COURS M2
MACHINE LEARNING
(2018 - 2019)

Chapitre 1
Introduction

I) Various types of machine learning

Predictive or supervised learning.

We want to predict an output y, knowing an input x.
We have access to a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$
(N = number of training examples)

ex: x can be a complex object, such as the photo of a cat/dog, y should be 'cat' or 'dog' according to the nature of the pictured animal

Descriptive or unsupervised learning

Inputs : $\mathcal{D} = \{x_i\}_{i=1}^{N}$ - the goal is to find "interesting pattern in the data". This is not a well defined problem

Reinforcement learning (out of the scope of this course)

ex: learn how to behave through reward and punishment
(ex: goldfish jumping through a loop)

II) Supervised learning

The goal is to learn, from input x to outputs y ($y \in \{1, 2, \dots, C\}$) with C being the number of classes.

- If $C = 2$, this is called binary classification.
- If $C > 2$, this is called multiclass classification.

Formalize the problem: assume $y = f(x)$ for some unknown function f

→ the goal is to learn f

and so make prediction on novel inputs

example: two classes of objects (corresponding to labels 0 and 1)

Color	Shape	Size	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

Here come: a blue crescent, a yellow circle, a blue arrow

We are required to generalize the training set.

Blue crescent → 1 (all blue shapes are labeled 1)

Yellow circle, blue arrow → ? (the correct answer is not clear)

the need for probabilistic predictions

We compute $\mathbb{P}(y = c \mid x, \mathcal{D})$ $\quad c \in C$

(x: input, $\mathcal{D}$: training set)

We can give our "best guess": $\hat{y} = \underset{1 \le c \le C}{\operatorname{argmax}}\, \mathbb{P}(y = c \mid x, \mathcal{D})$

Sometimes, it is better to say "I do not know" (in risk-averse settings such as medicine and finance).

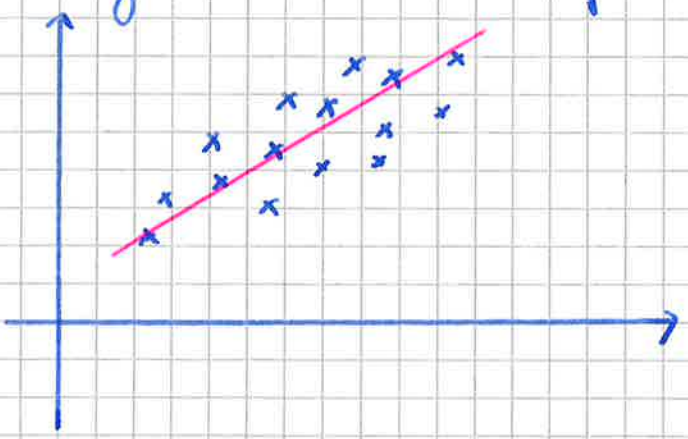
Real world applications

* Document classification: classify a document into G classes
  → special case: spam filter
* Image classification and handwriting recognition
  We might want to classify images as indoor/outdoor, landscape/portrait, dog/no dog ...
  (This is called image classification)
  In the special case where the image consists of isolated handwritten letters and digits, we use classification to perform handwriting recognition.
  A standard dataset used in this area is MNIST ("Modified National Institute of Standards")
  → each image is size 28×28 and each pixel has grayscale value in $\{0, 1, \dots, 255\}$

Regression

Regression is just like classification except the response variable is continuous.

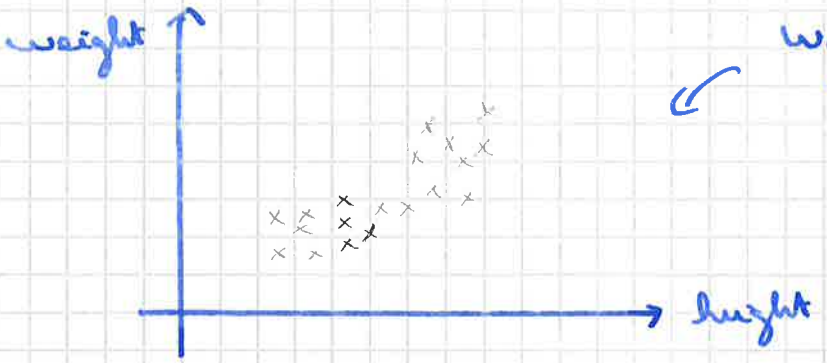
ex: single real-valued input x_i.
single real-valued response y_i:



- predict tomorrow stock-market given current market condition
- predict the t° at any location of a building using weather data, door sensors, etc

III) Unsupervised learning

1) Discovering clusters



Let k be the number of clusters. Our first goal is to approximate the distribution over the number of clusters, that is:

$\mathbb{P}(k=k \mid \mathcal{D})$ (for each k)

Let us set $k^* = \underset{k}{\operatorname{argmax}}\ \mathbb{P}(k=k \mid \mathcal{D})$.

In the supervised case we are told there are two clusters (male and female). But in the unsupervised case, we are free to choose k.

Our second goal is to estimate which cluster each point belongs to. Let $r_i \in \{1, \dots, k\}$ represent the cluster to which data point i is assigned. r_i is an example of a hidden or latent variable since it is never observed in the training set. We can infer which cluster each datapoint belongs to by computing $r_i^* = \underset{k}{\operatorname{argmax}}\ \mathbb{P}(r_i=k \mid x_i, \mathcal{D})$

2) Discovering latent factors

When dealing with high-dim. data, we can reduce the dim. by projecting the data to a lower dimensional subspace. This is called dimensionality reduction.

The motivation behind this technique is that, although the data may appear high-dim., there may only be a small number of degrees of variability, corresponding to latent factors.

Ex: face images, there are a few underlying latent factors such as lighting, pose, identity, ...

IV) Some basic concepts in machine learning

1) Parametric vs. non parametric models

model with fixed number of parameters: parametric model

model where number of parameters grow with the amount of training data: non-parametric model

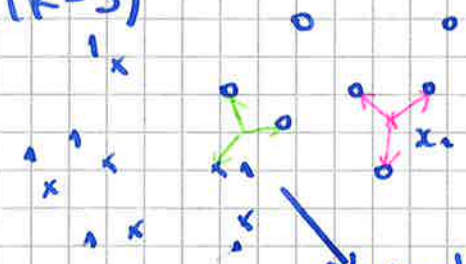
2) A simple non-parametric classifier: K-nearest neighbor (KNN)

We have a training set $\mathcal{D}$. For a new input x, look at the k points in $\mathcal{D}$ nearest to x. Return the empirical fraction: $\mathbb{P}(y=c \mid x, \mathcal{D}, K) = \frac{1}{K} \sum_{i \in N_K(x,\mathcal{D})} \mathbb{1}_{y_i = c}$

$\mathbb{1}_{y_i=c} = 1$ if $y_i = c$, $= 0$ if $y_i \neq c$

this is our prediction — the K nearest neighbors of x in $\mathcal{D}$

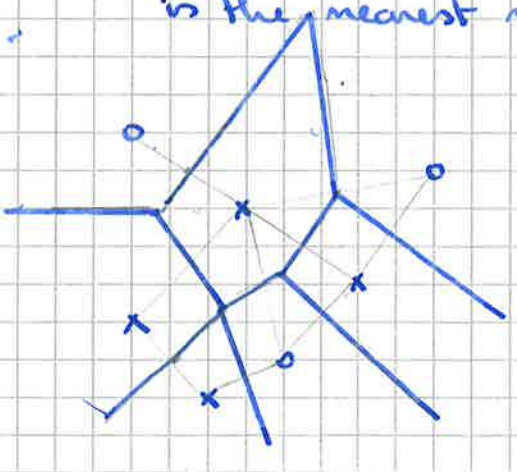
ex: ($K=3$)



→ predict $\mathbb{P}(y=o \mid x_1, \mathcal{D}, k) = 1$

predict $\mathbb{P}(y=o \mid x_1, \mathcal{D}, k) = \frac{2}{3}$

ex: ($K=1$) we look at the nearest neighbor → build a Voronoï tesselation to know which is the nearest neighbor



this method is an example of memory-based learning or instance-based learning. We usually use the euclidean distance (good if data is real-valued), although other metrics can be used.

ex: (p. 17 in the book)

(a) training set
(b) $\mathbb{P}(y=1 \mid x, \mathcal{D})$
(c) $\mathbb{P}(y=2 \mid x, \mathcal{D})$
(d) $g(x) = \text{argmax}_c \, \mathbb{P}(y=c \mid x, \mathcal{D})$

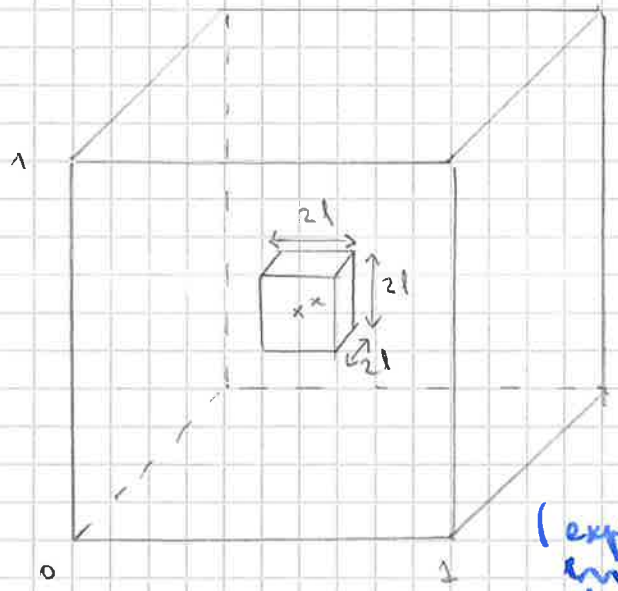
3) The curse of dimensionality

The KNN classifier performance decreases with the dimension (this is known as the curse of dimensionality).

ex: Inputs are uniformly distributed in the d-dimensional unit cube. Suppose we want to have a fraction f of the set $\mathcal{D}$ in a neighborhood to be able to make a prediction.

To make it simple, we use the following distance:

$$d(0, (x_1, \dots, x_d)) = \sup \{x_1, x_2, \dots, x_d\}$$



We look at a cube of side $2l$ centered in x.
(This is a ball of radius l for the above metric)

To get a fraction f of the data into this small cube, we need

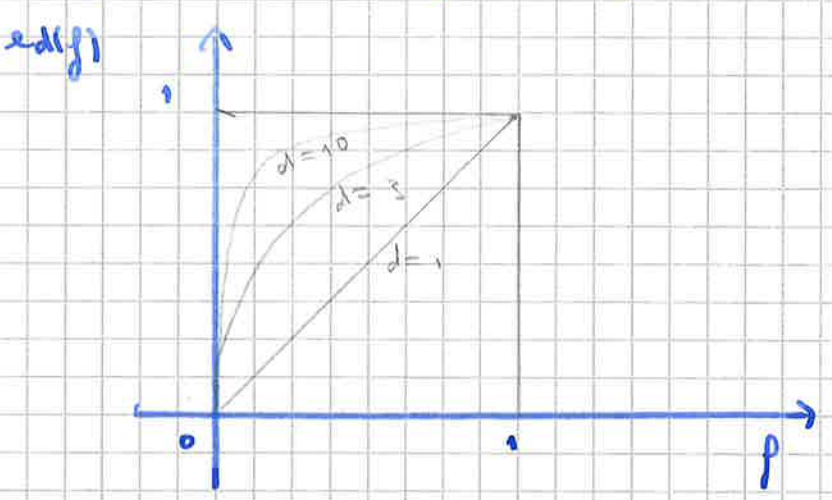
$$\frac{(2l)^d}{1^d} = f$$

(expectation of number of pts in the small cube is actually $(2l)^d \times \#\mathcal{D}$)

So $\ell = f^{1/d} =: e_d(f)$

If $d = 10$ and $f = 10\%$, then $e_d(f) = 0{,}8$.

If $d = 10$ and $f = 1\%$, then $e_d(f) = 0{,}63$.



So, to get a decent portion f of $\mathcal{D}$, we have to look at "neighbours" which are far away ($e_d(f) = 0{,}63$ above), so they might not be good predictors about the input-output function in x.

4) Parametric models for classification and regression

To combat the curse of dimensionality, we can make some assumptions about the nature of the data distribution. These assumptions are known as **inductive bias** and are often embodied in the form of a **parametric model**. We describe two examples below.

a) Linear regression

We suppose the response is a linear function of the inputs. More precisely:

$$y(x) = w^T x + \varepsilon = \sum_{j=1}^{d} w_j x_j + \varepsilon$$

- $\sum_{j=1}^{d} w_j x_j$: scalar product between vector w and vector x
- ε: residual error (often considered to be a random variable and Gaussian)

Linear regression can be made to model non-linear relationship by replacing x with some non-linear function of the inputs ($\phi(x)$).

this is known as basis function expansion. One can take

$$\phi(x) = [1, x, x^2, \dots, x^{d-1}]$$

b) Logistic regression.

We want to generalize linear regression to a case where we do classification (binary classification). i) We suppose

$$\mathbb{P}(y|x,w) = \mathcal{B}(y|\mu(x))$$

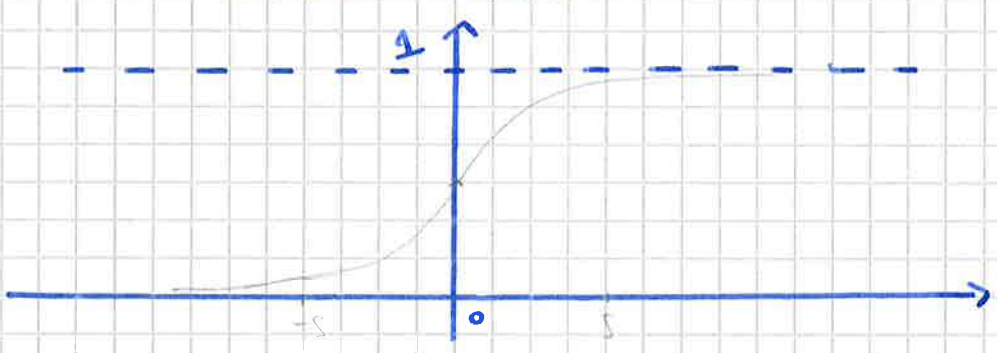
↳ Bernoulli with parameter $\mu(x)$ (probab. of having $\begin{cases} 1 \text{ is } \mu(x) \\ 0 \text{ is } 1-\mu(x) \end{cases}$)

ii) We compute a linear combination of the input and apply a function that ensures $0 \leq \mu(x) \leq 1$ by defining:

$$\mu(x) = \text{sigm}(w^T x)$$

↳ sigmoid function / logistic function / logit function

$$\text{sigm}(\eta) := \frac{1}{1+e^{-\eta}} = \frac{e^{\eta}}{1+e^{\eta}}$$



We get $p(y|x,w) = \text{Ber}(y|\text{sigm}(w^T x))$

this is called logistic regression (although this is a case of classification, not regression!).

example: $\begin{cases} x_i = \text{SAT score of a student} \\ y_i \text{ is whether he passed or failed a class} \\ (1 = \text{pass}, 0 = \text{fail}) \end{cases}$

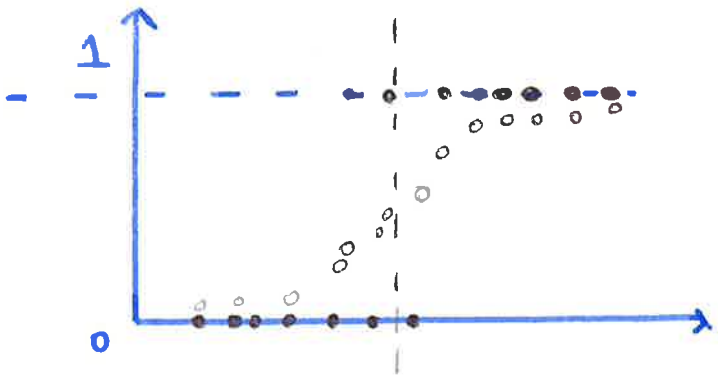
The solid black dots show the training data. From this training data, we get an estimate $\hat{w}$ (see Section 8.3.4 on how to compute $\hat{w}$). the red circles plot $\mathbb{P}(y=1|x_i,\hat{w}) = \text{sigm}(\hat{w}^T x_i)$.

Decision rule: $\hat{y}(x)=1 \iff \mathbb{P}(y=1|x) > 0,5$

↳ this is our prediction

On the Figure, we see that $\text{sigm}(\hat{w}^T x) = 0,5$ for $x \approx 545 = x^*$. We draw a vertical line at $x = x^*$ (= decision boundary). Everything to the left of this line is classified as 0 and everything on the right is classified as 1.

The decision rule has a non-zero error rate even on the training set. This is because the data is not linearly separable, i.e. there is no straight line separating



the 0s from the 1s. Later, we will see examples where we create examples with non-linear decision boundaries (it is possible if the input is multi-dimensional).

NYU 上海 SHANGHAI 纽约大学

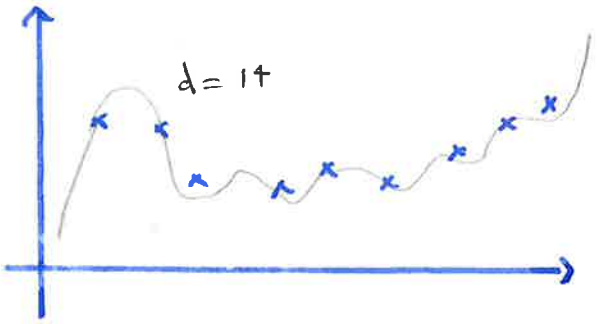
5) Overfitting

We need to be careful that we do not overfit the data, that is we should avoid to model every minor variation of the input, since this is more likely to be noise than true signal.

ex 1: output $y = w^T \phi(x) + \varepsilon$ ($\varepsilon \sim \mathcal{N}(0, \sigma^2)$)

$\phi(x) = [1, x, x^2, \dots, x^d]$

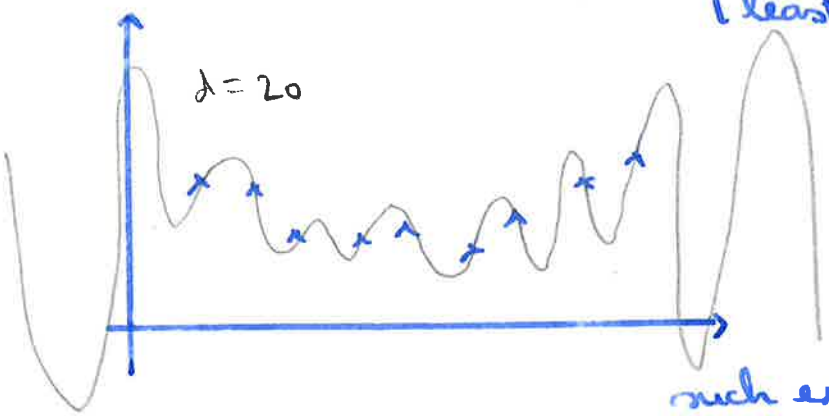
x is 1-dimensional



21 data points

We look for w of size $d+1$ such that $\sum_{i=1}^{21} (w^T \phi(x_i) - y_i)^2$ is minimal

(least-square regression)



↘ It is unlikely that the true function has such extreme oscillations. So using this model might result in bad predictions of future outputs.

NYU 上海 SHANGHAI 纽约大学

ex 2 : the choice of k in the KNN classifier.

K = 1 → the method makes not error on the training set but the resulting prediction zones are very "wiggly", so the method may not work well at predicting future data

K = 5 → smoother prediction zones (smoother borders in fact)

When k increases, the boundaries become smoother until, in the limit K = N, we end up predicting the majority label of the whole data set.

See below how to pick up the "right" value of K.

6) Model selection

We have a variety of models of different complexity :

- linear or logistic regression
- models with different degree polynomials
- KNN classifiers with different values of K

How do we choose ?

Compute the misclassification rate of the training set for each method:

$$\text{err}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(f(x_i) \neq y_i)$$

↳ an classifier

NYU SHANGHAI 上海 纽约大学

We plot this error for a KNN classifier. We see that k increases the error on the training set. We get the minimal error for $k=1$, since this model is just memorizing data.

What we want is the generalization error, which is the expected value of the misclassification rate when averaged over future data. What we can do is use part of the (x_i, y_i) as training data, and use another part as a test set. In this case, when we plot the test error vs. k, we see a U-shaped curve

* for complex models (small k), the method overfits
* for simple models (big k), the method underfits

We pick k such that it minimizes the error on the test set.

or validation set

then, when K is picked, we refit the model to all the available data.

↳ e.g.: when a new x arrives, we look at its k nearest neighbours in all the available data

Rule of thumb: use 80% of the data for the training set and 20% for the validation set.

But: if data is small, this technique runs into problems, because the model won't have enough data to train on, or won't have enough data to make a reliable estimate of the future performance.

Solution: use cross-validation. Split the data into P folds; then, for each fold $p \in \{1, 2, \ldots, P\}$, do the (P parts) training on all the folds except the p'th. Then compute the error averaged over all the folds

It is common to use $P = 5$ (it is called 5-fold CV).

If we set $P = N$, then we get a method called leave-one out cross validation (LOOCV)

Choosing K for a KNN classifier is a special case of a more general problem known as model selection where we have to choose between models with different degrees of flexibility

---

TP: p. 14-24 of the O'Reilly book