ARTIFICIAL INTELLIGENCE

SARAH ALKASASBEH

POWERUNIT

Artificial Intelligence

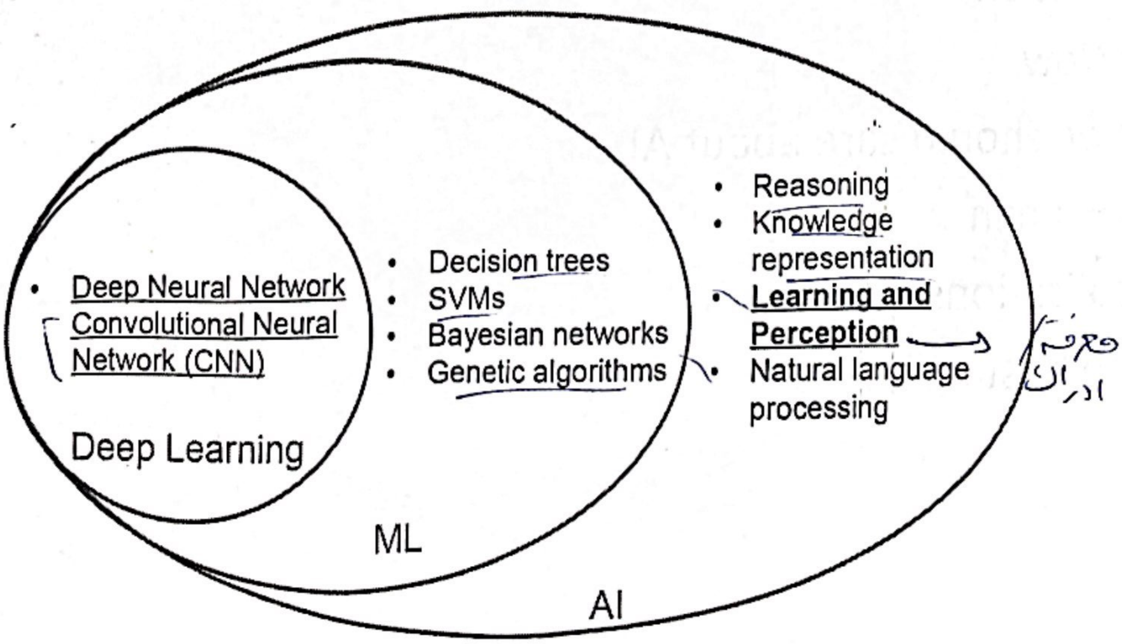
- **Artificial Intelligence**: Build Machines which are capable of thinking like humans (mimic human behavior)
  - if-then statements programmed by experts

- **Machine Learning**: Give computers the ability to learn/make decisions without being explicitly programmed to do so
  - Adjust themselves in response to the data they're exposed t

عكس Power ال

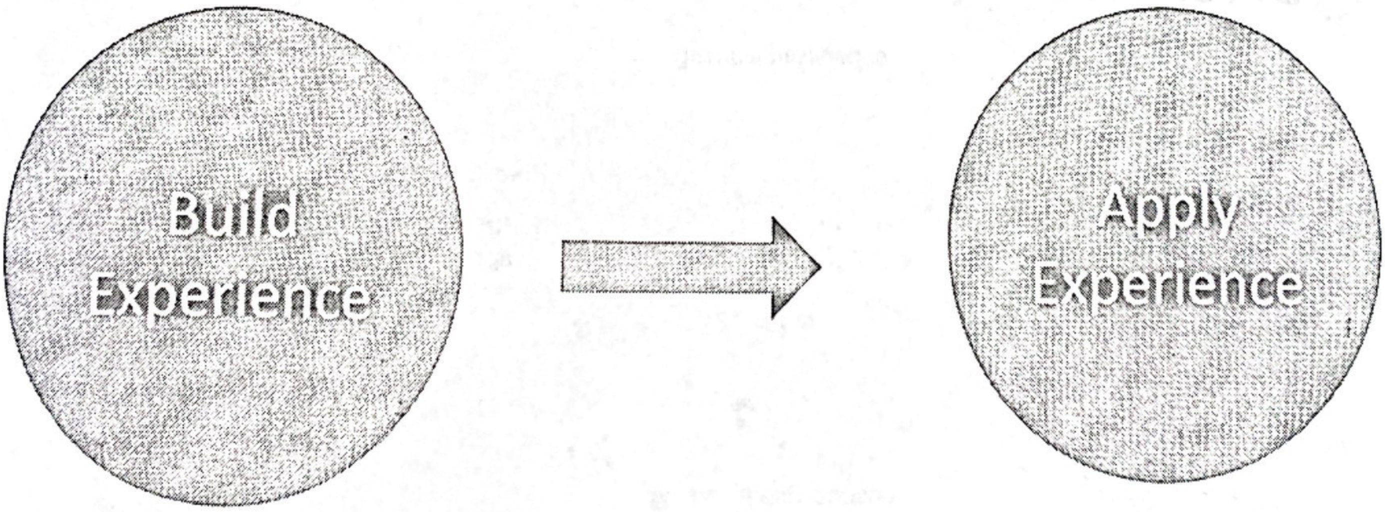
- **Deep Learning:** Using Neural Networks to solve complex problems; Automatically discover patterns for feature detection

AI Vs ML Vs Deep Learning



https://youtu.be/HcqpanDadyQ

Machine Learns Like the Human



Examples (Training)

Learning Mechanism (ML Algorithms)

Knowledge (Model)

ML Algorithms

مثلا بنعطي الصورة معاها label ومعلومات

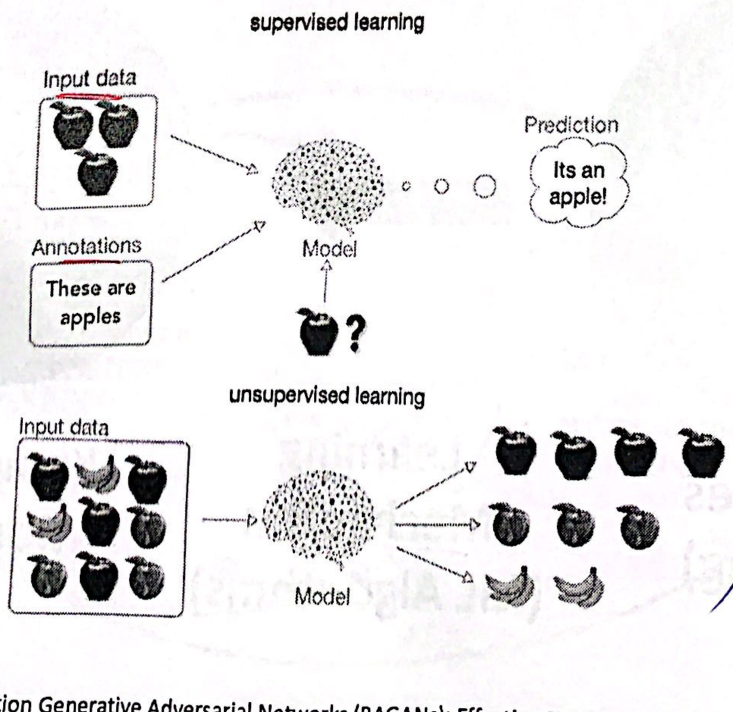
بنعطي inputs بدون labels او معلومات

Supervised Learning

UnSupervised Learning

بتعلم من ال label

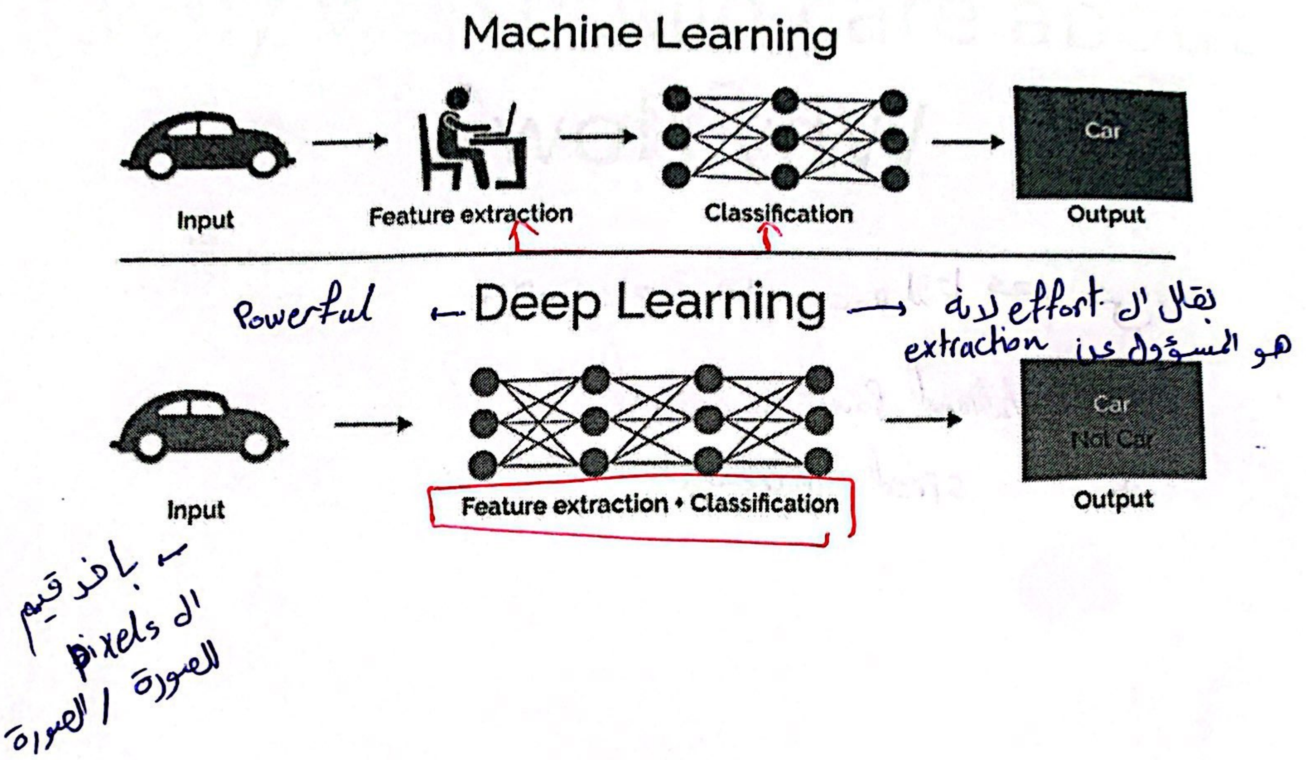
ML Algorithms



لتفهم لحاله حسب قواعد او التشابه بينهم

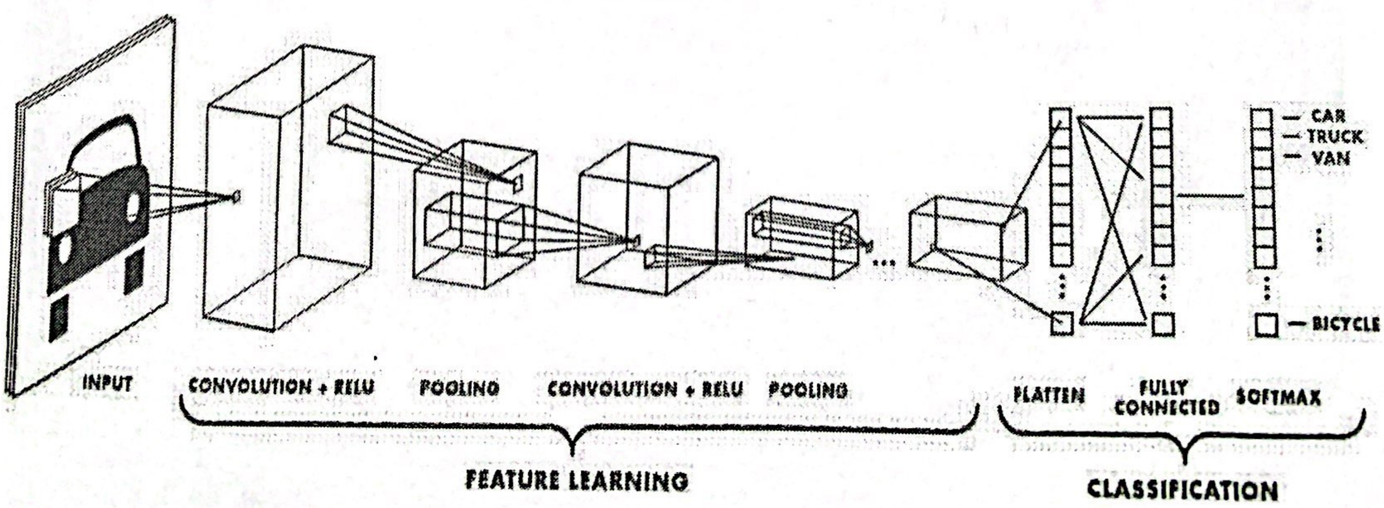
Source: Background Augmentation Generative Adversarial Networks (BAGANs): Effective Data Generation Based on GAN-Augmented 3D Synthesizing

Machine Learning vs Deep Learning



من الأمثلة على
Deep learning

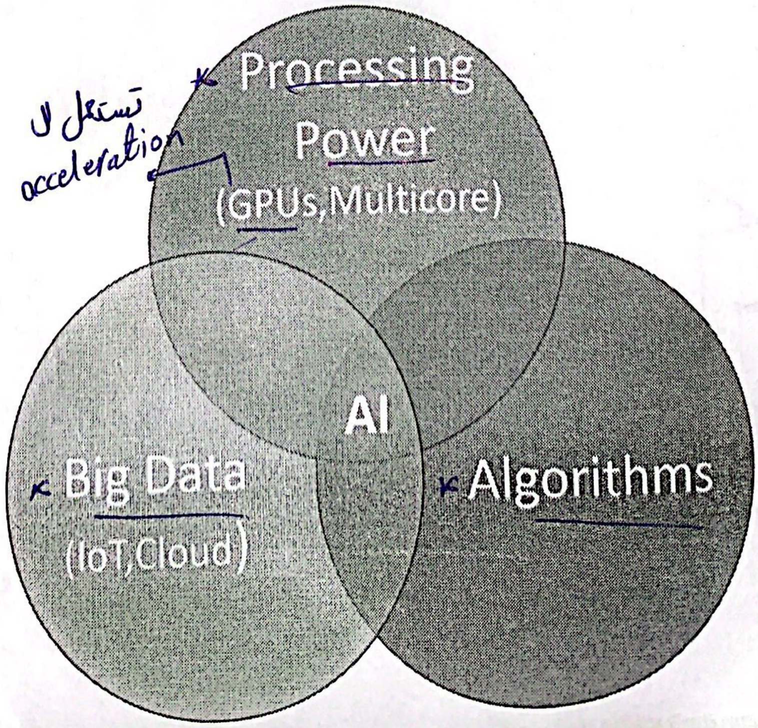
Convolutional Neural Network CNN



Why Now?

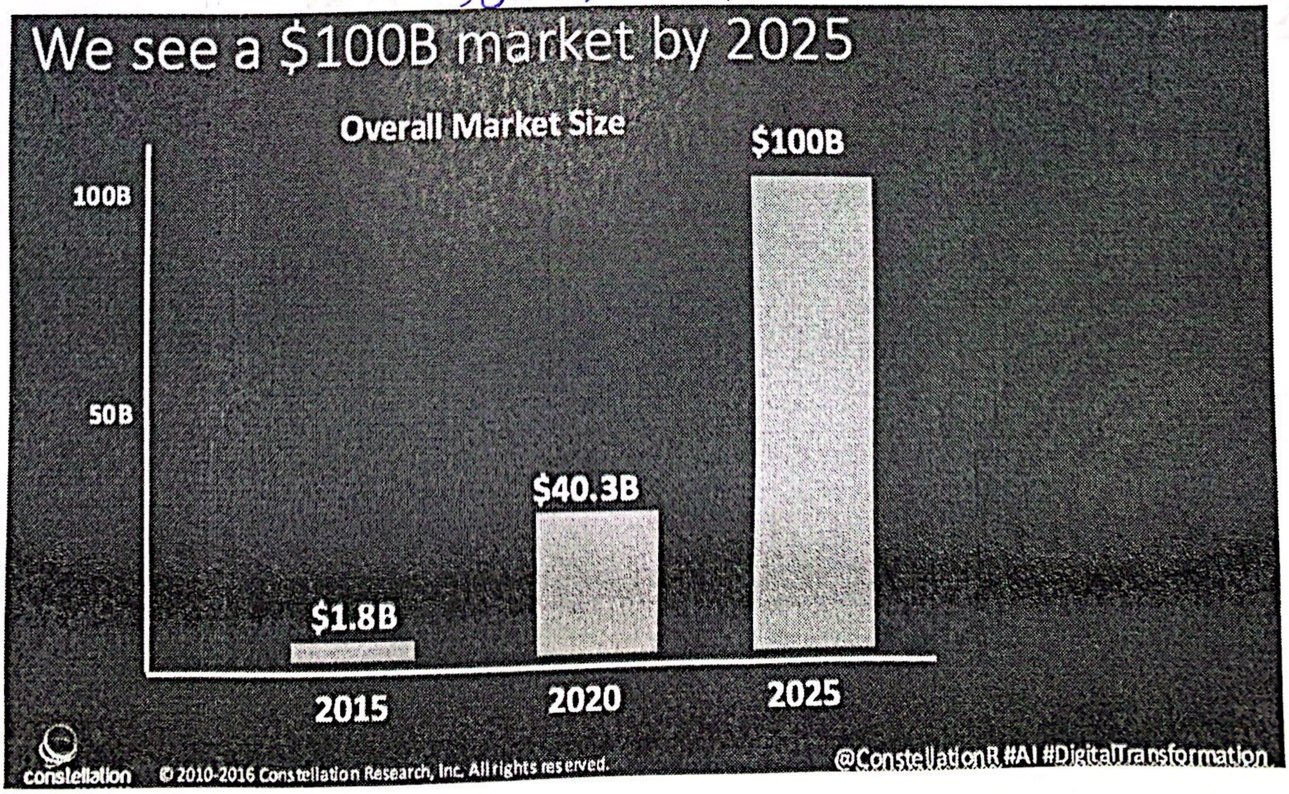
- صرنا بحاجة نعمل processing لداتا بحجم اكبر.
- Computational Power increased
- & // speed increased

AI emergence factors



Why we should care about AI?

• عملية الأستثمار بال AI رابح تزيد والتكلفه



The Jobs Landscape in 2022

WORLD ECONOMIC FORUM

emerging roles, global change by 2022

133 Million

Top 10 Emerging

1. Data Analysts and Scientists
2. AI and Machine Learning Specialists
3. General and Operations Managers
4. Software and Applications Developers and Analysts
5. Sales and Marketing Professionals
6. Big Data Specialists
7. Digital Transformation Specialists
8. New Technology Specialists
9. Organisational Development Specialists
10. Information Technology Services

declining roles, global change by 2022

75 Million

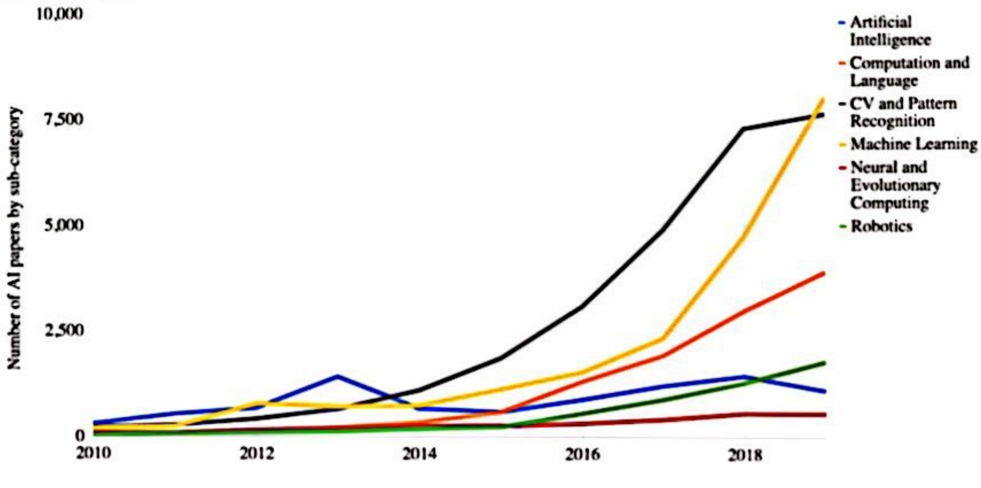
Top 10 Declining

1. Data Entry Clerks
2. Accounting, Bookkeeping and Payroll Clerks
3. Administrative and Executive Secretaries
4. Assembly and Factory Workers
5. Client Information and Customer Service Workers
6. Business Services and Administration Managers
7. Accountants and Auditors
8. Material Recording and Stock Keeping Clerks
9. General and Operations Managers
10. Postal Service Clerks

Source: Future of Jobs Report 2018, World Economic Forum https://hai.stanford.edu/sites/g/files/sbiybj10986/f/ai_index_2019_report.pdf

Number of AI papers on arXiv, 2010-2019

Source: arXiv, 2019.



Deep Learning Papers on arXiv

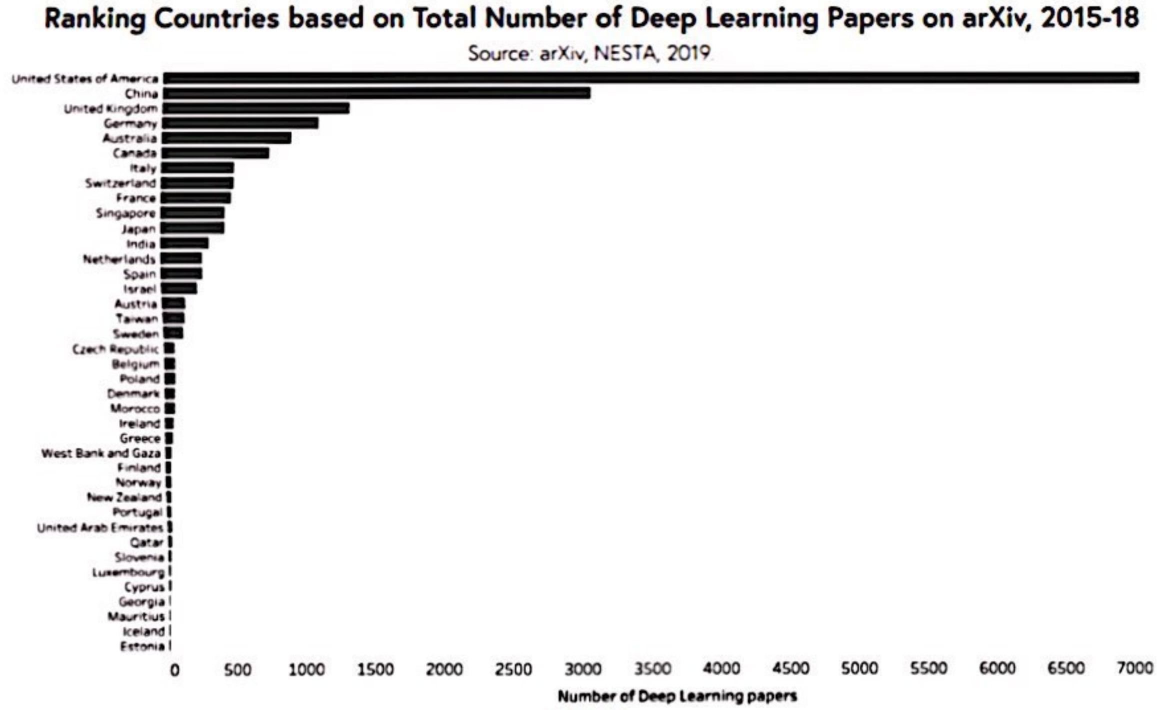
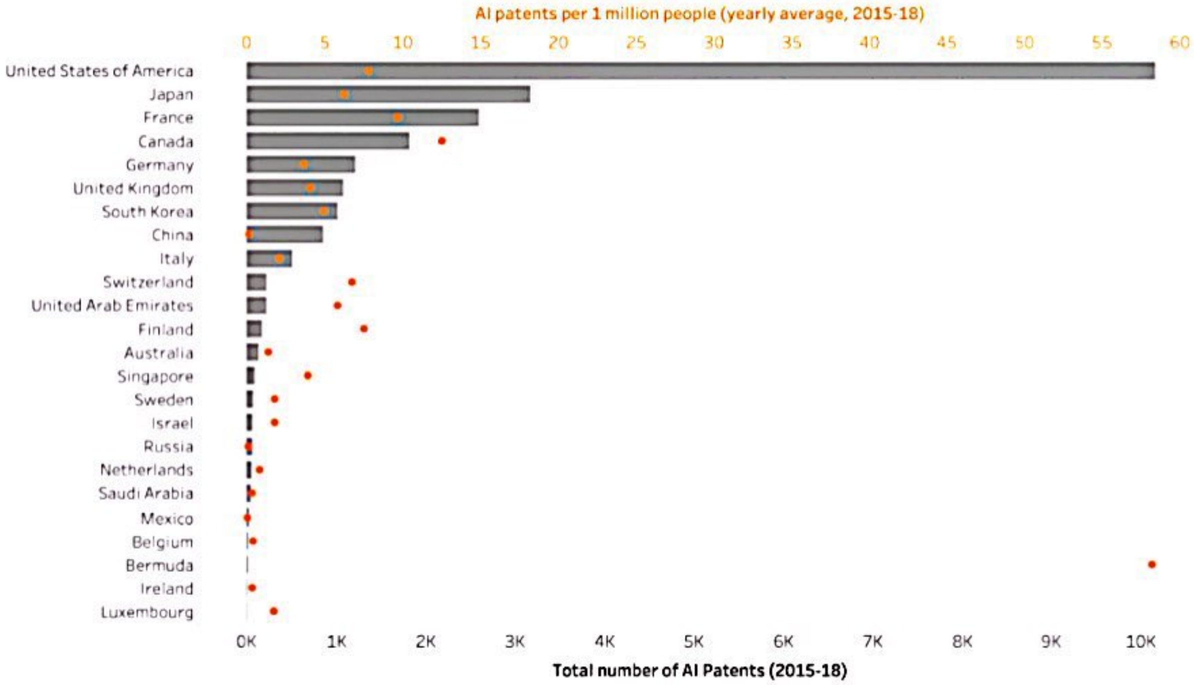


Fig. 1.7b.

Total Volume and average annual per capita AI Published Patents, 2015-2018

Source: MAG, 2019.



Global AI startups that have received funding within the last year (July 2018-July 2019)

Source: CAPIQ, Crunchbase, Quid, 2019.

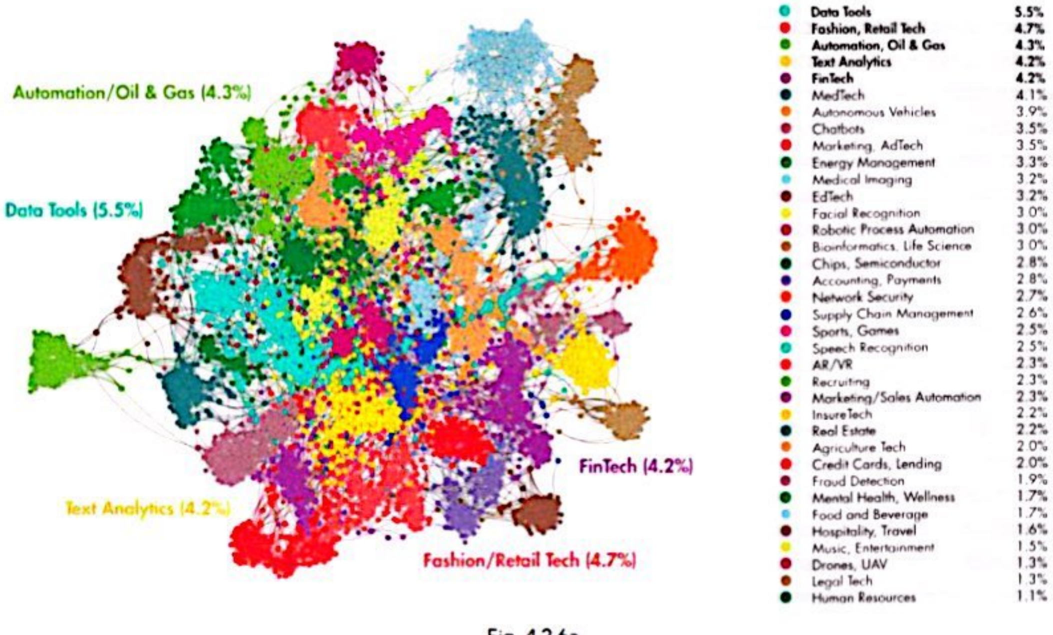


Fig. 4.2.6a.

Network showing 4,403 global AI startups that received investment between July 2018 and July 2019. Colored by sector with top five highlighted.

Appendix: How to Red a Quid Network

1980

Othello

In the 1980s Kai-Fu Lee and Sanjoy Mahajan developed BILL, a Bayesian learning-based system for playing the board game Othello. In 1989, the program won the US national tournament of computer players, and beat the highest ranked US player, Brian Rose, 56—8. In 1997, a program named Logistello won every game in a six game match against the reigning Othello world champion.

1995

Checkers

In 1952, Arthur Samuels built a series of programs that played the game of checkers and improved via self-play. However, it was not until 1995 that a checkers-playing program, Chinook, beat the world champion.

1997

Chess

Some computer scientists in the 1950s predicted that a computer would defeat the human chess champion by 1967, but it was not until 1997 that IBM's DeepBlue system beat chess champion Gary Kasparov. Today, chess programs running on smartphones can play at the grandmaster level.

2011

Jeopardy!

In 2011, the IBM Watson computer system competed on the popular quiz show Jeopardy! against former winners Brad Rutter and Ken Jennings. Watson won the first place prize of $1 million.

2015

Atari Games

In 2015, a team at Google DeepMind used a reinforcement learning system to learn how to play 49 Atari games. The system was able to achieve human-level performance in a majority of the games (e.g., Breakout), though some are still significantly out of reach (e.g., Montezuma's Revenge).

2016

Object Classification in ImageNet

In 2016, the error rate of automatic labeling of ImageNet declined from 28% in 2010 to less than 3%. Human performance is about 5%.

2016

Object Classification in ImageNet

In 2016, the error rate of automatic labeling of ImageNet declined from 28% in 2010 to less than 3%. Human performance is about 5%.

2016

Go

In March of 2016, the AlphaGo system developed by the Google DeepMind team beat Lee Sedol, one of the world's greatest Go players, 4—1. DeepMind then released AlphaGo Master, which defeated the top ranked player, Ke Jie, in March of 2017. In October 2017, a Nature paper detailed yet another new version, AlphaGo Zero, which beat the original AlphaGo system 100—0.

2017

Skin Cancer Classification

In a 2017 Nature article, Esteva et al. describe an AI system trained on a data set of 129,450 clinical images of 2,032 different diseases and compare its diagnostic performance against 21 board-certified dermatologists. They find the AI system capable of classifying skin cancer at a level of competence comparable to the dermatologists.

2017

Speech Recognition on Switchboard

In 2017, Microsoft and IBM both achieved performance within close range of "human-parity" speech recognition in the limited Switchboard domain.

2017

Poker

In January 2017, a program from CMU called Libratus defeated four to human players in a tournament of 120,000 games of two-player, heads up, no-limit Texas Hold'em. In February 2017, a program from the University of Alberta called DeepStack played a group of 11 professional players more than 3,000 games each. DeepStack won enough poker games to prove the statistical significance of its skill over the professionals.

2017

Ms. Pac-Man

Maluuba, a deep learning team acquired by Microsoft, created an AI system that learned how to reach the game's maximum point value of 999,900 on Atari 2600.

2018

Chinese - English Translation

A Microsoft machine translation system achieved human-level quality and accuracy when translating news stories from Chinese to English. The test was performed on newstest2017, a data set commonly used in machine translation competitions.

2018

Capture the Flag

A DeepMind agent reached human-level performance in a modified version of Quake III Arena Capture the Flag (a popular 3D multiplayer first-person video game). The agents showed human-like behaviours such as navigating, following, and defending. The trained agents exceeded the win-rate of strong human players both as teammates and opponents, beating several existing state-of-the art systems.

2018

DOTA 2

OpenAI Five, OpenAI's team of five neural networks, defeats amateur human teams at Dota 2 (with restrictions). OpenAI Five was trained by playing 180 years worth of games against itself every day, learning via self-play. *(OpenAI Five is not yet superhuman, as it failed to beat a professional human team)*

2018

Prostate Cancer Grading

Google developed a deep learning system that can achieve an overall accuracy of 70% when grading prostate cancer in prostatectomy specimens. The average accuracy of achieved by US board-certified general pathologists in study was 61%. Additionally, of 10 high-performing individual general pathologists who graded every sample in the validation set, the deep learning system was more accurate than 8.

2018

Alphafold

DeepMind developed Alphafold that uses vast amount of geometric sequence data to predict the 3D structure of protein at an unparalleled level of accuracy than before.

2019

Alphastar

DeepMind developed Alphastar to beat a top professional player in Starcraft II.

2019

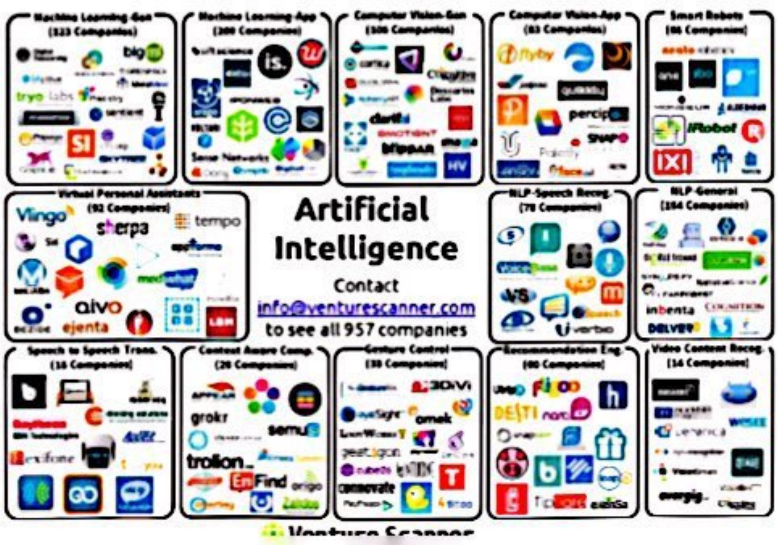
Detect diabetic retinopathy (DR) with specialist-level accuracy

Recent study shows one of the largest clinical validation of a deep learning algorithm with significantly higher accuracy than specialists. The tradeoff for reduced false negative rate is slightly higher false positive rates with the deep learning approach.

AI Applications

AI in Everyday Life

- Email Filters and smart replies in Gmail
- LinkedIn: match candidates
- Pinterest's LENS tool
- Chatbots
- Facebook : Relevant posts
- Product Recommendations
- Banking: Financial Institutions fraud prevention (not Common types of transactions)
- Ride-sharing Apps
- Unlock phone with face ID
- Voice assistants

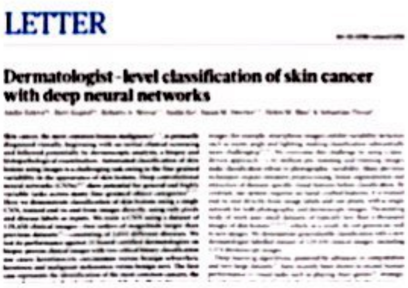


Healthcare

- Robot Assisted Surgery
- Administration and Workflow
- Cybersecurity
- Automated Image Diagnosis
- Fraud Detection
- Treatment Design
- Health Monitoring
- Drug Creation

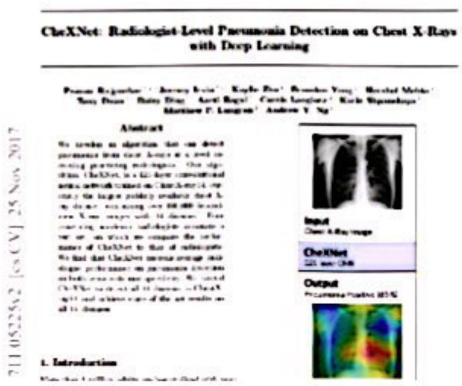
Source Harvard Business Review

Medical



LETTER

Dermatologist-level classification of skin cancer with deep neural networks

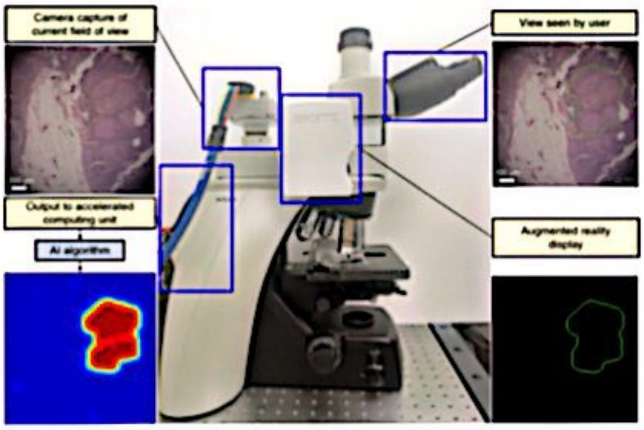


CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning

Abstract

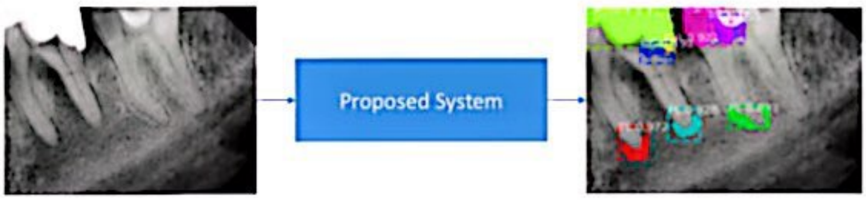
1. Introduction

Augmented Reality Microscope



Source: Augmented Reality Microscope for Real-time Automated Detection of Cancer

Dental Pathologies



Finance

- Portfolio Management:
  - Algorithms built to calibrate a financial portfolio to the goals and risk tolerance of the user (Betterment).
- Algorithmic Trading
  - Fast Trading Decisions
- Loan Insurance underwriting
  - Trained on millions of consumers examples
- Fraud Detection
  - Detect anomalies and flag then to the security team

Books on AI & Finance

Agriculture

- Monitoring
  - Agricultural crop conditions
  - Weather and climate
  - Ecosystems
- Planning and policy-making
- Intelligent environment control for plant production systems
- Intelligent robots in agriculture
- An expert geographical information system for land evaluation
- Artificial neural network for plant classification using image processing.
- Control of green house.

Crop and Soil Health Monitoring

- PEAT: agriculture tech startup
  - Plantix Mobile App
    - Identifies possible defects through images captured by the user's smartphone camera.
    - Users are then provided with soil restoration techniques, tips and other possible solutions as explained in the short video below:

Crop and Soil Health Monitoring

- **Trace Genomics: ML for diagnosing Soil Defects**

Monitoring Crop Health and Sustainability

- **FarmShots: high-resolution satellite imagery that detects plant health by analyzing absorbed light from field images**

Drones and Computer Vision for Crop Analysis

- **SkySquirrel Technologies: Data Analytics for drone-based imaging in agriculture**

- **aWhere: Deliver the most complete agricultural information and insight for real-time agriculture decisions, every day, globall**

Harvesting

Strawberry harvesting robot

Energy Industry

Source: https://www.next-kraftwerke.com/knowledge/artificial-intelligence

Sports

Source: https://www.thomasnet.com/insights/nfl-incorporates-machine-learning-ai-technology-to-prevent-player-injuries/

Sports

SportsPro

Liverpool partner with SkillCorner for AI-powered analysis

Machine learning platform to measure player performance.

Companies and AI

Field	Organization	Applications
Energy	Arco and Tenneco Oil Company	Neural networks used to help pinpoint oil and gas deposits
Government	Internal Revenue Service	Software used to read tax returns and spot fraud
Human services	Merced County, California	Expert systems used to decide if applicants should receive welfare benefits
Marketing	Spiegel	Neural networks used to determine most likely buyers from a long list
Telecommunications	BT Group	Heuristic search used for a scheduling application that provides work schedules for more than 20,000 engineers
Transportation	American Airlines	Expert systems used to schedule the routine maintenance of airplanes
Inventory/forecasting	Hyundai Motors	Neural networks and expert systems used to reduce delivery time by 20 percent and increase inventory turnover from 3 to 3.4
Inventory/forecasting	SCI Systems	Neural networks and expert systems used to reduce on-hand inventory by 15 percent, resulting in $180 million in annual savings
Inventory/forecasting	Reynolds Aluminum	Neural networks and expert systems used to reduce forecasting errors by 2 percent, resulting in an inventory reduction of 1 million pounds
Inventory/forecasting	Unilever	Neural networks and expert systems used to reduce forecasting errors from 40 percent to 25 percent, resulting in a multimillion-dollar savings

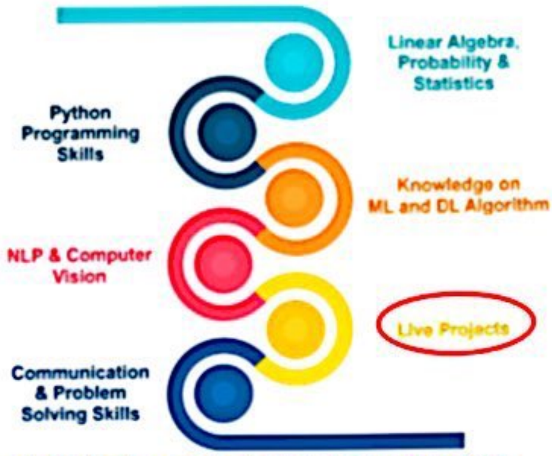
Required Skills

Source: https://dazeinfo.com/2018/09/07/top-en[illegible]/

Online Resources

edX
coursera
UDACITY
Udemy
FutureLearn
KHANACADEMY

Skills



Source:https://www.sonayukti.com/trainings/artificial-intelligence-and-machine-learning-training-program.php

Projects and Datasets

- Kaggle
  - www.kaggle.com
- Github

kaggle

What Is Machine Learning?

- YouTube Video: **What is Machine Learning?** from Google Cloud Platform

https://youtu.be/HcqpanDadyQ

What Is Machine Learning?

- The science (and art) of programming computers so they can **learn from data.**
- The field of study that gives computers the ability to **learn without being explicitly programmed.** Arthur Samuel, *1959*
- A computer program is said to learn from **experience E** with respect to some **task T** and some **performance** measure **P**, if its performance on T, as measured by P, **improves with experience E.** Tom Mitchell, *1997*

→ Data for training only to improve experience
  - **E: Training set** made of **training instances (samples)**
  - **T: Test set**
  - **P:** Such as **accuracy**

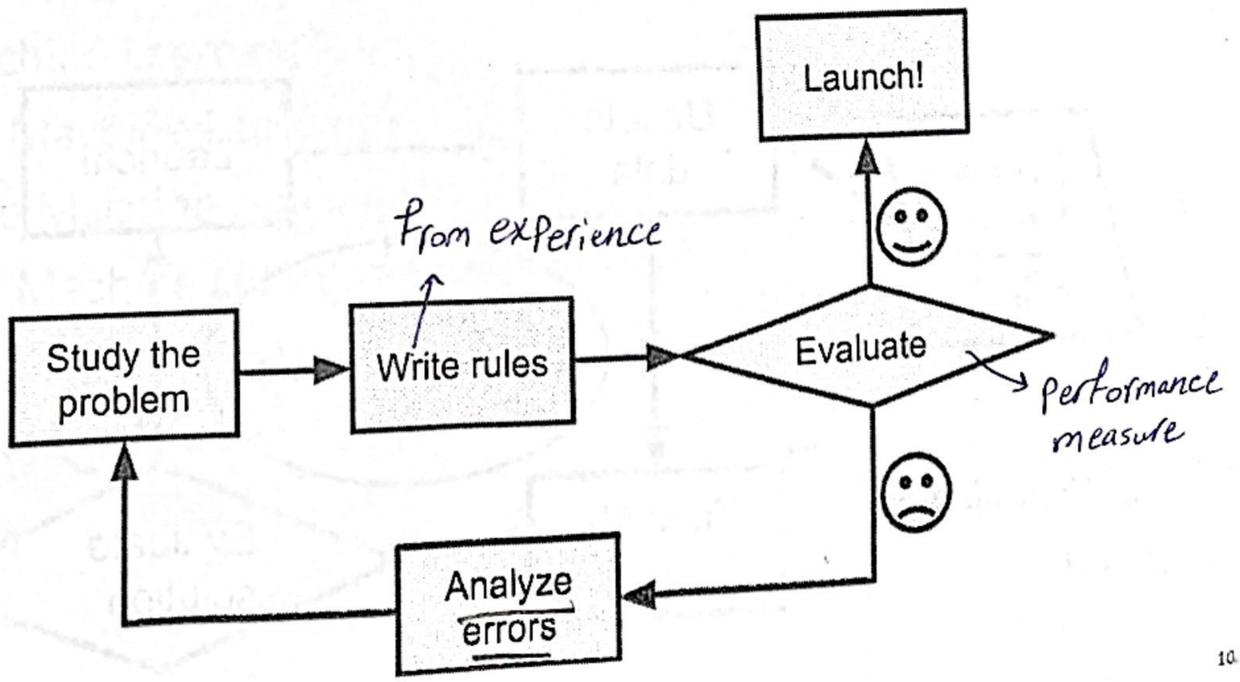
- Task → what to do
- Experience → ability to do it
- Performance measure → ex:- accuracy.

Outline

✓The Machine Learning Tsunami
✓What Is Machine Learning?
- Why Use Machine Learning?
- Types of Machine Learning Systems
- Main Challenges of Machine Learning
- Testing and Validating
- Summary
- Exercises

Why Use Machine Learning?

Spam filter using traditional programming techniques

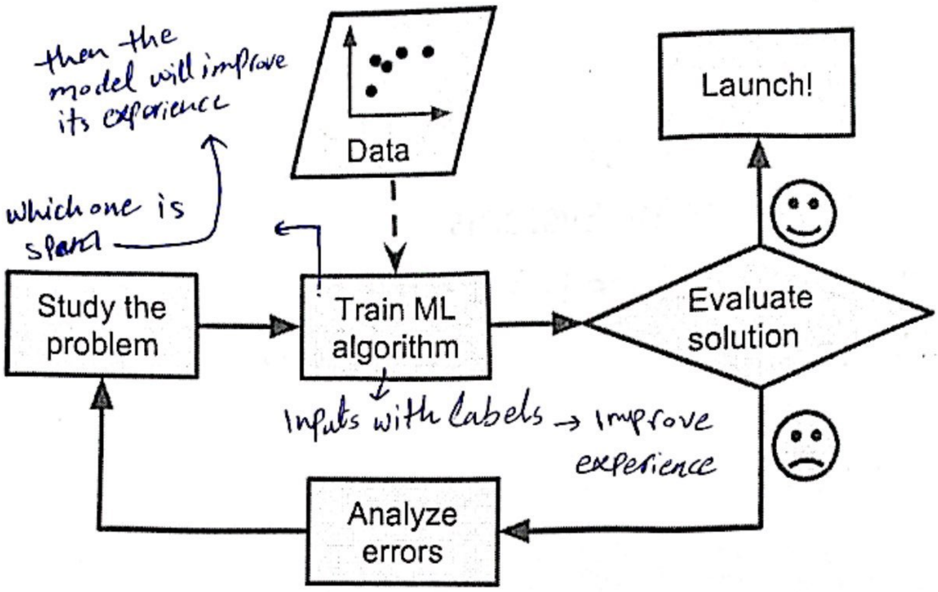


Why Use Machine Learning?

Spam filter using machine learning techniques 1/2

then the model will improve its experience

which one is spam

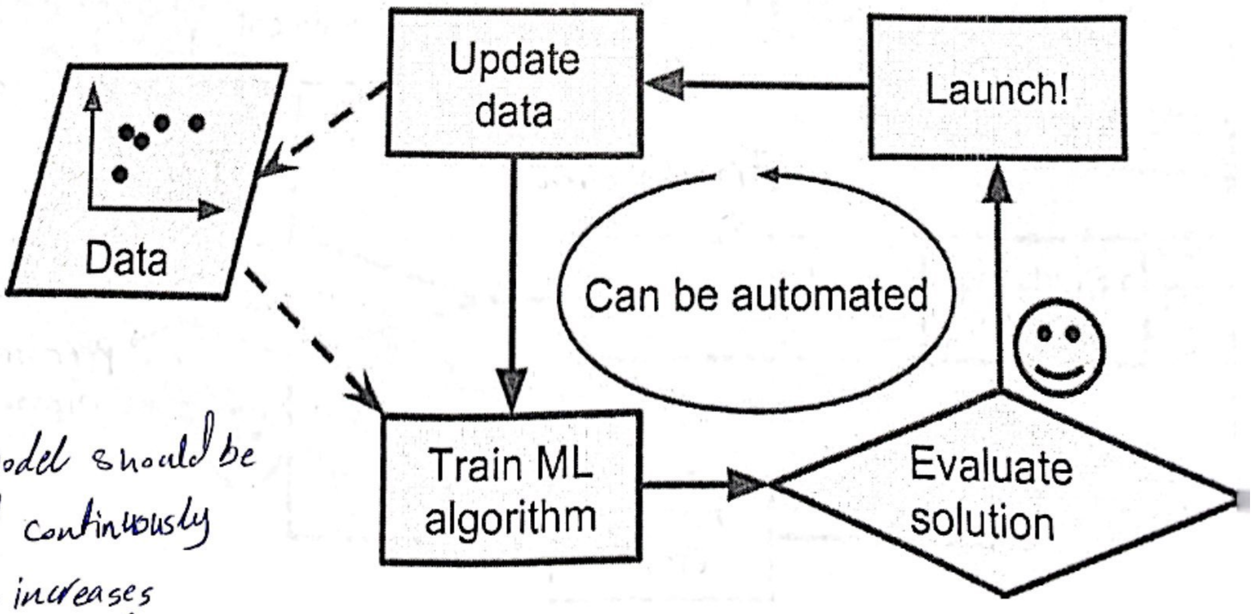


Inputs with labels → improve experience

• مثلا بعطي ال model مجموعة من الايميلات = بحكيله هدول spam او مش spam.

Why Use Machine Learning?

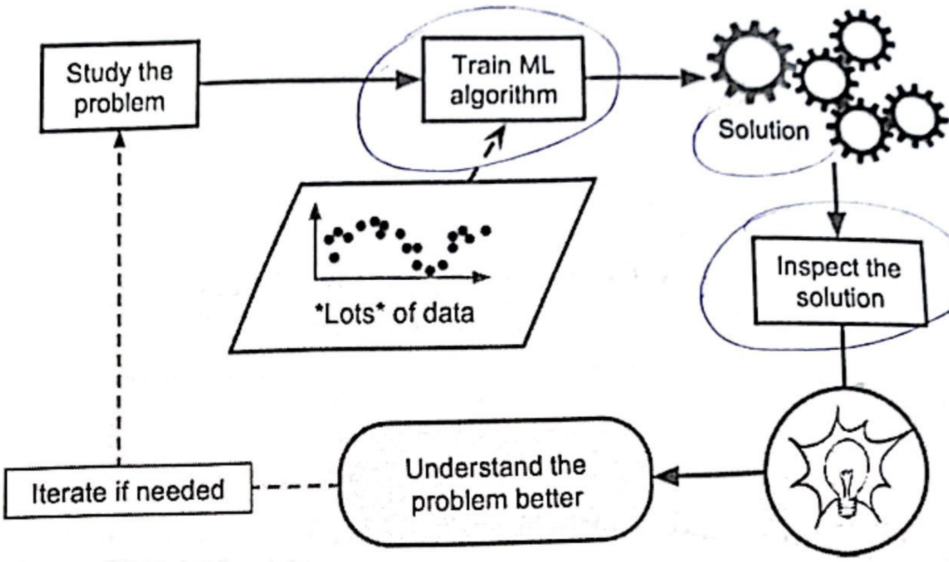
Automatically adapting to change 2/2



• The model should be updated continuously as data increases and ~~old~~ updated.

Why Use Machine Learning?

ML can help humans learn (Data mining)



Outline

✓The Machine Learning Tsunami

✓What Is Machine Learning?

✓Why Use Machine Learning?

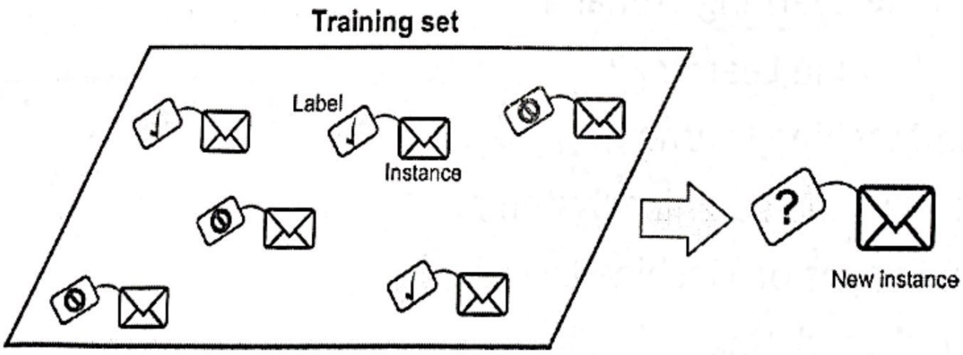
- Types of Machine Learning Systems
- Main Challenges of Machine Learning
- Testing and Validating
- Summary
- Exercises

Types of Machine Learning Systems

- **Involves human supervision?**
  1. Supervised learning → label training data
  2. Unsupervised learning
  3. Semi-supervised learning
  4. Reinforcement learning → 5. Self-supervised learning
- **Learns incrementally?**
  1. Batch learning
  2. Online learning
- **Generalization approach**
  1. Instance-based learning
  2. Model-based learning

• حسب المشكلة وكيفية تدريب ال model

1. Supervised Learning



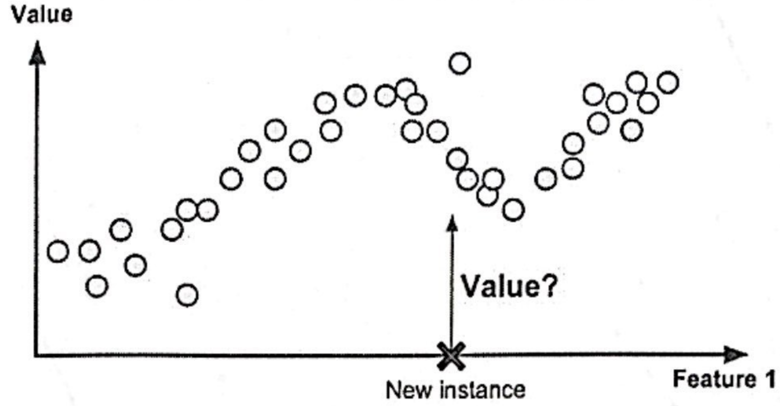
The training data you feed to the algorithm includes the desired solutions, called **labels**

Classification: finds the class, e.g., email type (spam or ham)

مثال علامات الطالب (A, B, C...)

overhead (labeling data)

1. Supervised Learning



Regression finds the value, e.g., car price

Continuous range

مثال :- توقع علامات الطالب (90.3 / 90.5 ...)

1. Supervised learning algorithms

Algorithm	Type
k-Nearest Neighbors	Both
Linear Regression	Regression
Logistic Regression	Classification
Support Vector Machines (SVMs)	Both
Decision Trees	Both
Random Forests	Both
Neural Networks	Both

find linear relationship between inputs and outputs

بشتغل classifier او regression

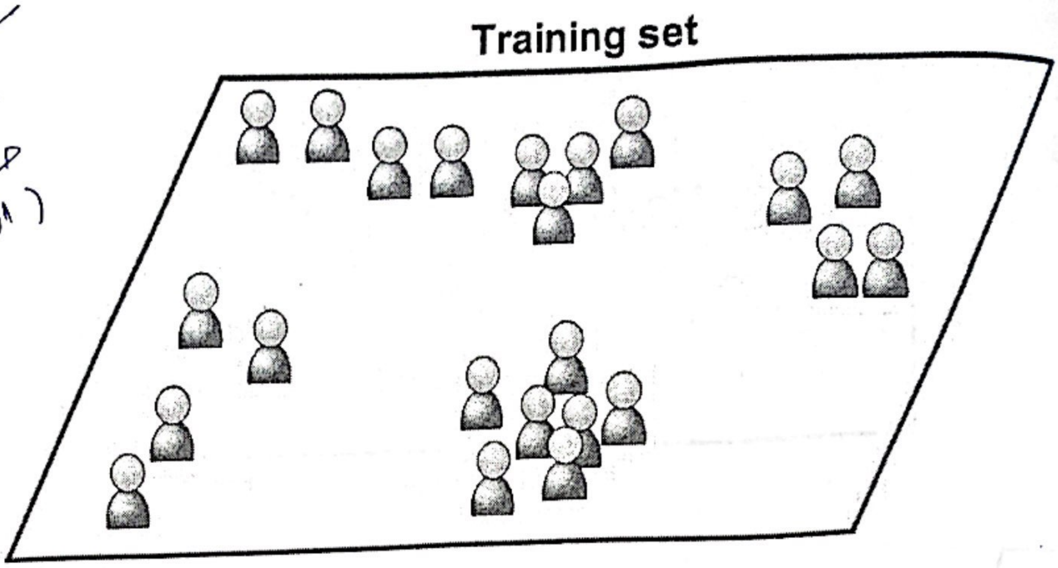
Scanned with CamScanner

Data without labels

بنحدد له عدد ال clusters

2. Unsupervised Learning

بتم التصنيف حسب خصائص معينة مشتركة (التشابه)



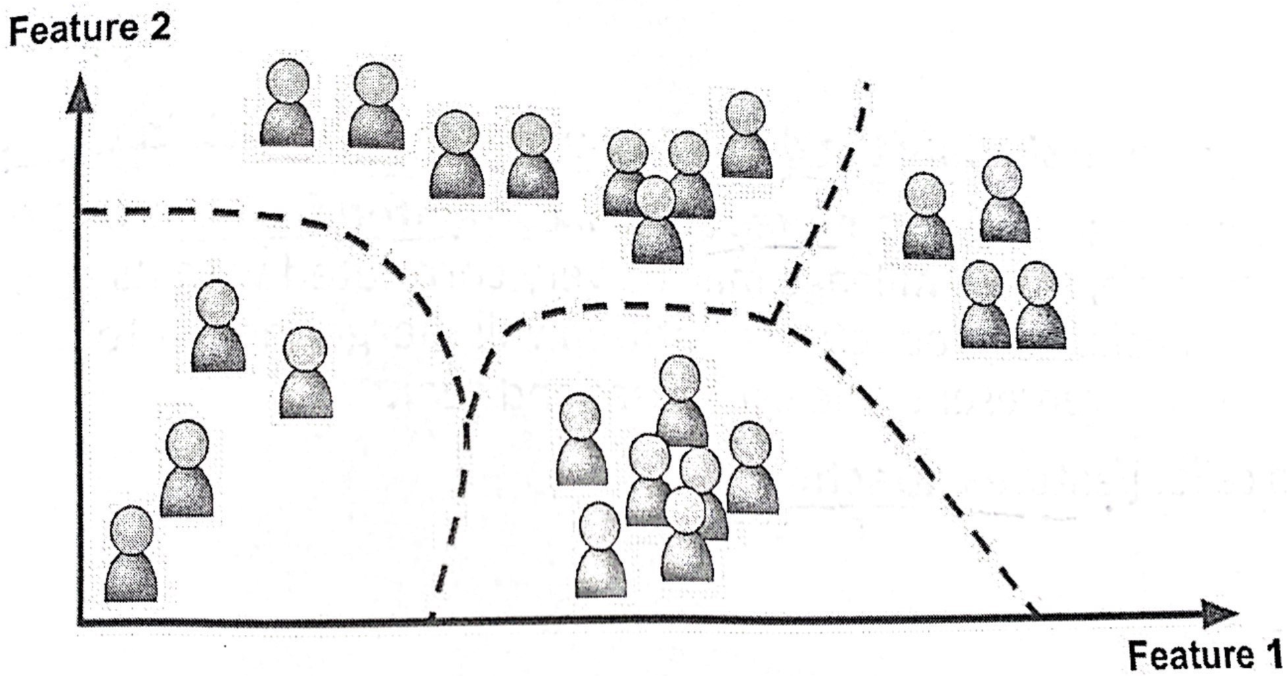
The training data is **unlabeled.**

2. Unsupervised learning algorithms

needs distance measures → عشان نرد الصورة اقرب لاي مجموعة

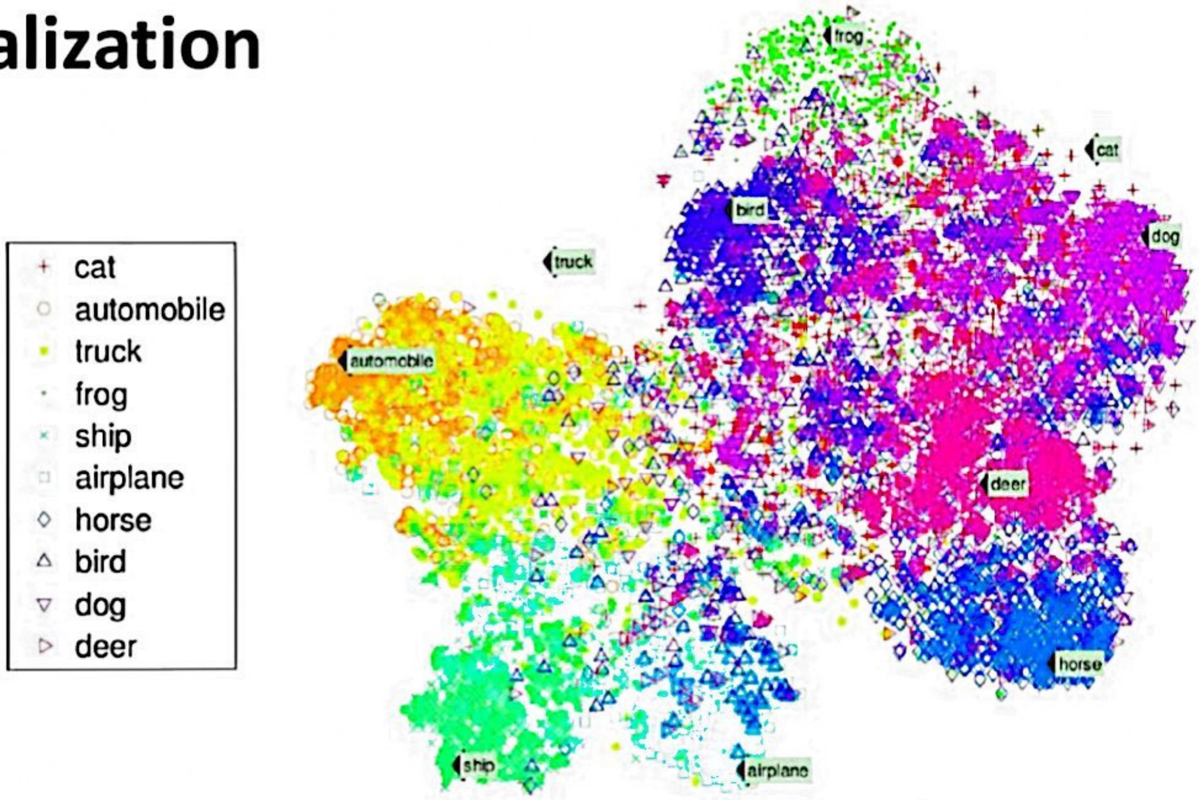
- Clustering
  - k-Means
  - Hierarchical Cluster Analysis (HCA)
  - Expectation Maximization
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori
  - Eclat

2.a Clustering



ex:- Pixels

2.b Visualization



نقلل ال features (نلغي منهم او ندمجهم)

2.c Dimensionality Reduction

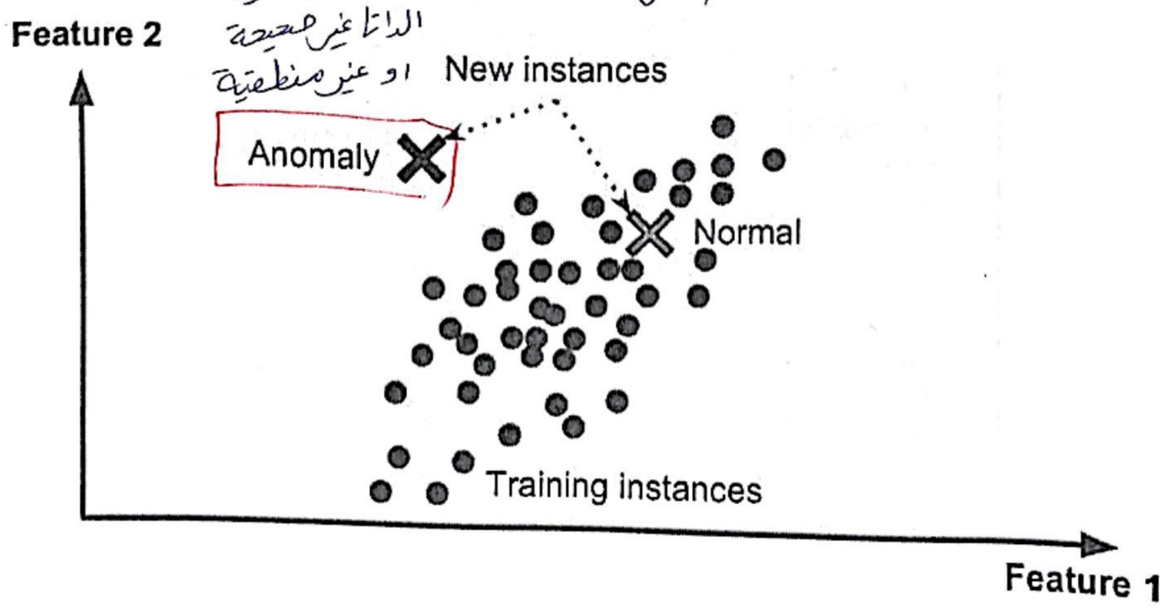
- The goal is to **simplify the data** without losing too much information.
- One way to do this is to **merge** several **correlated features** into one. For example, a car's mileage may be very correlated with its age, so the dimensionality reduction algorithm will merge them into one feature that represents the car's wear and tear.
- Also called **feature extraction.**

بعض ال samples ما بتبعوا cluster معين

يتم تعريفهم واستثنائها اذا كان عددهم قليل

2.d Anomaly Detection

يا ممكن تكون الداتا غير صحيحة او غير منطقية



خاصية
لا يكون حجم
الداتا كبير

2.e Association Rule Learning

ترتيب الداتا.

- The **goal is to dig into large amounts** of data and **discover interesting relations** between attributes.
- For example, suppose you own a supermarket. Running an association rule on your sales logs may reveal that people who purchase **barbecue sauce** and **potato chips** also tend to buy steak. Thus, you may want to place these items close to each other.

* Medical diagnosis : using the multi-relational association rule, we can determine the probability of disease occurance associated with various factors + symptoms in the data from past cases

* Entertainment : Netflix & spotify use association rules to fuel their content recommendation engines by analyzing user past behaviour

25

يعني مثلا الاشخاص حضروا فيلم معين وتبعاً اله حضروا اشي ثاني

او اشخاص اشتروا مثلا من امازون قطعة وتبعاً الها اشتروا اشي ثاني وهكذا

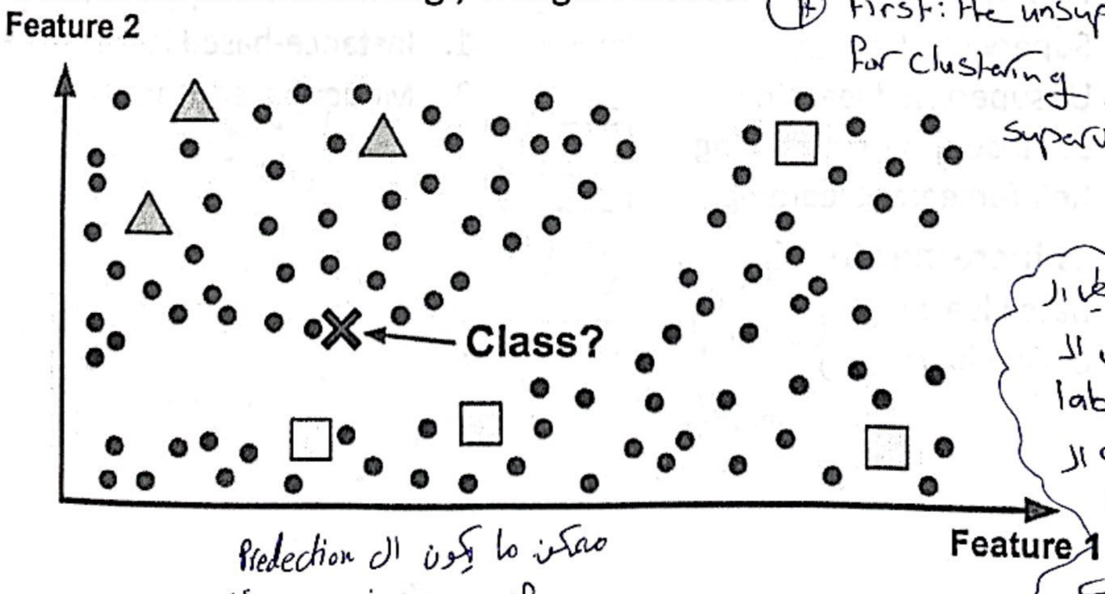
Got f hod

3. Semi-supervised Learning

لانه عملية ال labeling مكلفة

Partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data. E.g., Google Photos.

(1) first: the unsupervised for clustering, then supervised



أو

(2) اول شي بعطي ال model كل ال labeled data بعدين بعطيه ال unlabeled وبخليه يعطيني predection

26

ممكن ما يكون ال Predection صحيح ، لازم يتم التأكد منه وتعديله

4. Self-supervised Learning

- Generating a fully labeled dataset from a fully unlabeled one

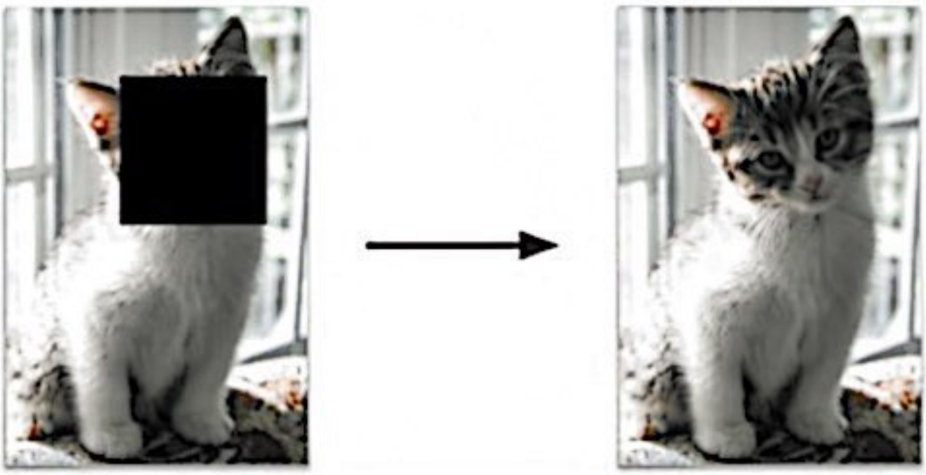


Figure 1-12. Self-supervised learning example: input (left) and target (right)

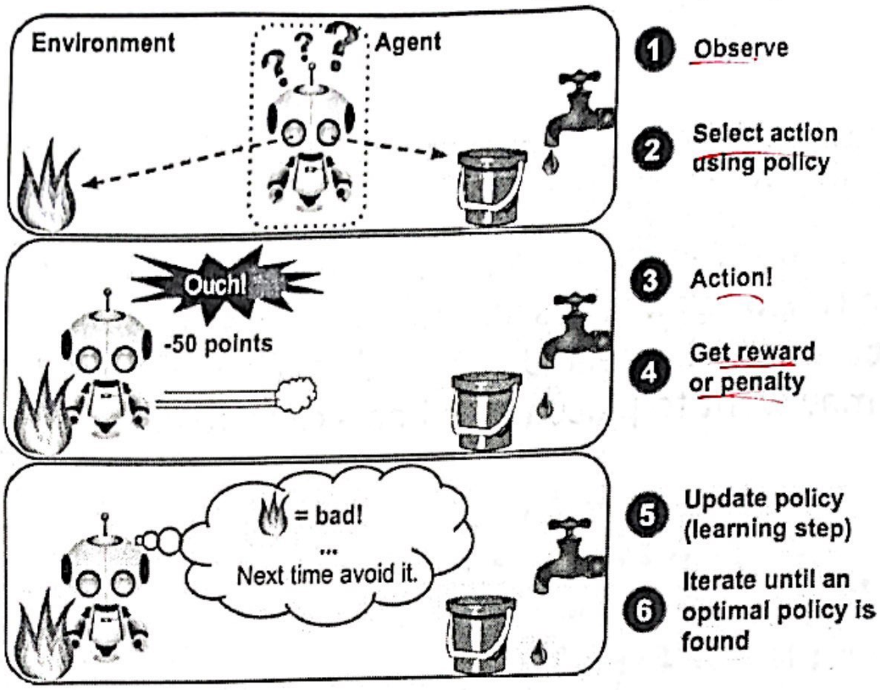
* will not be covered

learning by experience

شائع بال robotics / الألعاب وال self driving cars

4. Reinforcement Learning

بيكون تجربة الخبرة من كل محاولة او Task



Types of Machine Learning Systems

7/3

✓ **Involves human supervision?**

1. Supervised learning
2. Unsupervised learning
3. Semi-supervised learning
4. Reinforcement learning

- **Learns incrementally?**

1. Batch learning ← تدريب ال model على كل الداتا
2. Online learning → لتطوير ال model واعادة تدريبه لأخذ التعديلات بعين الاعتبار

- **Generalization approach**

1. Instance-based learning
2. Model-based learning

1. Batch (offline) Learning

- Must be **trained** using **all the available data**.
- This will generally take a **lot of time** and computing resources, so it is typically done **offline**.
- First the system is **trained**, and **then** it is **launched** into production and runs without learning anymore; it just applies what it has learned.

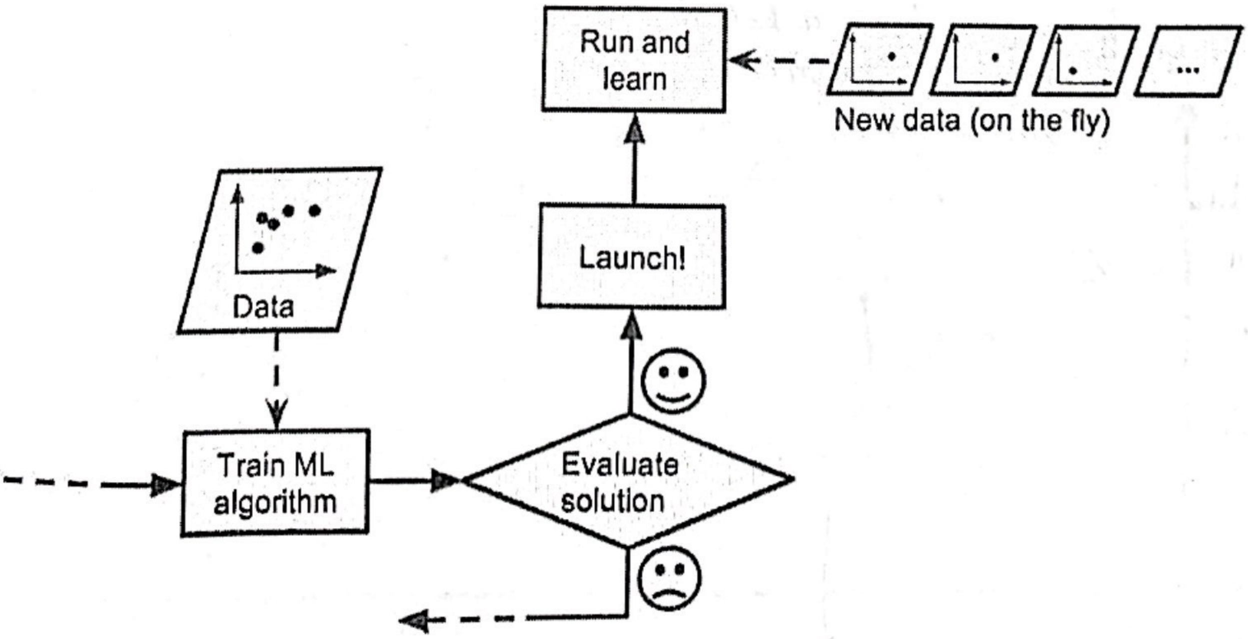
اي تعديل على الداتا بتطلب اعادة التدريب والـ Test
على الداتا القديمة والجديدة

باخذ الداتا
incrementally

تدريب
على الداتا الجديدة

2. Online Learning → faster

Examples: Stock prices, huge data



Types of Machine Learning Systems

✓**Involves human supervision?**

1. Supervised learning
2. Unsupervised learning
3. Semi-supervised learning
4. Reinforcement learning

✓**Learns incrementally?**

1. Batch learning
2. Online learning

• **Generalization approach**

1. Instance-based learning ← مافي general rule
2. Model-based learning

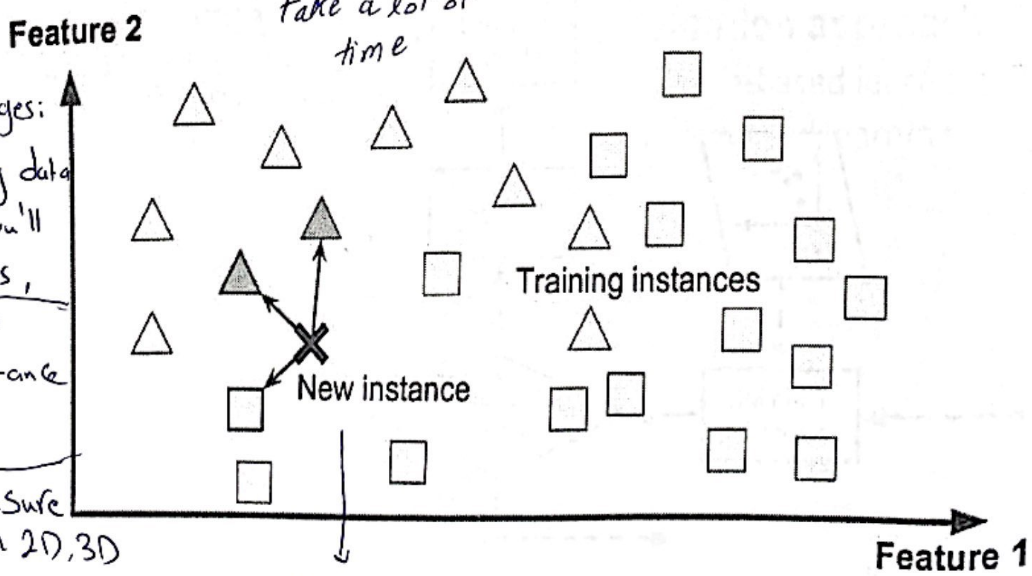
مثلاً بتقارن مواصفات صورة مع باقي الصور و بيطلع Prediction

1. Instance-based Learning

→ بتقارن الداتا الجديدة مع كل الداتا القديمة

take a lot of time

* disadvantages:
If the training data set is big you'll need resources,
What way to measure distance around the instance
* hard to measure distances in 2D, 3D



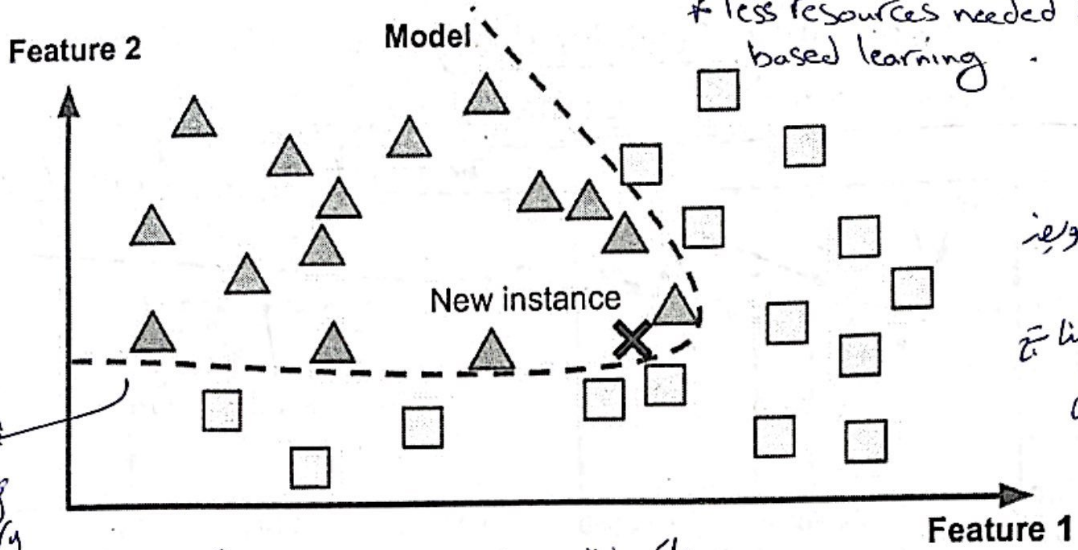
اعلى ٣ هيطلع ٢ مثلث و ١ مربع فغالباً هي مثلث

* use the entire dataset as the model like (K-Nearest Neighbors).

use training data to create a model that has parameters ~~learned~~ learned from the training data

2. Model-based Learning

* faster than instance based
* doesn't always need to bring out the training data while prediction
* less resources needed than instance based learning.



decision ~~boundary~~ boundary

• مثل معادلة بتم تعويض ال features فيها و بكون الناتج رقم مربوط بال decision boundary

• اكبر من ال decision boundary بكون مثلث
• اقل من ال // بكون مربع .

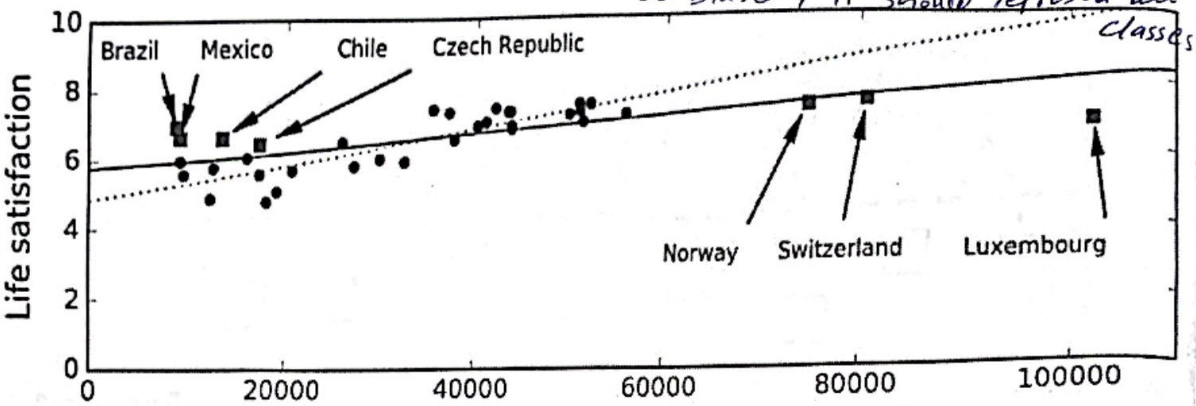
Outline

- ✓The Machine Learning Tsunami
- ✓What Is Machine Learning?
- ✓Why Use Machine Learning?
- ✓Types of Machine Learning Systems
- • Main Challenges of Machine Learning
- • Testing and Validating
- • Summary
- • Exercises

Main Challenges of Machine Learning (due to bad data)

حجم الداتا

1. **Insufficient** quantity of training data
2. **Non-representative** training data → data shouldn't be biased, it should represent all classes



$\boxed{\uparrow \text{data} \quad \uparrow \text{accuracy}}$

مثلاً في بلدان الدخل عندهم عالي وال life satisfaction عندهم = 7
و في بلدان الدخل عندهم قليل وال life satisfaction عندهم = 7

→ الداتا ما غطت ال classes وغير متوازنة.

Main Challenges of Machine Learning (due to bad data)

3. **Poor-quality** data that contains:
   - Errors
   - Outliers
   - Noise

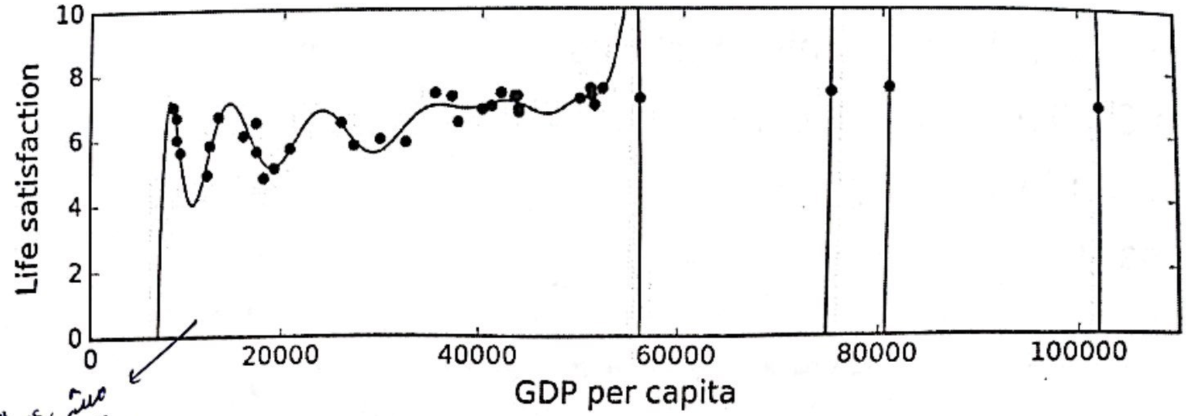
ما الها علاقة ال features بالنتيجة.

4. **Irrelevant features**: Need **feature engineering**:
   - **Feature selection**: selecting the most useful features.
   - **Feature extraction**: combining existing features to produce a more useful one.
   - **Creating new features** by gathering new data.

Main Challenges of Machine Learning (due to bad algorithm)

ماستوعب اي تغيير في الداتا

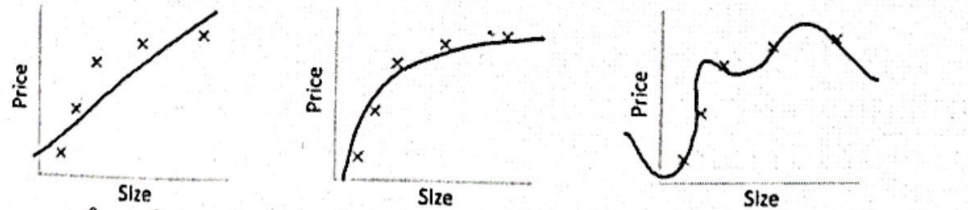
1. **Overfitting** the training data (not general model) مابقدر يتعامل مع داتا جديدة
   - **Regularization** constrains the model's **hyperparameters** to make it simpler and reduce the risk of overfitting.



مش دي النقاط بشكل حرفي، لو اعطيناه اي نقطة جديدة مارح يتعامل معها بشكل صحيح

Main Challenges of Machine Learning (due to bad algorithm)

2. Under-fitting the training data



$\theta_0 + \theta_1 x$ | $\theta_0 + \theta_1 x + \theta_2 x^2$ | $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

تحيز

High bias (underfit) → Poor accuracy in test data and training data

"Just right" → best at ~~train~~ testing data, good at training

High variance (overfit) → best at training data - bad at test.

Scanned with CamScanner

Outline

- ✓The Machine Learning Tsunami
- ✓What Is Machine Learning?
- ✓Why Use Machine Learning?
- ✓Types of Machine Learning Systems
- ✓Main Challenges of Machine Learning
- Testing and Validating
- Summary
- Exercises

Testing and Validating

- **Split** your data into two sets (**cross validation**):
  - The **training set** (80%)
  - The **test set** (20%)
- **Evaluate:**
  - The training error
  - The generalization error
- If the training error is low but the generalization error is high, it means that your model is overfitting the training data.
- When the ML algorithm is iterative, often we use a third set: **validation set.**

test during the training

i.e., is hint

third set

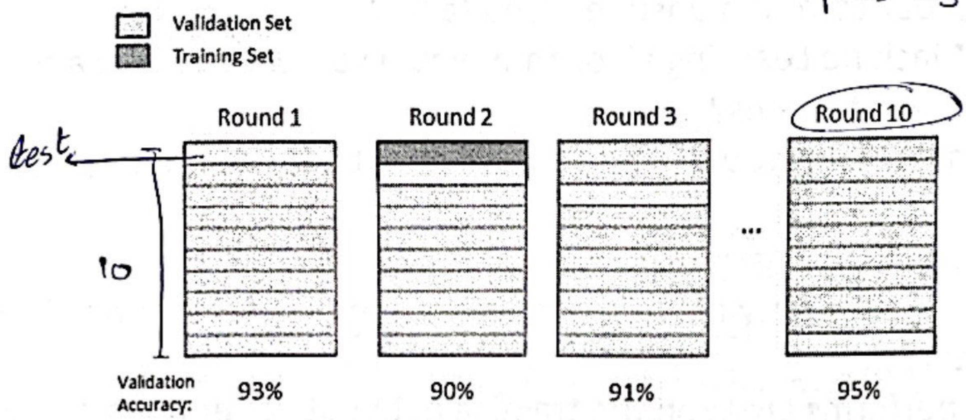
• classes should be well represented in the training data and test data.

Cross Validation

تقسيم الداتا لاجزاء واعمل على اجزاء train , test

• In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples.

* Cover all samples as Validation Set.



Final Accuracy = Average(Round 1, Round 2, ...)

• بكل مرة بنبدا ال model من الصفر

Summary

- ML is about making machines get better at some task by learning from data, instead of having to explicitly code rules.
- Types of ML systems: supervised or not, batch or online, and instance-based or model-based.
- A model-based algorithm tunes some parameters to fit the model to the training set, and then hopefully it will be able to make good predictions on new cases.
- An instance-based algorithm learns the examples by heart and uses a similarity measure to generalize to new instances.
- The system will not perform well if your training set is too small, not representative, noisy, or polluted with irrelevant features.
- Your model needs to be neither too simple (under-fit) nor too complex (over-fit).

Exercises

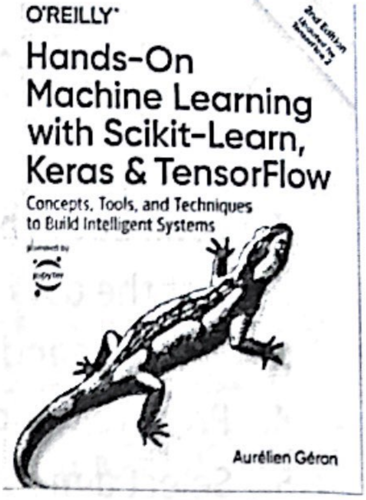
1 • How would you define Machine Learning?
2 • What is a labeled training set?
3 • Can you name four common unsupervised tasks?
4 • What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?
5 • What type of algorithm would you use to segment your customers into multiple groups?
6 • What is an online learning system?
7 • What is the difference between a model parameter and a learning algorithm's hyperparameter?
8 • If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?
9 • What is the purpose of a validation set?
↳ finding and optimizing the best model to solve a given problem.

1 giving computers the ability to learn without explicitly being programmed to.
2 training set with it's solutions.
3 clustering, visualization, dimensionality reduction, Association rule learning.
4 reinforcement learning.
5 supervised "if groups are labeled"
unsupervised "if no labels"
6 online learning system: is a method of machine learning in which data becoms available in a sequential order and it's used to update the best predictor for future data at each step.
7 model parameters are estimated from data automatically, hyperparameters are set manually and used in process to help estimate model parameters.
8 overfitting, solutions: ① cross validation
② Regularization
③ simplify the model.

End-to-End Machine Learning Project

Prof. Gheith Abandah

Reference



- Chapter 2: **End-to-End Machine Learning Project**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, ~~2nd~~ third Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

The 7 Steps of Machine Learning

- YouTube Video: **The 7 Steps of Machine Learning** from Google Cloud Platform

https://youtu.be/nKW8Ndu7Mjw

Caution: *Alcohol is forbidden in the Islamic religion and causes addiction and has negative effects on health.*

12/3

Outline

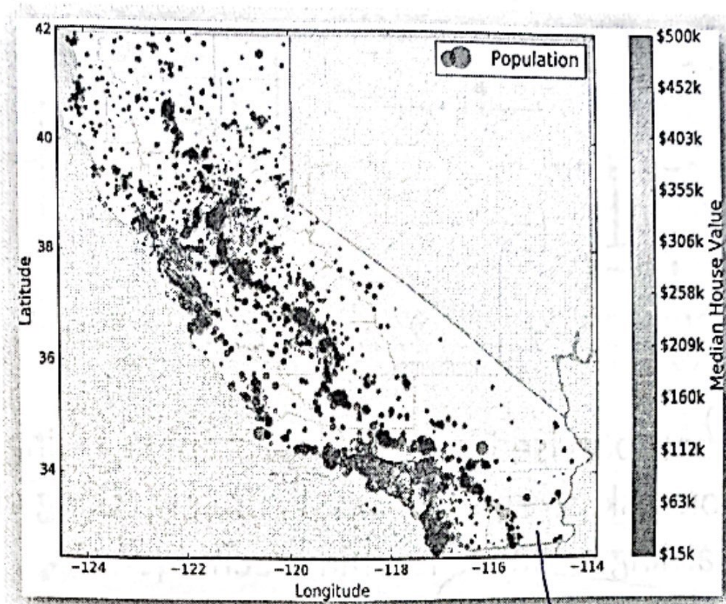
1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

Working with Real Data

- Popular open data repositories:
  - Tensorflow Datasets (GitHub)
  - UC Irvine Machine Learning Repository
  - Kaggle datasets
  - Amazon's AWS datasets
  - IEEE DataPort
- Meta portals (they list open data repositories):
  - Google Dataset Search
  - http://dataportals.org/
  - http://opendatamonitor.eu/
  - http://quandl.com/
- Other pages listing many popular open data repositories:
  - Wikipedia's list of Machine Learning datasets
  - Quora.com question
  - Datasets subreddit

California

1. Look at the Big Picture: CA Housing Data



data visualization

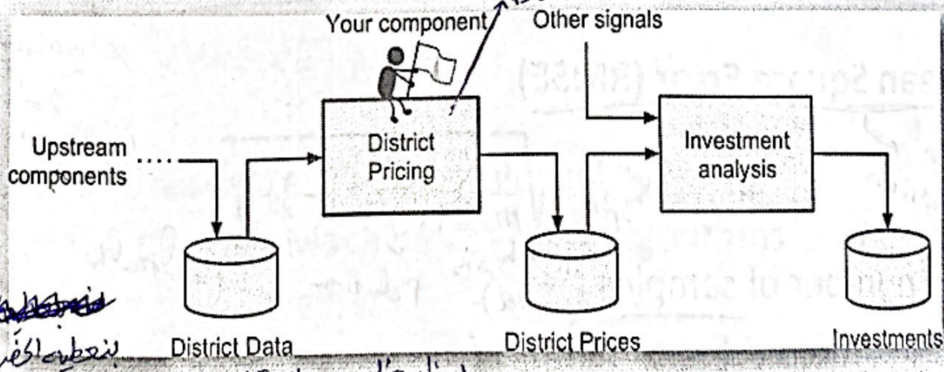
عشان نفهم طبيعة الداتا قبل ما نشتغل عليها.

data visualization

6

* predict median house values in Californian districts, given a number of features from these districts.

1.1. Frame the Problem



بنعطيه الخصائص والاسعار

بدنا سعر البيت بالارقام

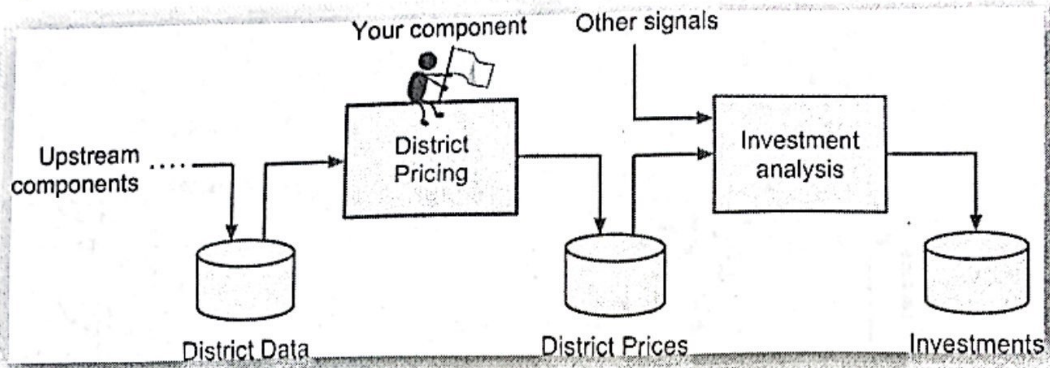
Is it supervised, unsupervised, or Reinforcement Learning?
Is it a classification task, a regression task, or something else? Should you use batch learning or online learning techniques?
Instance-based or Model-based learning?

better

لانه كل الداتا عندي . وبندربه على كل الداتا

7

1.1. Frame the Problem



Is it **supervised,** unsupervised, or Reinforcement Learning?
Is it a classification task, a **regression** task, or something else? Should you use **batch** learning or online learning techniques?
Instance-based or **Model-based** learning?

بالزمن
طريقة
distance measure ل

بنعطيه مواصفات و بعطينا سعر عن طريق مقارنة ال district مع الباقي وبشوف الأقرب

٤٥ +

1.2. Select a Performance Measure

بنحتاجه اكثر لما يكون regression

- **Root Mean Square Error (RMSE)**

حالة ال regression

$$\mathrm{RMSE}(\mathbf{X}, \underset{\text{hypothesis (model)}}{h}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\underset{\text{Prediction}}{h\left(\mathbf{x}^{(i)}\right)} - \underset{\text{groundtruth}}{y^{(i)}}\right)^2}$$

- m is the number of samples
- $\mathbf{x}^{(i)}$ is the feature vector of Sample i
- $y^{(i)}$ is the label or desired output
- $\mathbf{X}$ is a matrix containing all the feature values

$$\mathbf{X} = \begin{pmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \left(\mathbf{x}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(1999)}\right)^T \\ \left(\mathbf{x}^{(2000)}\right)^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,41 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Scanned with CamScanner

1.2. Select a Performance Measure

- Mean Absolute Error

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

- MAE is better than RMSE when there are outlier samples.

Mean absolute error is better than root mean square error because if there is an outlayer it'll have less impact but if you don't have oulayers use root mean square error.

Outline

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

~~*~~ 2. Get the Data 14/3

- If you didn't do it before, it is time now to **download** the **Jupyter notebooks** of the textbook from

  https://github.com/ageron/handson-ml2

- Start Jupyter notebook and open Chapter 2 notebook.
- Hint: If you get kernel connection problem, try

  ```
  C:\>jupyter notebook -port 8889
  ```
- The following slides summarize the code used in this notebook.

2. Get the Data

1. Download the `housing.tgz` file from **Github** using `urllib.request.urlretrieve()` from the `urllib` package
2. Extract the data from this compressed tar file using `tarfile.open` and `extractall()`. The data will be in the CSV file `housing.csv`
3. Read the CSV file into a Pandas DataFrame called `housing` using `pandas.read_csv()`

2.1. Take a Quick Look at the Data Structure

- Display the top five rows using the DataFrame's `head()` method
- The `info()` method is useful to get a quick description of the data
- To find categories and repetitions of some column use `housing.['key'].value_counts()`
- The `describe()` method shows a summary of the numerical attributes. ↳ Statistic
- Show histogram using the `hist()` method and `matplotlib.pyplot.show()`


```
using.info()
lass 'pandas.core.frame.DataFrame'>
ngeIndex: 20640 entries, 0 to 20639
ta columns (total 10 columns):
ngitude                20640 non-null float64
titude                 20640 non-null float64
using_median_age       20640 non-null float64
tal_rooms              20640 non-null float64
tal_bedrooms           20433 non-null float64
pulation               20640 non-null float64
useholds               20640 non-null float64
dian_income            20640 non-null float64
dian_house_value       20640 non-null float64
ean_proximity          20640 non-null object
ypes: float64(9), object(1)
mory usage: 1.6+ MB
```

207 missing features

```
>>> housing["ocean_proximity"].value_counts()
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

→ districts

manually
or by using
scikit learning

2.2. Create a Test Set

- **Split** the available data randomly to:
  - Training set (80%)
  - Test set (20%)

> must be representitive

- The example defines a function called `split_train_test()` for illustration.
- Scikit-Learn has `train_test_split()`.
- Scikit-Learn also has `StratifiedShuffleSplit()` that does stratified sampling.
- **Stratification** ensures that the test samples are representative of the target categories.

Preserve percentage of
Samples for each class

2.2.1. Create a Test Set: User-defined function

dataFrame

```
import numpy as np

def split_train_test(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
```

You can then use this function like this:

```
>>> train_set, test_set = split_train_test(housing, 0.2)
>>> print(len(train_set), "train +", len(test_set), "test")
16512 train + 4128 test
```

shuffled_indices [:test_set_size] → من البداية لـ test_set_size

shuffled_indices [test_set_size:] → من test_set_size للنهاية

Install Jupyter Notebook.

generate same random numbers on multiple executions.

np.random.seed(42) → save the state of randomness

2.2.2. Create a Test Set: Using Scikit-Learn functions

الدالة مش موجودة بال builtin libraries

Permutation بس عشان 42

default value = 42 → magic number

```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```

Stratification is usually done on the target class.

عدد ال splits

```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing, housing["income_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

to get the same train & test sets across different executions.

18

هون بنحكيله الداتا اللي بيعملها split

وهنا ع أي شو يعمل split

column

Outline

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

لازم كل class يتدرب عليها ال model ويفحص حاله فيها مثلا لو كان عنا class معين نسبته بالداتا 50% و class ثاني نسبته 50% لازم يكون ال training set وال test set مقسمة بين ال 2 classes بشكل متناسب مع حجم الداتا فيهم

3. Discover and Visualize the Data to Gain Insights

- **Visualize** geographical data using

```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
    s=housing["population"]/100, label="population",
    c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
)
plt.legend()
```

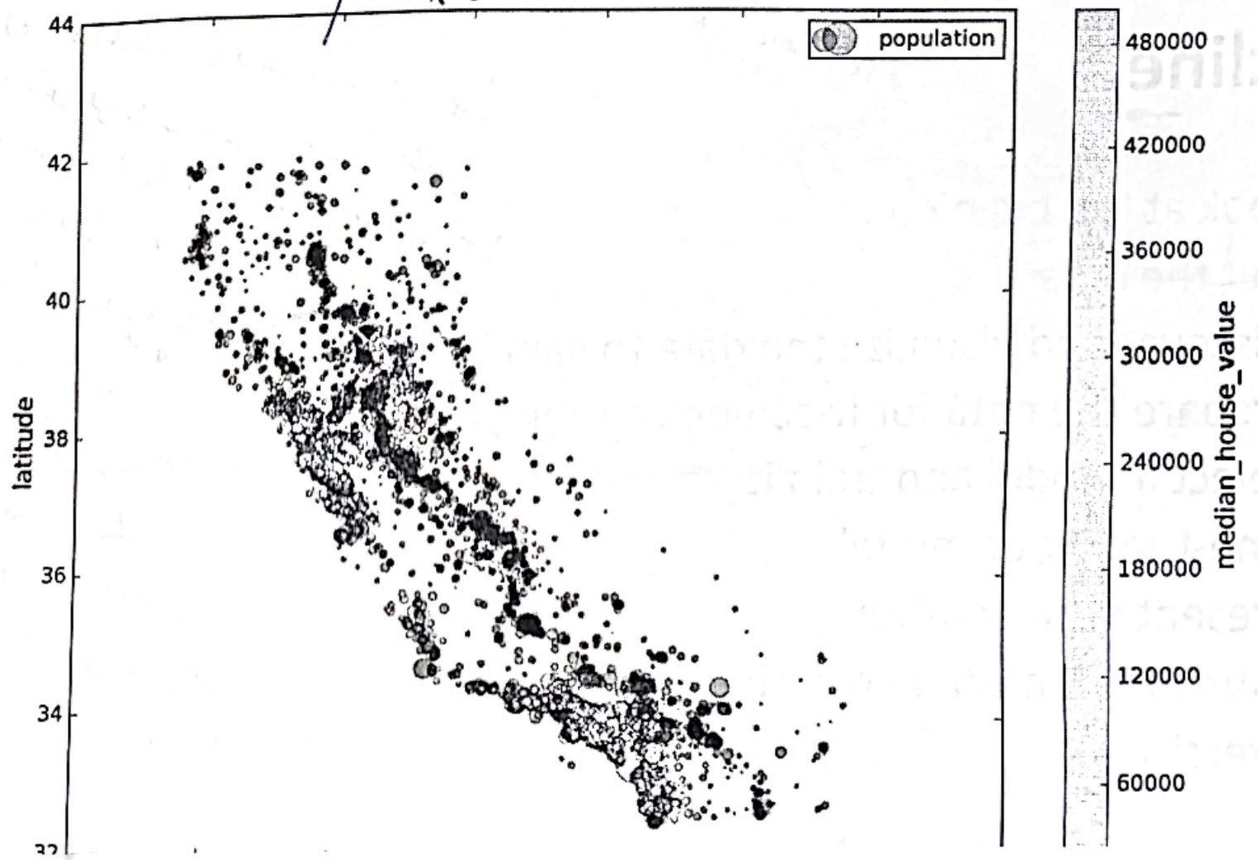
الداتا الي بنمثلها

الشفافية

`alpha`: Transparency, `s`: size, `c`: color, `cmap`: blue to red

size of the dot

Scatter Plot



Correlation can tell if two variables have a linear relationship

because I don't need all the features

3.1. Looking for Correlations

- Compute the **standard correlation coefficient** (also **called Pearson's** *r*) between every pair of attributes using `corr_matrix = housing.corr()`

(to find correlations between each feature with others) Pandas ال يعود

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value    1.000000
median_income         0.687170
total_rooms           0.135231
housing_median_age    0.114220
households            0.064702
total_bedrooms        0.047865
population           -0.026699
longitude            -0.047279
latitude             -0.142826
```


measured by
1. Corr function
2. scatter in pandas.

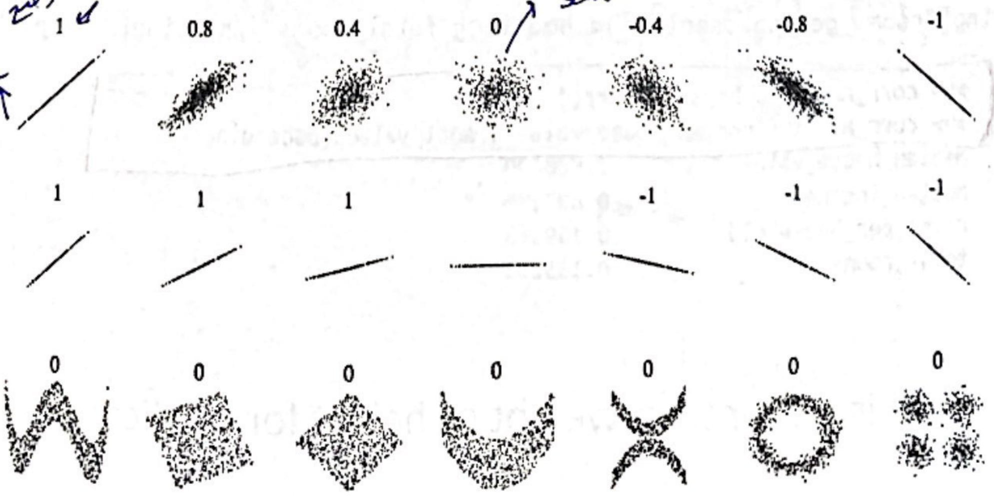
3.1. Looking for Correlations

- Zero linear correlation (*r* = 0) does not guarantee **independence**.

correlation ال اولى

linear between x and y

random scatter

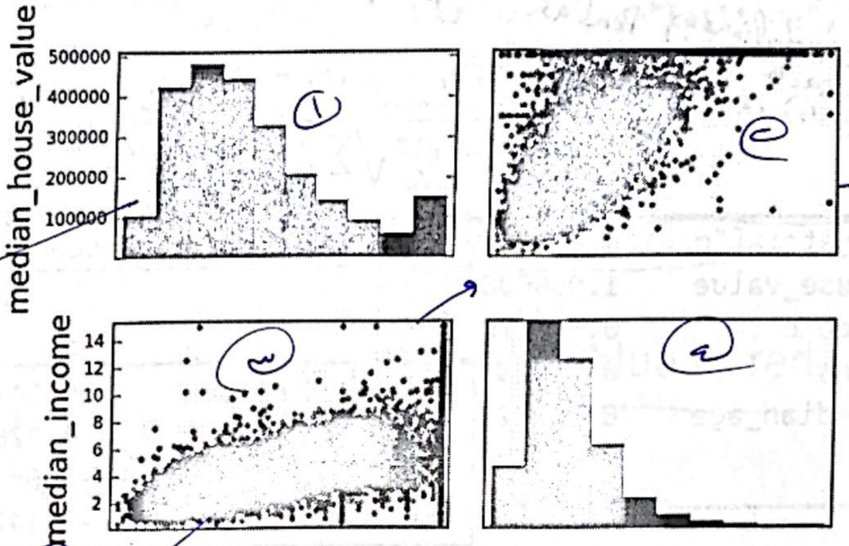


3.2. Pandas Scatter Matrix

pandas.plotting

```
from pandas.tools.plotting import scatter_matrix
attributes = ["median_house_value", "median_income"]
scatter_matrix(housing[attributes], figsize=(12, 8))
```

الي بدنا نوجد ال correlations بينها .



histogram

(ما في داعي يرسم ال correlation بين ال feature و نفسها لانه دايما = 1)

عكسنا ال axis مقارنة مع الصورة الثالثة

Correlation في income و house value

3.3. Experimenting with Attribute Combinations

- Rooms per household is better than total rooms:

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
```

```
>>> corr_matrix = housing.corr()
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value          1.000000
median_income               0.687170
rooms_per_household         0.199343
total_rooms                 0.135231
```

- Similarly, BMI is better than weight or height for medical purposes.

Outline

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

4. Prepare the Data for Machine Learning Algorithms

ما بيتريكونوا بنفس ال data frame

label ما بيتريكون ال

جيسمع training set

training test

Target Value

un supervised learning

- **Split** to train and test (Done)
- **Separate** features from response

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots$

- Handle **missing** data

بحولها لارقام

- Handle text and **categorical** features

مثلا بعطي Nearbay 0

Ocean 1

and so on.

- **Scale** (normalize) features
- Build preparation **pipeline**

4. Prepare the Data for Machine Learning Algorithms

ال data frame
الي فيه ال training data

- Separate the features from the response. remove from training data set Column

جدول نسخ ما عدا على ال data الاصلية

```
housing = strat_train_set.drop("median_house_value", axis=1)
housing_labels = strat_train_set["median_house_value"].copy()
```

صار هون ال labels

- Options of handling missing features:
  1. Get rid of the corresponding districts
  2. Get rid of the whole attribute
  3. Set the values to some value (0, mean, median, etc.)

ال value not available

drop row

```
housing.dropna(subset=["total_bedrooms"])     # option 1
housing.drop("total_bedrooms", axis=1)        # option 2
median = housing["total_bedrooms"].median()   # option 3
housing["total_bedrooms"].fillna(median, inplace=True)
```

عدل على نفس ال data frame

4.1. Handling Missing Features Using Scikit-Learn

- Use `SimpleImputer` on the numerical features (only). Need to remove categorical variables before doing the fit. The attribute `statistics_` has the means.

housing_num = housing.select_dtypes(include=[np.number])

```
from sklearn.~~preprocessing~~ impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fit(housing_num)
>>> imputer.statistics_
array([ -118.51 , 34.26 , 29. , 2119. , 433. , 1164. , 408. , 3.5414])
>>> housing_num.median().values
array([ -118.51 , 34.26 , 29. , 2119. , 433. , 1164. , 408. , 3.5414])
X = imputer.transform(housing_num)
```

missing features = median

not numerical

estimation median values

median of all features

NumPy array

next slide

روح على كل feature تبع missing values
عبيها بال median تبع هاي ال feature

Convert the array X into ~~dataFram~~

4.2. Handling Text and Categorical Attributes

- ocean_proximity is categorical feature.

```
>>> housing_cat = housing[["ocean_proximity"]]
>>> housing_cat.head(10)
      ocean_proximity
17606       <1H OCEAN
18632       <1H OCEAN
14650      NEAR OCEAN
3230           INLAND
3555        <1H OCEAN
19480          INLAND
8879        <1H OCEAN
13685          INLAND
4937        <1H OCEAN
4861        <1H OCEAN
```

indices why not 0, 1, 2 ...? because of shuffling

Shuffeling before splitting

4.2. Handling Text and Categorical Attributes

- Most machine learning algorithms prefer to work with numbers.
  Converting to numbers:

بالترتيب

categories لعدال وبعطي كل categor. رقم

```
>>> from sklearn.preprocessing import OrdinalEncoder
>>> ordinal_encoder = OrdinalEncoder()
>>> housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
>>> housing_cat_encoded[:10]
array([[0.],
       [0.],
       [4.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.]])
```

↳ like you call fit then transform

Numerical values imply distances

بيعطيني الترتيب اللي مشي عليه بالارقام

```
>>> ordinal_encoder.categories_
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
      dtype=object)]
```

0 1 2 3 4

برتبهم تصاعدي حسب ال ASCII code

↳ problem in ordinal encoding ?

الارقام اللي بحطها ما الها معنى وما الها علاقة ببعض.

4.2. Handling Text and Categorical Attribu

categories عدد ال = digits عدد ال

Common

- To ensure encoding neutrality, we can use the one-hot encoding.

```
>>> from sklearn.preprocessing import OneHotEncoder
>>> cat_encoder = OneHotEncoder()
>>> housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
>>> housing_cat_1hot
<16512x5 sparse matrix of type '<class 'numpy.float64'>'
 with 16512 stored elements in Compressed Sparse Row format>
>>> housing_cat_1hot.toarray()
array([[1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       ...,
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

Converts sparse matrix to dense matrix.

عدد ال rows = عددها بالاضبط

عدد ال columns = عدد ال categ

نوع من ال optimization يعني مثلا بكون عارف انو ال matrix مليانه اصفار عشان اقدر اوفر بال memory.

بكون فيها عدد ال اصفار كثير

اذا كان عندي اصفار كثير فلا بخزن بطريقة compact اكثر

بخزن وين ال ones موجودين (اماكنهم)

.info → بحكيلي كم عدد ال attributes الي ما الهم values

بنحتاج الداتا تكون numerical لانه بالنهاية بكون عنا mathematical model

4.3. Custom Transformers

- Scikit-Learn allows you to create your **own transformers.**
- You can create a transformer to create **derived features.**
- Create a class and implement three methods: `fit()` (returning self), `transform()`, and `fit_transform()`. Include base classes:
  - `TransformerMixin` to get `fit_transform()`
  - `BaseEstimator` to get `get_params()` and `set_params()`

fit then transform

4.3. Custom Transformers

get and set Parameters   fit_transform

```
from sklearn.base import BaseEstimator, TransformerMixin
```

Column رقم ال   base classes to inherit some methods from.

```
rooms_ix, household_ix = 3, 6

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X, y=None):
        rooms_per_household = X[:, rooms_ix] / X[:, household_ix]
        return np.c_[X, rooms_per_household]

attr_adder = CombinedAttributesAdder()
housing_extra_attribs = attr_adder.transform(housing.values)
```

rooms عدد ال   كل ال columns

Column جديد

numpy array دمجهم بـ

column بس قسمنا column على column لأنه fit ما احتجنا ال

وما احتجنا average مثلاً .

in Pandas

.describe → show min / average / max / Percentaise for each Column

4.4. Feature Scaling

بتكون قبل ال training

بيسهل على ML

features لما يكون في تفاوت بالقيم لل

يفضل نعمل scaling

- ML algorithms generally **don't perform well** when the input numerical attributes have **very different scales.**

بتزيد ال accuracy وبكون الأوزان ادق لل features

على كل قيمة

- Scaling techniques:

1. - **Min-max scaling** → ~~...~~

   ↳ more sensitive to outlayers values

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

sample

* بطبقه بكل قيمة في ال Feature

2. - **Standardization** (standard scaler)

   ↳ better when we have outlayer values

   more Popular

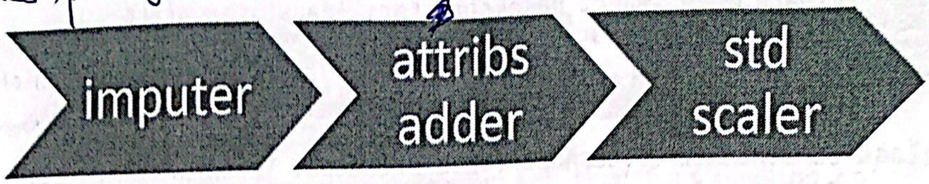
$$x' = \frac{x - \bar{x}}{\sigma}$$

sample   average   standard deviation

4.5. Transformation Pipelines

* Fill missing values

~~Adding~~




```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('attribs_adder', CombinedAttributesAdder()),
    ('std_scaler', StandardScaler()),
])

housing_num_tr = num_pipeline.fit_transform(housing_num)
```

اي تعديل هيصير ال Pipeline عنده

Work on numeric Values.

object

list of parameters "Stages" of the pipeline.

column3/column

بيعت الداتا الجديدة هون

housing_num_training

Data frame of numeric Values
without ocean proximity Column.
Column of Strings

4.6. Full Pipeline

```
from sklearn.compose import ColumnTransformer

num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]

full_pipeline = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", OneHotEncoder(), cat_attribs),
])

housing_prepared = full_pipeline.fit_transform(housing)
```

بخزن فيك اسماء Columns الي فيها numeric Values

Caterogical Values.

Super Scalar

Dense array

as : num_pipeline.fit_transform(housing[num attribs])

* اذا انا ما غطيت كل ال columns → "remainder function.

go read column Transformer sklearn

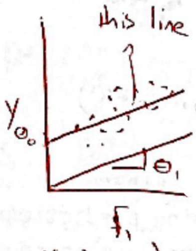
Outline

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

5. Select and Train a Model

- Linear regressor → linear relationship between feature and label
- Using RMSE for evaluation → Root Mean Square error
- Decision tree regressor → for non-linear relationships
- k-fold cross validation
- Random forests regressor → مجموعة من الـ Decision trees

this line is better



* we need to find relation between f_1 & Y

5. Select and Train a Model

- Let us start by training a simple **linear regressor.**

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)
```

→ object

ال colm للداتا الي بدي اتوقعها

Data → ما فيها الداتا الي بدي اتوقعها

- Try it out on five instances from the training set.

نفصل ال data عن ال labels

اخذ 5 samples

```
>>> some_data = housing.iloc[:5]
>>> some_labels = housing_labels.iloc[:5]
>>> some_data_prepared = full_pipeline.transform(some_data)
>>> print("Predictions:\t", lin_reg.predict(some_data_prepared))
Predictions:     [ 303104.   44800.  308928.  294208.  368704.]
>>> print("Labels:\t\t", list(some_labels))
Labels:          [359400.0, 69700.0, 302100.0, 301300.0, 351900.0]
```

50% off

↳ actual values

→ by pipeline

* all data preparation on training data must be also done on Test data or you can just do all preparations on the whole data befor you split it into train & test.

5.1. Evaluate the Model on the Entire Training Set

• linearRegession → بنجيب ال RMSE الها

- Use RMSE → manually

```
>>> from sklearn.metrics import mean_squared_error
>>> housing_predictions = lin_reg.predict(housing_prepared)
>>> lin_mse = mean_squared_error(housing_labels, housing_predictions)
>>> lin_rmse = np.sqrt(lin_mse)
>>> lin_rmse
68628.413493824875
```

This is not a satisfactory result as the **median_housing_values** range between \$120,000 and \$265,000.

Since this is huge error, we will try another Model which is ~~Descisio~~ Decision Tree Regressor.

5.2. Try the Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor()
tree_reg.fit(housing_prepared, housing_labels)

>>> housing_predictions = tree_reg.predict(housing_prepared)
>>> tree_mse = mean_squared_error(housing_labels, housing_predictions)
>>> tree_rmse = np.sqrt(tree_mse)
>>> tree_rmse
0.0
```

• عمل predict على نفسه الداتا اللي تدرب عليها فطلع ال 0 = error

Overfitting: It has memorized the entire training set!

→ If I want to test it, give the model test data that it ~~had~~ had never seen before and check it's prediction

26/3

Ai

Pages 139 - 150 from the book

- FunctionTransformer → estimator takes certain function we want to apply on a certain column as a parameter "we have to import it." → transform the data according to a certain function

optional

ex:- log_transformer = FunctionTransformer(np.log, inverse_func=np.exp)
log_pop = log_transformer(housing[["Population"]])

Takes the log of the elements in Population column.

- usually we use log when we have heavy tail. (then scaling).

- rbf_kernel → similarity measure function.

- fit function → calculates the mean, median ... etc
- transform → apply operations ~~on the mean~~ that includes the mean, ... etc

- check_array(x) → checks that x is an array with finite float values
- check_is_fitted(var.) → checks that the parameters are already initialized.

default_num_Pipeline → imputation then scaling

- get_feature_names_out() → returns Columns names

- ratio_pipeline() → calculates the ratio between 2 columns.
returns the word "ratio"

5.1. Evaluate the Model on the Entire Training Set

```
>>> from sklearn.metrics import mean_squared_error
>>> lin_rmse = mean_squared_error(housing_labels, housing_predictions,
...                               squared=False)
...
>>> lin_rmse
68687.89176589991
```

If squared = false, RMSE
If squared = true, MSE

5.2. Try the Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = make_pipeline(preprocessing, DecisionTreeRegressor(random_state=42))
tree_reg.fit(housing, housing_labels)
```

Now that the model is trained, you evaluate it on the training set:

```
>>> housing_predictions = tree_reg.predict(housing)
>>> tree_rmse = mean_squared_error(housing_labels, housing_predictions,
...                                squared=False)
...
>>> tree_rmse
0.0
```

Scanned with CamScanner

5.3. Better Evaluation Using Cross-Validation

Model to train

```
from sklearn.model_selection import cross_val_score

tree_rmses = -cross_val_score(tree_reg, housing, housing_labels,
                              scoring="neg_root_mean_squared_error", cv=10)
```

Negative root mean squared error

,Higher is better

هيك مبدأ ال scoring

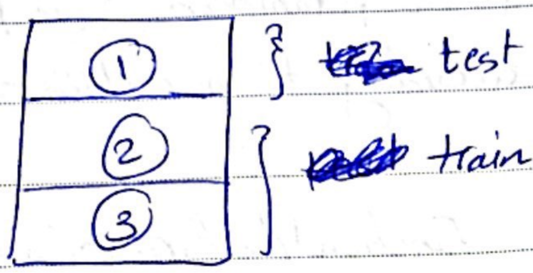
```
>>> pd.Series(tree_rmses).describe()
count      10.000000
mean    66868.027288
std      2060.966425
min     63649.536493
25%     65338.078316
50%     66801.953094
75%     68229.934454
max     70094.778246
dtype: float64
```

cross validation rounds

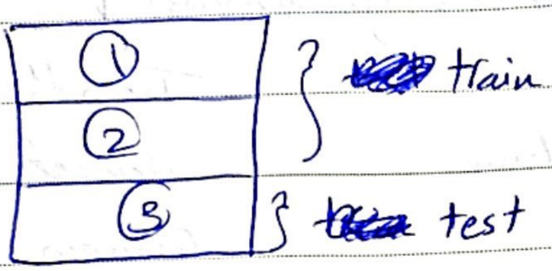
• Cross validation (CV)

if CV = 3 ⟶ 3 rounds (it'll divide the data into 3 parts, 3 times)

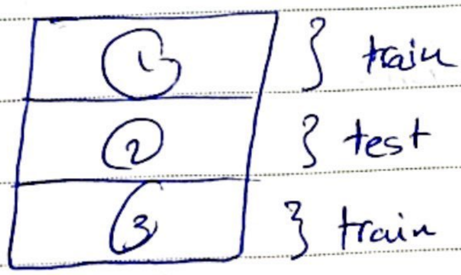
1



2



3



5.4. Try the Random Forests Regressor

- Repeating training and evaluation:

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = make_pipeline(preprocessing,
                           RandomForestRegressor(random_state=42))
forest_rmses = -cross_val_score(forest_reg, housing, housing_labels,
                                scoring="neg_root_mean_squared_error", cv=10)
```

```
>>> pd.Series(forest_rmses).describe()
count      10.000000
mean    47019.561281
std      1033.957120
min     45458.112527
25%     46464.031184
50%     46967.596354
```

Best Accuracy
Overfitting??

مجموعة من ال decision trees وكل واحد بعطي الجواب تبعه

5.4. Try the Random Forests Regressor

- Repeating training and evaluation:

```
>>> from sklearn.ensemble import RandomForestRegressor
>>> forest_reg = RandomForestRegressor()
>>> forest_reg.fit(housing_prepared, housing_labels)
>>> [...]
>>> forest_rmse
18603.515021376355
>>> display_scores(forest_rmse_scores)
Scores: [49519.80364233 47461.9115823  50029.02762854 52325.2806895
 49308.39426421 53446.37892622 48634.8036574  47585.73832311
 53490.10699751 50021.5852922 ]
Mean: 50182.303100336096
Standard deviation: 2097.0810550985693
```

on training data

on test data.

لما يكون جواب ال training افضل من جواب ال test ← overfitting

ال score

لما يكونوا الجوابين سيئين ← underfitting

Outline

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

ال Parameters الوحيدين اليوم المهم

Training data and labels ← default value

ولو بدنا نعدل ال hyperparameter بنحتاج عدد تجارب كبيرة.

6. Fine-Tune Your Model

- Fine-tune your system by fiddling with:
  - The hyperparameters
  - Removing and adding features
  - Changing feature preprocessing techniques
- Can experiment manually. But it is best to automate this process using Scikit-Learn: → Cross Validation
  - **GridSearchCV** →
  - or **RandomizedSearchCV**

بعطيه ال parameters اللي بدي اجربهم و قيم
اللي بدي اعطيها لهاي ال parameters و بعمل تجارب لل Parameters

6.1. Grid Search

- Can automate exploring a search space of $3 \times 4 + 2 \times 3 = 12 + 6 = 18$

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```

عدد ال decision trees

اول dictionary (١٢ تجربة)

list of dictionaries

ثاني dictionary

١٨ تجربة

كل تجربة 5 محاولات

بعمل كل التجارب بعدها بطلع النتائج

6.2 Examine the Results of Your Grid Sear

- Can examine the best hyperparameters using:

returns best result
احسن تجربة

```
>>> grid_search.best_params_
{'max_features': 8, 'n_estimators': 30}
```

- Can examine all search results using:

returns dictionary includes each set of parameters and thier scores.

```
>>> cvres = grid_search.cv_results_
>>> for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
...     print(np.sqrt(-mean_score), params)
...
63669.05791727153 {'max_features': 2, 'n_estimators': 3}
55627.16171305252 {'max_features': 2, 'n_estimators': 10}
        ...
49682.25345942335 {'max_features': 8, 'n_estimators': 30}
```

Best Tuned Accuracy

6.2 Evaluate Your System on the Test Set

- The final model is the best estimator found by the grid search.
- To evaluate it on the test set, transform the test features, predict using transformed features, and evaluate accuracy.

returns best model

Better than train set!

```
final_model = grid_search.best_estimator_
X_test = strat_test_set.drop("median_house_value", axis=1)
y_test = strat_test_set["median_house_value"].copy()
X_test_prepared = full_pipeline.transform(X_test)
final_predictions = final_model.predict(X_test_prepared)
final_mse = mean_squared_error(y_test, final_predictions)
final_rmse = np.sqrt(final_mse)   # => evaluates to 48,209.6
```

6.1. Grid Search (Updated)

- Can automate exploring a search space of $3 \times 3 + 2 \times 3 = 9 + 6 = 15$

مكون من اكتر من pipeline
وحسب طبيعة ال column كل
pipeline باخد المناسب

كل الداتا بتفوت على
random forest regressor

```python
from sklearn.model_selection import GridSearchCV

full_pipeline = Pipeline([
    ("preprocessing", preprocessing),
    ("random_forest", RandomForestRegressor(random_state=42)),
])
param_grid = [
    {'preprocessing__geo__n_clusters': [5, 8, 10],
     'random_forest__max_features': [4, 6, 8]},
    {'preprocessing__geo__n_clusters': [10, 15],
     'random_forest__max_features': [6, 8, 10]},
]
grid_search = GridSearchCV(full_pipeline, param_grid, cv=3,
                           scoring='neg_root_mean_squared_error')
grid_search.fit(housing, housing_labels)
```

الاسم الي بعد __ هو
Subclass
من الاسم الي قبل __

في pipeline اسمه geo هو subclass من ال pipeline
الي اسمه preprocessing،
و n_clusters هو parameter في geo.

6.2 Examine the Results of Your Grid Search (Updated)

- Can examine the best hyperparameters using:

```python
>>> grid_search.best_params_
{'preprocessing__geo__n_clusters': 15, 'random_forest__max_features': 6}
```

- Can examine all search results using:

```python
>>> cv_res = pd.DataFrame(grid_search.cv_results_)
>>> cv_res.sort_values(by="mean_test_score", ascending=False, inplace=True)
>>> [...] # change column names to fit on this page, and show rmse = -score
>>> cv_res.head() # note: the 1st column is the row ID
   n_clusters max_features split0 split1 split2 mean_test_rmse
12         15            6  43460  43919  44748          44042
13         15            8  44132  44075  45010          44406
14         15           10  44374  44286  45316          44659
7          10            6  44683  44655  45657          44999
9          10            6  44683  44655  45657          44999
```

6.2 Evaluate Your System on the Test Set

- The final model is the best estimator found by the grid search.

```
>>>final_model = grid_search.best_estimator
```

- If GridSearchCV is initialized with **refit=True** (which is the default), then once it finds the best estimator using cross-validation, it retrains it on the whole training set.

When refit=True
يعني باخد ال best model بالنهاية
وبدربه على كل الداتا
لانه مثلا في حالة CV=3 تم تدريبه
على ثلثين الداتا فقط
وفي حالة كان عنا ٨٠٪ من الداتا
training و ٢٠٪ test في حال refit
بتم تدريبه بالنهاية على ١٠٠٪ من الداتا

6.2 Evaluate Your System on the Test Set

```
X_test = strat_test_set.drop("median_house_value", axis=1)
y_test = strat_test_set["median_house_value"].copy()

final_predictions = final_model.predict(X_test)

final_rmse = mean_squared_error(y_test, final_predictions, squared=False)
print(final_rmse)  # prints 41424.40026462184
```

No overfitting

بس في حال كانت القيمة كتير بعيدة
عن القيمة المتوقعة يعني الداتا غير
كافية يعني في under-fitting

grid search

لكل ال parameters
الي بنعطيه اياهم combinations
وعلى عددهم بعمل التجارب
في ال RandomizedSearch هاد
بختار randomly عدد التجارب

6.2 Randomized Search

- Preferable, especially when the hyperparameter search space is large
- Run certain number of iterations
- Picks the hyperparameters values from the defined space.

بختار random number بين ال ٣ وال ٥٠

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {'preprocessing__geo__n_clusters': randint(low=3, high=50),
                  'random_forest__max_features': randint(low=2, high=20)}

rnd_search = RandomizedSearchCV(
    full_pipeline, param_distributions=param_distribs, n_iter=10, cv=3,
rnd_search.fit(housing, housing_labels) r', random_state=42)
```


6.3 Save Your Best Model for the Production System

- Save the model

```
import joblib

joblib.dump(final_model, "my_california_housing_model.pkl")
```

- Load the model

لما بدنا نعمله test على داتا جديدة

```
final_model_reloaded = joblib.load("my_california_housing_model.pkl")

new_data = [...]  # some new districts to make predictions for
predictions = final_model_reloaded.predict(new_data)
```

7. Present Your Solution

- Present your solution highlighting:
  - What you have learned
  - What worked and what did not
  - What assumptions were made
  - What your system's limitations are
- Document everything, and create nice presentations with:
  - Clear visualizations (scaling / size of the data / encoding)
  - Easy-to-remember statements, e.g., "the median income is the number one predictor of housing prices".

8. Launch, Monitor, and Maintain Your System

- Prepare your production program that uses your best trained model and launch it.
- Monitor the accuracy of your system. Also monitor the input data.
- Retrain your system periodically using fresh data.

Summary

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system
9. Exercises

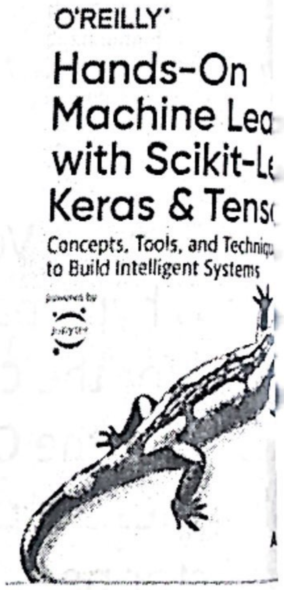
Exercise

- Try a Support Vector Machine regressor (`sklearn.svm.SVR`), with various hyperparameters such as `kernel="linear"` (with various values for the `C` hyperparameter) or `kernel="rbf"` (with various values for the C and gamma hyperparameters). Don't worry about what these hyperparameters mean for now. How does the best SVR predictor perform?

Classification

Prof. Gheith Abandah

Reference



- Chapter 3: **Classification**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Introduction

- YouTube Video: **Machine Learning - Supervised Learning Classification** from Cognitive Class

https://youtu.be/Lf2bCQlktTo

- classification → discrete values called classes

- مثلاً لو نعطيه صورة بيكيلنا خيارات شو ممكن تكون فبنختار الخيار الي اله اعلى احتمال
او ممكن يقارن الصورة مع اقرب خمس صور.

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

هي عبارة عن dataset مشهورة
تستعمل عادة في بداية اي موضوع
في ال ML وهي عبارة عن 70,000
صورة بكونوا hand written digits
كل صورة عبارة عن 28X28 pixels

1. MNIST Dataset

- **MNIST** is a set of 70,000 small images of **handwritten digits**.
- Available from mldata.org
- **Scikit-Learn** provides **download** functions.

1.1. Get the Data

*Fetching data needs Internet connection

```
from sklearn.datasets import fetch_openml

mnist = fetch_openml('mnist_784', as_frame=False)
```

Fetch the data from sklearn dataset

To get the data as Numpy array not Dataframe

Or we can use(make) to generate data

(Load) loads the data from your device

1.2. Extract Features and Labels

كل row عبارة عن صورة

- There are 70,000 images, and each image has **784** features. This is because each image is **28×28** pixels, and each feature simply represents one pixel's intensity, from **0** (**white**) to **255** (**black**).

```
>>> X, y = mnist.data, mnist.target
>>> X
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
>>> X.shape
(70000, 784)
>>> y
array(['5', '0', '4', ..., '4', '5', '6'], dtype=object)
>>> y.shape
(70000,)
```

مثلا اول row بمثل رقم 5 والتاني 0 وهكذا

1.3. Examine One Image

```
import matplotlib.pyplot as plt

def plot_digit(image_data):
    image = image_data.reshape(28, 28)
    plt.imshow(image, cmap="binary")
    plt.axis("off")
some_digit = X[0]
plot_digit(some_digit)
plt.show()
```

اول صورة بالداتا

```
>>> y[0]
'5'
```

1.4. Split the Data

- The MNIST dataset is actually already split into a **training set** (the first 60,000 images) and a **test set** (the last 10,000 images).
- The training set is **already shuffled.**

↳ that's why we split in this way.

```
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```

rows

In general, in machine learning:
X → for data
Y → for labels.

X
0
train
59999
test
69999
70000

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

Binary classifier
↓
No or yes جوابه
(two classes)

F
T
T
F
⋮

لازم نعدل على الـ y-labels لتصير 2 classes عبارة عن true or false
و على الداتا

2. Training a Binary Classifier → answer is True or False (yes or no)

- A binary classifier can classify two classes.
- For example, classifier for the number 5, capable of distinguishing between two classes, **5** and **not-5**.

labels بيعدل على ال → for every y element.

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)

from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

model

data labels

True for all 5s, False for all other digits.

Stochastic Gradient Descent (SGD) classifier

```
>>> sgd_clf.predict([some_digit])
array([ True])
```

fit → بيعمل boundary او curve بيفصل الداتا عن بعض (decision boundary)

return → binary array with 60000 raw including values of True & False

astype( ) → to change the type of data.

11

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

عشان نقيم
ال classifier

3. Performance Measures

correct answers / #of questions

غير كافية

- Accuracy: Ratio of correct predictions -> over all predictions.
- Confusion matrix
- Precision and recall
- F1 Score
- Precision/recall tradeoff

3.1. Accuracy

classifier

```
y_pred = clone_clf.predict(X_test_fold)
n_correct = sum(y_pred == y_test_fold)
print(n_correct / len(y_pred))
```

returns binary array

total number of predictions

correct predictions.

• y_test → فيها الاجابات الصح

• y_predict → اللي توقعه ال model

Example how to find the accuracy. (manual)

another method

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.96355, 0.93795, 0.95615])
```

يرجع accuracy لكل fold

Using the cross_val_score() function to find the accuracy on three folds

Check in documentation what are other options for this parameter?

sum of binary array

يرجع مجموع ال True

• لما نطبق اي algorithm لازم نكون عنا baseline نكون احسن منو

3.1. Accuracy

الأفضل نستعملها او cross_val_score
لما نيجي ندرب الداتا
عشان نقدر نقيمها

• Use cross_val_predict() to predict the targets of the entire training set.

```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

بيرجع لكل sample prediction values ~~xxxxxxxxxx~~

ما بيرجع score

Same Size

Predictions axis

	Not 5	5
Not5	TN	FP
5	FN	TP

Labels axis

Confusion matrix.

15

• هون بنكون الجواب بس ال diagonal (ideal)

افيد

$$\boxed{accuracy = \frac{TN + TP}{TN + TP + FN + FP}}$$

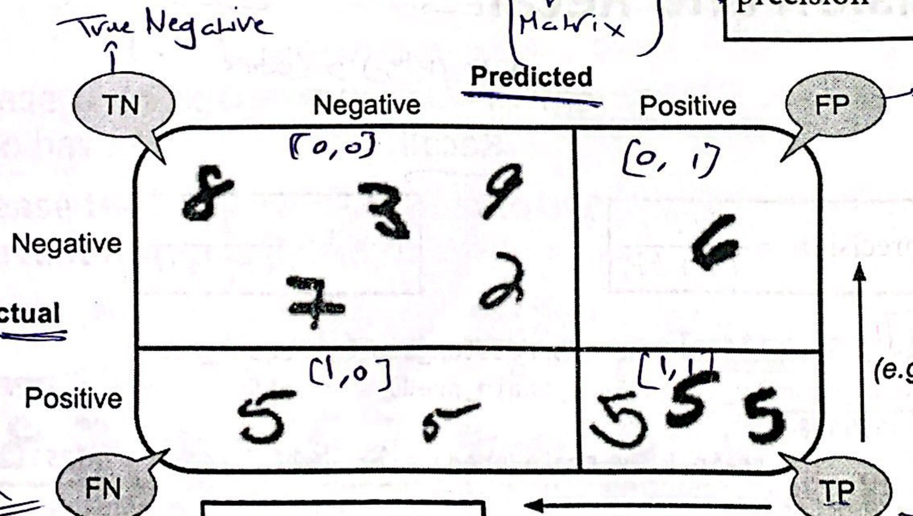
من كل الي توقعت اني 5 كم وحدة صح ؟

3.2. Confusion Matrix

Square Matrix

$$\boxed{precision = \frac{TP}{TP + FP}}$$

increase FP will decrease precision



• Positive → 5
• Negative → not5

ground Truth

False Negative

$$\boxed{recall = \frac{TP}{TP + FN}}$$

ratio of TP for all positive (Actual)

If all predictions are right the diagonal of Matrix will only have values, others zero

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & \checkmark & x & x \\ 2 & x & \checkmark & x \\ 3 & x & x & \checkmark \end{array}$$

16

• اذا كان ال recall قليل
في كثير FN

من كل ال 5 كم وحدة توقعت صح انو 5 ؟

.لما بدنا نعمل system الأفضل نركز على recall ونقلل FN (recall نركز على FN)
في حال ركزنا على Precision لازم نقلل FP . (Precision نركز على FP)
وحسب ال system بنختار Precision او recall او التنين.

3.2. Confusion Matrix

- Scikit Learn has a function for finding the **confusion matrix.**

```
>>> from sklearn.metrics import confusion_matrix
>>> confusion_matrix(y_train_5, y_train_pred)
array([[53057,  1522],
       [ 1325,  4096]])
```

- The first row is for the non-5s (the **negative** class):
  - 53,057 correctly classified (**true negatives**)
  - 1,522 wrongly classified (**false positives**)
- The second row is for the 5s (the **positive** class):
  - 1,325 wrongly classified (**false negatives**)
  - 4,096 correctly classified (**true positives**)

$$\begin{array}{ccc} & N & P \\ N & 53{,}057 & 1{,}522 \\ P & 1325 & 4096 \end{array}$$

3.3. Precision and Recall

بنعطيه ال label وال Predection

(ال Predection من cross_val_Predict)

Precision | **Recall**

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

open documentation

```
>>> from sklearn.metrics import precision_score, recall_score
>>> precision_score(y_train_5, y_train_pred) # == 4096 / (4096 + 1522)
0.7290850836596654
>>> recall_score(y_train_5, y_train_pred) # == 4096 / (4096 + 1325)
0.7555801512636044
```

> The precision and recall are smaller than the accuracy.
> Why?

لانه FP وال FN

accuracy_score (yt— — —) Comparable قيمتهم للقيمة TP

(y_test, y_pred) تأخذ

في حال اهتمينا بال precision وال recall

بدمج ال recall وال precision

3.4. F1 Score

بدنا نركز على FP و FN

• The F1 Score combines the precision and recall in one metric (harmonic mean). → Both precision & recall must be high to get high harmonic mean.

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

```
>>> from sklearn.metrics import f1_score
>>> f1_score(y_train_5, y_train_pred)
0.7420962043663375
```

بنعطيه ال labels وال predictions

بالإضافة لـ 8 لازم يكون عنا CM: [ ] والها Performance measure نعتمده.

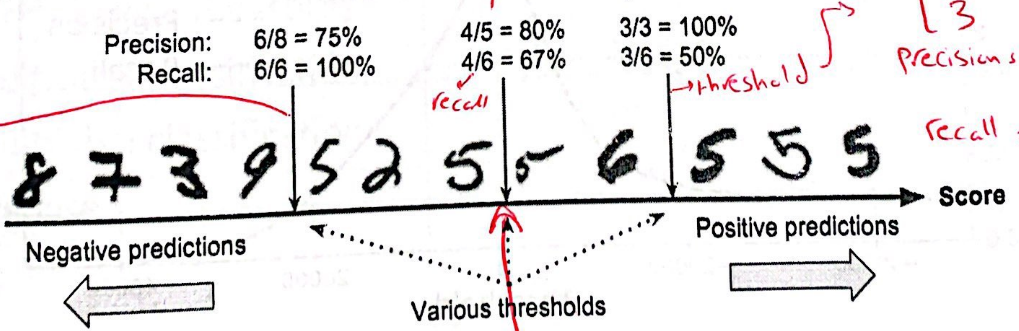
→ For this threshold

3.5. Precision/Recall Tradeoff *

* بحدد ال threshold ← على يمينه positive وعلى يساره Negative

• Increase the **decision threshold** to improve the precision when it is bad to have FP.

• Decrease the decision threshold to improve the recall when it is important not to miss FN.



Precision

recall

threshold

CM: $\begin{bmatrix} 6 & 0 \\ 3 & 3 \end{bmatrix}$

precision = $\frac{3}{3+0} = \frac{3}{3}$

recall = $\frac{3}{3+3}$

هدول عنا مو خمسة

For this threshold

هدول عنا خمسة

CM → $\begin{bmatrix} 5 & 1 \\ 2 & 4 \end{bmatrix}$

Confusion Matrix.

Accuracy = $\frac{5+4}{5+1+2+4} = \frac{9}{12}$

precision = $\frac{4}{4+1} = \frac{4}{5}$

recall = $\frac{4}{4+2} = \frac{4}{6}$

Scanned with CamScanner

- ↓threshold ↑FP ↓Precision
- ↓threshold ↓FN ↑recall

ممكن نعدل ال Threshold default وممكن احنا نحدده.

3.5. Precision/Recall Tradeoff

- The function **cross_val_predict()** can return decision scores instead of predictions.

```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                             method="decision_function")
```

عشان يرجع ال score

- These scores can be used to compute precision and recall for all possible thresholds using the **precision_recall_curve()** functio

```
from sklearn.metrics import precision_recall_curve

precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

يحسب ال precisions & recalls for all thresholds

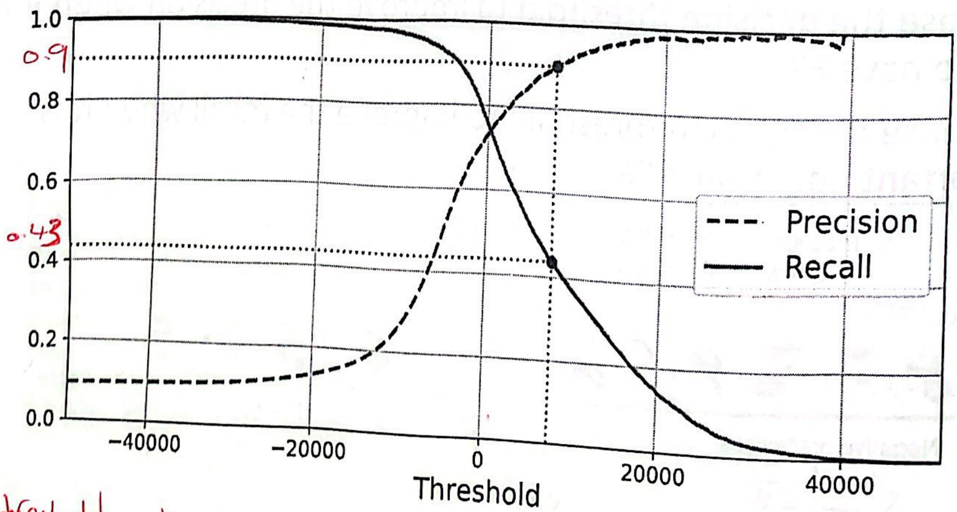
زي رسمة سلايد 20

(scores)

data X	label Y	y_score مثلا
5	true	3000
1	False	-100

بتبدأ ال threshold من اقل قيمة لل decision func لأعلى قيمة

3.5. Precision/Recall Tradeoff



اذا بدي ازيد ال precision بزيد ال threshold

اقلل ال recall

3.5. Precision/Recall Tradeoff

argmax() returns the index of the largest element in an array. If the array is boolean then it will return the index of the first occurrence of True

- For **larger precision**, **increase the threshold**, and **decrease it** for **larger recall**.

Precisions increase with the threshold

- **Example**: To get 90% precision.

```
>>> idx_for_90_precision = (precisions >= 0.90).argmax()
>>> threshold_for_90_precision = thresholds[idx_for_90_precision]
>>> threshold_for_90_precision
3370.0194991439557
y_train_pred_90 = (y_scores >= threshold_for_90_precision)
>>> precision_score(y_train_5, y_train_pred_90)
0.9000345901072293
>>> recall_at_90_precision = recall_score(y_train_5, y_train_pred_90)
>>> recall_at_90_precision
0.4799852425751706
```

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

our data is not only 5 or not 5 we have many classes "numbers"

لازم نعملهم detection ونميز بينهم ونعطي الخوارزمية جوابه

4. Multiclass Classification

- Multiclass classifiers can distinguish between **more than two classe**
- Some **algorithms** (such as Random Forest classifiers or Naive Bayes classifiers) are **capable of handling multiple classes** directly.
- **Others** (such as Support Vector Machine classifiers or Linear classifiers) are **strictly binary classifiers.**
- There are **two main strategies** to perform multiclass classification using multiple binary classifiers.

0-detector ⇒ 0 → True O.W False

5-detector = 5 → True O.W False

Probability باقدالعلى

4.1. One-versus-All (OvA) Strategy

لداتا صغيرة

- For example, classify the digit images into 10 classes (from 0 to 9) to **train 10 binary classifiers**, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).
- Then to classify an image, get the decision score from each classifier for that image and select the class whose classifier outputs the **highest score.**

بحالتنا عنا 10 classes يعني بنحتاج 10 models

وبنختار ال model اللي اله اعلى score

داتا كبيرة

مثلاً classifier - كانلنا هاي 1 or 5

True or False مش بس

بنختار 2 classes و بنعمل classifier يميز بينهم

4.2. One-versus-One (OvO) Strategy

- Train a binary classifier **for every pair** of digits.
- If there are N classes, need $\boxed{N \times (N - 1) / 2}$ classifiers. For MNIST, **need 45 classifiers.**
- To classify an image, run the image through all 45 classifiers and see which class **wins the most duels.**
- The main advantage of **OvO** is that each classifier only needs to be **trained on a subset** of the training set.
- OvO is preferred for algorithms (such as **Support Vector Machine**) that scale poorly with the size of the training set.

SVM

في هالحالة 10 classes بنحتاج 45 model

which one is better? OVA or OVO?

حسب طبيعة ال model وال complexity

4.3. Scikit Learn Support of Multiclass Classification

- **Scikit-Learn** detects when you try to use a binary classification algorithm for a multiclass classification task, and it automatic~~ally~~ runs **OvA** (except for **SVM** classifiers for which it uses **OvO**).

ال labels للداتا الاصلية الي القيم
فيها ممكن تاخد اي قيمة من صفر ل
٩

```
from sklearn.svm import SVC

svm_clf = SVC(random_state=42)
svm_clf.fit(X_train[:2000], y_train[:2000]) # y_train, not y_train_5
>>> svm_clf.predict([some_digit])
array(['5'], dtype=object)
>>> some_digit_scores = svm_clf.decision_function([some_digit])
>>> some_digit_scores.round(2)
array([[ 3.79, 0.73, 6.06, 8.3 , -0.29, 9.3 , 1.75, 2.77, 7.21,
        4.82]])
>>> class_id = some_digit_scores.argmax()
>>> class_id
5
```

4.3. Scikit Learn Support of Multiclass Classification

- Note that the multiclass task is harder than the binary task.
- Binary task

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.96355, 0.93795, 0.95615])
```

بس بتحكي 5 او مش 5 (Binary)

- Multiclass task

```
>>> cross_val_score(sgd_clf, X_train, y_train, cv=3, scoring="accuracy")
array([0.8489802 , 0.87129356, 0.86988048])
```

10 labels (more than 1 class)

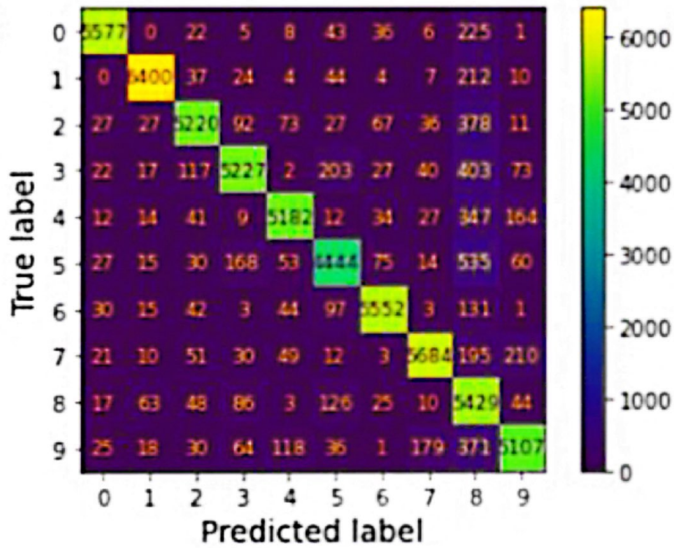
الفرق بال accuracy

Confusion matrix
صارت اكبر
بدل 2X2 بحالة binary classifier
صارت (10labels) 10X10

4.4. Error Analysis

```
from sklearn.metrics import ConfusionMatrixDisplay

y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred)
plt.show()
```

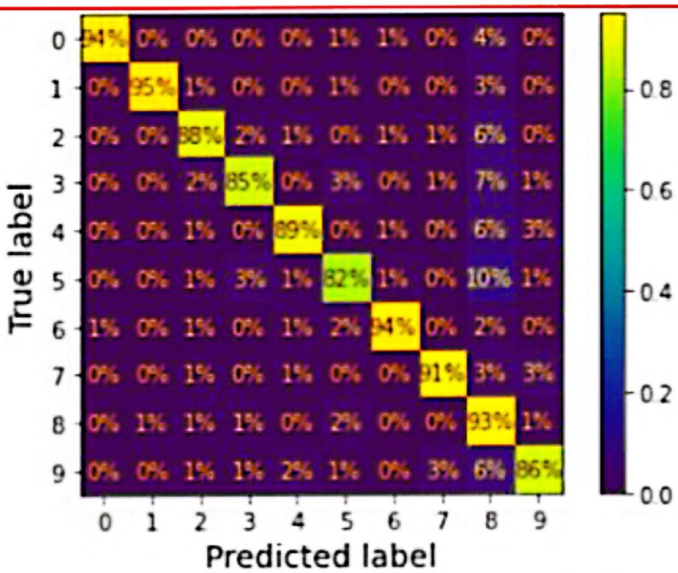


اي اشي برا ال diagonal يعني false prediction

مجموع عناصر ال row لازم يكون بساوي 1

4.4. Error Analysis

```
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
                                        normalize="true", values_format=".0%")
plt.show()
```



Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification →
6. Exercise

ال sample له اكثر من label
و ال labels ما الهم علاقة ببعض

5. Multilabel Classification

• Column for each label
#of pred. = #of labels

• Classifiers that output multiple classes for each instance.

```
y_train_large = (y_train >= 7)
y_train_odd = (y_train % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]
```

→ عبارة عن labels 2 columns

concatenation is a numpy array

```
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_multilabel)
```

← Popular algorithm

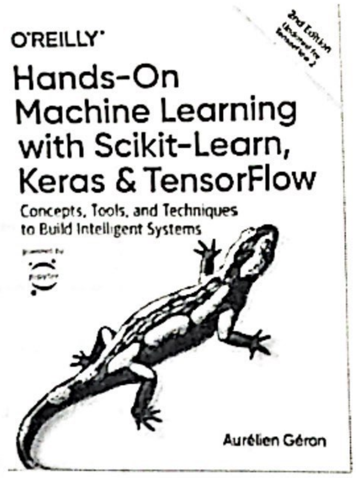
samples — multilabel for each sample

```
>>> knn_clf.predict([some_digit])
array([[False,  True]], dtype=bool)
```

Training Models and Regression

Prof. Gheith Abandah

Reference



- Chapter 4: **Training Models**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Outline

1. Linear Regression
2. Gradient Descent ⟶ optimization techniques وحدة من ال
3. Gradient Descent Variants
   1. Batch Gradient Descent
   2. Stochastic Gradient Descent
   3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

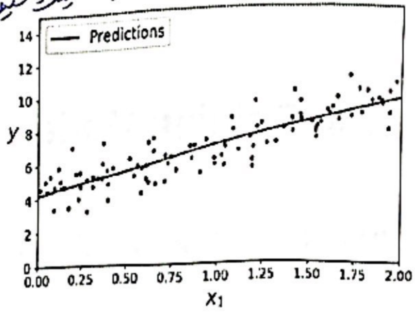
bias term → نتحكم بال curve
كيف بطلع او بنزل
$\theta_1 \to$ slope

Linear Regression → Relationship between x and y
(" العلاقة الي لو اعطيتك قيمة x توقعت y")

~~bias value~~ → bias term

$$\boxed{\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n}$$

label / الجواب ←
↳ weight of the feature



- $\hat{y}$ is the predicted value.
- n is the number of features.
- x_i is the i^{th} feature value.
- θ_j is the j^{th} model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \cdots, \theta_n$).

بضرب ال row بال Colum كامل.

في حالة MNIst → $n = 784$

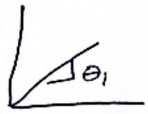
$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad x \quad m \begin{bmatrix} x_1 & \cdots & x_n \\ x_2 & \cdots & x_n \\ & & \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_n \end{bmatrix} \hat{y}$$

$$\boxed{\hat{y} = h_\theta(\mathbf{x}) = \theta \cdot \mathbf{x}}$$

↳ hypothesis (prediction)
فرضية

random values ل

عدد ال Samples



Analytical Solution

كم ال prediction
بعيد عن ال actual value

بدي اقلله .

- The Root Mean Square Error (RMSE) is used as **cost function.**

$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}$ في حال كان عالي بنعدلهم

$$\boxed{\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T \mathbf{x}^{(i)} - y^{(i)}\right)^2}$$

- Minimizing this cost gives the following solution (**normal function**):

→ higher complexity

optimization

قيم θ النهائية ← $\hat{\theta} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \ \mathbf{X}^T \ \ \mathbf{y}$ ← Complexity $O(mn^2)$

بدل ما نفترض ارقام ونعدل.
more complexity بس

- $\hat{\theta}$ is the value of θ that minimizes the cost function.
- $\mathbf{y}$ is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

m: no. of Samples
n: no. of features.

Outline

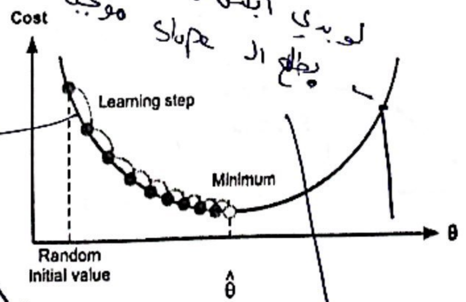
1. Linear Regression
2. Gradient Descent → optimization algorithm ("to modify the values of θ) لنوصل لنتيجة ~~x~~ اقرب لل target
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

→ this is not cost function it is a way (algorithm) to minimize the cost function (MSE)

Gradient Descent → Can be run in parallel.

- **Generic optimization algorithm** capable of finding optimal solutions to a wide range of problems.
- **Tweaks parameters** iteratively in order **to minimize a cost function.**

لو بدي ابلش شو صوب موجب يطلع ال Slope



لما اشتق بطلع ال slope سالبي

بزيد θ لحتى اوصل لل min

بالتالي ح اقلل θ لحتى اوصل لل min

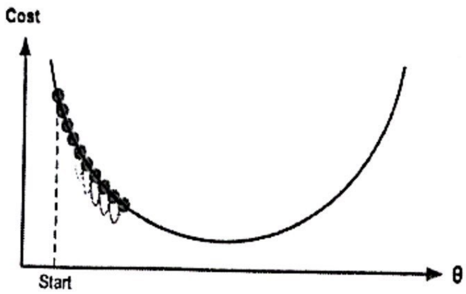
$$\boxed{\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \text{MSE}(\theta)}$$

derivative → بالنسبة لل θ

learning rate

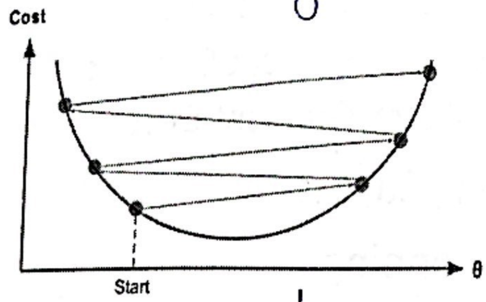
Learning Rate <1

Too Small



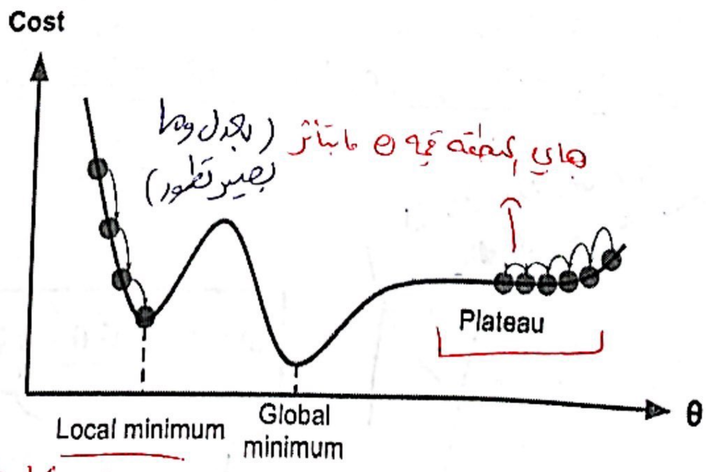
Too Large

overshooting



مثلاً في حال كان learning rate =1
ممكن يعمل skip لل optimal value

Gradient Descent Pitfalls



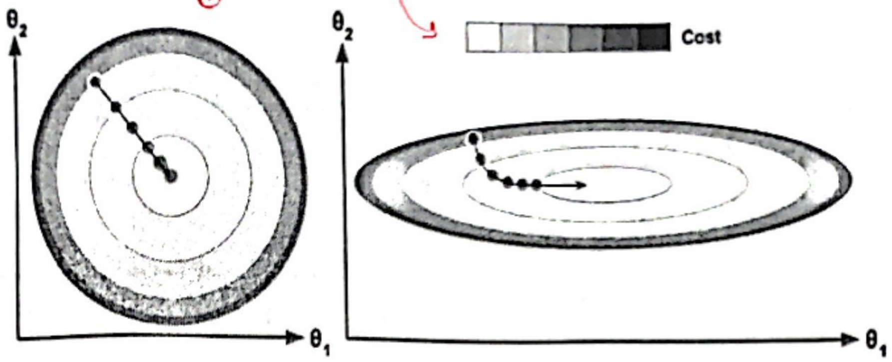
لما يلاقيها ما رح يقدر يوصل
لل Globalmin

Feature Scaling

الهدف تكون قيم ال features قريبة من بعض.

بسرع ال convergence

- Ensure that all features have a similar scale (e.g., using Scikit-Learn's `StandardScaler` class). → used in Transformation pipeline
- Gradient Descent with and without feature scaling.



مع ال Scaling ال Cost بتقل

Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
   1. Batch Gradient Descent
   2. Stochastic Gradient Descent
   3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

$$MSE(X, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})$$

Batch Gradient Descent

بمر على كل training set
ويحسب ال Average loss

- **Partial derivatives** of the cost function in θ_j

$$\frac{\partial}{\partial \theta_j} \text{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T \mathbf{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

بعد اشتقاق الطرفين من معادلة MSE المرة الأولى بالنسبة θ_0 بعدين θ_1 ...

$i \rightarrow$ رقم السامبل
$j \rightarrow$ feature

- **Gradient vector** of the cost function

بشتق بالنسبة ل θ

$$\nabla_\theta \text{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\theta) \end{pmatrix} = \boxed{\frac{2}{m} \mathbf{X}^T (\mathbf{X}\theta - \mathbf{y})}$$

X transpose

The entire training Batch

Features

$$\boxed{\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \text{MSE}(\theta)}$$

$\theta^T x = \theta_0 + \theta_1 F_1 + \theta_2 F_2$

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(\overbrace{(\theta_0 + \theta_1 F_1^{(i)} + \theta_2 F_2^{(i)})}^{\theta^T x} - y \right)$$

$$\frac{dMSE}{\theta_0} = \frac{2}{m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 F_1^{(i)} + \theta_2 F_2^{(i)}) - y \right) \cdot 1$$

$$\frac{dMSE}{\theta_1} = \frac{2}{m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 F_1^{(i)} + \theta_2 F_2^{(i)}) - y \right) \cdot F_1$$

بعد الاشتقاق بطلعلنا ال slope لل cost function باتجاه θ

12

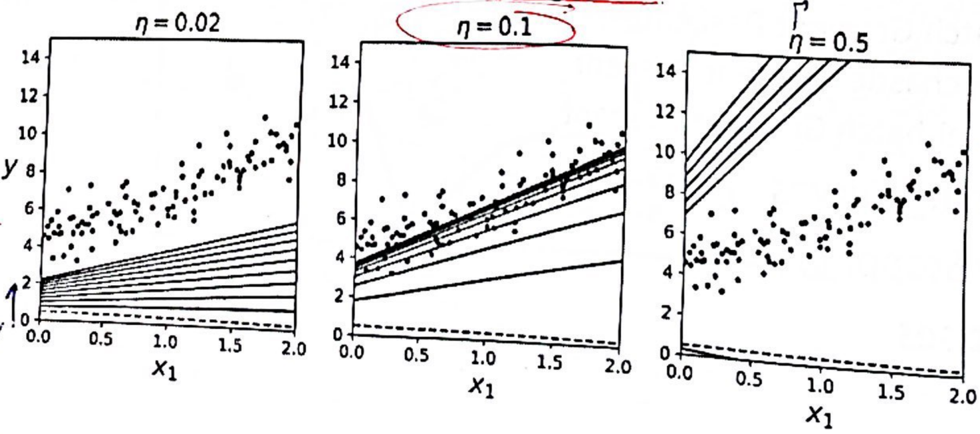
Batch Gradient Descent

- **Gradient Descent step**

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \text{MSE}(\theta)$$

- Gradient Descent with **various learning rates**

عالي



رح ياخذ عدد خطوات كبير لحتى يوصل ل min

ال curve بتبدل بس لما وصلنا

13

Full gradient descent → لازم نمر على كل ال samples

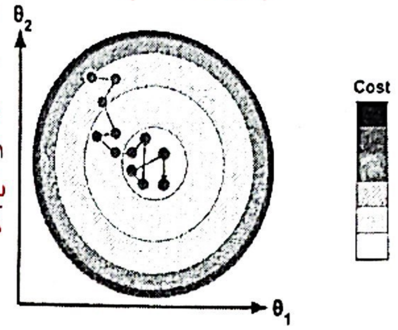
stochastic " " → باخذ sample واحدة بكل مرة

Stochastic Gradient Descent (SGD)

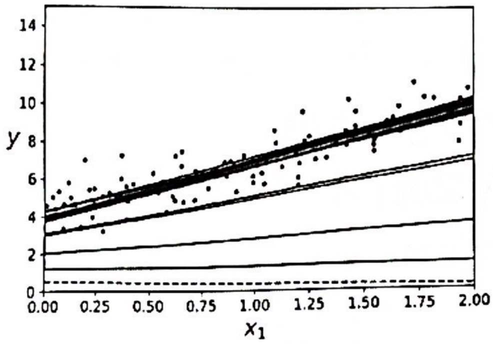
$((\theta_0+\theta_1 F_1+\theta_2 F_2)-y)^2 \rightarrow$ بس من المعادلة الي بشتغل عليها

- SGD picks a random instance in the training set at every step and computes the gradients.
- SGD is faster when the training set is large.
- Is bouncy
- Eventually gives **good solution**
- Can escape local minima

بناءً على النتيجة اللي بتطلع منه بنعدل ال weights of θ



* هاي الطريقة تعتمد على ال Sample .

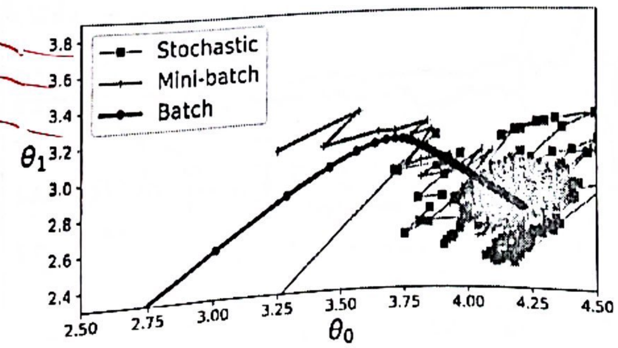


نختار مجموعة samples مش بس واحد

Mini-batch Gradient Descent

ما باخذ ال data كاملة وبنفس الوقت ما باخذ Samples بشكل عشوائي.

- Computes the gradients on small random sets of instances called **mini batches.**
- Benefits from **hardware accelerators** (e.g., GPU).
- Less bouncy, better solution, escapes some local minima



Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

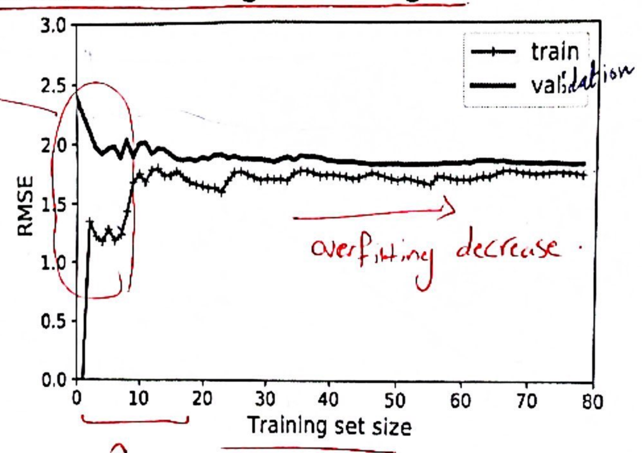
Learning Curves

- The **accuracy** on the **validation set** generally **increases** as the training set size increases.
- **Overfitting decreases** with larger training set.

Accuracy

انو مثلا 90% عال training و 50% عال test

Over fitting



because the training set size is small

Outline

. بالعادة لما تكون ال data set صغيرة تكون underfitting لأنه الداتا ما بتكون شاملة كل ال samples

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
   1. Batch Gradient Descent
   2. Stochastic Gradient Descent
   3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

اذا ما بتحسن ال RMSE "ما في تطور" بوقف

Early Stopping

→ to save time & prevent overfitting:

. مش شرط نلاقي ال Best model لما نشوف كل ال Ephochs.

- **Stop** training when the **validation error reaches a minimum.**
- Need to **save the best model.**

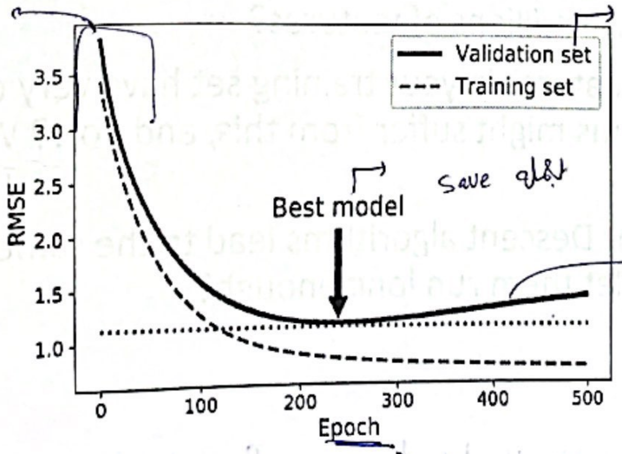
(bias) بالبداية

Model isn't good niether at training set nor validation

قل بعدين زاد

save بنحفظ

رجع يزيد



we can say here is overfitting

لـ كل ما زدنا يقل ال error & training loss

لانه ال training RMSE بقل وال validation RMSE بتزيد.

يعني لما ادرب ال samples وأعدل ال weight

هاد اسمه epoch ← [

دليل انه حفظ ال training data

يعني مثلا بعطيه صورة وبدي يعطيني الجواب بالزبط
او الكلاس، بنطلع probability ازا كانت
اعلى من threshold معين يعني positive class
ازا اقل يعني negative class

ما بعطينا قيمة
بعطينا احتمال

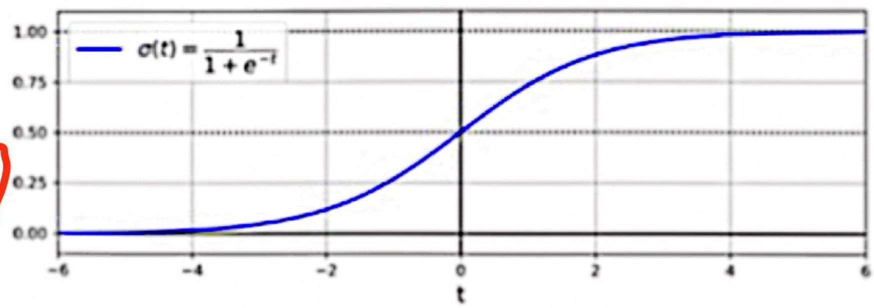
Logistic Regression

- Estimates the probability that an instance belongs to a particular class
  - **Positive Class: Probability greater than a given threshold**
- Instead of outputting the result directly like the linear regression model does, it outputs the *logistic* of this result

$$\boxed{\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\boldsymbol{\theta}^{\intercal}\mathbf{x}\right)}$$

$$\boxed{\sigma(t) = \frac{1}{1+\exp(-t)}}$$

$$\boxed{\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}}$$



t=hypotheses (h(x))

حولنا ال prediction لقيمة بين صفر
وواحد عشان نفهمه ك probability

القيم صارت بين صفر وواحد فال
RMSE اكيد بين الصفر وواحد
فبنستخدم ال log loss

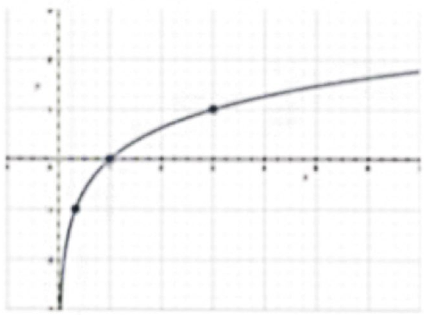
Logistic Regression-Training and Cost Function

- Log loss: Instances follow Gaussian distribution around the mean of their class
  - Log(**p**) is close to 0 when **p** is close to 1
  - Log(1-**p**) is close to 0 when **p** is close to 0

اذا ال probability قريبة من 1 وال
label (y) بساوي 1، ال cost قريبة
من صفر

$$\boxed{c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1-\hat{p}) & \text{if } y = 0 \end{cases}}$$

$$\boxed{J(\boldsymbol{\theta}) = -\tfrac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right)+\left(1-y^{(i)}\right)log\left(1-\hat{p}^{(i)}\right)\right]}$$



$$\boxed{\frac{\partial}{\partial\theta_j}\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{\theta}^{\intercal}\mathbf{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}}$$

label

$\log(\hat{p})$ $\quad y = 1$ $\quad , \hat{p} \approx 1$

$$y \log(\hat{p}) + \underbrace{(1-y)\log(1-\hat{p})}$$

هادا مخصص لما

لما يكون ال label صفر

$1 . \log(\hat{p})$

$1 . \log(1) = 1 . 0 = 0$

$1 . \log(0.2) \rightarrow$ قيمة عالية

زي كأنه ال Cost صار عالي

if $y = 0$ $\quad \hat{p} \approx 0$

$(1-y) \log(1-\hat{p})$

$(1-0)\log(1-\hat{p}) \rightarrow 1 . \log(1) = 0$ ← التوقع يطابق ال ground truth

$(1-0)\log(1-0.9) \rightarrow 1 \log(0.1)$ ← قيمة عالية

. كل ما كانت القيمة داخل log اصغر بتكون الجواب الاكبر.

. لهيك بنعمل minimization لل Cost فبنشتق ال Cost لكل قيمة من قيم θ

Iris Dataset

- A famous dataset that contains the sepal and petal length and width of **150 iris flowers** of three different species: **Setosa**, **Versicolor**, and **Virginica**.


```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> list(iris.keys())
['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
```

هي فعليا multiclass problem لانه
عنا 3 classes
بس احنا حليناها على انها binary
classification problem

Logistic Regression-Example (binary classifier)

- Predict_proba(): returns the probability of the instance
- Predict(): return the predicted class for the instance

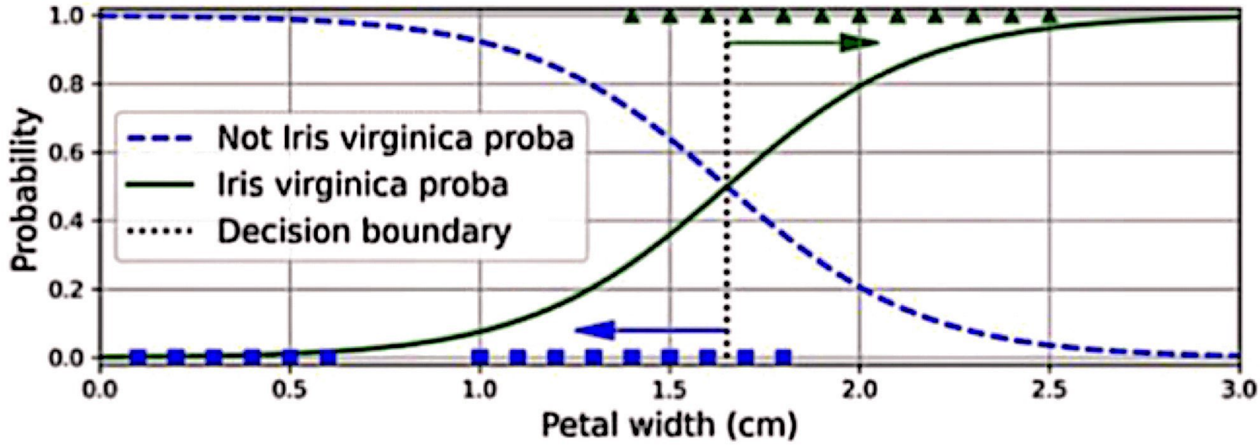
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X = iris.data[["petal width (cm)"]].values
y = iris.target_names[iris.target] == 'virginica'
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42)

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

Logistic Regression-Example (binary classifier)

لما ناخد ال petal width = 1.6
بتكون ال probability =0.5
يعني اذا اقل من 1.6 not virginica



Predict -> returns if its virginica or not
Predict_propa -> returns the probability

Logistic Regression-Multiclass

بال predict باخد ال max
probability وبرجعلي اياها لما يكون
عنا multiclass

- Softmax Regressor: Normalize the probability for each class.
- Cross entropy cost function

$$\boxed{J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)}$$

Cost function here is the categorical cross entropy
بنضرب كل label باحتمال ال class
تبعته

```
>>> softmax_reg.predict([[5, 2]])
array([2])
>>> softmax_reg.predict_proba([[5, 2]]).round(2)
array([[0. , 0.04, 0.96]])
```

<- soft max regressor
بعمل normalize لقيم ال
prediction بحيث يصير مجموعهم
بساوي 1

Classical Techniques

Prof. Gheith Abandah

Reference

- Chapter 5: **Support Vector Machines**
- Chapter 6: **Decision Trees**
- Chapter 7: **Ensemble Learning and Random Forests**

اجمع أكثر من Classifier مع بعض
وباخذ رأيهم

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

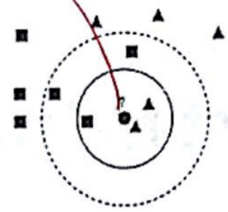
Outline

1. k-Nearest Neighbors
2. Support Vector Machines
3. Decision Trees
4. Ensemble Learning and Random Forests
5. Exercises

Classifiers

Class بدي اتوقع هاي ال instance لأي
بناء على اللي حواليها
مثلا بقيس المسافة بينها وبين اللي حواليها
1□ و 2△

k-Nearest Neighbors



- Find a predefined number of training samples (*k*) closest in distance to the new point and predict the label from them: regression or classification.
- The number of samples can be a user-defined constant (**k-nearest neighbor learning**), or vary based on the local density of points (**radius-based neighbor learning**).
- The distance can be any metric measure: standard **Euclidean distance** is the most common choice.
- Reference: https://scikit-learn.org/stable/modules/neighbors.html

ما عندي mathmodel انو يعمل predict، هون حسب المسافة دهي نفس reg و class
Classification ← بتطلع على ٣ اللي حولي وبعطي ال class الاقرب
regression ← ~ ~ ~ ~ ~ ~ ال avg
← او مثلا شو ال instance اللي بتبعد قطر ١ او قطر ٢ ← بعمل radius based neighbour learning
• الهدف نوصل بسرعة لاقرب k-elements فبنستخدم BallTree / KDTree اللي بسرعوا عملية ال indexing

Nearest Neighbors Classification

#of neighbors by default = 5

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5,
                    weights='uniform', … )
```

#of neighbors — اقرب 5
يفضل تكون odd

don't forget to import

- **weights** can be: **uniform**: All points in each neighborhood are weighted equally, and **distance**: Weight points by the inverse of their distance.

لما كلما زاد البعد بتقل ال decision تبعه

- Example:

```
from sklearn.neighbors import KNeighborsClassifier
knn_clf = KNeighborsClassifier() → object (default values)
knn_clf.fit(X_train, y_train)
```

على فرض X_3 بعيدة
$x_1 + x_2 + 0.8\, x_3$
أقل weight

* check the documentation of k-nearest neighbors.

← بدل ما احسب المسافة بين ال nodes - في عندي indexing
وكل نقطة اذا بدي استخدمها مع large dataset بدي احسب باي data واحسب المسافة بينهم
وهذا كله بيستهلك وبياخذ memory

Nearest Neighbors Regression

regression

```
class sklearn.neighbors.KNeighborsRegressor(n_neighbors=5,
                    weights='uniform', … )
```

- The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors.
- Example:

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=3)
model.fit(X, y)
```

Outline

1. k-Nearest Neighbors
2. Support Vector Machines
3. Decision Trees
4. Ensemble Learning and Random Forests
5. Exercises

complexity of SVM → $O(m^2)$

* اذا جربت عليه data كبيرة بكون بطيء.

عشان يفصل ال classes عن بعض

linear → برسم خط مستقيم

non-linear → مش خط مستقيم

Support Vector Machine (SVM)

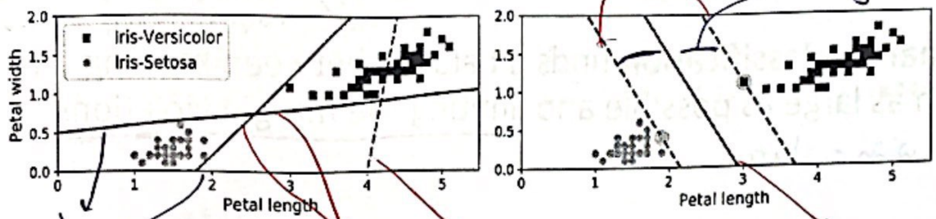
binary

multi

- Very **powerful** and **versatile** Machine Learning model, capable of performing **linear** or **nonlinear classification**, **regression**, and outlier detection.
- Well suited for classification of **complex** but **small- or medium-sized** datasets.
- SVM gives **large margin classification.**

سيء

margins (width ال ال)



ليس هون المشكلة في نقاط قريبة من ال line ما في مسافة بين ال line وال samples

ليس هو ممتاز classifier

ممتاز

سيء لانه قاطع نقاط بنفس ال class

ممتاز لانه فاصل بين ال classes

المهم يكون ال decision boundary عريض

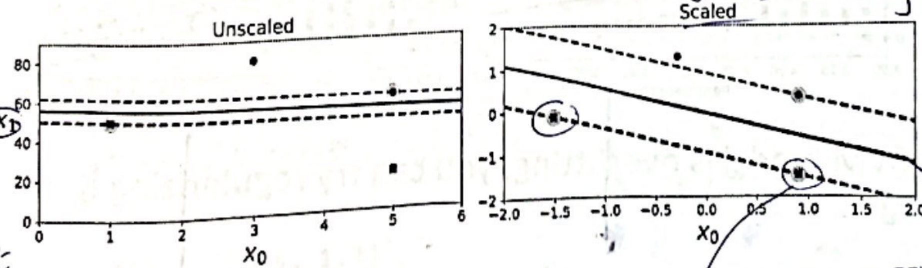
Linear SVM Classification

```
from sklearn.svm import LinearSVC
svc_linear = LinearSVC()
svc_linear.fit(x_train, y_train)
y_pred = svc_linear.predict(x_test)
```

* why to do feature scale?

- The **decision boundary** is fully determined by the instances located on the edge. These instances are called the **support vectors.**
- SVMs are **sensitive** to the **feature scales.**

لما اكون بقرر أو بمشي على اساس مسافات او بعتمد على distance لازم اعمل scaling (الأفضل)



x_1 الوزن تبع x_1 رح يكون تأثيره على المسافة اكبر

رح يقلل من تأثير x_0 في المعادلة.

Hard margin classification

ممنوع يكون عندي instances داخل ال margin

ليس ال surface اللي بفصل

ال margin هو المسافة بين الخط الفاصل واقرب نقاط عليه

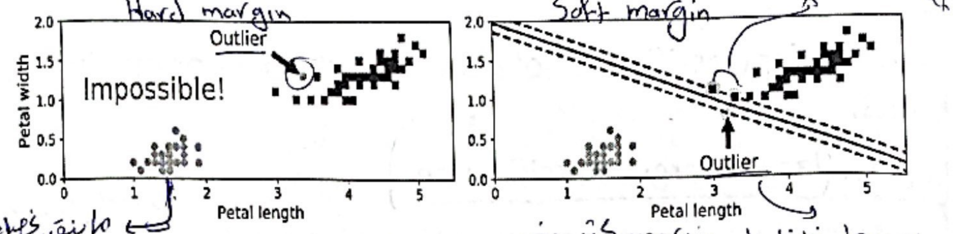
↳ (support vector) -

Soft Margin Classification

↱ All instances must be off the street and on the correct side

- **Hard margin classification** cannot handle linearly inseparable classes and is sensitive to outliers.

Hard margin — Soft margin — Support vector



decision boundary ما بنقدر نحطها

بسببها خلينا ال margin كثير صغير

- **Soft margin classification** finds a balance between keeping the margin as large as possible and limiting the margin violations.

↳ OK with outliers.

instances ليسمح لبعض ال
يكونوا على wrong side

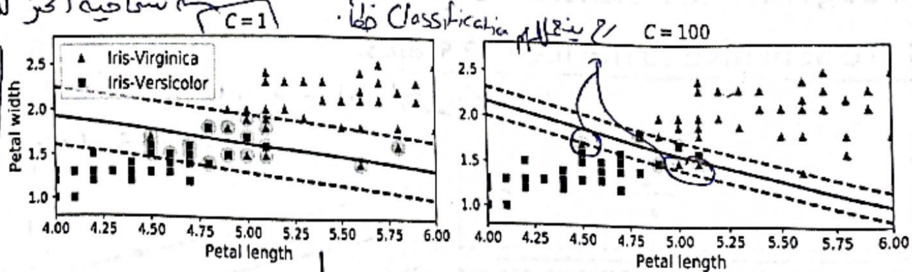
Soft Margin Classification (by default = 1) soft حكم كم هو

بأثر على ال accuracy

- You can control the number of violations using the C hyperparameter

سماحية اكثر لل violations (C=1)

Classification خطأ // يتعلم ال



اي شي داخل الخطوط المقطعة يعتبر Violation.

- If your SVM model is **overfitting**, you can try **regularizing** it by **reducing C**.

this model accuracy is less than this model (because more violations).

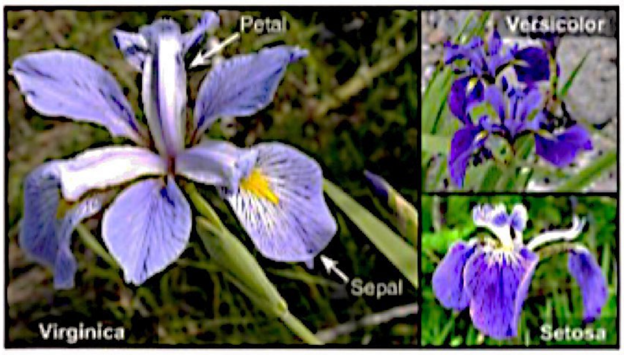
C↑ Violations↓ ↓margins

C = ∞ → hard margin

Scanned with CamScanner

Iris Dataset

- A famous dataset that contains the sepal and petal length and width of **150 iris flowers** of three different species: **Setosa**, **Versicolor**, and **Virginica**.




```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> list(iris.keys())
['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
```

If we want to use SVM then we
must scale the data

SVM Classification Example

```
from sklearn.datasets import load_iris
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = load_iris(as_frame=True)
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = (iris.target == 2)  # Iris virginica

svm_clf = make_pipeline(StandardScaler(),
                        LinearSVC(C=1, random_state=42))
svm_clf.fit(X, y)
>>> X_new = [[5.5, 1.7], [5.0, 1.5]]
>>> svm_clf.predict(X_new)
array([ True, False])
>>> svm_clf.decision_function(X_new)
array([ 0.66163411, -0.22036063])
```

columns اسماء ال

We have 3 labels, (0,1,2)

C=1 -> Soft

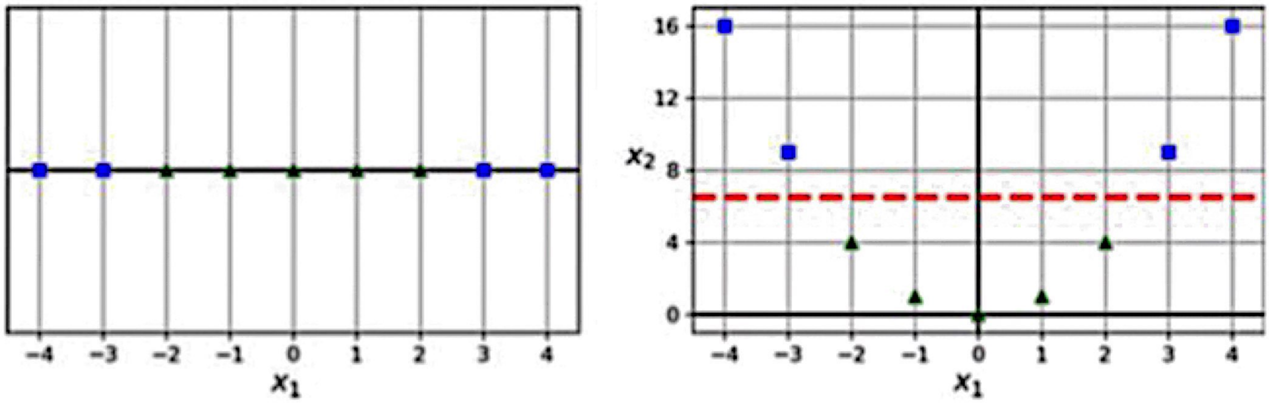
```
poly_kernel_svm_clf.fit(X, y)
```

Nonlinear SVM Classification

مرات الداتا ما بتكون linearly
separable يعني ما بنقدر نعمل خط
مستقيم يفصل الداتا عن بعض

- Some dataset are not linearly separable
- Adding more features such as polynomial features can making the function linearly separable

كانت بس X1 فعملناها X1^2



او ممكن نستعمل classes تانية غير
ال linear SVC وجوبها بكون curve

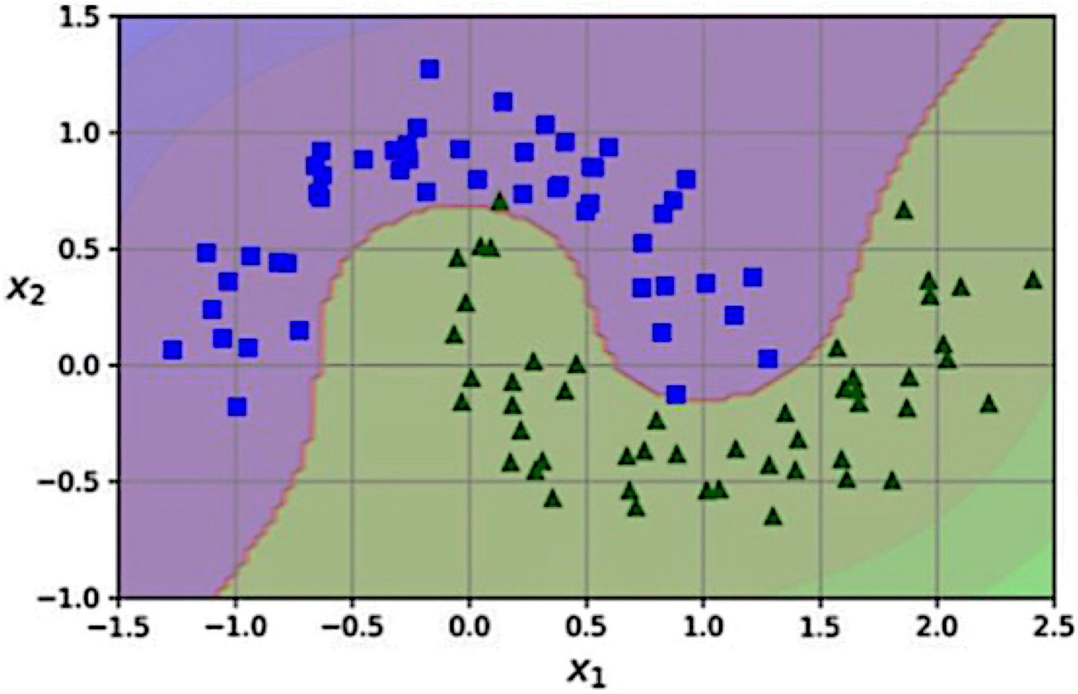
Nonlinear SVM Classification

fro
fro

X,

pol

)
pol



state=42)

state=42)

Nonlinear SVM Classification

كيف نحسب X1^2 ؟ باستخدام ال
polynomialFeatures و بنحدد ال
degree واستخدمنا linearSVC لانه
بدنا خط مستقيم

```
from sklearn.datasets import make_moons
from sklearn.preprocessing import PolynomialFeatures

X, y = make_moons(n_samples=100, noise=0.15, random_state=42)

polynomial_svm_clf = make_pipeline(
    PolynomialFeatures(degree=3),
    StandardScaler(),
    LinearSVC(C=10, max_iter=10_000, random_state=42)
)
polynomial_svm_clf.fit(X, y)
```

From sklearn.SVM import SVC
SVC_nonLinear = SVC()
SVC_nonlinear.fit(X-train, Y-train)
Y-pred = SVC_nonlinear.predict(X-test)
واضح ....

لازم اعمل Scaling

Nonlinear SVM Classification

استخدم هاي

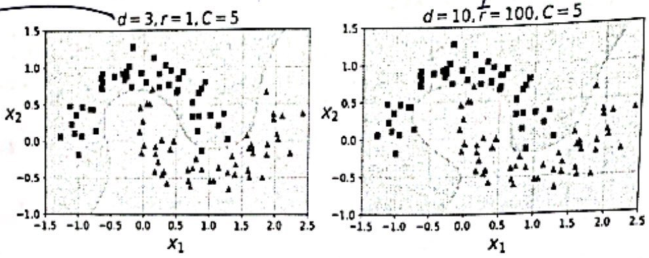
- The SVM class supports nonlinear classification using the kernel option

Controls how much the model is influenced by high-degree polynomials versus low-degree

```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

degree

يفضل بالعادة اغير parameter واحد بالمرة الواحدة.



14

زيادة ال degree بخلي ال curve يمشي مع ال data بشكل احسن.
بس الزيادة الكبيرة ممكن تعمل overfitting

Gaussian Radial Basis Function → RBF function

X بضيف features جديدة.

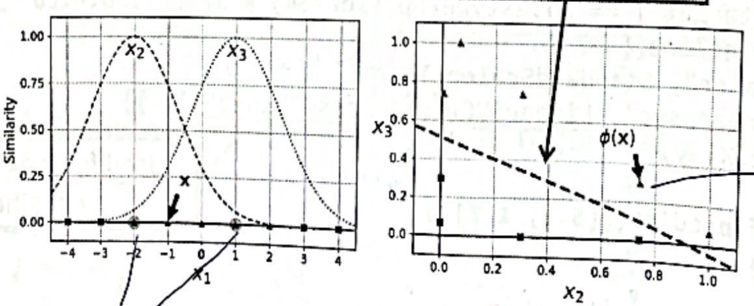
landmarks.

هنا

$$\phi_\gamma(x, \ell) = \exp\left(-\gamma \| x - \ell \|^2\right)$$

- The Gaussian RBF can be used to find **similarity features** (x_2 and x_3) of the one-dimensional dataset with two **landmarks** to it at $x_1 = -2$ and $x_1 = 1$

حولنا ال data



بحولها المسافة بينها وبين النقطة الأولى والثانية صارت هون.

اوجد reference point
فصار كل نقطة الها 2 features الي هم بعد النقطة عن النقطة الأولى والثانية

15

Gaussian RBF Kernel

- Is **popular** with SVM to **solve nonlinear problems.**

```
rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ])
rbf_kernel_svm_clf.fit(X, y)
```

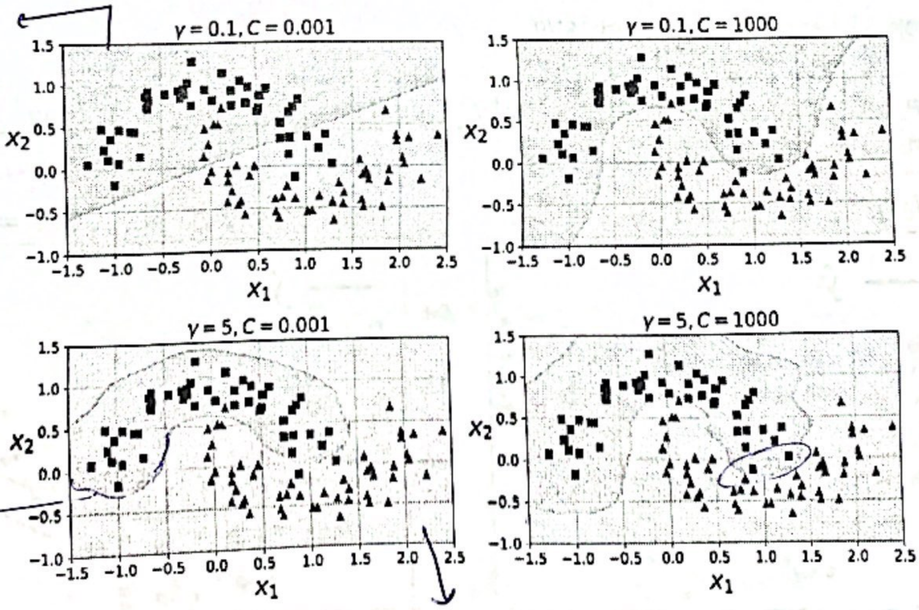
- **Transforms** a training set with *m* instances and *n* features to *m* instances and *m* features.
- **gamma** and **C** are used for **regularization** with smaller values.

↳ like learning rate.

. remember :- C ↑ violation ↓

Gaussian RBF Kernel

violations كدا ال كبيرة



~~desi~~ desicion boundaries.

non-liner kernel

وكدا ال violation قليل

* الاقي الخط أو العلاقة اللي بتمثل اغلب النقاط (line بوصف العلاقة بين X & Y)

Linear SVM Regression → output (cont. values).

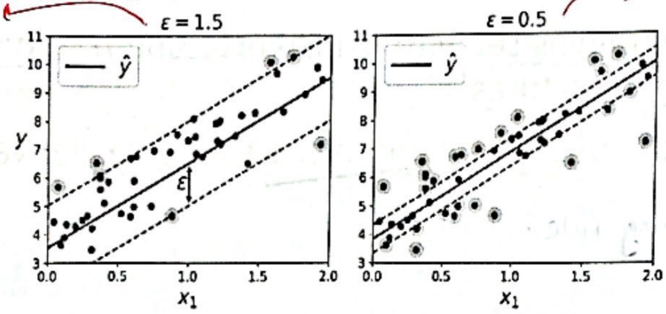
→ can be used as classifier & as regressor.

- Fits as many instances as possible on the margin while limiting margin violations. The width of the street is controlled by a hyperparameter ϵ.

```
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5)
svm_reg.fit(X, y)
```

margins كبيرة



margins قليلة

" many violations"

هي النقاط اللي خارج

SVC → classifier
SVR → regressor

margins ال ← عكس ال classification

* بدنا نوجد ال margin اللي بتشمل معظم النقاط.

Nonlinear SVM Regression

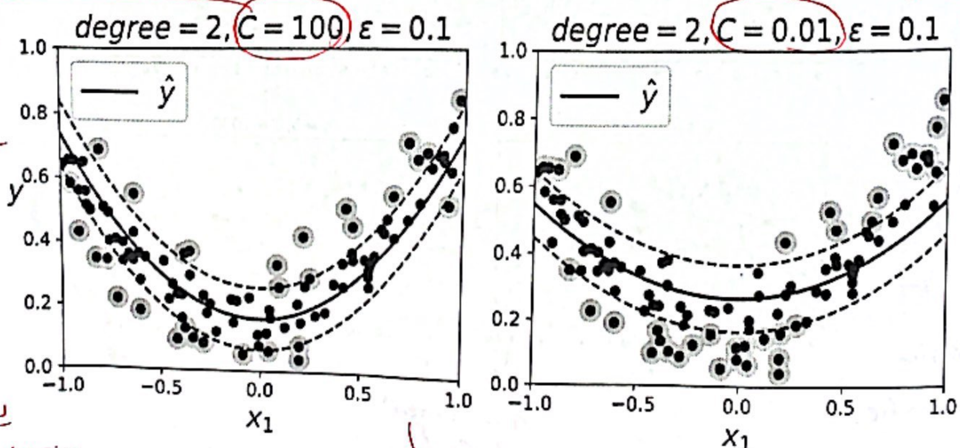
```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_poly_reg.fit(X, y)
```

less violations

more violations

Quadratic



* اذا زدت ال degree بيصير الخط يمشي مع النقاط اكثر وكل ما ازيد ال degree تزيد احتمالية حدوث overfitting

better results

SVM Conclusion

Can be multi classifier

- The `LinearSVC` has complexity of $O(m \times n)$.

#of samples

- The **SVC** time complexity is usually between $O(m^2 \times n)$ and $O(m^3 \times n)$.

#of samples

- This algorithm is perfect for complex but small or medium training sets. However, it scales well with the number of features.

SVC →

بالحكي عاطمة ب
المعطيات =

↑m ↑complexity

Outline

1. k-Nearest Neighbors
2. Support Vector Machines
3. Decision Trees
4. Ensemble Learning and Random Forests
5. Exercises

Decision Trees

classification
regression

- Decision Trees are **versatile** Machine Learning algorithms that can perform both **classification** and **regression** tasks, and even multioutput tasks.
- They are very powerful algorithms, capable of fitting complex datasets.

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
iris = load_iris()  → iris dataset
X = iris.data[:, 2:] # petal length and width
y = iris.target  → labels
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
y_pred = tree_clf.predict (X_test)
```

we have predict_proba

no. of samples class 1   class 2

$$gini = 1-\left(\frac{50}{150}\right)^2-\left(\frac{50}{150}\right)^2-\left(\frac{50}{150}\right)^2$$

total no. of samples.

لـ بيكون اقرب للصفر كل ما يكونوا ال elements كلهم في class واحد

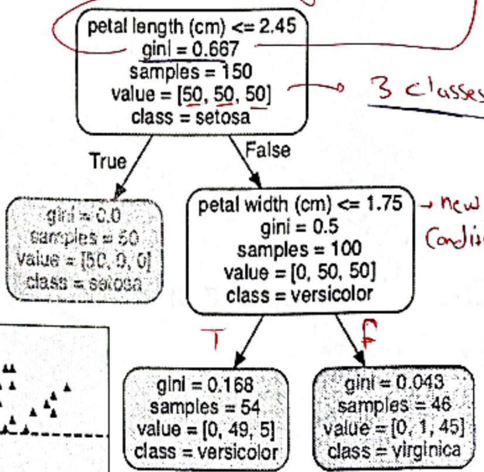
22

Visualizing a Decision Tree

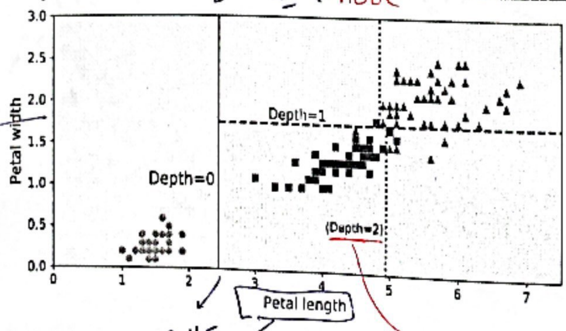
كلما كانت أقل أحسن
impurity measure

```
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=image_path("iris_tree.dot"),
        feature_names=iris.feature_names[2:],
        class_names=iris.target_names,
        rounded=True,
        filled=True
    )
```



3 classes
new condition
50 فيها leaf node
T F



1.75
2.45
للبس يفصل ال classes عن بعض فوقف هون

بيوقف عند قيمة ال Petal width الها اقل impurity

max depth = 2

عشان هيك وقف هون

23

• Decesion tree

* لازم ال max depth يكون صغير عشان اتجنب حدوث ال overfitting ,

Regularization Hyperparameters

عدد ال splits

↑# of features ↑max depth

• Increase min_* or decrease max_*: max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_leaf_nodes=None, max_features=None,

default ال

int or floating point

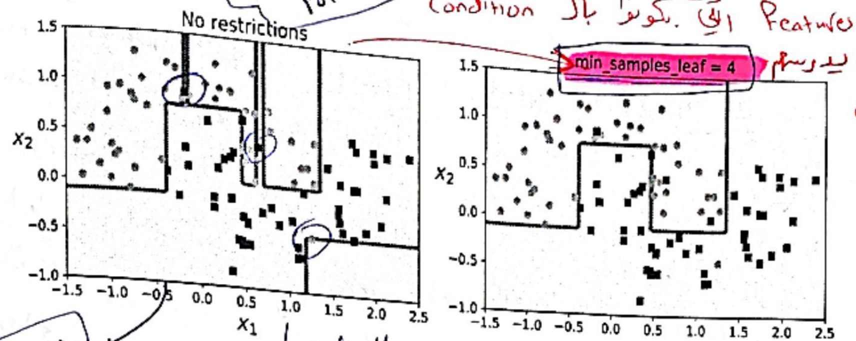
بشمل كل ال Features الموجودة

عدد ال Features الي بكونوا بال Condition (الي بدو يدرسهم ويا خذ قرار على اساسهم)

اذا كانت floating point بتنضرب بعدد ال features .

min number of samples in each node

split تبعها



default behavior

عشان ما يعمل splits مالها داعي .

overfitting في لانه عمل call على ال outliers

24

بالنهاية بعطيني رقم مش class و probability

Decision Trees Regression

① predict بعدها

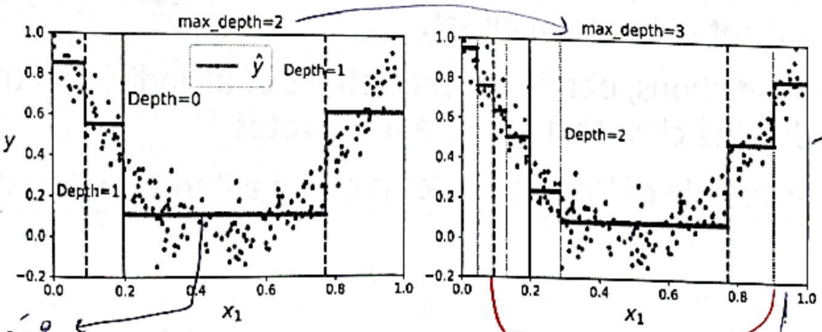
```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X, y)
```

tree_reg_mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt( )

rmse

linear بكون بال range الواحد



قيمة واحدة "بعطيني اياها" ثابتة لل "samples"

كلما صغرنا ال depth بكون أحسن

new boundaries

صار smooth اكثر وبشبه العلاقة التربيعية بال SVM .

صغرنا الفترات عشان يكون الحل احسن . عن طريق زيادة ال max depth فزاد عدد ال splits

25

Outline

1. k-Nearest Neighbors
2. Support Vector Machines
3. Decision Trees
4. Ensemble Learning and Random Forests
5. Exercises

classifier contains decesion trees.

كل tree بدربها على subset من الداتا ، لاني لو دربتهم كلهم على كل الداتا او نفس الداتا بتطوني نفس الجواب .

أدرب أكثر من model مع بعض وأخذ برأيهم كلهم مجتمعة .

مثلاً استعمل اكثر من classifier وناخذ برأيهم مثلا ناخذ ال majority

ممكن كل واحد منهم بتدربه على الداتا كلها وكل واحد منهم اله طريقته

احد الامثلة على Ensemble learning

بندربها على عدد هائل من ال Decision trees مع بعض .

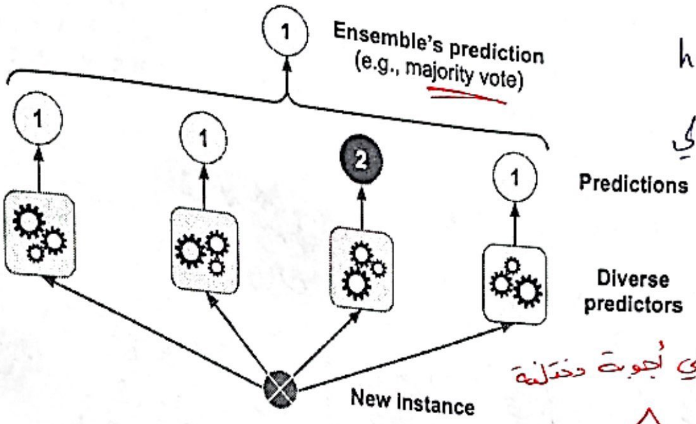
Ensemble Learning and Random Forests

- A group of predictors is called an **ensemble.**
- You can train a group of Decision Tree classifiers, each on a different **random subset** of the training set.
- To make predictions, obtain the predictions of all individual trees, then predict the class that gets the **most votes.**
- Such an ensemble of Decision Trees is called a **Random Forest.**

Voting Classifiers

مثل: اذا في عنا weak classifiers وطبقهم مع البعض بعطوني strong

- If each classifier is a **weak learner** (meaning it does only slightly better than random guessing), the ensemble can be a **strong learner** (achieving high accuracy).



hard voting → بختار ال class الي pred. العالي

اعطوني أجوبة مختلفة

28

example

Data → C1 [0.51 , 0.49]; C2 [0.2 , 0.8]; C3 [0.51, 0.49]

Hard voting: 0, 1, 0 → 0

Soft voting:

$P(0) = \frac{0.51+0.2+0.51}{3} \approx 0.4$

$P(1) = \frac{0.49+0.49+0.8}{3} \approx 0.6$

P(1) is higher → 1

Scikit-Learn Voting Classifier 1/2

Hard Voting → لما يكون عدد ال classifiers كبير.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
voting_clf.fit(X_train, y_train)
```

فكرة الدرس كلام على نفس ال data كلهم مختلفين classifiers

voting='soft' predict the class with the highest class probability

بحسب avg احتمال 0 واحتمال 1 لكل قيم 0 و 1

القيمة الاكثر تكرار. اذا binary يعني كل classifier ح يعطيني 1 أو 0 بشوف شو اكثر شي طلع.

29

Scikit-Learn Voting Classifier 2/2

Soft Voting → بنجمع ال Prob. وباخذ ال max وبناءً على ويه بقرر.

تكملة

```
>>> from sklearn.metrics import accuracy_score
>>> for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
...     clf.fit(X_train, y_train)
...     y_pred = clf.predict(X_test)
...     print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
...
LogisticRegression 0.864
RandomForestClassifier 0.896
SVC 0.888
VotingClassifier 0.904
```

لوكان hard voting بنختار class 1

ex:- $C_1 \begin{array}{c}0\\1\\2\end{array}\begin{bmatrix}0.3\\.2\\\boxed{.8}\end{bmatrix} + C_2 \begin{array}{c}0\\1\\2\end{array}\begin{bmatrix}.8\\\boxed{.5}\\.2\end{bmatrix} + C_3 \begin{array}{c}0\\1\\2\end{array}\begin{bmatrix}.2\\\boxed{.6}\\.2\end{bmatrix} = \begin{bmatrix}0.8\\1.12\\\boxed{1.2}\end{bmatrix}$

لوكان soft voting بنختار هاد class 2

Sample يعني ممكن تتكرر ال

With replacement → Samples can be given more than once to the same Classifier

كيف نختار الداتا لل classifiers

Bagging and Pasting

اما لما يكونوا classifiers مختلفين بقدر اعطيهم كلهم نفس ال data

ممكن استخدمها لما تكون ال data تبعتي صغيرة

- Use the **same training algorithm** for every predictor, but train them on different random subsets of the training set.
- When sampling is performed **with replacement**, this method is called **bagging** (short for **bootstrap aggregating**).
- When sampling is performed **without** replacement, it is called **pasting**.
- The aggregation function is the most frequent prediction (**hard voting**) for classification, or the **average** for regression.

bagging → with replacement (يعني مثلاً لوعنا 10,000 row و randomly اول classifier اختار 2 rows، التاني لما نختار بيكونوا من نفس الداتا.

Pasting → العكس.

Bagging and Pasting

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=100, bootstrap=True, n_jobs=-1)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

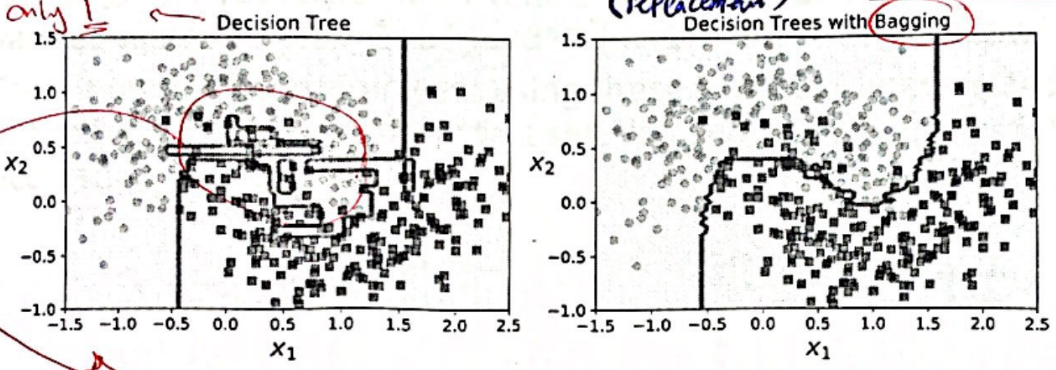
→ no. of decision tree classifiers

bagging (replacement)

with replacement and use all available cores

-1

only 1



→ more general

overfitting

Random Forests

هون لازم نستخدم decesion tree فقط

الأفضل بالواجب نستخدم voting او bagging.

- An ensemble of Decision Trees trained via the bagging with `max_samples` set to the size of the training set, and choosing the best random splits.

```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)

y_pred_rf = rnd_clf.predict(X_test)
```

هدول نفس الأشي

- Equivalent to:

→ split randomly

```
bag_clf = BaggingClassifier(
    DecisionTreeClassifier(splitter="random", max_leaf_nodes=16),
    n_estimators=500, max_samples=1.0, bootstrap=True, n_jobs=-1)
```

Outline

1. k-Nearest Neighbors
2. Support Vector Machines
3. Decision Trees
4. Ensemble Learning and Random Forests
5. Exercises

Exercises

1. Train an **SVM classifier** on the **MNIST** dataset. Since SVM classifiers are binary classifiers, you will need to use one-versus-all to classify all 10 digits. You may want to tune the hyperparameters using small validation sets to speed up the process. What accuracy can you reach?

Exercises

2. Train and fine-tune a **Decision Tree** for the **moons dataset.**
   a) Generate a moons dataset using `make_moons(n_samples=10000, noise=0.4)`.
   b) Split it into a training set and a test set using `train_test_split()`.
   c) Use grid search with cross-validation (with the help of the `GridSearchCV` class) to find good hyperparameter values for a `DecisionTreeClassifier`. Hint: try various values for `max_leaf_nodes`.
   d) Train it on the full training set using these hyperparameters, and measure your model's performance on the test set. You should get roughly 85% to 87% accuracy.

Exercises

3. Load the **MNIST** data and split it into a training set, a validation set, and a test set (e.g., use 50,000 instances for training, 10,000 for validation, and 10,000 for testing). Then train various classifiers, such as a **Random Forest classifier**, an **Extra-Trees** classifier, and an **SVM**. Next, try to combine them into an **ensemble** that outperforms them all on the validation set, using a **soft** or **hard** voting classifier. Once you have found one, try it on the test set. How much better does it perform compared to the individual classifiers?

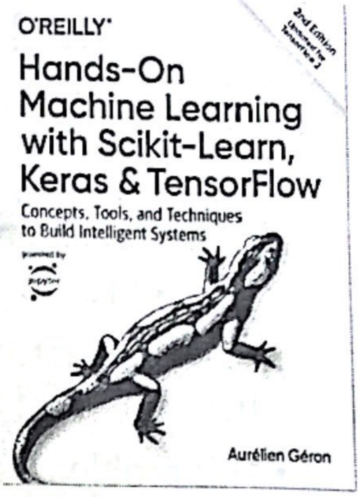
Summary

1. k-Nearest Neighbors
2. Support Vector Machines
3. Decision Trees
4. Ensemble Learning and Random Forests
5. Exercises

Unsupervised Learning and Clustering

Prof. Gheith Abandah

Reference



- Chapter 8: **Dimensionality Reduction**
- Chapter 9: **Unsupervised Learning Techniques**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

* لما يكون الـ features ما في بينها variance ← ما بتتغير وما الها تأثير على الكل ← حكما اشيلها
(اخلي الـ factor تبعها قليل).
← بقلل الـ execution time.

Outline

ما بنقلل عدد الـ samples

- Dimensionality Reduction
  - Projection and Manifold
  - Principal Component Analysis (PCA) → used when data has alot of features
- Unsupervised Learning
  - Clustering
    - K-Means
    - DBSCAN
  - Gaussian Mixtures and Anomaly Detection
  - Exercises

بستخدم PCA عشان اقلل عدد الـ features
واحاول احافظ على الـ variance
بهيك يصير loss صغير

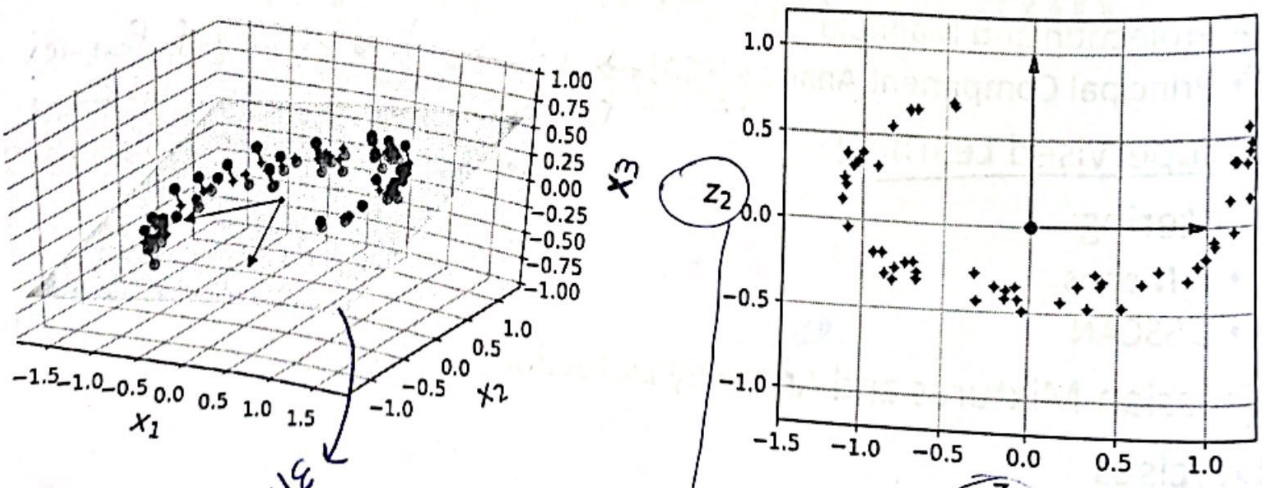
Dimensionality Reduction

- Many Machine Learning problems involve **thousands** or even **millions of features** for each training instance.
- All these features make training extremely **slow** and make it much **harder** to find a good solution.
- This problem is often referred to as the **curse of dimensionality.**
- **Dimensionality reduction approaches**
  1. **Drop** not useful features
  2. **Merge** correlated features
  3. **Projection** and manifold
  4. **Transform** features

مثلاً بال MNIst لو لخينا حواف الصور بنقل عدد ال features بالتالي بنقال 784 على افتراض انه كل الصور الرقم فيها بالنص.

Projection and Manifold

x فعليا كل feature هي عبارة عن dimension



Projection line

2 new features

بوصفوا ال plane السكني

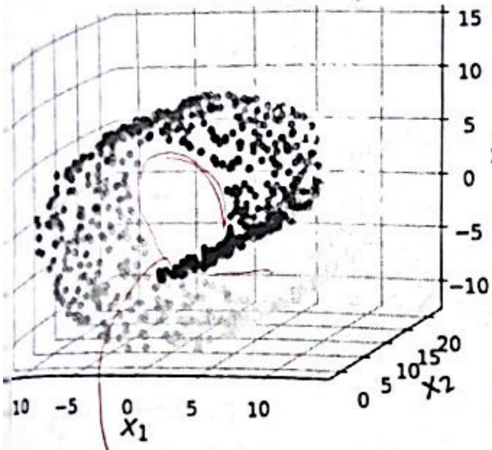
فقللنا عدد ال features من 3 ل 2

بدي أخد ال data

Projection and Manifold

← وأقلل ال features.

Swissroll data

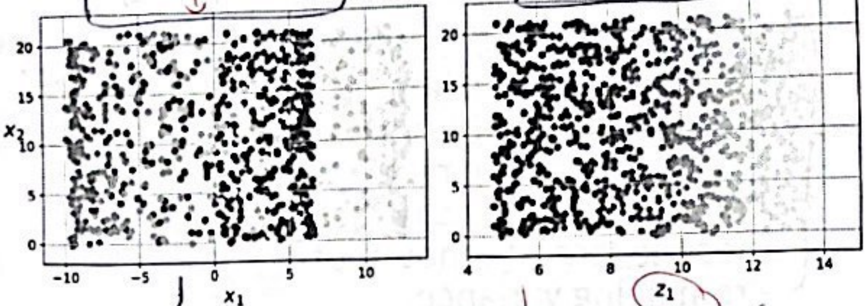


كأنه بعد axis-z الشكل

- Simply projecting onto a plane may not give better solution.
- Projecting to a proper manifold is better.

في تداخل !!

Manifold



صارت الداتا أعقد لما عملنا projection

Surface

new feature

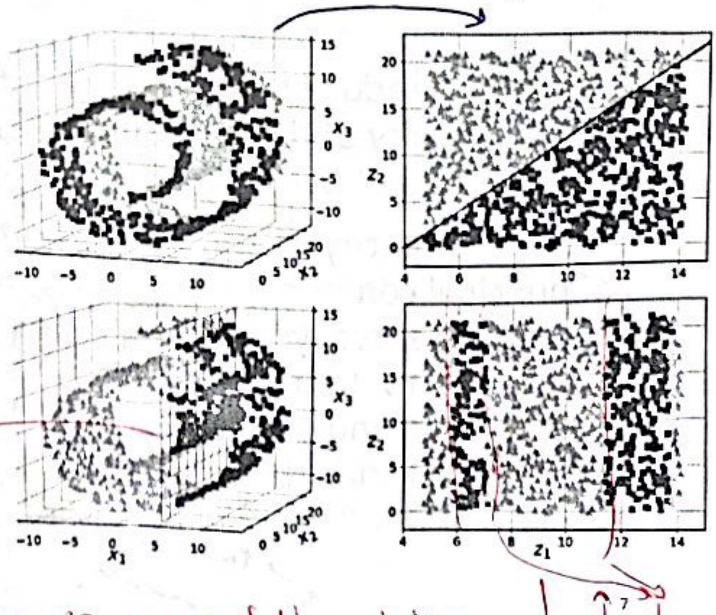
6

Projection and Manifold

- The decision boundary may not always be simpler with lower dimensions.

Projection

decision boundary / تقسيم



هون ال manifold مش كثير مفيد لل classification .

← بس ممكن نزبط على ال decision tree with depth 3

decision boundaries

7

بمعنى ال K surface الجديد اللي بدي اعمل عليه projection يكون بحافظ على ال

variance لل data ← عشان الأكبر new features

لازم يحافظوا على ال Variance

one of dimensionality reduction techniques.

Principal Component Analysis (PCA)

• وحدة من الطرق اللي نعمل فيها Projection

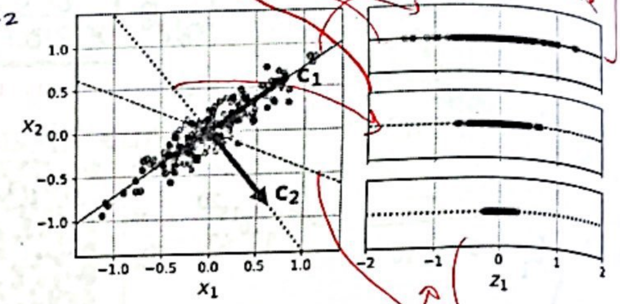
- Is the most **popular** dimensionality reduction algorithm.
- First it identifies the **hyperplane** that lies **closest to** the **data**, and then it **projects** the data onto it.
- PCA identifies the **axis** that accounts for the **largest amount of variance** in the training set. Then it finds the **next orthogonal axes** that accounts for the largest amount of remaining variance.

C_1 + C_2 متعامدات

c_1, c_2

new features along this surface

projection on this line



8

يعني هاد الخط حتى كثير بعبر عن ال variation في الداتا

وهاد في داتي الكبيرة new feature

هاينا: بحافظ على ال feature اي الها

ال variance عالي (التغير بال feature عالي).

اول شي طلعت C_1 بعدين C_2

كل next principal component بكون عمودي على اللي قبله ← C_2 عمودي على C_1

Principal Component Analysis (PCA)

هي اصلا ٣ بدنا نقللهم ل

نعمل عليهم Projection

- Use PCA to reduce the dimensionality of the dataset down to **two dimensions.**
- Instead of specifying the number of principal components you want to preserve, you can set `n_components` to be a float between `0.0` and `1.0`, indicating the ratio of variance you wish to preserve.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)
```

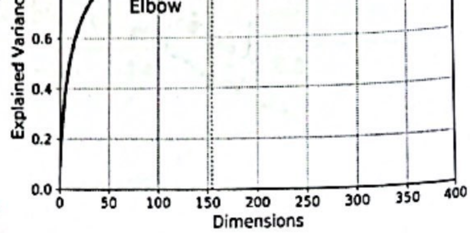
3-D

الداتا الجديدة يكون فيها 95% من ال Variance

```
pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X_train)
```

عدد ال components

MNIST



9

بنزيد features ما بنكسب variance

بدل X_1 & X_2 & X_3

يعني داتا جديدة Z_1 & Z_2

Outline

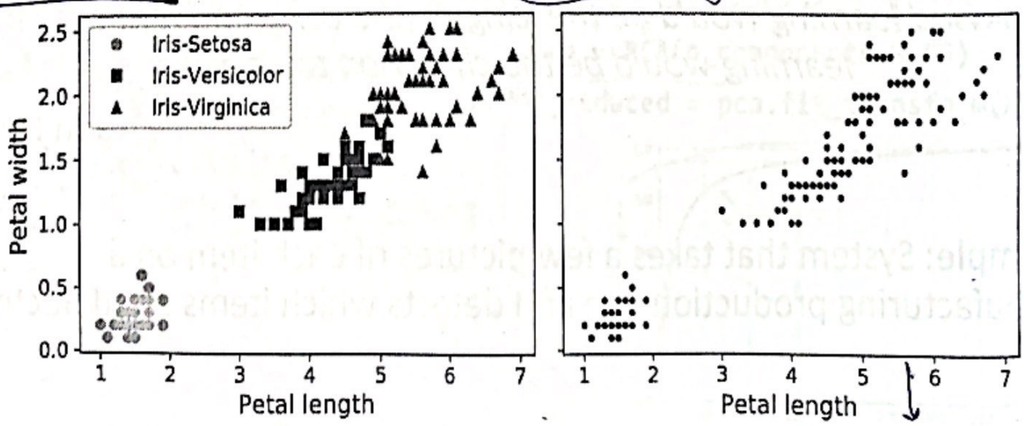
- Dimensionality Reduction
  - Projection and Manifold
  - Principal Component Analysis (PCA)
- Unsupervised Learning
- Clustering
  - K-Means
  - DBSCAN

باخدو الداتا الunlabeled
وبرجعوا ال cluster center لكل cluster
مع label لكل cluster

- Gaussian Mixtures and Anomaly Detection
- Exercises

Clustering

- The task of identifying similar instances and assigning them to clusters, i.e., groups of similar instances.
- Classification (left) versus clustering (right)



ما في اي طريقة تميز الداتا
لازم نطبق algorithms

Clustering Applications

شوبينج مثلا حسب ال shopping behavior

- **Customer segmentation**: useful for recommender systems.
- **Data analysis**: discover clusters of similar instances as it is often easier to analyze clusters separately.
- **Dimensionality reduction**: find affinity features to the found clusters
- **Anomaly detection**: any instance that has a low affinity to all the clusters is likely to be an anomaly.
- **Semi-supervised learning**: perform clustering and propagate the labels to all the instances in the same cluster.
- **Search engines** for images نحدد كل مجموعة من ال Pixels شو تمثل / كل Pixels ب cluster مع بعض
- **Image segmentation**

اذا كانت القيمة عاليه مش كويس يعني المسافة كبيرة

كل نقطة في الداتا

Score → avg distance between data & Centre point "MSE"

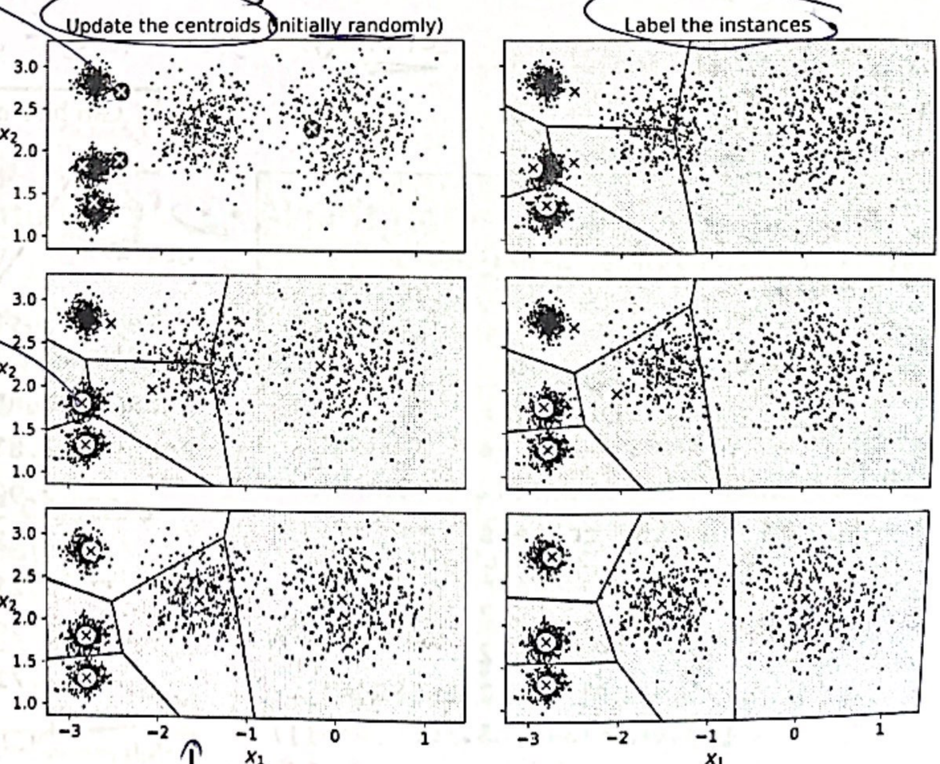
الداتا تتبع اقرب Centre اليها.

أرقام اذا ال Center point ب لا تلزم منتج و تستخدم موقعها

K-Means

التحديد المسافة بين ال sample وال center

- **Quick** and **efficient** algorithm
- **Scale** before clustering
- **Need to specify the number of clusters**



Centre point of cluster

the best

لانه بنتعامل وبنقيس مسافة

نجرب أرقام

كل نقطة رح تاخذ label وهو رقم ال Cluster الي هي فيه

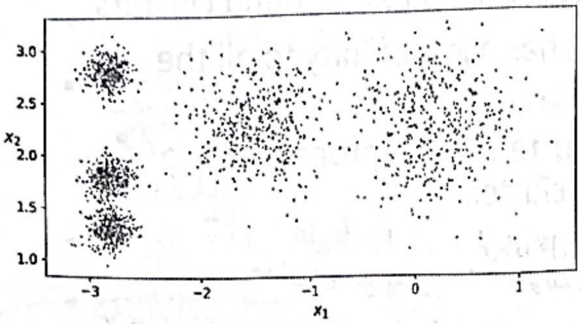
score = interia

K-Means

برجع array معطى فيه رقم ال cluster لكل نقطة في ال data

(لكل instance x في أي cluster موجودة) تنتمي له

- Cluster to 5 clusters

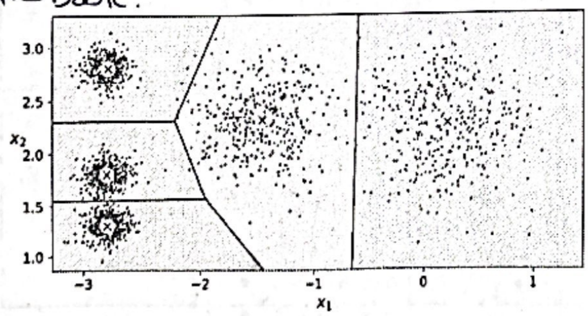


```
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
y_pred
array([4, 0, 1, ..., 2, 1, 0],
      dtype=int32)

# Hard clustering:   cluster 1     cluster 2
X_new = np.array([[0, 2], [-3, 3]])
kmeans.predict(X_new)
array([1, 2], dtype=int32)
```


Ch.9 in the book.

K-Means



```
kmeans.cluster_centers_
array([[-2.80389616, 1.80117999],
       [ 0.20876306, 2.25551336],
       [-2.79290307, 2.79641063],
       [-1.46679593, 2.28585348],
       [-2.80037642, 1.30082566]])
```

Can be a dimensionality reduction technique.

5 centers

المسافة بين كل sample و ال 5 cluster center

```
# Soft clustering, a score per
#   cluster:
kmeans.transform(X_new)
array([[2.81093633, 0.32995317,
        2.9042344 , 1.49439034,
        2.88633901],
       [1.21475352, 3.29399768,
        0.29040966, 1.69136631,
        1.71086031]])
```

cluster 1

هاي نقطة تابعة ل cluster 2

باخذ اقل قيمة

K-Means- Centroid Initialization

5 cluster centers "user defined"
لما يبدا عملية ال clustring يبدا
منهم

- User Defined Initial values

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

بعمل التجربة مرة وحدة، لما تكون ٢
مثلا بعمل التجربة مرتين ب
initialization مختلف

- Random Initialization
  - Randomly initialize centroids
  - Repeat experiment **n_init** times
  - Select the model with lower **inertia (**Minimum mean diastance between the instances and the centroids)

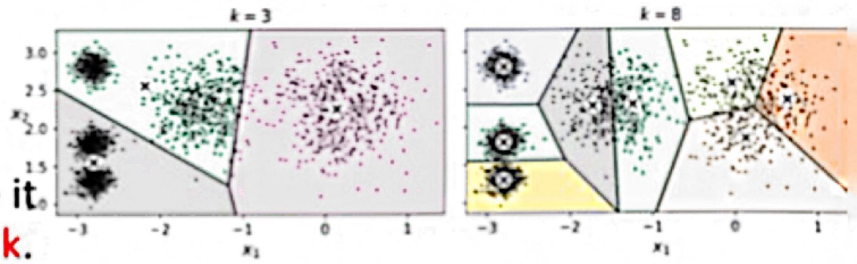
بحكي كيف شغلي وكم المسافة الي
وصلنالها عشان نقدر نقارن
(score) المسافة بين كل نقطة وال
cluster center الي ربطتها فيه

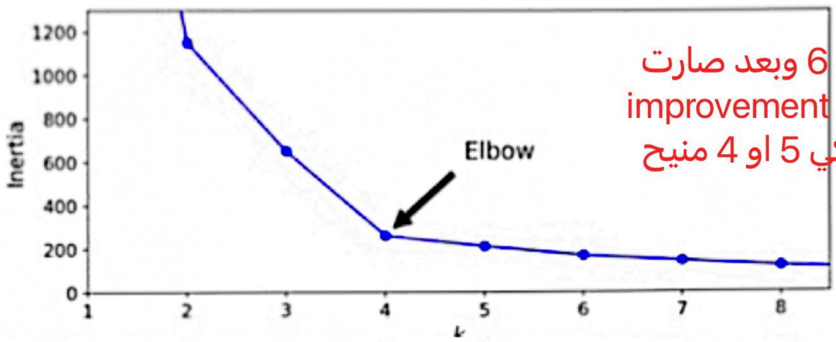
```
>>> kmeans.inertia_
211.59853725816856
```

كل ما كانت قيمته اقل بكون افضل

K-Means

- It is important to specify the **right** number of clusters ***k***.
- Inertia is not a good performance measure because it is getting lower as we increase **k**.





من 4 ل 5 ومن 5 ل 6 وبعد صارت
التعديلات قليلة وال improvement
قل كتير فممكن نحكي 5 او 4 منيح

امتى ممكن يطلع معنا 1 ؟ لما تكون a
قليلة جدا مقارنة بال b او تساوي
صفر
يعني بنكون بالمكان الصح يعني كل ال
samples بنفس ال cluster قريبة
وباقي ال clusters بعيدة مسافة

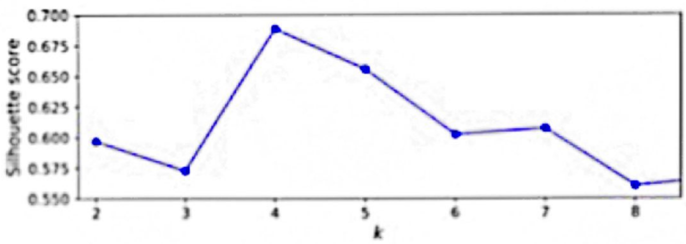
K-Means

- Find *k* that gives highest mean silhouette coefficient.

$$\text{Silhoutte coefficient} = \frac{b-a}{\max(a,b)}$$

  - a: the mean distance to the instances in the same cluster
  - b: the mean distance to the instances in the next closest clutser
  - The score is between -1 and 1

```
from sklearn.metrics import
                    silhouette_score
silhouette_score(X, kmeans.labels_)
0.655517642572828
```



K-Means

The best

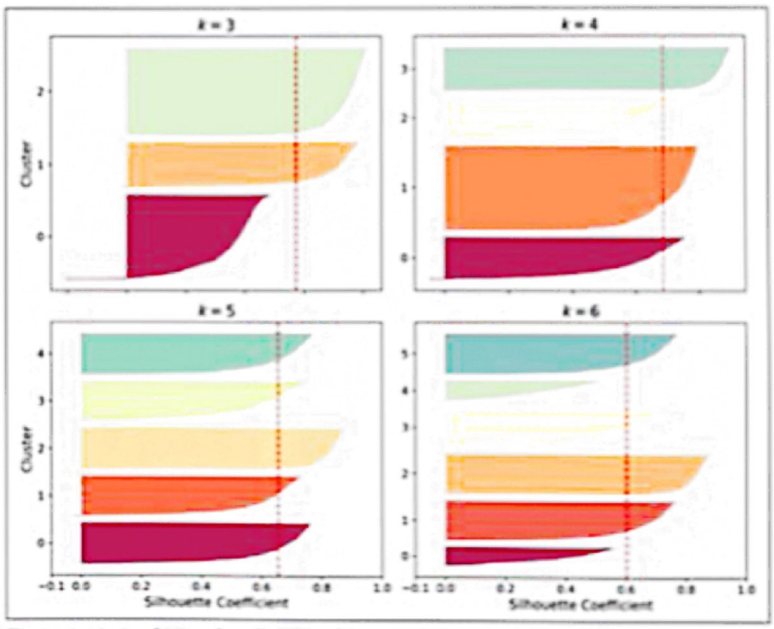
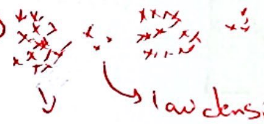


Figure 9-10. Analyzing the silhouette diagrams for various values of k

DBSCAN

ماجدله عدد clusters) حسا كل مايشوف تجمع نقاط بعتبره cluster
هو لحاله بعمل clustering للداتا ←

- Defines **clusters** as **continuous** regions of high density.
- Works well if all the **clusters are dense enough**, and they are well **separated by low-density regions.**
- Behaves well when the clusters have ***varying sizes*** or ***non-spherical*** shapes.



Can detect anomalies

• النقاط الي المسافة بينهم اقل او تساوي ε بكونوا بنفس ال cluster

- **Algorithm**

  •DBSCAN is faster than K-means

  - For each instance, counts how many instances are located within a small distance **ε-neighborhood**.

    • بنعطيه قيمة ε

  - If an instance has at least **min_samples** instances in its ε-neighborhood, then it is considered a **core instance.**
  - All instances in the neighborhood of a core instance belong to the same cluster. This may include other core instances; therefore, a long sequence of neighboring core instances forms a single cluster.
  - Any instance that is not a core instance and does not have one in its neighborhood is considered an **anomaly** (-1). → cluster number

    ← اذا ما لقيت حولين النقطه اي core قريب منها
    او ولا نقطة المسافة بينه وبينها اقل من ε.

يعني يكون عندي نقطه وحولها نقاط
المسافة بينها وبينهم ε أو اقل min-samples
← بعتبرها core instance.

• كل النقاط البعيدة يكون الها behavior واحد
ويكون الها (1- = cluster)
↳ cluster number

DBSCAN

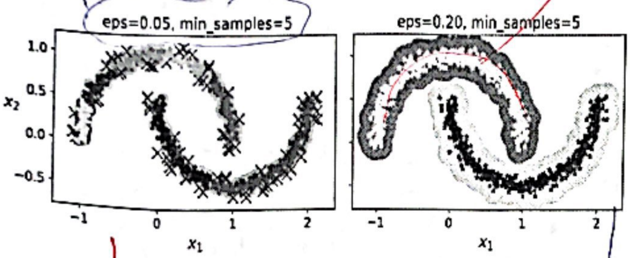
بيساعدون اعرّف core ال

Continuous

• Cluster the **moons** dataset

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import
                                    make_moons
X, y = make_moons(n_samples=1000,
                  noise=0.05)
dbscan = DBSCAN(eps=0.2,
                min_samples=5)
dbscan.fit(X)
```

↳ ε



7 clusters

↳ 2 clusters

ماكو هنا
anomaly اي

red x → anomalys

كل هاي إشكله هنا طريق
اني أكثر ε

DBSCAN

→ have fit-predict()

• DBSCAN class does not have a `predict()` method.

• Can use other classifiers.

* Check DBSCAN doc.

→ supervised learning algorithm

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(dbscan.components_, dbscan.labels_[dbscan.core_sample_indices_])
X_new = np.array([[-0.5, 0], [0, 0.5], [1, -0.1], [2, 1]])
knn.predict(X_new)
array([1, 0, 1, 0])
```

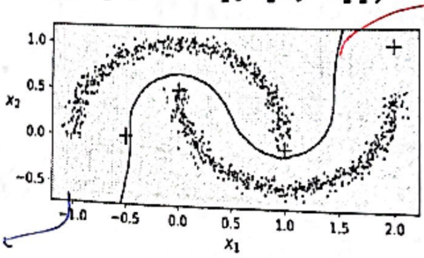
core samples

labels

يعني بتاخذ

رقم ال
cluster

non linear decision boundary.



2 clusters

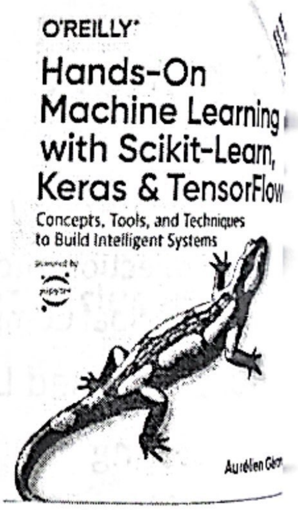
بتحاول تقلد mimic
للجهاز العصبي بجسم
الانسان

Neural Networks

ميزتها انها بتعمل
features extraction

Prof. Gheith Abandah

Reference



- Chapter 10: **Introduction to Artificial Neural Networks with Keras**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Introduction

- YouTube Video: *But what *is* a Neural Network?* from 3Blue1Brown

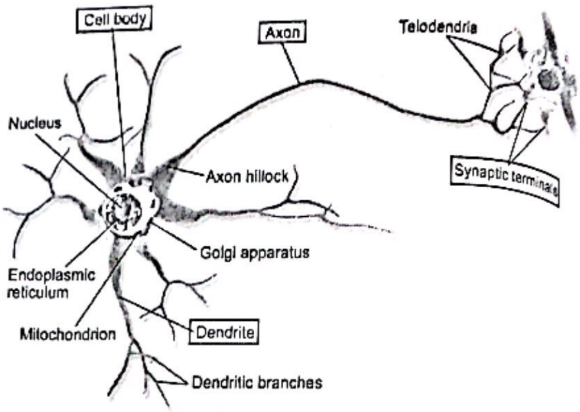
https://youtu.be/aircAruvnKk

Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

1. Introduction

- **Artificial neural networks** (ANNs) are inspired by the brain's architecture.
- First suggested in 1943. Is now **flourishing** due to the availability of:
  - Data
  - Computing power
  - Better algorithms



2. The Perceptron

ممكن نغير ال activation function او ال weights وال feature

- The **Perceptron** is a simple ANN, invented in 1957 and can perform linear binary classification or regression.

activation function

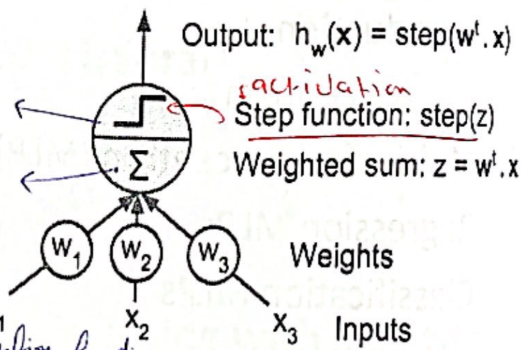
$sum = x_1 w_1 + x_2 w_2 + x_3 w_3$

if $sum < 0 \rightarrow 0$

$sum \geq 0 \rightarrow 1$

Threshold from activation function

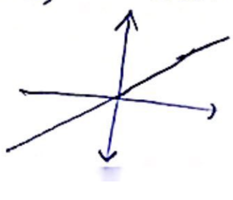
Linear threshold unit (LTU)



- Common **step function**:

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

another activation function

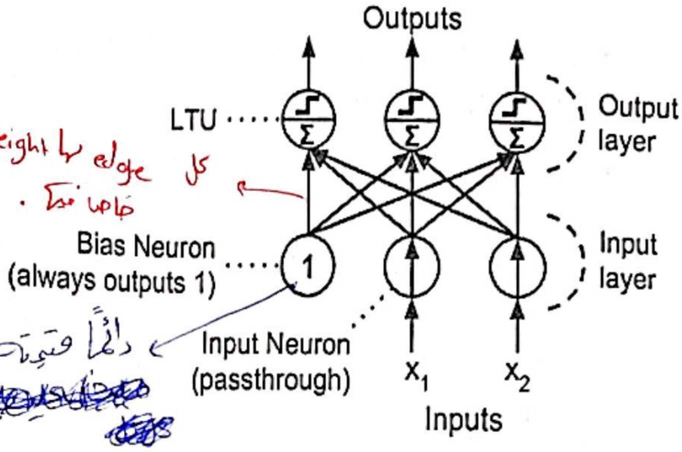


2. The Perceptron

• ممكن يكون عنا اكثر من neuron

- The Perceptron has an **input layer** with **bias** and **output layer**.
- With **multiple output nodes**, it can perform multiclass classification.
- Hebbian learning **"Cells that fire together, wire together."**

كل edge لها weight خاص فيه .



دائماً نتيجته 1

$$\boxed{w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i}$$

→ زي فكرة ال learning rate .

• ممكن نعدل على ال weights لكل neuron عشان نحصل على نتيجة مفيدة ،

كيف نعدل على ال weights ؟ باستخدام gradient descent

• يعني ممكن ال weight ل x_2 ب neuron1 تختلف عن neuron 2 .

• كل neuron بدرس ال combination لمجموعة ال features وبعطي weight او بلغي weight .

2. The Perceptron

- Scikit-Learn provides a **Perceptron** class.

• ما رح نستخدم ال sklearn بال neural network ممكن نستخدمها بال data preprocessing وال data split .

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron

iris = load_iris()
X = iris.data[:, (2, 3)]  # petal length, petal width
y = (iris.target == 0).astype(np.int)  # Iris Setosa?
per_clf = Perceptron(random_state=42)
per_clf.fit(X, y)

y_pred = per_clf.predict([[2, 0.5]])
```

* بيشتغل اكتر من Mathematical model مع بعض .

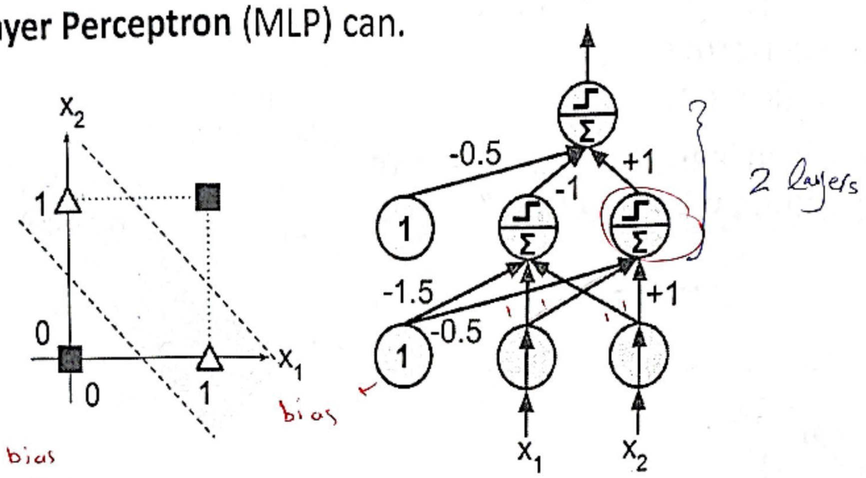
. هدفنا نعرف مشوال weights الي نستعملها
عشان يطلع acarate answer
. حل ما هتطول network deeper
بتحسن العلاقات = ادق .

2. The Perceptron

- The perceptron **cannot solve non-linear problems** such as the XOR problem.
- The **Multi-Layer Perceptron** (MLP) can.

x_1	x_2	F
0	0	0
0	1	1
1	0	1
1	1	0

Try for all

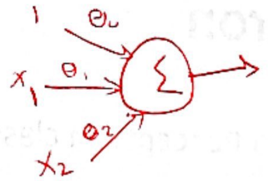


bias

x_2 w x_1 w
$0*1 + 0*1 + 1*-0.5 = -0.5 \rightarrow$ step function = 0

Cont.

$F = \theta_0 + \theta_1 x_1 + \theta_2 x_2$



Outline

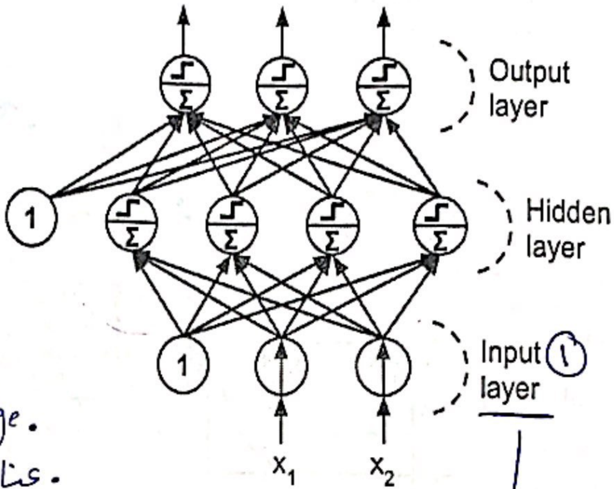
1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

Scanned with CamScanner

3. Multi-Layer Perceptron (MLP)

*needs Computational power

- An MLP is composed of a (pass-through) **input layer**, one or more layers of LTUs, called **hidden layers**, and a final layer of LTUs called the **output layer**.
- When an ANN has **two or more** hidden layers, it is called a **deep neural network** (DNN).



output ال hidden ال من 15 edge.

هنا input layer واحدة و output layer واحدة.

كما نزيد عدد ال hidden layers بتصير deep network.

كل feature بتدخل على كل neuron.

يعني هنا 12 edges 12 weights

ما بتكون neuron هي عبارة عن features

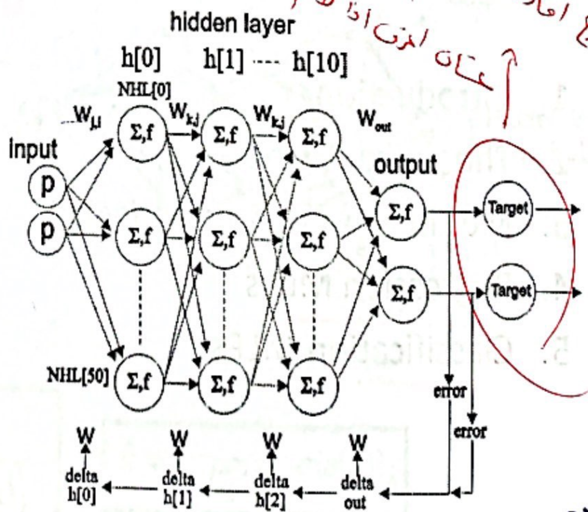
عدد ال weights او ال edges هو نفسه عدد ال Parameters

11

3. Multi-Layer Perceptron (MLP)

Weights لازم اعدل ال ... عشان اقربها من ال Ground truth لما اقارنها بال

- Trained using the **backpropagation training algorithm.**
  - For each training instance the algorithm first makes a prediction (**forward pass**), measures the error,
  - then goes through each layer in reverse to measure the error contribution from each connection (**reverse pass**),
  - and finally slightly tweaks the connection weights to reduce the error (**Gradient Descent step**).



loss

- forward pass "forward propagation" → from input to output. (to find pred.)

بعدها حسب قيمة ال Prediction برجع نعدل ال weights.

المسؤول عن تعديل ال weights ← backpropagation.

backpropagation ↘

بنشتق ال cost funct. بالنسبة لكل weight وبنشوف كيف كل weight تأثير
و بنحدد اذا نزيده او نقلله وهيك بنعرف قيم ال w الجديدة

*3. Multi-Layer Perceptron (MLP)

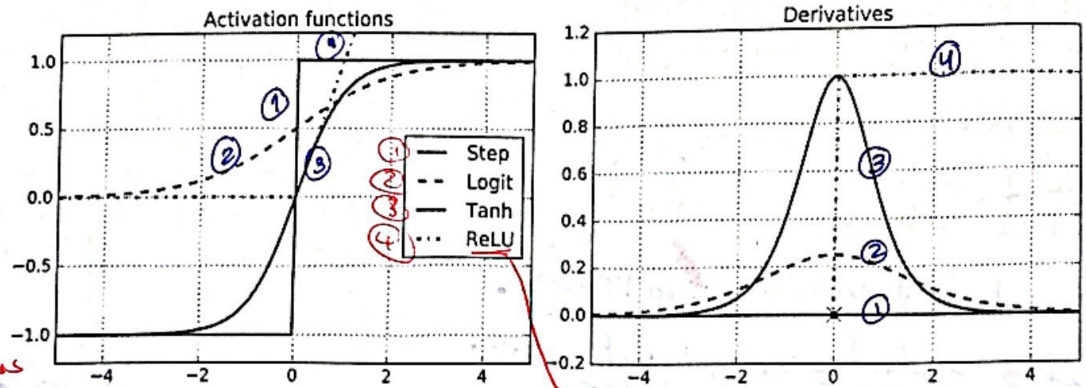
- **Common activation functions: logistic, hyperbolic tangent, and rectified linear unit.** 4 step function

فهو بنستعمله

غير قابل للاشتقاق عند ال Zero

$$\sigma(z) = 1 / (1 + \exp(-z))$$
$$tanh\ (z) = 2\sigma(2z) - 1$$
$$\text{ReLU}\ (z) = \max\ (0, z)$$

بيكون ال activation function يكون قابل للاشتقاق عشان بحتاج عملية الاشتقاق لما اكونا بدي اعرف شو لازم اعدّل على ال edges weight.



13

is the best activation function.

بناءً على قيمة المشتقة → بعرف قيمة ال step الي لازم اخدها. > 1 0

Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

• "Backpropagation"

$$\boxed{\sigma = \frac{1}{1+e^{-x}}}$$

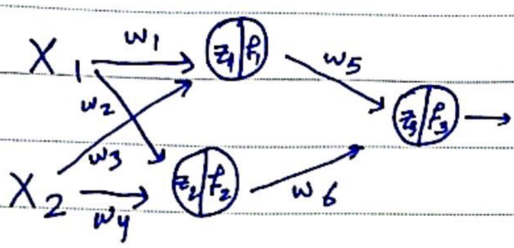
input layer → 2 inputs (features)

1 hidden layer

one output layer

$z_1 / z_2 / z_3$ → summation

$f_1 / f_2 / f_3$ → activation functions



$f = \sigma(z)$

$z_1 = X_1 w_1 + X_2 w_3$

$f_1 = \sigma(z_1)$

$z_2 = X_1 w_2 + X_2 w_4$

$f_2 = \sigma(z_2)$

$z_3 = f_1 w_5 + f_2 w_6$

$f_3 = \sigma(z_3)$

cost funct. → MSE

• assume "Regression" and our target is 500, and the output is 700
↳ prediction

• error = 200

• if binary classifier → cost function = $y\log(\hat{y}) + 1-y\log(1-\hat{y})$

learning rate ← → كيف ال cost بتغير

• $w_5 \text{ new} = w_5 \text{ old} - \lambda \left(\frac{\partial l}{d w_5}\right)$. بتغير w_5

• remember : chain rule → $\frac{dy}{dz} = \frac{dy}{dx} \cdot \frac{dx}{dz}$ • هاي العملية عبارة عن matrix multiplication

so → $\frac{dl}{dw_5} = \frac{dl}{f_3} \cdot \frac{df_3}{dz_3} \cdot \frac{dz_3}{dw_5}$ where $f_3 = \hat{y}$

↳ بنشتق ال cost بالنسبة $\hat{y}$ وبعوض من f_3 $= f_3(1-f_3)$ ↳ $= f_1$

4. Regression MLPs

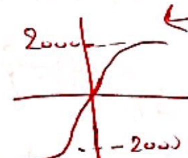
- Typical MLP architecture for **regression**: لازم نحدد activation function لكل neuron

Hyperparameter	Typical Value
# input neurons	One per input feature (e.g., 28 x 28 = 784 for MNIST) (input بعطي feature لكل)
# hidden layers	Depends on the problem. Typically 1 to 5.
# neurons per hidden layer	Depends on the problem. Typically 10 to 100.
# output neurons	1 per prediction dimension → في حالة ال regression بكون عندي 1 neuron / في حالة ال binary ممكن نحط 1 بس / ما بلزمه activation function
Hidden activation	ReLU (or SELU, see Chapter 11)
Output activation	None or ReLU/Softplus (if positive outputs) or Logistic/Tanh (if bounded outputs)
Loss function	MSE or MAE/Huber (if outliers)

If regression

ε|x| → no activation → بتطلعلي القيم الاصلية سواء موجبة او سالبة
or you can use ReLU.

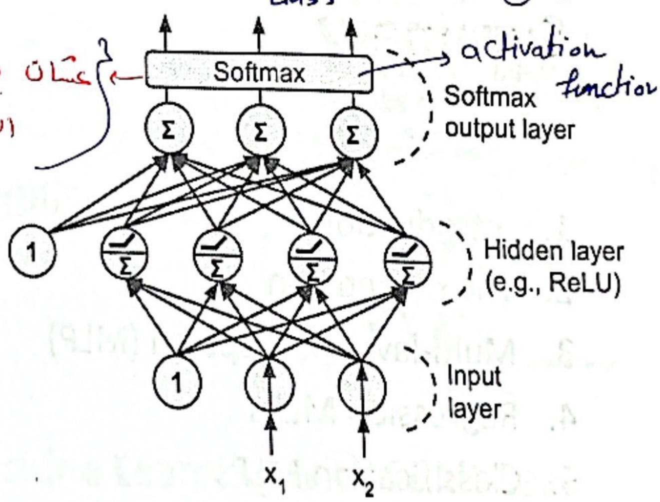
→ Mean Absolute Error



ال output ال multi class ← لكل class neurons

5. Classification MLPs

تحتاج الى Normalization لل probabilities التي طلعت بحيث يكون كلها مجموعها = 1



- For **classification**, the output layer uses the **softmax function.**
- The output of each neuron corresponds to the estimated **probability** of the corresponding class.

$$\boxed{\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}}$$

$$\boxed{\hat{y} = \underset{k}{\operatorname{argmax}}\ \sigma(\mathbf{s}(\mathbf{x}))_k}$$

ال MNIST فيه 10 classes يعني 10 neurons لل output

وال activation function لل multi class هو ال softmax

5. Classification MLPs

- Typical MLP architecture for **classification:**

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross-Entropy	Cross-Entropy	Cross-Entropy

كل label بيعطي جواب معين .

y: Actual

$\hat{y}$: predicted

$$\sum y \log \hat{y} + (1-y) \log(1-\hat{y})$$

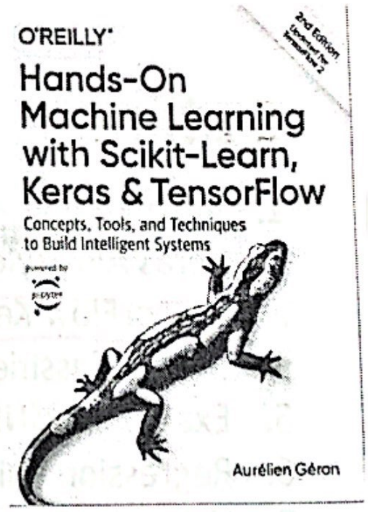
Summary

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

Artificial Neural Networks with Keras

Prof. Gheith Abandah

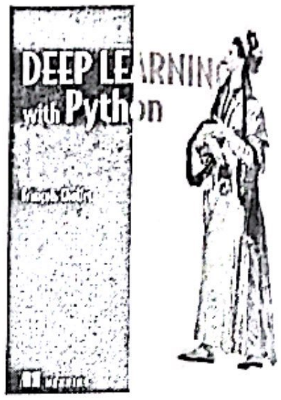
Reference



- Chapter 10: **Introduction to Artificial Neural Networks with Keras**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Reference



- **Deep Learning with Python**, by François Chollet, Manning Pub. 2018
- **Introduction to Keras** by Francois Chollet, March 9th, 2018 (slides)

Outline

* Keras is higher level than TensorFlow

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

Introduction

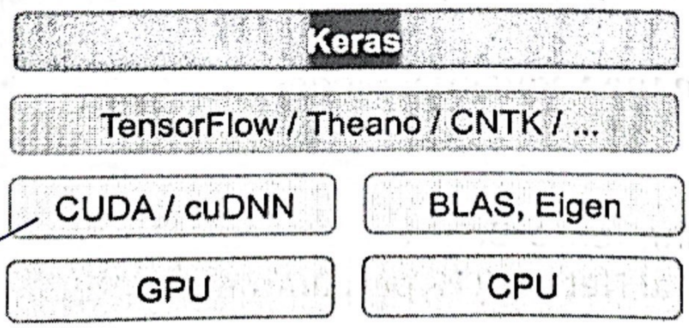
- YouTube Video: *Keras Explained* from Siraj Raval

https://youtu.be/j_pJmXJwMLA

1. Introduction

- Keras is a high-level API to build and train deep learning models.

by default ال Keras بتستخدم ال TensorFlow.



→ Front-end

→ back-end

لانه ال deep neural كال CPU بطيئة فبنحتاج acceleration لهيك صار عال GPU.

1. Introduction – Advantages

- **User friendly**: Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
- **Modular and composable**: Keras models are made by connecting configurable building blocks together, with few restrictions.
- **Easy to extend**: Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models.

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

*In Nueral Network, you don't need to extract Features

كل layer بتاخذ من
ال layer اي قبلها

2. Keras API Styles

كيف نكتب ال model ؟

1. **The Sequential Model**
   - Dead simple
   - Only for single-input, single-output, sequential layer stacks
   - Good for 70+% of use cases
2. **The functional API** → ال layer ممكن تاخذ من اي layer ثانية
   - Like playing with Lego bricks
   - Multi-input, multi-output, arbitrary static graph topologies
   - Good for 95% of use cases
3. **Model subclassing** → more advanced.
   - Maximum flexibility
   - Larger potential error surface

Flexibility ↑, Error ↑

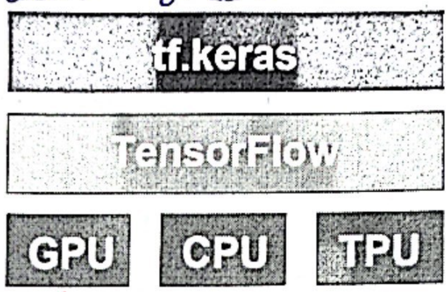
Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

3. TensorFlow Keras

- Keras is the official high-level API of TensorFlow
- tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.4
- Full Keras API
- With useful extra features such as **tf.data**

طلع ال Keras من
ضمن ال tensorflow



For acceleration

بنستخدمها عشان
نعمل ال
data
preparation

11

3. TensorFlow Keras

بكون من ضمنه
ال Keras

- To install TensorFlow

```
$ pip install --upgrade tensorflow
```

- To import Keras from TensorFlow

```
>>> import tensorflow as tf
>>> from tensorflow.keras import Layers
>>> from tensorflow import keras
>>> tf.__version__
'2.1.0'
>>> keras.__version__
'2.2.4-tf'
```

عشان نفهم
الجواب

- Dense
- Activations
- Dropout
- Conv1D, 2D, 3D
- Polling
- RNN, LSTM, GRU
- ...

12

Outline

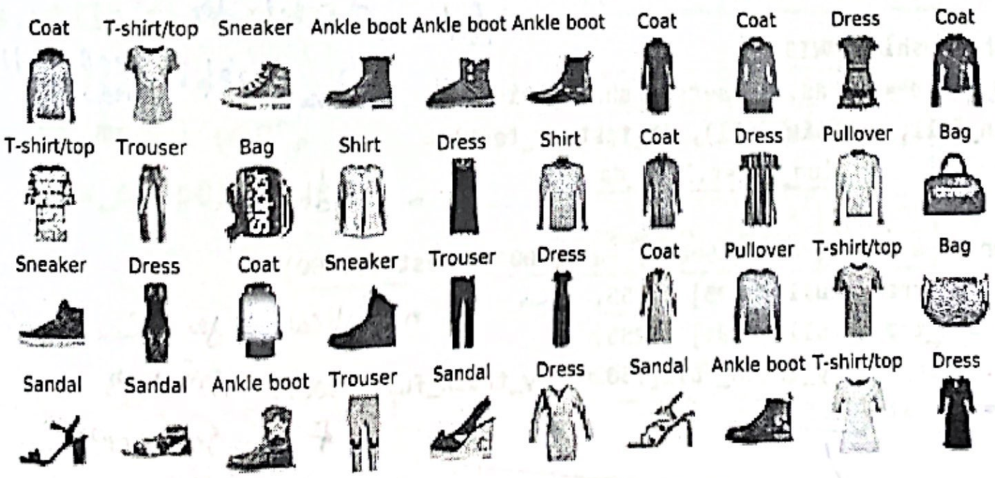
بدنا نبني Neural network تعرف نوع الكلاس بالصورة

4. Image Classifier Using the Sequential Model

→ 10 classes بتتكون من

→ input

- **Fashion MNIST** is similar to MNIST (70,000 grayscale images of 28×28 pixels each, with 10 classes).



4. Fashion MNIST

بناءاً عليه
يكون عنا فكرة
بشكل عم neuron hidden layer
و عم عددهم
وعم neuron بال output

1. Get and prepare the dataset.
2. Build sequential model of layers that maps your inputs to your targets.
   → loss عشان نحدد شو ال
3. Compile the model and configure the learning process by choosing a loss function, an optimizer, and some metrics to monitor.
4. Train the model by calling the `fit()` method of your model.
5. Evaluate and use the model.

ال validation ما بتدخل بعملية ال training بس كأنها بتعطي hint
خلال ال training.

4.1 Get and Prepare the Dataset

```
import tensorflow as tf
from tensorflow import keras

# Get the Fashion MNIST
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) =
              fashion_mnist.load_data()

# Prepare the data train (55000), val (5000), test (10000)
X_valid = X_train_full[:5000] / 255.
X_train = X_train_full[5000:] / 255.
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
X_test = X_test / 255.
```

1) بنحول كل صورة ل 1D array وهاي
ال 1D هي ال "feature" القيمة -
بكل Pixel هي "feature"

ال deep neural ما بافدوا 2D Arrays.

→ Validation

→ labels

عشان نعمل normalization
للPixels
ويصير قيمها بين 0 & 1
بدل ما هي من 255 → 0

4.2 Build the Model

بتحول لـ 1D

بحول الصورة من 28*28 ← الى vector

↪ sequential model

The default is no activation function, i.e., linear layer.

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

hidden

output layer

[0, - - - - 784]

= (؟ ,) عدد الـ Columns

10 Classes لأنو

```
>>> model.summary()
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
dense_3 (Dense)              (None, 300)               235500
dense_4 (Dense)              (None, 100)               30100
dense_5 (Dense)              (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
```

عدد الـ edges

samples

column 784

neuron

عدد الـ rows (samples) غير محدد

↓

بتحدد لا عطيه samples

(انا حاليا بس بنيت model

⟶ model.Summary ( )

لإنه بدي الـ probabilites اللي بتطلع يكون مجموعها = 1

$300 * 784 = 235200 + 300$

From bias term

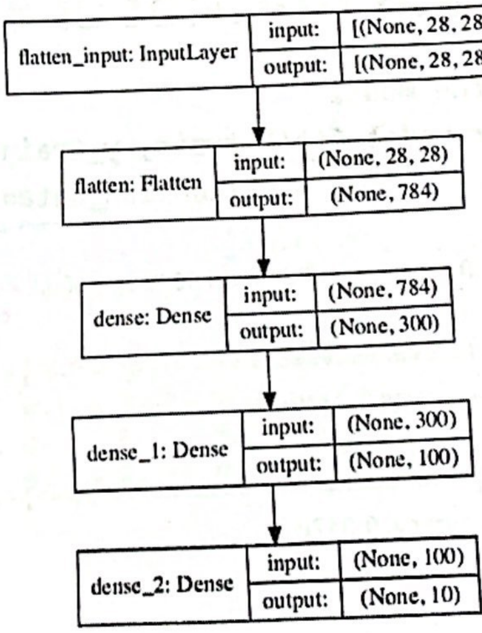
بطلع منه 300 edge

$(300 * 100) + 100$ From bias

☆ for more details: keras.io

4.2 Build the Model

```
# Plot the model
keras.utils.plot_model(
    model,
    "my_model.png",
    show_shapes=True)
```



☆ for binary classification

⟶ Single neuron output

activation ⟶ could be $\boxed{\text{sigmoid}}$

4.3 Compile the Model

multi Classification

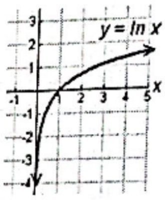
```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

كيف نعدل ال weight

Stochastic Gradient Descent

```
# For sparse labels (0-9):
loss = "sparse_categorical_crossentropy"
# For one-hot labels:
loss = "categorical_crossentropy"
# For binary labels:
loss = "binary_crossentropy"
# For regression:
loss = "mean_squared_error"
```

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$



بيحسب ال loss & cost
→ optimization for loss

Validation data مثلا
X_valid = X_train.iloc[1000:]
y_valid = y_train.iloc[1000:]

• categorical_crossentropy
↓
اذا كانت 1Hotencoding

4.4 Train the Model

one epoch:
→ round around all data

```
# Train the model
history = model.fit(X_train, y_train, epochs=30,
                    validation_data=(X_valid, y_valid))
```

بيخزن فيه

عشان كل مرة اشيك
ال accuracy و ال loss بكل epoch

Train on 55000 samples, validate on 5000 samples

Epoch 1/30

55000/55000 [==============================] - 2s 44us/sample - loss: 0.7226 - accuracy: 0.7641 - val_loss: 0.5073 - val_accuracy: 0.8320

Epoch 2/30

55000/55000 [==============================] - 2s 39us/sample - loss: 0.4844 - accuracy: 0.8321 - val_loss: 0.4541 - val_accuracy: 0.8478

...

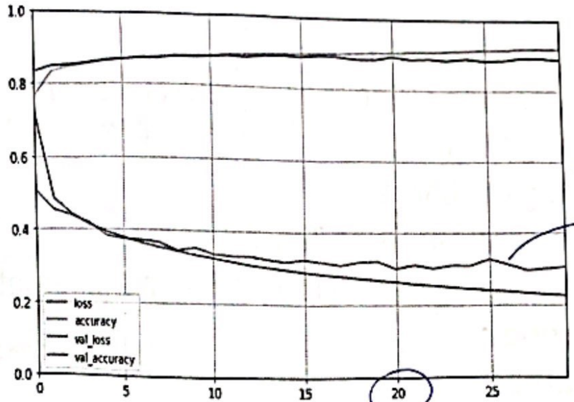
Epoch 30/30

55000/55000 [==============================] - 2s 39us/sample - loss: 0.2256 - accuracy: 0.9195 - val_loss: 0.3049 - val_accuracy: 0.8882

• بكل epoch بيبين التحسن

4.4 Train the Model

```
import pandas as pd
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
save_fig("keras_learning_curves_plot")
plt.show()
```



epochs

وصلنا لمرحلة انه ال Validation loss ما بتغير في حين انه ال training loss تقل

21

4.5 Evaluate and Use the Model

للداتا كبيرة → same as predict() in scikit learn.

```
model.evaluate(X_test, y_test)
10000/10000 [==============================] - 0s 21us/sample - loss: 0.3378 -
accuracy: 0.8781
[0.33780701770782473, 0.8781]
```

للداتا جديدة و موديل الجديد / ما بتطلع ال loss وال accuracy بس

بشوف كل sample شو جوابه

```
X_new = X_test[:3]
y_proba = model.predict(X_new)
y_proba.round(2)
array([[0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.01, 0.  , 0.99],
       [0.  , 0.  , 0.99, 0.  , 0.01, 0.  , 0.  , 0.  , 0.  , 0.  ],
       [0.  , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ]],
      dtype=float32)
```

→ sample 1
→ sample 2
→ " 3

```
model.predict_classes(X_new)
array([9, 2, 1])
```

classes

كأنه اخذنا نتائج ال Predict وعملنا argmax

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

5. Example - MNIST

1. **Define your training data**: input tensors and target tensors.
2. **Define a network** of layers (or **model**) that maps your inputs to your targets.
3. **Configure the learning process** by choosing a loss function, an optimizer, and some metrics to monitor.
4. **Iterate on your training data** by calling the `fit()` method of your model.

5. Example – Prepare the data

```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) =
      mnist.load_data()
#(60000, 28, 28), (60000), #(10000, 28, 28), (10000)
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
```

→ reshape to make them as 1 vector "1D"

عشان قيم ال pixels بالصورة بدل ما تكون من 0 - 255 بتصير من 0 - 1

حسب طبيعة الداتا مو شرط نعملها

```
from keras.utils import to_categorical   #one hot
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

→ one hot encoding

بحل هذا كل label صارت عبارة عن array فيها 10 ارقام واحد منهم بيساوي 1 الي هو رقم الكلاس

5. Example – Define and configure the network

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

عدد ال neurons

first hidden

output layer

لانه استعملنا ال 1Hot encoding

epoch كل جوا الخطوات عدد ← $\boxed{60000/128}$

5. Example – Training and evaluation

```
network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

ما اعطيته Validation data ⇓ حاجز يحطني Validation accuracy + loss

```
Epoch 1/5
60000/60000 [==============================] - 2s - loss: 0.2577 - acc: 0.9245
Epoch 2/5
60000/60000 [==============================] - 1s - loss: 0.1042 - acc: 0.9690
Epoch 3/5
60000/60000 [==============================] - 1s - loss: 0.0687 - acc: 0.9793
Epoch 4/5
60000/60000 [==============================] - 1s - loss: 0.0508 - acc: 0.9848
Epoch 5/5
60000/60000 [==============================] - 1s - loss: 0.0382 - acc: 0.9890
```

```
test_loss, test_acc = network.evaluate(test_images, test_labels)

9536/10000 [===========================>..] - ETA: 0s

test_acc: 0.9777
```


Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

1 neuron output

بال sequential لازم يكونوا ال layers ورا بعض

dense
كل neuron ماخد ال input من ال layer الي قبله ويعطي ال output لليبعده

6. Regression Using the Sequential Model

- Solve the **California housing** problem using a regression neural network.
- Scikit-Learn has `fetch_california_housing()` function to load the data
- This dataset contains **only numerical features** and there are **no missing values.**

Use encoding.

29

. Regression → مابنستعمل soft max
لانه بيعطي رقم مش Probability

عادي نستخدم ال sklearn هون

Check if ocean proximity موجودة

6.1 Get and Prepare the Dataset

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

housing = fetch_california_housing()

X_train_full, X_test, y_train_full, y_test =
    train_test_split(housing.data, housing.target, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,
                 y_train_full, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)
```

The default is 75% : 25%

حصتنا ال train مازيدها
هنا 75% لـ 80%

هونا فعليا قسم ال train
الى train & Validate

train 75% test 25%

train 75% Validate 25%

30

6.2 Build and Compile the Model

طريقة تانية
عشان نعمل الموديل.

```
# Building by passing a list of layers when creating
#  the Sequential model
model = keras.models.Sequential([
        keras.layers.Dense(30, activation="relu",
              input_shape=X_train.shape[1:]),
        keras.layers.Dense(1)
])
```

اول layer

layer

layer

size

لو لاني بتوقع من ال model
رقم (سعر البيت)

برجع ال
rows وال
للداتا columns

يعني لو كانت الداتا
15 feature
[1:]
برجع 15

```
# Compile with creating an optimizer object
model.compile(loss="mean_squared_error",
      optimizer=keras.optimizers.SGD(lr=1e-3))
```

The default is 0.01

1e-2

لل regression

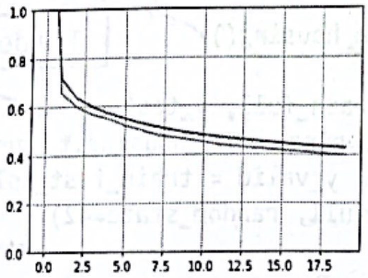
learning rate

31

6.3 Train and Evaluate the Model

```
history = model.fit(X_train, y_train, epochs=20,
        validation_data=(X_valid, y_valid))
```

بتكون
optional
بس الأفضل احطها
عشان نعمل monitoring



ال
error
عم
بقل.

```
mse_test = model.evaluate(X_test, y_test)
5160/5160 [==============================] - 0s
15us/sample - loss: 0.421
```

32

لو لاني عامل
check ← Scaling
. lables لل

6.4 Save and Restore the Model

- After training a model save it to a file.

```
model.save("my_keras_model.h5")
```

بنحتاج نعمل save للمودل
النهائي عشان نرجع
نستخدمه على داتا
ثانية بدون ما نرجع
ندربه كمان مرة.

- In the production program, load the trained model.

```
model = keras.models.load_model("my_keras_model.h5")
```

لما اعمله load ما ارجع اعمل Fit
بكمل من عند آخر weights وقفت عندهم
ما بعيد من أول.

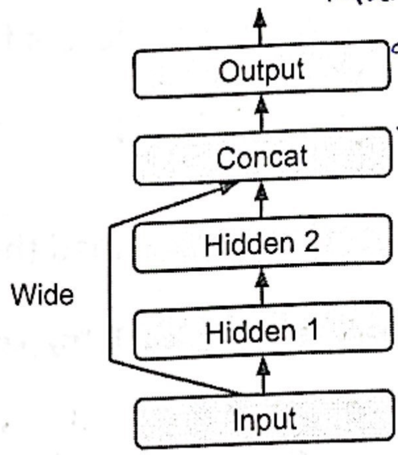
Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

بدل ال sequential API

7. Using the Functional API

- Keras functional API can be used to build arbitrary **static graph topologies**.
- Create a layer and as soon as it is created, **call it like a function**, passing it the input.
- Example 1: the **wide and deep** network that learns both deep patterns (using the deep path) and simple rules (through the short path).



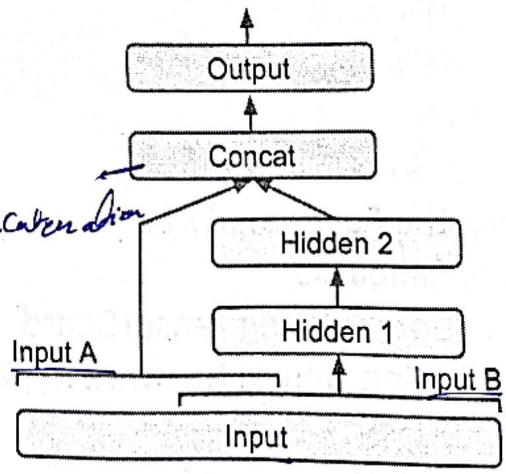
هون ال input عن ال output ممكن جي من ال input عشان اواي network ثانية

ممكن نبني sequential model باستخدام ال functional API

7. Using the Functional API

2. **Multi-input**: You can send a subset of the features through the wide path, and a different subset (possibly overlapping) through the deep path.

Concatenation



مباشرة، اجت حال output .) (feature جاهزة

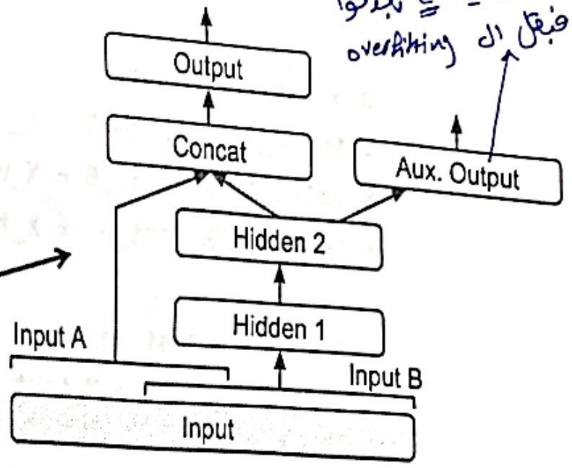
Scanned with CamScanner

7. Using the Functional API

3. Multiple Outputs

في ال loss وفي ال backpropagation
وصار في ع بدلوا
فبقلل ال overfitting

- To **locate and classify** the main object in a picture.
- **Multiple independent tasks** to perform based on the same data.
- **Regularization technique** (to ensure that the deep network learns something useful on its own).



تقلل احتمال ال overfitting (by randomness)

بيعطي weight لكل output.

7.1 Auxiliary Output for Regularization

اسم ال inputs

```
# Build the model
input_A = keras.layers.Input(shape=[5], name="wide_input")
input_B = keras.layers.Input(shape=[6], name="deep_input")

hidden1 = keras.layers.Dense(30, activation="relu")(input_B)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)

concat = keras.layers.concatenate([input_A, hidden2])

output = keras.layers.Dense(1, name="main_output")(concat)

aux_output = keras.layers.Dense(1, name="aux_output")(hidden2)

model = keras.models.Model(inputs=[input_A, input_B],
                           outputs=[output, aux_output])
```

ال input تبعها

Concat the 2 outputs

input

7.1 Auxiliary Output for Regularization

* overlap between A & B "2,3,4"

```
# Split the input
X_train_A, X_train_B = X_train[:, :5], X_train[:, 2:]
X_valid_A, X_valid_B = X_valid[:, :5], X_valid[:, 2:]
X_test_A,  X_test_B  = X_test[:, :5],  X_test[:, 2:]

# Take some test samples
X_new_A, X_new_B = X_test_A[:3], X_test_B[:3]
```


هنا ع loss ممكن يكون كل loss واحدة مختلف

7.1 Auxiliary Output for Regularization

for output — for aux-output — * التزم بالترتيب — Regularization

```
# Compile, train, evaluate, and predict
model.compile(loss=["mse", "mse"], loss_weights=[0.9, 0.1],
      optimizer=keras.optimizers.SGD(lr=1e-3))
```

90% من تعديل ال output
10% // // aux.

```
history = model.fit([X_train_A, X_train_B], [y_train, y_train], epochs=20,
      validation_data=([X_valid_A, X_valid_B], [y_valid, y_valid]))

total_loss, main_loss, aux_loss = model.evaluate([X_test_A, X_test_B],
                                                 [y_test, y_test])

y_pred_main, y_pred_aux = model.predict([X_new_A, X_new_B])
```

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

مش شرط يكون احسن مودل هو المودل عند ال epoch الأخير.
ومش شرط كل ما زدنا عدد ال models يصير المودل احسن.
مرات يكون اسوء لما نكمل ال epochs.
فبستخدم ال callbacks

8. Using Callbacks → لما نيجي نعمل fit

- The `fit()` method accepts a `callbacks` argument that lets you specify a list of objects that Keras will call during training
  - at the start and end of **training**
  - at the start and end of each **epoch**
  - before and after processing each **batch**
- There are many callbacks available in the `keras.callbacks` package. See

https://keras.io/callbacks/

8.1 Saving Best Model

- **Save your best model** when its performance on the validation set is the best so far.

```
checkpoint_cb = keras.callbacks.ModelCheckpoint(
        "my_keras_model.h5", save_best_only=True)
history = model.fit(X_train, y_train, epochs=10,
        validation_data=(X_valid, y_valid),
        callbacks=[checkpoint_cb])

# rollback to best model
model = keras.models.load_model("my_keras_model.h5")
mse_test = model.evaluate(X_test, y_test)
```

مكان تخزينه

to Save the best model "epoch with best performance"
→ min loss or
→ highest accuracy

parameter

بعمل load للـ model مع الـ Weights

train: read data, prepare, Fit, Save best model — on Hard disk

test: load model, evaluate

→ 2 files (notebooks)

8.2 Early Stopping

- Interrupt training when there is no progress on the validation set for a number of epochs (defined by the patience argument)
- Optionally roll back to the best model.

```
early_stopping_cb = keras.callbacks.EarlyStopping(
        patience=10, restore_best_weights=True)

history = model.fit(X_train, y_train, epochs=100,
        validation_data=(X_valid, y_valid),
        callbacks=[checkpoint_cb, early_stopping_cb])
```

10 epochs

اذا ال 10 epochs بعد ال best model ما جابوا نتائج احسن من ال best بوقف ال train

. ال Callbacks الي ممكن نستعملها :
1) Learning Rate Scheduler
2) TensorBoard
بتجمع statistics عن ال epochs وبرجعها ب GUI.

read about them

Outline

9. Visualization Using TensorBoard

- TensorBoard is a great interactive visualization tool that comes with TensorFlow.
- Use it using its callback

```
tensorboard_cb =
      keras.callbacks.TensorBoard(run_logdir)
history = model.fit(X_train, y_train, epochs=30,
      validation_data=(X_valid, y_valid),
      callbacks=[tensorboard_cb])
```

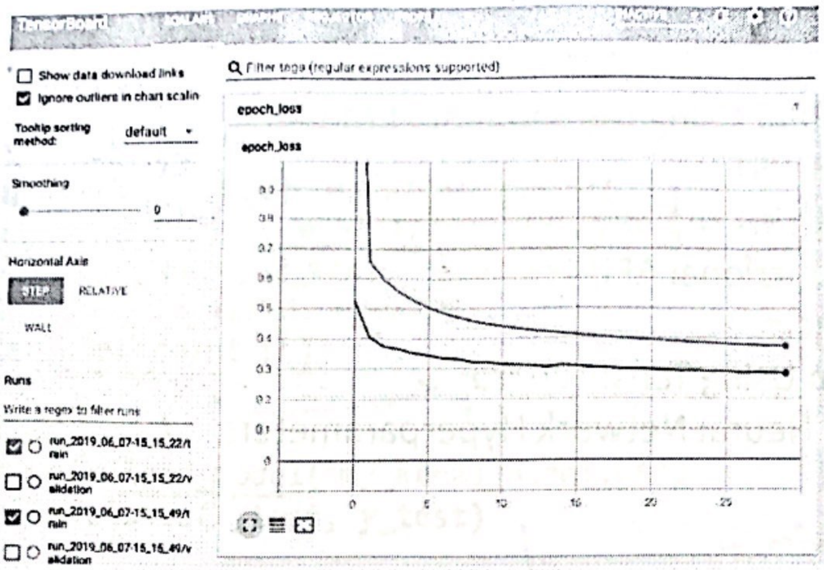
- Start TensorBoard server → عشان نعرضه

```
$ tensorboard --logdir=./my_logs --port=6006
```

بنعمل run من ال terminal

→ Port #

9. Open http://localhost:6006



Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

10. Fine-Tuning Neural Network Hyperparameters

. لو استخدمنا early-stopping عادي
لو استخدمنا عدد layers او neurons
اكثر من الي بنحتاجه .

- **Number of Hidden Layers**
  - One hidden layer can theoretically model even the most complex functions, provided it has enough neurons.
  - But for complex problems, deep networks have a much higher parameter efficiency than shallow ones.
- **Number of Neurons per Hidden Layer**
  - Pyramid across layers or same size → بنعمل نقلل عدد ال neuron بكل layer
  - Stretch pants: pick a model with more layers and neurons than you actually need, then use early stopping and other regularization techniques to prevent it from overfitting.
- Better to increase the number of layers instead of the number of neurons per layer.

from keras . wrappers . scikit_learn import // .

. Keras classifier
. Keras regressor } → شي مكافئ ل grid-search

. نخلي ال model جوا wrapper بخليه Compatible للمودل اي نتعامل معه .

او بنستعمل Keras tuner

10. Fine-Tuning Neural Network Hyperparameters

بأثر على سرعة ال convergence / بالعادة بتكون 10^{-3} / 10^{-2} → ممكن نستعمل learning Rate Decay (Callback)

- **Learning Rate**: the optimal LR is about half of the maximum LR.
- **Optimizer**: There are other than the Mini-batch Gradient Descent optimizer. → بنبدا 128 وبنقلل
- **Batch Size** ٥ باخذ ال batch بعمل forward-pass وبطلع ال loss وبعدل عال weights
  - Larger gives better speed up with hardware accelerators.
  - Smaller makes the models more general.
- **Activation Functions** → ال RELU من اكثر من الي funct. بتعطينا نتائج جيدة

بالعادة
128 او اقل
عشان ما تستهلك
ال memory او ليستعمل resources
كل الداتا وما يقدر يكمل .

11. Tutorials

- https://keras.io/
- https://www.tensorflow.org/guide/keras
- Keras Tutorial: Deep Learning in Python from DataCamp, https://www.datacamp.com/community/tutorials/deep-learning-python
- Keras Tutorial: The Ultimate Beginner's Guide to Deep Learning in Python, from EliteDataScience, https://elitedatascience.com/keras-tutorial-deep-learning-in-python

12. Exercise

From Chapter 10, solve exercise:

- 10. Train a deep MLP on the **MNIST** dataset (you can load it using `keras.datasets.mnist.load_data()`. See if you can get over **98%** precision. Try searching for the optimal learning rate by using the approach presented in this chapter (i.e., by growing the learning rate exponentially, plotting the error, and finding the point where the error shoots up). Try adding all the bells and whistles—save checkpoints, use **early stopping**, and plot learning curves using **TensorBoard**.

Summary

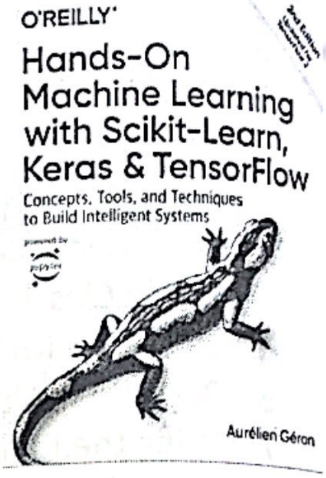
1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

Hands-on github ml2

Deep Neural Networks

Prof. Gheith Abandah

Reference



- Chapter 11: **Training Deep Neural Networks**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Outline

تعديلات كبيرة على ال weights

1. Introduction
2. Vanishing/Exploding Gradients Problems — تعديلات تختفي
   - Glorot and He Initialization
   - Nonsaturating Activation Functions
   - Batch Normalization
   - Gradient Clipping
3. Reusing Pretrained Layers
4. Faster Optimizers → we have used SGD
5. Avoiding Overfitting
   - ℓ_1 and ℓ_2 Regularization
   - Dropout
6. Summary
7. Exercise

1. Introduction

- Deep neural networks can solve **complex problems** and provide **end-to-end** solutions.
- When you train a deep network, you may face the following **problems:**
  1. • **Vanishing or exploding gradients:** The gradients grow smaller and smaller, or larger and larger.
  2. • **Not enough data**
  3. • **Long training time**
  - **Overfitting**

التعديلات اللي بنعملها على weight.
ممكن قيمها توصل للصفر (يعني مفيش اي تعديل)
وممكن تصير exploding ويصير في تعديلات كبيرة

ممكن نحلها
ب optimizers
تعمل convergence
بشكل fast.

Outline

1. Introduction
2. Vanishing/Exploding Gradients Problems
   - Glorot and He Initialization
   - Nonsaturating Activation Functions
   - Batch Normalization
   - Gradient Clipping
3. Reusing Pretrained Layers
4. Faster Optimizers
5. Avoiding Overfitting
   - ℓ_1 and ℓ_2 Regularization
   - Dropout
6. Summary
7. Exercise

2. Vanishing/Exploding Gradients Problems

$W_{old} = W_{new} - LR(\frac{dL}{dw}) \rightarrow \boxed{W_{old} = W_{new}}$ اذا كان $\boxed{\text{loss ثابت}}$

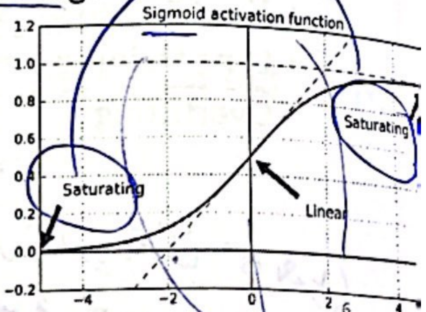
- **Vanishing Problem**: In the backpropagation algorithm, gradients often get smaller and smaller as the algorithm progresses down to the lower layers.
  - Lower layers' connection are left unchanged.

ال loss بتغير بشكل كبير

- **Exploding Problem**: the gradients can grow bigger and bigger.
  - Layers get very large weight updates and the algorithm diverges.
- **Main Reasons**: Using activation functions (1) (logistic sigmoid) and weight initialization (2) (normal distribution with 0-mean and 1-standard deviation).

ما بتأثر بعدد ال Features.

Vanishing



* Relu activation function doesn't have this problem.

ال deepneural بال layers ال اي بالبداية ← الحل نستخدم RELU مش sigmoid

Authors

بيكون ال gradient الها قريبة من الصفر، يعني ما بتعلم صح وما بعطي out صح.

على ال weights اعمل ال linear

كأنه اول ما ينضغط ال linear

2.1 Glorot and He Initialization

لما نعمل ال initialization لل weight علنا بقيم قليلة (ما نمنع ال exploding) بس بقلله (بنتجنبه)

- **Glorot and Bengio**: In order for the signal not to die out, nor to explode and saturate, the variance of the outputs of each layer should be equal to the variance of its inputs.
- **Solution**: the connection weights of each layer must be initialized randomly as follows:

Normal distribution with mean 0 and variance $\sigma^2 = \frac{1}{fan_{avg}}$

Or a uniform distribution between $-r$ and $+r$, with $r = \sqrt{\frac{3}{fan_{avg}}}$

$$fan_{avg} = (fan_{in} + fan_{out})/2.$$

اي طبقة

عدد ال Features

عدد ال edges الي داخله على neuron

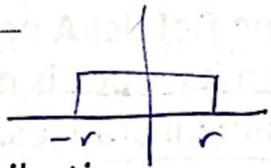
Fan_{in} fan_{out}

2.1 Glorot and He Initialization

- **Recommended** initialization parameters for each type of activation function.

Initialization	Activation functions	σ^2 (Normal)
Glorot	None, Tanh, Logistic, Softmax	$1 / fan_{avg}$
He	ReLU & variants	$2 / fan_{in}$
LeCun	SELU	$1 / fan_{in}$

- For the uniform distribution, use $r = \sqrt{3\sigma^2}$



- Keras uses **Glorot initialization** with a **uniform** distribution.

↳ default

2.1 Glorot and He Initialization

- To change it to **He initialization**:

```
keras.layers.Dense(10, activation="relu",
    kernel_initializer="he_normal") # Or "he_uniform"
```

- **He initialization** with a uniform distribution but based on **fan$_{avg}$**:

```
he_avg_init = keras.initializers.VarianceScaling(scale=2.,
                              mode='fan_avg', distribution='uniform')

keras.layers.Dense(10, activation="sigmoid",
                   kernel_initializer=he_avg_init)
```

* In Keras, no default for activation.
→ Check Keras.io

ال activation function كنا نستخدمها كانت saturating

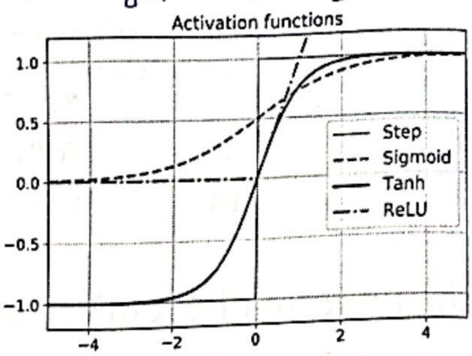
2.2 Nonsaturating Activation Functions

ال RElU Nonsaturating كل ما زاد ال input ال output ما بتغير

- **Step** does not work with the back propagation algorithm.

بـ saturating في حالة input سالب يعني output = zero

- **ReLU** is better than **sigmoid** because it does not saturate for positive values and is fast.
- **Dying ReLUs**: A neuron dies when its input is negative for all training instances.



saturating on negative values

ال activation = 0 ← ممكن توصل ل dying RELU

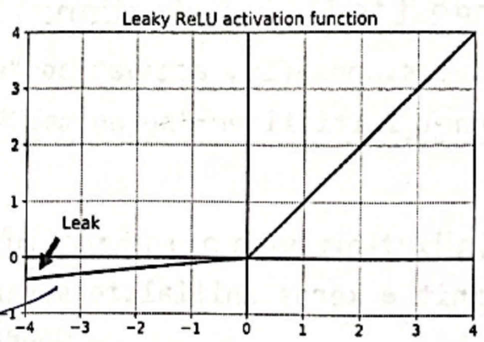
عشان كذا ال dying RELU او ال dying neuron

2.2 Nonsaturating Activation Functions

- **Leaky ReLU** performs better than ReLU.

$$\text{LeakyReLU}_\alpha(z) = \max(\alpha z,\ z)$$

- α between 0.01 and 0.3



small slope

```
model = keras.models.Sequential([
    …
    keras.layers.Dense(10, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(alpha=0.2), # added as a layer
    …
])
```

zero Parameter layer

slope

2.2 Nonsaturating Activation Functions

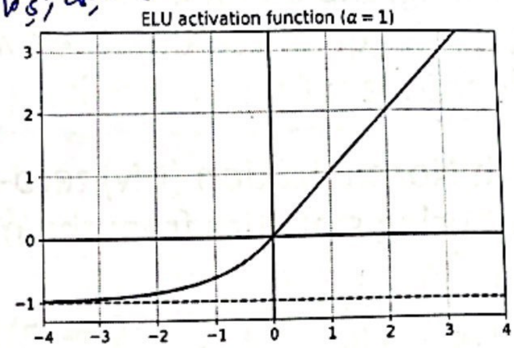
\# increase simulation time & it's slower

- **Exponential linear unit (ELU)** also performs better than ReLU but is slower. → complex derivative
- **Scaled ELU (SELU)** performs best with dense and CNN, (but must scale inputs and use **`lecun_normal`**.) → ال training time طويل

complex term

$$\mathrm{ELU}_{\alpha}(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

القيمة زي ماهي




```
layer = keras.layers.Dense(10, activation="selu",
    kernel_initializer="lecun_normal")
```


2.2 Nonsaturating Activation Functions

- **Summary:**
  - SELU > ELU > leaky ReLU > ReLU > tanh > logistic
- If you cannot use SELU, use ELU.
- For fast response, use leaky ReLU or ReLU.

Initialization بالعل الصح

normalization ممكن نعمل لكل activation بعد كل Layer

بخلي ال scale لل activation و convergence اسرع

2.3 Batch Normalization

→ to solve Vanishing / Exploding

يعدل على نفس ال weights و dampers

- The techniques in §2.1 and §2.2 can significantly reduce the vanishing/exploding gradients problems at the **beginning of training**, but don't guarantee that they won't **come back during training.**

- **Batch Normalization (BN)** zero-centers and normalizes each layer input using statistics from the mini batch (> 30).

- **Other benefits**: Works even without §2.1 and §2.2, allows using larger LR, and have regularization effect.

std و mean في across all batches (training samples)

batch لكل جديد

layer بعد كل بنضيفها

2.3 Batch Normalization

○ → 7
○ → 10
○ → 20

$\frac{7+10+20}{3} = 12.5$

- Implementing batch normalization with Keras is easy.

$\frac{7-12.5}{std} = \square$ ← std العينة

$\frac{10-12.5}{std} = \square$

$\frac{20-12.5}{std} = \square$

هاي القيم تمرر على next layer

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, activation="elu",
    kernel_initializer="he_normal"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(100, activation="elu",
    kernel_initializer="he_normal"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(10, activation="softmax")
])
```

ممكن يكونوا ال weight عالي
خاصة بال exploding

2.4 Gradient Clipping

اذا تجاوز ال max value → بيستعمل ال clip value

- Mitigates the exploding gradients problem by **clipping the gradients** during backpropagation so that they never exceed some threshold.
- Use it when you observe that the gradients are exploding during training. You can **track the size of the gradients** using TensorBoard.

```
optimizer = keras.optimizers.SGD(clipvalue=1.0)

model.compile(loss="mse", optimizer=optimizer)
```

كيف ال Gradient بيتغير at run time

Outline

1. Introduction
2. Vanishing/Exploding Gradients Problems
   - Glorot and He Initialization
   - Nonsaturating Activation Functions
   - Batch Normalization
   - Gradient Clipping
3. Reusing Pretrained Layers
4. Faster Optimizers
5. Avoiding Overfitting
   - ℓ_1 and ℓ_2 Regularization
   - Dropout
6. Summary
7. Exercise

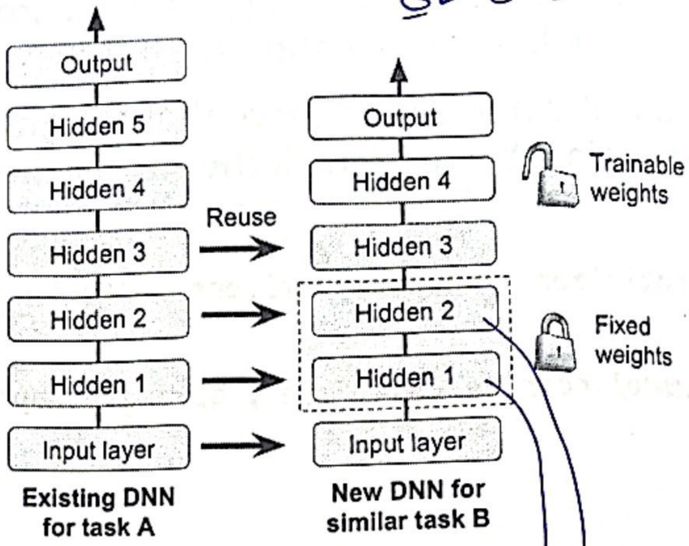
مهم

3. Reusing Pretrained Layers

كأنه المودل تعلم اشي ونقل معرفته لمودل تاني

- **Transfer Learning:** Using one NN developed for a certain task to solve another task.

- Useful to **shorten training time** or with **small datasets.**



ما بنعدل ال weights بهدول ال layers

Transfer Learning with Keras

```
# Load the ready model
model_A = keras.models.load_model("my_model_A.h5")
# Create a new model using all but the last layer
model_B_on_A = keras.models.Sequential(model_A.layers[:-1])
model_B_on_A.add(keras.layers.Dense(1, activation="sigmoid"))
# Freeze loaded layers then compile
for layer in model_B_on_A.layers[:-1]:
      layer.trainable = False
model_B_on_A.compile(loss="binary_crossentropy",
      optimizer="sgd", metrics=["accuracy"])
```

برجع كل ال layers ب model A

بستثني ال output layer

ضفنا new output layer

(locked) يعني ممنوع نعدل على ال weights

لما بستثناها في layer

Transfer Learning with Keras

```
# Train the model for a few epochs
history = model_B_on_A.fit(X_train_B, y_train_B, epochs=4,
      validation_data=(X_valid_B, y_valid_B))
# Unreeze loaded layers
for layer in model_B_on_A.layers[:-1]:
      layer.trainable = True
# Compile with small learning rate (defalut = 1e-2)
optimizer = keras.optimizers.SGD(lr=1e-4)
model_B_on_A.compile(loss="binary_crossentropy",
      optimizer=optimizer, metrics=["accuracy"])
```

في حال ما عجبنا النتيجة بنعمل unlock (بعدما خلص تدريب) عشان يعطينا نتائج افضل

Transfer Learning with Keras

```
# Train the model for more epochs
history = model_B_on_A.fit(X_train_B, y_train_B, epochs=16,
      validation_data=(X_valid_B, y_valid_B))
```

Outline

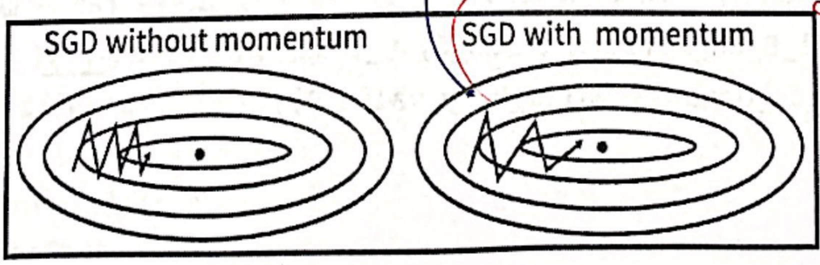
لو كانوا
اكثرهم اتجاه
بناخذ average.

4. Faster Optimizers

(history لما انه دايمًا (بال
المشي للامام
يصير ياخذ خطوات
اكبر بنفس الاتجاه

- The SGD optimizer can be made faster using **momentum optimization**

خطوات اسرع
oscillation
اقل



$$\theta \leftarrow \theta - \eta \nabla_{\theta} J(\theta)$$

1. $\mathbf{m} \leftarrow \beta \mathbf{m} - \eta \nabla_{\theta} J(\theta)$
2. $\theta \leftarrow \theta + \mathbf{m}$

$\boxed{\beta}$

```
optimizer = keras.optimizers.SGD(lr=0.001, momentum=0.9)
```

ال weight اي نعطيه لل history.

4. Faster Optimizers

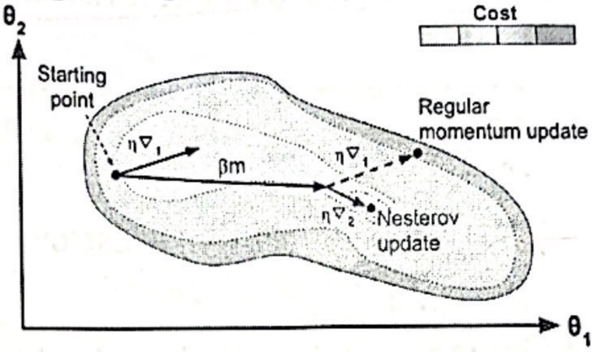
بحسب ال gradient
بعدما يتحرك للاتجاه.

- **Nesterov momentum optimization** measures the gradient of the cost function not at the local position **θ** but slightly ahead in the direction of the momentum, at **θ + βm**

average history

momentum coeff.

1. $\mathbf{m} \leftarrow \beta\mathbf{m} - \eta\nabla_{\theta}J(\theta + \beta\mathbf{m})$
2. $\theta \leftarrow \theta + \mathbf{m}$

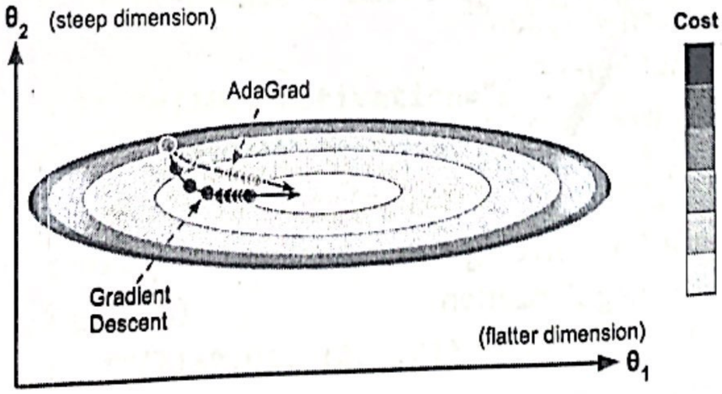



```
optimizer = keras.optimizers.SGD(lr=0.001, momentum=0.9,
                                 nesterov=True)
```


4. Faster Optimizers

بتعتمد على ال history عشان

- The **adaptive optimizers** such as **AdaGrad**, **RMSProp**, **Adam**, and **Nadam** scale down the gradient vector along the steepest dimensions.




```
optimizer = keras.optimizers.RMSprop()
optimizer = keras.optimizers.Adam()
```

4. Faster Optimizers

- RMSProp, Adam and Nadam often **converge fast**. But they can give poor **generalization.**
- Solution: Use Nesterov accelerated gradient

Class	Speed	Quality
SGD	*	***
SGD with momentum, Nestrov	**	***
Adagrad	***	*
RMSProp, Adam, Nadam, AdaMax	***	** or ***

Outline

1. Introduction
2. Vanishing/Exploding Gradients Problems
   - Glorot and He Initialization
   - Nonsaturating Activation Functions
   - Batch Normalization
   - Gradient Clipping
3. Reusing Pretrained Layers
4. Faster Optimizers
5. Avoiding Overfitting
   - ℓ_1 and ℓ_2 Regularization
   - Dropout
6. Summary
7. Exercise

جيد لانه بحط
عدد epochs
كبير وما بعرف
كم عددهم المناسب.

5. Avoiding Overfitting

- Deep neural networks typically have many parameters, giving them ability to fit a huge variety of complex datasets.

→ to avoid overfitting → الموديل مايكون مصمم لعدد samples معينة بدنا يكون general

- Useful regularization techniques:
  - Early stopping → Call back
  - Batch normalization
  - ℓ_1 and ℓ_2 regularization
  - Dropout

بعمل stop بعد epochs معينة وبعمل restore ال best model

تعديل عال loss

5.1 ℓ_1 and ℓ_2 Regularization

- Constrain a neural network's connection weights. → regularization factor

- ℓ_1: $Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|$ → التعديل هون بكون احسن لانه بتعامل مع ال weights نفسها بالتالي ال features المهمة ال weight تبعها بكون اكبر

ال loss زي ما يعدل

- ℓ_2: $Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$

```
layer = keras.layers.Dense(100, activation="elu",
        kernel_initializer="he_normal",
① → kernel_regularizer=keras.regularizers.l1(0.01))
# The other regularization functions:
② keras.regularizers.l2(0.01)
③ keras.regularizers.l1_l2(l1=0.01, l2=0.01)
```

λ → $0.01 \le \lambda \le 0.05$

للعملية التدريب

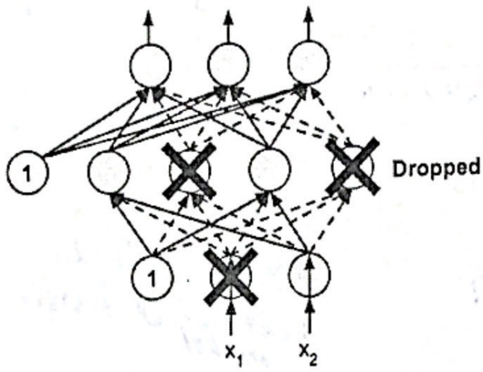
نلغي بعض ال neurons و ما بدخلوا بال forward pass.

في الحاسبة بتحدد كم neuron نلغي.

5.2 Dropout

* better generalization → ما بيعتمد على عدد معين من ال neurons او ما بكون في neuron محدد مسؤول عن النتيجة

- Popular technique to improve accuracy.
- At every training step, every neuron (excluding the output neurons) has a probability *p* of being temporarily **dropped out**.



المودل النهائي ما بنعمل فيه dropout.

النسبة بالعادة 20% يعني 0.2 بكل iteration

30

• في حال صار عندي مشكلة ب neuron معين مثل dying neuron ما بتأثر المودل كامل وهاي برضو طريقة لتجنب ال overfitting.

بنضاف بشكل layer منفصلة

5.2 Dropout

بستخدم بعد ال activation وقبل ال BatchNormalization

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(300, activation="elu",
          kernel_initializer="he_normal"),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(100, activation="elu",
          kernel_initializer="he_normal"),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(10, activation="softmax")
])
```

20% من 300

بقدر أزيد نسبة ← بس يفضل ما تعلى

in train

* لما اعمل test بفضل اضرب ال weights كلهم ب (1-rate) ← keep ratio

لانه ما في drop لما اعمل test

3

Outline

1. Introduction
2. Vanishing/Exploding Gradients Problems
   - Glorot and He Initialization
   - Nonsaturating Activation Functions
   - Batch Normalization
   - Gradient Clipping
3. Reusing Pretrained Layers
4. Faster Optimizers
5. Avoiding Overfitting
   - ℓ_1 and ℓ_2 Regularization
   - Dropout
6. Summary
7. Exercise

6. Summary

- **Recommended default DNN** configuration

1epoch can be divided into steps

Hyperparameter	Default value
Kernel initializer	He initialization
Activation function	ELU or leaky
Normalization	None if shallow; Batch Norm if deep
Regularization	Early stopping (+ℓ_2 reg. if needed)
Optimizer	Momentum optimization (or RMSProp or Nadam)
Learning rate schedule	1 cycle

in Callbacks

6. Summary

- For a simple **stack of dense** or **CNN layers.**

Hyperparameter	Default value
Kernel initializer	LeCun initialization
Activation function	SELU
Normalization	None (self-normalization)
Regularization	Alpha dropout if needed
Optimizer	Momentum optimization (or RMSProp or Nadam)
Learning rate schedule	1 cycle

7. Exercise

11.8. Practice training a deep neural network on the **CIFAR10 image dataset:**

a) Build a DNN with 20 hidden layers of 100 neurons each (that's too many, but it's the point of this exercise). Use He initialization and the ELU activation function.
b) Using Nadam optimization and early stopping, train the network on the CIFAR10 dataset. You can load it with keras.datasets.cifar10.load_ data(). The dataset is composed of 60,000 32 × 32–pixel color images (50,000 for training, 10,000 for testing) with 10 classes, so you'll need a softmax output layer with 10 neurons. Remember to search for the right learning rate each time you change the model's architecture or hyperparameters.
c) Now try adding Batch Normalization and compare the learning curves: Is it converging faster than before? Does it produce a better model? How does it affect training speed?
d) Try replacing Batch Normalization with SELU, and make the necessary adjustments to ensure the network self-normalizes (i.e., standardize the input features, use LeCun normal initialization, make sure the DNN contains only a sequence of dense layers, etc.).
e) Try regularizing the model with alpha dropout. Then, without retraining your model, see if you can achieve better accuracy using MC Dropout.
f) Retrain your model using 1cycle scheduling and see if it improves training speed and model accuracy.

Deep Computer Vision Using Convolutional Neural Networks

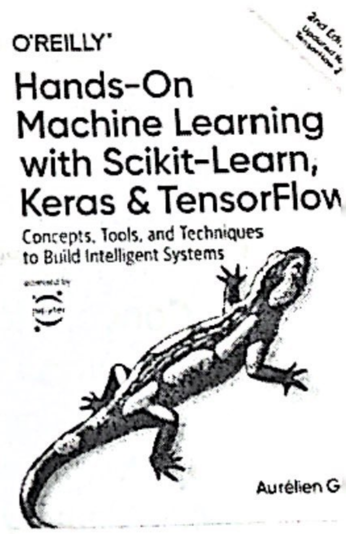
تستعمل ← CNN
بشكل اساسي بالصور
يتم تحويل الصورة ل ~~vector~~ vector
وبعمل matching بينه وبين الصور.

الهدف منها تعمل ال extract features.

Prof. Gheith Abandah

بنختزل عدد ال features الكبير بعدد اقل تكون representation للعدد الكبير
مجهزين بناخذها على neural network

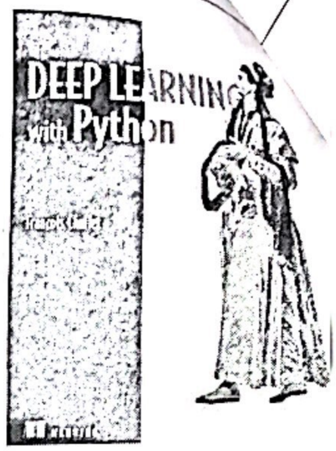
Reference



- Chapter 14: **Deep Computer Vision Using Convolutional Neural Networks**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Reference



- **Deep Learning with Python**, by François Chollet, Manning Pub 2018

Outline

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
    3. Mathematical summary
    4. Memory requirements
3. Pooling layer
4. CNN architectures
    1. Example – Fashion MNIST
    2. ResNet
5. Using pretrained models
6. Pretrained models for transfer learning
7. Classification and localization
8. Object detection
9. Semantic segmentation
10. Exercises

Introduction

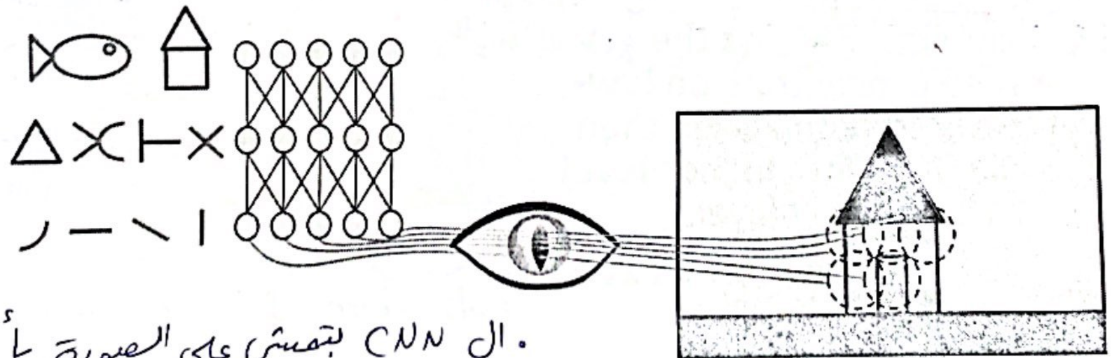
• YouTube Video: **Convolutional Neural Networks (CNNs) explained** from Deeplizard

https://youtu.be/YRhxdVk_sIs

1. Introduction

• كيف عرفنا انه الي بالصورة عبارة عن بيت ؟ ← هاد الأشي مسؤول عنه ال CNN
لانه في مثلث ومربع ومستطيل ضمن الشكل

- **Convolutional neural networks (CNNs)** emerged from the study of the brain's **visual cortex.**
- Many neurons in the visual cortex have a small **local receptive field.**



• ال CNN بتمشي على الصورة بأول مراحل بتشوف ال edges عن طريق الفروقات و بآخر مرحلة بحولها ل vector.

• هو بميز بس اذا في بيت او لا بس ما بقدر يحدد وين او موقعه بالصورة

Outline

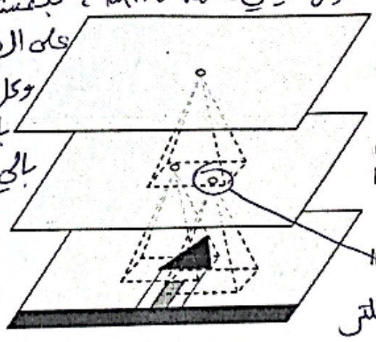
2. Convolutional Layer

• ال neurons ما بكونوا بالتركيبة نفسها
ال dense network .

Kernel
filters

بكون في عنا window اسمها Kernel وهاد ال Kernel او ال filter
اله قيم معينة وساين يعني initial values ، بنمشي ال filter
على الصورة الأصلية
وكل cell
بال filter بضربه
بالي مقابله بالصورة

- **Neurons** in one layer are not connected to every single pixel/neuron in the previous layer, but only to pixels/neurons in their **receptive fields**.
- This architecture allows the network to concentrate on **low-level features** in one layer, then assemble them into **higher-level features** in the next layer.
- Each layer is represented in **2D**.



ناتج الفلتر

• لما ننتقل من layer ال layer التانية
بنختزل ال features

1	1	1			
1	1	1			
1	1	1			

صورة واول مرة مشينا
ال filter على اول
4 Pixels
بنضرب كل نقطة بالفلتر
بالقيمة الي بتقابلها وبنجمعهم

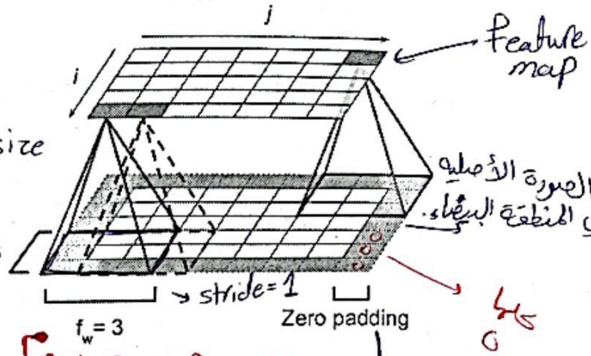
الناتج بروح
على ال next layer

بنمشي
الفلتر على باقي
ال Pixels

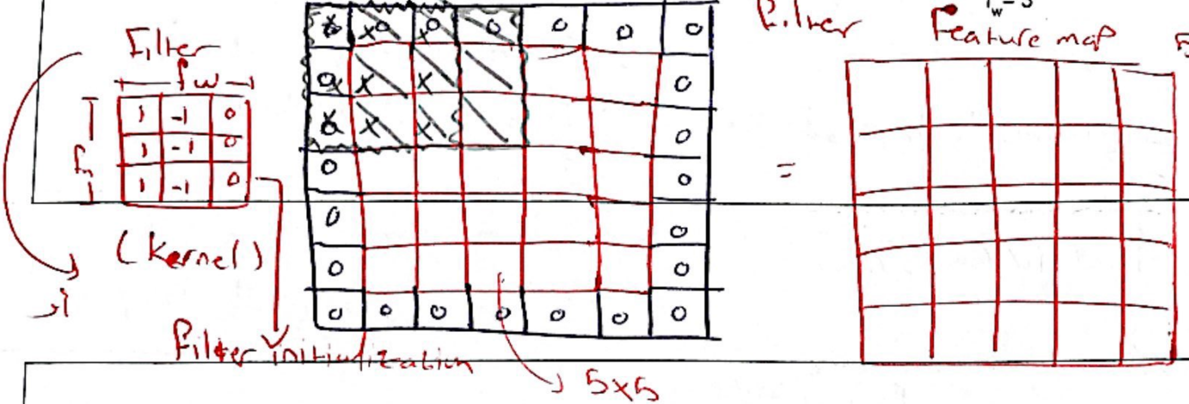
2. Convolutional Layer

- f_h and f_w are the height and width of the receptive field.
- **Zero padding**: In order for a layer to have the same height and width as the previous layer, it is common to add zeros around the inputs.

•stride → size of step



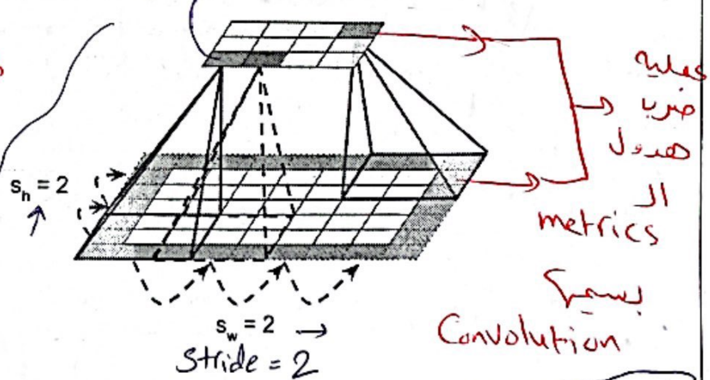
تحرك ال Filter



2. Convolutional Layer

- It is also possible to connect a large input layer to a smaller layer by **spacing out** the receptive fields.
- The distance between two consecutive receptive fields is called the **stride**.
  طريقة حركة ال Filter
  horizontal & vertical.
- A neuron located in row *i*, column *j* is connected to the neurons in the previous layer located in: Kernel or filter
  - Rows: $i \times s_h$ to $i \times s_h + f_h - 1$
  - Cols: $j \times s_w$ to $j \times s_w + f_w - 1$

بالعادة $f_h = f_w$



بالعادة $s_h = s_w$

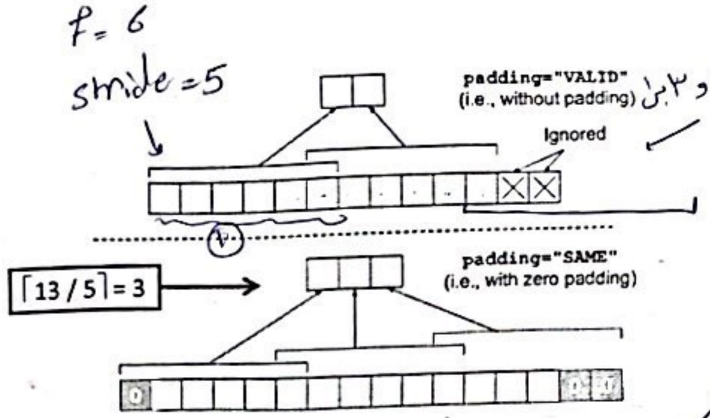
• لما نزيد ال stride بصير في مناطق ما مر عليها الفلتر غير مرة وحدة مثلا stride=2

• لما تكون stride=1 بقدر يشوف ال low level details

لا يتبدأ أمشي الأخرى Filter على نص الصورة

2. Convolutional Layer

- Keras supports
  - No padding (default)
    padding="VALID"
  - Zero padding
    padding="SAME"
- Example:
  - Input width: 13
  - Filter width: 6
  - Stride: 5



٣ جوا و ٣ برا

. if no Padding → لو كان ال filter ما بتطبق على Pixels كلهم من الصورة خلص ما بكمل (ما بنعمل الخطوة) و عمل بالاتجاه الثاني ← بس بصير في معلومات ما خليناهم.

Padding = ⌈Pixels Per row / stride⌉
ceiling

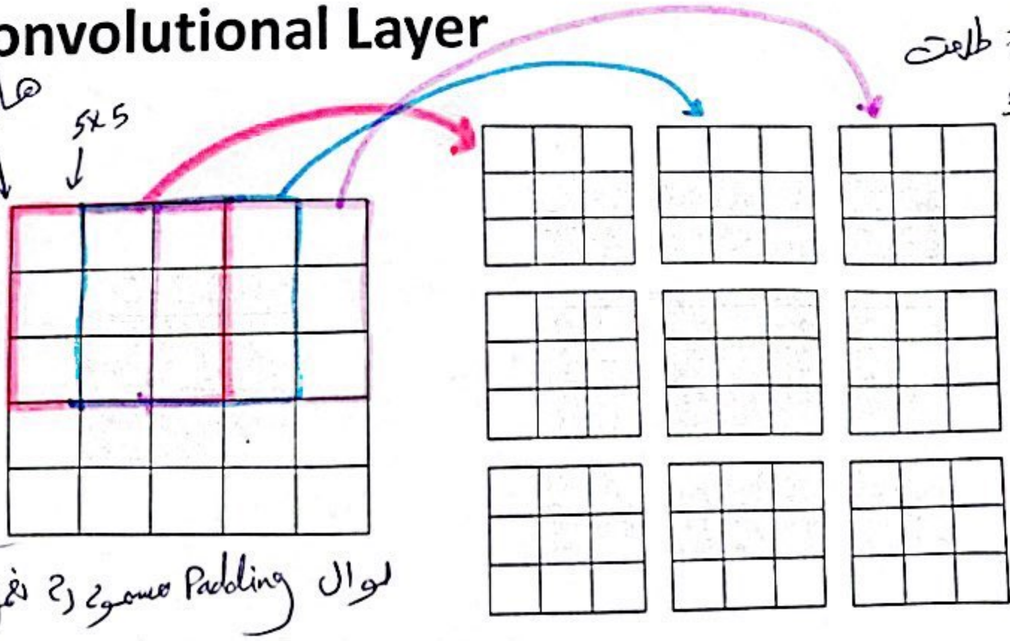
والأفضل نوزع ال Padding عالشمال و عاليمين.

. if Padding = same → ال Pixels الي زيادة اعتبرهم أصفار, بالتالي بنشمل كل الصورة.

2. Convolutional Layer

. ال Feature map طلعت معنا 3X3

هاي الصورة الأصلية بدنا نمشي filter عليها 3x3 stride=1 f=5x3



لحد هون بنزل خطوة لتحت

لو ال Padding مسموح ح نعمل كمان خطوتين زيادة Virtically

Figure 5.5 Valid locations of 3 × 3 patches in a 5 × 5 input feature map

. لو ال Padding = same ح يطلع الجواب (Feature map) = 5X5

Size of Feature map before convolution = " " after

بشرط ↳ stride=1

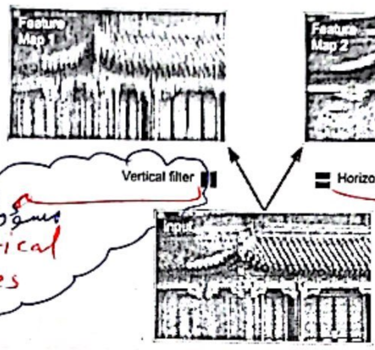
ال filter يعادل ال weights الي استخدمناها بال DNN
وعادي ارجع اعدل ع أعدل ال initialization تبع ال filter

2.1 Filters

. ال filter هو نفسه ال kernel.

• كل filter بلقط behavior معينة

- A neuron's weights can be represented as a small image the size of the receptive field, called **filters.**
- When all neurons in a layer use the same line filters, we get the **feature maps** on the top.



بلقط ال Vertical edges

بلقط ال Horizontal edges

*every filter gives a Feature map.

مثل

0	1	0
0	1	0
0	1	0

مثلا

0	0	0
1	1	1
0	0	0

بس 0 , 1 لانه عنا ابيض واسود الألوان
ممكن تكون -1,0,1
الأسود ← -1 , 0 عادي , 1 → ابيض

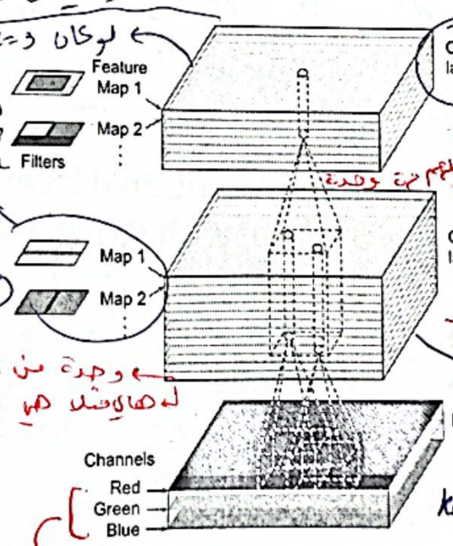
2.2 Stacking Feature Maps

الصور ممكن تكون gray scale ← بتكون 1 channel.
ال Pixels فيها من 0 ل 255 / الألوان فيها درجات الرمادي والابيض والأسود

- In reality, each layer is **3D** composed of **several feature maps** of equal sizes.
- **Within one feature map**, all neurons **share** the same parameters, but **different** feature maps may have **different** parameters.
- Once the CNN has learned to **recognize a pattern in one location**, it can recognize it in any other location.

لو كان k=3 ← (3,3,10)
kernel size
بطلع منهم جواب واحد.

Size of each one depends on stride and Padding



(f,f,#of layers in previous)

بياخذ كل filter كل مرة pixel وحدة ويمشي

Gray Scale

عدد ال layers بيساوي عدد ال kernels

وحدة من ال layers هاي مثلا هي ال

feature map1 from filter 1

3 channels (3D image)

الصور الملونة

وال kernel صار 3D

OpenCV → library to apply image Processing techniques.

منها: تحويل الصورة من color coding ل color coding تاني

مثلا لو ال kernel size عبارة عن 7x7 بال 2D
تكون 7x7x3 بال 3D
3 channels

2.3 Mathematical Summary

Equation 14-1. Computing the output of a neuron in a convolutional layer

Filter height — width — عدد ال Channels

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{with} \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

- $z_{i,j,k}$ is the **output** of the neuron located in row *i*, column *j* in feature map *k*
- $f_{n'}$ is the number of **feature maps** in the previous layer

الصورة الأصلية RGB

100 — 150

الصورة الأصلية = 150x100x3

2.4 Memory Requirements

• كونه ال Padding same يعني نفس ال Size الصورة feature map

- Convolutional layers require a **huge amount of RAM.**

في معظم 200 — يعني 200 kernel

- **Example:** Convolutional layer with 5 × 5 filters, 200 feature maps of size 150 × 100, with stride 1 and "same" padding. Input is RGB image (three channels).
  - Parameters = (5 × 5 × 3 + 1) × 200 = 15,200
  - Size of feature maps (single precision) = 200 × 150 × 100 × 4 = 12 MB of RAM
  - 1.2 GB of RAM for a mini batch of 100 instances

بعدد الألوان — 3 channels — size of the kernel — bias — عشان ننتقل من ال input لاول layer — byte — لصورة واحدة — 200 feature map لكل صورة

* Run on GPU for acceleration & parallelism

• Single precision = 4 byte
↳ for each cell in the feature map

• بنستخدم ال batch size لانه ماخذا memory تستوعبه حجم الجي

Outline

اذا استخدمت strides 1 و padding فلية same رح يضل size الصورة بعد ال Convolution نفسه

3. Pooling Layer

ما الها Parameters

فبنقلل ال Pixels الي بتكون قريبة من بعض.

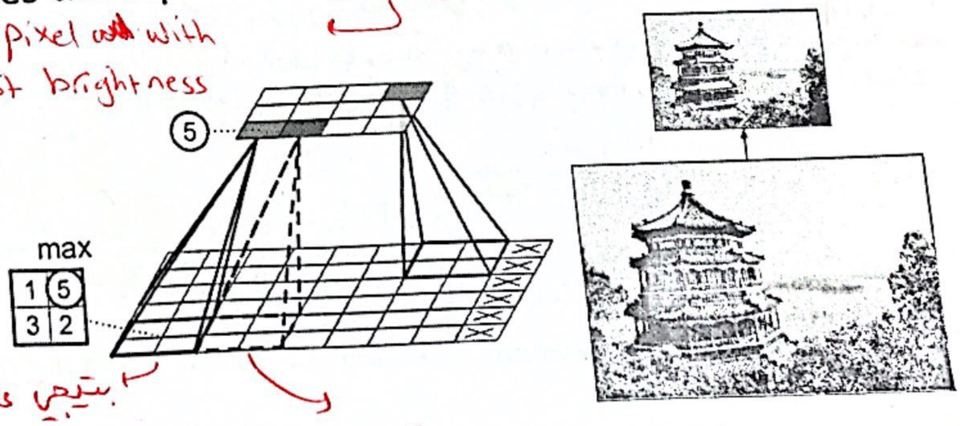
لـ بنطبقها بعد ال Conv.

- Its goal is to **subsample** (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters.

تقليل ال size لل feature maps.

- It aggregates the inputs using **max** or **mean**.

pixel with most brightness



بتيجي على 4 بتاخذ ال max أو min أو mean ← Pooling

بتروح على ال 4 الي بعدهم ← ما في overlap

Outline

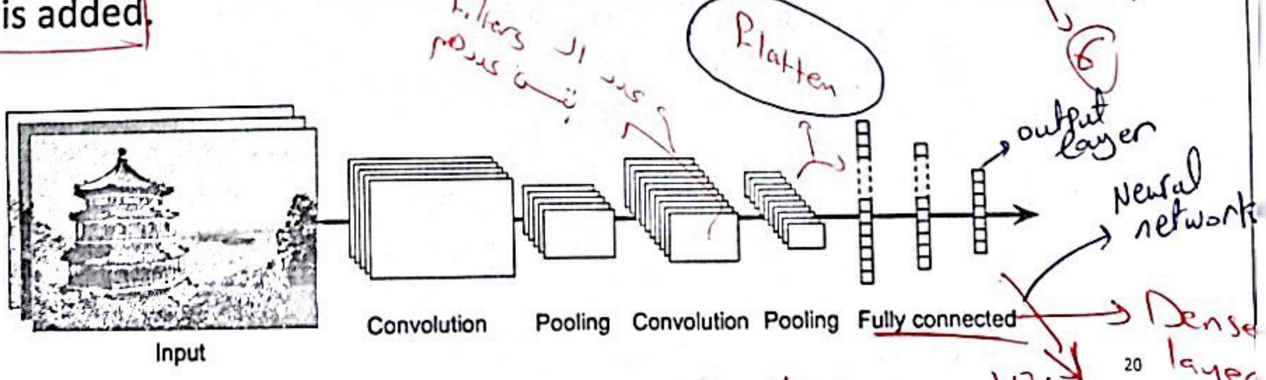
1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
    3. Mathematical summary
    4. Memory requirements
3. Pooling layer
4. CNN architectures
    1. Example – Fashion MNIST
    2. ResNet
5. Using pretrained models
6. Pretrained models for transfer learning
7. Classification and localization
8. Object detection
9. Semantic segmentation
10. Exercises

4. CNN Architectures

١. اختيار ال features غير كافي

عشان نطلع ال predictions فبنضيف بالنهاية dense layers

- **Stack few convolutional layers** (each one generally followed by a **ReLU layer**), then a **pooling layer**, then another few convolutional layers, then another pooling layer, and so on. The image gets **smaller and smaller**, but it also gets **deeper and deeper**. At the end, a **dense NN is added**.



CNN not stand alone, it's almost feature extraction

4.1 Example – Fashion MNIST

```
model = keras.models.Sequential([
    keras.layers.Conv2D(64, 7, activation="relu", padding="same",
        input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation="softmax")
])
```

Filter size

Feature maps

2×2 window and stride 2

عدد ال filters

kernel size (7,7,1)

= 7 x 7 x 1

for one image

output size = 28 x 28 x 64

mnist 1 channel

2x2

out size = 14 x 14 x 64

3 x 3 x 64 previous layer

14 x 14 x 128

7 x 7 x 128

3 x 3 x 128

حولها 1D

بنزيد عدد ال filters و بنصغر ال size للصورة

10 classes

جواب 10 prob

4.1 Example – Fashion MNIST

```
model.compile(loss="sparse_categorical_crossentropy",
        optimizer="nadam", metrics=["accuracy"])

history = model.fit(X_train, y_train, epochs=10,
        validation_data=(X_valid, y_valid))
Train on 55000 samples, validate on 5000 samples
Epoch 1/10 55000/55000 [==============================] - 51s 923us/sample - loss:
0.7183 - accuracy: 0.7529 - val_loss: 0.4029 - val_accuracy: 0.8510
…
Epoch 10/10
55000/55000 [==============================] - 50s 911us/sample - loss: 0.2561 -
accuracy: 0.9145 - val_loss: 0.2891 - val_accuracy: 0.9036
```

بالتعديل ال labels

optional

4.1 Example – Fashion MNIST

```
score = model.evaluate(X_test, y_test)
X_new = X_test[:10] # pretend we have new images
y_pred = model.predict(X_new)

10000/10000 [==============================] - 2s 239us/sample - loss:
0.2972 - accuracy: 0.8983
```

Can reach 92% with more epochs

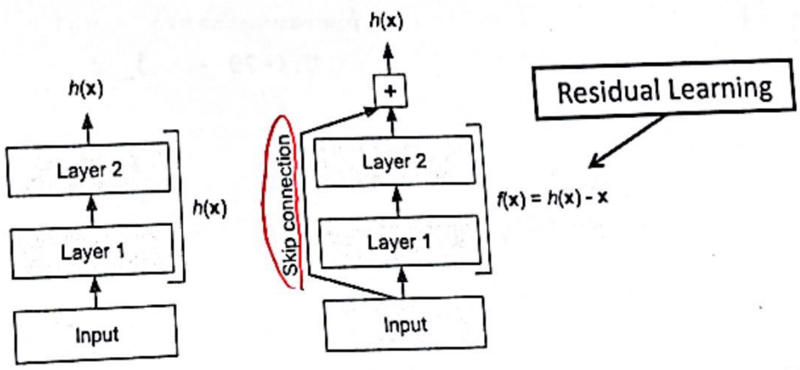
4.2 ResNet

net-work مدربة

- **Residual Network** (or ResNet) won the ILSVRC 2015 challenge.
- Top-5 error rate under 3.6%, using an extremely deep CNN composed of **152 layers**.
- To train such a deep network, it uses **skip connections**.

اعلى 5 pred.

وبنشوف هل الصورة من اعلى 5 pred.

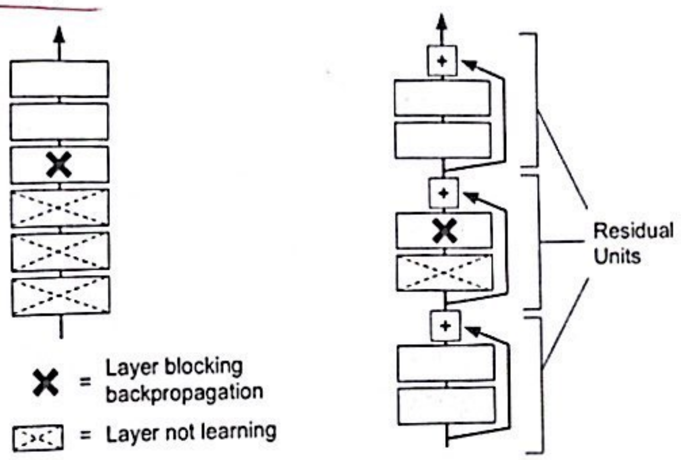


4.2 ResNet

.ما استخدموا ال sequential عشان لو بدنا نعمل update لـ layer لازم نمر على كل الي قبلها . فاستخدموا ال skip connections.

عشان ال backpropagation أسرع.

- The network can start making progress even if several layers have not started learning yet.



25

4.2 ResNet

موجود بـ keras.model

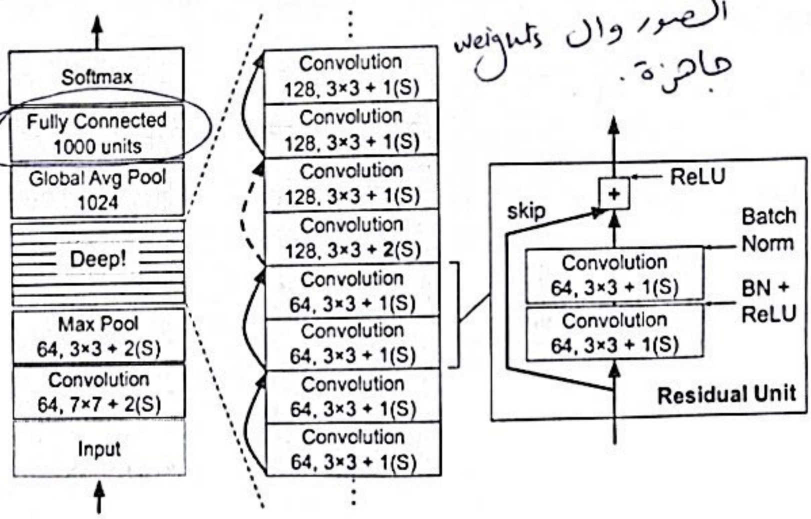
model جاهز مدرب على الاف الصور وال weights جاهزة.

- ResNet is a stack of residual units.

ال output layer

.لو كان بدنا نستخدم لعدد كلاسس أقل من ال 1000 او اشي ما تدرب عليه ، لازم نغير ال output layer

او ممكن ندرب ال hidden layers القريبة من الأوت



26

Scanned with CamScanner

Outline

5. Using Pretrained Models

- Pretrained networks are readily available from the **keras.applications** package.
- Check https://github.com/keras-team/keras-applications
- You can load the **ResNet-50** model, pretrained on **ImageNet**, with the following line of code:

```
model = keras.applications.resnet50.ResNet50(weights="imagenet")
```

صار عنا model جاهز.

اعمل load مع ال weights الي وصلها لما تدرب على الاف الصور.

5. Using Pretrained Models

```
# Input: 224 × 224-pixel images
images_resized = tf.image.resize(images, [224, 224])

# Preprocess images, should be scaled 0-255
inputs = keras.applications.resnet50.preprocess_input(
        images_resized * 255)

Y_proba = model.predict(inputs)

# Get top predictions out of the 1000-class probs.
top_K = keras.applications.resnet50.decode_predictions(Y_proba, top=3)
```

بعمل resize للصور
حسب ال model
شو ال Input shape
الي بقبله

5. Using Pretrained Models

```
# Print results
for image_index in range(len(images)):
    print("Image #{}".format(image_index))
    for class_id, name, y_proba in top_K[image_index]:
        print(" {} - {:12s} {:.2f}%".format(class_id, name, y_proba * 100))
    print()
Image #0
  n03877845 - palace       42.87%
  n02825657 - bell_cote    40.57%
  n03781244 - monastery    14.56%

Image #1
  n04522168 - vase         46.83%
  n07930864 - cup          7.78%
  n11939491 - daisy        4.87%
```

Correct Class

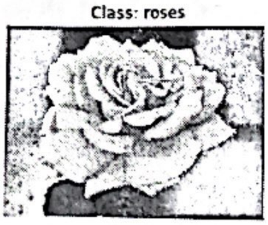
Outline

1. Introduction
2. Convolutional layer
   1. Filters
   2. Stacking feature maps
   3. Mathematical summary
   4. Memory requirements
3. Pooling layer
4. CNN architectures
   1. Example – Fashion MNIST
   2. ResNet
5. Using pretrained models
6. Pretrained models for transfer learning
7. Classification and localization
8. Object detection
9. Semantic segmentation
10. Exercises

6. Pretrained Models for Transfer Learning

- Training a pretrained network (**Xception**) for a dataset from TFDS (https://www.tensorflow.org/datasets).
- `tf_flowers`: 3670 images, 5 classes

Class: roses



```
# Load the dataset
import tensorflow_datasets as tfds

dataset, info = tfds.load("tf_flowers",
      as_supervised=True, with_info=True)

dataset_size = info.splits["train"].num_examples # 3670
n_classes = info.features["label"].num_classes   # 5
class_names = info.features["label"].names
```

6. Pretrained Models for Transfer Learning

```
# Reloaad the dataset with three splits tf.data.Dataset
test_set_raw, valid_set_raw, train_set_raw = tfds.load(
        "tf_flowers", split=["train[:10%]",
        "train[10%:25%]", "train[25%:]"],
        as_supervised=True)

# Define the preprocessing function
def preprocess(image, label):
    resized_image = tf.image.resize(image, [224, 224])
    final_image = keras.applications.xception.preprocess_input(
        resized_image)
    return final_image, label
```


6. Pretrained Models for Transfer Learning

```
# Apply this preprocessing function to the 3 datasets
# Shuffle the training set
# Add batching and prefetching to all the datasets
batch_size = 32   →   32 صورة
train_set = train_set_raw.shuffle(3000).repeat()
train_set = train_set.map(preprocess).batch(
        batch_size).prefetch(1)
valid_set = valid_set_raw.map(preprocess).batch(
        batch_size).prefetch(1)
test_set = test_set_raw.map(preprocess).batch(
        batch_size).prefetch(1)
```

6. Pretrained Models for Transfer Learning

```
# Load an Xception model, pretrained on ImageNet
#  excluding the global avg pool. and dense o/p layers
base_model = keras.applications.xception.Xception(
       weights="imagenet", include_top=False)
```

model. Summary() top layer (output) (deleted)

```
# Add global avg pool. layer based on model output
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(n_classes, # Add desnse o/p
       activation="softmax")(avg)
model = keras.models.Model(inputs=base_model.input,
       outputs=output) # Create the Keras Model
```


6. Pretrained Models for Transfer Learning

```
# Freeze the weights of the pretrained layers
for layer in base_model.layers:
    layer.trainable = False
```

ماتنجمع
weights ال نفسن بلي تعديلهم وطليعهم

$\eta(t) = \eta_0 \,/\, (1 + t/k)$

```
# Compile the model and start training
optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9,
      decay=0.01) # LR=0.2 with scheudle, k=1/0.01
model.compile(loss="sparse_categorical_crossentropy",
      optimizer=optimizer, metrics=["accuracy"])
history = model.fit(train_set, epochs=5,
      validation_data=valid_set) # Tops at 75-80% acc.
```

6. Pretrained Models for Transfer Learning

```
# Unfreeze the weights of the pretrained layers
for layer in base_model.layers:
    layer.trainable = True
```

→ unfreeze

• دربنامين اول وجديد

```
# Recompile with lower LR and decay
optimizer = keras.optimizers.SGD(lr=0.01, momentum=0.9,
     nesterov=True, decay=0.001)
model.compile(loss="sparse_categorical_crossentropy",
    optimizer=optimizer, metrics=["accuracy"])
history = model.fit(train_set, epochs=40,
    validation_data=valid_set) # Result: 95% acc.
```


Outline

1. Introduction
2. Convolutional layer
   1. Filters
   2. Stacking feature maps
   3. Mathematical summary
   4. Memory requirements
3. Pooling layer
4. CNN architectures
   1. Example – Fashion MNIST
   2. ResNet
5. Using pretrained models
6. Pretrained models for transfer learning
7. Classification and localization
8. Object detection
9. Semantic segmentation
10. Exercises

وين استعملنا CNN

اذا كان عندي مثلا وردة وحدة في الصورة

7. Classification and Localization

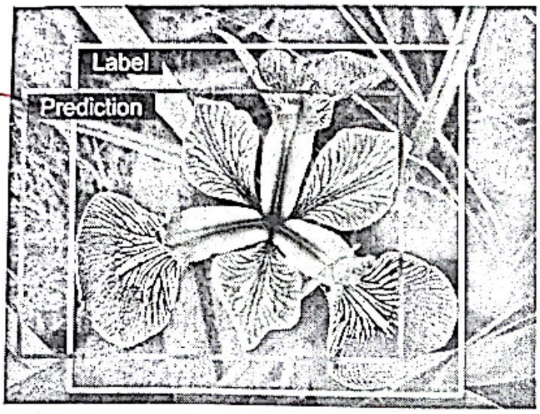
ل ـــــ Classification

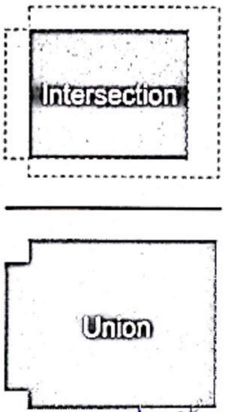
يقارن ال Pred. Box مع ال ground truth

تحديد مكان ال object

- **Localizing** an object in a picture can be expressed as a **regression** task.
- Predict the horizontal and vertical coordinates of the object's center and its height and width.

bounding box.





metric to determine if the bounding box is true

Common metric: the Intersection over Union (IoU) → If overlapped 100% intersection = $\boxed{1}$

↑ IoU better

classification.

2D bounding box لازم نتأكد انه مكان ال
ولازم نتأكد من حجم ال bounding box

3 metrices.
3 losses

7. Classification and Localization

network with softmax classification + regression → 2 out-puts

```
base_model = keras.applications.xception.Xception(
    weights="imagenet", include_top=False)
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
class_output = keras.layers.Dense(n_classes, activation="softmax")(avg)
loc_output = keras.layers.Dense(4)(avg)
model = keras.Model(inputs=base_model.input,
    outputs=[class_output, loc_output])

model.compile(loss=["sparse_categorical_crossentropy "mse"],
    loss_weights=[0.8, 0.2],
    optimizer=optimizer, metrics=["accuracy"])
```

→ 4 neurons

يرجعوا احداثيات ال bounding box

Scanned with CamScanner

Outline

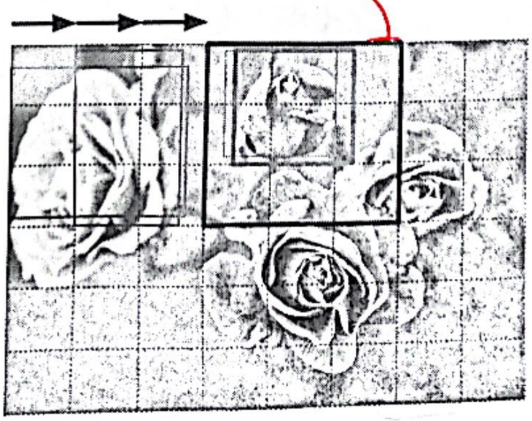
كل object شو هو ووين موجود .

8. Object detection

لما يكون عندي اكثر من object بنفس الصورة

- The task of **classifying and localizing** multiple objects in an image.
- A **slow** approach is use a CNN trained to classify and locate a single object, then **slide** it across the image.

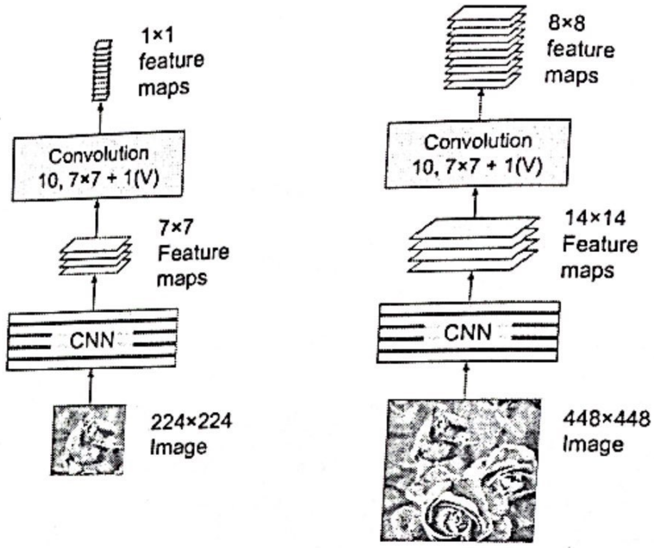
هاد خط عريض ← الاعلى Confidence



* بطلعلي اكثر من banding box
بضلي اللي اله
اعلى Confidence
اللي بكون خطه أعرض

8.1 Fully Convolutional Networks

- FCN has also a **convolution layer at the output** with valid padding. ما بحط zeros حوالين ال map
- FCN can process images of **any size.**
- **Example:**
  - Train the CNN for classification and localization on small images, 10 outputs.
  - For larger image, it output 8 × 8 grid where each cell contains 10 numbers.



من ال models المعروفة ال object detection

there are eight versions from Yolo

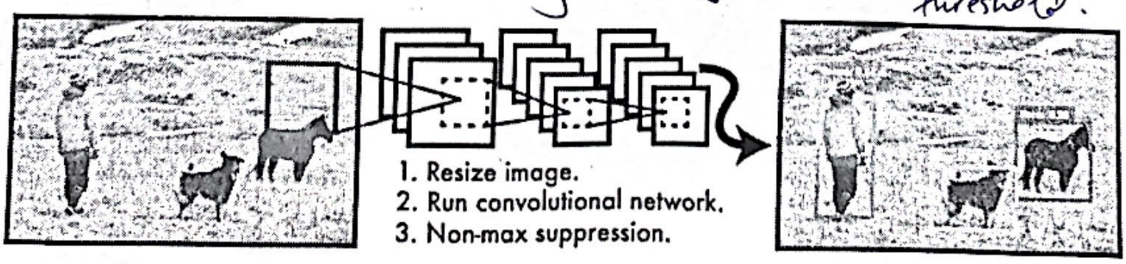
8.2 You Only Look Once (YOLO)

↳ you only look once.

- **YOLO** is an extremely fast and accurate object detection architecture.
  1. Resizes the input image to 448 × 448
  2. Runs a single convolutional network on the image
  3. Thresholds the resulting detections by the model's confidence.

- Sliding window

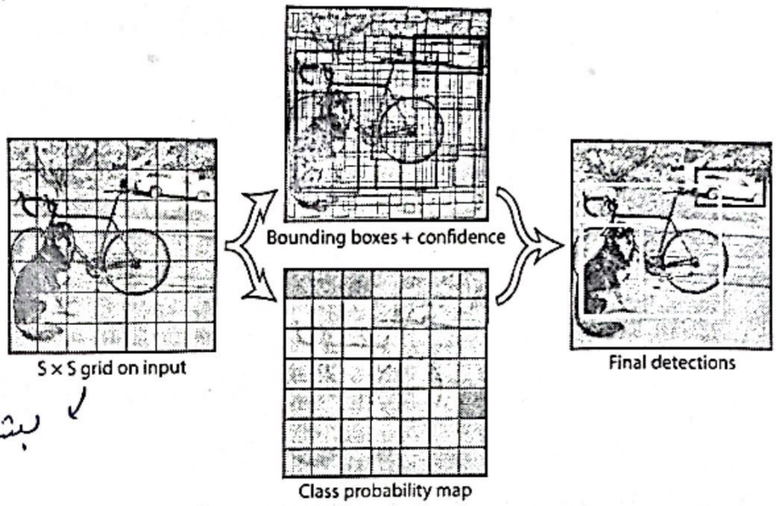
confidence threshold. ممكن نستخدم.



8.2 You Only Look Once (YOLO)

- Models detection as a regression problem. It divides the image into an $S \times S$ grid.

- For each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities.



بيشتغل كأنه sliding window
بتقسم الصورة لمربعات
وبشوف تفاصيل كل مربع.

Outline

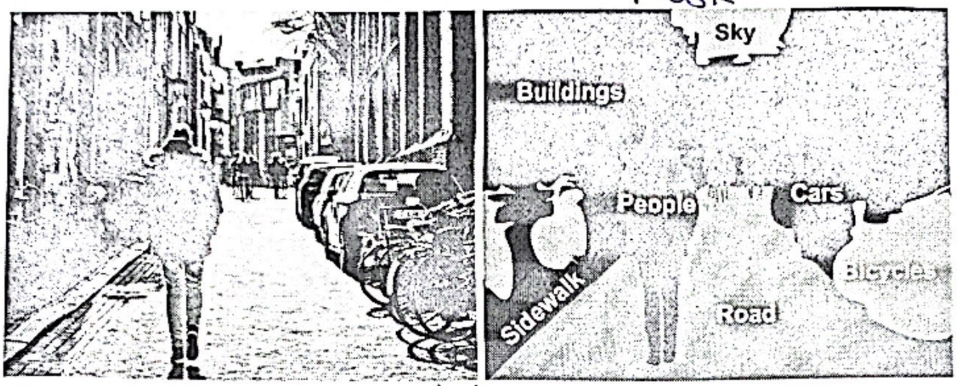
كل Pixel بالصورة لشو بتبع

9. Semantic Segmentation

ما بحكيلي وين البني ادم ← بيعطيني الحدود تبعته

- Each pixel is classified according to the class of the object it belongs to. بعطيه مع الصورة ماسك
- Can use **FCN** followed by up **sampling** layers.

Mask



supervised learning

. instant segmentation → object باخذ كل
بعطينا حدوده وشو هو

more complex

Exercises

14.9. Build your own CNN from scratch and try to achieve the highest possible accuracy on **MNIST**.

14.10. Use **transfer learning** for large image classification, going through these steps:

a) Create a training set containing at least 100 images per class. For example, you could classify your own pictures based on the location (beach, mountain, city, etc.), or alternatively you can use an existing dataset (e.g., from TensorFlow Datasets).

b) Split it into a training set, a validation set, and a test set.

c) Build the input pipeline, including the appropriate preprocessing operations, and optionally add data augmentation.

d) Fine-tune a pretrained model on this dataset.

Good Luck!