Machine Learning

Lecture 1:

* machine = computer
* Learning = improving performance

input →
output →

→ program

⇒ A computer program improves its performance on a given task with experience

[1] Task:

a. Classification → map input into one of a set of classes [discrete]

b. Regression → map input into continuous output

c. Transcription → optical character recognition [OCR]

d. Machine translation

[2] Experience ⇒ what is the data (examples) that the program is using to improve performance?

[3] Performance measure ⇒ How is the performance evaluated?

a. Accuracy: produces the correct output [maximize]

b. Error rate: produces an incorrect output [minimize]

* Supervised Learning: ⟨ classification / regression

⇒ We are given a dataset and already know what our correct output should look like

* unsupervised learning:

⇒ Little or no idea what our results should look like / no feedback

	Supervised	unsupervised
Discrete	classification	clustering
continuous	regression	Dimensionality reduction

* Semi-supervised learning

↳ mix of supervised & unsupervised

بيطلع الى مش معلوم من الجزء المعلوم

* Reinforcement learning

↳ Model learns from a series of actions by maximizing a reward function

* Machine learning depends on a number of algorithms.

{ Computing resources available
  nature of data

Lecture2

* Linear regression

- $h(x) = \theta_0 + \theta_1 x$ → feature ⟹ hypotesis function

  (θ_0: y-intercept, θ_1: slope)

- $\frac{1}{2}\sum_{i=1}^{m} (Y_i - h\theta(x_i))^2 = \sum_{i=1}^{m} \hat{\varepsilon}_i^2$ ⟹ Least squares

  (Y_i: actual, $h\theta(x_i)$: predicted)

  ↳ sum of the squared difference

  كل ما ال difference يكون اصغر كل ع يبقى افضل

$$\frac{1}{2m}\sum_{i}^{m} (h_\theta(x^i) - y^i)^2$$

$$= \frac{1}{2m}\sum_{i}^{m} (\theta_0 + \theta_1 x^i - y^i)^2$$

- $J(\theta_1) = \frac{1}{2m}\sum_{i}^{m} (\theta_1 x^i - y^i)^2$ ⟹ cost function

at $\theta_1 = 2$ = $\frac{1}{2\times 3}$ (no. of inputs)

$$\begin{aligned} &(2\times 1 - 1)^2 = 1^2 \\ +\ &(2\times 2 - 2)^2 = 2^2 \\ +\ &(2\times 3 - 3)^2 = 3^2 \end{aligned} \quad = \frac{14}{6} = 2.33$$

(θ_1, x_1, y^i)

- $j=0: \frac{d}{d\theta_0} j(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x_i) - y_i)$

  $j=1: \frac{d}{d\theta_1} j(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x_i) - y_i)\cdot x_i$ ⟹ partial direvative

repeat until convergence

{ new = old − learning rate

$$\theta_0 = \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$ ⟹ gradient decent

$$\theta_1 = \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\cdot x^{(i)}$$

* provides <u>continuous</u> output

* How to find the best value of θ_1?

→ plotting: nopractical specially in high dimensions

→ analytical solution: not applicable for large datasets

→ numerical solution ex: gradient descent

→ underfitting gamed

• Lecture 3

* Linear regression with multiple values

$\Longrightarrow h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

↳ the weights used by the model indicates the effect of each descriptive feature on the prediction returned by model.

$\Longrightarrow$ Same gradient descent equations

① feature scaling: make sure features are on a similar scale

⇝ $0 \leq x_i \leq 1$ [Mean normalization] $\quad x_1 = \dfrac{x_1 - \text{average}}{\text{Range}}$ — Range — اكبر قيمة $\quad$ or $x_1 = \dfrac{x}{\text{اكبر قيمة}}$

$x_0 = 1$ [always]

② Learning rate [0.001 : 1]

α too small ⇝ slow convergence

α too large ⇝ may fail to converge = divergence

$\Longrightarrow$ if gradient is working properly then $J(\theta)$ should decrease after each iteration

③ polynomial regression

* overfitting:

⇝ models the training data too well.

⇝ learns the detail & noise in the training data to the extent that it negatively impacts the performance of the model on new data.

⇝ Ex: Decision tree

* underfitting:

⇝ poor performance

⇝ Ex: Linear regression

* Linear → minimize error

Logistics → maximize probability (likehood)

* gradient descent: $\theta_j = \theta_j + \alpha \sum_{i=1}^{m} \left(y_i - \dfrac{1}{1+e^{-\theta x_i}}\right) \cdot x_{ij}$

Lecture 4

* Logistic regression

p: probability of an event occurring

1-p: probability of an event not occurring

odds $= \dfrac{p_i}{1-p_i} \Longrightarrow \log\left(\dfrac{p_i}{1-p_i}\right) = \sum_j x_{ij} = x_i\theta$ ← remove range restriction

$\dfrac{p_i}{1-p_i} = e^{x_i\theta}$

$p_i = (1-p_i)e^{x_i\theta}$

$p_i = e^{x_i\theta} - p_i e^{x_i\theta}$

$p_i + p_i e^{x_i\theta} = e^{x_i\theta}$

$p_i(1+e^{x_i\theta}) = e^{x_i\theta}$

$p_i = \dfrac{e^{x_i\theta}}{1+e^{x_i\theta}} = \dfrac{1}{\frac{1}{e^{x_i\theta}}+1} = \dfrac{1}{e^{-x_i\theta}+1}$

Sigmoid [0 or 1]

* after what we did in linear regression:

$P(y_i \mid x;\theta) = \left(\underbrace{\dfrac{1}{1+e^{-\theta x}}}_{\geq 0.5}\right)^{y_i} \left(\underbrace{1-\dfrac{1}{1+e^{-\theta x}}}_{<0.5}\right)^{1-y_i}$

↳ for all observations:

$\dfrac{d}{d\theta} J(\theta) = \dfrac{d}{d\theta} \sum_{1}^{m} \left[y_i \log\left(\dfrac{1}{1+e^{-\theta^t x}}\right) + (1-y_i)\log\left(1-\dfrac{1}{1+e^{-\theta^t x}}\right)\right]$

$= \sum_{i=1}^{m} \left(y_i - \dfrac{1}{1+e^{-\theta^t x_i}}\right) \cdot x_i$

maximum likehood estimation

MLE

* provide discrete output

Lecture 5

* KNN ↳ smallest k
  - classification & regression algorithm ↳ average of k
  - Lazy learner
  - Supervised

⇒ **Eager learner**

* long time learning
  less time classifing
* Linear Regression
  Decision tree

Lazy learner

* less time learning
  more time classifing
* K. Nearest neighbors

* Euclidean distance $= \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$

⇒ How to choose k?

1. odd k value for the 2 classes
2. k must not be a multiple of the number of classes
3. k too small → overfitting [noise]
4. k too big → mis- classify
5. Learn k by cross- validation

Advantages:

1. Effective
2. No linear
3. can add data seamlessly

Disadvantages

1. Sensitive to noise
2. Large memory requirements

Lecture 6.

* Decision tree < supervised / classification

→ Entropy: $H(S) = -P_{(+)} \log_2 P_{(+)} - P_{(-)} \log_2 P_{(-)}$ [To measure impurity]

  if equal 1 ⇝ equal number of +ve & -ve examples

  if equal 0 ⇝ Same class

  Entropy ↓ purity ↑

→ Information gain [tells us how important a given attribute of the feature vector is]

  Information gain = entropy (parent) − [average entropy (children)]

  $$Gain(S,A) = H(S) - \frac{probability}{total} * H(S_{\alpha}) - \frac{probability}{total} * H(S_{\alpha_1})$$

→ overfitting gained

1. Stop growing the tree
2. pre-pruning: stop the algorithm before it becomes a fully grown tree
3. post-pruning: trim the nodes of the decision tree / validation

Scanned with CamScanner

Lecture 7

* Naive Bayes
  - classification
  - independent assumptions
  - useful for very large datasets

* MAP rule [Discrete values]

→ $P(Yes|x')$ } هنحسب دول ⟹ الاكبر هيكون
$P(No|x')$ هو اللي تم اختياره

* Continuous-valued features:

$$\hat{P}(x_j|c_i) = \frac{1}{\sqrt{2\pi}\,\sigma_{ji}} \left(- \frac{(x_j - \mu_{ji})^2}{2\sigma^2_{ji}} \right)$$

$$= \frac{1}{\sqrt{2\pi} \times \text{varience}} \exp\left(- \frac{(x - \text{mean})^2}{2 \times \text{varience}} \right)$$

* Zero conditional probability ⇝ if no example contains the feature value
- m-estimate

$$\hat{P}(a_{jk}|c_i) = \frac{n_c + mp}{n + m}$$

n_c: no. of samples of a specific feature
n: no. of samples of yes/no
p: values of specific feature
m: 1

* parametric methods summarize data into a fixed number of parameters whose count does not increase as the number of data points increase.

↳ Ex: Linear regression (single & multivariate)
Logistic regression
Naive bayes

* Non parametric methods do not summarize data into a fixed number of parameters

↳ Ex: knn / Desicion tree

* As model complexity increases, bias ↓ & variance ↑

* ML: maximize the data likehood given the model, $\arg\max_w P(Data|w)$ ↳ Logistic
  - maximum likehood estimation

MAP: $\arg\max_w P(w|Data)$
  - maximum a posterior

* for small training sets, naive bayes generally is more accurate that logistic regression

Lecture 8:

- SVM for linearly separable binary set
- used for classification and regression
- SVM decides that the best separating line is the line that bisects and is perpendicular to the connecting line.

* Margin in terms of ω

$\omega . x_2 + b = 1$

$\omega . x_1 + b = -1$

$\omega . x_2 + b - (\omega . x_1 + b) = 1 - (-1)$

$\omega . x_2 + \cancel{b} - \omega . x_1 - \cancel{b} = 2$

$\omega (x_2 - x_1) = 2$ [not yet perpendicular]

$\hookrightarrow \frac{\omega}{|\omega|}(x_2 - x_1) = \frac{2}{|\omega|}$ [perpendicular]

$\hookrightarrow$ unit vector

* SVM as a minimization problem:

- Maximizing $\frac{2}{|\omega|}$ = minimizing $\frac{|\omega|}{2}$
- min: $\frac{1}{2}|\omega|^2$ $\Rightarrow$ quadratic problem

$y_i(\omega x_i + b) \geq 1$ $\Rightarrow$ Linear constrain

$\hookrightarrow$ if $y = 1 \Rightarrow \omega x + b = 1$

$y = -1 \Rightarrow \omega x + b = -1$

* Lagarange multiplier:

$L_p \equiv \frac{1}{2}|\omega|^2 - \alpha[y_i(x_i \cdot \omega + b) - 1 \ \forall i]$

$\equiv \frac{1}{2}|\omega|^2 - \sum_{i=1}^{L} \alpha_i [y_i(x_i \cdot \omega + b) - 1]$

$\equiv \frac{1}{2}|\omega|^2 - \sum_{i=1}^{L} \alpha_i y_i (x_i \cdot \omega + b) + \sum_{i=1}^{L} \alpha_i$

$\frac{\partial L_p}{\partial \omega} = 0 \Rightarrow \omega = \boxed{\sum_{i=1}^{L} \alpha_i y_i x_i}$

$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = \boxed{0}$

Lecture 9

* properties of artifical neural network:
  - Learning from experience
  - Adaptivity
  - Non-Linearity
  - Fault tolerance
* processing of ANN:
  (1) Network topology:
  - Single layer network: input layer is fully connected to the output layer [no hidden layer]
  - Multi layer network: one or more layers between input and the output layer.

  (2) Adjustments of weights or learning $\Rightarrow$ modifying weights

  (3) Activation functions
  - Binary: $f(x)=\begin{cases} 0 & \text{for } x<0 \\ 1 & \text{for } x\geq 0 \end{cases}$
  - Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$ [0,1]
  - tanh: $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$ [-1,1]
  - ReLU: $f(x) = \begin{cases} 0 & \text{for } x<0 \\ x & \text{for } x\geq 0 \end{cases}$
  - Softmax: $f(x) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$ for $i=1,\dots J$

* weight adaptation:

[1] $w(n+1) = \underbrace{w(n)}_{\text{old weight}} + \eta\,[\overbrace{d(n)}^{\text{desired}} - \underbrace{Y(n)}_{\text{predicted}}]\,\overbrace{x(n)}^{\text{inputs}}$

[2] |prediction error| $\rightarrow$ |prediction output| $\rightarrow$ |SOP| $\rightarrow$ |input weights|

$E = \frac{1}{2}(d-y)^2$ $\quad f(s) = \frac{1}{1+e^{-s}}$ $\quad S = \sum_{j}^{m} x_i w_{ji} + b_i$ $\quad w_1, w_2$

$\rightsquigarrow$ chain rule: $\overset{①}{\frac{\partial E}{\partial Y}} \times \overset{②}{\frac{\partial Y}{\partial S}} \times \overset{③}{\frac{\partial S}{\partial w_i}} = \overset{④}{\frac{\partial E}{\partial w_i}}$

① $\frac{\partial E}{\partial Y} = y-d$

② $\frac{\partial Y}{\partial S} = \frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right)$

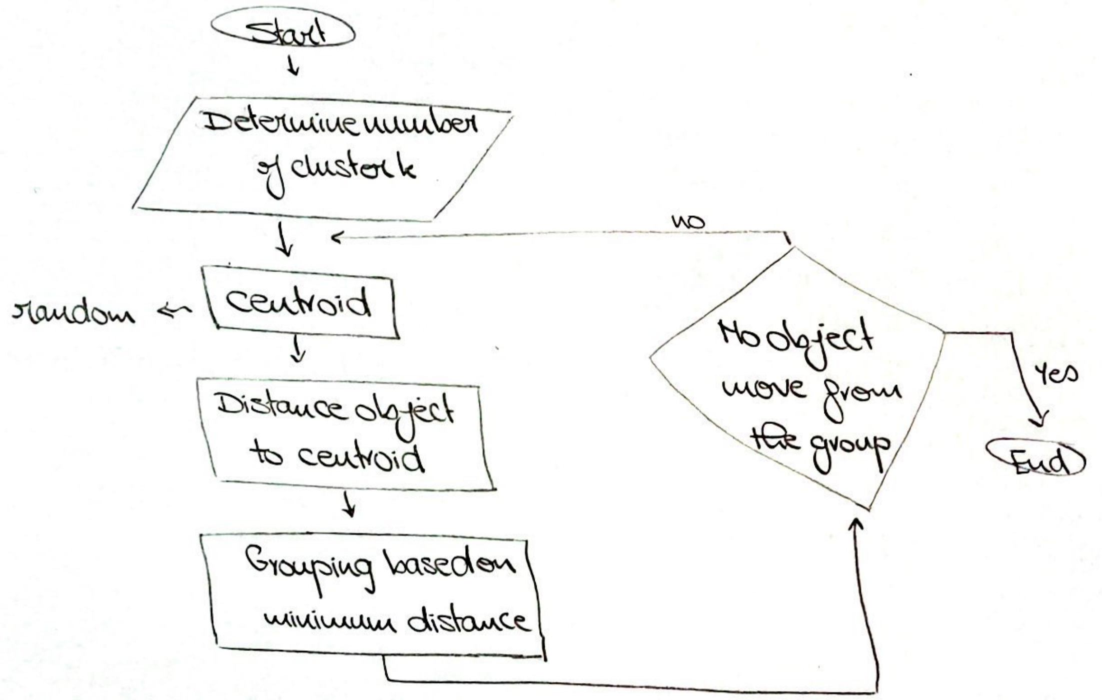
③ $\frac{\partial E}{\partial w_1} = x_1$ and $\frac{\partial E}{\partial w_2} = x_2$

④ $\frac{\partial E}{\partial w_i} = (y-d)\frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right)x_i$

$\hookrightarrow w_{i\,new} = w_{i\,old} - \eta * \frac{\partial E}{\partial w_i}$

Lecture 10

* cluster analysis is like classification, but the class lable of each object is not known



* complexity: $O(n * k * I * d)$
  - n → no. of points
  - k → no. of clusters
  - I → no. of iterations
  - d → no. of attributes

* complexity of each round: $O(n*k)$

* **Strength**
  ~> relatively efficient

* **Weakness**
  ~> applicable only with numerical values
  ~> need to specify k
  ~> Sensitive to noise data

Lecture 11

* Two main types of hierarchical clustering:
  - Agglomerative — merge
    ↳ Bottom-up approach [from 1 → N]
  - Divisive — split
    ↳ Top-down approach [from N → 1]

* Distance calculations:
  - Single-linkage clustering → min
  - complete-linkage clustering → max
  - Average linkage
  - centroid [distance between two centers]

* complexity:
  - Space: $O(n^2)$
  - time: $O(n^3)$ or $O(n^2 \log(n))$

* Strength:
  - no need to specify the number of clusters
  - Easy to implement
  - Dendrogram is very useful

* Weakness:
  - can never undo any previous steps
  - Time complexity is large
  - Difficult with large dataset

Lecture 12

• Less number of features:
  interpretability ↑ accuracy ↓
• More number of features:
  interpretability ↓ accuracy ↑

* performance of ML model:
  ① choice of algorithm
  ② feature selection
  ③ feature creation
  ④ Model selection

* Feature selection methods:

① Filter:
  - calculate the feature relevance score and remove low-scoring features
  - Scalable, simple and fast. ✓
  - each feature is considered separate ×

② Wrapper:
  - Generate and evaluate various subsets of features. Based on accuracy
  - feature dependency ✓
  - very computationally intensive
  - Ex: forward selection / backward elimination

③ Embedded:
  - While the model is being created, it learns which is the best contribute to the accuracy
  - Lasso: absolute value
  - Ridge: Square

* precision: how many selected items relevant?
Recall: how many relevant items selected?

$$F\text{-measure} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{precision} + \text{Recall})}$$