Recap: Optimization.

1) Critical pts, max/min on boundary
(find max/min of easy function on a region)

2) Lagrange multiplies
(max/min a function with a constraint)
easy

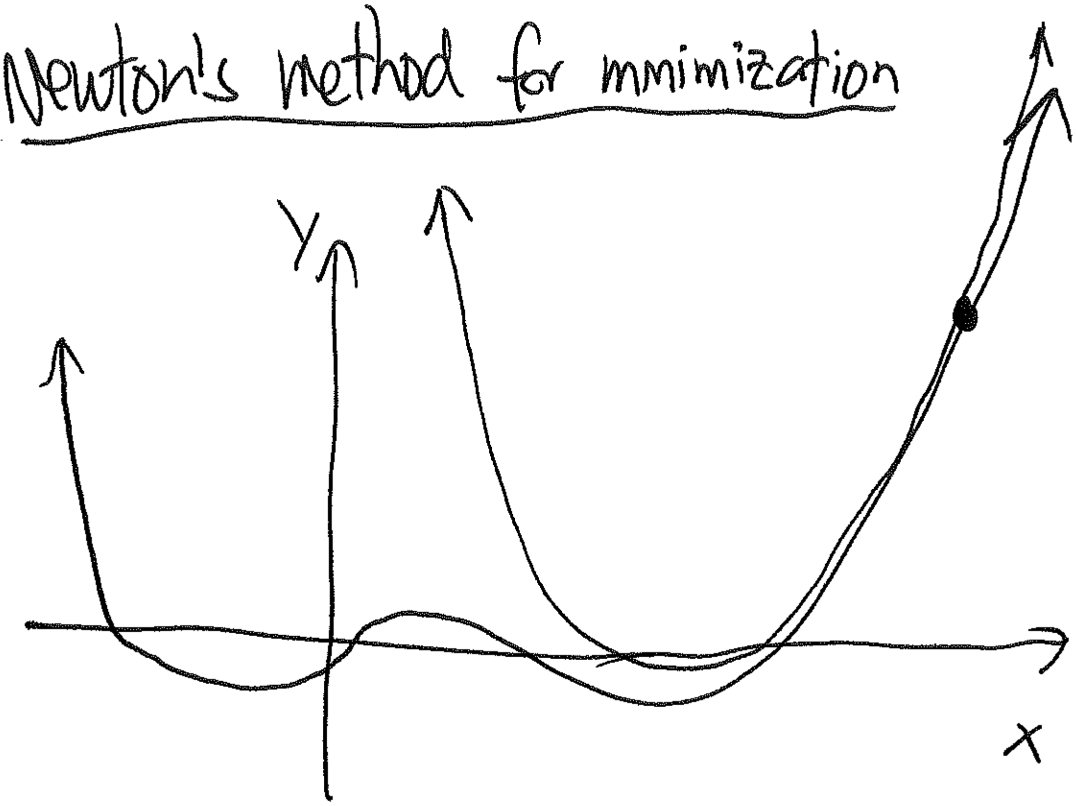
3) Linear programming
(max/min of linear function with many liner constraints: there's an algorithm)
simplex method

4) Newton's method (for roots)
(for max/min)

(complicated functions: decimal approx of max/min without finding it exactly)

5) Gradient descent: solves similar problems to Newton's method.

Newton's method for minimization



To find minimum of $f(x)$:

1) Make an initial guess x_0

2) Find parabola $g(x)$ through $(x_0, f(x_0))$

   that has $f(x_0)=g(x_0)$, $f'(x_0)=g'(x_0)$, $f''(x_0)=g''(x_0)$

$$g(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2}(x-x_0)^2$$

Guess: minimum of f is near minimum of g:

To find minimum of $g(x)$:

$$g'(x) = f'(x_0) + f''(x_0)(x - x_0) = 0.$$

$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)}$$

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

our next guess for minimum!

$$\boxed{x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}}$$

this is the same as Newton's method for a root of f'!

(probably finds a local minimum only)

To find absolute: find all local minima by trying different x_0, then compare them.

What about optimizing functions of two variables?
(or more)

<u>Newton's method for optimization:</u>

Given $f(x,y)$:

want a point (x,y) so $f_x(x,y)=f_y(x,y)=0$

1) Make a guess (x_0,y_0)

2) Find a paraboloid

$$g(x,y)=ax^2+bxy+cy^2+dx+ey+f$$

that has maximum to $f(x,y)$ at (x_0,y_0):

$$g(x_0,y_0)=f(x_0,y_0) \qquad g_{xx}=f_{xx}$$
$$g_x(x_0,y_0)=f_x(x_0,y_0) \qquad g_{xy}=f_{xy}$$
$$g_y(x_0,y_0)=f_y(x_0,y_0) \qquad g_{yy}=f_{yy}$$

~~osc~~ Lat: "osculating paraboloid"

3) Take (x_1,y_1) to be minimum, and iterate

Gradient descent.

To find minimum of $f(x,y)$.

1) Make an initial guess (x_0, y_0)

2) Then move in direction of fastest decrease, which is

$$-\nabla f(x_0, y_0).$$

γ_n = step size: how far do you in that direction?

$$(x_{n+1}, y_{n+1}) = (x_n, y_n) - \gamma_n \nabla f(x_n, y_n)$$

If γ_n is small, you'll frequently update direction, but you'll have to compute gradient many times!

Picking correct γ_n is tricky.