Deep Learning التعلم العميق

- الدرس الأول : **التعلم العميق و الشبكات العصبية**
  - الأسبوع الأول : مقدمة للتعلم العميق
  - الأسبوع الثاني : أساسيات الشبكات العصبية
  - الأسبوع الثالث : الشبكات العصبية المجوفة
  - الاسبوع الرابع : الشبكات العصبية العميقة
- الدرس الثاني : **تطوير الشبكات العميقة : المعاملات العليا**
  - الأسبوع الأول : السمات العملية للتعلم العميق
  - الأسبوع الثاني : الحصول علي القيم المثالية
  - الأسبوع الثالث : ضبط قيم الشبكات العميقة
- الدرس الثالث : **هيكلية مشاريع الـ ML**
  - الأسبوع الأول : استراتيجيات الـ ML - 1
  - الأسبوع الثاني : استراتيجيات الـ ML - 2
- الدرس الرابع : **الشبكات العصبية الملتفة CNN**
  - الأسبوع الأول : أساسيات الشبكات العصبية الملتفة
  - الأسبوع الثاني : حالات عملية من الشبكات العصبية الملتفة
  - الأسبوع الثالث : التعرف علي الأشياء
  - الاسبوع الرابع : التعرف علي الوجه
- الدرس الخامس : **الشبكات العصبية المتكررة RNN**
  - الأسبوع الأول : مفهوم الشبكات العصبية المتكررة
  - الأسبوع الثاني : المعالجة اللغوية الطبيعية NLP
  - الأسبوع الثالث : نماذج التتابع

.

درس 2: تطوير الشبكات العميقة : المعاملات العليا

الأسبوع الثاني : الحصول علي القيم المثالية Optimization

==================================

- عقب الانتهاء من هذا الكورس , ستكون قادر علي :
  - التعرف علي عدد من طرق الحصول علي القيم المثلي زي الطريقة العشوائية , او الزخم , او RMS
  - استخدام اسلوب الكميات الكبيرة العشوائي , لتسريع عملية الـ optimization
  - التعرف علي مميزات تضائل معامل التعلم , واستخدامه في الـ optimization

-

هذا الاسبوع نتناول بالتحديد الاساليب والتكنيكات الخاصة بما يسمي الـ Optimization و هي الطرق المستخدمة للحصول علي الارقام المناسبة للخوارزم

و من عيوب التعلم العميق و الشبكات العميقة , أنها لا تعمل بفعالية مع البيانات الضخمة Big Data وذلك لان الشبكات العميقة اساسا تستغرق وقتا كبيرا , وإذا ما أضيف عليها مشكلة التعامل مع كم كبير من البيانات , فيعني هذا وقتا مهولا

لذا كان من الضروري الوصول لطرق الحصول علي القيم المثالية في وقت قصير , وهو نقطة بحثنا هنا

وسنتناول الان تكنيك : الانحدار الاشتقاقي للمجاميع الصغيرة mini-batch gradient descent

ولفهمها , علينا ان نتناول مثالا مثل هذا :

لو قلنا ان لدينا عدد من عناصر العينة m , وكل عنصر لديه عدد من الـ features بقيمة n , فتكون لدينا مصفوفة X كابيتال هكذا

$$X = [x^{(1)}\ x^{(2)}\ x^{(3)} \ldots\ x^{(1000)} \mid x^{(1001)} \ldots x^{(2000)} \mid \ldots \qquad \mid \ldots\ x^{(m)}]$$

حيث أن كل x من هؤلاء هي فيكتور من عمود واحد و الصفوف هي الـ features

إذن يكون الابعاد النهائية للمصفوفة X هي (n x m)

و بالتالي تكون مصفوفة y بالمثل هي :

$$Y = [y^{(1)}\ y^{(2)}\ y^{(3)} \cdots y^{(1000)} \mid y^{(1001)} \cdots y^{(2000)} \mid \cdots \mid \cdots y^{(m)}]$$

ذات الابعاد (1 x m)

والمشكلة التي تواجهنا لدي التعامل مع البيانات الضخمة هي أن , إذا ما اردنا تطبيق تكنيك الانحدار الاشتقاقي gradient descent فيجب علي الخوارزم , أن يقوم بمعالجة العينة كلها حتي لو كان عددها 5 ملايين مثلا , وحينما ينتهي يقوم بتعديل قيم w ثم نقوم مرة اخري بمعالجة العينة كلها , وهو ما يستغرق وقتا مهولا , وهذه الطريقة القديمة اسمها Batch Gradient Descent او الانحدار الاشتقاقي للمجموعة كلها

أما فكرة تكنيك المجاميع الصغيرة (mini-batch gradient descent) , تقوم علي أنه , بدلا من أن يقوم الخوارزم بمعالجة كل العينة الكبيرة , ثم يقوم بتعديل قيم w , فإنه يقوم بتعديل تدريجي كلما قام بمعالجة جزء من العينة . .

وبالتالي لو قمنا بتقسيم العينة إلي 100 جزء مثلا , فيقوم الخوارزم بمعالجة الجزء الاول , وعمل تعديل update لقيم الثيتات , ثم التعامل مع الجزء الثاني بالقيم المعدلة من الثيتات , مما يجعل نسبة الخطأ أقل , ثم يقوم بتعديل الثيتات عقب الجزء الثاني , ثم يقوم بمعالجة الجزء الثالث بقيم ثيتات المعدلة , وهكذا . . .

وحين يصل لنهاية العينة , يكون الخوارزم بالفعل قد قام بتعديل قيم ثيتات مئات المرات , فيصل لقيم قريبة جدا من المطلوبة , وتكون معادلة الخطأ cost function قليلة للغاية

وبالتالي إذا ما أعاد الكرة مرة اخري , فيكفي ان يقوم بعدد محاولات iterations قليلة حتي نصل للقيم المثالية

وأول خطوة تكون تقيم العينة إلي عدد متساوي من الأجزاء , ولنقل كل جزء يكون الف عنصر , فإذا كانت العينة 5 مليون عنصر , نقوم بتجزيئها لـ 5 آلاف جزء , كل جزء الف عنصر هكذا :

$$\underset{(n_x,m)}{X} = [\underbrace{x^{(1)}\ x^{(2)}\ x^{(3)} \dots x^{(1000)}}_{X^{\{1\}}\ (n_x,1000)} | \underbrace{x^{(1001)} \dots x^{(2000)}}_{X^{\{2\}}\ (n_x,1000)} | \dots | \underbrace{\dots x^{(m)}}_{X^{\{5000\}}\ (n_x,1000)}]$$

الجزء الأول من رقم 1 إلي 1000 , الثاني من 1001 إلي 2000 , وهكذا , وهي ما تسمي المجاميع الصغيرة mini-batches

ونقوم بتسمية الأجزاء بطريقة X{3} , X{2} , X{1} (الرقم لأعلي) بحيث يشير الرقم لرقم الجزء

لا تنس ان التسمية العليا superscript تختلف حسب نوع القوس . .

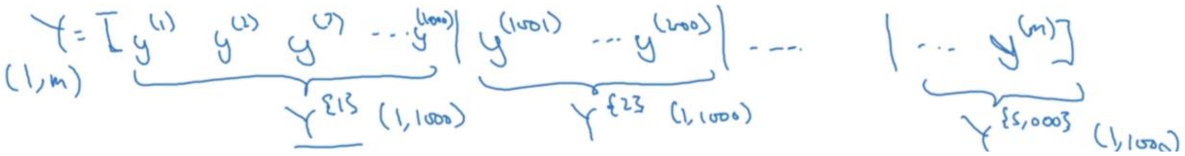
فحينما نقوم بعمل تسمية عليا بقوس دائري x(5) , فنحن نعني العنصر الخامس من العينة

وحينما نقوم بعمل تسمية عليا بقوس مربع z[3] ,فنحن نعني الطبقة الثالثة من الشبكة العصبية

بينما حينما نشير بالقوس المتعرج x{15} فنحن نعني الجزء الخامس عشر من المجاميع المقسمة

و بالتالي فإن أبعاد مصفوفات المجاميع x ستكون (n x 1000) حيث أن صفوفها هي نفس صفوف x الاصلية و هي عدد الـ features بينما اعمدتها هي عدد 1000 عنصر تم اجتزائه

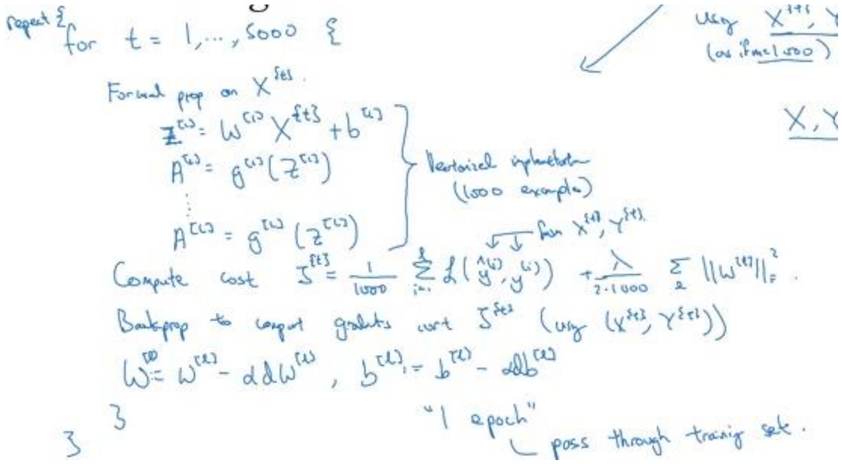
ونقوم بعمل نفس التجزيئ بنفس العدد المستقطع المستخدم في x , نقوم به في y , حتي يقوم الخوارزم بعمل حسابه الدقيق دون اضطراب , وبالتالي ستكون y :



علي أن تكون ابعاد كل منها (1 x 1000)

وبالتالي ستقوم الشبكة العصبية بعمل المسار الامامي والخلفي لكل جزء علي حدة , ومن ثم عمل تعديل لقيم b , w ثم تناول الجزء التالي و هكذا , و عقب الانتهاء من العينة بالكامل (5 الاف جزء) يتم تكرار العملية بالكامل عددا من المرات , واقوم بفحص قيمة J للوصول الي القيمة المناسبة

نبدأ بكتابة الكود الخاص لها



نري أولا أن هناك for تبدا من 1 إلي 5000 , وهي التي ستتناول الاجزاء من الجزء الاول للاخير

بعدها اقوم بحساب المسار الامامي للـ X{t} , لا تنس أن t هو العنصر الذي يحمل الرقم المتغير من 1 إلي 5000

فنقوم بحساب قيم Z ثم A لكل طبقة بالترتيب حتي أصل لقيمة y^ , ثم اقوم بحساب دالة الخطأ cost function و لا انسي قيمة التنعيم اذا ما كنت ساستخدمها

بعدها اقوم بالمسار الخلفي للوصول لقيم dw , db , ثم اقوم بالخطوة الأهم و هي تعديل قيم w , b

كل هذا يسمي 1 epoch اي خطوة واحدة , فيتم الرجوع لدالة for ليتناول الجزء الثاني X{2} و يتعامل معه بالقيم المعدلة للـ w , b , ثم الجزء الثالث و هكذا

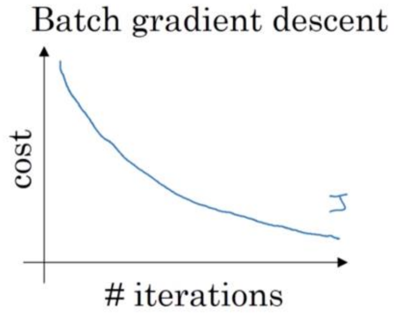
بعدما ينتهي من العينة بالكامل , نقوم بتكرار العملية من بدايتها مرات عديدة , حتي نصل للقيمة المثلي للأوزان w , b

-

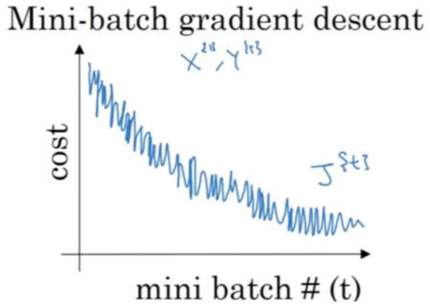
دعنا نتناول عدد من خصائص الـ mini-batch gradient descent

أولا إذا تكلمنا عن معادلة الخطأ cost function و الذي نرمز لها بـ J

ففي الاسلوب التقليدي Batch Gradient Descent نري أن قيمة J دائما ما تنخفض مع كل محاولة جديدة iteration هكذا



بينما في تكنيك المجاميع الصغيرة , فقد تري ان الشكل يختلف قليلا :



والسبب في هذا التذبذب , أنه في خلال نفس المحاولة قد يصادف الخوارزم جزء X{1} مثلا يكون قيمة الخطا فيه قليلة , بينما بعدها قد يصادف X{6} تكون قيمة الخطأ كبيرة , لانها عينة كبيرة

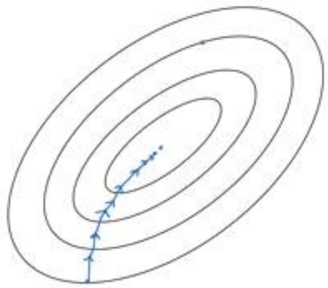
لكن ستلحظ أن الاتجاه العام للمنحني يكون لاسفل , كما أنه اسرع كثيرا

نصل لسؤال مهم , علي أي أساس اقوم بتحديد عدد عناصر الجزء الواحد , هنا قلناه أنه 1000 , لكن هل احدد رقم اكبر او اصغر ؟ ؟

لنجيب علي السؤال , علينا التعرف علي القيمتين المتطرفتين في هذا الأمر :

نتعامل بداية مع القيمة القصوي , وهي أن نفرض أن عدد عناصر الجزء هو نفسه العدد الكلي للعناصر وهو m , وهو ما يجعل عدد الأجزاء هو واحد فقط , وبهذا سنصل لمفهوم Batch Gradient Descent مرة أخري

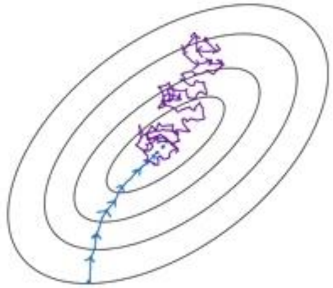
وإذا اردنا تتبع مسار قيمة J هنا سيكون كالتالي :



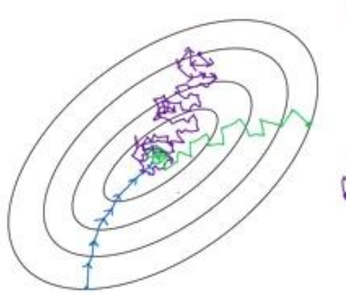
نري أنه مسار مباشر دون التفاف , لكن يعيبه الوقت الطويل اللازم لتنفيذه , فكل خطوة فيه يستلزم المرور بالملايين الخمسة للعناصر

وإذا تناولنا الحد الأدني لعدد عناصر الجزء , سيكون 1 , لانه يستحيل ان يتم رقم اقل من هذا , وهذا معناه أنه سيكون لدينا خمسة ملايين جزء , كل جزء من عنصر واحد , وعقب معالجة كل جزء فيهم , يقوم الخوارزم بعمل update لقيم w , b

و هذا النظام يسمي Stochastic gradient descent اي الانحدار الاشتقاقي العشوائي , وستري أن خطوات تعديل قيمة الاوزان و بالتالي تغيير معادلة الخطأ J سريعة جدا , لكن مشكلتها الاساسية انها ستكون متذبذبة المسار , حيث أن كل عنصر من عناصر العينة يأخذها في اتجاه , وهو ما يجعل وصولها للقيم المثلي شديد الصعوبة و الوقت الطويل , وقد لا يصل في النهاية



المنطقة الوسطي بينهما هي ما نتناوله , وهو المجاميع الصغيرة , حيث نستفيد من ميزة كلا منهما , التعديل السريع , والاتجاه شبه المباشر



وكلما اقترب عدد عناصر الجزء من m كلما أخذ اكثر من خصائص الـ Stochastic , وكلما اقترب اكثر من 1 كلما اقترب من خصائص الـ Batch

وأخيرا . .

لو كان العدد الكلي لعناصر العينة رقم قليل نوعا (اقل من 2000) فيمكن استخدام تكنيك Batch باعتبار أن الرقم معقول و لن يصعب علي الخوارزم تكراره

أما إذا زاد عن ذلك فعليك بالـ mini-batch فورا , ولا تستخدم الـ stochastic في اي حالة

و يفضل أن يكون عدد عناصر كل جزء فيهم هو من مشتقات رقم اثنين (64 , 128 , 256 , 512 , 1024) وهكذا , لانه يكون اسرع كثيرا من الارقام الاخري , وذلك لتطابقها مع النظام الثنائي الذي يعمل به اي بروسيسور , واغلب الارقام المستخدمة تكون في هذه الارقام

-

نتكلم الان عن عدد من الخوارزميات الأخري , والتي قد تكون أفضل في معالجة الشبكات العصبية من الانحدار الاشتقاقي . .

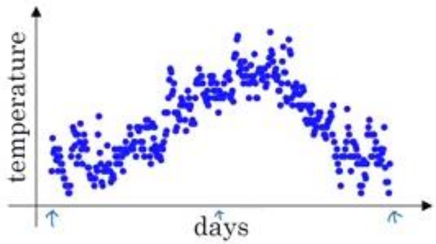
منها ما يسمي (المتوسطات الأسية Exponentially weighted averages)

ولنفهم ماهيتها , علينا أن نتناول عددا من الارقام مثل هذه :

لو أخذنا درجات الحرارة في لندن علي مدي ايام السنة , علي اعتبار ان ثيتا 1 تمثل يوم 1 يناير , وثيتا 180 تمثل يوم 1 يونيو , وثيتا 365 تمثل 31 ديسمبر , فستكون الدرجات كالتالي :

$$\theta_1 = 40°F$$
$$\theta_2 = 49°F$$
$$\theta_3 = 45°F$$
$$\vdots$$
$$\theta_{180} = 60°F$$
$$\theta_{181} = 56°F$$
$$\vdots$$

إذا ما اردنا أن نرسمها علي جراف فستكون كالتالي :



ومشكلة هذا الرسم انه شديد التذبذب , وذلك بسبب كمية كبيرة من المعلومات لا احتاجها , فانا اريد رسم أكثر نعومة و إيضاحا من هذا , فسنقوم بعمل بحساب المتوسط الأسي

وهذا عبر تحديد معامل معين بيتا β , الذي سنقوم بضربه في قيمة اليوم السابق (ليس درجة الحرارة لكن قيمة V) , ونجمعه علي قيمة 1 ناقص بيتا , مضروبة في درجة حرارة اليوم نفسه , ليكون اسمه Vt , كالتالي :

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

وهو ما سيجعل قيمة Vt لا تمثل فقط درجة اليوم ,لكنها مرتبطة بقوة بالدرجات السابقة لها , وبالتالي فأي ارتفاع او انخفاض في درجات الحرارة في اليوم , لا تجعل القيمة تتغير بشكل كبير , لكن بدرجة قليلة

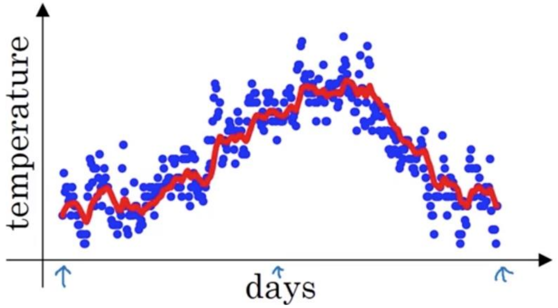
وإذا اخترنا لبيتا قيمة 0.9 , تكون المعادلات كالتالي :

$$\begin{aligned}
V_0 &= 0 \\
V_1 &= 0.9\,V_0 + 0.1\,\theta_1 \\
V_2 &= 0.9\,V_1 + 0.1\,\theta_2 \\
V_3 &= 0.9\,V_2 + 0.1\,\theta_3 \\
&\vdots \\
V_t &= 0.9\,V_{t-1} + 0.1\,\theta_t
\end{aligned}$$

وبالتالي :

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$
$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$

ويكون الرسم كالخط الأحمر هنا :



فالخط هنا أكثر هدوء و استقرار , و اقل تذبذب , ويجعل التغييرات ابطئ قليلا

وكلما زاد مقدار بيتا , زاد البطئ في التغير حتي يتحول الرسم لما يشبه المعادلات الخطية , وكلما قلت قيمة بيتا , كلما زاد التذبذب

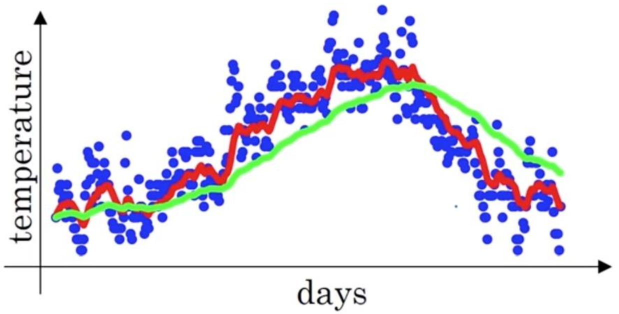
وذلك لأن قيمة بيتا يتم ضربها في قيمة الايام السابقة , فكلما زادت قيمة بيتا , قلما قل تأثير قيمة اليوم و زاد ارتباطها بالايام السابقة

و يقال أن الارقام الحالية لا تمثل العلاقة بين درجات الحرارة لكل يوم ,لكن لمتوسط عدد من الايام , يرتبط بقيمة بيتا , بالمعادلة

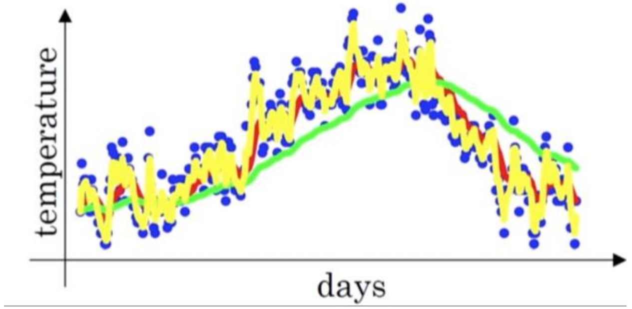
$1 / (1 – β)$

وهذا معناه أن , في حالة تحديد بيتا برقم معقول , مثلا 0.8 , فيكون المقام 0.2 , بالقسمة عليه يكون 5 , فوقتها يكون الرسم يمثل متوسط لكل خمسة ايام

وإذا زاد الرقم ووصل مثلا لـ 0.98 , فالمقام سيكون 0.02 , وبالقسمة عليه سيكون خمسين , وهذا معناه متوسط خمسين يوما , وهو ما سيتم رسمه بالخط الأخضر :



اما اذا ما اخترنا رقم صغير نسبيا لبيتا , مثلا نصف , وهو ما يجعل المقام بـ 0.5 , و يكون المتوسط يومين , أي سيكون الرسم لمتوسط يومين فقط , وهو ما يجعل قيمة المتوسطات غير مرتبطة كثيرا بالقيم السابقة , و سيكون قيمة تذبذبها كبيرا , مثل الخط الاصفر :



واذا قمنا بمزيد من البحث في نفس النظرية . .

فالمعادلة المستخدمة هي :

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

وبالتالي

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$
$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$

وإذا قمنا باستبدال v99 بمعادلتها , ثم قيمة v98 بمعادلتها الداخلية و هكذا , تصير المعادلة :

$$V_{100} = 0.1\theta_{100} + 0.1 \cdot 0.9 \cdot \theta_{99} + 0.1 \cdot 0.9^2 \cdot \theta_{98} + 0.1 \cdot 0.9^3 \cdot \theta_{97} + \ldots$$

أي أن قيمة أي v لاي يوم , هو قيمة (β-1) في ثيتا لنفس اليوم , مجموعة علي (β-1) في (β) في ثيتا اليوم السابق , مجموعة علي (β-1) في (β) تربيع , في اليوم السابق , مجموعة علي (β-1) في (β) تكعيب في اليوم السابق وهكذا

ولاحظ ان الصيغة الرياضية الخاصة ببيتا هنا :

(1-x)^(-1/x)

علي اعتبار ان x هي $(1-\beta)$, الصيغة ديه مع كبر ارقام البيتا , تؤدي للرقم e

وبالتالي المعادلة العامة هتكون :

$$
\begin{aligned}
v_0 &= 0 \\
v_1 &= \beta v_0 + (1-\beta)\,\theta_1 \\
v_2 &= \beta v_1 + (1-\beta)\,\theta_2 \\
v_3 &= \beta v_2 + (1-\beta)\,\theta_3
\end{aligned}
$$

يعني الخوارزم الخاص بيها :

$$
\begin{aligned}
V_\theta &:= 0 \\
V_\theta &:= \beta V + (1-\beta)\,\theta_1 \\
V_\theta &:= \beta V + (1-\beta)\,\theta_2
\end{aligned}
$$

ولو هنتعامل ككود ,

$V_\theta = 0$

Repeat {

Get next θ_t

$V_\theta := \beta V_\theta + (1-\beta)\theta_t$

}

واللي نكون فيها قيمة لـ v و نقوم بعمل update لها بضرب بيتا في v الاصلية , مجموعة علي 1 ناقص بيتا , مضروبة في ثيتا لدرجة اليوم نفسه

و ميزة هذا الكود انه سريع و سهل التعامل مع البروسيسور , ولا يستهلك ذاكرة كثيرة

-

ولدي تطبيق هذا الخوارزم , ستلحظ سلوكا غريبا , خاصة في حسابات الايام الاولي من درجات الحرارة . .

فنحن نفترض دائما V0 علي انها صفر . .

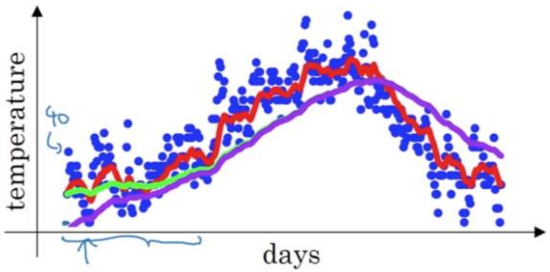
وبما أننا نتعامل مع هذه المعادلة :

$$v_0 = 0$$
$$v_1 = \beta v_0 + (1-\beta)\,\theta_1$$
$$v_2 = \beta v_1 + (1-\beta)\,\theta_2$$
$$v_3 = \beta v_2 + (1-\beta)\,\theta_3$$

فتجد أنه لو كانت درجة حرارة اول يوم مثلا تساوي 40 , فستكون قيمة v1 والتي يفترض ان تمثل متوسط اليوم الأول , ستكون بيتا في v0 والتي تساوي صفر , مضروبة في 0.1 في 40 , والتي ستساوي فقط 4 درجات

في اليوم التالي إذا كانت درجة الحرارة 45 مثلا , ستكون 0.9 في 4 , زائد 0.1 في 45 , اي 8 درجات

وهكذا ستجد أنه في اول ايام فإن درجات الحرارة المحسوبة بعيدة جدا عن الدرجات الحقيقية , مما يرسم هذا الشكل



فإذا كانت النقاط الزرقاء هي الارقام الحقيقية , والخضراء هي المفترض تمثل بيتا 0.98 , فتجد أن الخط البنفسجي هو ما سيتم رسمه

وتلحظ ان في بداية الايام فإن الخط البنفسجي ذو قيمة قليلة للغاية , ويرتفع تدريجيا حتي يتطابق مع الاخضر

ولاصلاح هذا الامر , نقوم بتطبيق ما يسمي Bias Correction أي معامل اصلاح الخطأ

و تكون معادلته :

$$\frac{V_t}{1-\beta^t}.$$

أي عقب حساب قيمة Vt لاي يوم , قم بقسمتها علي 1 ناقص بيتا أس t , و الرقم t يمثل رقم اليوم

فإذا كنا في اليوم الاول , فسيكون المقام 1 ناقص بيتا , وهو الذي يمثل 0.1 , وإذا قسمنا قيمة Vt القليلة علي 0.1 , فهذا يكبرها جدا
و بعد 5 ايام , تكون قيمة 0.9 أس 5 تساوي 0.6 , اي المقام سيكون 0.4 , والقسمة عليه ستزيد الرقم Vt بعض الشئ
و بعد 20 يوم , قيمة 0.9 أس 20 هي 0.12 , والمقام سيكون 0.88 , اي القسمة عليه ستزيد Vt قليلا
و بعد 5 يوم , قيمة 0.9 اس 50 هي 0.005 , والمقام 0.995 , فالقسمة عليه ستزيده بلا شئ تقريبا . .

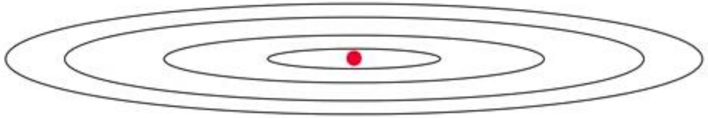
و هذا معناه ان هذه المعادلة , يتم تطبيقها علي كافة الايام , وستقوم تلقائيا , بزيادة الايام الاولي بارقام كبيرة , و تقل الزيادة تدريجيا , حتي تصل لصفر تقريبا مع الايام البعيدة

-

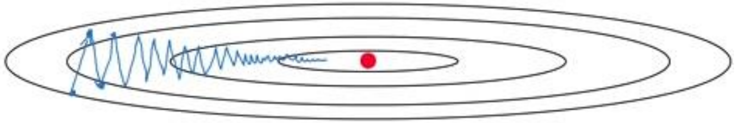
ولاستخدام هذا التكنيك في الشبكات العميقة , سنتعرف علي مفهوم هام يسمي الانحدار الاشتقاقي بالزخم Gradient descent with momentum

وهو يعتبر تكنيك سريع و قوي للوصول للقيم المثالية , مقارنة بالانحدار الاشتقاقي التقليدي

والفكرة تبدأ من هذا الشكل :



إذا كان لدينا القيمة المثلي هي النقطة الحمراء , و سنقوم بالبدء من أحد النقاط , فيمكن أن يكون المسار التلقيدي للانحدار الاشتقاقي هو :



ومشكلة هذا المسار ان خطواته طويلة و كثيرة , و أن التحرك الرأسي بيعمل ازعاج و نوع من تضييع الوقت و الجهد , لان المسار الرأسي لا يقوم بحل اي مشكلة او التقدم للقيم المثلي , فكأننا نريد الحفاظ علي التقدم الأفقي و ليس الرأسي

وهنا نقوم باستخدام فكرة الـ momentum والذي تم استقاؤه من فكرة المتوسطات الأسية

فإذا قمنا بإنشاء قيمة تسمي Vdw كذلك قيمة تسمي Vdb بحيث نطبق عليها مبدئ المتوسطات الأسية . .

بحيث يكون :

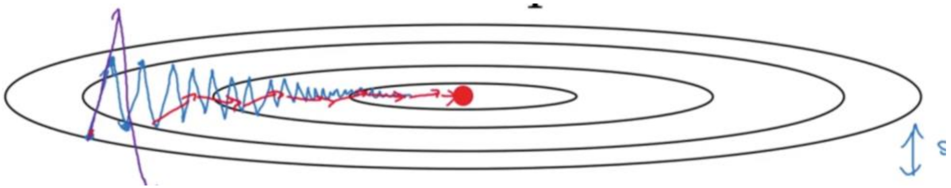
$$v_{dW} = \beta v_{dW} + (1-\beta)dW$$

$$v_{db} = \beta v_{db} + (1-\beta)db$$

وبالتالي نعمل update لقيمة w , b بدل من استخدام dw , dw نستخدم :

$$W = W - \alpha v_{dW}, b = b - \alpha v_{db}$$

و لأن تكنيك المتوسطات الأسية بيجعل الخطوة اهدي و اقل , ويجعل التغييرات بتمشي بشكل ابطئ شوية , فهذا هو السبب الرئيسي , الذي يجعل خطوات الانحدار ابطئ رأسيا , لكنها تتجه افقيا بشكل جيد , كما في الخط الأحمر



وبالتالي ببساطة , يتم استخدام القوانين

$$v_{dW} = \beta v_{dW} + (1-\beta)dW$$

$$v_{db} = \beta v_{db} + (1-\beta)db$$

وأحيانا بيتم ازالة قيمة (β-1) , بحيث تكون المعادلة :

$$v_{dW} = \beta v_{dW} + dW$$

و ايضا هذه الصيغة فعالة , لكن الأولي أفضل

وغالبا ما تكون قيمة بيتا بـ 0.9 , و غالبا ايضا لا يتم استخدام فكرة معامل الخطأ $\frac{V_t}{1-\beta^t}$. وذلك لأن القيم سرعان ما يتم تصحيحها سريعا

-

أحد الطرق الأخري المستخدمة , عوضا عن الانحدار الاشتقاقي , او ذو الزخم , هو ما يسمي الانحدار الاشتقاقي باستخدام الجذر التربيعي RMS Prop

والـ RMS اختصار root mean square

وتقوم فكرتها علي تعديل المعادلات , بإنشاء رمز جديد يسمي Sdw و Sdb والذان هما :

$$S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$$

$$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db$$

بحيث بيتا 2 , هي كمان قيمة لبيتا , لكن تم عمل 2 لمنع التداخل مع بيتا الأولي

ويتم عمل update لقيم w , b هكذا :

$$W := W - \alpha \frac{V_{dW}^{\text{corrected}}}{\sqrt{S_{dW}^{\text{corrected}}} + \epsilon}$$

$$b := b - \alpha \frac{V_{db}^{\text{corrected}}}{\sqrt{S_{db}^{\text{corrected}}} + \epsilon}$$

مع ملاحظة أنه تم وضع قيمة ابسلون مضافة إلي S في المعادلتين , لتجنب القسمة علي صفر , في حالة كانت قيمة S شديدة الضآلة

-

تحدثنا في الفقرتين السابقتين , عن انحدار الزخم , و انحدار الجذر التربيعي , ونقوم الان بتناول اسلوب (آدم للقيم المثلي) (Adam optimization algorithm) والذي يقوم بدمج الاسلوبين معا

هنتعامل بكلا من الـ V , S لكلا من w , b , وهيكون قيمهم الأولي بصفر , كدة :

$$\text{Initialize } V_{dW} = 0, S_{dW} = 0, V_{db} = 0, S_{db} = 0.$$

بعدها يتم استخدام نفس المعادلات السالف شرحها

$$V_{dW} = \beta_1 V_{dW} + (1 - \beta_1) dW$$

$$V_{db} = \beta_1 V_{db} + (1 - \beta_1) db$$

$$S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$$

$$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db$$

بعدها يتم حساب ما يسمي V و S قيمة correct

$$V_{dW}^{\text{corrected}} = V_{dW} / (1 - \beta_1^t)$$

$$V_{db}^{\text{corrected}} = V_{db} / (1 - \beta_1^t)$$

$$S_{dW}^{\text{corrected}} = S_{dW} / (1 - \beta_2^t)$$

$$S_{db}^{\text{corrected}} = S_{db} / (1 - \beta_2^t)$$

ومنها يتم عمل update لقيم w , b

$$W := W - \alpha \frac{V_{dW}^{\text{corrected}}}{\sqrt{S_{dW}^{\text{corrected}}} + \epsilon}$$

$$b := b - \alpha \frac{V_{db}^{\text{corrected}}}{\sqrt{S_{db}^{\text{corrected}}} + \epsilon}$$

وستري أن هذه الصيغة , فعالة بقوة لعدد كبير من الخوارزميات المرتبطة بالشبكات العصبية

و نري هنا عددا من الـ hyper parameters والواجب ضبطها جيدا قبل العمل

و غالبا ما نقوم بضبطها بهذه القيم :

$$\alpha: \text{ needs to be tuned}$$

$$\beta_1: 0.9 \leftarrow (dW)$$

$$\beta_2: 0.999 \leftarrow (dW^2)$$

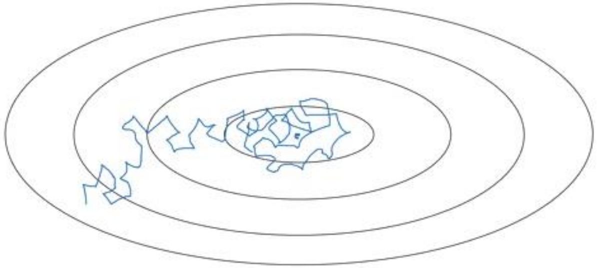
$$\epsilon: 10^{-8}$$

-

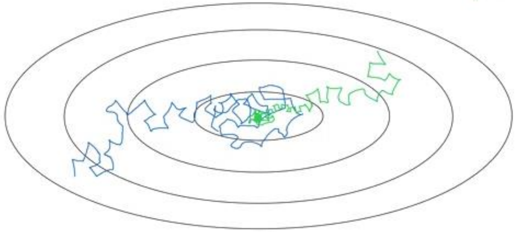
ننتقل لنقطة أخري في ذات الموضوع , وهي المتعلقة بمعامل التدريب learning rate المسمي الفا

لو تم تحديد الفا برقم كبير فستقفز عملية الـ optimization قفزات واسعة و لن يصل للحل أبدا

وإذا اخترنا قيمة متوسطة , فسنري أنه لن يصل بسهولة , وسيظل الخوارزم يدور حول نفسه حينما يقترب



لكن إذا قمنا بعمل تقليل تدريجي لقيمة الفا , سنري أن الخوارزم يسصل الي ارقام قريبة جدا من المطلوب , مثل الخط الأخضر:



فعلينا أن نقوم بعمل تقليل تدريجي لقيمة ألفا , ويكون التقليل مرتبط بعدد المحاولات , ومن الصيغ المستخدمة لهذا هي :

$$\alpha = \frac{1}{1 + \text{decay-rate} * \text{epoch-num}} * \alpha_0$$

حيث الفا 0 , هي القيمة الاولي لالفا , و الـ decay rate رقم نقوم بتحديده و ليكون 1 , و الـ epoch number هو عدد المحاولات

و بالتالي كلما زادت المحاولات , زاد المقام , فقلت الفا النهائية , فمثلا لو قلنا أن الفا 0 تساوي 0.2 , فيكون الجدول كالتالي :

Epoch	Alpha
0	0.2
1	0.1
2	0.06
3	0.05
4	0.04
5	0.03

ويمكن تغيير هذه الارقام عبر التحكم في و الـ decay rate

كما أن هناك صيغ أخري لتقليل الفا , مثل الصيغة الأسية :

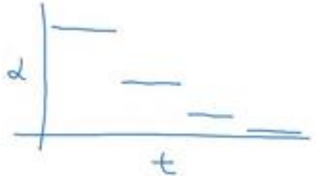
$$\alpha = 0.95^{\text{epoch-num}} \cdot \alpha_0$$

ففي حين تزيد عدد المحاولات , يزداد أس الرق المكتوب و ليكن 0.95 , فيقل قيمته , فتقل الفا

أو الصيغة الجذرية

$$\alpha = \frac{k}{\sqrt{epoch\text{-}num}} \cdot \alpha_0$$

أو الصيغة السُلمية



والتي فيها نحدد رقم معين لالفا لرينج عدد محاولات و يقل بالتتابع

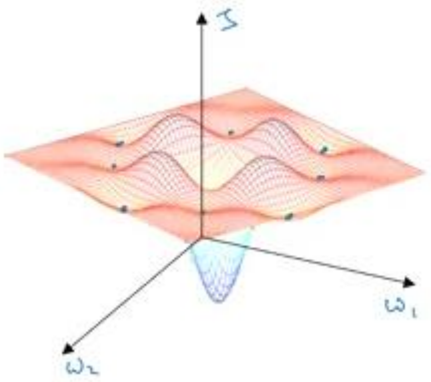
أو الطريقة اليدوية , وهي التي نغير فيها الفا بأنفسنا , وهي تصلح غالبا حينما يكون عدد البيانات هائل , فتستغرق المرة الواحدة عدد من الساعات , ففي خلال هذه الساعات و مع مراقبة المعالجة , اتخذ قرار بتقليل او زيادة الفا

-

أخيرا , سنتكلم عن مشكلة لن تحدث , ومشكلة أخري قد تحدث

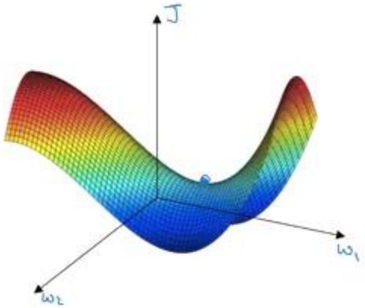
المشكلة التي لن تحدث هي الوقوع في فخ local optima

والمقصود بها , هو القيم الدنيا المحلية , في هين هناك قيم دنيا عظمي مثل هذا :



فكما هو واضح , هناك عددا من النقاط الزرقاء وهي دنيا محلية , بينما القاع به الدنيا العظمي , وهذه المشكلة تحدث فقط في المسائل البسيطة ذات البعدين , وهي شبه مستحيل حدوثها مع الـ NN

والواقع ان التصميمات الخاصة بالشبكات العميقة , تكون اقرب لهذا . .



والتي تحمل عددا كبيرا من الأبعاد , قد تصل لعشرات الآلاف , وبالتالي لن يتواجد قيمة محلية , ولكن ما يسمي الـ saddle point او نقطة السرج

فكما نري في الشكل , لا يوجد امكانية لوجود قيمة محلية في الاساس , والاشتقاقات دائما توصلنا للقيمة الدنيا العظمي . .

و هي تسمي نقطة السرج لان الشكل نفسه يشبه سرج الحصان

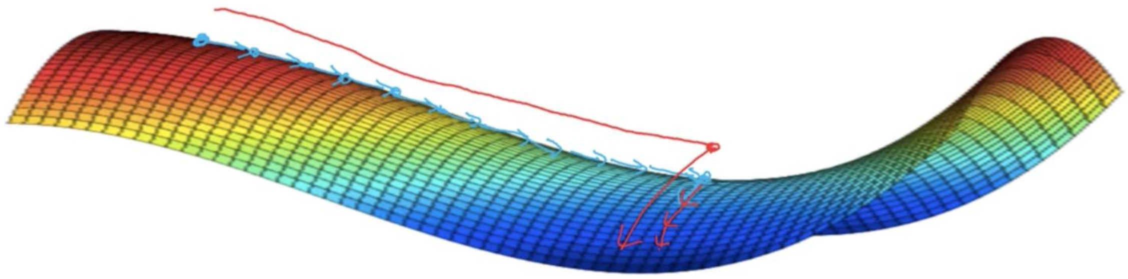
لكن المشكلة التي قد تحدث , هي ما يسمي مشكلة التسطح plateaus problem

والتي يكون خلالها السطح اساسا شبه افقي بالكامل , مما يعني ان الاشتقاقات اقرب للصفر , وبالتالي معدل التغيير يكون بطئ للغاية

فالانتقال من النقطة اليسري لليمني سيستغرق وقتا رهيبا , لتسطح المستوي

وهذه المشكلة قد تحل بالتكنيكات السالف ذكرها في ايجاد الانحدارات الاشتقاقية

Problem of plateaus



-

نهاية الأسبوع الثاني