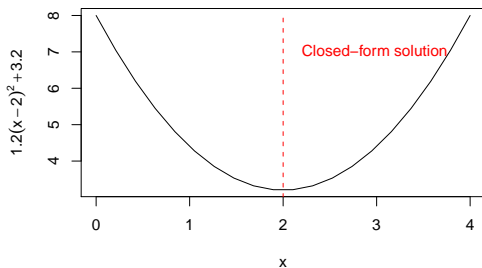
- Find  $x$  that is minimum of  $f(x) = 1.2(x - 2)^2 + 3.2$  or, said another way, find  $\operatorname{argmax}_x f(x)$ . How?
- Yep. Take derivative, set equal to zero, and try to solve for  $x$ .

$$f(x) = 1.2(x - 2)^2 + 3.2$$

$$\frac{df(x)}{dx} = 1.2(2)(x - 2) = 2.4(x - 2)$$

$$\frac{df(x)}{dx} = 0 = 2.4(x - 2)$$

$$x = 2$$



# Gradient Descent

- But, if  $\frac{df(x)}{dx}$  is cannot be solved directly for  $x$ , what can we do?

# Gradient Descent

- But, if  $\frac{df(x)}{dx}$  is cannot be solved directly for  $x$ , what can we do?
- Start at some  $x$  value, use derivative at that value to tell us which way to move, and repeat. Gradient descent.

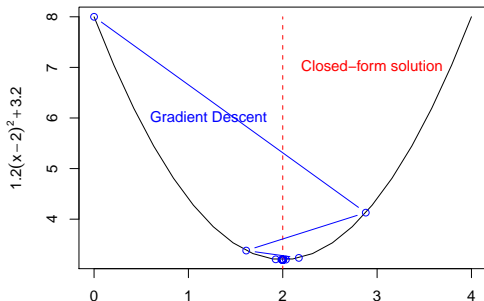
# Gradient Descent

- But, if  $\frac{df(x)}{dx}$  is cannot be solved directly for  $x$ , what can we do?
- Start at some  $x$  value, use derivative at that value to tell us which way to move, and repeat. Gradient descent.
- $\rho$  is factor of derivative to control how far to go

$$\frac{df(x)}{dx} = 2.4(x - 2)$$

$$x(0) = 0 \text{ (for example)}$$

$$x(n) = x(n - 1) - \rho 2.4(x - 2)$$



- For a parabola, can get there much faster if we also know the second derivative, which is what?

- For a parabola, can get there much faster if we also know the second derivative, which is what?

- For a parabola, can get there much faster if we also know the second derivative, which is what?

$$\frac{df(x)}{dx} = f' = 2.4(x - 2)$$

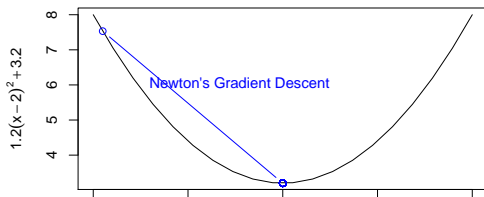
$$\frac{d^2f(x)}{dx^2} = f'' = 2.4$$

- and use Newton's method (see the Wikipedia entry for "Newton's method")

$$x(n) = x(n-1) - \frac{f'}{f''}$$

$$x(n) = x(n-1) - \frac{2.4(x-2)}{2.4}$$

$$x(n) = x(n-1) - (x-2)$$



- If the function is not a parabola, what can we do?  
Cannot solve directly for  $x$ . Can still do gradient descent. Can we always use Newton's method?

- If the function is not a parabola, what can we do?  
Cannot solve directly for  $x$ . Can still do gradient descent. Can we always use Newton's method?
- No. Reason 1: If  $x$  has 1000 components, the second derivative (Hessian) is a  $1000 \times 1000$  matrix. May be too big.

- If the function is not a parabola, what can we do?  
Cannot solve directly for  $x$ . Can still do gradient descent. Can we always use Newton's method?
- No. Reason 1: If  $x$  has 1000 components, the second derivative (Hessian) is a  $1000 \times 1000$  matrix. May be too big.
- Reason 2: If not a parabola the second derivative information may lead you very far away. When?

# Approximating the Second Derivative

- Say we have picked a direction,  $p$ , to go. Rather than compute the second derivative in that direction, we can approximate it using two first derivative values.

$$f''(x)p \approx \frac{f'(x + \alpha p) - f'(x)}{\alpha} \text{ for } 0 < \alpha \ll 1$$

# Approximating the Second Derivative

- Say we have picked a direction,  $p$ , to go. Rather than compute the second derivative in that direction, we can approximate it using two first derivative values.

$$f''(x)p \approx \frac{f'(x + \alpha p) - f'(x)}{\alpha} \text{ for } 0 < \alpha \ll 1$$

- In practice, Moller found he had to modify this by adding  $\lambda p$  where  $\lambda$  is set to a value for which the resulting approximated second derivative is well behaved.

$$f''(x)p \approx \frac{f'(x + \alpha p) - f'(x)}{\alpha} + \lambda p, \text{ for } 0 < \alpha \ll 1$$

# Approximating the Second Derivative

- Say we have picked a direction,  $p$ , to go. Rather than compute the second derivative in that direction, we can approximate it using two first derivative values.

$$f''(x)p \approx \frac{f'(x + \alpha p) - f'(x)}{\alpha} \text{ for } 0 < \alpha \ll 1$$

- In practice, Moller found he had to modify this by adding  $\lambda p$  where  $\lambda$  is set to a value for which the resulting approximated second derivative is well behaved.

$$f''(x)p \approx \frac{f'(x + \alpha p) - f'(x)}{\alpha} + \lambda p, \text{ for } 0 < \alpha \ll 1$$

- This gives us a way to scale the step size.

# Picking a Good Direction

- Now, how about that direction? How do we decide that?

# Picking a Good Direction

- Now, how about that direction? How do we decide that?
- Moller uses conjugate gradients. (See the wikipedia entry for “conjugate gradient”)

# Picking a Good Direction

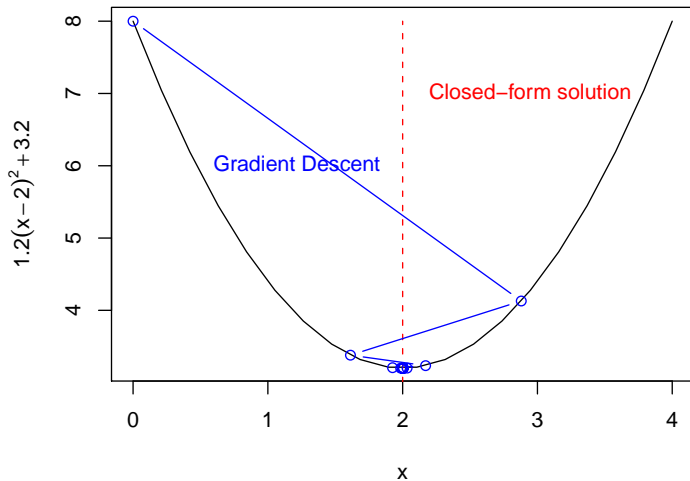
- Now, how about that direction? How do we decide that?
- Moller uses conjugate gradients. (See the wikipedia entry for “conjugate gradient”)
- The conjugate gradient direction is based on the previous direction and the current gradient.

# Parabola Example

```
f <- function(x) {  
  1.2 * (x-2)^2 + 3.2  
}  
  
grad <- function(x) {  
  1.2 * 2 * (x-2)  
}  
  
secondGrad <- function(x) {  
  2.4  
}
```

# Steepest Descent

```
xs <- seq(0,4,len=20)
plot(xs, f(xs), type="l", xlab="x", ylab=expression(1.2(x-2)^2 + 3.2))
### df/dx = 2.4(x-2)
### df/dx = 0 ----> 0 = 2.4x - 4.8 ----> x = 2
lines(c(2,2), c(3,8), col="red", lty=2)
text(2.1,7, "Closed-form solution", col="red", pos=4)
### gradient descent
x <- 0.1
xtrace <- x
ftrace <- f(x)
stepFactor <- 0.6 ### try larger and smaller values (0.8 and 0.01)
for (step in 1:100) {
  x <- x - stepFactor * grad(x)
  xtrace <- c(xtrace,x)
  ftrace <- c(ftrace,f(x))
}
lines(xtrace, ftrace, type="b", col="blue")
text(0.5,6, "Gradient Descent", col="blue", pos=4)
```



# Steepest Descent with gradientDescents.R

CS545: Gradient  
Descent

Chuck Anderson

Gradient Descent  
Parabola  
Examples in R

```
source("gradientDescents.R")
x <- 0.1
result <- steepest(x, f, grad, stepsize=0.6, niterations=100, xtrace=TRUE, ftrace=TRUE)
plot(xs, f(xs), type="l", xlab="x", ylab=expression(1.2(x-2)^2 + 3.2))
lines(result$xtrace, result$ftrace, type="b", col="blue")
text(0.5, 6, "Gradient Descent with steepest()", col="blue", pos=4)
```

# Steepest Descent scaled with Newton's Method

```
plot(xs, f(xs), type="l", xlab="x", ylab=expression(1.2(x-2)^2 + 3.2))
x <- 0.1
xtrace <- x
ftrace <- f(x)
for (step in 1:100) {
  x <- x - grad(x)/secondGrad(x)
  xtrace <- c(xtrace,x)
  ftrace <- c(ftrace,f(x))
}
lines(xtrace, ftrace, type="b", col="blue")
text(0.5, 6, "Newton's Gradient Descent", col="blue", pos=4)
```

# With Scaled Conjugate Gradient from gradientDescents.R

```
source("gradientDescents.R")
x <- 0.1
result <- scg(x, f, grad, nIterations = 100, xTrace = TRUE, fTrace = TRUE)
plot(xs, f(xs), type = "l", xlab = "x", ylab = expression(1.2(x-2)^2 + 3.2))
lines(result$xTrace, result$fTrace, type = "b", col = "blue")
text(0.5, 6, "Gradient Descent with scg()", col = "blue", pos = 4)
```

# Results

