بسم الله الرحمن الرحيم

يلا علي الريجريشن .. اللي هنعمول انهارده .. اننا هنفكر تاني في اللينير ريجريشن .. وديه خطوه جميله الحقيقه لإنا هنفكر فجه كإننا بنحل أوبتيمايزيشن بروبليم .. اللي هنعملو انهارده اننا هنقعد نترنح ما بين اننا ندور علي طريقه لل optimization و بعدين نشوف ازاي نطبقها علي ال linear regression .. وهنعمل الترنّح ده حوالي 3 او 4 مرات هنفضل نترنّح كدا لحد ما المحاضره تخلص .. طب افرض المحاضره مخلصتش .. هتفضل تترنّح برضو .. يلا بينا .. هومورك 3 نزل يا حلاوه .. هه

آخر مره اتكلمنا علي التعريف بتاع ال linear function و ازاي ال linear regression بيقلل ال residuals ما بين البريديشكن و الأوبزرفيشن .. تعال بس الاول نقول كام حاجه .. الاوبتيمازيشن ليه سيتتنج معين .. ان الفانكشن اللي انت عاوز تعملها اوبتيمايزيشن مش شرط تبقا هي اللي احنا مهتمين بيه ... زي مثلاً اللايكلي هوود فانكشن بتاعت اللوجستيك ريجريشن .. بس الي انت مهتم بيه اصلا هو الايرور بتاع ال test data عشان الجينيرالايزيشن .. .تاني حاجه مهمه ف الستتج بتاع الأوبتيمايزيشن هو ان الداتا اصلا ممكن تبقا نويزي و في الحاله ديه انك تجيب optimal percentage مش هتبقا اهم حاجه يعني .. بس في شوية حاجات تانيه ممكن تخلي ال precision مهمه يعني .. تالت حاجه هو ال early stopping وده مهم جداً الحقيقه عشان بيساعد ال generalization ..

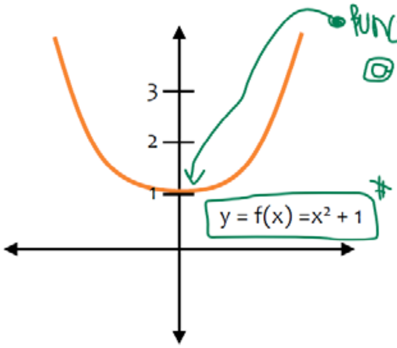
Optimization for ML

Not quite the same setting as other fields…

- Function we are optimizing might not be the true goal
  (e.g. likelihood vs generalization error)
- Precision might not matter
  (e.g. data is noisy, so optimal up to 1e-16 might not help)
- Stopping early can help generalization error
  (i.e. "early stopping" is a technique for regularization – discussed more next time)

الدكتور بيقول لما بيتكتب ال argmin .. هل انت بتبقا فاهم ايه المقصود منها .. طبعاً انا حمار فبفترض انه عاوز الحاجه اللي تخلي الحاجه اقل حاجه .. هه .. بطيخه .. فالدكتور هيقول ايه اللي بيبقا مقصود

min vs. argmin



function's value @ its minimum

min of all values of x over f(x)

$v^* = \min_x f(x)$

$x^* = \text{argmin}_x f(x)$

value of x that gives rise to that min.

1. Q: What is v*?

   v* = 1, the minimum value of the function

2. Q: What is x*?

   x* = 0, the argument that yields the minimum value

فال agmin x هي قيمة الإكس اللي إدت قيمة الفانكشن المنيمم ... احنا هنستخدم الحاجات ديه عشان نقدر نفكّر في اللينير ريجريشن كفانكشن أبروكسميشن .. هنا الداتا هتبقي شوية x and y pairs و ال x هتبقا فيكتور طوله M و الواي هتبقا رقم حقيقي .. احنا هنا هنفترض ان الداتا D كانت Generated باستخدام البروسيس ديه: البروسيس بتقول ان إكس كانت sampled from some probability distribution p star .. واحنا مش عارفين ايه هي ال P star اصلا هي unknown prob. Distribution .. وبعدين هنجيب ال y عن طريق اننا نباصي الإكسات لل unknown function h star .. فهنا الفرضيه علي إن ال p star و ال h star مش عارفينهم ... قدام شوية هنعدل علي الفرضيات ديه و هنحط كمان شوية نويز .. يعني المصدر بتاع ال y مش هيبقا deterministic ....

تعال نختار hypothesis space .. وهنا ال hypothesis space هيبقا هو ال all linear functions in M dimensional space .. فده هيبقا زي ال perceptron hypothesis space ماعدا اننا معندناش sign حوالين ال theta transpoce x .. وبعدين هنختار an objective function هتبقا ال mean squared error وده اللي اتكلمنا عليه المره اللي فاتت اننا هنقل ال Sum of residuals .. احنا هناخد كل residual و نربعو وبعدين نجمعهم كلهم .. فين ال w و ال b .. في الثيتا .. فهنا انت تقدر تفترض ان أأول فيتشر في الإكس هي دايماً بواحد .. و ده هيبقا ال bias بتاعك ..

Linear Regression as Function Approximation

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

unknown

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)}) + \epsilon$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} e_i^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \underbrace{\boldsymbol{\theta}^T\mathbf{x}^{(i)}}\right)^2$$

4. Solve the unconstrained optimization problem via favorite method:
   - *gradient descent*
   - *closed form*
   - *stochastic gradient descent*
   - ...

look for value of vector θ that minimizes that function J(θ) → θ

$$\boxed{\hat{\boldsymbol{\theta}}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \boxed{J(\boldsymbol{\theta})}$$

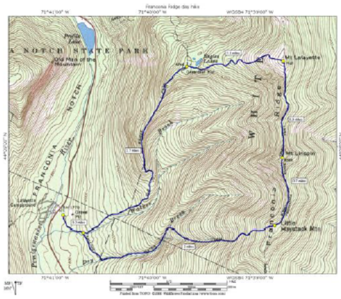
5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \underbrace{\hat{\boldsymbol{\theta}}^T}\underbrace{\mathbf{x}}$$

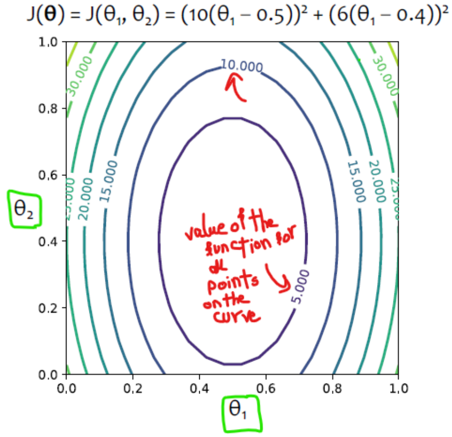
10

Contour Plots

Contour Plots
1. Each level curve labeled with value
2. Value label indicates the value of the function for all points lying on that level curve
3. Just like a topographical map, but for a function

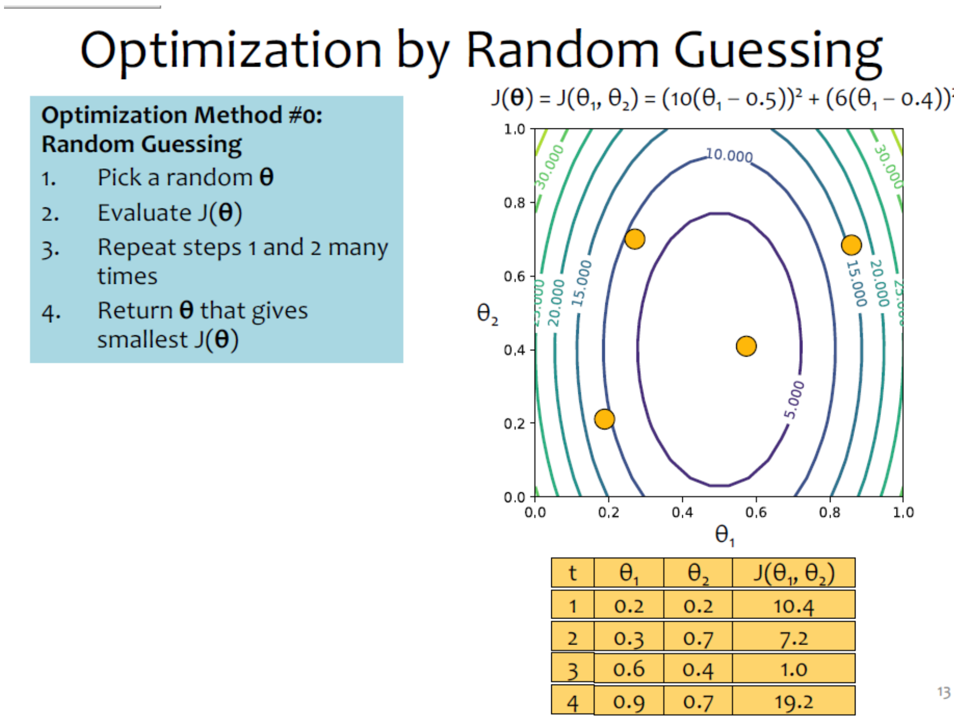


$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = (10(\theta_1 - 0.5))^2 + (6(\theta_1 - 0.4))^2$



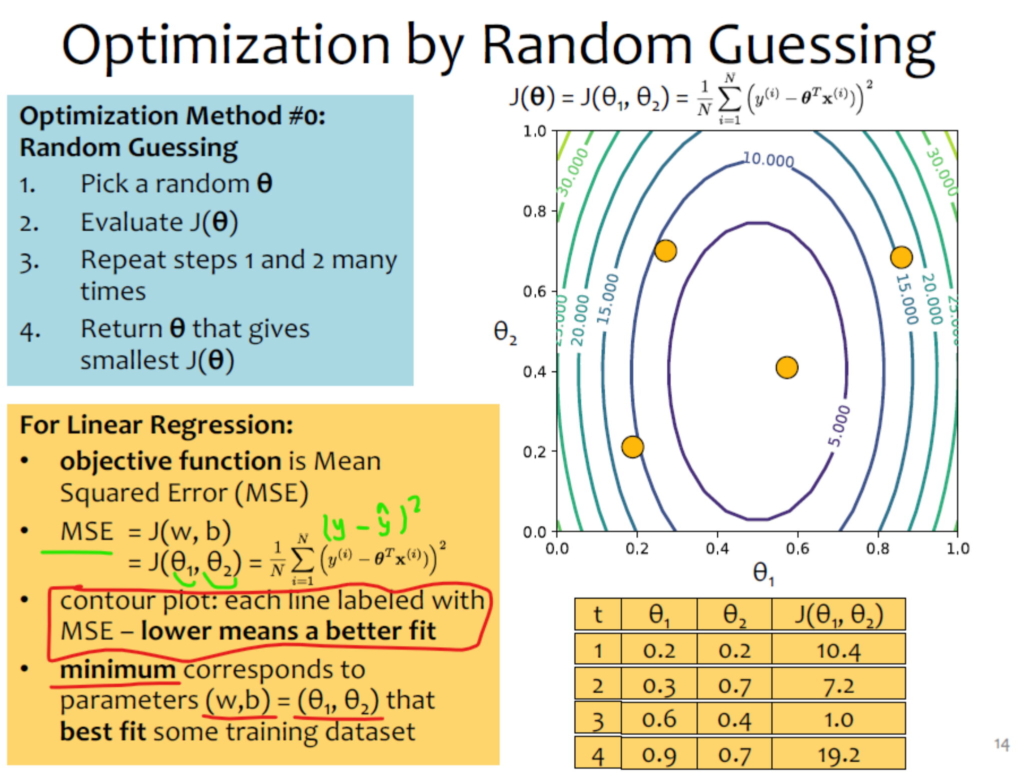
12

هنا بقي أول أوبتيمايزيشن ألجورذم .. اسمو random guessing ... لو هنيجي نختار البارمترز بتاعت الموديل بتاعنا بناءً علي الألجورذم ده .. تعال نشوف ايه اللي هيحصل ..

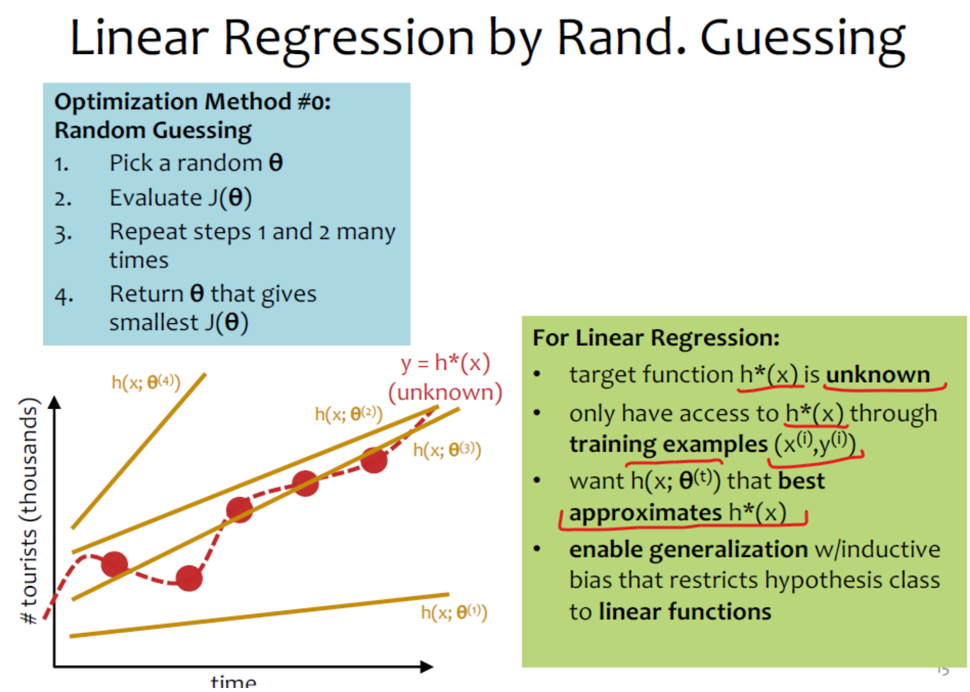


لو جيت تبص علي اللي بيحصل هتلاقي انك بتجيب أرقام راندم ما بين الثيتا 1 و 2 ... وبعدين تقوم معوض وتجيب الناتج بتاع ال J(theta1, theta2) ..... هتلاقي ان الارقام بتاعت ال J كبرت فجأه .. فهتسأل نفسك سؤال .. هو ليه بيعمل كدا .. ما الكونتور عندو اهو ما يروح عليه و يجيب ال minimum .. الدكتور اعترض ع النقطه ديه وقال طب لو انت عندك 10000 ثيتا.. هتبص ع الكونتور ازاي ... وهتعمل الأوبتيمايزيشن ازاي ..

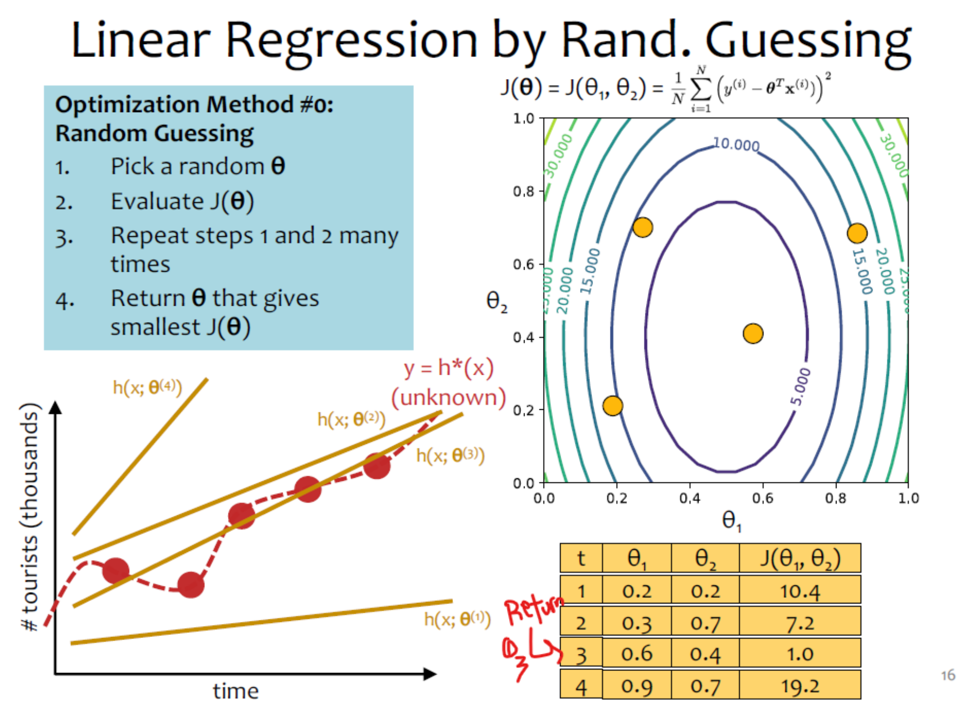
تعال نبص علي الحوار ده من سكوب ال linear regression



طب تعال نرن ال random guessing علي ال linear regression .. فهنحتاج لكل hypothesis حاجتين slope and intersect ..



الخطوط ديه هي الراندم guessing اللي حصل .. تعال نحط الصورتين مع بعض .. اللي هو الجراف والكنتور مع بعض ..



طيب تعال نفكّر في ال optimization method #1 .. عشان نعمل كدا هنحتاج شوية ديفينيشنز .. أول واحد هو unconstrained optimization .. فلما بنتكلم علي unconstrained optimization .. انت بتقول ان عندك فانكشن اسمها J(theta) و ال J ديه بت map من M dimensional space ل رقم واحد بس .. وهدفنا اننا نلاقي ال Theta hat اللي هي ال argmin of all possible theta of J(theta) .. للماشين ليرننج .. ال J هنا هي أوبجكتيف فانكشن ... والثيتا هتبقا البارمترز بتاعتك .. لو هنيجي نشغل ال gradient descent .. لازم نفهم ايه هو ال gradient ... فدلوقت هنعوز نفهم يعني ايه ال deeravtives .. هنفتكر كدا بس ايه هو ال interpretations بتاعت ال derivatives.. بص ع الصوره ..

Unconstrained Optimization

Given Fn. $J(\vec{\theta})$ $\quad J: \mathbb{R}^M \to \mathbb{R}$

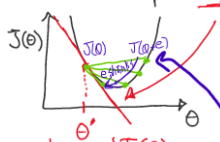
Goal is $\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, J(\theta)$

params

for ML J's are obj. fn.

Derivatives

① Deriv. as Slope of Tangent



slope $= \dfrac{\partial J(\theta)}{\partial \theta}$

③ Deriv as Tangent Plane

$\left\langle \dfrac{\partial J(\theta_1,\theta_2)}{\partial \theta_1}, \dfrac{\partial J(\theta_1,\theta_2)}{\partial \theta_2} \right\rangle$

② Deriv as a Limit

$$\frac{\partial J(\theta)}{\partial \theta} = \boxed{\lim_{e \to 0}} \frac{J(\theta+e) - J(\theta)}{e}$$

limit of secants is tangent

e shrinks → secant line approaches the tangent line

Gradient

The gradient of J

$$\nabla J(\vec{\theta}) = \begin{bmatrix} \dfrac{\partial J(\vec{\theta})}{\partial \theta_1} \\ \dfrac{\partial J(\vec{\theta})}{\partial \theta_2} \\ \vdots \\ \dfrac{\partial J(\vec{\theta})}{\partial \theta_M} \end{bmatrix}$$

first order partial derivatives

Gradient Descent

Algorithm:

① Choose an initial point $\vec{\theta}$

② Repeat for $t = 1, 2, 3, \ldots$

a) Compute gradient $\vec{g} = \nabla J(\vec{\theta})$

b) Choose a step size $\gamma_t > 0$

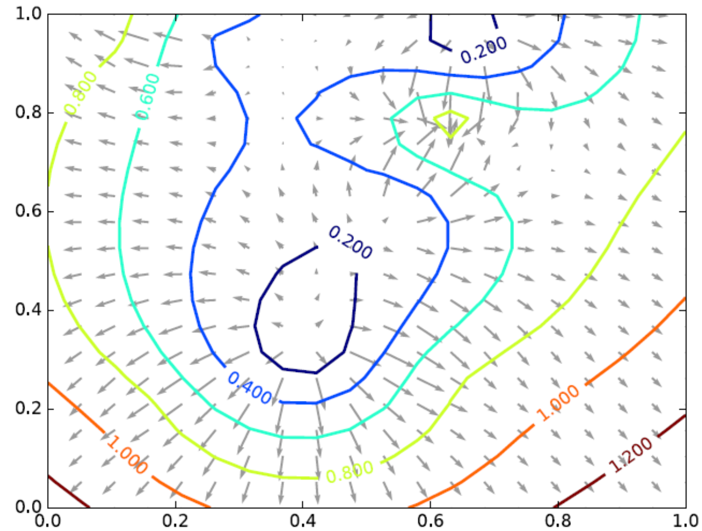
c) Update $\vec{\theta} \leftarrow \vec{\theta} - \gamma \vec{g}$

③ Return $\vec{\theta}$ when stopping criterion is met

الدكتور شرح مثال عليه هو و مراته ... الحقيقه المثال حلو فاسمعو من المحاضره مش هكتبو . المهم انو في الاخر بيتارجت حته انك عارف طول منت بتنزل من ع الجبل انت عارف انك مسيرك تلاقي العربيه بتاعتك .. الألجورذم ده هو ال gradient descent .. تعال نشوف الكونتور بلوت بتاع الفانكشن

انت هتحتاج شوية معلومات من الجريدينت .. عشان انت لما هتيجي تevaluate ال gradient علي نقطه معينه في الفانكشن ... هيديك فيكتور دايماً بيشاور ناحية ال steepest ascend اللي هو الاتجاه اللي لو انت مشيتو .. فهتزود قيمة الفانكشن .. فبالتالي الرسمه اللي تحت هي grid of gradients ولكل gradient عندك فيكتور بتاع التانجنت بليين ..

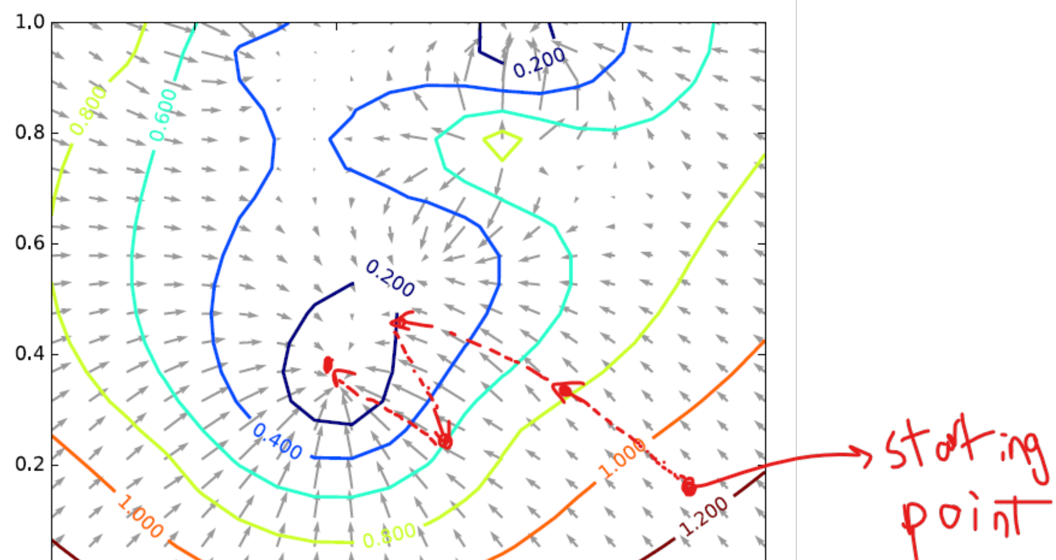
Gradients



These are the **gradients** that
Gradient **Ascent** would follow.

لو جينا نبص علي النيجاتيف جريدينتس .. هيشاورو ناحية ال steepest descent

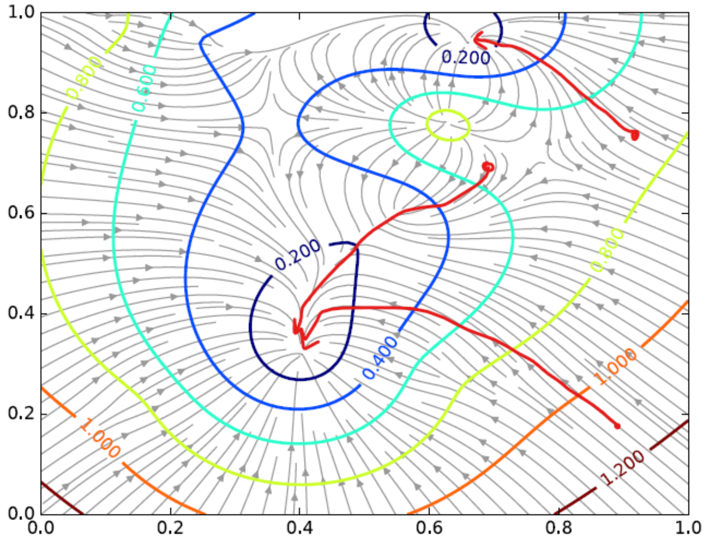
(Negative) Gradients



These are the **negative** gradients that
Gradient **Descent** would follow.

لو جينا نبص علي الطريق اللي هنمشي عليه من أي starting point ..

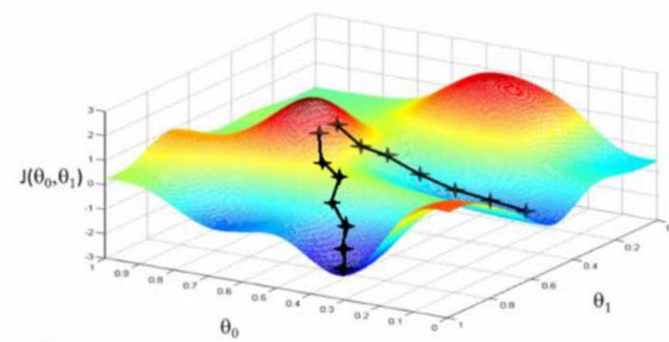
(Negative) Gradient *Paths*



Shown are the **paths** that Gradient Descent would follow if it were making **infinitesimally small steps.**

Pros and cons of gradient descent

- Simple and often quite effective on ML tasks
- Often very scalable
- Only applies to smooth functions (differentiable)
- Might find a local minimum, rather than a global one



e courtesy of William Cohen

25

تعال بقي نشوف الألجورذم ده هيشتغل ازاي .. او هيبقا شكلو عاملازاي .. ال gradient descent ك ألجورذم . هيمشي كالآتي :

1. اختار نقطة البداية بتاعتك .. اللي هي ال Theta vector
2. Repeat the following steps:
   a. Compute gradient g = gradient of J(theta)
   b. Choose step size, which is some value Gamma > 0 and it's a real number. It says how much we extend or shrink that gradient vector to decide how far to step
   c. Update Our paramtere Theta <- Theta – Gamma * g
3. Return theta when Some stopping criterion is met

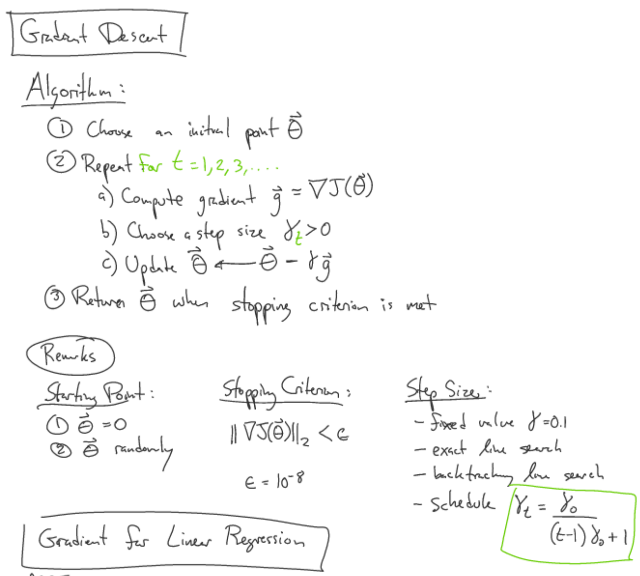
وده يشباب هو الجريدينت ديسينت .. في شوية أسئله .. ازاي نختار ال starting point .. أول طريقه هي اللي عملناها لل perceptron .. اللي هو حط كلو اصفار .. تاني طريقه اختارهم random ..

طب بالنسبه لل stopping criterion .. انت محتاج ت stop taking steps .. أياً كان ال steps اللي هتاخدها هي هتقل .. ليه هتقل .. عشان لو جيت تبص ع الصوره هتلاقي ان الجريدينتس اللي عند الحتت اللي فعلاً steep .. بتلاقي الماجنتيود بتاعها كبير .. ماجنتيود جامد .. بس عند المنيمم الفيكتور بيبقا رفيع أوي وصغير كدا ... ليه .. لإن الفيكتور بيربرزنت ان السلوب بتاع البلين هنا هيبقا بصفر .. وفعلاً عند الإكزاكت منيمم .. هو فعلا هيبقا سلوب صفر .. فالستوبنج كرايترين هتبقا ان

Whenever the gradient of J(theta) and taken its L2 norm is < some epilon (small number i.e. 10^- 3), then in this case the length of that gradient vector must be really short and so it's almost 0 and we are probably close to the minimum

آخر حاجه هو ال step size ... اول اختيار هو الفيكسيد فاليو .. جاما .. Gamma = 0.1 .. وفي أوبشن تاني هو ال exact line search .. وفي اوبشن تالت اللي هو ال backtracking line search .. فكرة اللاين سيرش هو .. لو افترضنا ان عندنا فانكشن و المنيمم بتاعها هو في نص الأوضه اللي احنا فيها .. والجرادينت بيشاور بشكل مباشر ناحية المنيمم ده .. هيبقا حاجه جامده جداً لو عرفت انت محتاج كام ستيب .. بس انت مش هتعرف كام ستيب .. فهتاخد ستيب واحده ... وبعدين تشوف هل انا احسن .. وبعدين تقول انا اخد 2 ستيبس .. هل انا احسن .. هتقول اه فعلاً انا كدا احسن .. وبعدين تاخد 4 ستيبس وتشوف هل كدا احسن .. تقول انا ايه اللي جابني ف غريبه احنا ف عجيبه .. ارجع خطوتين لورا "Backtrack" .. هل انت احسن .. اه .. خد خطوه قدام .. هل انت احسن .. اه وتقريباً كدا انا وصلت عجيبه ..هه ..

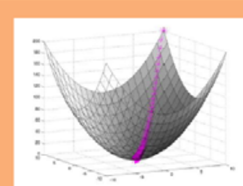
في تكنيك رابع وهو انك تستخدم schedule Gamma t = Gamma 0 / ((t-1)*Gamma0 + 1)



Gradient Descent

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$ **while** not converged **do**
4: $\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$ **return** $\boldsymbol{\theta}$



In order to apply GD to Linear Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

27

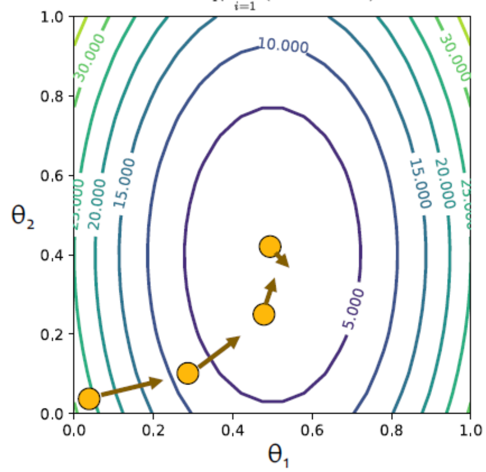
تعال نشوف بقا ايه هو ال gradient في ال linear regression ... لما كنا بنبص من فوق كدا . احنا كنا عاوزين ناخد ال mean squared error function و نعملها minimization فاحنا هنعوز نحسبلها ال gradient بتاع ال mean squared error .. وده هيدينا طريقه اننا نحسب ال gradient descent ..

Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random **θ**
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return **θ** that gives smallest $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$



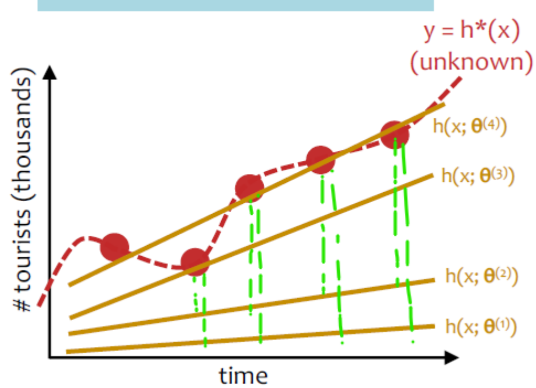
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

المهم تعال نرجع للمثال بتاعنا اللي هو ال linear function اللي بنحاول ن fit on the data .. الإكس أكسيس هو التايم و عدد السياح علي الواي أكسيس و عندك التروو فانكشن اللي انت مش عارفها .. و هنعوز نبص علي السلوب والإنترسيبت ... هنفضل ناخد خطوات .. هتلاحظ ان ال Mean squared error بقا احسن .. اللي هو قيمته قلت .. ليه .. عشان المسافات ما بين التروو و البريدكشن قلت

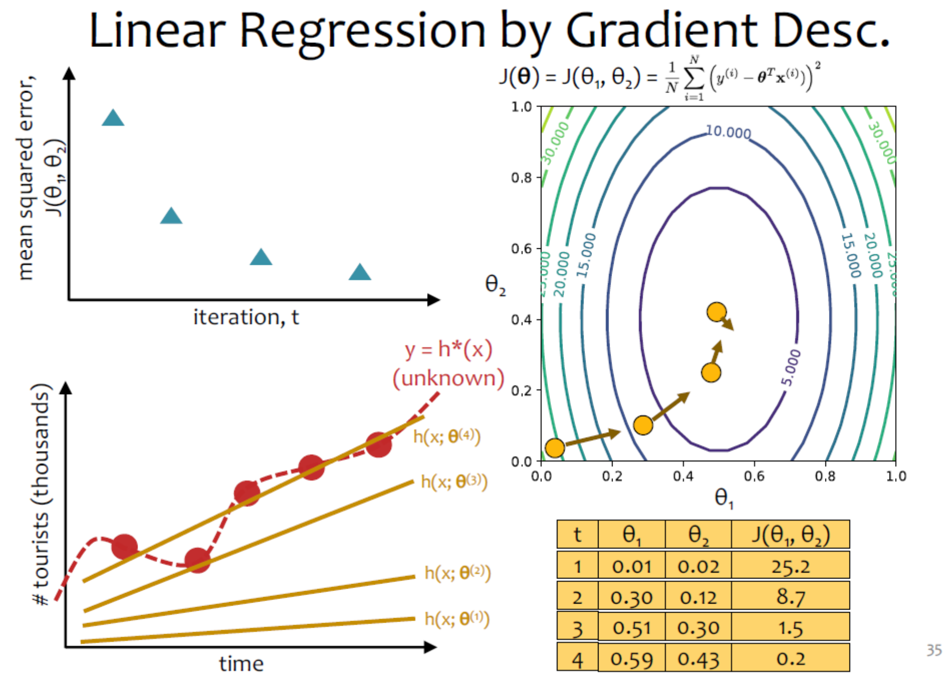
Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random **θ**
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return **θ** that gives smallest $J(\boldsymbol{\theta})$



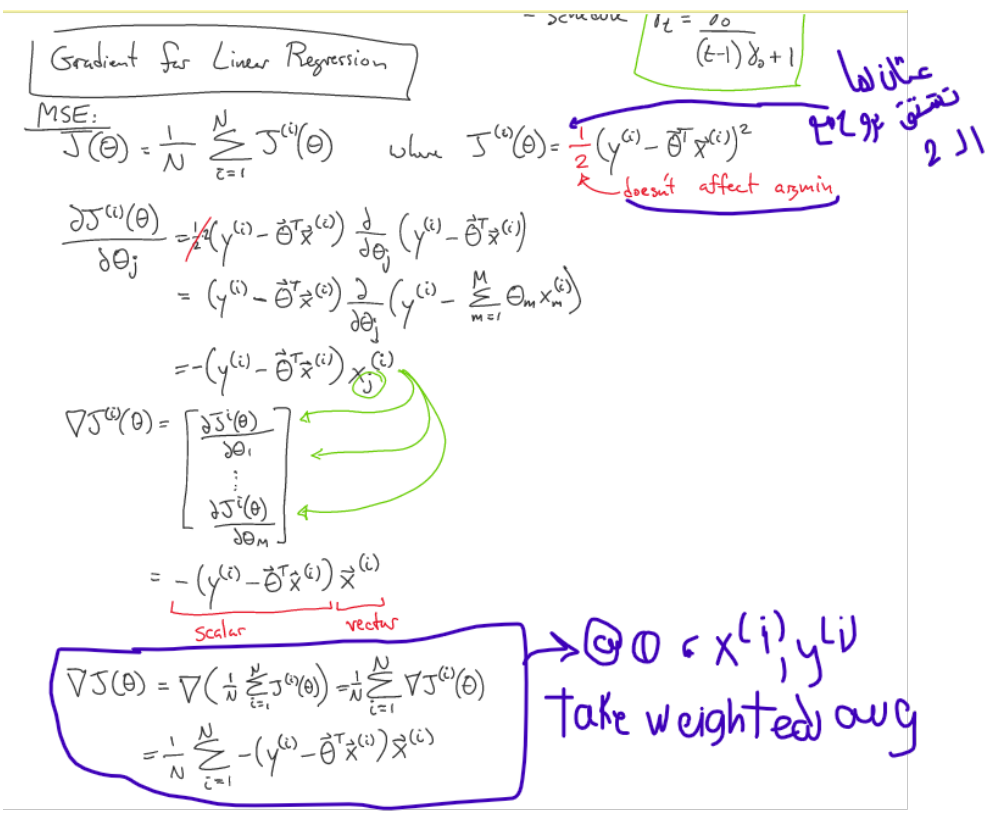
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2



t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

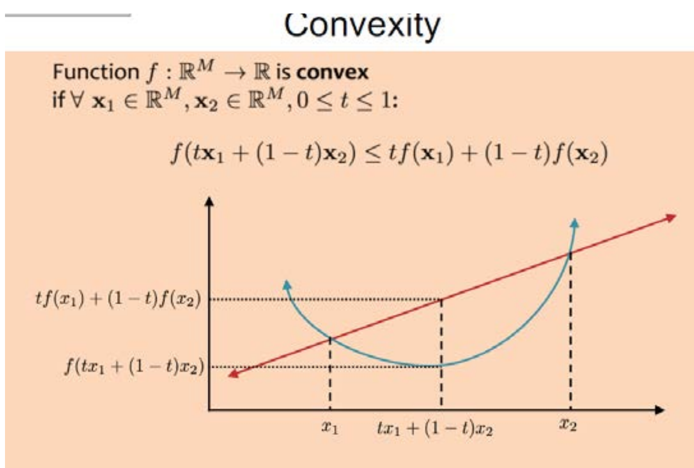
احنا اللي احنا محتاجينو دلوقت هو ال actual gradient for linear regression.. هنشوف دلوقت عظمه في الدرفيجن لل gradient for linear regression

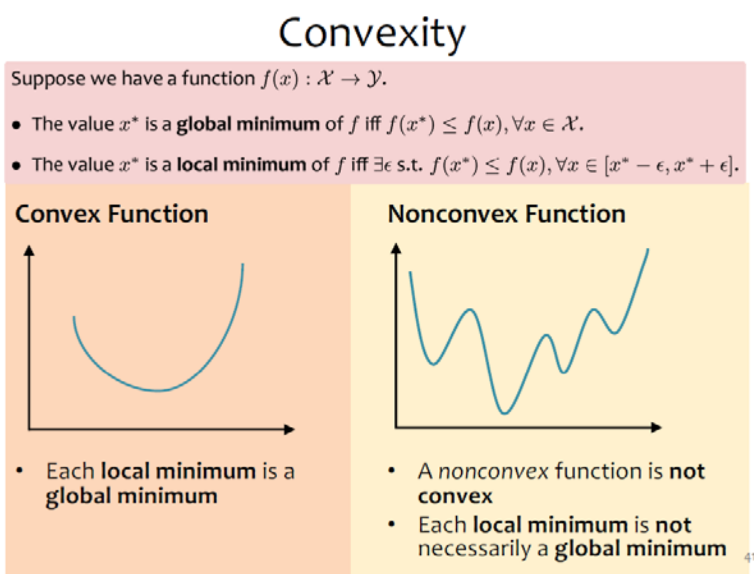
هنبدأ بإن عندنا معادلة ال mean squared errors ..

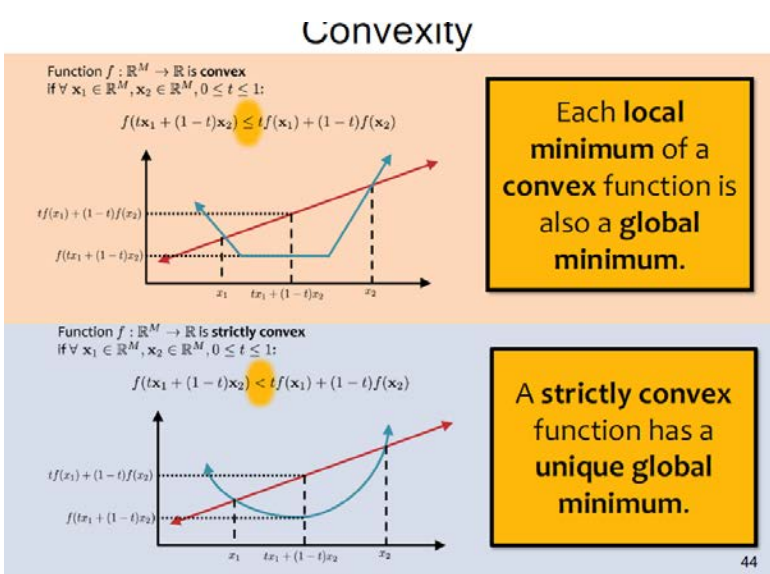


طيب دلوقت هنتكلم علي ال convexity ... احنا نقول فانكشن تبقا كونفيكس لو لكل الpairs of points x1 and x2 .. و لكل قيم ال t ما بين الصفر و الواحد .. انت تقدر تاخد funcation of the average of the 2 points و ده يبقا اقل من ال average function evaluated at those 2 points .. افترض ان عندك نقطة x1 و نقطه x2 .. وانت عاوز تتأكد إن لأي قيمه لل t ما بين الصفر و الواحد .. هيديك نقطه علي ال red sequin اللي بيوصل ما بين الإكس 1 و 2 ... اللي هو الخط الأحمر ... و convexity بتقول إن أياً كان القيمه اللي علي الخط الأحمر ده .. هتبقا أكبر من القيمه الحقيقه اللي هتجيبها اللي هي tx1 + (1-t)x2 .. وده لكل ال possible pairs لل x1 and x2 .. فلو انت عندك convex function ديه ليها شوية خصايص .. بس

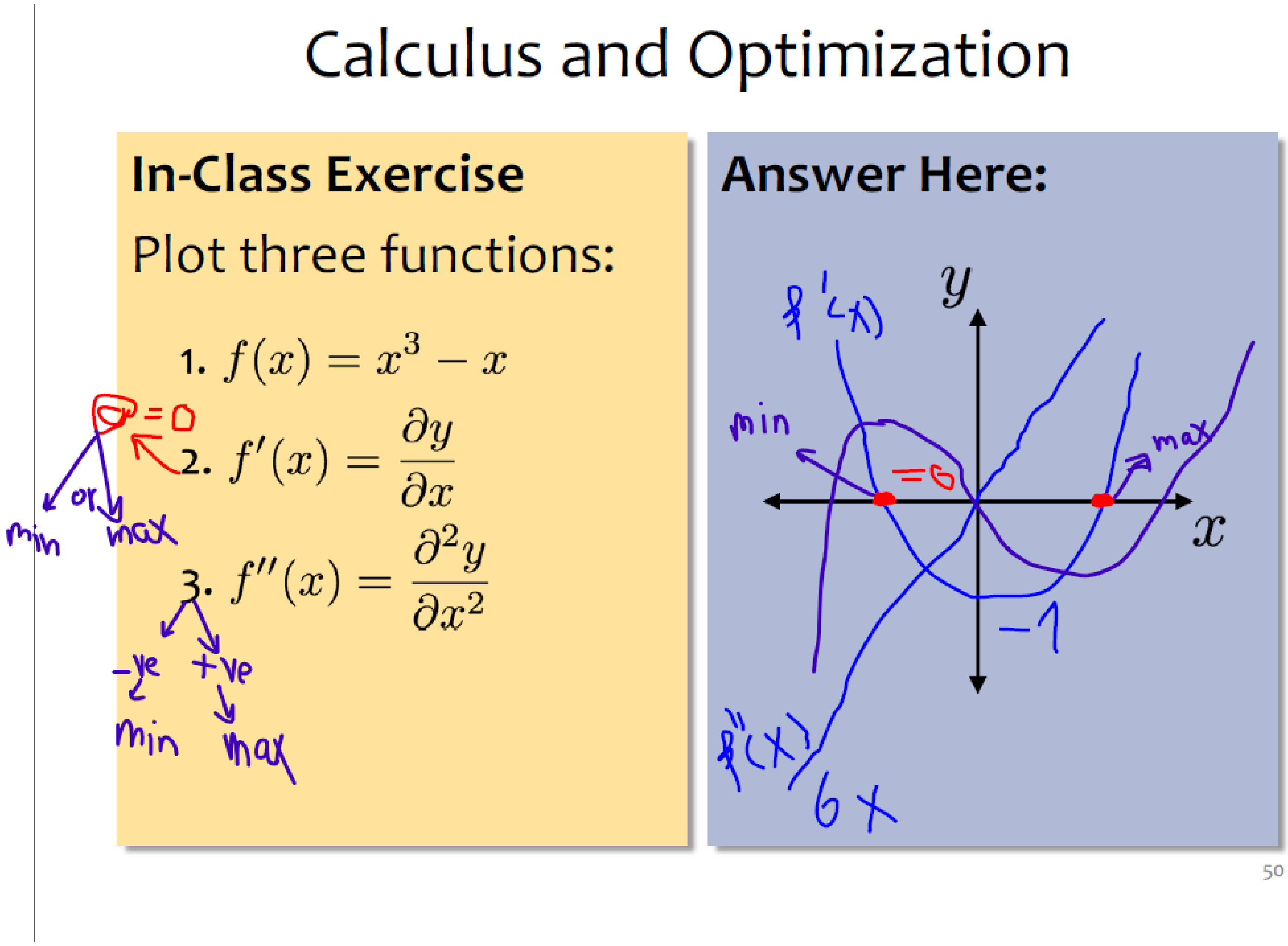
تعال نحط شوية تعريفات .. لو عندك جينيريك فانكشن اسمها f(x) القيمه x star هي جلوبال منيمم if and only if .. ال f(x star) اقل من ال f(x) لكل القيم المحتمله لل x .. هنقول انها لوكال منيمم ده لو موجود some epsilon اللي هو some window .. +epsilon and –epsilon .. هتلاقي ان عندك local minimum .. فلو عندك convex function .. single local minimum هي global minimum .. لو لاقيت أي منيمم .. أمسك فيه ... " أخيراً لاقيتك .. رايح فين" ... انما لو NonConvex ... مش شرط







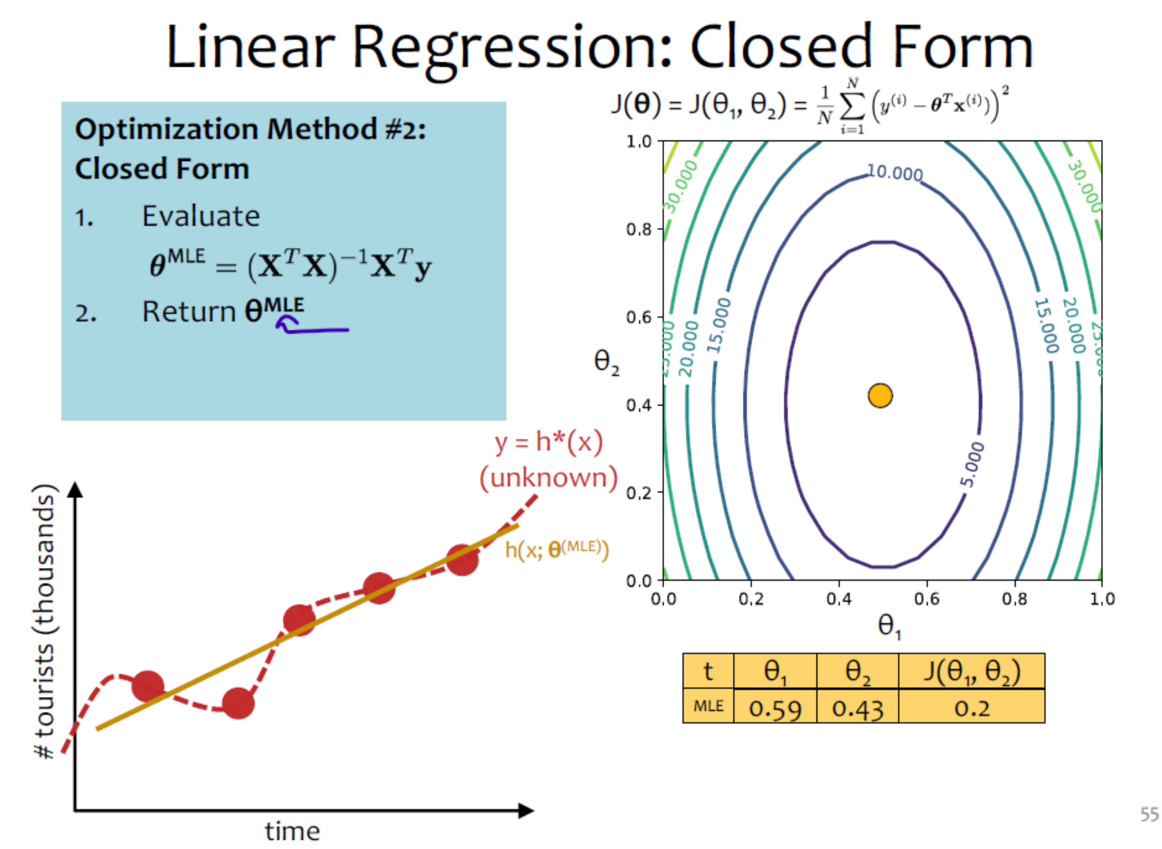
تعال نشوف ال closed form ..



فبالتالي لما تيجي تحل في الكلوزد فورم ... تقدر ت jump علي طول لل local minimum .. لو انت بس عرفت حاجه عن ال gradient بتاع الفانكشن .. فالحل في الكلوزد فورم .. انك مثلا عندك M dimensional function J(theta) and theta is M dim. Function ...

1. هنحل المعادله بتاعت ال Gradient (J(theta)) = 0 for theta
2. Test for min, max or saddle point using second derviative

فالدكتور بيقول .. هو مش هيبقا حاجه جامده جداً لو بدل ما كنا بناخد شوية STEPS صغيره بجريدينت ديسنت .. احنا نحل علي طول المين سكويرد إيرور ونجيب قيمة للثيتا ليها الجريدينت بصفر .. لو لاقينا القيمه ديه .. والفانكشن اصلا كونفيكس .. يبقا خلاص يا ماااان يبقا انت وصلت للجلوبال منيمم ...



هي خطوه واحده وخلصنا فعلاً ... ده الكلوزد فورم سلوشن .. فالكلوزد فورم سلوشن قول مثلاً عندك فيكتور اسمو Y جاواه كل الأوتبوت فاليوز .. و بعدين هيبقا عندك ماتركس اسمها X وديه بتتكون من كل الترينينج إكزامبلز بتاعتنا .. من أول فيتشر لحد آخر فيتشر .. الماتركس ديه الدايمنشنز بتاعتها

N examples * M features .. بحيث إن أول فيكتور هو اول مثال و آخر رووو فيكتور هو آخر مثال .. والماتركس إكس اسمها Design Matrix ...

وبعدين نكتب ال J(Theta) ...

$\boxed{\text{Closed Form Solution}}$

Ex: M-dim fn. $J(\vec{\theta})$, $\vec{\theta} \in \mathbb{R}^M$

① Solve $\nabla J(\vec{\theta}) = 0$ for $\vec{\theta}$

② test for min, max, or saddle point using second derivatives

$\boxed{\text{Closed Form Solution for Lin Reg.}}$

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix} \qquad \mathbf{X} = \underbrace{\begin{bmatrix} x_1^{(1)} \cdots x_M^{(1)} \\ \vdots \\ x_1^{(N)} \cdots x_M^{(N)} \end{bmatrix}}_{\text{Design Matrix}} \begin{matrix} \leftarrow \vec{x}^{(1)} \\ \\ \leftarrow \vec{x}^{(N)} \end{matrix}$$

① Write $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(y^{(i)} - \vec{\theta}^T \vec{x}^{(i)}\right)^2 \leftarrow \text{Ordinary Least Squares (OLS)}$$

$$= \frac{1}{N}\cdot\frac{1}{2}\left(\mathbf{X}\vec{\theta} - \vec{y}\right)^T\left(\mathbf{X}\vec{\theta} - \vec{y}\right)$$

② "Normal Equations" ← gradient set to zero

$$\nabla J(\theta) = X^T X \vec{\theta} - X^T \vec{y} = 0$$

① Write $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(y^{(i)} - \vec{\theta}^T\vec{x}^{(i)}\right)^2$$ ← Ordinary Least Squares (OLS)

$$= \frac{1}{N}\cdot\frac{1}{2}\left(X\vec{\theta} - \vec{y}\right)^T\left(X\vec{\theta} - \vec{y}\right)$$

② "Normal Equations" ← gradient set to zero

$$\nabla J(\theta) = X^TX\vec{\theta} - X^T\vec{y} = 0$$

③ Solve for $\vec{\theta}$

$$\vec{\theta}^{MLE} = (X^TX)^{-1}(X^T\vec{y})$$

$$= \underset{\theta}{\text{argmin}}\, J(\vec{\theta})$$

وده ال closed form solution ..

Computational Complexity of OLS

To solve the Ordinary Least Squares problem we compute:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^{(i)} - (\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y})$$

The resulting shape of the matrices:

$$\underbrace{(\underbrace{\mathbf{X}^T}_{M\times N}\ \underbrace{\mathbf{X}}_{N\times M})^{-1}}_{M\times M}\ \underbrace{(\underbrace{\mathbf{X}^T}_{M\times N}\ \underbrace{\mathbf{Y}}_{N\times 1})}_{M\times 1}$$

Background: Matrix Multiplication Given matrices $\mathbf{A}$ and $\mathbf{B}$

- If $\mathbf{A}$ is $q \times r$ and $\mathbf{B}$ is $r \times s$, computing $\mathbf{AB}$ takes $O(qrs)$
- If $\mathbf{A}$ and $\mathbf{B}$ are $q \times q$, computing $\mathbf{AB}$ takes $O(q^{2.373})$
- If $\mathbf{A}$ is $q \times q$, computing A^{-1} takes $O(q^{2.373})$.

Computational Complexity of OLS:

$\mathbf{X}^T\mathbf{X}$	$O(M^2N)$
$(\quad)^{-1}$	$O(M^{2.373})$
$\mathbf{X}^T\mathbf{Y}$	$O(MN)$
$(\quad)^{-1}(\quad)$	$O(M^2)$
total	$O(M^2N + M^{2.373})$

Linear in # of examples, N
Polynomial in # of features, M

57

طبعاً دلوقت فرحت .. هيبيه هوب دابل كيك نعمل الطريقه ديه بقا لكل الماشين ليرننج ولأي حاجه فيها لينير ريجريشن .. طبعاً انت دافع تمن الحل ده ك computational cost و التمن ده مش سهل .. كل ما عدد الفيتشرز زاد و عدد ال examples زادو .. كل ما هتدفع أكتر .. فده مش حل كويس الحقيقه غير لما يكون ال N and M small