



كيف تتدكم في تعقيد الموديل؟

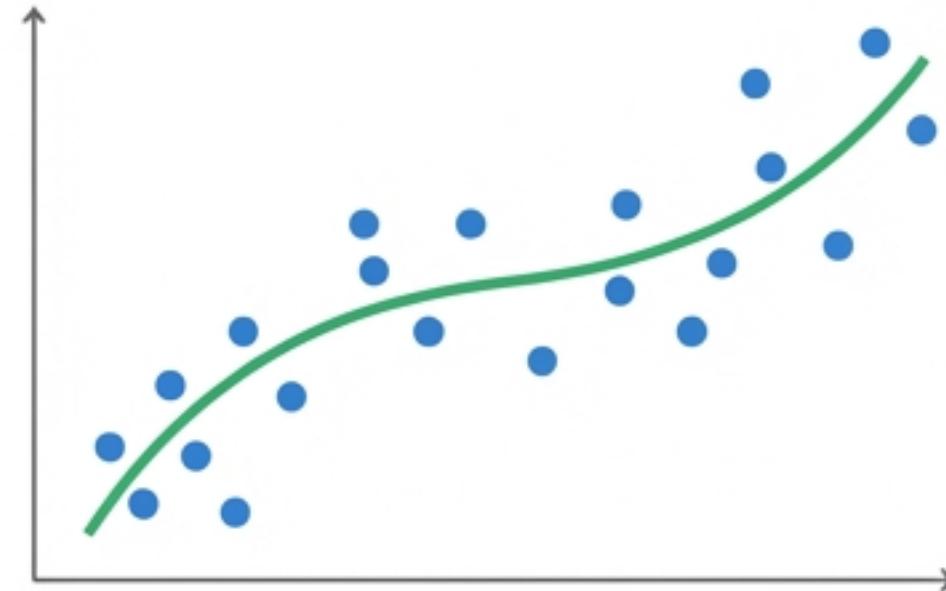
شرح مُفصّل لتقنيات الـ'Regularization' في تعلم الآلة.

مقدمة قصيرة تشرح أن الهدف من هذا العرض هو فهم عميق لكيفية منع الموديل من 'الحفظ' الزائد للبيانات، وكيفية إجباره على تعلم الأنماط الحقيقية فقط. سأناشد، سيتم التركيز على الشرح بالصور والرسوم البيانية.

المشكلة الأساسية: الـ`Overfitting`

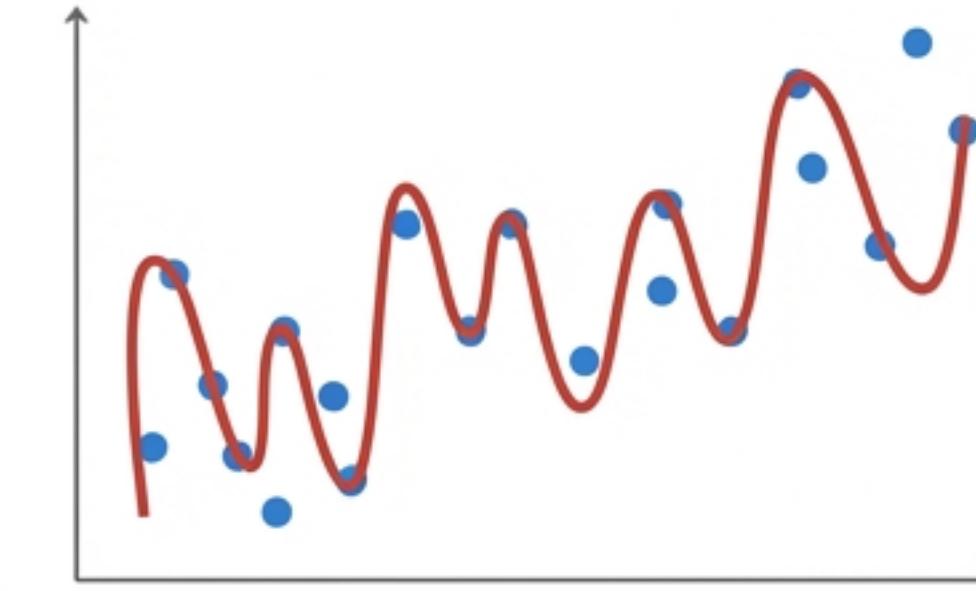
لما الموديل بتاعك يحفظ الداتا بدل ما يفهمها.

نموذج متوازن (Good Fit)



أداء جيد على بيانات التدريب والاختبار

نموذج مُفرط في التعلم (Overfitting)



أداء ممتاز على بيانات التدريب، وسيء جداً على بيانات الاختبار

↓ خطأ التدريب (Train Error): صغير جداً.

↑ خطأ الاختبار (Test Error): كبير جداً.

السبب الشائع: الأوزان (weights) بتكبر جداً عشان الموديل يطارد كل نقطة في بيانات التدريب.

الحل: فلنجر الموديل على تفضيل الأوزان الصغيرة

إ هنا بنعّدّل الـ cost function .weights على كبر الـ **عقوبة (penalty)** ونضيف إليها.

$$J_{\text{reg}}(w) = J_{\text{original}}(w) + \lambda \cdot R(w)$$



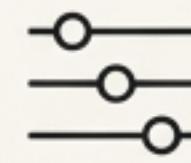
$J_{\text{original}}(w)$

دالة التكلفة الأصلية (e.g., MSE)



$R(w)$

حد التنظيم (Regularization Term)
- هو ده اللي ييعاقب الأوزان الكبيرة.

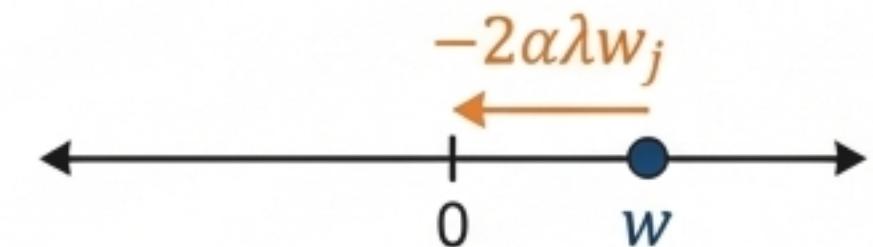


λ

معامل فائق (hyperparameter) بيتحكم في قوة العقوبة.
لو $\lambda = 0$ ← لا يوجد regularization.
لو λ كبير ← عقوبة قوية تجبر الأوزان على الاقتراب من الصفر.

How it Works (Gradient Descent Intuition)

شرح مبسط لكيف أن إضافة λ للمعادلة يجعل كل خطوة في Gradient Descent تسحب الوزن قليلاً نحو الصفر. الجزء $-2\alpha\lambda w_j$ – كأنه كل step بيشد الـ w ناحية الصفر (shrinkage).



‘Ridge Regression’ الأولى: (L2 Regularization)

→ ‘Shrinkage’ الأوزان نحو الصفر

$$J_{\text{ridge}}(\mathbf{w}) = \frac{1}{m} \sum (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \sum w_j^2$$

نلاحظ أن العقوبة هي مجموع مربعات الأوزان
(sum of squared weights)

غالباً لا يتم تطبيق العقوبة
على الـ bias (w_0).

التأثير - The Effect

- كل الأوزان تتقلص وتقرب من الصفر.



- نادراً جداً ما يصل وزن إلى الصفر تماماً.
(لا يقوم باختيار الميزات -
.No feature selection $\neq 0$)



- مفيد جداً مع الميزات المتراكبة
(multicollinearity).



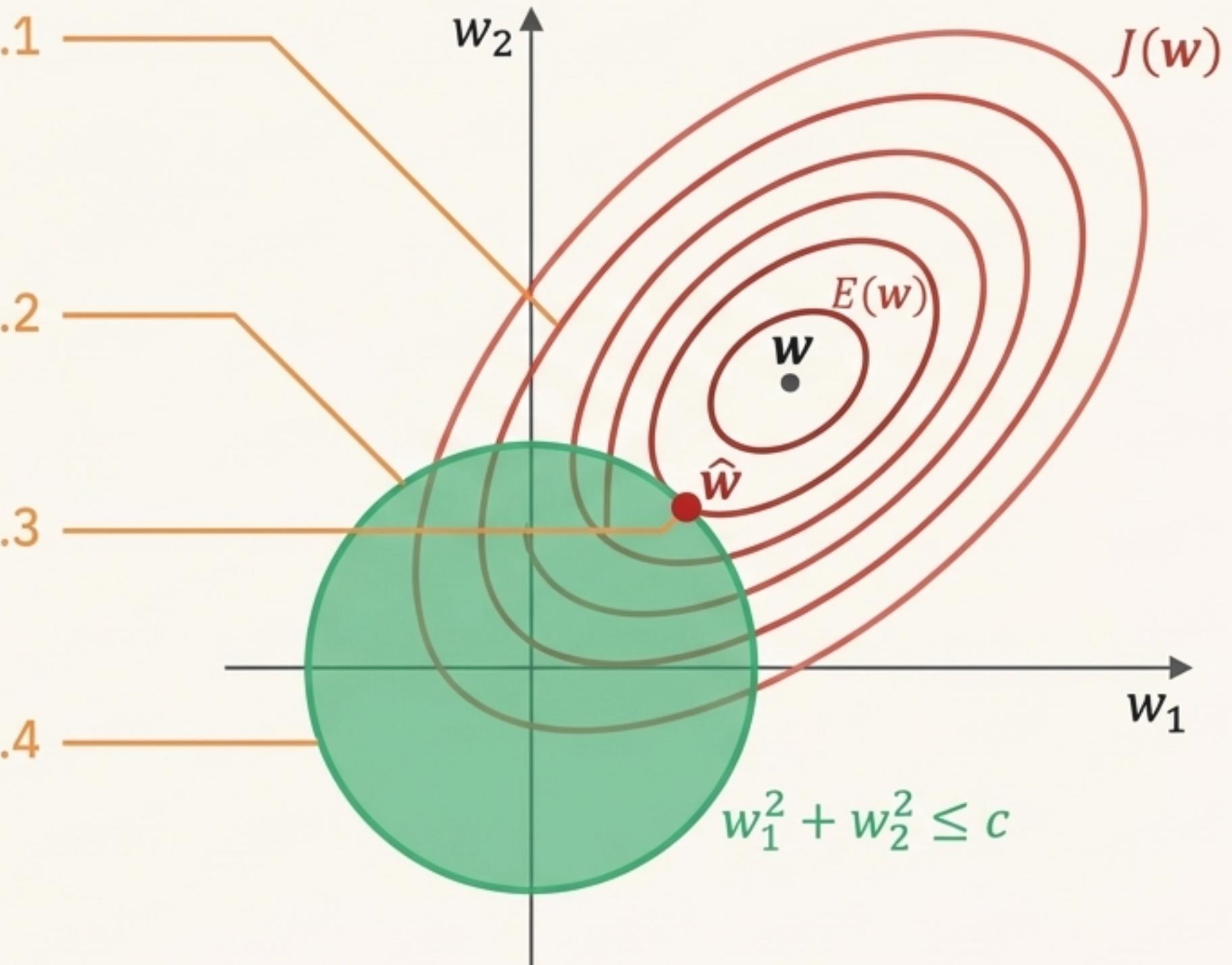
الفهم البصري لـ`Ridge`: قصة الدائرة والـ`Ellipses`

الخطوط الحمراء (Ellipses): تمثل مستويات متساوية من دالة التكلفة الأصلية ($J(w)$). أصغر ellipse في المنتصف هو الحل الأمثل بدون regularization.

الدائرة الخضراء (The Constraint): تمثل شرط Ridge $w_1^2 + w_2^2 \leq c$. الموديل مجبر على اختيار حل داخل هذه الدائرة.

الحل الأمثل (\hat{w}): هو نقطة التلامس بين أصغر ellipse-ممكن والدائرة. هذه هي أقل تكلفة ممكنة مع الالتزام بالشرط.

لماذا لا تصل الأوزان للصفر؟: لأن حدود الدائرة ملساء (smooth). نقاط التلامس غالباً لا تحدث على المحاور (where mere $w_1=0$ or $w_2=0$), مما يجعل الأوزان صغيرة ولكن ليست صفرية.



الطريقة الثانية: 'Lasso Regression' (L1 Regularization)

اختيار الميزات عن طريق تصفير الأوزان
(`Feature Selection`)

$$J_{\text{Lasso}}(\mathbf{w}) = \frac{1}{m} \sum (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \sum |w_j|$$

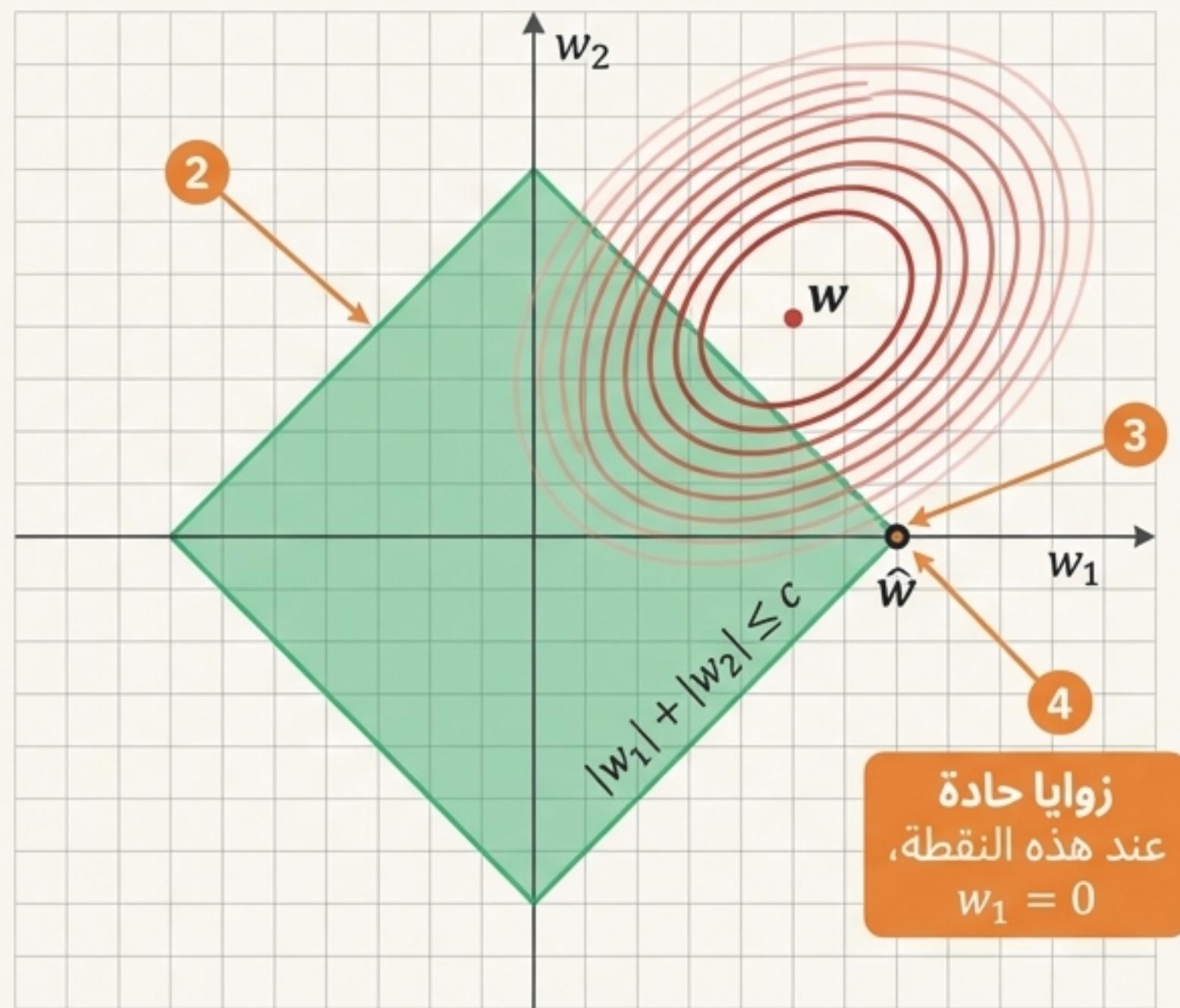
نلاحظ أن العقوبة هي مجموع القيم المطلقة للأوزان (sum of absolute values).

التأثير - The Effect

- يجبر بعض الأوزان على أن تصبح صفرًا تماماً. 
- ينتج عنه حل متفرق (sparse solution), مما يعني أنه يقوم باختيار الميزات الأكثر أهمية فقط. 
- مفيد جدًا عندما تتوقع أن عددًا قليلاً فقط من الميزات هو المهم. 

الفهم البصري لـ`Lasso`: سر المعین (الدايموند)

1. الـ`Ellipses` الحمراء: كما في `Ridge`، تمثل دالة التكلفة.
2. المعین الأخضر (The Constraint): يمثل شرط $|w_1| + |w_2| \leq c$ لـ`Lasso` له زوايا حادة (sharp corners) على المحاور.
3. الحل الأمثل (\hat{w}): هو نقطة التلامس. بسبب الزوايا الحادة، من المرجح جدًا أن يحدث التلامس عند إحدى الزوايا.
4. لماذا تصل الأوزان للصفر؟: عندما يكون الحل عند زاوية مثل $(0, c)$ ، هذا يعني أن $w_1 = 0$. هذه الزوايا هي السبب المباشر الذي يجعل `Lasso` قادرًا على تصفيير الأوزان وإلغاء بعض الميزات.



‘Ridge’ مقابل ‘Lasso’: مقارنة مباشرة

Ridge (L2)

Constraint Formula

$$\sum w_j^2 \leq c$$

Effect

Shrinkage. يقلص جميع الأوزان بشكل متناسب. متناسب.

Outcome

نادراً ما تكون الأوزان صفرية.

Best Use Case

عندما تكون معظم الميزات مهمة، أو عند وجود ‘multicollinearity’.

Lasso (L1)

Constraint Formula

$$\sum |w_j| \leq c$$

Effect

‘Sparsity / Feature Selection’. Sparsity يصغر العدد من الأوزان.

Outcome

حل متفرق (sparse) يحتوي على أوزان صفرية.

Best Use Case

عندما تتوقع أن عدداً قليلاً من الميزات هو المهم.

أفضل ما في العالمين: 'Elastic Net'

هو مزيج بين 'Ridge' و 'Lasso', يجمع بين قوة كليهما.

$$J_{\text{EN}}(\mathbf{w}) = J_{\text{original}}(\mathbf{w}) + \lambda \left(\alpha \sum_{j=1}^n |w_j| + (1 - \alpha) \sum_{j=1}^n w_j^2 \right)$$

(L1 part)
(L2 part)

يتذكّم في قوّة الـ regularization بـشكل عام.
يتذكّم في المزج بين L1 و L2.



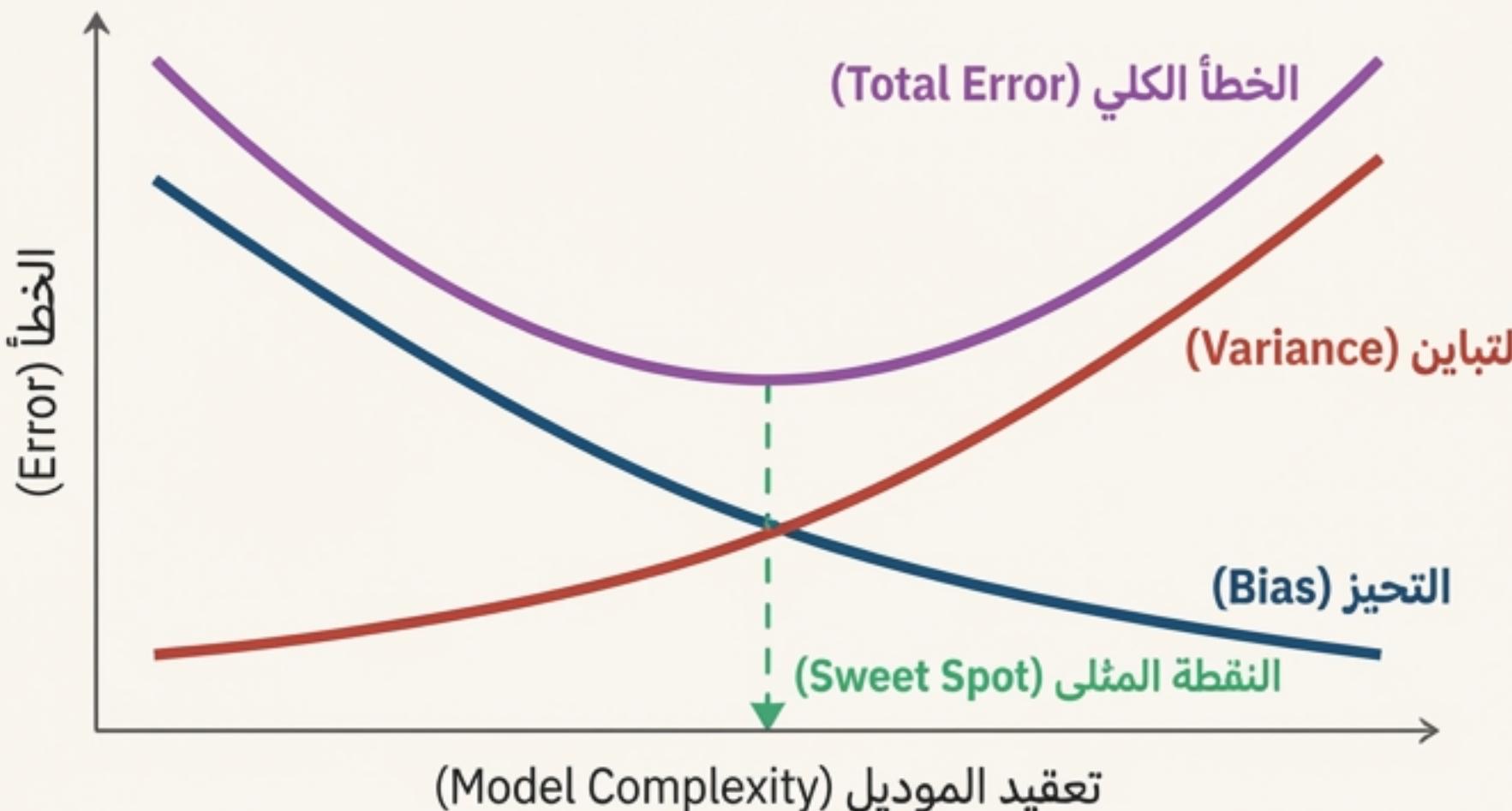
لماذا نستخدمه؟ (Why use it?)

- 'Lasso' قد يختار ميزة واحدة بـشكل عشوائي من بين مجموعة ميزات متربطة.
 - Elastic Net يحتفظ بقدرة 'Lasso' على اختيار الميزات (sparsity) وفي نفس الوقت يتعامل مع الميزات المتربطة بشكل أفضل مثل 'Ridge' (تأثير المجموعة - (grouping effect)).



الصورة الأكبر: الـ 'Regularization' ومقاييس التحييز-البيان (Bias-Variance Tradeoff)

الـ 'Regularization' هو أداة للتحكم في هذه المقاييس.



زيادة الـ (More Regularization)

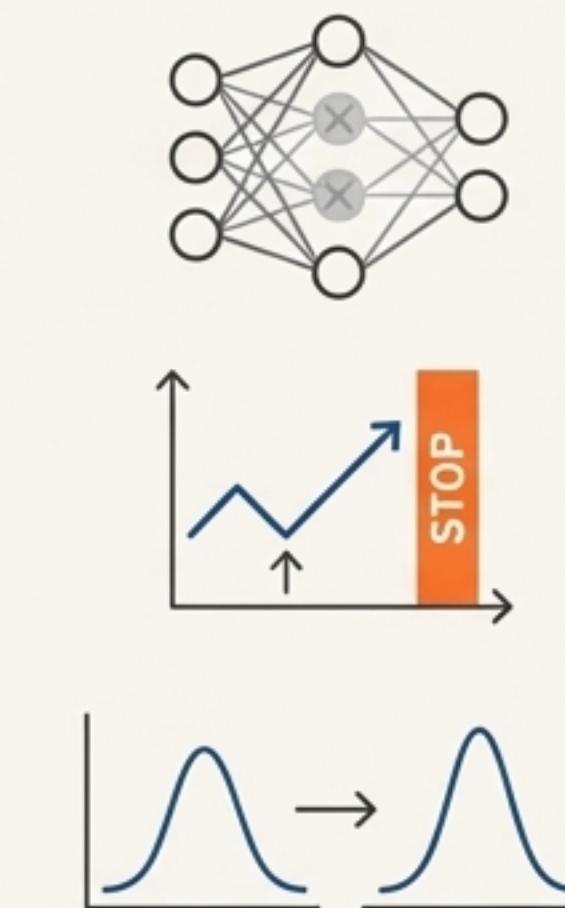
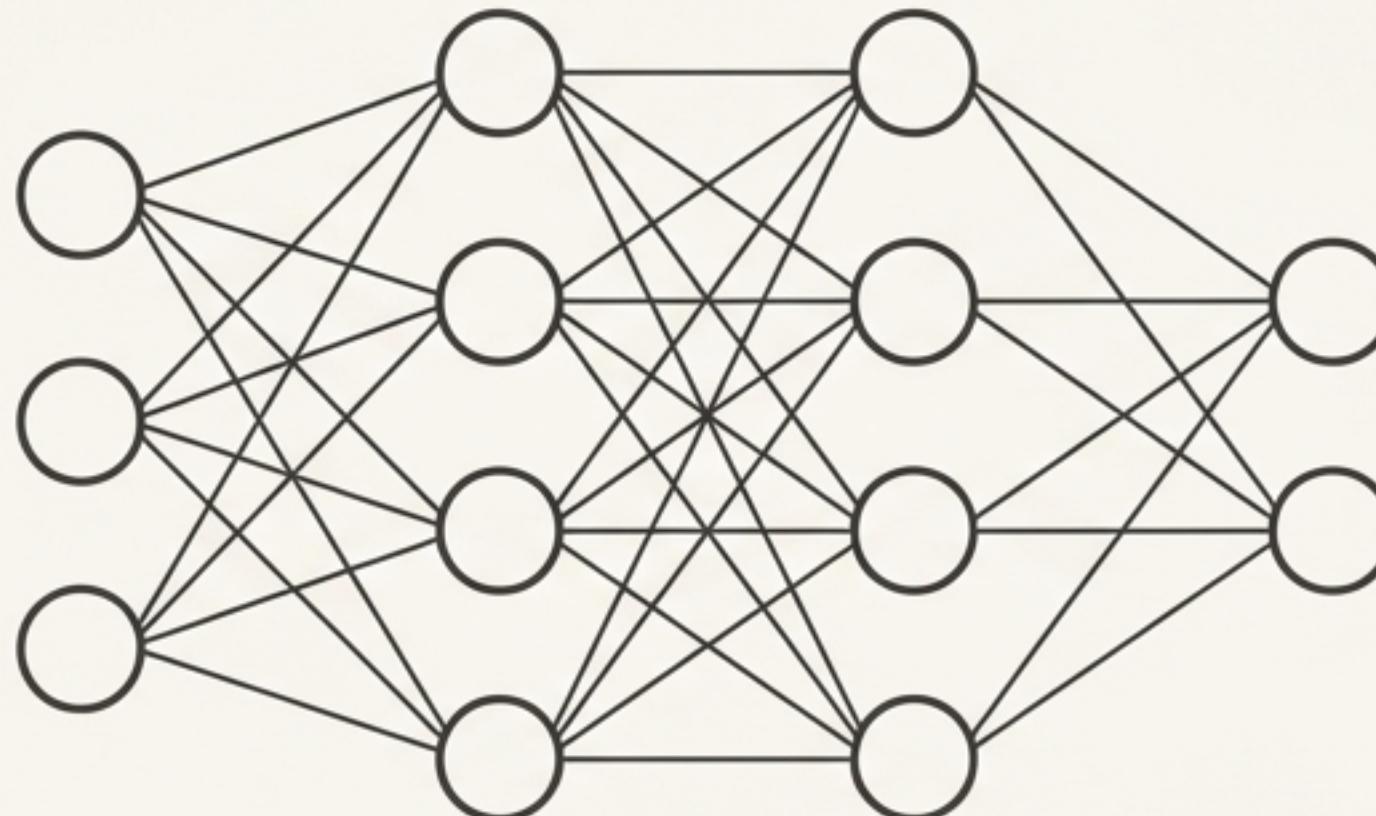
- تقلل الـ **Variance** (الموديل أقل حساسية للضوضاء، أقل overfitting). ↓
- تزيد الـ **Bias** (الموديل قد يصبح أبسط من اللازم, underfitting). ↑

الهدف

العثور على قيمة λ المثلثي باستخدام validation set أو cross-validation، للوصول إلى أفضل نقطة في منحنى الخطأ الكلي.

الـ'Regularization' في الشبكات العصبية: أساليب متقدمة

نفس مبدأ التحكم في التعقيد ينطبق على الشبكات العصبية، ولكن بأدوات مختلفة.



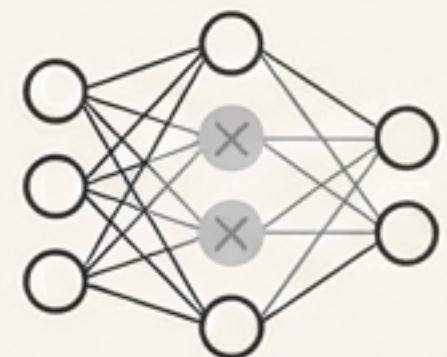
Dropout

Early Stopping

Batch Normalization

هذه التقنيات تساعد الشبكات العصبية العميقه على التعميم بشكل أفضل وتجنب الـ'overfitting'.

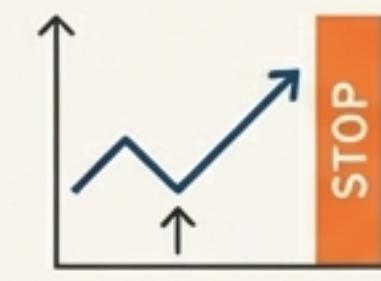
لمحة سريعة: `Batch Normalization` & `Stopping`, `Dropout`



`Dropout`

الفكرة: أثناء التدريب، يتم 'إطفاء' نسبة من الخلايا العصبية (neurons) بشكل عشوائي في كل خطوة.

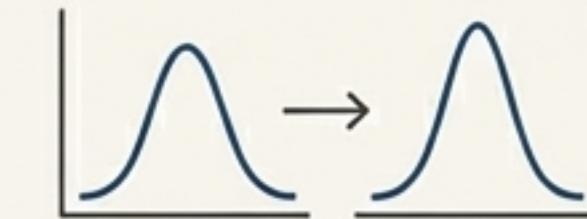
التأثير: يجبر الشبكة على عدم الاعتماد على خلايا عصبية معينة، مما يمنع التكيف المشترك (co-adaptation).



`Early Stopping`

الفكرة: مراقبة أداء الموديل على validation set وإيقاف التدريب عندما يبدأ خطأ التحقق في الزيادة.

التأثير: يمنع الموديل منمواصلة التدريب والدخول في مرحلة overfitting.



`Batch Normalization`

الفكرة: تطبيع (normalize) مخرجات كل طبقة داخل ال .mini-batch

التأثير: يسرّع التدريب، يسمح بمعدلات تعلم أعلى، وله تأثير regularization طفيف بسبب الضوضاء في إحصائيات ال .batch

ملخص أدواتك للتحكم في التعقيد (Your Toolkit for Controlling Complexity)

Ridge (L2)



العقوبة: $\sum w_j^2$
الشكل الهندسي:
دائرة

Shrinkage: يقلص الأوزان.

Lasso (L1)



العقوبة: $\sum |w_j|$
الشكل الهندسي:
معين

Sparsity: يصفر الأوزان (Zeroes weights). Feature Selection

Elastic Net

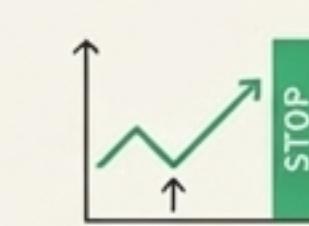
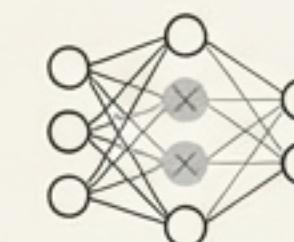


العقوبة: مزيج من L1 و L2

الشكل الهندسي:
مزيج من الدائرة والمعين

Sparsity: مع التعامل الجيد مع الميزات المتراكبة.

Dropout / Early Stopping



السياق: الشبكات العصبية
الأدوات:

التأثير الأساسي: منع الـ`overfitting` في النماذج المعقدة.

الهدف دائماً هو تقليل الـ`overfitting` عن طريق إضافة نوع من العقوبة أو القيد على تعقيد الموديل.

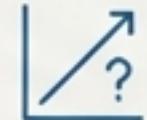


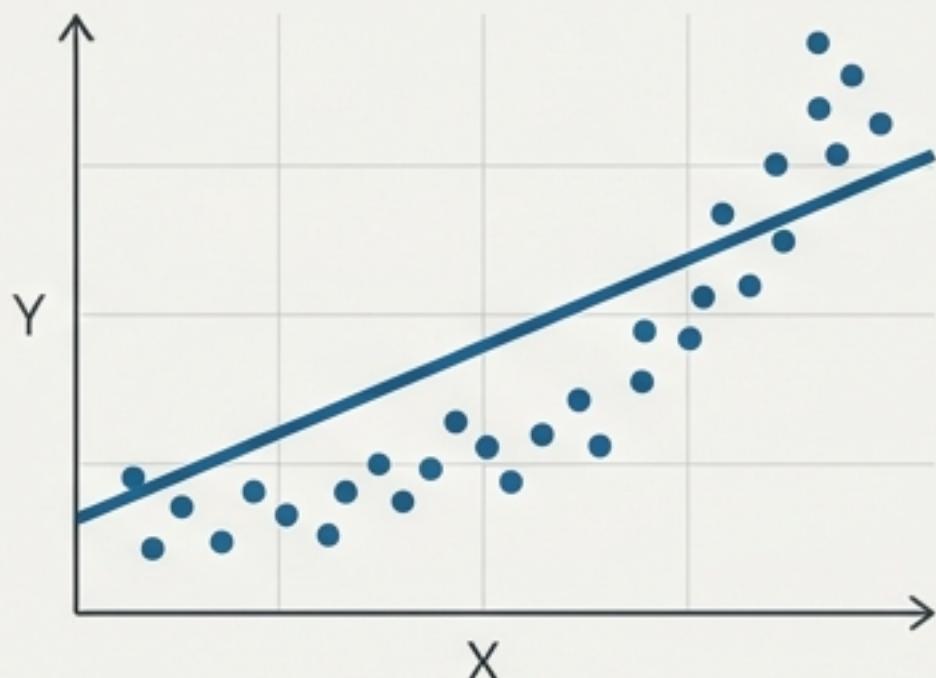
فَأَكْ شُفَرَةُ أَبَادِعُ نَمَادِبَكْ

دليل مصوّر لتشخيص نماذج التعلم الآلي
باستخدام منظريات الأداء

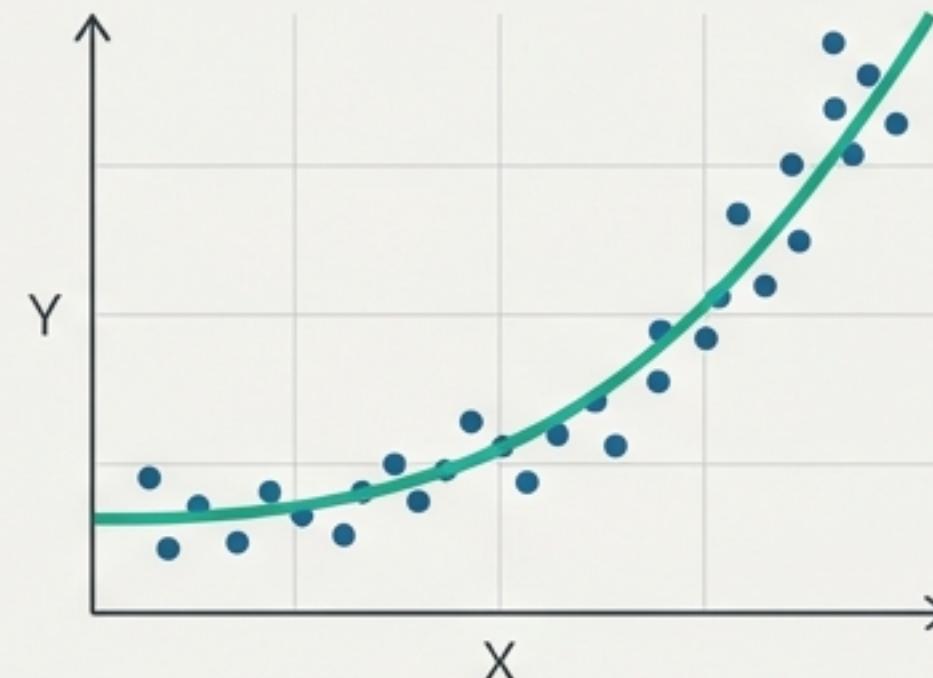
التحدي الأساسي: هل نموذجك يتعلم أم يحفظ؟

الهدف من أي نموذج تعلم آلي هو **التععميم** (Generalization) - أي القدرة على أداء جيد على بيانات جديدة لم يرها من قبل. وتواءمة من المخاطر حكفيات المسليمة:

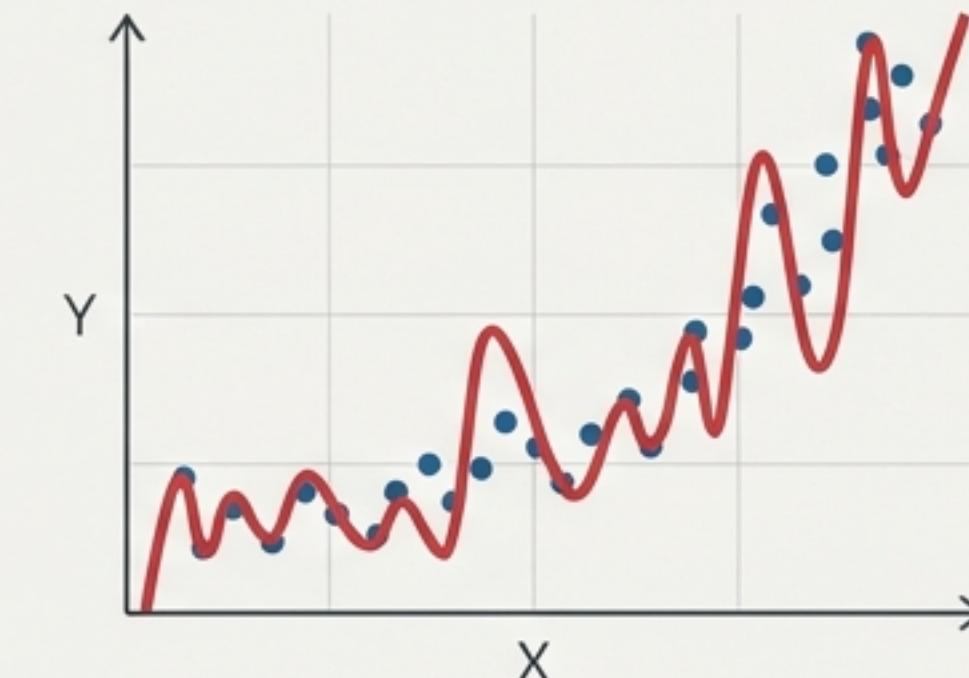
- **التحيّز العالي (High Bias / Underfitting)**: النموذج أبسط من اللازم ولا يستطيع التقاط النمط الأساسي في البيانات.  إنه "لا يتعلم" بشكل كافٍ.
- **التباین العالی (High Variance / Overfitting)**: النموذج معقد أكثر من اللازم ويحفظ البيانات التدريبية بما فيها من ضوضاء. إنه "يحفظ" بدلًا من أن يتعلم. 



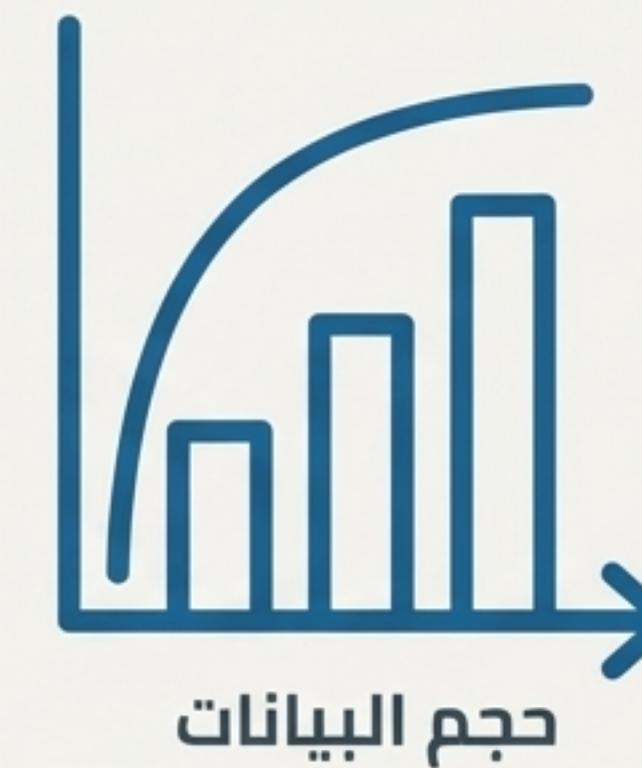
Underfitting



Good Fit



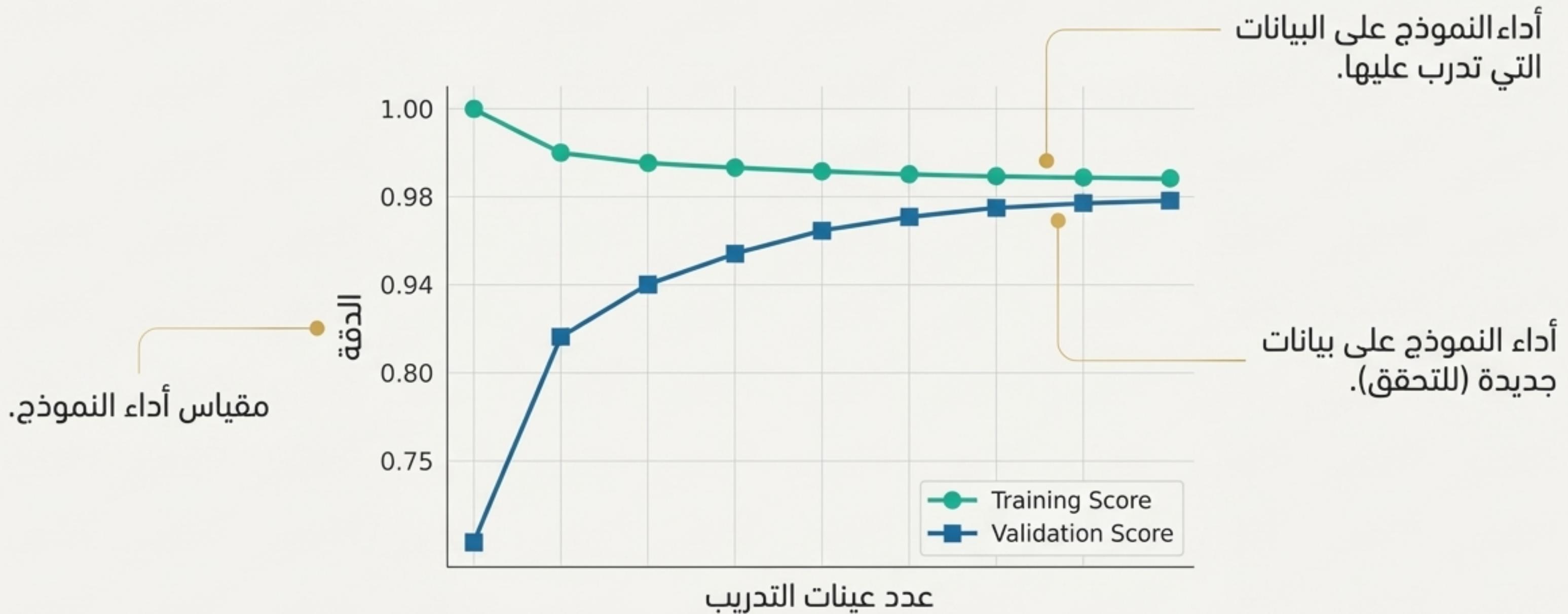
Overfitting



الأداة الأولى: منحنى التعلم

كيف يؤثر جم البيانات على دقة النموذج؟

تشريح منحني التعلم

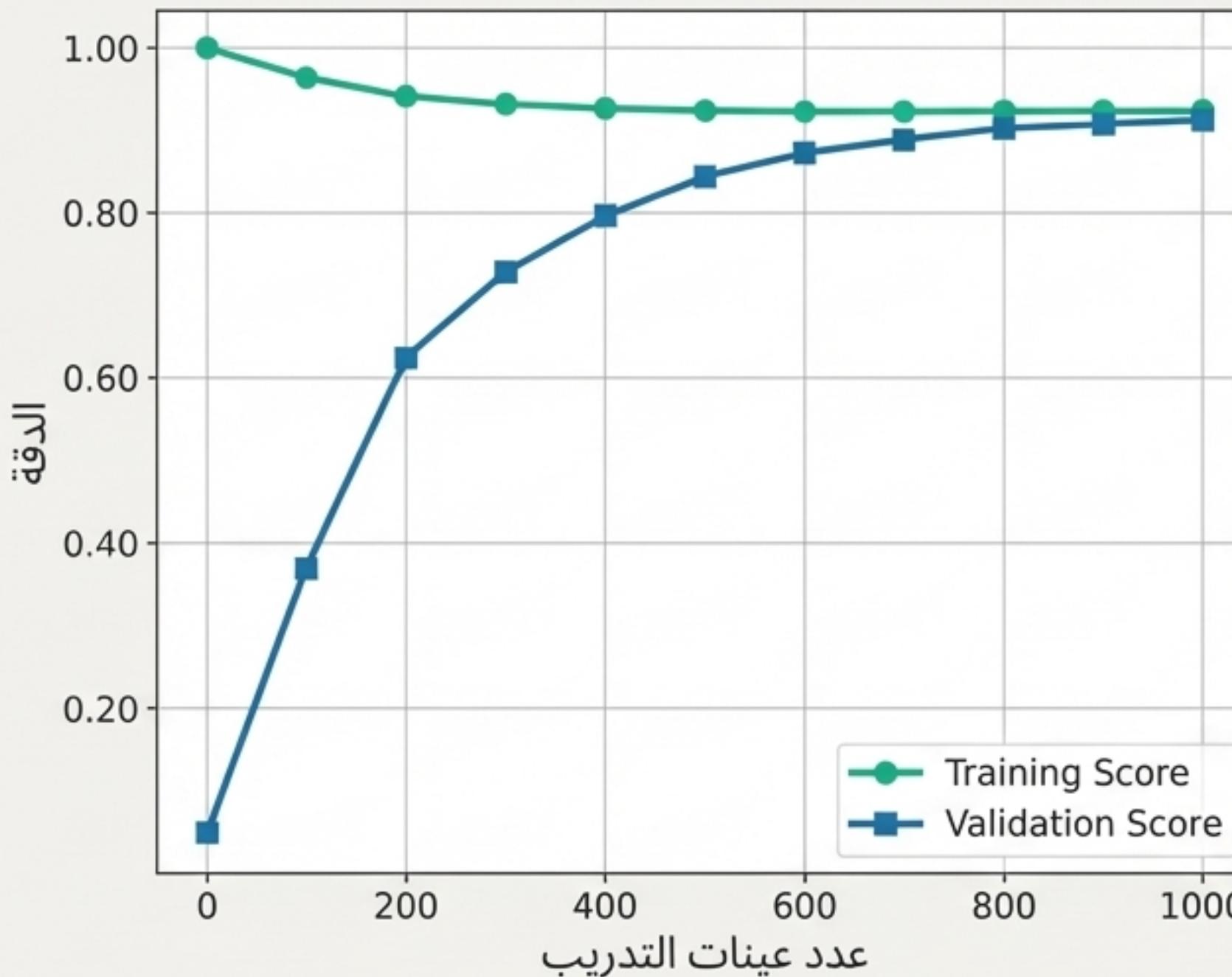


نزيد حجم البيانات تدريجياً ونراقب الأداء.

السلوك الطبيعي: في البداية، يكون أداء التدريب مثاليًا وأداء التحقق ضعيفاً. مع زيادة البيانات، يتقارب المنحنيان ويثبتان.



الحالة المثالية: النموذج المناسب (Good Fit)



النمط (The Pattern)



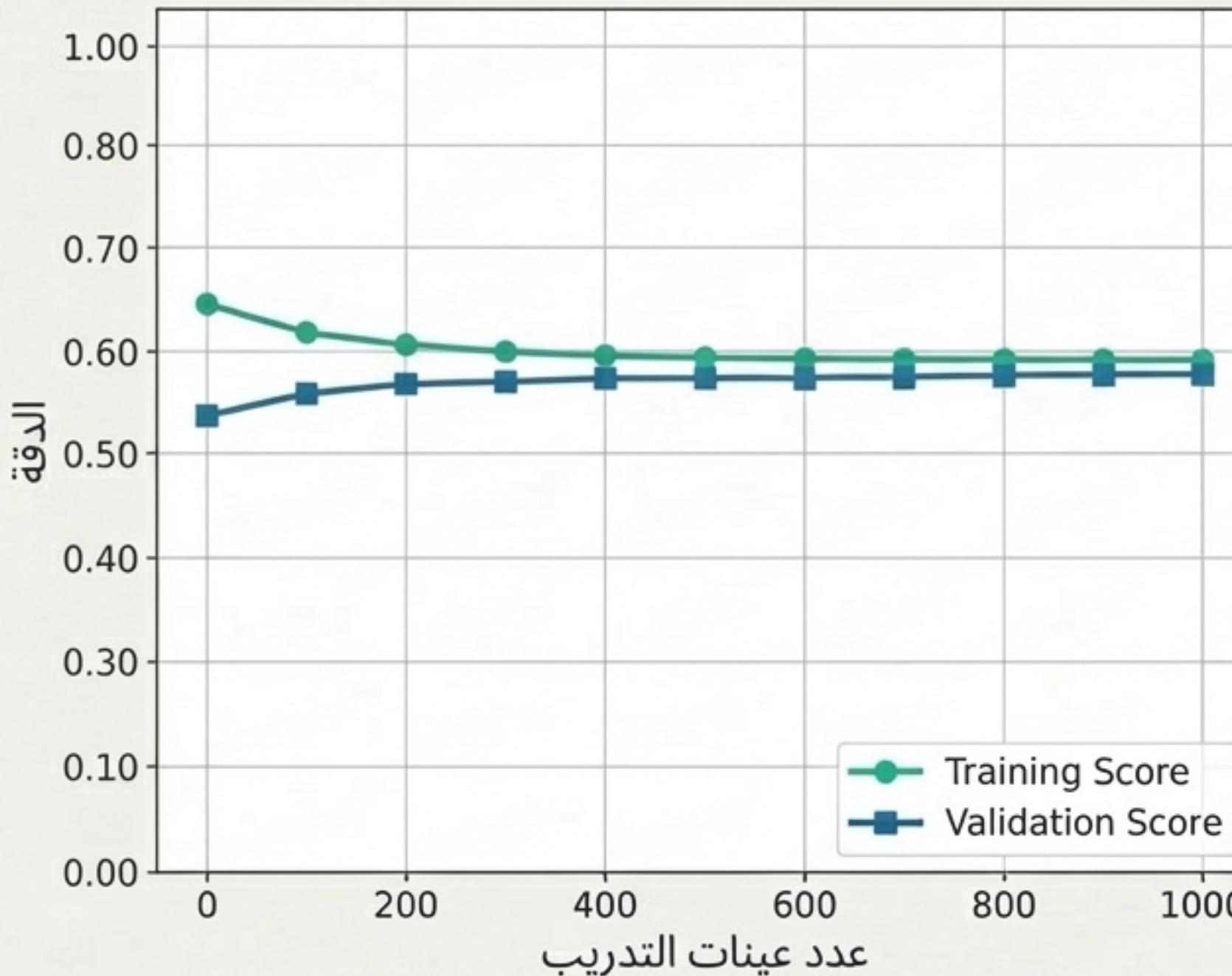
- دقة التدريب (Training accuracy) عالية.
- دقة التحقق (Validation accuracy) عالية أيضاً.
- الفجوة بين المنحنيين صغيرة جداً في النهاية.

التشخيص (The Diagnosis)



- النموذج قوي بما يكفي لتعلم النمط الأساسي في البيانات.
- في نفس الوقت، هو لا يحفظ الضوضاء (noise).
- هذا هو الهدف الذي نسعى إليه.

المشكلة الأولى: التعلم الناقص (Underfitting)



النمط (The Pattern)



- دقة التدريب منخفضة.
- دقة التحقق منخفضة.
- الفجوة بينهما صغيرة.

التشخيص (The Diagnosis)



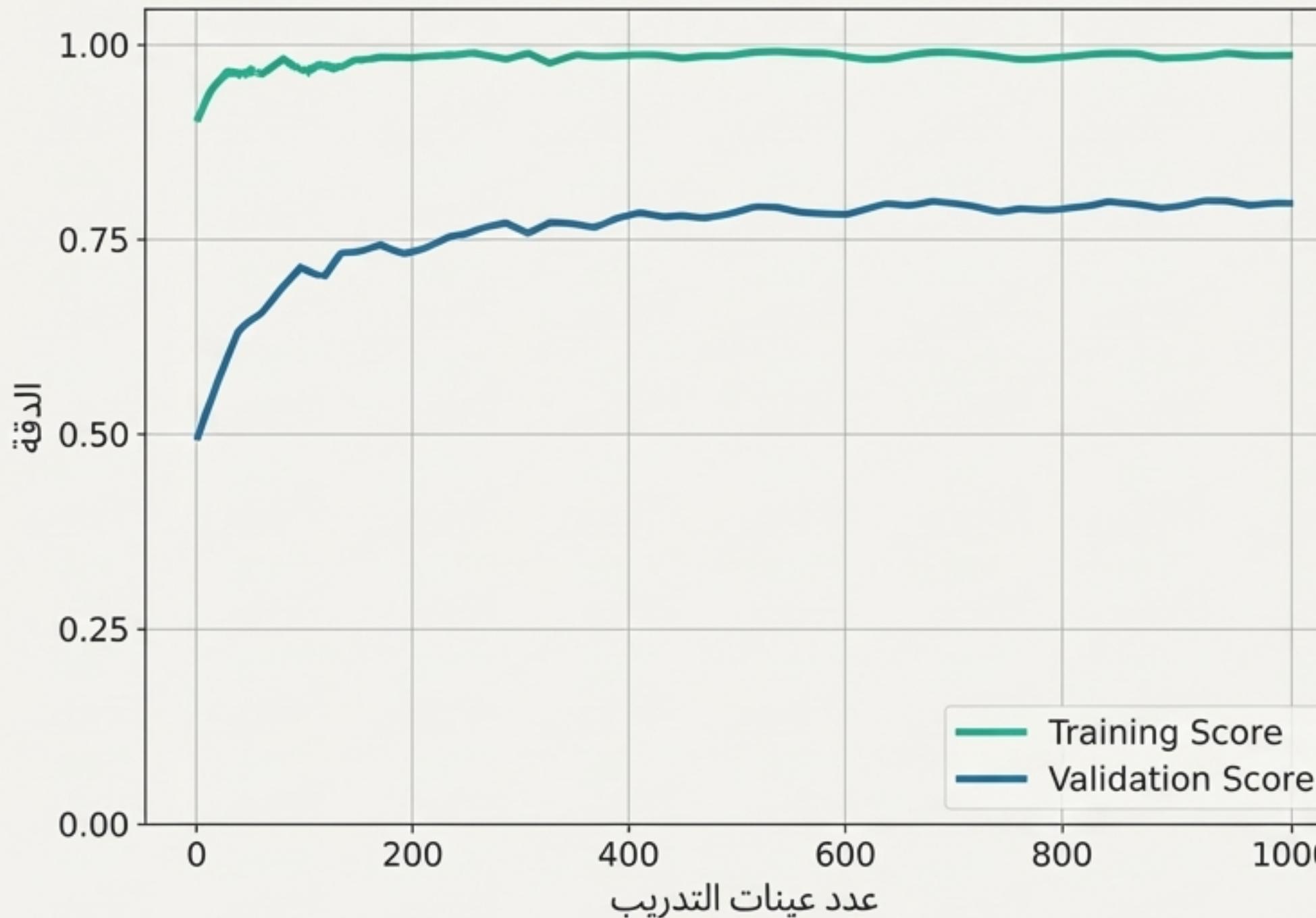
- تحييز عالي (High Bias).
- النموذج بسيط أكثر من اللازم، ولم يتمكن من تعلم النمط حتى من بيانات التدريب.

الحلول المقترحة (Prescription)



- استخدم نموذجاً أكثر تعقيداً (e.g., non-linear model instead of linear.)
- أضف المزيد من الميزات الهندسية (feature engineering).
- قلل من قوة التنظيم (Regularization).

المشكلة الثانية: الإفراط في التعلم (Overfitting)



النمط (The Pattern)

- دقة التدريب عالية جداً (تقرب من 100%).
- دقة التحقق أقل بكثير.
- فجوة كبيرة وواضحة بين المنحنيين.



التشخيص (The Diagnosis)

- تباين عالي (High Variance).
- النموذج 'حفظ' بيانات التدريب بدلاً من 'تعلم' النمط العام.



الحلول المقترحة (Prescription)

- اجمع المزيد من البيانات (إذا أمكن).
- زد من قوة التنظيم (Regularization).
- قلل من تعقيد النموذج (e.g., less depth in a decision tree)



الكود: رسم منحنى التعلم باستخدام `sklearn`

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
from sklearn.linear_model import LogisticRegression

# X, y: Your data
model = LogisticRegression(max_iter=1000)

train_sizes, train_scores, val_scores = learning_curve(
    estimator=model,
    X=X,
    y=y,
    train_sizes=np.linspace(0.1, 1.0, 5),
    cv=5,
    scoring='accuracy',
    shuffle=True,
    random_state=42
)

train_mean = train_scores.mean(axis=1)
val_mean = val_scores.mean(axis=1)

plt.figure(figsize=(8,5))
plt.plot(train_sizes, train_mean, 'o-', label='Training score')
plt.plot(train_sizes, val_mean, 's-', label='Validation score')
plt.xlabel('Number of training samples')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```

شرح الكود: أهم المكونات

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
from sklearn.linear_model import LogisticRegression

# X, y: Your data
model = LogisticRegression(max_iter=1000)

train_sizes, train_scores, val_scores = learning_curve(
    estimator=model,
    X=X,
    y=y,
    train_sizes=np.linspace(0.1, 1.0, 5),
    cv=5,
    scoring='accuracy',
    shuffle=True,
    random_state=42
)

train_mean = train_scores.mean(axis=1)
val_mean = val_scores.mean(axis=1)

plt.figure(figsize=(8,5))
plt.plot(train_sizes, train_mean, 'o-', label='Training score')
plt.plot(train_sizes, val_mean, 's-', label='Validation score')
plt.xlabel('Number of training samples')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()

```

الدالة الأساسية من `sklearn` التي تقوم بكل العمل.

تحديد نقاط حجم البيانات التي سيتم اختبارها (هنا من 10% إلى 100%).

استخدام التحقق المتقاطع (Cross-Validation) من 5 طيات للحصول على نتائج أكثر استقراراً.

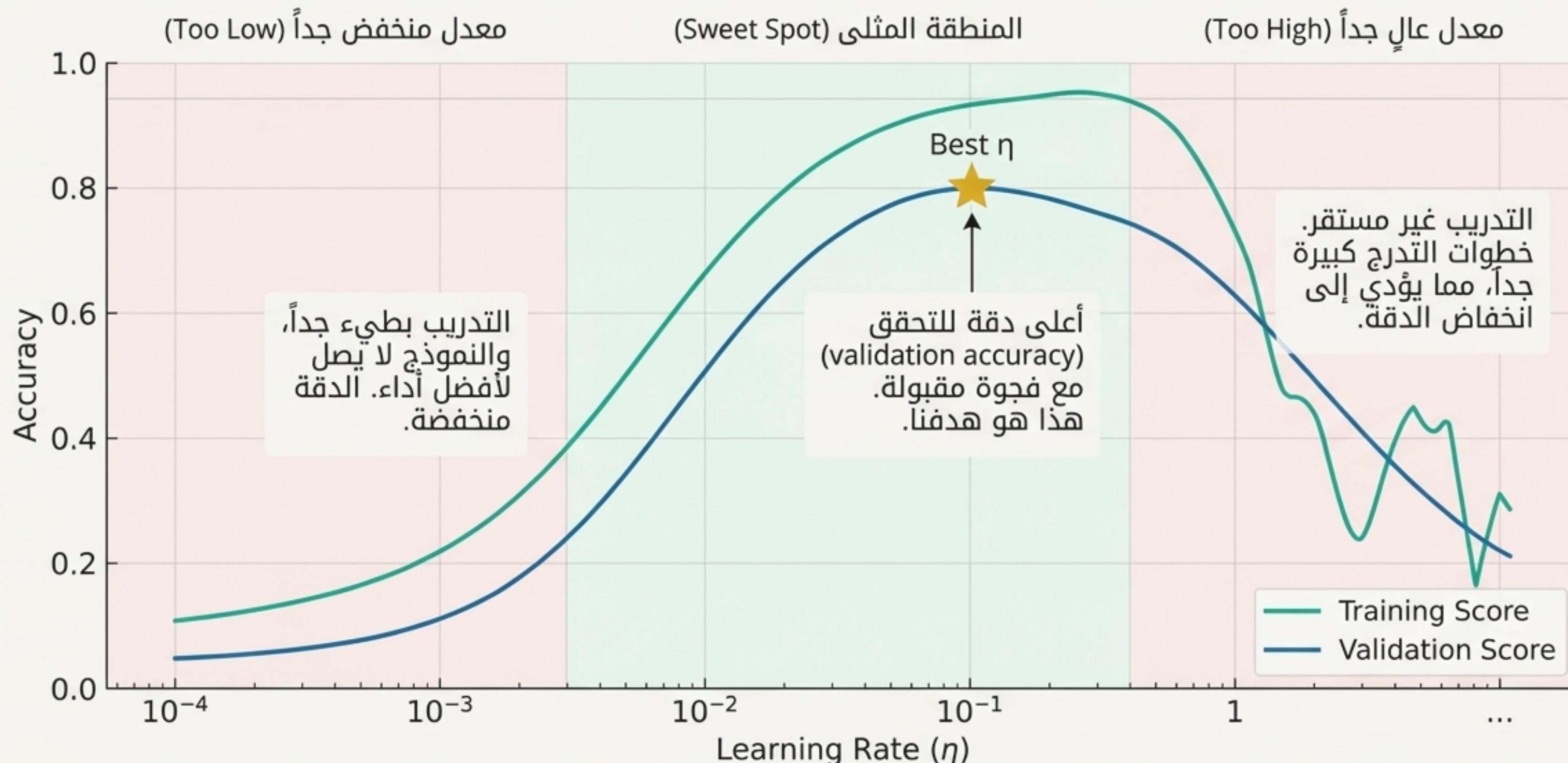
نحسب متوسط الأداء عبر الطيات الخمس للحصول على خط واحد لكل من التدريب والتحقق.



الأداة الثانية: منحنى معدل التعلم

إيجاد السرعة المثلث لتدريب نموذجك

قراءة منحنى معدل التعلم: البحث عن النقطة المثلثى



الكود: رسم منحنى معدل التعلم باستخدام `sklearn`

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import validation_curve

model = SGDClassifier(loss='log_loss', learning_rate='constant', max_iter=1000)
# Define the range of learning rates to test
param_range = np.logspace(-4, -2, 5) # From 1e-4 to 1e-2

train_scores, val_scores = validation_curve(
    estimator=model,
    X=X,
    y=y,
    param_name='eta0', # The parameter name for learning rate in SGDClassifier
    param_range=param_range,
    cv=5,
    scoring='accuracy'
)

train_mean = train_scores.mean(axis=1)
val_mean = val_scores.mean(axis=1)

plt.figure(figsize=(8,5))
plt.semilogx(param_range, train_mean, 'o-', label='Training score')
plt.semilogx(param_range, val_mean, 's-', label='Validation score')
plt.xlabel('Learning rate (eta0)')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True, which="both")
plt.show()
```

شرح الكود: فهم `validation_curve`

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import validation_curve

model = SGDClassifier(loss='log_loss', learning_rate='constant', max_iter=1000)
# Define the range of learning rates to test
param_range = np.logspace(-4, -2, 5) # From 1e-4 to 1e-2

train_scores, val_scores = validation_curve(
    estimator=model,
    X=X,
    y=y,
    param_name='eta0', # The parameter name for learning rate in SGDClassifier
    param_range=param_range,
    cv=5,
    scoring='accuracy'
)

train_mean = train_scores.mean(axis=1)
val_mean = val_scores.mean(axis=1)

plt.figure(figsize=(8,5))
plt.semilogx(param_range, train_mean, 'o-', label='Training score')
plt.semilogx(param_range, val_mean, 's-', label='Validation score')
plt.xlabel('Learning rate (eta0)')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True, which="both")
plt.show()

```

دالة عامة لاختبار أي متغير فائق (hyperparameter) معدل التعلم.

اسم المتغير الذي نريد اختباره. يختلف من مصنف لآخر.

ختار نطاقاً لوغاريتmic لأنه الخيار الأفضل عند اختبار معدلات التعلم.

استخدام مقياس لوغاريتمي للمحور السيني (X-axis) لعرض النتائج بوضوح.

الملخص: أي أداة تستخدم ومتى؟

منحنى التعلم (Learning Curve)

السؤال الذي يجب عليه: هل نموذجي بسيط جداً أم معقد جداً؟ (Is my model too simple? or too complex?)

المحور السيني (X-Axis): عدد عينات التدريب

الهدف الأساسي: تشخيص مشكلة التحييز مقابل التباين (Bias vs. Variance).

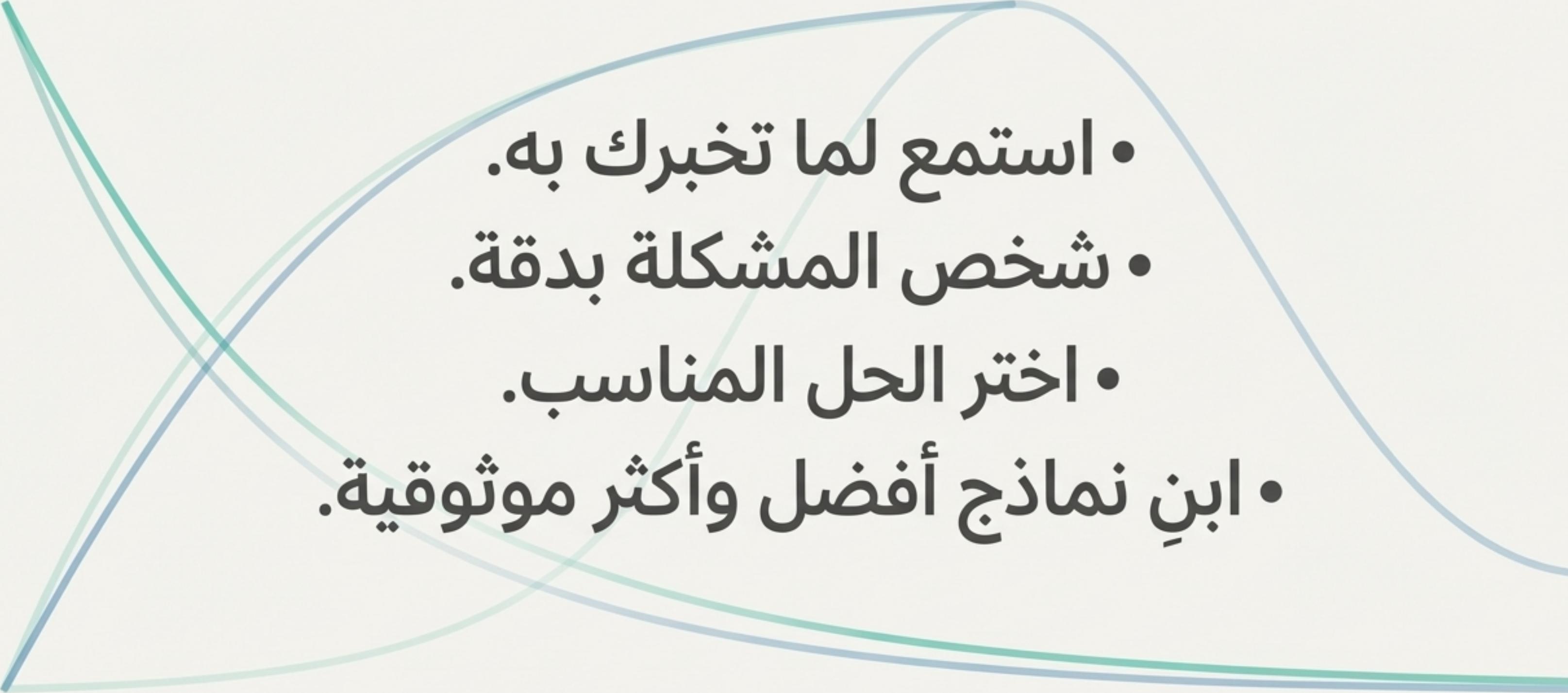
منحنى معدل التعلم (Learning Rate Curve)

السؤال الذي يجب عليه: ما هي السرعة المثلث لتدريب نموذجي؟ (What is the optimal speed to train my model?)

المحور السيني (X-Axis): قيمة معدل التعلم (η)

الهدف الأساسي: ضبط متغير فائق مهم لضمان تدريب مستقر وفعال.

هذه المنحنيات ليست مجرد رسومات، إنها حوار مع نموذجك

- 
- استمع لما تخبرك به.
 - شخص المشكلة بدقة.
 - اختر الحل المناسب.
 - ابني نماذج أفضل وأكثر موثوقية.