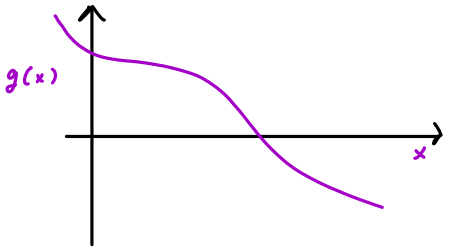
Optimization of continuous functions

Preliminary related problem: root finding

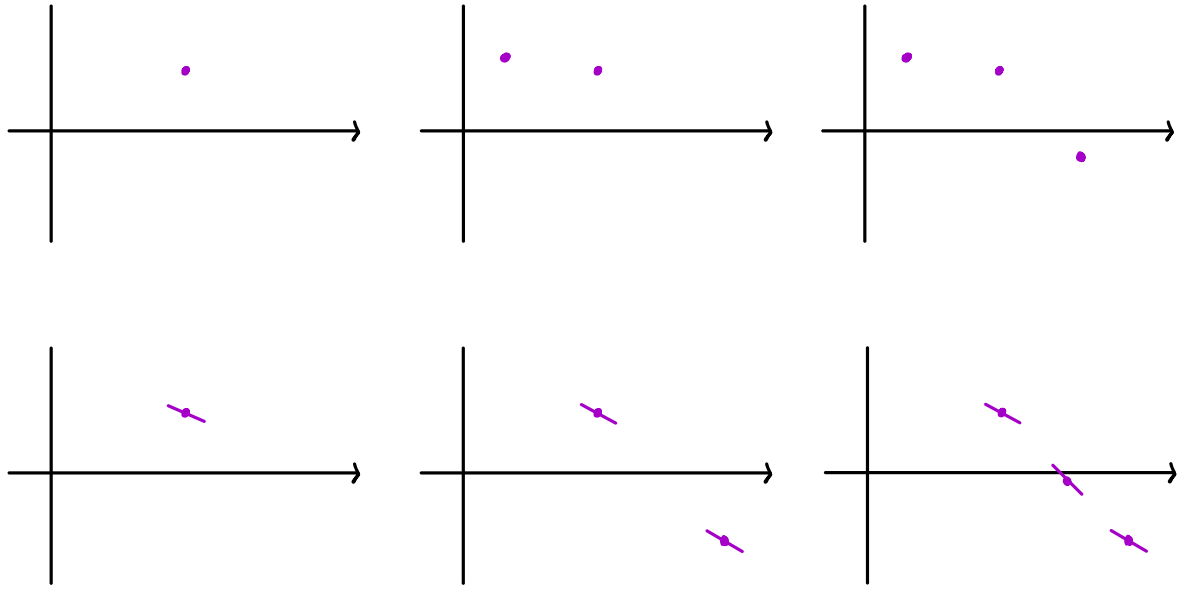


Find x such that $g(x) = 0$

Problem is a bit more complicated than it looks

What you get to see: $(x_1, g(x_1))$, $(x_2, g(x_2))$, ...

possibly: $(x_1, g(x_1), g'(x_1))$, $(x_2, g(x_2), g'(x_2))$, ...



Newton's method of root finding

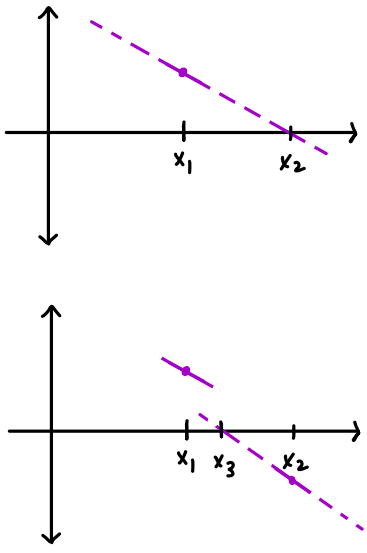
Requires knowing $g(x)$, $g'(x)$

* guess x_1, calculate $g(x_1)$, $g'(x_1)$
* find root of linear surrogate

$$l(x) = g(x_1) + g'(x_1)(x - x_1)$$

$$0 = g(x_1) + g'(x_1)(x_2 - x_1)$$

$$x_2 = x_1 - \frac{g(x_1)}{g'(x_1)}$$



* repeat until $g(x_i)$ is close enough to 0.

Example:

calculate square root of 7 $\longleftrightarrow$ find x such that $x^2 - 7 = 0$

$g(x) = x^2 - 7 \qquad g'(x) = 2x$

$$x_{i+1} = x_i - \frac{x_i^2 - 7}{2x_i} = \frac{1}{2}\left(\frac{x_i}{2} + \frac{7}{2x_i}\right)$$

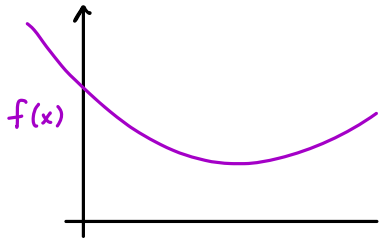
Babylonian Method

Finding the minimum (or maximum) of a function

find x such that $f(x) \leq f(x^*)$ for every x^*

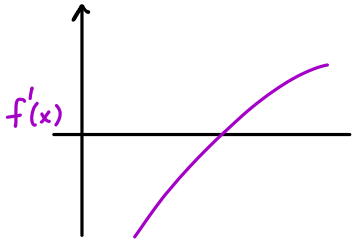
convert to a root-finding problem.

find x such that $f'(x) = 0$, $f''(x) > 0$



Newton's method for minimization

* guess x_0
* set $x_1 = x_0 - f'(x_0) / f''(x_0)$
* set $x_2 = x_1 - f'(x_1) / f''(x_1)$
* keep going. you're doing great.
* repeat until $f'(x)$ is small enough
* check $f''(x) > 0$



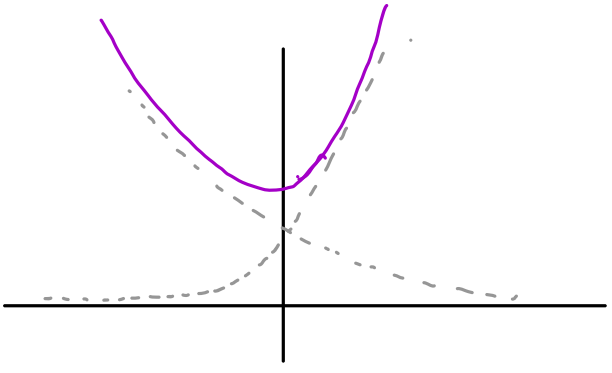
Example:

$$f(x) = \exp(2x-4) + \exp(2-x)$$

$$f'(x) = 2\exp(2x-4) - \exp(2-x)$$

$$f''(x) = 4\exp(2x-4) + \exp(2-x)$$

$$x_{i+1} = x_i - \frac{2\exp(2x-4) - \exp(2-x)}{4\exp(2x-4) + \exp(2-x)}$$



Moving downhill

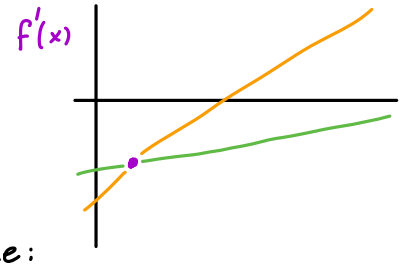
If we can calculate $f'(x)$ but not $f''(x)$,

we can't use newton, But we are not out of luck

we should move in the direction of $-f'(x_i)$.

but need to guess how far to move

$x_{i+1} = x_i - f'(x_i)\,c$ (c is our guess)



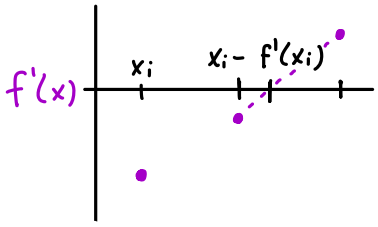
This is called <u>gradient descent</u>

There are various guessing algorithms. Here's one:

* Given x_i , calculate $f'(x_i)$
* set $c = 1$.

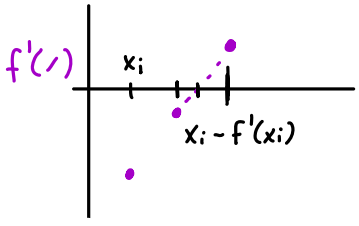
if $f'(x_i - f'(x_i)c)$ has same sign as $f'(x_i)$

double c until $f'(x_i - f'(x_i)c)$ changes sign.



if $f'(x_i - f'(x_i)c)$ has opposite sign as $f'(x_i)$

halve c until $f'(x_i - f'(x_i)c)$ changes sign.



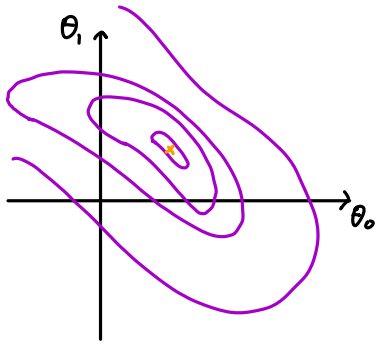
x_{i+1} is root of linear approx to $f'(x)$ using

two values that straddle 0.

Minimization of multivariable functions

Example: multiple linear regression. minimize

$$f(\underline{b}) = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{p} x_{ij} b_j \right)^2 \quad \text{over } \underline{b} = (b_0, \ldots, b_p)$$



Example: maximum likelihood estimation

Data: $(y_1, \ldots, y_n) = \underline{y}$

Model: $Y_i \overset{ind}{\sim} p(z\,;\, \theta_1, \ldots, \theta_k) \qquad \underline{\theta} = (\theta_1, \ldots, \theta_k)$

maximize **log-likelihood function** over $\underline{\theta}$

$$\ell(\underline{\theta}) = \log\left(\prod_{i=1}^{n} p(y_i\,;\, \theta_1, \ldots, \theta_k) \right) = \sum_{i=1}^{n} \log p(y_i\,;\, \theta_1, \ldots, \theta_k)$$

First and second derivatives of multivariable functions:

$f(\underline{x})$: vector input $\underline{x} = (x_1, \ldots, x_k)$, scalar output

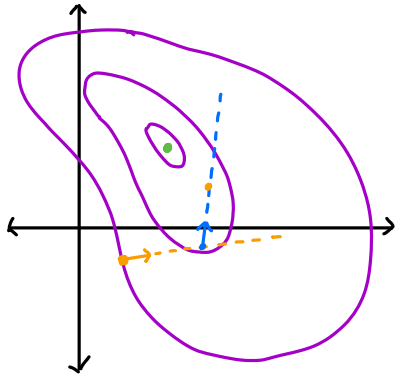
Gradient: Derivative of function with respect to each input

$$\underset{k\times 1}{\nabla f(\underline{x})} = \begin{bmatrix} \frac{\partial f(\underline{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\underline{x})}{\partial x_k} \end{bmatrix}$$

Example: $f(\underline{b}) = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2$

$$\underset{2\times 1}{\nabla f(\underline{b})} = \begin{bmatrix} -2 \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) \\ -2 \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) x_i \end{bmatrix}$$

Gradient descent



* given $\underline{x}_i$, compute $f(\underline{x}_i)$, $\nabla f(\underline{x}_i)$
* set $\underline{x}_{i+1} = \underline{x}_i - \nabla f(\underline{x}_i)\, c$, choosing c so that $f(\underline{x}_{i+1}) < f(\underline{x}_i)$
* various methods for finding c, but within each iteration, do a minimization over c

Moving along gradient guarantees that you go downhill "locally"
But it's not necessarily the optimal search direction.

Newton's method tries to find a better search direction,
AND gives a suggestion for how far to move.

Hessian: "Gradient of the gradient

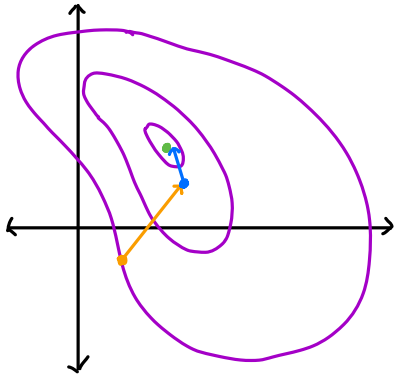
$$\underset{k\times k}{\nabla^2 f(\underline{x})} = \begin{bmatrix} \frac{\partial}{\partial x_1}\left(\frac{\partial f(\underline{x})}{\partial x_1}\right) & \frac{\partial}{\partial x_2}\left(\frac{\partial f(\underline{x})}{\partial x_1}\right) & \cdots & \frac{\partial}{\partial x_k}\left(\frac{\partial f(\underline{x})}{\partial x_1}\right) \\ \frac{\partial}{\partial x_1}\left(\frac{\partial f(\underline{x})}{\partial x_2}\right) & \frac{\partial}{\partial x_2}\left(\frac{\partial f(\underline{x})}{\partial x_2}\right) & \cdots & \frac{\partial}{\partial x_k}\left(\frac{\partial f(\underline{x})}{\partial x_2}\right) \\ \vdots & \vdots & & \vdots \\ \frac{\partial}{\partial x_1}\left(\frac{\partial f(\underline{x})}{\partial x_k}\right) & \frac{\partial}{\partial x_2}\left(\frac{\partial f(\underline{x})}{\partial x_k}\right) & \cdots & \frac{\partial}{\partial x_k}\left(\frac{\partial f(\underline{x})}{\partial x_k}\right) \end{bmatrix}$$

Example: $\underset{2\times 2}{\nabla^2 f(\underline{b})} = \begin{bmatrix} -2\sum_{i=1}^{n}(-1) & -2\sum_{i=1}^{n}(-x_i) \\ -2\sum_{i=1}^{n}(-x_i) & -2\sum_{i=1}^{n}x_i^2 \end{bmatrix} = \begin{bmatrix} 2n & 2n\bar{x} \\ 2n\bar{x} & 2\sum x_i^2 \end{bmatrix}$

Note the symmetry $\frac{\partial}{\partial x_i}\left(\frac{\partial f(\underline{x})}{\partial x_j}\right) = \frac{\partial}{\partial x_j}\left(\frac{\partial f(\underline{x})}{\partial x_i}\right) := \frac{\partial^2 f(\underline{x})}{\partial x_i \partial x_j}$

Newton's method for multivariable functions

* given $\underline{x}_i$,
  compute $f(\underline{x}_i)$, $\nabla f(\underline{x}_i)$, $\nabla^2 f(\underline{x}_i)$
* set $\underline{x}_{i+1} = \underline{x}_i - \left(\nabla^2 f(\underline{x}_i)\right)^{-1}\left(\nabla f(\underline{x}_i)\right)$



Variations on Newton's method

Most of them replace or modify the Hessian matrix

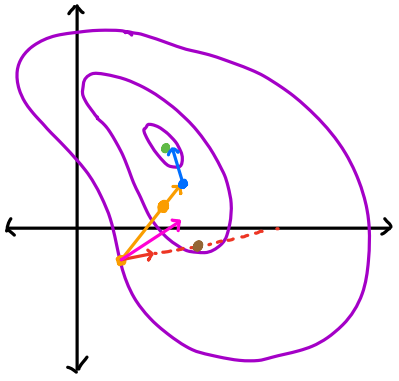
* replace $\nabla^2 f(\underline{x}_i)$ with $\frac{1}{c} I$ $\longrightarrow$ **Gradient descent**

$$\underline{x}_{i+1} = \underline{x}_i - \left(\frac{1}{c} I\right)^{-1} \left(\nabla f(\underline{x}_i)\right) = \underline{x}_i - \nabla f(\underline{x}_i) c$$

* Divide $\nabla^2 f(\underline{x}_i)$ by a constant

$$\underline{x}_{i+1} = \underline{x}_i - \left(\frac{1}{c} \nabla^2 f(\underline{x}_i)\right)^{-1} \left(\nabla f(\underline{x}_i)\right)$$

$$= \underline{x}_i - c \left(\nabla^2 f(\underline{x}_i)\right)^{-1} \left(\nabla f(\underline{x}_i)\right)$$



* Add to diagonal of Hessian

$$\underline{x}_{i+1} = \underline{x}_i - \left(\nabla^2 f(\underline{x}_i) + c I\right)^{-1} \left(\nabla f(\underline{x}_i)\right)$$

* approximate Hessian with gradient information $\longrightarrow$ **BFGS algorithm**

Given $\underline{x}_i$, B_i, set $\underline{x}_{i+1} = \underline{x}_i - B_i^{-1} \left(\nabla f(\underline{x}_i)\right)$

calculate $\nabla f(x_{i+1})$, $y_i = \nabla f(\underline{x}_{i+1}) - \nabla f(\underline{x}_i)$

update $B_{i+1} = B_i + \frac{y_i y_i^T}{y_i^T B_i^{-1} \nabla f(\underline{x}_i)} + \frac{\nabla f(\underline{x}_i) \nabla f(\underline{x}_i)^T}{\nabla f(\underline{x}_i) B_i^{-1} \nabla f(\underline{x}_i)}$

* replace Hessian with Fisher information $\longrightarrow$ **Fisher Scoring**

Fisher information and Fisher scoring

Data: $y_1, \ldots, y_n$

Model: $Y_i \overset{ind}{\sim} p(z; \theta_1, \ldots, \theta_p)$

log likelihood: $\ell(\underline{\theta}) = \sum_{i=1}^{n} \log p(y_i; \underline{\theta})$

gradient: $\nabla \ell(\underline{\theta}) = \sum_{i=1}^{n} \nabla \left(\log p(y_i; \underline{\theta}) \right)$

GRADIENT: $\nabla L(\theta) = \sum_{i=1}^{n} \nabla \left(\log p(Y_i; \underline{\theta}) \right)$

Fisher Information: $I(\theta) = E\left[\left(\nabla L(\underline{\theta}) \nabla L(\underline{\theta})^T \right) \right]$

Fisher Scoring: $\underline{\theta}_{i+1} = \underline{\theta}_i - I(\underline{\theta}_i)^{-1} \left(-\nabla \ell(\underline{\theta}_i) \right)$

Example: Poisson Regression

Data: counts $y_1, \ldots, y_n$

Covariate: $x_1, \ldots, x_n$

Model: $Y_i \sim \text{Poisson}(\exp(b_0 + b_1 x_i))$

$$p(z; \underline{b}) = P(Y_i = z) = \exp(b_0 + b_1 x_i)^z \exp(-\exp(b_0 + b_1 x_i)) / z!$$

$$\log p(y_i; \underline{b}) = y_i (b_0 + b_1 x_i) - \exp(b_0 + b_1 x_i) - \log(y_i!)$$

$$\nabla \log p(y_i; \underline{b}) = \begin{bmatrix} y_i - \exp(b_0 + b_1 x_i) \\ y_i x_i - \exp(b_0 + b_1 x_i) x_i \end{bmatrix}$$

$$\nabla \ell(\underline{b}) = \sum_{i=1}^{n} \nabla \log p(y_i; \underline{b}) = \begin{bmatrix} \sum_{i=1}^{n} y_i - \exp(b_0 + b_1 x_i) \\ \sum_{i=1}^{n} y_i x_i - \exp(b_0 + b_1 x_i) x_i \end{bmatrix}$$

$$\nabla L(\underline{b}) = \begin{bmatrix} \sum_{i=1}^{n} Y_i - \exp(b_0 + b_1 x_i) \\ \sum_{i=1}^{n} Y_i x_i - \exp(b_0 + b_1 x_i) x_i \end{bmatrix}$$

$$\left(\nabla L(\underline{b}) \nabla L(\underline{b})^T\right)_{11} = \left(\sum_{i=1}^{n} Y_i - \exp(b_0 + b_1 x_i)\right)\left(\sum_{j=1}^{n} Y_j - \exp(b_0 + b_1 x_j)\right)$$

$$\left(\nabla L(\underline{b}) \nabla L(\underline{b})^T\right)_{21} = \left(\sum_{i=1}^{n} Y_i - \exp(b_0 + b_1 x_i)\right)\left(\sum_{j=1}^{n} Y_j x_j - \exp(b_0 + b_1 x_j) x_j\right)$$

$$\left(\nabla L(\underline{b}) \nabla L(\underline{b})^T\right)_{12} = \left(\sum_{i=1}^{n} Y_i - \exp(b_0 + b_1 x_i)\right)\left(\sum_{j=1}^{n} Y_j x_j - \exp(b_0 + b_1 x_j) x_j\right)$$

$$\left(\nabla L(\underline{b}) \nabla L(\underline{b})^T\right)_{22} = \left(\sum_{i=1}^{n} Y_i x_i - \exp(b_0 + b_1 x_i) x_i\right)\left(\sum_{j=1}^{n} Y_j x_j - \exp(b_0 + b_1 x_j) x_j\right)$$

$$E(Y_i Y_j) - E(Y_i)\exp(b_0 + b_1 x_j) - E(Y_j)\exp(b_0 + b_1 x_i) + \exp(b_0 + b_1 x_i)\exp(b_0 + b_1 x_j)$$

$$= E(Y_i)E(Y_j) - E(Y_i)\exp(b_0 + b_1 x_j) - \exp(b_0 + b_1 x_i)E(Y_j) + \exp(b_0 + b_1 x_i)\exp(b_0 + b_1 x_j) = 0$$

$$E\left(\nabla L(\underline{b}) \nabla L(\underline{b})^T\right)$$

$$(1,1) = \sum_{i=1}^{n} E\left[Y_i^2 - 2Y_i \exp(b_0 + b_1 x_i) + \exp(2b_0 + 2b_1 x_i)\right] = \sum_{i=1}^{n} \exp(b_0 + b_1 x_i)$$

$$(1,2) = \sum_{i=1}^{n} E\left[Y_i^2 x_i - 2Y_i x_i \exp(b_0 + b_1 x_i) + \exp(2b_0 + 2b_1 x_i) x_i\right] = \sum_{i=1}^{n} x_i \exp(b_0 + b_1 x_i)$$

$$(2,1) = \sum_{i=1}^{n} E\left[Y_i^2 x_i - 2Y_i x_i \exp(b_0 + b_1 x_i) + \exp(2b_0 + 2b_1 x_i) x_i\right] = \sum_{i=1}^{n} x_i \exp(b_0 + b_1 x_i)$$

$$(2,2) = \sum_{i=1}^{n} E\left[Y_i^2 x_i^2 - 2Y_i x_i^2 \exp(b_0 + b_1 x_i) + \exp(2b_0 + 2b_1 x_i) x_i^2\right] = \sum_{i=1}^{n} x_i^2 \exp(b_0 + b_1 x_i)$$

why? $E(Y_i) = \exp(b_0 + b_1 x_i)$

$$E\left(Y_i^2 - 2Y_i \exp(b_0 + b_1 x_i) + \exp(2b_0 + 2b_1 x_i)\right)$$

$$= E(Y_i^2) - 2\exp(2b_0 + 2b_1 x_i) + \exp(2b_0 + 2b_1 x_i)$$

$$= E(Y_i^2) - \exp(2b_0 + 2b_1 x_i)$$

$$= E(Y_i^2) - E(Y_i)^2$$

$$= \mathrm{Var}(Y_i) = E(Y_i) = \exp(b_0 + b_1 x_i)$$

↑ property of Poisson.