

# التعلم العميق Deep Learning

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

## درس 2: تطوير الشبكات العميقة : المعاملات العليا

## الأسبوع الأول : السمات العملية للتعلم العميق

- عقب الانتهاء من هذا الكورس , ستكون قادر علي :

- التعرف علي الانواع المختلفة من الارقام المبدئية (المعاملات المستخدمة) , وكيف ترددي لنتائج مختلفة
- التعرف علي اهمية المعاملات , في الشبكات المعقدة
- التعرف علي الفارق بين عينة التدريب , التطوير و الاختبار
- التعامل مع معامل الخطأ bias للنموذج
- التعرف علي كيفية عمل تنعيم للبيانات
- فهم نماذج تطبيقية في التعلم العميق , ومعرفة كيفية التعامل معها
- استخدام الفحص التدريجي , للتأكد من صحة العملية في المسار الخلفي

\* \* \* \* \*

بعد ما تعرفنا علي كيفية بناء و تدريب شبكة عصبية , هنتعرف الان علي كيفية ضبط معاملاتها , زي المعاملات العليا , والتأكد إن الشبكة شغالة كويس في وقت مناسب

هنتكلم في البداية عن ما يسمى ال randomization , او القيم العشوائية

كمان عايزين نتعرف علي ما الفرق بين عينة التدريب training , التطوير development و الاختبار test , واللي تحديدهم ببساطة بقوة علي زيادة كفاءة الشبكة العصبية , و جعل ادائها افضل

و من العوامل المهمة الواجب تحديدها عند بناء اي شبكة عصبية :

- عدد الطبقات الخفية
- عدد الوحدات في الطبقة الخفية
- معامل التعلم learning rate
- معادلة الاكتيفاشن (سيجمويد او تاننش او غيرها)

لاحظ ان قيم  $w$  ,  $b$  مش من العوامل الواجب تحديدها , لانها اصلا بتبدا عشوائي و المطلوب ايجاد قيم مثالية ليها

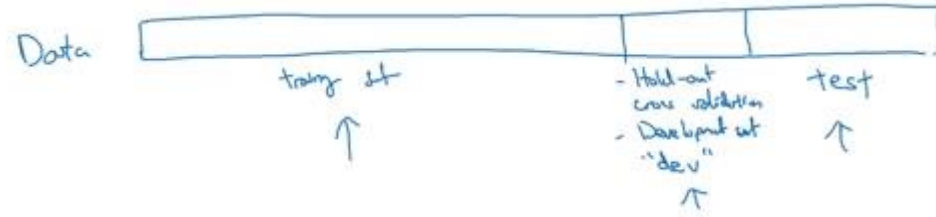
ولان مفيش قواعد ثابتة تحدد علي اي اساس يتم اختيار هذه القيم , فالفكرة تعتمد بالاساس علي التجريب , واختيار القيم الافضل اللي تتناسب معاها

والشئ المهم المتعلق بالنقطة ديه , ان تطبيقات تعليم الآلة كثيرة , ولو انت شغال في تطبيق معين بمعايير معينة , وبارقام معينة في النقط السابقة هنا , فمن الممكن الا يكون مناسب استخدام نفس المعايير و القيم في تطبيق اخر لتعليم الآلة , وهنا تأتي أهمية التجريب و تحديد نسبة الخطأ

و النقطة التي سنبدأ بها اليوم هي تقسيم عينة البيانات , إلي : تدريب , تطوير , اختبار

والأصل اننا بنقسم العينة عندنا لتلات اقسام هي :

- عينة التدريب : والتي يتم تحديد منها المعاملات  $w$  ,  $b$
- عينة التطوير : والتي فيها يتم تجريب النماذج المختلفة للخوارزم , باستخدام المعاملات  $w$  ,  $b$  , ولكن نبدا نجرب المعاملات العليا , او النقاط الاربعة السالف ذكرها , لرؤية اي منها بالتحديد سيعطي نتائج ايجابية افضل
- عينة الاختبار : والتي يتم استخدامها لتجريب الخوارزم بقيم المعاملات  $w$  ,  $b$  و المعاملات العليا السابق اختيارها , لتحديد قيمة كفاءة الخوارزم



وتحديد النسب الخاصة بكل قسم فيهم يختلف , ففي الكميات المعقولة للبيانات (عدد عناصر العينة اقل من 10 الاف) , فتكون غالبا النسبة :

- عينة التدريب : 60 %
- عينة التطوير : 20 %
- عينة التدريب : 20 %

ولكن مع الكميات الهائلة من عناصر البيانات التي تتجاوز مئات الملايين , فالنسبة ستختلف

لان غالبا الكمية المطلوبة لعينة التطوير او عينة الاختبار لا يجب ان تكون كثيرة جدا , فمثلا 10 الاف عنصر كافي , خاصة انه لدي العدد الكبير من العناصر , فنحن بحاجة لأكبر كمية من البيانات في التدريب , حتي يتمكن الخوارزم من الالمام بجميع العناصر المتواجدة

وبالتالي اذا كان لدي مليون عنصر , فيمكن تحديد 10 الاف عنصر لعينة التطوير , و 10 الاف عنصر لعينة الاختبار , ويكون الباقي وهو 980 الف عنصر لعينة التدريب , وبالتالي تكون النسب :

- عينة التدريب : 99 %
- عينة التطوير : 1 %
- عينة التدريب : 1 %

و احيانا تكون نسبة عينة التدريب اكثر من هذا

و لا تنس أن تقم بمزج و توزيع العينة جيدا قبل تقسيمها . .

فلو كانت لدينا عينات من مدن مختلفة من احد البلاد , فلا تقم بتحديد مدينة 1 و 2 و 3 لعينة التدريب , ومدينة 4 للتطوير , ومدينة 5 للتدريب

إذ أن هذا الأمر سيجعل الخوارزم يتدرب فقط علي العينة المتعلقة بالمدن المعطاة , وبالطبع لن تكون مناسبة ولا فعالة مع عينات اخري

لكن عليك ان تقوم بخلط العينات خلطا جيدا , وانتقاء النسب المطلوبة بشكل عشوائي لكل قسم من اقسام العينات

و احيانا ويتم الاستغناء عن عينة الاختبار , والاكتفاء بعمل الاختبار علي عينة التطوير , باعتبار ان تشغيل الخوارزم علي عينة التطوير سيقوم بالفعل بتقييم الخوارزم و تحديد كفاءته

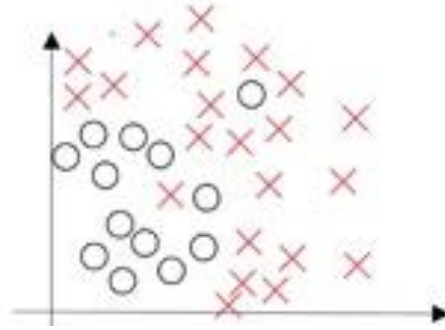
\* \* \* \* \*

نبدأ بالتكلم الان عن مصطلحين هامين , و يقتربان ببعضها دائما , ويسهل فهمهما , بينما يصعب التعامل معهما هما : الانحراف Bias و التنوع Variance

و منهما نشأ مصطلح هام هو (Bias-Variance trade off) , وكلمة trade off معناها التنازل عن شئ مقابل الحصول علي شئ آخر

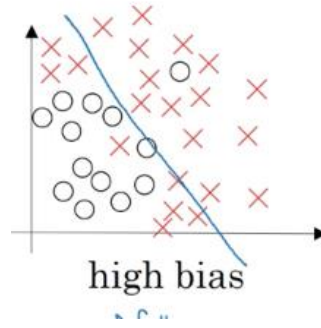
فلأن الـ bias , variance قرينان و ضدان في نفس الوقت , فكان لزاما علينا تعلمهما جيدا , والتعرف علي كيفية التعامل معهما , وفن المقارنة بين مميزات و عيوب كلا منهما

لو نظرنا إلي المثال التالي :

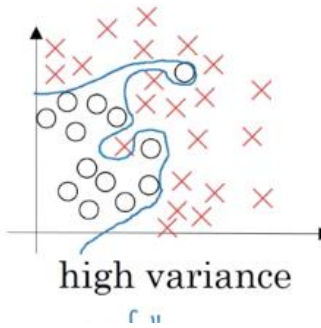


لدينا عدد من النقاط الحمراء و الزرقاء , ونريد عمل classifier مناسب لها

فيمكن عمل كلاسيفاير حاد , معادلة من الدرجة الاولى مثلا هنا :

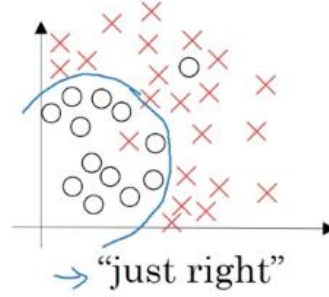


او يمكن عمل شبكة عصبية عميقة للغاية , تكون قادرة علي الالمام بجميع النقاط مثل هنا :



او يمكن عمل خوارزم مناسب , يقوم بتجميع اغلب النقاط مثل هذا





ومن الواضح أن المثال الأول يعاني من الـ  $UF$  كما أن نسبة الانحراف فيه كبيرة, بينما التنوع قليل , لذا يسمى  $High\ bias$

المثال الثاني يعاني من الـ  $OF$  , ونسبة التنوع فيه كبيرة , بينما نسبة الانحراف قليلة , لذا يسمى  $High\ variance$

المثال الثالث جيد , به نسبة من الانحراف و التنوع معا , دون الوقوع في مشكلة  $UF$  او  $OF$  , لذا يسمى  $just\ right$

وللتعامل مع رقمي الانحراف والتنوع , علينا ان ننظر لرقمين هامين مرتبطان بهما , وهما : نسبة الخطأ في عينة التدريب , ونسبة الخطأ في عينة التطوير (او الاختبار)

والمقصود بنسبة الخطأ معادلة الخطأ التي نريد الحصول عليها  $cost\ function$

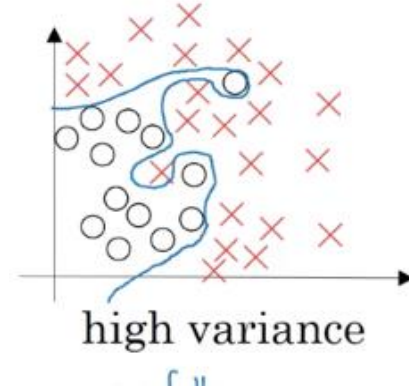
فلو تخيلنا ان لدينا كلاسيفير يقوم بالتمييز بين عنصرين محددين

فسيكون لدينا عدد من الحالات المتوقعة بالنسبة للارقام :

**الحالة الأولى :** لو وجدنا ان نسبة الخطأ في عينة التدريب كانت فقط 1 % , بينما في عينة الاختبار او التطوير كانت 10 % , فهذا معناه ان الخوارزم تم ضبطه جيدا علي عينة التدريب بلا اخطاء تقريبا , بينما يفشل في تقييم عناصر جديدة (عينة الاختبار)

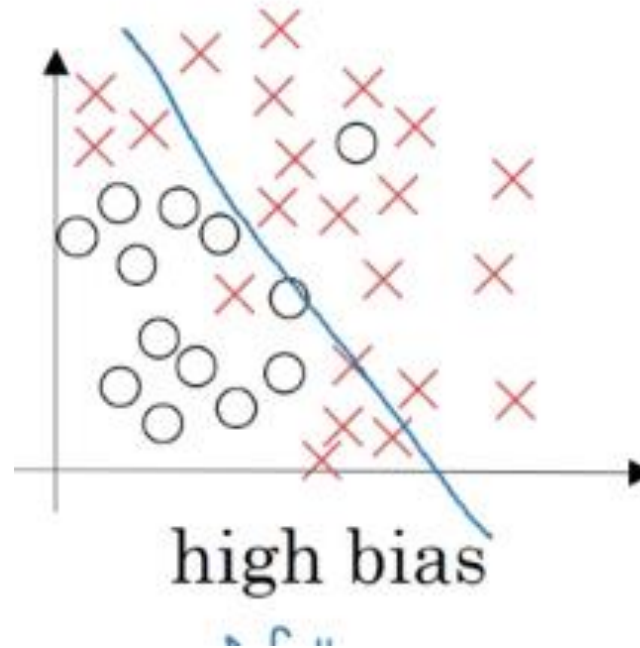
و هذا معناه ان نسبة الانحراف قليلة (الانحراف في عينة التدريب) , بينما التنوع كبير High Variance , اي أنها حالة OF

وهي تتمثل بالصورة :



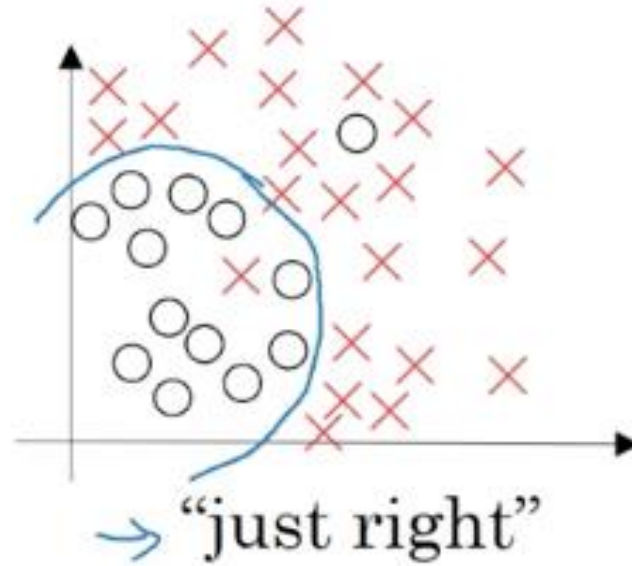
**الحالة الثانية :** لو وجدنا ان كلا من قيمة خطأ عينة التدريب و الاختبار كبيرة , ومتقاربة من بعض , يعني مثلاً 15 او 20 % , فهذا يدل علي ان الخوارزم اساساً هو  $UF$  , وان نسبة الانحراف فيه كبيرة وهو ما أثر علي كلا من العينتين , وفي الحالة ديه بنسبها **High Bias**

وهي تتمثل بالصورة :



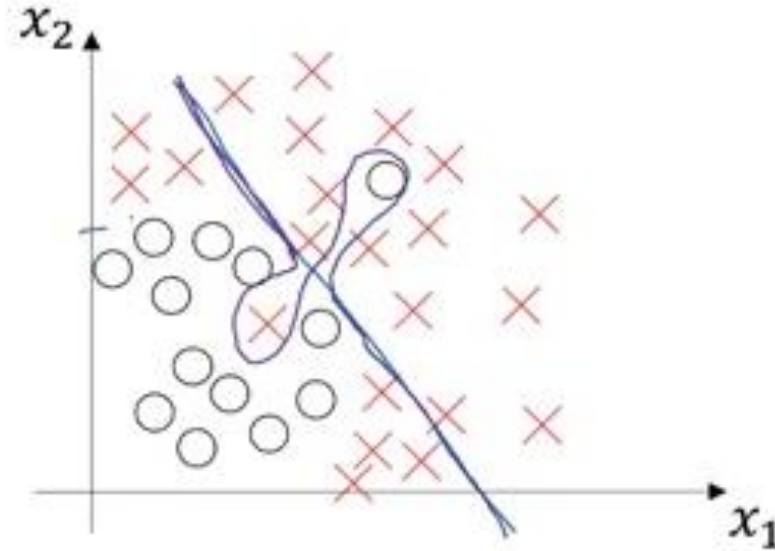
**الحالة الثالثة :** في حالة وجود رقم قليل خطأ عينة التدريب , ورقم اعلي منه قليل (لكن لازال قليل) خطأ عينة التدريب , فهذه هي الحالة المثالية , و تسمى  
Low Variance , Low Bias

وهي تتمثل بالصورة :



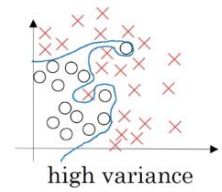
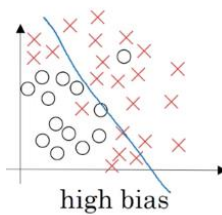
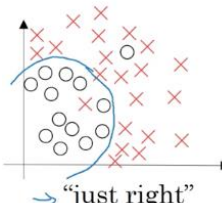
**الحالة الرابعة :** في حالة وجدنا ان خطأ عينة التدريب كان رقم متوسط (لكن لازال غير مقبول) مثلا 15 % , و خطأ عينة التدريب اكبر منه , مثلا 30 % , فهذا معناه ان الخوارزم يجمع بين المشكلتين معا , اي أن به High Bias , High Variance معا

ويمكن التمثيل لها بالصورة :



ستجد أن الصورة بها مزيج من الانحراف العالي (لوجود اخطاء عديدة ) والتنوع العالي (لوجود تغير كبير في الاماكن) معا .

يعني ممكن نلخصها كالتالي :

Tr. Set	Ts. Set	Description	Photo
Low	High	High Variance OF	 high variance
High	High	High Bias UF	 high bias
Low	Low	Right Fit	 → "just right"



سنتكلم حالا عن الخطوات الواجب اتباعها للوصول للشكل المثالي في أمر الـ Bias , Variance و تجنب النماذج السلبية السالف ذكرها

#### في حالة الانحراف الكبير High Bias :

- حاول أن تزيد من حجم الشبكة الخاصة بك (طبقات أكثر أو خلايا أكثر)
- ممكن ان تزيد من عدد المحاولات (No of Iterations)
- استخدام خوارزم مختلف عن المستخدم حاليا
- احيانا تغيير هيكل الشبكة نفسه architecture يساعد في حل المشكلة

#### في حالة التنوع الكبير High Variance :

- زيادة عدد عناصر العينة  $m$
- عمل تنعيم البيانات Regularization
- تغيير هيكل الشبكة نفسه architecture

وعلي العامل في مجال تعليم الالة , معرفة سبب المشكلة بالتحديد للوصول لآلية الحل بشكل مباشر

نصل للمصطلح السالف ذكره : (Bias-Variance trade off) , وكلمة trade off معناها التنازل عن شئ مقابل الحصول علي شئ آخر

فمن الواضح أن كلا من الانحراف و التنوع مرفوض , ونريد التخلص منهما بقدر الامكان , وفي الماضي لم يكن من السهل التخلص من كلاهما معا , فأغلب الحلول تقوم بتقليل أحدهما و زيادة الآخر رغما عني



لذا كان استخدام هذا المصطلح , و الذي يشير لاحتمية الوصول لمساحة مشتركة بين كمية الانحراف و التنوع المقبولين معا , بدلا من التلخص من واحد و المعاناة من الآخر بشكل كبير

لكن و حاليا , وفي عصر التعلم العميق , و البيانات الضخمة , والوصول لعدد ضخم من الطبقات الخفية للشبكة العصبية , صار من السهل حل المشكلتين معا فالشبكة العصبية الضخمة تقوم بحل مشكلة الانحراف دون زيادة التنوع (بشرط تنعيم البيانات قبل استخدامها), وكمية البيانات الكبيرة تحل مشكلة التنوع دون المساس بالانحراف

فصار حاليا من المتاح حل المشكلتين معا , دون الاضطرار للتضحية بمشكلة جديدة

\*\_\*\*

## عملية تنعيم البيانات Regularization

وهي من اهم خطوات معالجة مشكلة الـ OF الناشئة من التنوع الكبير High Variance

صحيح ان من الحلول الاخرى هو زيادة عدد البيانات  $m$  لكن هذا الحل قد لا يكون متاحا دائما , او قد يكون باهظ التكلفة ولنفهم التنعيم , علينا أن نتذكر معادلة الخطأ :

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) .$$

هناك نوعين من التنعيم  $L1$  ,  $L2$

النوع الأول ( $L1$ ) كان قديما , وكان بإضافة القيمة التالية :

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{m} \|w\|_1$$

وهي بضرب لمبدأ في نورم الـ  $w$  , مقسوم علي الـ  $m$  , والذي يتم بسهولة عبر امر sum لجمع مصفوفة  $w$

لكن مشكلته الاساسية , أن هذا الخوارزم يجعل اغلب قيم  $w$  باصفار , وهو معناه الاستغناء عن inputs كثيرة , هذا يجعل الشبكة اسهل و اسرع , لكنها ليست ادق

النوع الثاني عبر إضافة القيمة التالية :

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \|w\|_2^2$$

وهي بضرب لمبدأ في نورم الـ  $w$  مربع , مقسوم علي ضعف الـ  $m$

و قيمة النورم هي مجموع مربعات قيم دبليو , وتكون بالمعادلة :

$$\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$$

أي أنها ببساطة حاصل ضرب تدوير مصفوفة  $w$  في المصفوفة نفسها بلا تدوير , وهذا سيقوم بضرب كل قيمة من قيم  $w$  في نفسها , وجمعها

والنوع ده يسمى غالبا النورم الفروبينيوس Frobenius norm ويرمز ليه بالرمز  $\|w^{[l]}\|_F^2$  بدل الرمز  $\|w\|_2^2$  و ده لسبب رياضي نحن في حل من ذكره

فيكون الشكل النهائي لمعادلة الخطأ :

$$\frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^L \|w^{[l]}\|_F^2$$

حيث النورم الفروبينيوس :

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} (w_{ij}^{[l]})^2$$

لا تنسي أنها نفسها الصيغة السابقة مع اختلاف الشكل فقط

ولاحظ ان التنعيم بيتم فقط لمصفوفة  $w$  , دون عمل تنعيم لـ  $b$  , والسبب ان  $b$  هي قيمة واحدة , وحتى الحصول علي قيمة مثالية لها لن يؤثر كثيرا في المسألة , بينما  $w$  هي مصفوفة من عدد كبير من الارقام يهمننا الحصول علي الارقام المناسبة لها

و قيمة لمبدا بيتم تحديدها في خطوة عينة التطوير , والتي تفصل التدريب عن الاختبار , حيث يتم تجريب اكثر من رقم و اختيار الافضل

ولاحظ ان كلمة  $\lambda$  هي كلمة يستخدمها برنامج بايثون في شئ آخر , فعليك ان تستخدم كلمة اخري مثل  $\text{lambd}$  او تغير من الكابيتال و السمول حتي لا تصطدم مع الكلمة المعروفة في بايثون

نأتي لنقطة مهمة , وهي كيفية حساب قيمة  $dw$  الان , وعمل update لقيمة  $w$

قبل استخدام التنعيم , كنا نحسب  $dw$  بالقانون :

$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L]T}$$

وبعدها كنا نقوم بعمل update لقيمة  $w$  هكذا :

$$w := w - \alpha dw$$

بضرب قيمة الاشتقاق في الفا , وطرحه من قيمة  $w$  السابقة

لكن مع عمل التنعيم فإن قيمة اشتقاق  $dw$  نفسها ستتغير و ستكون نفس القيمة القديمة , مضاف إليها حاصل ضرب لمبدا في  $w$  مقسوم علي  $m$  اي انها :

$$[(\text{From Backprop}) + \frac{\lambda}{m} W^{[l]}]$$

وبالتالي لدي طرحها من  $w$  ستكون :

$$W^{[l]} = W^{[l]} - \alpha [(\text{From Backprop}) + \frac{\lambda}{m} W^{[l]}]$$

وبضرب الفا في الاقواس :

$$W^{[l]} = W^{[l]} - \frac{\alpha \lambda}{m} W^{[l]} - \alpha (\text{From Backprop})$$

وبالتالي القيمة الاولى هنا  $W^{[l]} - \frac{\alpha\lambda}{m} W^{[l]}$  تعتبر تساوي القيمة ديه , من اجل الكود

$$(1 - \frac{\alpha\lambda}{m})W^{[l]} :$$

\* \* \* \* \*

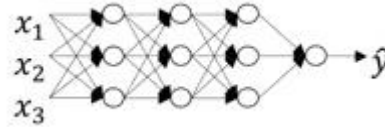
ويأتي سؤال هام , كيف يقوم التنعيم بتقليل الـ Variance , وبالتالي الـ OF

بالبحث في معادلة الخطأ هنا (سواء L1 , L2)

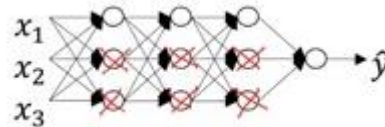
$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} ||w||_2^2$$

فحينما يتم تحديد لمبدأ بقيمة عالية , تجد ان الخوارزم يجبر قيم  $w$  (الثبتات) علي ان تقل قيمتها للغاية , و ان تقترب من الصفر , حتي ان عددا من الثبتات بالفعل يكون صفرا

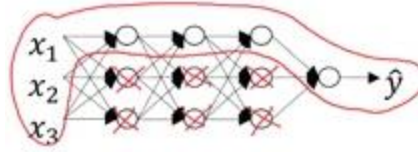
و بزالة او تقليل عدد من الثبتات , فهذا معناه الغاء عددا من الوحدات (الخلايا) الموجودة بالطبقات الخفية فتتحول الشبكة هذه :



إلي هذه :

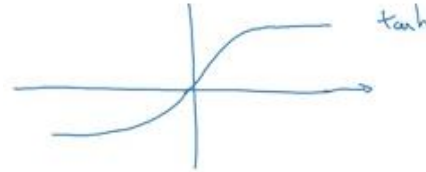


ويكون شكلها النهائي كأنه :

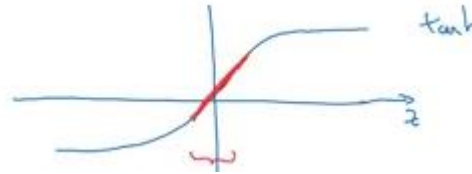


وهي التي تقترب من دوال التوقع ذات الدرجة الاول , والتي يتم تقليل الـ OF فيها بشكل كبير

ولو انت لاحظت الرسم الخاص بدالة التانش سيكون هكذا :



وفي حالة كانت قيم المدخلات قليلة , فستري أن المساحة التي سيتم التعامل معها ستكون هنا :

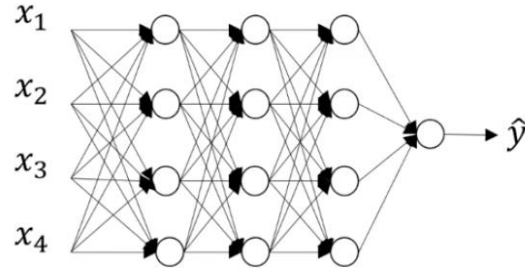


وهذا الجزء يتشابه كثيرا مع الدالة الخطية وبالتالي حينما تقل قيم  $w$  فستكون قيم  $z$  قليلة بالتبعية , و التي ستجعل قيم  $a$  قليلة لانها تانش الـ  $z$  القليلة , مما يؤدي لأن تكون اغلب دوال الشبكة خطية , وبالتالي يقلل من حدوث الـ variance الذي يسبب الـ OF



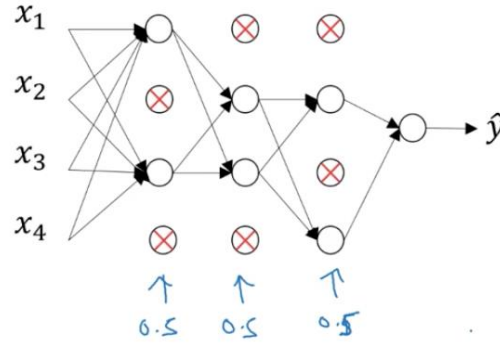
اسلوب آخر للتنعيم يسمى اسلوب الحذف dropout regularization

وهو اسلوب غريب نوعا لكنه يعمل بفعالية , و فكرته كالتالي , لو أن لدي شبكة مثل هذه :



ففكرة هذا النوع من التنعيم , يعتمد علي حذف عدد من الخلايا في الطبقات الخفية , بشكل عشوائي , في كل مرة نقوم بعمل مسار أمامي و خلفي

فلو أن لدي عدد 500 عنصر من عناصر العينة , فعندما أقوم بتنفيذ المسار الأمامي للعنصر الأول , اقوم بعمل حذف لنصف خلايا كل طبقة من الطبقات الخفية (قد تزيد او تقل النسبة عن النصف) , و يكون اختيار الخلايا المحذوفة عشوائية وبالتالي تتحول الشبكة لهذا الشكل :



وبعدها اقوم بعمل المسار الخلفي لنفس الشكل , دون تغيير , وأقوم بعمل update لقيم  $w, b$

ثم اذهب للعنصر الثاني من عناصر العينة , هنا اقوم بعمل حذف عشوائي اخر لنصف خلايا الشبكة , وقد تكون بعض الخلايا المحذوفة قد حذفت في الدورة السابقة , وقد لا تكون , الأمر عشوائي , ثم اقوم بعمل المسار الخلفي لنفس الشكل , وعمل  $w, b$  update , و اقوم بعمل تكرار لنفس العملية لدي كل عنصر من عناصر العينة , كل مرة يتم عمل حذف عشوائي للبعض , كمسار امامي و خلفي , ويتم تغيير الخلايا المحذوفة كل مرة

وبالرغم من انها طريقة غريبة , لكن هي فعالة و بتعمل تقليل بالفعل للـ variance

طيب كيف يتم هذا بالكود :

ببساطة يمكن استخدام عددا من السطور البسيطة في بايثون لتفعيل الأمر :

ففي البداية لدينا مصفوفة  $a$  وهي قيم الـ activation الخاصة بالطبقة , ولنقل أن أبعادها هي 4 في 3 (يتم تحديد الأبعاد بناء على عدد خلايا الطبقة الحالية و السابقة)

ونقوم بتحديد النسبة المراد حذفها من الخلايا , ولنقل أننا نريد إبقاء 60 % , وحذف 40 % من الخلايا , فنستخدم الرقم هنا في صياغة المصفوفة الجديدة  $d$  :

```
import numpy as np
```

```
d = np.random.rand(4,3) < 0.6
```

وهي معناها ان المصفوفة  $d$  هي مصفوفة ارقام عشوائية 4 صفوف في 3 اعمدة

لكن إضافة العملية الحسابية ( $0.6 <$ ) سيجعل المصفوفة بدلا من أن تكون ارقام , ستكون بوليان , بالتالي ستتحوّل لمصفوفة بنفس الأبعاد , لكن كلها  $true$  ,  $false$  مثل هذا

```
[[ True True False]
 [False True True]
 [ True True True]
 [False True True]]
```

هنا نقوم بضربها في مصفوفة  $a$  التي بنفس الأبعاد , ولاحظ أن مصفوفة  $a$  هي ارقام , بينما مصفوفة  $d$  هي بولياني , فورا سيقوم بايثون بتحويل مصفوفة  $d$  من بولياني , لأرقام اصفار و واحد , وبالتالي ستكون

```
[[ 1 1 0]
 [0 1 1]
 [ 1 1 1]
 [0 1 1]]
```

وبالتالي حينما يتم ضربها في مصفوفة  $a$  ستختفي بعض الارقام منها , والتي تعني اختفاء الخلية هكذا :

```
c = np.multiply(a , d)
```

وبعدها نأتي لخطوة مهمة , وهي قسمة مصفوفة  $a$  علي الرقم الذي تم اختياره لابقاء الخلايا , أي قسمة  $a$  علي 0.6

وهذا لأن عددا من خلايا  $a$  تم إزالتها , فستجد أن قيمة  $z$  النهائية قلت كثيرا , ولا نريد هذا , وبالتالي يتم قسمة قيم  $a$  علي 0.6 , والقسمة علي رقم اقل من 1 تجعل القيم نفسها تزيد , وبالتالي قيم  $a$  ستكون اعلي قليلا , فستعوض غياب عددا من الخلايا

لاحظ ان حينما استخدم ( $0.6 <$ ) فهذا ليس معناه أن بالتحديد 60 % من القيم ستكون true لاننا نعتمد علي وجود ارقام عشوائية لا نعرفها , لكن ستكون قريبة جدا من هذه النسبة

والتكنيك الغريب ده اسمه الحذف العكسي Inverted Dropout

و لاحظ شئ هام , ان تكنيك حذف الخلايا , يتم فقط في مرحلة التدريب , وليس الاختبار , فنحن نريد تعيين قيم لـ  $w$  تكون قادرة علي تخطي الـ OF , لكن لدي التعامل مع عينة الاختبار علينا ان نتعامل مع كل قيم  $w$  دون استثناء

طب كيف يقوم تكنيك "حذف الخلايا Dropout" بعمل تنعيم للبيانات . .

يتم هذا عبر عدة تأثيرات

أولا ان في كل مرة يتم تطبيق المسار الامامي او الخلفي , يكون عدد الخلايا المستخدم اقل , وهو ما يعني شبكة ابسط , وبالتالي تقل فرص حدوث الـ OF من ناحية أخرى , فإنه عندما يتم الغاء عدد من الخلايا , فهذا معناه ان كل خلية ستتلقى بيانات من عدد اقل من الخلايا السابقة لها , فمثلا هذه الخلية بدلا من أن تتلقى بيانات من اربع خلايا هنا :



ستكون كذلك

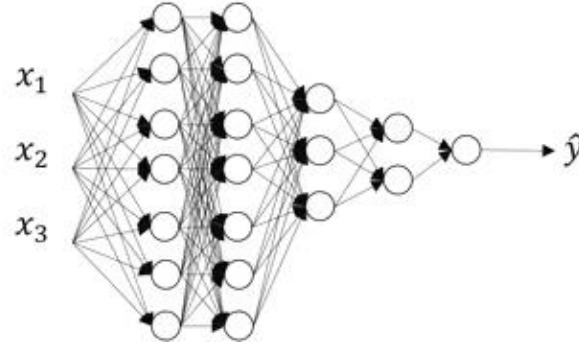


وهذا يجعل الخلية نفسها لا تقوم بالاعتماد علي مداخل inputs محددة بشكل اساسي , لان كل مدخل منها قد يختفي في مرة من المرات , وهو ما يجبر الخلية  
لدي تحديد الـ  $w$  في توزيع قيم الثيتات بشكل متوازن الي حد ما , ومراعاة جميع المداخل معا , مما يقلل من حدوث الـ OF

لاحظ أيضا ان كل طبقة يكون لها معامل أبقاء مختلف , أي أن لكل طبقة النسبة التي نحددها من الخلايا الواجب حذفها و الواجب الابقاء عليها

وغالبا ما يعتمد هذا علي عدد الخلايا في الطبقة و الخلايا في الطبقة السابقة لها , فكلما زاد العددين , كلما تعقدت أكثر مصفوفة الـ  $w$  , وهو ما يعني تعرضها  
أكثر للـ OF و بالتالي زيادة نسبة الحف و تقليل نسبة الابقاء

فشبكة مثل هذه :



طبقة المدخل 3 , و الطبقة الاولى و الثانية كلا منهما 7 , بينما الثالثة 3 , والرابعة اثنين و الخامسة (المخرج) واحدة

لا تنس أن ابعاد مصفوفة  $w$  تكون عدد خلايا الطبقة السابقة كصفوف , و خلايا الطبقة الحالية كأعمدة

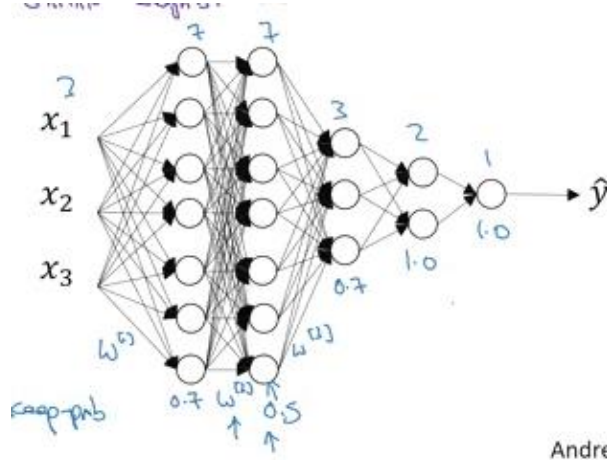
إذن ستكون المصفوفات كالتالي :

$(3 \times 7)$	$w_1$
$(7 \times 7)$	$w_2$
$(7 \times 3)$	$w_3$
$(3 \times 2)$	$w_4$
$(2 \times 1)$	$w_5$

والخوف يزداد لدي المصفوفات الاكبر , ويقل في الاصغر , فيمكن تحديد نسب للطبقات مثل هذه :

0.7	(3x7)	w1
0.5	(7x7)	w2
0.7	(7x3)	w3
1	(3x2)	w4
1	(2x1)	w5

مثل هذا



Andrew



فمن الواضح ان المصفوفات ذات العدد الأكبر , حظت بمعامل ابقاء أقل و معامل حذف أكبر , لان احتمال حدوث OF فيها أكبر من المصفوفات الأصغر

بالنسبة للمدخل inputs هل نطبق عليها معامل ابقاء ؟

نادرا ما يتم تطبيقه عليها , باعتبار اننا لا نريد اساسا التوضيحية او الاستغناء عن تاثير ايا من المدخلات , وان اضطررنا لتطبيقه , فيكون رقم بسيط , أي معامل ابقاء 0.9 مثلا

\* \* \* \* \*

تعرفنا علي نوعين من أنواع التنعيم , دعنا نتعرف علي عدد من الأنواع الأخرى وهي :

- تشويه البيانات Data Augmentation
- التوقف المبكر Early Stopping

نبدأ بالنوع الأول : تشويه البيانات . .

و يقصد به , أنها بما أن العلاج الأساسي للـ OF هو زيادة عدد البيانات  $m$  , وبما أن غالبا ما يقترن زيادة البيانات بالتكلفة العالية , فهناك طريقة مجانية لمضاعفة أعداد البيانات و بمجهود قليل

الطريق هي القيام بعمل تشويه و تغيير في البيانات نفسها, بأكثر من طريقة , وإضافتها إلي عينة التدريب , مما يجعل الخوارزم يتدرب علي المزيد من الاشكال المختلفة , ويقلل نسبة الـ OF

فمثلا لو كان لدينا كلاسيفير لصور القطط , فيمكن أخذ صورة قطعة مثل هذه :



و نقوم بعمل انعكاس افقي للصورة حتي تكون هكذا



مما يجعل الخوارزم يتدرب علي صورة بهذا الاتجاه

كذلك يمكن عمل نوع من الـ crop او الزووم او التدوير للصورة هكذا



وهو الامر الذي يجعل عدد البيانات الموجودة يتضاعف عدة اضعاف , و هذا الامر لن يكلف اموالا او مجهودا يذكر

بالطبع فإن اضافة بيانات إضافية حقيقية هو افضل من هذا الامر , لكن البيانات الإضافية تعني المزيد من التكلفة كذلك لو كان لدينا خوارزم يقوم بقراءة الحروف , فيمكن عينة التدريب ان اقوم بتشويه الحرف هكذا

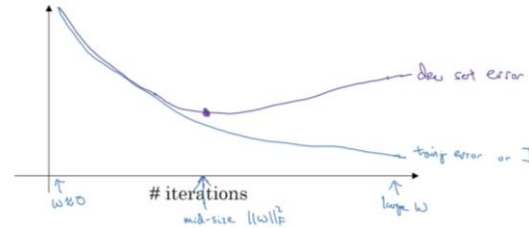


## Early Stopping المسمى : التوقف المبكر

يقصد به أنه حينما نزيد عدد المحاولات iterations في التدريب , فإن قيمة  $J$  تقل , لكن احتمالية الحصول علي OF تزيد , وبالتالي قد يكون هناك نقطة ما , يجب التوقف عندها في التدريب , للحصول علي قيمة  $J$  قليلة , دون زيادة الـ OF

والشئ الذي يربط القيمتين معا هو قيمة الـ  $J$  في عينة التدريب , وفي عينة الاختبار (او التطوير)

فيمكن القيام بعمل رسم بياني , يوضح قيمة  $J$  في العينتين معا :



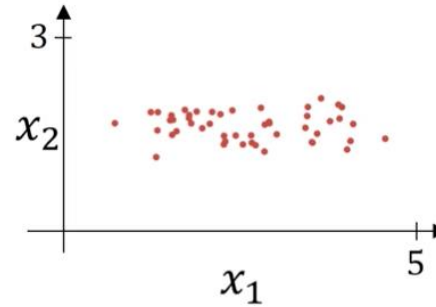
وتري فورا أنه منذ البداية و حتي نقطة ما , فإن قيمة  $J$  لكلا من العينتين تقل , لكن عند هذه النقطة , تزداد قيمة  $J$  لعينة الاختبار , وتقل في عينة التدريب , ولا تنس أن الذي يهمنا هو عينة الاختبار لذا فإننا نتبع خطوتها

فيمكننا أن نتوقف مبكرا في تلك النقطة , قبل ان تزداد قيمة  $J$  لعينة الاختبار

## Data Normalizing هنتعرف علي تكتيك : تسوية البيانات

المقصود بها هو إزالة هذا التنوع الكبير في البيانات , وتحديد الحد الاقصى و الادني له كارقام صغيرة معرفة

فلو ان لدينا بيانات في  $x_1$  و  $x_2$  هكذا



فالخطوة الأولى من عمل التسوية , هو ان نجعل متوسط قيم  $x$  هو الصفر , , وهذا يتم كالتالي :

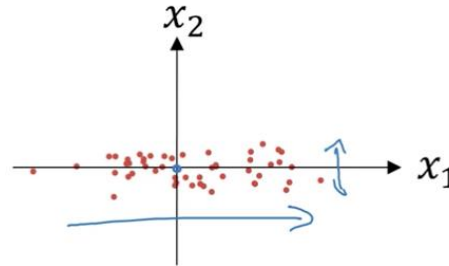
أولا تحديد قيمة المتوسط الحالي , عبر جمع قيم اكسات و قسمتها علي عددها :

$$\mu = \frac{1}{n} \sum_{i=1}^n x^{(i)}$$

ثم طرح كل قيم اكس من هذا المتوسط الحالي mue

$$x := x - \mu$$

فلو ان هذه هي درجات عدد من الطلاب و الحد الاقصى لها هو 100 و الحد الادني هو 50, وتم حساب متوسطها أنه 70 , فعقب الطرح , سيكون الحد الاقصى 30 , والادني -20 و سيكون متوسطها صفرا , وستكون البيانات هكذا



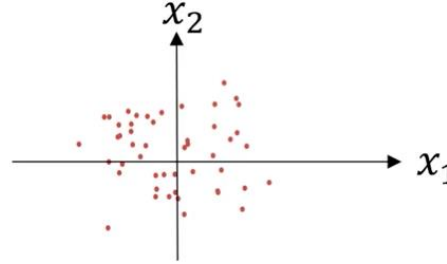
الخطوة التالية هو عمل تسوية للـ variance , و ذلك عبر حساب قيمة مجموع مربعات بعد قسمها علي عددهم :

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n x_i^2$$

ومن ثم قسمة كل عناصر اكس علي هذا الرقم (سيجما تربيع)

$$x / \sigma^2$$

وهو الذي سيحول البيانات لـ هكذا :

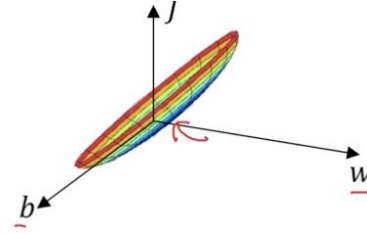


وبهذه الطريقة , قمنا بعمل تسوية للبيانات , وهو الذي يسرع كثيرا من عملية التدريب و الحصول علي القيم المناسبة لـ  $w$  لا تنس أن تقوم بعمل نفس التسوية بنفس الطريقة لعينة التدريب و الاختبار معا , و بالتالي يتم عملها في كل العينة قبل تقسيمها ونأتي للسؤال , كيف تقوم "تسوية البيانات" بتسريع الحصول علي القيم المثالية لـ  $w$  ؟

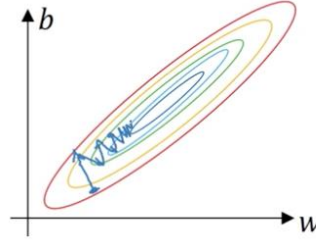
إذا تخيلنا بيانات غير مسواة , فمن الممكن أن يكون لبيانات معينة مثلا 3 features , وتكون كالتالي :

إكس 1 (الراتب)	:	يتراوح بين 1000 إلي 10 الاف
إكس 2 (العمر)	:	يتراوح بين 20 و 60
إكس 3 (الدرجة)	:	تتراوح بين 0.0 إلي 0.1

فالمشكلة أن هذه الأرقام , لأن المدي لكل منها يختلف تماما عن الآخر , فإذا رسمناها فستكون شكل منبعج مثل هذا :



والذي إن قمنا بعمل انحدار تدريجي له للوصول لقيم مثلي سيكون هكذا :



وهو ما يعني كم كبير من الوقت , والاضطرار لتحديد معامل تدريب قليل (الفا) حتي نصل للقيم المثالية

بينما إن قمنا بتسوية البيانات معا , وصارت الإكسبات الثلاثة تتراوح بين سالب واحد وواحد , أو من صفر لواحد , فسيكون الرسم كالتالي :



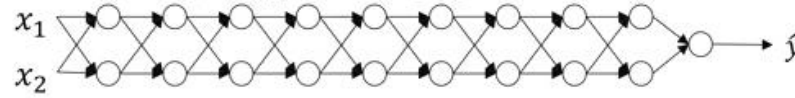


## تضخم و تضائل الاشتقاقات Vanishing & Exploding Gradients

أحد المشاكل التي تواجهنا في الشبكات العميقة , هي أن الاشتقاقات  $dw$  ,  $db$  قد يتضخم حجمها جدا أو يتضائل بشكل كبير , مما يزيد من صعوبة أن تسير الشبكة بالشكل المثالي . .

و لنفهم السبب تعالي نري هذه الشبكة :

### Vanishing/exploding gradients



في حين نري هنا شبكة من 10 طبقات , فهناك شبكات يزيد عدد طبقاتها عن الـ 200 طبقة , لكننا نتناول مثالا مبسطا

لنفترض مثالا مبسطا , سنقول أن دالة  $g(z)$  ستكون دالة خطية بحيث :  $g(z) = z$

وسنفترض أيضا أن قيمة  $b$  تساوي صفر , فيكون قيمة الأكتيفاشن  $a$  تساوي  $w*x$  فقط

ولما كنا نتعامل مع طبقات عميقة , فممكّن ان نقول ان  $w1*x$  يساوي  $a1$  , وإذا اردنا حساب  $a2$  فسيكون  $w2*a1$  وهو ما معناه  $w2*w1*x$  , وبالتالي قيمة  $a3$  ستكون  $w3*w2*w1*x$  , وهكذا

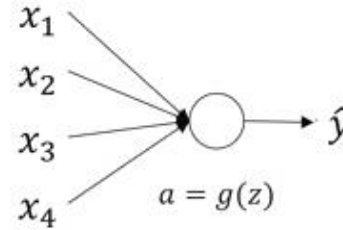
وبالتالي إذا كنا نريد حساب قيمة  $y^8$  النهائية فستكون حاصل ضرب كل قيم  $w$  في بعضها ضرب  $x$  هكذا



أحد الوسائل التي تقوم بحل المشكلة السابقة هي القيام بتحديد دقيق لقيم  $w$  ,  $b$

و من الواضح أنه كلما زادت عدد الطبقات , وكلما زادت عدد الخلايا في كل طبقة , فإن القيم النهائية ستتطرف بشكل كبير (تتضخم او تتصائل) وبالتالي فإنه يفضل أن تكون القيم المبدئية لـ  $w$  ,  $b$  تتعلق بعدد الخلايا و الطبقات , وتصغر بقدر الامكان (تقترب من الـ 1 و ليس الـ 0) حتي لا تتطرف القيم النهائية

فإذا رأينا مثالا لخلية واحدة بها عدد من المدخلات مثل هذه :



فقيمة  $z$  النهائية ستكون (مع اعتبار ان  $b$  بصفر للتسهيل)

$$z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

وبالتالي يهمننا ان تصغر قيمة  $w$  كلما زادت عدد المدخلات (الإكسات)

فنقوم بإضافة جزء صغير إلي الكود الذي يقوم باختيار قيم  $w$  المبدئية

كنا نكتب الكود هكذا :

$$W^{[1]} = np.random.randn(shape)$$

والذي يحدد قيم  $W$  كقيم عشوائية بناء على الابعاد المطلوبة لمصفوفة  $W$

لكننا سنضيف هذا الجزء

$$W^{[1]} = np.random.randn(shape) * np.sqrt\left(\frac{1}{n^{(1-1)}}\right)$$

وهو الذي يقوم بقسمة الارقام العشوائية التي تم اختيارها , على الجذر التربيعي لعدد الوحدات (الخلايا)

وبالتالي لو كان لدينا عدد 15 مدخل (إكسات) فنقوم بقسمة الارقام العشوائية على جذر 15 , وهو ما يساوي تقريبا 4 , أي أننا حصلنا فقط على ربع القيم العشوائية

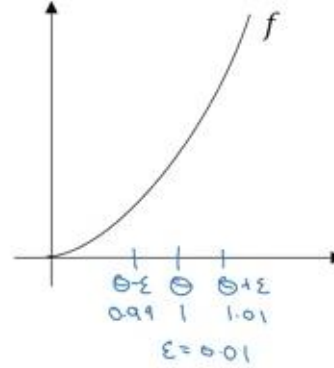
و نفس الفكرة لو كان لدينا 100 مدخل , فنحصل على عشر القيم فقط

و هذا التقليل يساعدنا على تجنب تطرف القيم النهائية بشكل كبير

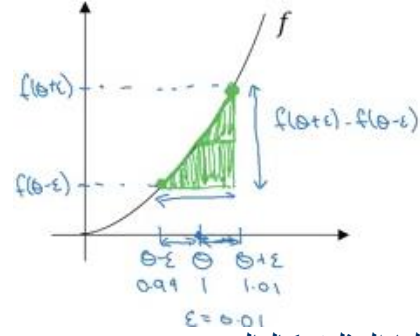
بينما لو أنك تستخدم دالة ReLU فيفضل تغيير هذا الجزء  $np.sqrt\left(\frac{1}{n^{(1-1)}}\right)$  الي 2 على  $n$  أي بضرب جذر 2 في المعادلة

نذهب الان لنقطة سبق الحديث عنها , وهو الفحص التدريجي Gradient Checking

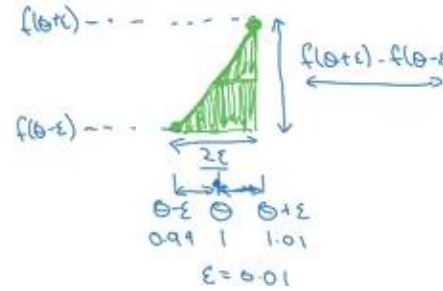
ولنفهمه جيدا , تعالي نتكلم أولا عن تعريف الاشتقاق نفسه gradient



لو تناولنا دالة بسيطة , مثل الدالة التكعيبية هنا , و قمنا بتحديد رقم نيتا , يساوي 1 , ثم انتقلنا لليمين مسافة صغيرة جدا (0.01) و سنسميها ايسلون , وانتقلنا لليساار مسافة مشابهة , و قمنا بتحديد قيمة  $f(x)$  للقيمتين مثل هنا :



وبالتالي قيمة الاشتقاق ستكون الارتفاع علي العرض , وإذا تناولنا المثلث كالتالي :



فتكون قيمة الاشتقاق هي :

$$\frac{f(1.01) - f(0.99)}{0.02}$$

وإذا طبقنا قيمة ثابتا تساوي 1 , وأبسلون تساوي 0.01 , تكون قيمة الاشتقاق :

$$\frac{(1.01)^3 - (0.99)^3}{2(0.01)} = 3.0001 \approx 3$$

وإذا علمنا ان اشتقاق الدالة التكعيبية يساوي :

$$g(\theta) = 3\theta^2 = 3$$

والتي تساوي 3 لذي تعويض ثيتا تساوي 1

يكون وقتها الفارق بين قيمة الاشتقاق بالقانون , وبالطريقة الهندسية هو 0.0001

وهو الذي يمثل قيم الخطأ الذي نريد تجنبه

ولاحظ أنه في حاله استخدام ايسلون من اتجاه واحد , وصارت المقارنة بين ثيتا , وثيتا زائد ايسلون فقط , وقتها يصير قيمة الخطأ 0.03 , وهو اكبر بكثير من القيمة السابقة

وهو ما يؤكد أهمية استخدام اتجاهين معا

\* \* \* \* \*



## الفحص الاشتقاقي Gradient Checking

وهو من الادوات بالغة الاهمية , والتي توفر كمية كبيرة من الوقت والمجهود اثناء التدريب

وفكرته تقوم علي المقارنة بين رقمين (أو مصفوفتين) شديديتي الأهمية , تسميا :

- الاشتقاق العددي Numerical Gradient

- الاشتقاق التحليلي Analytical Gradient

وعلي قدر الفارق بين القيمتين , علي قدر مقدار الخطأ الذي يقع فيه الخوارزم الذي أعمل به , والحالة المثالية (التي لا تحدث) أن تكون القيمتين متطابقتين تماما

والاشتقاق العددي يتم عبر استخدام ايسلون , بينما التحليلي عبر تجميع مصفوفات الاشتقاقات من المسار الخلفي

و تبدأ الخطوات اولا , بتناول كل قيم  $w$  ,  $b$  الموجودة في الشبكة و التي هي :

$$w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}$$

و تجميع كل قيمها في فيكتور واحد ضخم , من عمود واحد , بحيث تبدأ  $w1$  يليها  $b1$  ثم  $w2$  و هكذا , ويسمي هذا الفيكتور :  $J(\theta)$

كذلك الأمر نعيد الفكرة في الاشتقاقات , ونجمع كل قيم :

$$dw^{[1]}, db^{[1]}, \dots, dw^{[L]}, db^{[L]}$$

في فيكتور واحد ضخم يسمي  $d(\theta)$

و هذا الفيكتور هو الاشتقاق التحليلي , والذي سنستخدمه للمقارنة مع الاشتقاق العددي الذي سنحسبه حالا

لا تنس ان ابعاد كل  $w$  هي نفسها  $dw$  و كل  $b$  هي نفسها  $db$  و بالتالي ابعاد  $J(\theta)$  ستكون هي نفسها  $d(\theta)$

ثم نقوم بحساب الاشتقاق العددي , عبر ايجاد الميل الكلي لمعادلة الخطأ  $J$  , والتي جمعناها حالا

و يأتي هذا بنفس المعادلة المستخدمة سابقا في الفيديو السابق

$$\frac{f(\theta+\epsilon) - f(\theta-\epsilon)}{2\epsilon}$$

ولأن معادلة الخطأ  $J$  هي دالة في العديد من الثبتات (والذي يساوي مجموع عدد ثبتات  $w_1, b_1, w_2, b_2, \dots$ ) فسنقوم بتطبيق حساب الميل بهذا العدد من المرات

$$\text{for each } i: \\ \rightarrow \underline{d\theta_{\text{approx}}[i]} = \frac{J(\theta_1, \theta_2, \dots, \theta_i + \epsilon, \dots) - J(\theta_1, \theta_2, \dots, \theta_i - \epsilon, \dots)}{2\epsilon}$$

وهذا يتم عبر أولا , اختيار أول ثبتا فيهم , وإضافة قيمة ابلسون السالف شرحها , علي قيمة ثبتا المختارة , و ايجاد قيمة  $J$  , ثم طرحها من نفس القيمة لكن بعد طرح ابلسون , وقسمتها علي ضعف ابلسون

وبعدها اقوم بعدل نفس العملية علي ثبتا التالية وهكذا حتي اصل لآخر ثبتا , واقوم بتخزين كل قيمة  $d\theta$  احصل عليها في مصفوفة جديدة و تكون هي الاشتقاق العددي

ولحساب الفارق بدقة , نستخدم الصيغة :

$$\frac{\|2\mathcal{O}_{\text{approx}} - d\|_2}{\|2\mathcal{O}_{\text{approx}}\|_2 + \|d\|_2}$$

والبسط بالاعلى يشير إلى الجذر التربيعي لمجموع مربعات الفارق بين كل ثيتا و ثيتا المقابلة لها , فلو كانت قيم ثيتا العددية هي :

$n_1, n_2, n_3 \dots$

وقيم ثيتا التحليلية هي :

 $a_1, a_2, a_3 \dots$ 

فيكون البسط هو :

$$\text{Root}( (n_1-a_1)^2 + (n_2-a_2)^2 + (n_3-a_3)^2 \dots )$$

أما المقام فهو :

$$\text{Root}((n1)^2 + (n2)^2 + (n3)^2 \dots) + \text{Root}((a1)^2 + (a2)^2 + (a3)^2 \dots)$$

وإذا كان الفارق يقل عن  $10^{-7}$  فهو رقم مناسب , وإذا كان في رينج  $10^{-5}$  فهذا يعني بداية وجود مشاكل , أما إذا كان  $10^{-3}$  فهو يعني وجود أخطاء في الكود

\* \* \* \* \*

وهنا نقوم بسرد عدد من النصائح الهامة الخاصة بالفحص الاشتقاقي :

أولاً : لا تقم بعمل الفحص الاشتقاقي بشكل مستمر لكل عنصر من عناصر العينة , فلو كان لدي 10 الاف عنصر في العينة , فالقيام بعمل فحص اشتقاقي في كل خطوة امر يستغرق وقتا طويلا , خاصة اذا ما كانت الشبكة عميقة , وكان عدد الثيتات كبير لكبر عدد الخلايا والطبقات , فقط فم به مرات معدودة

ثانياً : عندما تجد ان الفارق كبير بين الاشتقاق التحليلي و العددي , قم بفحص الارقام واحد تلو الاخر لتحدد ايهما متباعد بشكل كبير . .

فقد تلاحظ مثلا ان اغلب قيم  $dw$  قريبة من بعضها بشكل كبير , بينما قيم  $db$  هي التي تسبب الفارق , وقتها ابدأ بتغيير اسلوب تعاملك مع  $db$  , او قد تجد ان بعض ارقام  $dw$  هي التي تقوم بعمل مشاكل , فتعلم ان هذه طبقة معينة من الطبقات هي التي تسبب هذا الفارق , فنقوم بعمل تعديل لها

ثالثا : لا تنس إضافة قيمة التنعيم إذا ما كنت تستخدمه في إيجاد  $d\theta$  الحسابية (باستخدام ايسلون )

رابعا : لا يمكن الجمع بين تكتيك حذف الخلايا (dropout) و الفحص الاشتقاقي , لان الفحص الاشتقاقي قائم علي جمع قيم ثيتات لكل الخلايا , فحذف احد الخلايا سيؤدي لاضطرابات في الحساب , فيجب أن نحدد معامل الابقاء بـ 1 اثناء الفحص الاشتقاقي

\* \* \* \* \*

## نهاية الاسبوع الاول