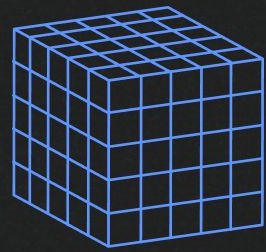
Заняття 8.
Beyond linear regression

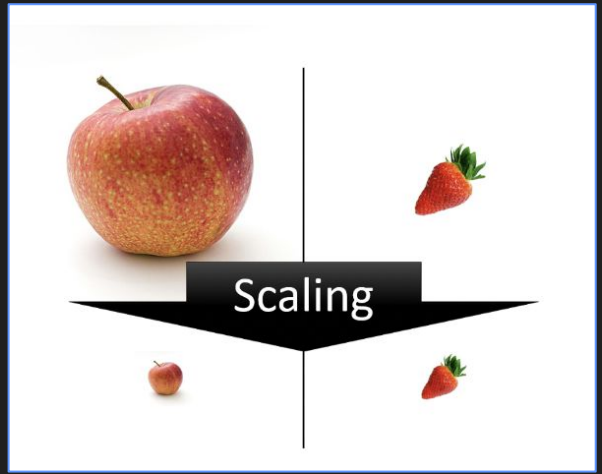
План заняття

- Масштабування змінних
- Перенавчання та недонавчання (overfitting & underfitting)
- Регуляризація
- Модель багатомірної лінійної регресії
- Поліноміальна регресія



Масштабування змінних (Feature scaling)

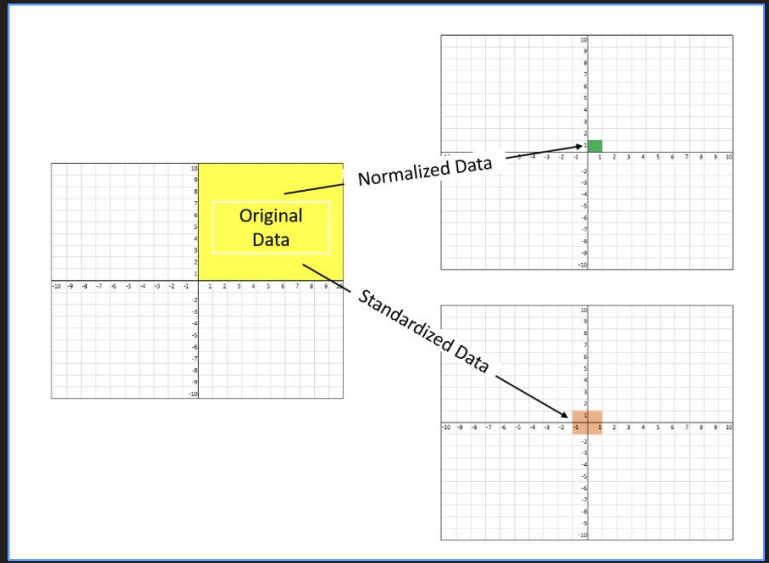
Машинне навчання схоже на приготування соку «мультифрукт». Якщо ми хочемо отримати найкращий змішаний сік, нам потрібно змішувати всі фрукти не за розміром, а в правильній пропорції. Нам просто треба пам’ятати, що яблуко та полуниця неоднакові, якщо ми не зробимо їх подібними в якомусь контексті, щоб порівняти їхні властивості. Подібним чином у багатьох алгоритмах машинного навчання, щоб розглядати всі змінні в рівних умовах, нам потрібно виконати **масштабування**, щоб одне значне число не впливало на модель лише через свою величину.



Масштабування змінних (Feature scaling)

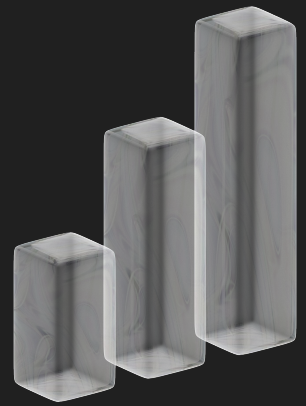
Найпоширенішими методами масштабування ознак є **нормалізація** та **стандартизація**.

Нормалізацію використовують, коли хочемо обмежити наші значення двома числами, переважно [0,1] або [-1,1]. **Стандартизація** ж перетворює дані так, щоб вони мали нульове середнє значення та дисперсію, рівну одиниці. Так дані стають безрозмірними.



Why do we need scaling?

- Багато алгоритмів працюють або працюють краще, якщо дані масштабовані
- Багато алгоритмів збігаються швидше, коли дані масштабовані



Why do we need scaling?

Алгоритм машинного навчання бачить лише число. Якщо є велика різниця в діапазоні, скажімо, одна фіча в діапазоні тисяч, а інша в діапазоні десятків, він робить основне припущення, що більші числа мають певну перевагу. Таким чином, ця більша за діапазоном змінна починає відігравати більш вирішальну роль під час навчання моделі.

Припустимо, що ми маємо дві змінні: вага та ціна. Ми, люди, розуміємо, що «Вагу» не можна порівнювати з «Ціною», але для алгоритму обидві змінні — лише числа. Отже, алгоритм припущення робить, що, оскільки «Вага» > «Ціна», отже, «Вага» важливіша за «Ціну».

Name	Weight	Price
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5

Why do we need scaling?

Ми можемо пришвидшити градієнтний спуск за допомогою масштабування, оскільки градієнт швидко спадає на малих діапазонах і повільно — на великих діапазонах, і коливається неефективно до оптимуму, коли змінні дуже нерівномірні.

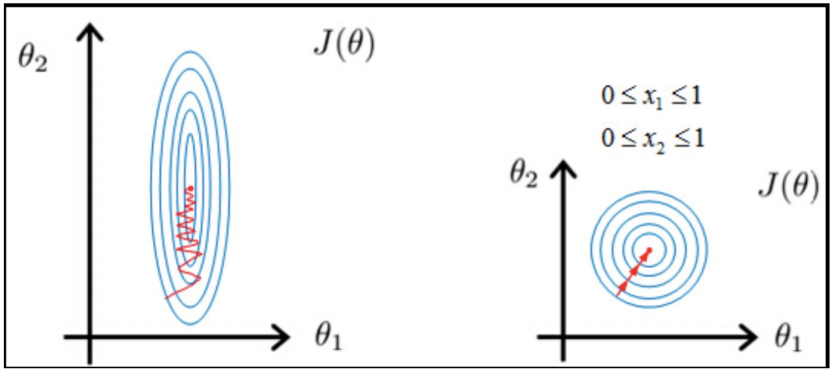


image from Stackoverflow Explaining Gradient Descent Scaled VS Unscaled

When to do scaling?

Масштабування змінних є важливим для алгоритмів машинного навчання, які **обчислюють відстані** між даними. Якщо не масштабувати, функція з вищим діапазоном значень починає домінувати під час обчислення відстаней.

У багатьох алгоритмах, коли ми хочемо **швидшої збіжності**, масштабування ОБОВ'ЯЗКОВЕ, як у нейронних мережах.

Алгоритми, які **не потребують нормалізації/масштабування**, покладаються на правила. На них не вплинуть жодні монотонні перетворення змінних. Масштабування — це монотонне перетворення. Прикладами алгоритмів у цій категорії є всі деревоподібні алгоритми: CART, Random Forests, Gradient Boosted Decision Trees. Ці алгоритми використовують правила (серії нерівностей) і не потребують нормалізації.

When to do scaling?

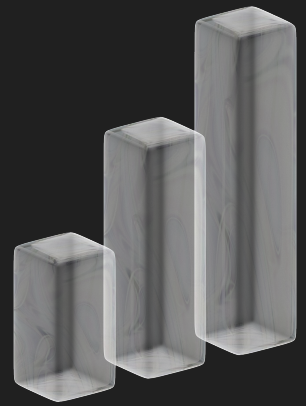
Sr No.	Algorithms	Feature Scaling
1.	Linear/Non-Linear Regressions	Yes
2.	Logistic Regression	Yes
3.	KNN	Yes
4.	SVM	Yes
5.	Neural Networks	Yes
6.	K-means clustering	Yes
7.	CART	No
8.	Random Forests	No
9.	Gradient Boosted Decision Trees	No
10.	Naïve Bayes	N0
11.	PCA	Yes
12.	SVD	Yes
13.	Factorization Machines	Yes

Fig: Feature Scaling requires based on Machine Learning

Feature scaling methods

Існує досить велика кількість методів масштабування змінних, ми розглянемо три найбільш популярні:

- ➔ Min-Max Scaler
- ➔ Standard Scaler
- ➔ Robust Scaler



Min-Max Scaler

Фактично звужує діапазон таким чином, що значення змінної варіюється від 0 до 1 (або від -1 до 1, якщо є від’ємні значення).

Цей метод працює краще у випадках, якщо розподіл змінної не є нормальним або стандартне відхилення дуже мале.

Однак він **чутливий до викидів**.

$$x_{new} = \frac{x - min(x)}{max(x) - min(x)}$$

Standard Scaler

Standard Scaler передбачає, що дані нормально розподіляються в межах кожної змінної, і масштабує їх таким чином, щоб середнє розподілу приблизно дорівнювало 0, а стандартне відхилення — 1.

Центрування та масштабування відбуваються незалежно для кожної фічі шляхом обчислення відповідних статистичних даних для навчальних прикладів. Якщо дані розподілені не нормально, Standard Scaler не є найкращим вибором.

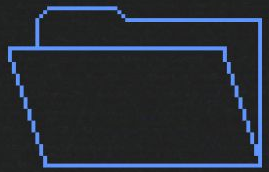
$$x_{new} = \frac{x - \bar{x}}{\sigma}$$

Robust Scaler

Як випливає з назви, цей масштабувальник стійкий до викидів. Якщо наші дані містять багато викидів, масштабування за допомогою середнього значення та стандартного відхилення даних не працюватиме належним чином.

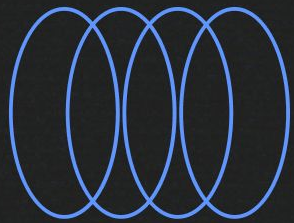
Формула центрування та масштабування цього масштабувальника базується на процентилях і, отже, на них не впливають кілька чисел великих граничних викидів. Зауважте, що самі викиди ще наявні в перетворених даних.

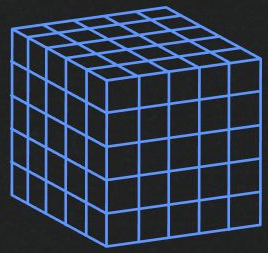
$$x_{new} = \frac{x - Q_1(x)}{Q_3(x) - Q_1(x)}$$



???

Q&A





Live coding / Практика

Generalization

Головна мета навчання з учителем полягає в тому, щоб використовувати навчальні дані для побудови моделі, яка зможе робити точні прогнози на основі нових, невідомих даних, які мають ті самі характеристики, що й початковий навчальний набір.

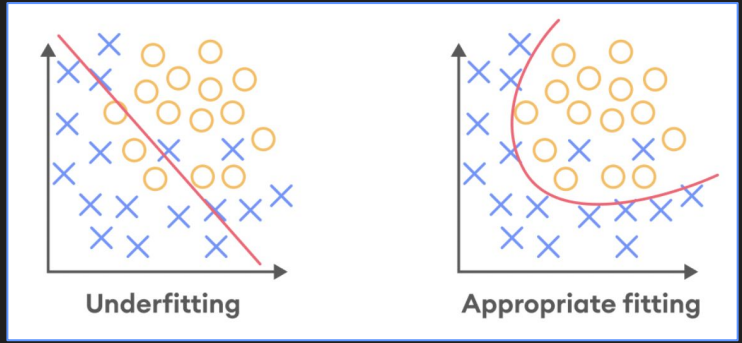
Це відомо як узагальнення (generalization). Узагальнення стосується того, наскільки ефективно концепції, вивчені моделлю машинного навчання, застосовують до конкретних прикладів, які не використовували протягом навчання. Ви хочете створити модель, яка може узагальнювати якомога точніше.

Однак у реальному світі це складна проблема.

Underfitting

Недонавчання виникає, коли модель не може робити точні прогнози на основі навчальних даних і, отже, не має можливості добре узагальнювати нові дані. Інший випадок недостатнього пристосування — це коли модель не в змозі отримати достатньо інформації з навчальних даних, що ускладнює охоплення домінантної тенденції (модель не може створити відтворення між вхідною та цільовою змінними).

Моделі машинного навчання у разі недонавчання здебільшого мають велику похибку як на навчальній вибірці (train set), так і на тестовій (test set).
Underfitting models usually have high bias and low variance.

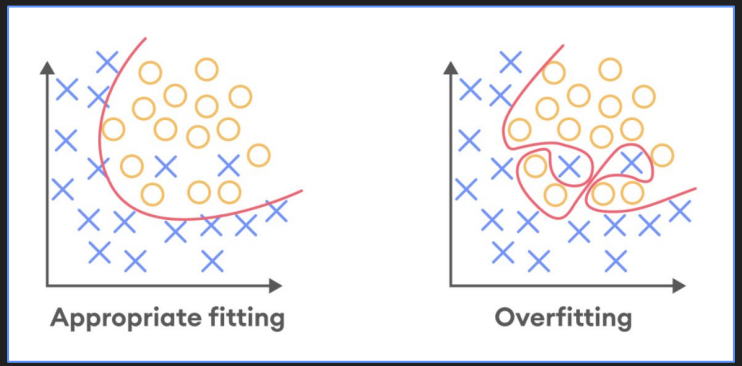


How to avoid underfitting

- Тренуйте більш складну модель
- Спробуйте іншу модель (більш потужну з більшою кількістю параметрів, ансамблі)
- Більше часу на навчання
- Усунення шуму з даних
- Налаштування параметрів регуляризації (зменшення регуляризації)

Overfitting

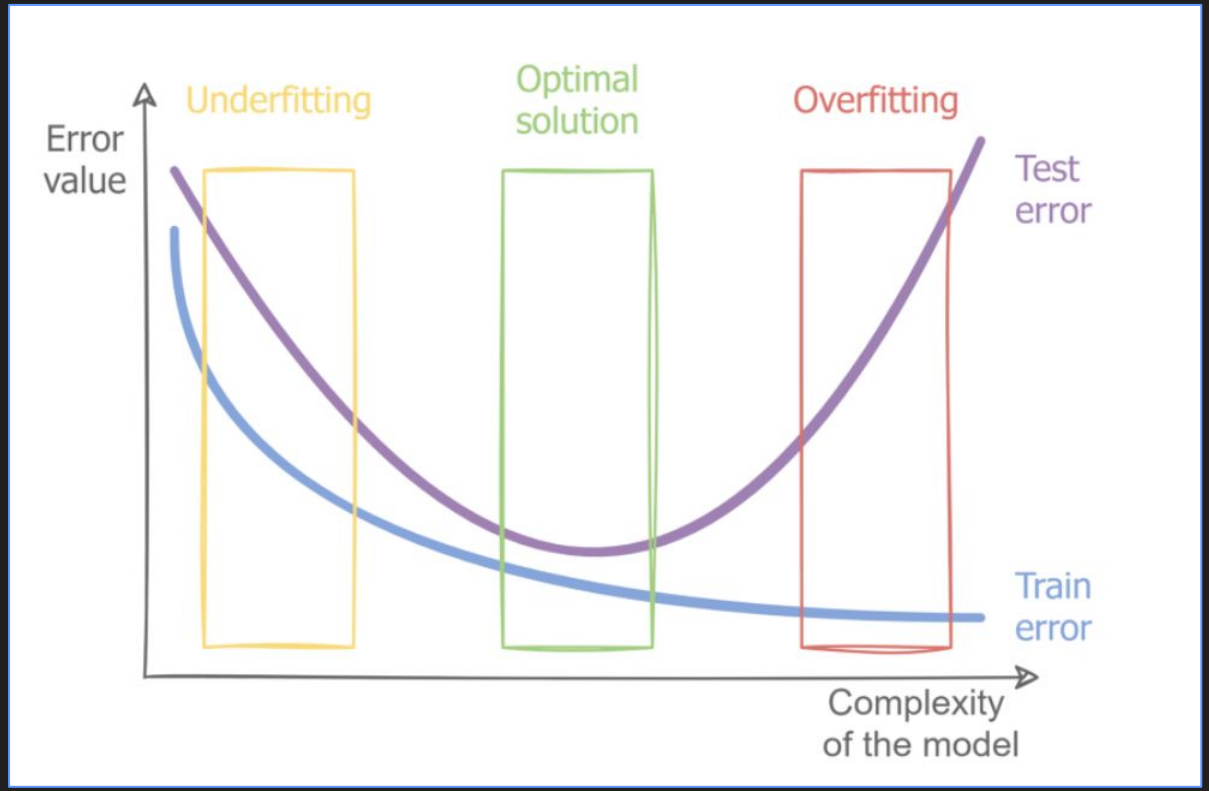
Модель вважають перенавченою, якщо вона дуже добре працює з навчальними даними, але не працює на тому самому рівні з тестовими. Перенавчена модель не може добре узагальнити, оскільки вона вивчає шум і шаблони навчальних даних до тієї міри, де це негативно впливає на продуктивність моделі на нових даних. Якщо модель перенавчена, навіть незначна зміна вихідних даних спричинить значну зміну моделі. **Models that are overfitting usually have low bias and high variance.**



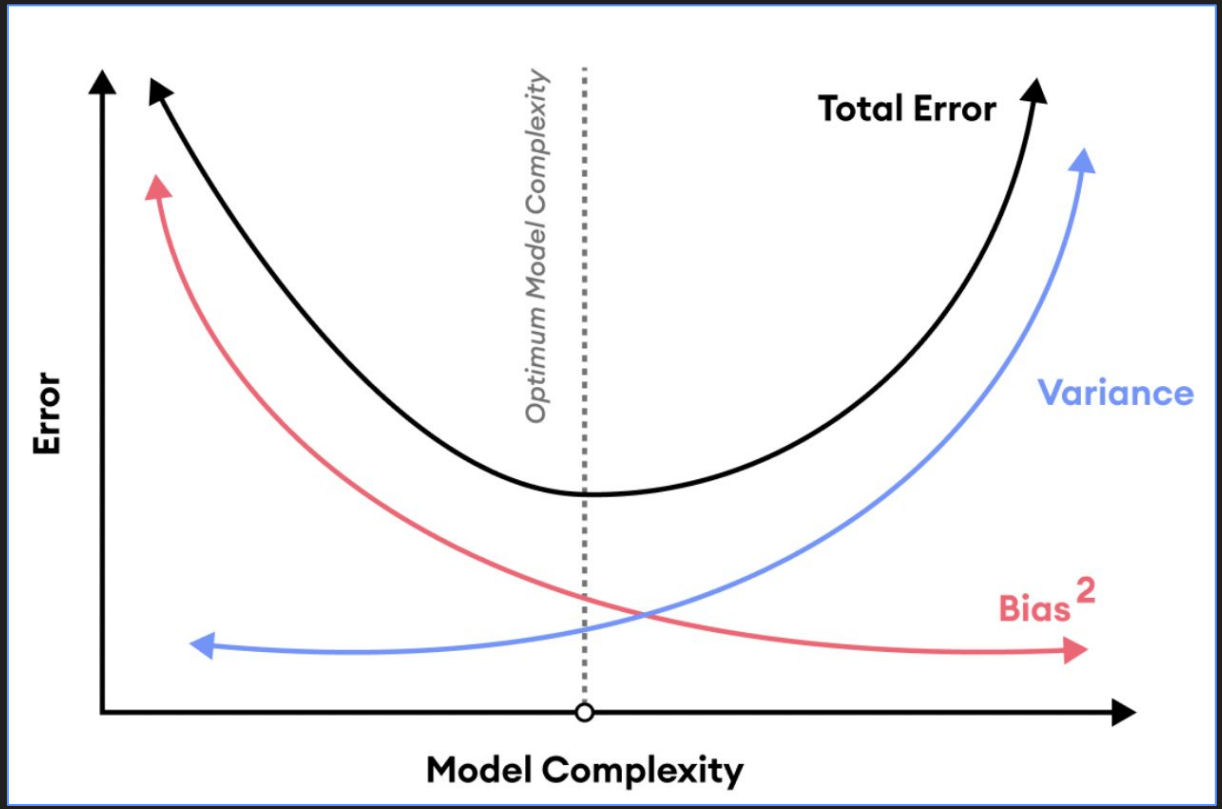
How to avoid overfitting

- Внесення додаткових даних
- Рання зупинка (для ітераційних алгоритмів)
- Аугментація даних — Data augmentation (збільшити обсяг даних, дещо змінивши раніше наявні дані та додавши нові точки даних або створивши синтетичні дані з попередньо наявного набору даних)
- Видалення ознак — Feature selection
- Регуляризація
- Простіша модель

Overfitting/Underfitting (High variance/High bias)

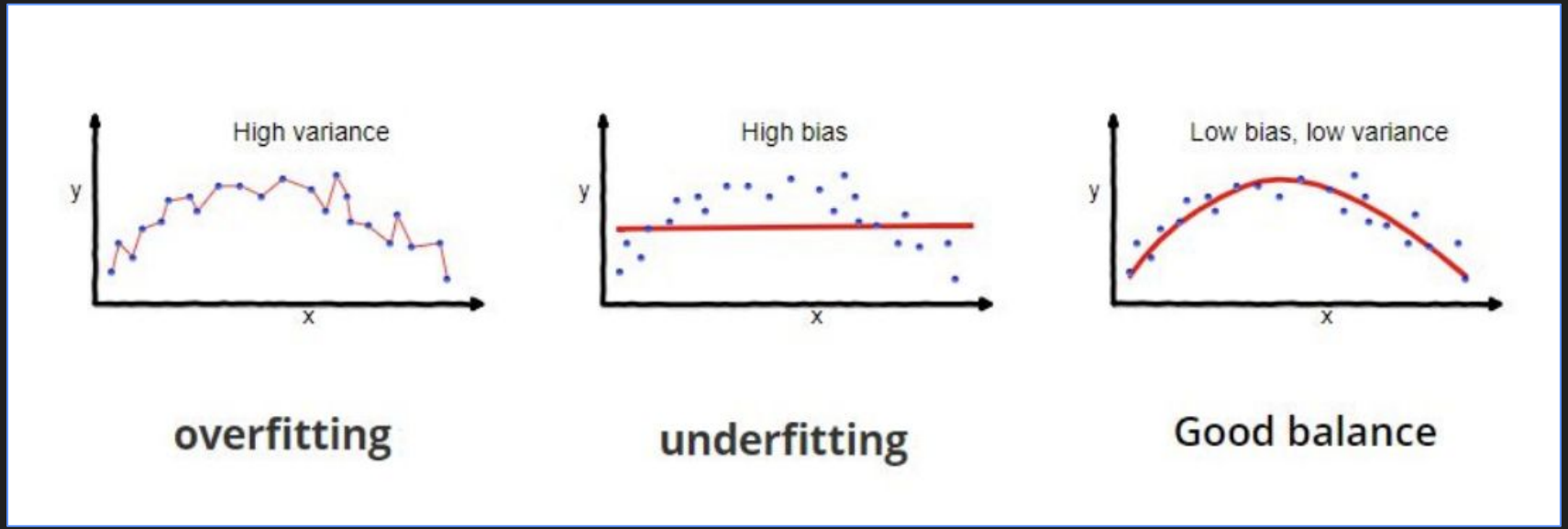


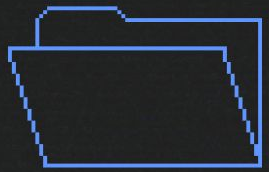
Overfitting/Underfitting (High variance/High bias)



Overfitting/Underfitting (High variance/High bias)

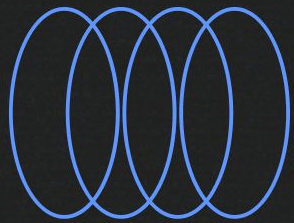
Проблема пошуку балансу між bias і variance є однією з важливих в ML.

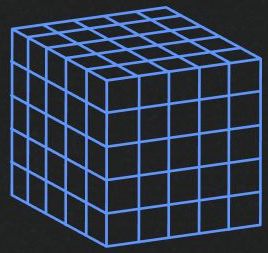




???

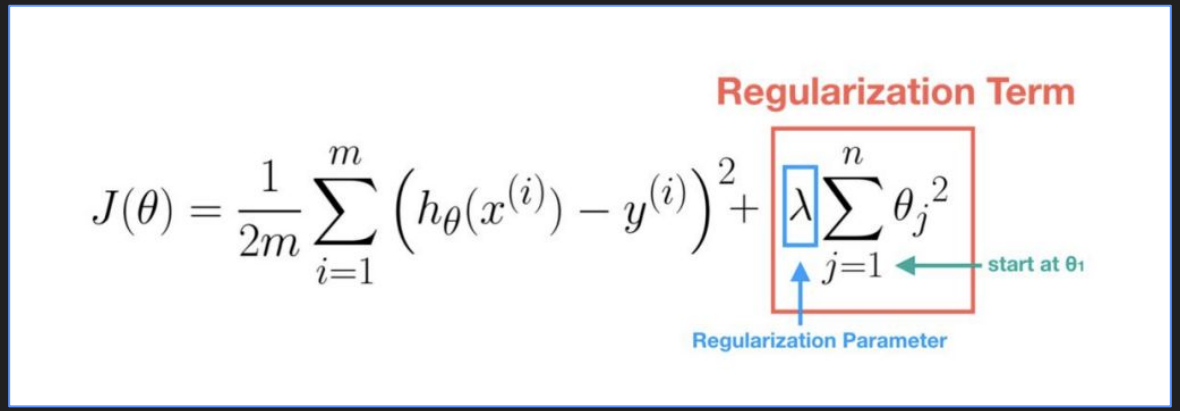
Q&A





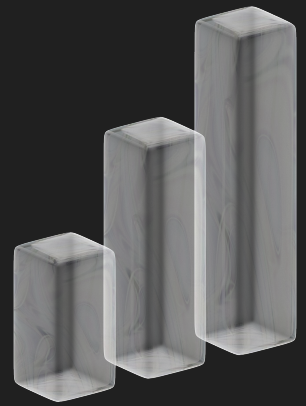
Regularization

Це форма регресії, яка обмежує/регулює або звужує оцінки коефіцієнтів до нуля. Іншими словами, ця техніка не заохочує вивчати більш складну або гнучку модель, щоб уникнути ризику перенавчання.
Фактично регуляризація додає штраф, коли складність моделі зростає. Параметр регуляризації (лямбда) штрафує всі параметри, окрім вільного члена (intercept), щоб модель узагальнювала дані та не перенавчалася.



Основні види регуляризації

- L2 regularization (Ridge Regression)
- L1 regularization (Lasso)
- Elastic Net (L1 and L2 combination)



L2 Regularization (Ridge regression)

Ridge regression додає squared magnitude коефіцієнта як штраф до функції втрат.

cost function → $J(\theta) = \sum_{i=1}^{m}(y_i - \theta_0 - \sum_{j=1}^{n} \theta_j x_{ij})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$ ← Regularization term

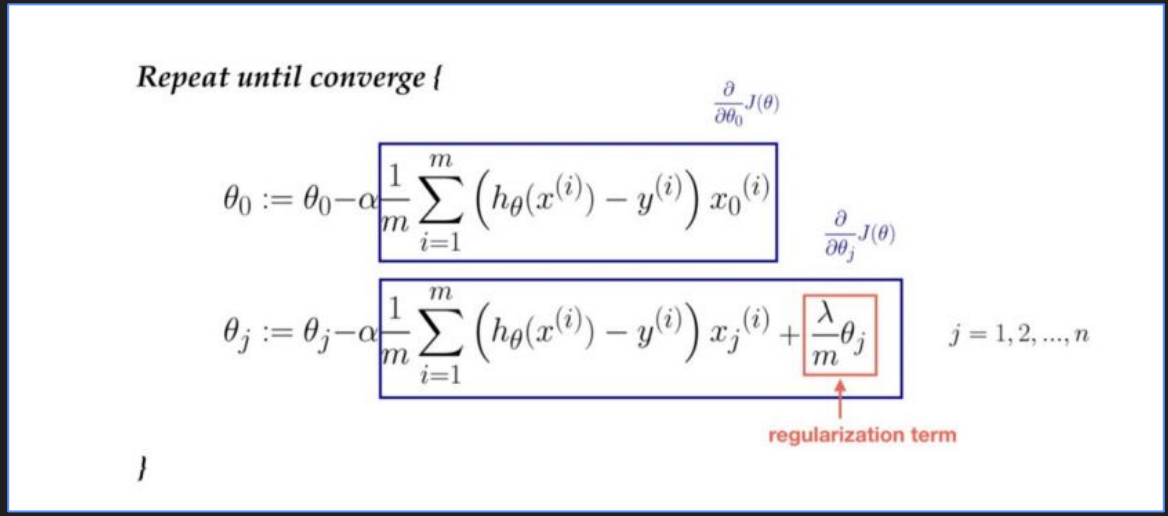
m - number of examples in training set

n - number of features

Отже, тепер для визначення коефіцієнтів регресії нам треба мінімізувати саме цю функцію.

L2 Regularization (Ridge regression)

Алгоритм градієнтного спуску у випадку Ridge regression матиме такий вигляд:



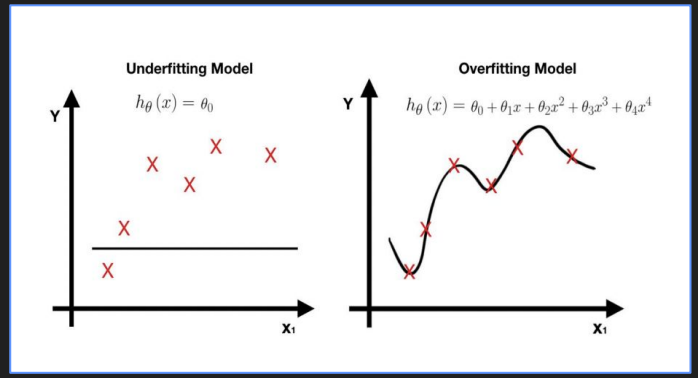
L2 Regularization (Ridge regression)

Параметр регуляризації λ

$$J(\theta) = \sum_{i=1}^{m}(y_i - \theta_0 - \sum_{j=1}^{n} \theta_j x_{ij})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

Якщо λ занадто велике, модель стане занадто простою і може виникнути недонавчання.

Якщо λ рівне нулю або занадто мале, вплив регуляризації буде незначним і може виникнути перенавчання.



λ — гіперпараметр моделі. Ми не навчаємо λ у процесі тренування, а задаємо перед навчанням. Щоб обрати оптимальний, ми можемо задати сітку зі значень λ (наприклад, [0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.5]), навчити з кожним значенням модель і вибрати ту, в якої краще значення помилки на валідаційних даних.

L1 Regularization (Lasso)

cost function →

$$J(\theta) = \sum_{i=1}^{m}(y_i - \theta_0 - \sum_{j=1}^{n}\theta_j x_{ij})^2 + \lambda \sum_{j=1}^{n}|\theta_j|$$

← Regularization term

m - number of examples in training set

n - number of features

Зрозуміло, що ця варіація регуляризації відрізняється від L2 regularization лише штрафом за високі коефіцієнти. Lasso використовує $|\theta_j|$(модуль) замість квадратів як штраф. У статистиці це відомо як норма L1.

L1 vs L2 Regularization

Тоді, згідно з наведеним вище формулюванням, Ridge regression виражається як (як приклад, маємо лише дві фічі): $\theta_1^2 + \theta_2^2 <= s$

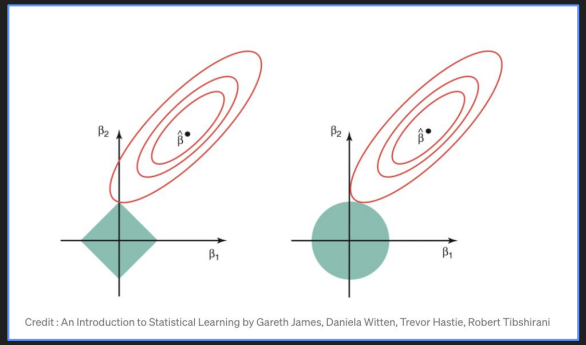
Це означає, що коефіцієнти Ridge regression мають найменше значення функції втрат для всіх точок, які лежать у межах кола, визначеного:

$$\theta_1^2 + \theta_2^2 <= s$$

Аналогічно, для Lasso рівняння набуває вигляду:

$$|\theta_1| + |\theta_2| <= s$$

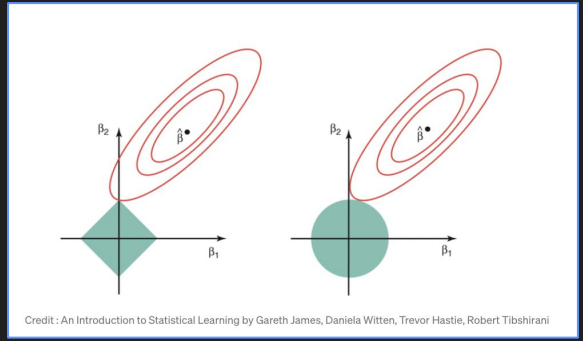
Це означає, що коефіцієнти Lasso мають найменше значення функції втрат для всіх точок, які лежать усередині ромба, визначеного: $|\theta_1| + |\theta_2| <= s$



Credit : An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

L1 vs L2 Regularization

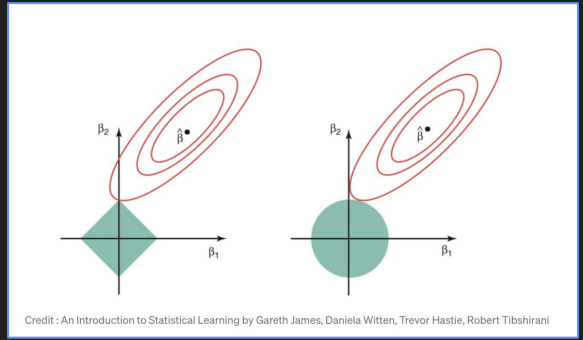
У цьому разі оцінки коефіцієнтів **lasso** та **ridge regression** визначаються першою точкою, в якій еліпс стикається з областю обмежень. Оскільки ridge regression має кругове обмеження без гострих точок, це перетин зазвичай не відбуватиметься на осі, і тому оцінки коефіцієнта **ridge regression** будуть винятково ненульовими. Однак обмеження **lasso** має кути на кожній з осей, тому еліпс часто перетинатиме область обмеження на осі. Коли це станеться, **один із коефіцієнтів дорівнюватиме нулю**. У вищих вимірах (де параметрів набагато більше, ніж 2) багато оцінок коефіцієнтів можуть дорівнювати нулю одночасно.



Credit : An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

L1 vs L2 Regularization

Це проливає світло на очевидний **недолік ridge regression**, який полягає в інтерпретованості моделі. Це скоротить коефіцієнти для найменш важливих предикторів, дуже близькі до нуля. Але це ніколи не зробить їх точно нульовими. Іншими словами, **остаточна модель міститиме всі предиктори**. Однак у випадку Lasso пенальті L1 змушує деякі оцінки коефіцієнта точно дорівнювати нулю, коли параметр налаштування λ є достатньо великим. Таким чином, **метод Lasso також виконує вибір змінних (feature selection)** і, як кажуть, виробляє розріджені моделі.



Credit : An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

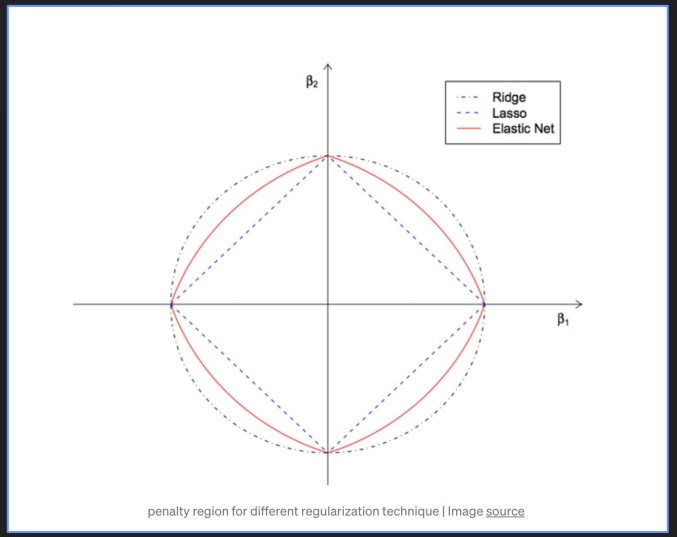
Elastic Net

Elastic Net — це комбінація L1 і L2 регуляризацій

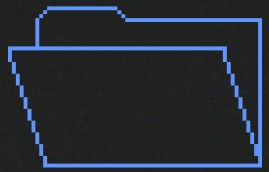
$$J(\theta) = \sum_{i=1}^{m}(y_i - \theta_0 - \sum_{j=1}^{n} \theta_j x_{ij})^2 + \lambda_1 \sum_{j=1}^{n} |\theta_j| + \lambda_2 \sum_{j=1}^{n} \theta_j^2$$

m - number of examples in training set

n - number of features

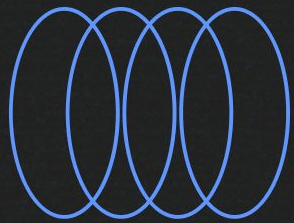


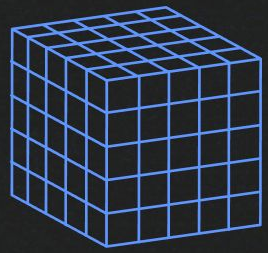
penalty region for different regularization technique | Image source



???

Q&A





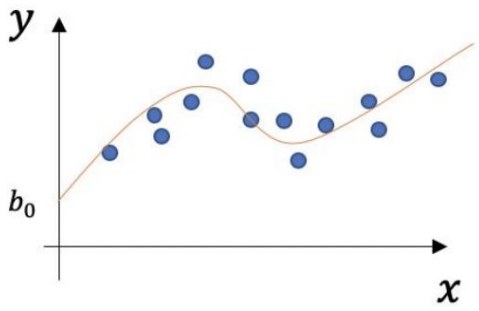
Поліноміальна регресія

Суть поліноміальної регресії полягає в тому, що в ролі додаткових фіч додають початкові у визначеному степені.

$$y = b_0 + b_1 x + b_2 x^2 + \cdots$$

where,

y: dependent variable
b_*: coefficients
x: Independent variable



Поліноміальна регресія

Regression

Simple Linear Regression	$y = b_0 + b_1 x_1$
Multiple Linear Regression	$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$
Polynomial Linear Regression	$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$

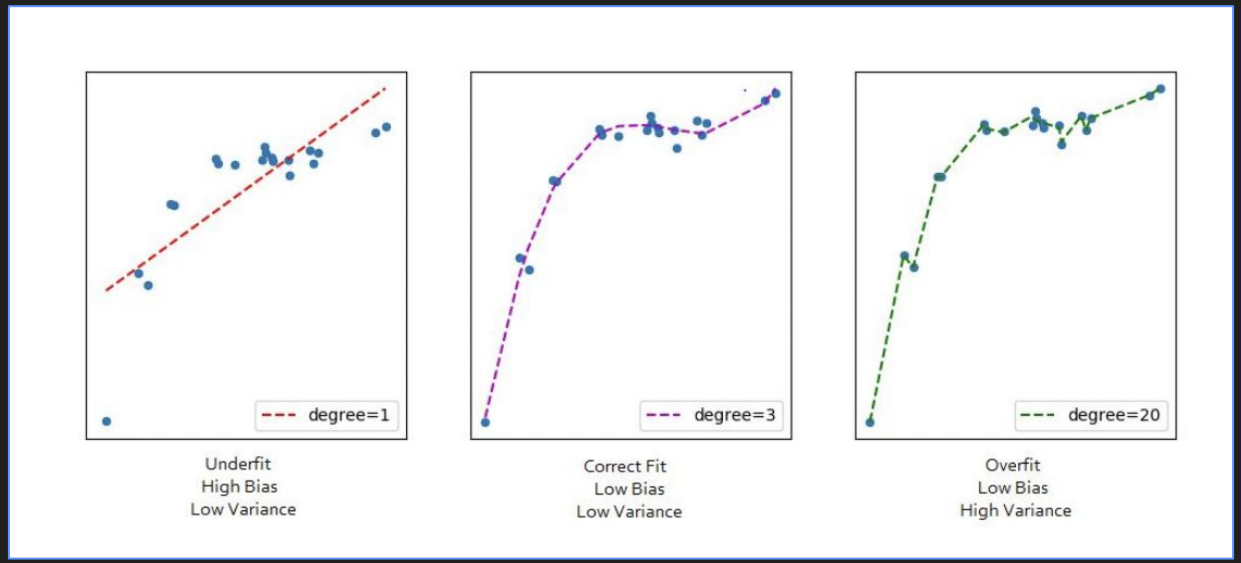
Поліноміальна регресія (multivariate)

Якщо фіч більше, ніж одна, то формула поліноміальної регресії матиме перехресні добутки фіч. Такий вигляд має формула поліноміальної регресії у випадку двох фіч:

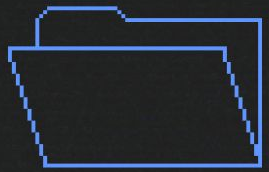
$$y = b_0 + b_1 * (x_1) + b_2 * (x_2) + b_3 * (x_1)^2 + b_4 * (x_2)^2 + b_5 * (x_1 * x_2)$$

Поліноміальна регресія

Варто бути обережними в роботі з високими степенями, адже такі моделі схильні до перенавчання.



Live coding / Практика



???

Q&A