Lecture 7: 2/18/17

Unconstrained Optimization

Given function $J(\vec{\theta})$, $J: \mathbb{R}^M \to \mathbb{R}$

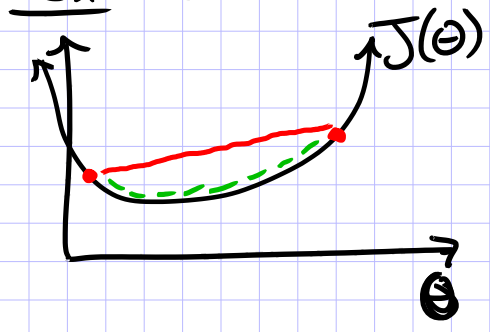
Goal is $\hat{\vec{\theta}} = \underset{\vec{\theta}}{\operatorname{argmin}} J(\vec{\theta})$ — by convention

$= \underset{\vec{\theta}}{\operatorname{argmax}} -J(\vec{\theta})$

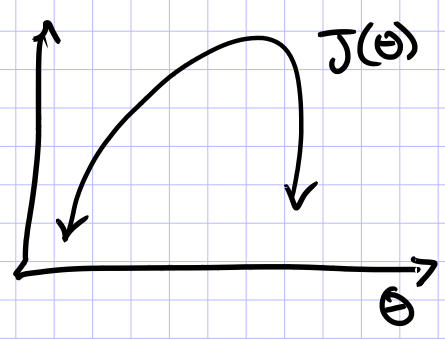
For ML: J is usually an objective fn.
$\vec{\theta}$ is vector of parameters

Convexity

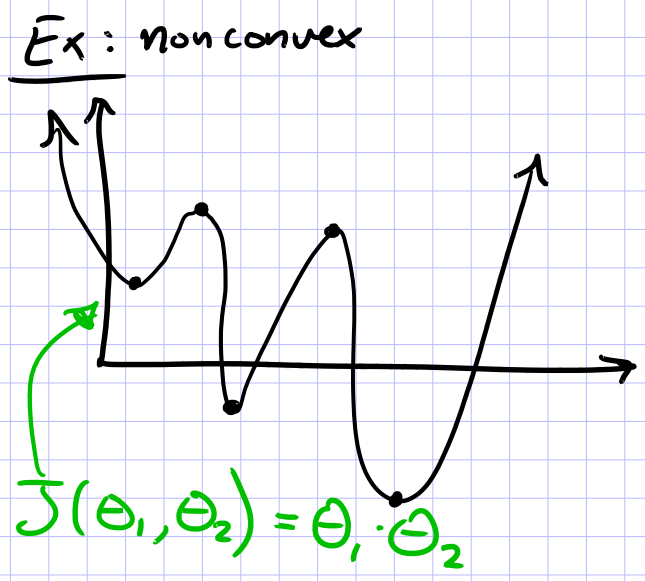
Ex: convex



Ex: concave



Ex: nonconvex



Ex: $J(\theta_1, \theta_2) = \theta_1 \cdot \theta_2$

Def: The gradient of J

$$\nabla J(\vec{\theta}) \triangleq \begin{bmatrix} \frac{dJ(\vec{\theta})}{d\theta_1} \\ \frac{dJ(\vec{\theta})}{d\theta_2} \\ \vdots \\ \frac{dJ(\vec{\theta})}{d\theta_M} \end{bmatrix}$$

first order partial derivatives

Def: The **Hessian matrix**: second-order partial derivs

$$\nabla^2 J(\vec{\theta}) \triangleq H_J(\vec{\theta}) \triangleq \begin{bmatrix} \frac{d^2J(\vec{\theta})}{d\theta_1^2} & \cdots & \frac{d^2J(\vec{\theta})}{d\theta_M d\theta_1} \\ \vdots & & \vdots \\ \frac{d^2J(\vec{\theta})}{d\theta_1 d\theta_M} & \cdots & \frac{d^2J(\vec{\theta})}{d\theta_M^2} \end{bmatrix}$$

$$H_{ij} = \frac{d^2J(\vec{\theta})}{d\theta_i d\theta_j}$$

Def: $J'(\theta) \triangleq \frac{dJ(\theta)}{d\theta}$

$J''(\theta) \triangleq \frac{d^2J(\theta)}{d\theta^2}$

Closed-Form Opt.

Ex #1: 1D function $J(\theta)$

① Solve $J'(\theta) = 0$ for a point $\hat{\theta}$

② Q: Is $\hat{\theta}$ a min, max, or saddle point?

A: If $J''(\theta) > 0 \Rightarrow$ min

If $J''(\theta) < 0 \Rightarrow$ max

If $J''(\theta) = 0 \Rightarrow$ use "extremum test" (higher-order pdv)

← second derivative test.

Ex #2: M-dim function $J(\vec{\theta})$, $\vec{\theta} \in \mathbb{R}^M$

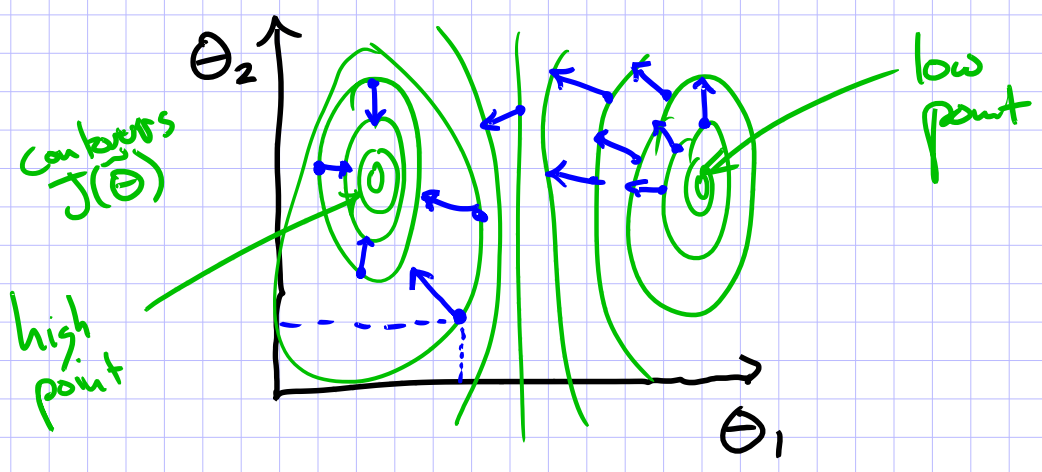
test for min, max, s.p. requires Hessian

Gradient Descent

Motiv: - What if we can't solve $\nabla J(\vec{\theta}) = \vec{0}$ analytically? (e.g. Logistic Regression)

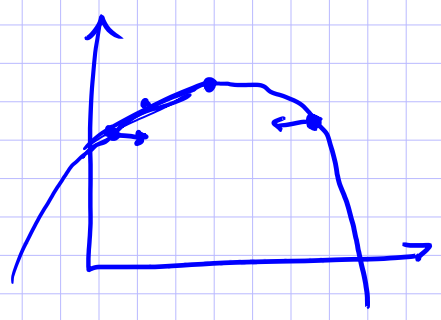
- What if we can, but it's inefficient to compute. (e.g. Linear Reg.)

Ex: Gradients in 2D



$\nabla J(\vec{\theta}) = \begin{bmatrix} \vdots \end{bmatrix}$

blue arrows indicate the direction of the gradient, but gradients also have magnitude



$\nabla J(\theta) = [\ \ ]$

Algorithm:

1) Choose an initial **starting point** $\vec{\theta}$
2) Repeat
   a) Compute the gradient $\nabla J(\vec{\theta})$
   b) Choose a **step size** size γ
   c) Update $\vec{\theta} = \vec{\theta} - \gamma \nabla J(\vec{\theta})$
3) Return $\vec{\theta}$ when **stopping criterion** satisfied.

Starting Point:

Two good options for ML:

1. $\vec{\theta} = 0$
2. $\vec{\theta}$ randomly

Stopping Criterion

Typically

$\| \nabla J(\vec{\theta}) \|_2 < \epsilon$

$\|\vec{v}\|_2 \triangleq \sqrt{\sum_{i=1}^{|v|} v_i^2}$

Step Size:

Lots of Options

- ~~fixed value eg. $\gamma = 0.1$~~
- Exact line search
- Backtracking line search

Stochastic Gradient Descent

Assume: $J(\vec{\theta}) \triangleq \frac{1}{N}\sum_{i=1}^{N} J_i(\vec{\theta})$ where $J_i : \mathbb{R}^M \to \mathbb{R}\ \forall i$

Ex: $p(D|\theta) = \prod_{i=1}^{N} p(x^{(i)}, y^{(i)}|\theta)$

$$\ell(\theta) = \log p(D|\theta) = \sum_{i=1}^{N} \underbrace{\log p(x^{(i)}, y^{(i)}|\theta)}_{\ell_i(\theta)}$$

neg. log. like $\longrightarrow$ $J(\theta) = -\frac{1}{N}\ell(\theta) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{-\ell_i(\theta)}_{J_i(\theta)}$

Expectations of Gradients

$$\frac{dJ(\vec{\theta})}{d\theta_j} = \frac{1}{N}\sum_{i=1}^{N} \frac{d}{d\theta_j}\left(J_i(\vec{\theta})\right)$$

$$\nabla J(\vec{\theta}) = \begin{bmatrix} \vdots \\ \cdots \\ \vdots \end{bmatrix} \leftarrow j^{th} = \frac{1}{N}\sum_{i=1}^{N} \nabla J_i(\vec{\theta})$$

Recall: for any discrete r.v. X $\mathbb{E}_X[f(x)] \triangleq \sum_x P(X=x) f(x)$

Q: What is the expected value of a randomly chosen $\nabla J_i(\theta)$?

Let $I \sim \text{Uniform}(\{1, \ldots, N\})$

$\Rightarrow P(I=i) = \frac{1}{N}$ if $i \in \{1 \ldots N\}$

$$\mathbb{E}_I[\nabla J_I(\vec{\theta})] = \sum_{i=1}^{N} P(I=i)\nabla J_i(\vec{\theta})$$

$$= \frac{1}{N}\sum_{i=1}^{N} \nabla J_i(\vec{\theta})$$

$$= \nabla J(\vec{\theta})$$

SGD Algo:

1) Choose starting point $\vec{\theta}$
2) For t in $1 \ldots T$
   a) sample $i \sim Unif(\{1, \ldots, N\})$
   b) compute $\vec{g} = \nabla J_i(\vec{\theta})$ ← stochastic approx. to true gradient $\nabla J(\vec{\theta})$
   c) choose step size $\gamma^{(t)}$
   d) update $\vec{\theta} = \vec{\theta} - \gamma^{(t)} \vec{g}$
3) Return $\vec{\theta}$ when stopping criterion satisfied

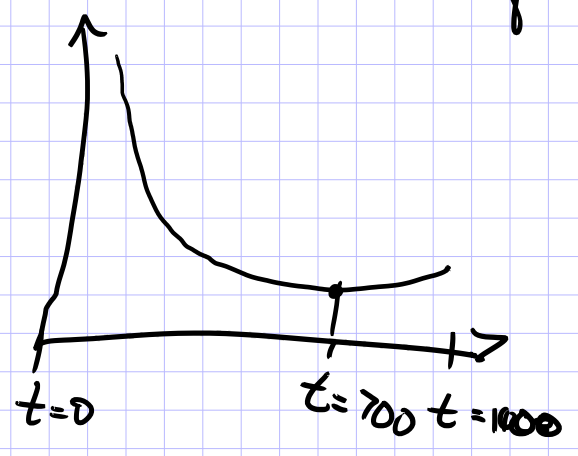
Step Size:

- larger values early on
  smaller values closer to opt.
- heuristic:
  ① pick an initial $\gamma = 0.1$
  ② run until no improvement
  ③ $\gamma = \gamma/2$
- learning rate schedule

$$\gamma^{(t)} = \frac{\gamma^{(0)}}{(t-1)\gamma^{(0)} + 1}$$

Stopping Criteria:

For ML:
plot validation error
and choose lowest point



Convergence Analysis:

Def: convergence is when $J(\vec{\theta}) - J(\vec{\theta}^*) < \epsilon$ ← true unknown min

Methods	Steps to Converge	Computation per iteration
Newton's Method	$O(\ln \ln 1/\epsilon)$	$\nabla J(\theta)$ $\nabla^2 J(\theta)$ ← $O(NM^2)$
GD	$O(\ln 1/\epsilon)$	$\nabla J(\vec{\theta})$ ← $O(NM)$
SGD	$O(1/\epsilon)$	$\nabla J_i(\theta)$ ← $O(M)$

not correct

"almost sure" convergence

lots of caveats and conditions

very less complex

Takeaway: SGD has much slower asymptotic convergence.
but is often faster in practice.

Regression Problem

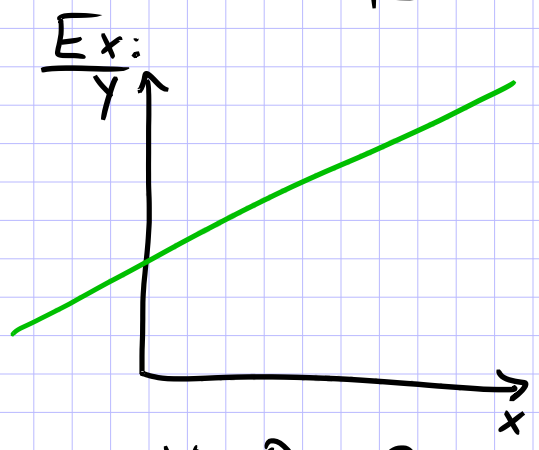
$$D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

$\vec{x} \in \mathbb{R}^M$ ← input, features

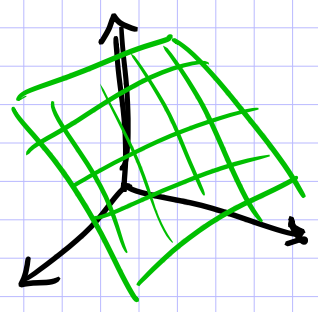
$y \in \mathbb{R}$ ← output

Linear Function

Ex:

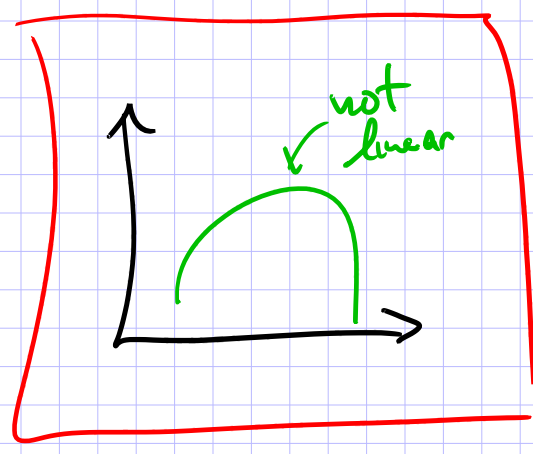


$$y = \theta x + \beta$$



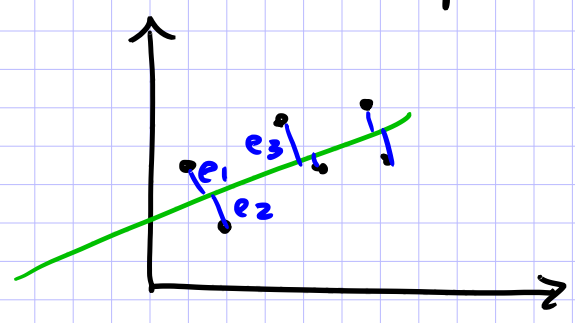
$$y = \vec{\theta}^T \vec{x} + \beta$$

$\vec{\theta} \in \mathbb{R}^M$, $\beta \in \mathbb{R}$



Def: residuals: distance from observed value to predicted value

$$e_i \triangleq y^{(i)} - (\vec{\theta}^T \vec{x} + \beta)$$



Notation Trick:

Define $x_0 \triangleq 1$

$$y = \vec{\theta}^T \vec{x} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_M \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_M \end{bmatrix}$$

θ_0: replacement for β

x_0: always 1

Linear Regression as Functn Approx.

Story:

① Assume D generated: $x^{(i)} \sim p^*(x)$, $y^{(i)} = h^*(x^{(i)})$ — unkn.

② Choose hypothesis space, H

$$H = \{h(\vec{x}) = \vec{\theta}^T \vec{x} : \vec{\theta} \in \mathbb{R}^M\}$$

↑ space of all linear fn. in M-dim.

③ Choose an objective fn.

Minimize mean squared error (MSE)

$$J(\vec{\theta}) = \frac{1}{N} \sum_{i=1}^{N} e_i^2 \qquad (h^*(x^{(i)}) - h(x^{(i)}))$$

$$= \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \vec{\theta}^T \vec{x}^{(i)})^2$$

④ Solve unconstrained opt. problem via favorite method:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\vec{\theta})$$

- closed-form ← "Normal Eq.s"
- SGD ← "Least Mean Squares"
- GD
- Newton
- ⋮

⑤ Predict: $\hat{y} = h(\vec{x}) = \hat{\theta}^T \vec{x}$