

Abdelrahman Rezk

Web Developer

NLP & ML student

AOU University

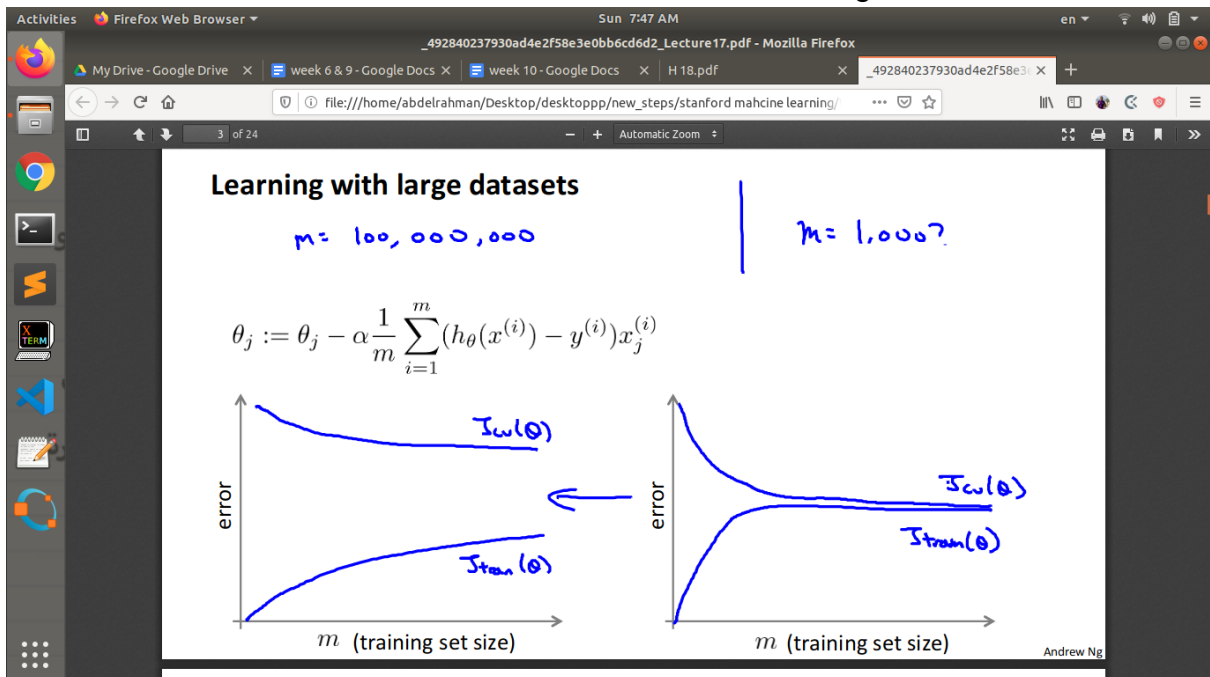
Big Data

"It's not who has the best algorithm that wins, It's who has the most data". It is given by Andrew Ng

من المهم جدا قبل أى تفكير فى ال algorithm او اى حاجة تانية هو التفكير ال data نفسها لأن ال data هى ال بيتبنى عليها كل حاجة بعد كده وال data بتفرق جدا مع كبر حجمها وعدد ال features ال فيها وده بيختلف هل زيادة ال data مفيد بعد وقت معين ولا لا يعنى بعد استخدام algorithm ما على ال data لى عندى ممكن ساعتها اعرف زيادة عدد ال examples فى ال data ها يكون مفيد ولا لا لكن قبل أى شىء المهم اول حاجة بنفكر فيها هي وجود ال data.

بتقابلنا مشكلة تانية الا وهي بعد وجود ال data بكميات ضخمة وهائلة كيفية التعامل مع ال data دي بيتم ازاي وانا هنا بتكلم عن ملايين الصفوف من ال data مش مجرد 1000 صف بنتعامل معاهم فهنا اى algorithm مثلا لو قلنا ال linear regression فهنا العملية بتاعت ضرب ال thetas فى xs وطرحها من ال actual value ال هى ال y وبعد ده كله اجمعهم فأنا هنا لو بتكلم فى مليون مثال فقط العملية بتكون مكلفه جدا سواء على ال machine او ك cost او ك time ما بالك بقا لو ساعتها محتاجين نعمل gradient descent عشان نبتدى نجيب القيم المثلى لل thetas ال بتقلل ال cost function لل global minimum فلو قلنا عدد ال 100 iterations فانتا هتضرب 100 فى عدد ال examples ال انت بتحاول تنطبق عليها المعادلة والكلام ده ممكن يتطلب اسابيع او اكثر عشان يتم عمله.

ومن الحاجات ال ممكن تساعد فى كده هو انى اعمل shuffle لل data الى عندى ثم ابتدى انى استخدم جزء منهم و قسمه بال cross validation ارسم ال cost function واشوف ساعتها هل جزء البيانات ده كافى ولا لا انى اعمل model كويس يعمل fitting for data.



Stochastic gradient descent

المشكلة الى بتبدأ تظهر عندى مع وجود بيانات ضخمة هو انى بحاول قلل قيمة ال cost function عن طريق ال thetas parameters الى بحاول انى اشوف القيمة ال optimal ليها عشان تقدر استخدمها فى الاخر مع اى data تانية انا مشفوتهاش وبعتر انى قيم ال thetas ديه بتكون المثلى وانها هى بتقلل قيمة ال cost function الى عندى ال global minimum.

ولكن بتبدي تظهر مشكله معايا انى عشان ققل قيم ال thetas ديه محتاج انى اعمل gradient descent الى بيكون فيه تقليل ال thetas ديه مكلف جدا لانى مبيقاش عارف بظبط انى محتاج اعمل كام iteration عشان تقدر اوصل للقيم المثلى لل thetas ومع كل iteration انا بعمل عملية على بيانات كبيرة جدا وده بيكون مكلف جدا فى كذا حاجة سواء وقت فلوس او على ال machine نفسها.

وهنا ببدي افكر فى طريقة تانية للتعامل مع ال data وهى Stochastic gradient descent التدرج العشوائى وهو انى ببدي بدل ما كنت بغير قيم ال thetas بعد كل iteration وعمل مجموع لفرق كل قيم ال predictive من ال actual وبعد كده نضربهم فى عدد الامثلة اللى عندى والى ممكن يوصل لاكثر من مليون. هنا بقا ببدي افكر فى طريقة زى الى فوق ديه وانى هغير قيم ال theta بناء على كل مثالا لوحده. بس لازم اكون اولاً عامل shuffle لل data لى عندى وتكون مرتبة بشكل عشوائى عشان تقدر اشتغل بالطريق دى وانى قيم ال theta لما هانتغير مش هتكون بتتغير بناء على رؤية جزء معين من ال data لا هى هانتغير بعد كل مثال لحد أما اوصل للقيم المثلى ليها.

The screenshot shows a Firefox browser window with the address bar displaying a file path: `file:///home/abdelrahman/Desktop/desktopppp/new_steps/stanford machine learning/`. The slide content is in Arabic and discusses the challenges of Batch Gradient Descent and introduces Stochastic Gradient Descent.

• فين بقي المشكلة :

- المشكلة ان التعامل مع المعادلة ديه , بما فيها السمين $\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_j^{(i)}$ (وهي اسمها Batch Gradient Descent) ده معناه وقت رهيب جدا في قراءة عشرات الملايين من الصفوف , واعمل لها سميثن , واشوف الفروق و اطرح , بعدها هعيد نفس الخطوة كذا مرة , وده معناه وقت رهيب , وتكليف كبير عشان كدة هنعمل فكرة , عشان نتجنب ضياع الوقت
- قيل ما نفهم الفكرة , عايزين نعرف اصلا هو ال Batch Gradient Descent (الطريقة العادية) بيشغل ازاى

Batch gradient descent

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \dots, n$)

$m = 300,000,000$

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.
2. Repeat {
 - for $i = 1, \dots, m$ {

$$\theta_j := \theta_j - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$
 (for $j = 0, \dots, n$)

$\rightarrow (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

Andrew Ng

Mini--- batch gradient descent

فيه طريقة تانية ببتدى من خلالها أشتغل مع ال data هي ال mini-batch هي ببساطة اني بمسك ال data الى عندي مقسمة لأجزاء معينة مثلا في range 2-100 فلو كان عندي مثلا ال data مليون صح وعابز ققسمهم لاربعة اجزاء فقسمهم ربع مليون لكل جزء وبعدها ببتدى أشتغل بطريقة ال batch gradient descent.

Mini-batch gradient descent

- Batch gradient descent: Use all m examples in each iteration
- Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

$b = \text{mini-batch size}$. $b = 10$. $\frac{2-100}{10}$

Get $b = 10$ examples $(x^{(1)}, y^{(1)}) \dots (x^{(b)}, y^{(b)})$

$$\theta_j := \theta_j - \alpha \frac{1}{b} \sum_{k=1}^b (h_{\theta}(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

$j := j + 1$

Andrew Ng