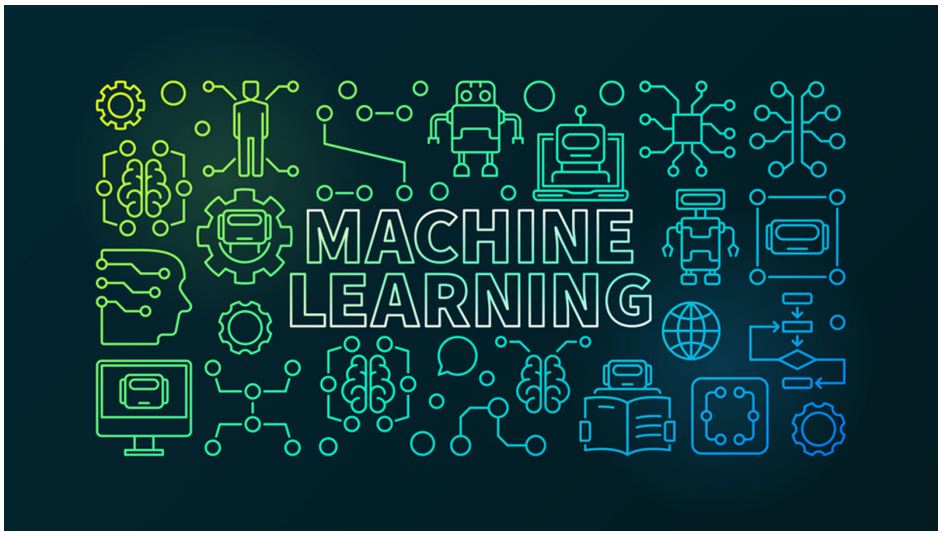
Statistical Methods for Machine Learning

UNIVERSITÀ
DEGLI STUDI
DI MILANO

Data Science and Economics
Università degli Studi di Milano

Andrea Ierardi

Lecture notes



Contents

1 Lecture 1 - 09-03-2020 **6**
- 1.1 Introduction of the course . . . 6
- 1.2 Examples . . . 6
  - 1.2.1 Spam filtering . . . 9

2 Lecture 2 - 10-03-2020 **10**
- 2.1 Argomento . . . 10
- 2.2 Loss . . . 10
  - 2.2.1 Absolute Loss . . . 10
  - 2.2.2 Square Loss . . . 11
  - 2.2.3 Example of information of square loss . . . 12
  - 2.2.4 labels and losses . . . 13
  - 2.2.5 Example TF(idf) documents encoding . . . 15

3 Lecture 3 - 16-03-2020 **17**
- 3.1 Overfitting . . . 19
  - 3.1.1 Noise in the data . . . 19
- 3.2 Underfitting . . . 21
- 3.3 Nearest neighbour . . . 21

4 Lecture 4 - 17-03-2020 **24**
- 4.1 Computing h_{NN} . . . 24
- 4.2 Tree Predictor . . . 26

5 Lecture 5 - 23-03-2020 **30**
- 5.1 Tree Classifier . . . 30
- 5.2 Jensen's inequality . . . 32
- 5.3 Tree Predictor . . . 36
- 5.4 Statistical model for Machine Learning . . . 37

6 Lecture 6 - 24-03-2020 **39**
- 6.1 Bayes Optimal Predictor . . . 39
  - 6.1.1 Square Loss . . . 40
  - 6.1.2 Zero-one loss for binary classification . . . 41
- 6.2 Bayes Risk . . . 44

7 Lecture 7 - 30-03-2020 **46**
- 7.1 Chernoff-Hoffding bound . . . 46
- 7.2 Union Bound . . . 47
- 7.3 Studying overfitting of a ERM . . . 51

8 Lecture 8 - 31-03-2020 **53**
8.1 The problem of estimating risk in practise 54
8.2 Cross-validation 56
8.3 Nested cross validation 58

9 Lecture 9 - 06-04-2020 **59**
9.1 Tree predictors 59
9.1.1 Catalan Number 61

10 Lecture 10 - 07-04-2020 **65**
10.1 TO BE DEFINE 65
10.2 MANCANO 20 MINUTI DI LEZIONE 65
10.3 Compare risk for zero-one loss 67

11 Lecture 11 - 20-04-2020 **69**
11.1 Analysis of K_{NN} 69
11.1.1 Study of K_{NN} 72
11.1.2 study of trees 73
11.2 Non-parametric Algorithms 74
11.2.1 Example of parametric algorithms 75

12 Lecture 12 - 21-04-2020 **76**
12.1 Non parametrics algorithms 76
12.1.1 Theorem: No free lunch 76
12.2 Highly Parametric Learning Algorithm 78
12.2.1 Linear Predictors 78
12.2.2 MinDisagreement 82

13 Lecture 13 - 27-04-2020 **83**
13.1 Linear prediction 83
13.1.1 MinDisOpt 83
13.2 The Perception Algorithm 86
13.2.1 Perception convergence Theorem 87

14 Lecture 14 - 28-04-2020 **90**
14.1 Linear Regression 90
14.1.1 The problem of linear regression 90
14.1.2 Ridge regression 91
14.2 Percetron 92
14.2.1 Online Learning 93
14.2.2 Online Gradiant Descent (OGD) 95

15 Lecture 15 - 04-05-2020 **96**
15.1 Regret analysis of OGD 96
15.1.1 Projected OGD 97

16 Lecture 16 - 05-05-2020 **98**

17 Lecture 17 - 11-05-2020 **99**

18 Lecture 18 - 12-05-2020 **100**
18.1 Kernel functions 100
18.1.1 Feature expansion 100
18.1.2 Kernels implements feature expansion (Efficiently . . . 101
18.2 Gaussian Kernel 102

List of Figures

2.1 Example of domain of K_{NN} . . . 11
2.2 Example of domain of K_{NN} . . . 11

3.1 Example of domain of K_{NN} . . . 17
3.2 Example of domain of K_{NN} . . . 22
3.3 Example of domain of K_{NN} . . . 22

4.1 Example of domain of K_{NN} . . . 25
4.2 Example of domain of K_{NN} . . . 26
4.3 Example of domain of K_{NN} . . . 27
4.4 Example of domain of K_{NN} . . . 28
4.5 Example of domain of K_{NN} . . . 28
4.6 Example of domain of K_{NN} . . . 28
4.7 Example of domain of K_{NN} . . . 29

5.1 Example of domain of K_{NN} . . . 30
5.2 Example of domain of K_{NN} . . . 31
5.3 Example of domain of K_{NN} . . . 32
5.4 Example of domain of K_{NN} . . . 32
5.5 Example of domain of K_{NN} . . . 33
5.6 Example of domain of K_{NN} . . . 34
5.7 Example of domain of K_{NN} . . . 35
5.8 Example of domain of K_{NN} . . . 35
5.9 Example of domain of K_{NN} . . . 35
5.10 Example of domain of K_{NN} . . . 36

6.1 Example of domain of K_{NN} . . . 40
6.2 Example of domain of K_{NN} . . . 42
6.3 Example of domain of K_{NN} . . . 42
6.4 Example of domain of K_{NN} . . . 43
6.5 Example of domain of K_{NN} . . . 44

7.1 Example . . . 47
7.2 Example . . . 48
7.3 Example . . . 48
7.4 Example . . . 49
7.5 Example . . . 49
7.6 Draw of how $\hat{h}$, h^* and f^* are represented . . . 50

8.1 Representation of $\hat{h}$, h^* and f^* . . . 53
8.2 Example . . . 54

8.3 Splitting test and training set . . . 56
8.4 K-folds . . . 57
8.5 Nested Cross Validation . . . 58

9.1 Tree building . . . 59
9.2 Tree with at most N node . . . 60
9.3 Algorithm for tree predictors . . . 63

10.1 Point (2) - where $y = cx + q$ $y = -cx + q$ . . . 67
10.2 Point . . . 68

11.1 Example of domain of K_{NN} . . . 69
11.2 Diagonal length . . . 70
11.3 Shape of the function . . . 71
11.4 Parametric and non parametric growing as training set getting larger . . . 75

12.1 Tree building . . . 77
12.2 Dot product . . . 78
12.3 Dot product . . . 79
12.4 Hyperplane . . . 79
12.5 Hyperplane . . . 80
12.6 Hyperplane . . . 80
12.7 Example of one dimensional hyperplane . . . 81

13.1 Tree building . . . 84
13.2 Tree building . . . 84
13.3 Tree building . . . 85
13.4 Feasibilty problem . . . 85
13.5 . . . 86
13.6 . . . 87

14.1 . . . 91
14.2 . . . 93
14.3 . . . 94
14.4 . . . 95

15.1 . . . 97

18.1 . . . 100
18.2 . . . 103
18.3 . . . 103

Lecture 1 - 09-03-2020

1.1 Introduction of the course

In this course we look at the principle behind design of Machine learning. Not just coding but have an idea of algorithm that can work with the data.

We have to fix a mathematical framework: some statistic and mathematics.
Work on ML on a higher level
ML is data inference: make prediction about the future using data about the past

- Clustering → grouping according to similarity
- Planning → (robot to learn to interact in a certain environment)
- Classification → (assign meaning to data) example: Spam filtering
  I want to predict the outcome of this individual or i want to predict whether a person click or not in a certain advertisement.

1.2 Examples

Classify data into categories:

- Medical diagnosis: data are medical records and categories are diseases
- Document analysis: data are texts and categories are topics
- Image analysts: data are digital images and for categories name of objects in the image (but could be different).
- Spam filtering: data are emails, categories are spam vs non spam.
- Advertising prediction: data are features of web site visitors and categories could be click/non click on banners.

Classification : **Different from clustering** since we do not have semantically classification (spam or not spam) → like meaning of the image.
I have a semantic label.

Clustering: i want to group data with similarity function.

Planning: Learning what to do next

Clustering: Learn similarity function

Classification: Learn semantic labels meaning of data

Planning: Learn actions given state

In classification is an easier than planning task since I'm able to make prediction telling what is the semantic label that goes with data points.
If i can do classification i can clustering.
If you do planning you probably classify (since you understanding meaning in your position) and then you can also do clustering probably.
We will focus on classification because many tasks are about classification.

Classify data in categories we can image a set of categories.
For instance the tasks:
'predict income of a person'
'Predict tomorrow price for a stock'
The label is a number and not an abstract thing.

We can distinguish two cases:

- The label set $\rightarrow$ set of possible categories for each data point. For each of this could be finite set of abstract symbols (case of document classification, medical diagnosis). So the task is classification.
- Real number (no bound on how many of them). My prediction will be a real number and is not a category. In this case we talk about a task of regression.

Classification: task we want to give a label predefined point in abstract categories (like YES or NO)
Regression: task we want to give label to data points but this label are numbers.

When we say prediction task: used both for classification and regression tasks.
Supervised learning: Label attached to data (classification, regression)
Unsupervised learning: No labels attached to data (clustering)

In unsupervised the mathematical modelling and way algorithm are score and can learn from mistakes is a little bit harder. Problem of clustering is harder to model mathematically.
You can cast planning as supervised learning: i can show the robot which is the right action to do in that state. But that depends on planning task is formalised.
Planning is higher level of learning since include task of supervised and unsupervised learning.

Why is this important ?
Algorithm has to know how to given the label.
In ML we want to teach the algorithm to perform prediction correctly. Initially algorithm will make mistakes in classifying data. We want to tell algorithm that classification was wrong and just want to perform a score. Like giving a grade to the algorithm to understand if it did bad or really bad. So we have mistakes!

Algorithm predicts and something makes a mistake $\rightarrow$ we can correct it.
Then algorithm can be more precisely. We have to define this mistake.
Mistakes in case of classification:

- If category is the wrong one (in the simple case). We have a binary signal where we know that category is wrong.

How to communicate it?
We can use the loss function: we can tell the algorithm whether is wrong or not.

Loss function: measure discrepancy between 'true' label and predicted label.
So we may assume that every datapoint has a true label. If we have a set of topic this is the true topic that document is talking about. It is typical in supervised learning.

How good the algorithm did?

$$\ell(y, \hat{y}) \leq 0$$

were y is true label and $\hat{y}$ is predicted label

We want to build a spam filter were 0 is not spam and 1 is spam and that's a Classification task:

$$\ell(y, \hat{y} = \begin{cases} 0, & \text{if } \hat{y} = y \\ 1, & \text{if } \hat{y} \neq y \end{cases}$$

The loss function is the "interface" between algorithm and data.
So algorithm know about the data through the loss function.
If we give a useless loss function the algorithm will not perform good: is important to have a good loss function.

1.2.1 Spam filtering

$Y = \{spam, no\, spam\}$
Binary classification $|Y| = 2$
We have two main mistake:

- False positive: y =non spam, $\hat{y}$ = spam
- False negative: y =spam, $\hat{y}$ = no spam

It is the same mistake? No if i have important email and you classify as spam that's bad and if you show me a spam than it's ok.
So we have to assign a different weight.

$$\ell\left(y, \hat{y}\right) = \begin{cases} 2 & if\ FP \\ 1 & if\ FN \\ 0 & otherwise \end{cases}$$

We have to take more attention on positive mistake
Even in binary classification, mistakes are not equal.

Lecture 2 - 10-03-2020

2.1 Argomento

Classification tasks
Semantic label space Y
Categorization Y finite and
small Regression Y appartiene ad |R
How to predict labels?
Using the lost function → ..
Binary classification
Label space is Y = -1, +1
Zero-one loss

$$\ell(y, \hat{y} = \begin{cases} 0, & \text{if } \hat{y} = y \\ 1, & \text{if } \hat{y} \neq y \end{cases}$$

$FP \quad \hat{y} = 1, \quad y = -1$
$FN \quad \hat{y} = -1, \quad y = 1$

Losses for regression?
y, and $\hat{y} \in \mathbb{R}$,
so they are numbers!
One example of loss is the absolute loss: absolute difference between numbers

2.2 Loss

2.2.1 Absolute Loss

$$\ell(y, \hat{y} = |y - \hat{y}| \Rightarrow absolute \quad loss$$

Some inconvenient properties:

- ...
- Derivative only two values (not much informations)

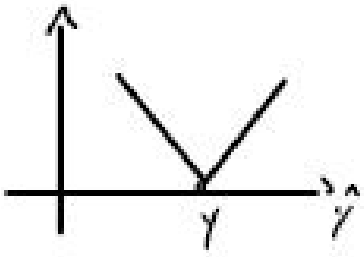


Figure 2.1: Example of domain of K_{NN}

2.2.2 Square Loss

$$\ell(y, \hat{y} = (y - \hat{y})^2 \Rightarrow \textit{square loss}$$

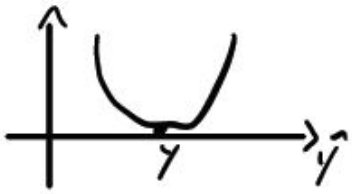


Figure 2.2: Example of domain of K_{NN}

Derivative :

- more informative
- and differentible

Real numbers as label → regression.
Whenever taking difference between two prediction make sense (value are numbers) then we are talking about regression problem.
Classification as categorization when we have small finite set.

2.2.3 Example of information of square loss

$\ell(y, \hat{y}) = (y - \hat{y})^2 = F(y)$
$F'(\hat{y}) = -2 \cdot (y - \hat{y})$

- I'm under sho or over and how much
- How much far away from the truth

$\ell(y, \hat{y}) = |y - \hat{y}| = F(y') \cdot F'(y) = Sign(y - \hat{y})$

Question about the future
Will it rain tomorrow?
We have a label and this is a binary classification problem.
My label space will be Y = "rain", "no rain"
We don't get a binary prediction, we need another space called prediction space (or decision space).
$Z = [0, 1]$
$\hat{y} \in Z$ $\hat{y}$ *is my prediction of rain tomorrow*
$\hat{y} = \mathbb{P}(y = "rain")$ $\rightarrow$ *my guess is tomorrow will rain (not sure)*

$y \in Y$ $\hat{y} \in Z$
quadHow can we manage loss?
Put numbers in our space
$\{1, 0\}$ *where 1 is rain and 0 no rain*

I measure how much I'm far from reality.
So loss behave like this and the punishment is gonna go linearly??

26..

However is pretty annoying. Sometime I prefer to punish more so i going quadratically instead of linearly.
There are other way to punish this.
I called **logarithmic loss**
We are extending a lot the range of our loss function.

$$\ell(y, \hat{y}) = |y - \hat{y}| \in |0, 1| \qquad \ell(y, \hat{y}) = (y - \hat{y})^2 \in |0, 1|$$

If i want to expand the punishment i use logarithmic loss

$$\ell(y, \hat{y} = \begin{cases} ln\frac{1}{\hat{y}}, & \text{if } y = 1 \textit{(rain)} \\ ln\frac{1}{1-\hat{y}}, & \text{if } y = 0 \textit{(no rain} \end{cases}$$

$F(\hat{y}) \to$ *can be 0 if i predict with certainty*
If $\hat{y} = 0.5 \quad \ell(y, \frac{1}{2}) = ln2$ *constant losses in each prediction*

$\lim_{\hat{y}\to 0^+} \ell(1, \hat{y}) = +\infty$
We give a vanishing probability not rain but tomorrow will rain.
So this is $+\infty$
$\lim_{\hat{y}\to 1^-} \ell(0, \hat{y}) = +\infty$

The algorithm will be punish high more the prediction is not real. Algorithm will not get 0 and 1 because for example is impossible to get a perfect prediction.
This loss is useful to give this information to the algorithm.

Now we talk about labels and losses

2.2.4 labels and losses

Data points: they have some semantic labels that denote some true about this data points and we want to predict this labels.
We need to define what data points are: number? Strings? File? Typically they are stored in database records
They can have very precise structure or more homogeneously structured
A data point can be viewed as a vector in some d dimensional real space. So it's a vector of number

$$\mathbb{R}^d X = (x_1, x_2..., x_d) \in \mathbb{R}^c$$

Image can be viewed as a vector of pixel values (grey scale 0-255).
I can use geometry to learn because point are in my Euclidean space. Data can be represented as point in Euclidean space. Images are list of pixel that are pretty much the same range and structure (from 0 to 255). It's very natural to put them in a space.

Assume X can be a record with heterogeneous fields:
For example medical records, we have several values and each fields has his meaning by it's own. (Sex, weight, height, age, zip code)

Each one has a different range, in some cases is numerical but something have like age ..
Does have any sense to see a medical record as a point since coordinates have different meaning.
Fields are not comparable.
This is something that you do: when you want to solve some inference you have to decide which are the label and what is the label space and we have to encode the data points.

Data algorithm expect some homogenous interface. In this case algorithm has to build records with different values of fields.
This is something that we have to pay attention too.
You can always each range of values in number. So ages is number, sex you can give 0 and 1, weight number and zip code is number.
How ever geometry doesn't make sense since I cannot compare this coordinates.
Linear space i can sum up as vector: i can make linear combination of vectors.
Inner product to measure angles! (We will see in linear classifier).

I can scramble the number of my zip code.
So we get problems with sex and zip code

Why do we care about geometry? I can use geometry to learn.
However there is more to that, geometry will carry some semantically information that I'm going to preserve during prediction.
I want to encode my images as vectors in a space. Images with dog.....

PCA doesn't work because assume we encode in linear space.
We hope geometry will help us to predict label correctly and sometimes i hard to convert data into geometry point.
Example of comparable data: images, or documents.
Assume we have documents with corpus (set of documents).
Maybe in English and talk about different thing and different words.
X is a document and i want to encode X into a point fix in bidimensional space.
There is a way to encode a set of documents in point in a fixed dimensional space in such way it make sense this coordinate are comparable.
I can represent fields with [0,1] for Neural network for example. But they have no geometrical meaning

2.2.5 Example TF(idf) documents encoding

TF encoding of docs.

1. Extract where all the words from docs
2. Normalize words (nouns, adjectives, verbs ...)
3. Build a dictionary of normalized words

Doc $x = (x_1, .., x_d)$
I associate a coordinate for each word in a dictionary.
d = number of words in dictionary
I can decide that
$x_i = 1$ *If i-th word of dictionary occurs in doc.*
$x_i = 0$ *Else*

X_i *number of time i-th word occur in doc.*
Longer documents will have higher value of coordinates that are not zero.
Now i can do the TF encoding in which xi = frequency with which i-th word occur in dictionary.
You cannot sum dog and cat but we are considering them frequencies so we are summing frequency of words.
This encoding works well in real words.
I can choose different way of encoding my data and sometime i can encode a real vector

I want

1. A predictor $f : X \longrightarrow Y$ (in weather $X \longrightarrow Z$
2. X is our data space (where points live)
3. $X = \mathbb{R}^d$ images
4. $X = X_1 x ... x X_d$ Medical record
5. $\hat{y} = f(x)$ predictor for X

(x, y)

We want to predict a label that is much closer to our label. How?
Loss function: so this is my setting and is called and example.
Data point together with label is a "example"

We can get collection of example making measurements or asking people. So we can always recover the true label.
We want to replace this process with a predictor (so we don't have to bored a person).
y is the ground truth for x $\rightarrow$ mean reality!
If i want to predict stock for tomorrow, i will wait tomorrow to see the ground truth.

Lecture 3 - 16-03-2020

Data point x represented as sequences of measurement and we called this measurements features or attributes.

$x = (x_1, ..., x_d)$ x_1 *feature valuex* $\in X^d$ $X = \mathbb{R}^d$ $X = X_1 \cdot x \cdot ... \cdot X_d \cdot x$

Label space Y
Predictor $f : X \to Y$

Example (x, y) y is the label associated with x
($\to y$ is the correct label, the ground truth)

Learning with example $(x_1, y_1)...(x_m, y_m)$ *training set*

Training set is a set of examples with every algorithm can learn.....

Learning algorithm take training set as input and produces a predictor as output.

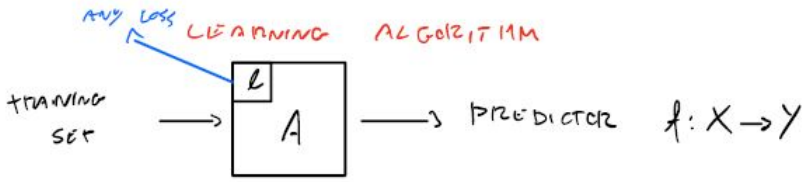


Figure 3.1: Example of domain of K_{NN}

With image recognition we use as measurement pixels.
How do we measure the power of a predictor?
A learning algorithm will look at training set, algorithm and generate the predictor. Now the problem is verify the score.
Now we can consider a test set collection of example

$$\textit{Test set} \qquad (x'_1, y'_1)...(x'_n, y'_n)$$

Typically we collect big dataset and then we split in training set and test set randomly.

Training and test are typically disjoint
How we measure the score of a predictor? We compute the average loss.
The error is the average loss in the element in the test set.

$$\textit{Test error} \qquad \frac{1}{n} \cdot \sum_{t=1}^{n} \ell(f(x'_t), y')$$

In order to simulate we collect the test set and take the average loss of the predictor of the test set. This will give us idea of how the..
Proportion of test and train depends in how big the dataset is in general.
Our **Goal**: A learning algorithm 'A' must output f with a small test error.
A does not have access to the test set. (Test set is not part of input of A).
Now we can think in general on how a learning algorithm should be design.
We have a training set so algorithm can say:
'A' may choose f based on performance on training set.

$$\textit{Training error} \qquad \hat{\ell}(f) = \frac{1}{m} \cdot \sum_{t=1}^{m} \ell(f(x_t), y_t)$$

Given the training set $(x_1, ..., x_m)(y_1, ..., y_m)$
If $\hat{\ell}(f)$ for same f, then test of f is also small
Fix F set of predictors output $\hat{f}$

$$\hat{f} = arg\,min\,\hat{\ell}(f) f \in F$$

This algorithm is called Empirical Risk Minimiser (ERM)
When this strategy (ERM) fails?
ERM may fails if for the given training set there are:
Many $f \in F$ with small $\hat{\ell}(f)$, but not all of them have small test error

There could be many predictor with small error but some of them may have big test error. Predictor with the smallest training error doesn't mean we will have the smallest test error.
I would like to pick f^* such that:

$$f^* = arg\,min \frac{1}{n} \cdot \sum_{t=1}^{m} \ell(f(x'_t), y_t) \qquad f \in F$$

where $\ell(f(x'_t), y_t)$ is the test error
ERM works if $f^* such\ that \qquad f^* = arg\,min\,\hat{\ell}(f) \qquad f \in F$
So minimising training and test????? Check videolecture

We can think of f as finite since we are working on a finite computer.
We want to see why this can happen and we want to formalise a model in which we can avoid this to happen by design: We want when we run ERM choosing a good predictor with ..... PD

3.1 Overfitting

We called this as overfitting: specific situation in which 'A' (where A is the learning algorithm) overfits if f output by A tends to have a training error much smaller than the test error.
A is not doing his job (outputting large test error) this happen because test error is misleading.
Minimising training error doesn't mean minimising test error. Overfitting is bad.
Why this happens?
This happen because we have **noise in the data**

3.1.1 Noise in the data

Noise in the data: y_t is not deterministically associated with x_i.

Could be that datapoint appears more times in the same test set. Same datapoint is repeated actually I'm mislead since training and dataset not coincide. Minimising the training error can take me away from the point that minimise the test error.
Why this is the case?

- Some **human in the loop**: label assigned by people.(Like image contains certain object but human are not objective and people may have different opinion)
- **Lack of information**: in weather prediction i want to predict weather error. Weather is determined by a large complicated system. If i have humidity today is difficult to say for sure that tomorrow will rain.

When data are not noise i should be ok.
Labels are not noisy

Fix test set and trainign set.

$$\exists f^* \in F \qquad y'_t = f^*(x'_t) \qquad \forall (x'_t, y'_t) \quad \textit{in test set}$$

$$y_t = f^+(x_t) \qquad \forall (x_t, y_t) \quad \textit{in training set}$$

Think a problem in which we have 5 data points(vectors) :
$\vec{x_1}, ...\vec{x_5}$ *in some space* X
We have a binary classification problem $Y = \{0, 1\}$
$\{\vec{x_1}, ..., \vec{x_5}\} \in X \qquad Y = \{0, 1\}$

F contains all possible calssifier $2^5 = 32 \qquad f : \{x_1, ..., x_5\} \to \{0, 1\}$

Example					
	x_1	x_2	x_3	x_4	x_5
f	0	0	0	0	0
f'	0	0	0	0	1
f"	..	..	..	..	..

$$\textit{Training set} \quad x_1, x_2, x_3 \quad f^+$$

$$\textit{Test set} \quad x_4, x_5 \quad f^*$$

4 *classifier* $f \in F \qquad$ *will have* $\hat{\ell}(f) = 0$

$(x_1, 0) \quad (x_2, 1) \quad (x_3, 0)$
$(x_4, ?) \quad (x_5, ?)$
$f^*(x_4) \quad f^*(x_5)$
If not noise i will have deterministic data but in this example (worst case) we get problem.
I have 32 classifier to choose: i need a larger training set since i can't distinguish predictor with small and larger training(?) error. So overfitting noisy or can happen with no noisy but few point in the dataset to define which predictor is good.

3.2 Underfitting

'A' underfits when f output by A has training error close to test error but they are both large.
Close error test and training error is good but the are both large.

$A \equiv ERM$, *then A undefits if F is too small* $\rightarrow$ *not containing too much predictors*

In general, given a certain training set size:

- Overfitting when $|F|$ is too large (not enough points in training set)
- Underfitting when $|F|$ is too small

Proportion predictors and training set

$$|F|, \ i \ need \ ln|F| \quad bits \ of \ info \ to \ uniquely \ determine \ f^* \in F$$
$$m >> ln|F| \qquad when \quad |F| < \infty where \ m \ is \ the \ size \ of \ traning \ set$$

3.3 Nearest neighbour

This is completely different from ERM and is one of the first learning algorithm. This exploit the geometry of the data. Assume that our data space X is:
$X \equiv \mathbb{R}^d \qquad x = (x_1, ..., x_d) \qquad y - \{-1, 1\}$
S is the traning set $(x_1, y_1)...(x_m, y_m)$
$x_t \in \mathbb{R}^d \qquad y_t \in \{-1, 1\}$

$d = 2 \rightarrow$ *2-dimensional vector*

where + and - are labels

Point of test set
If i want to predict this point?

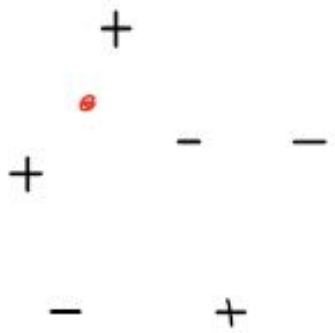


Figure 3.2: Example of domain of K_{NN}

Maybe if point is close to point with label i know then. Maybe they have the same label.
$\hat{y} = +$ *or* $\hat{y} = -$

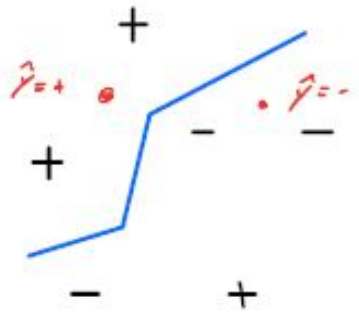


Figure 3.3: Example of domain of K_{NN}

I can came up with some sort of classifier.

Given S training set, i can define h_{NN} $X \to \{-1, 1\}$
$h_{NN}(x) =$ label y_t of the point x_t in S closest to X
(the breaking rule for ties)
For the closest we mean euclidian distance

$X = \mathbb{R}^d$

$$\|x - x_t\| = \sqrt{\sum_{e=1}^{d} (x_e - x_t, e)^2}$$

$$\hat{\ell}(h_{NN}) = 0$$
$$h_{NN}(x_t) = y_t$$

training error is 0!

Lecture 4 - 17-03-2020

We spoke about Knn classifier with voronoi diagram

$$\hat{\ell}(h_{NN}) = 0 \qquad \forall \, Traning \; set$$

h_{NN} predictor needs to store entire dataset.

4.1 Computing h_{NN}

Computing $h_{NN}(x)$ requires computing distances between x and points in the traning set.

$$\Theta(d) \quad time \; for \; each \; distance$$

NN $\rightarrow$ 1-NN
We can generalise NN in K-NN with $k = 1, 3, 5, 7$ so odd K
$h_{k-NN}(x) =$ label corresponding to the majority of labels of the k closet point to x in the training set.

How big could K be if i have n point?
I look at the k closest point
When $k = m$?
The majority, will be a constant classifier h_{k-NN} is constant and corresponds to the majority of training labels
Training error is always 0 for h_{NN}, while for h_{k-NN} will be typically > 0, with $k > 1$
Image: one dimensional classifier and training set is repeated. Is the plot of 1-NN classifier.
Positive and negative. $K = 1$ error is 0.
In the second line we switch to $k = 3$. Second point doesn't switch and third will be classify to positive and we have training mistake.
Switches corresponds to border of voronoi partition.

$$K_{NN} \qquad For \; multiclass \; classification$$

$$(|Y| > 2) \qquad for \; regression \; Y \equiv \mathbb{R}$$

Average of labels of K neighbours $\rightarrow$ i will get a number with prediction.
I can weight average by distance

You can vary this algorithm as you want.

Let's go back to Binary classification.
The k parameter is the effect of making the structure of classifier more complex and less complex for small value of k.

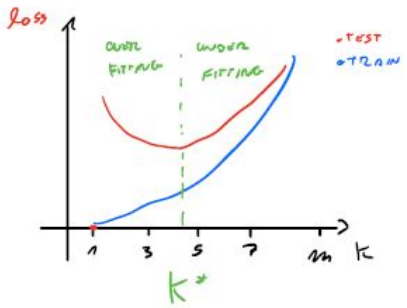


Figure 4.1: Example of domain of K_{NN}

Fix training set and test set
Accury as oppose to the error

Show a plot. Training error is 0 at $k = 0$.
As i go further training error is higher and test error goes down. At some point after which training and set met and then after that training and test error goes up (accuracy goes down).
If i run algorithm is going to be overfitting: training error and test error is high and also underfitting since testing and training are close and both high.
Trade off point is the point in $x = 23$ (more or less).
There are some heuristic to run NN algorithm without value of k.

History

- K_{NN}: from 1960 $\rightarrow X \equiv \mathbb{R}^d$
- Tree predictor: from 1980

4.2 Tree Predictor

If a give you data not welled defined in a Euclidean space.
$X = X_1 \cdot x \cdot ... \cdot X_d \cdot x$ Medical Record
$X_1 = \{Male, Female\}$
$X_2 = \{Yes, No\}$
so we have different data

I want to avoid comparing x_i with x_j, $i \neq j$
so comparing different feature and we want to compare each feature with each self. I don't want to mix them up.
We can use a tree!

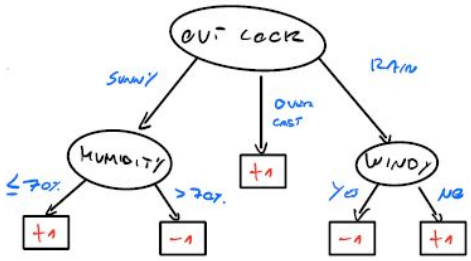


Figure 4.2: Example of domain of K_{NN}

I have 3 features:

- outlook = $\{sunny, overcast, rain\}$
- humidity = $\{[0, 100]\}$
- windy = $\{yes, no\}$

Tree is a natural way of doing decision and abstraction of decision process of one person. It is a good way to deal with categorical variables.
What kind of tree we are talking about?
Tree has inner node and leaves. Leaves are associated with labels (Y) and inner nodes are associated with test.

- Inner node $\rightarrow$ test
- Leaves $\rightarrow$ label in Y

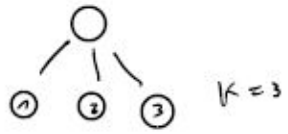


Figure 4.3: Example of domain of K_{NN}

Test if a function f (NOT A PREDICTOR!)
Test $f_i\, X_i \rightarrow \{1, ..., k\}$
where k is the number of children (inner node) to which test is assigned
In a tree predictor we have:

- Root node
- Children are ordered(i know the order of each branch that come out from the node)

$X = \{Sunny, 50\%, No\} \quad \rightarrow \quad$ *are the parameters for* $\{outlook.humidity, windy\}$

$$f_i = \begin{cases} 1, & \text{if } x_2 \in [30\%, 60\%] \\ 2, & \text{if } otherwise \end{cases}$$

where the numbers 1 and 2 are the children
A test is partitioning the range of values of a certain attribute in a number of elements equal to number of children of of the node to which the test is assigned.
$h_T(x)$ is always the label of a leaf of T
This leaf is the leaf to which x is **routed**
Data space for this problem (outlook,..) is partitioned in the leaves of the tree. It won't be like voronoi graph. How do I build a tree given a training set? How do i learn a tree predictor given a training set?

- Decide tree structure (how • many node, leaves ecc..)
- Decide test on inner nodes
- Decide labels on leaves

We have to do this all together and process will be more dynamic. For simplicity binary classification and fix two children for each inner node.

$Y = \{-1, +1\}$
2 children for each inner node

What's the simplest way?
Initial tree and correspond to a costant classifier

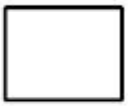


Figure 4.4: Example of domain of K_{NN}

Majority of all example

Figure 4.5: Example of domain of K_{NN}

$(x_1, y_1)...(x_m, y_m)$
$x_t \in X \qquad y_t \in \{-1, +1\}$
Training set $S = \{(x, y) \in S$, x is routed to $\ell\}$
S_ℓ^+

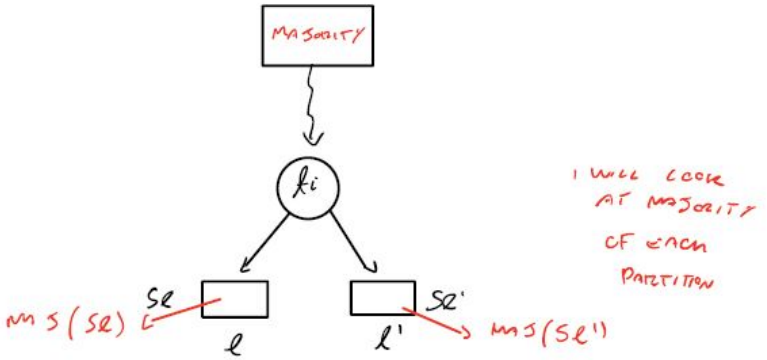


Figure 4.6: Example of domain of K_{NN}

S_ℓ and S'_ℓ are given by the result of the test, not the labels and ℓ and ℓ'.

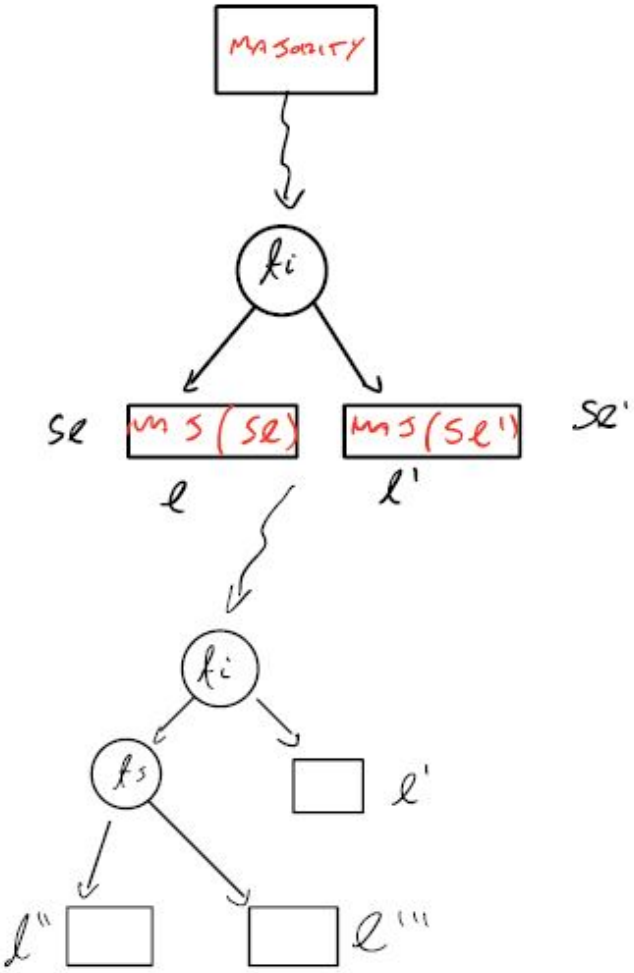


Figure 4.7: Example of domain of K_{NN}

Lecture 5 - 23-03-2020

5.1 Tree Classifier

Supposed we groped a tree up to this point and we are wandering how to grow it.
S Training set $(x_1, y_1)...(x_m, y_m)$, $x_1 \in X$

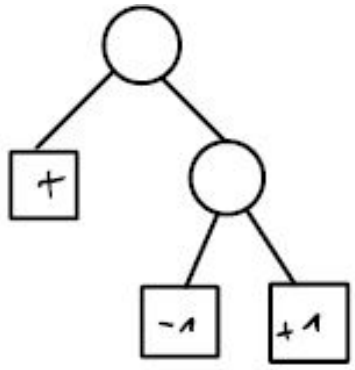


Figure 5.1: Example of domain of K_{NN}

$$S_\ell \equiv \{(x_1, y_1)\, x_t \quad \textit{is router to } \ell\}$$

$y_1 \in \{-1, 1\}$

$$S_\ell^+ \equiv \{(x_1, y_1) \in S_\ell : \quad y_t = +1\}$$
$$S_\ell^- \equiv \{(x_1, y_1) \in S_\ell : \quad y_t = -1\} \qquad S_\ell^+ \cap S_\ell^- \equiv 0 \qquad S_\ell \equiv S_\ell^+ \cup S_\ell^-$$

$$N_\ell = |S_\ell| \qquad N_\ell^+ = |S_\ell^+| \qquad N_\ell^- = |S_\ell^-|$$
$$N_\ell = N_\ell^- + N_\ell^+$$

leaf ℓ classifies all traning example (S_ℓ)

$$Y_\ell = \begin{cases} +1, & \text{If } N_\ell^+ \geq N_\ell^- \\ -1, & \text{If } otherwise \end{cases}$$

ℓ makes a mistake on $min\{N_\ell^+, N_\ell^-\}$ example in S_ℓ

$$\hat{\ell}(h_T) = \frac{1}{m} \cdot \sum_\ell min\{\frac{N_\ell^+}{N_\ell}, \frac{N_\ell^-}{N_\ell}\} \cdot N_\ell =$$

$$= \frac{1}{m} \cdot \sum_\ell \psi \cdot (\frac{N_\ell +}{N_\ell}) \cdot N_\ell \quad \longrightarrow \quad \frac{N_\ell^+}{N_\ell} = 1 - \frac{N_\ell}{N_\ell ??}$$

where $\psi(a) = min\{a, 1-a\} \qquad a \in [0,1]$
I want to replace inner node with other leaves.

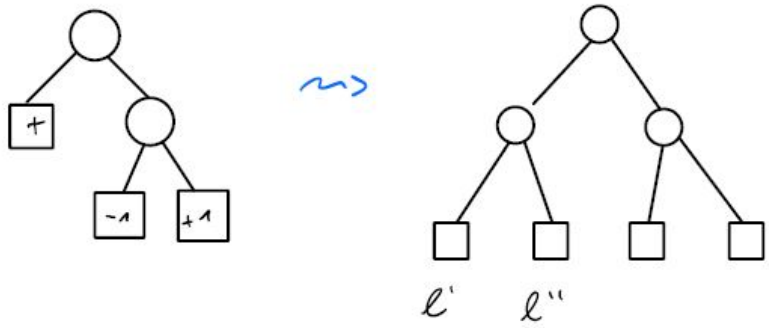


Figure 5.2: Example of domain of K_{NN}

How is traning error going to change? (when i replace inner nodes with other leaves)
I'm hoping my algorithm is not going to overfit (if training error goes to 0 also testing error goes to 0).

5.2 Jensen's inequality

If ψ is a concave function $\longrightarrow$ (like *log* or $\sqrt[2]{\cdot\cdot}$)

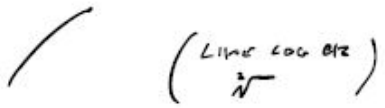


Figure 5.3: Example of domain of K_{NN}

Also ψ is a function that map 0 to 1, $\longrightarrow$ $\psi\,[0,1] \to \mathbb{R}$

$\psi(\alpha \cdot a + (1-\alpha) \cdot b) \geq \alpha \cdot \psi(a) + (1-\alpha) \cdot \psi(b)$ *Also 2° derivative is negative*

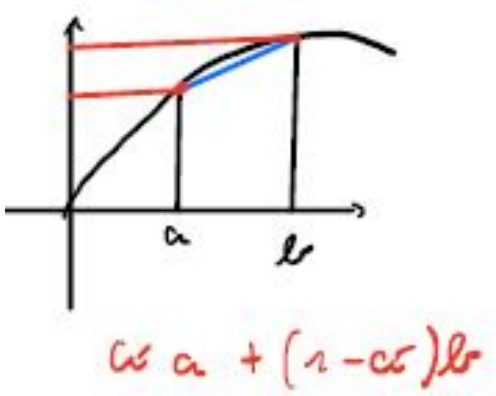


Figure 5.4: Example of domain of K_{NN}

$$\hat{\ell}(h_T) = \frac{1}{m} \cdot \sum_{\ell} \psi(\frac{N_\ell^+}{N_\ell}) \cdot N_\ell$$

Look a single contribution fo a leaf ℓ to training error

$$\psi(\frac{N_\ell^+}{N_\ell}) \cdot N_\ell = \psi(\frac{N_\ell'^+}{N_\ell'} \cdot \frac{N_\ell'}{N_\ell} + \frac{N_\ell"^+}{N_\ell"} \cdot \frac{N_\ell"}{N_\ell}) \cdot N_\ell$$

where $\frac{N_\ell'}{N_\ell} = \alpha$ and $\frac{N_\ell"}{N_\ell} = 1-\alpha$ so $\frac{N_\ell'}{N_\ell} + \frac{N_\ell"}{N_\ell} = 1$ $\longrightarrow \alpha + 1 - \alpha = 1$

$N_{\ell'}^{+} + N_{\ell"}^{+} = N_{\ell}$

I want to check function *min* concave between 0 and 1.

$$min(0,1) = 0 \qquad \psi(a) = min(\alpha, 1-\alpha)$$

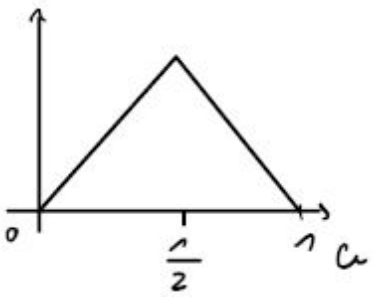


Figure 5.5: Example of domain of K_{NN}

This is a concave function and now I can apply Jensen's inquality

$$\psi(\frac{N_{\ell}^{+}}{N_{\ell}}) \cdot N_{\ell} \geq (\frac{N'_{\ell}}{N_{\ell}} \cdot \psi(\frac{N_{\ell}'^{+}}{N'_{\ell}}) + \frac{N_{\ell}"}{N_{\ell}} \cdot \psi(\frac{N_{\ell}"^{+}}{N_{\ell}"})) \cdot N_{\ell} =$$

$$= \boxed{\psi(\tfrac{N_{\ell}'^{+}}{N'_{\ell}}) \cdot N'_{\ell} + \psi(\tfrac{N_{\ell}"^{+}}{N_{\ell}"}) \cdot N_{\ell}"}$$

This are the contribuion of ℓ' and $\ell"$ to the training error

Every time i split my tree my training error is never going to increase since we have a concave function.
Whenever I'm growing my tree training error is going to be smaller.

Every time a leaf is expanded the training error never goes up. (Hopelly will go down)
I'll should always grow the tree by expanding leave that decrease the training error as much as possible.
If i take the effort of growing the tree i should get benefits. I can imaging that if i grow the tree at random my training error is going to drop down error (but maybe will derive overfitting).
For now is just an intuition since we will introduced statistical learning model.

Could be complicated and tree big may have 100 leave and there could be many way of associating a test with that leaves.
I can spent a lot of time to select which leave is the best promising to split.

- Grow the tree by expanding leave that decrease the training error as much as possible
- In general we can assume:
  greedy algorithm at each step pick the pair leaf and test that cause (approximative) the largest decrease in training error

In practise we want optimise this all the way since it's time expensive. That's the approximately since we are not every time sure.

— MANCA PARTE —

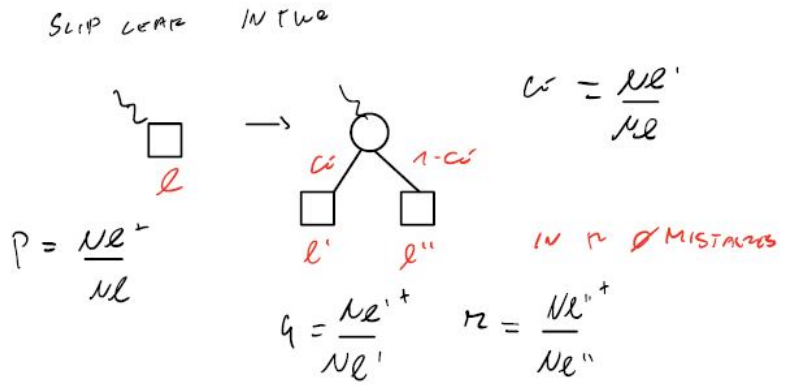


Figure 5.6: Example of domain of K_{NN}

$p = 0.8 \qquad q = 1 \qquad r = 1 \qquad \alpha = 60\%$
Net Change in number of mistakes

$$\psi(p) - (\alpha \cdot \psi(q) + (1-\alpha) \cdot \psi(r)) =$$
$$\ell \quad - \quad \ell' \quad + \quad \ell''$$

Fraction of example miss classified $\ell-$ error $\ell'+$ error ℓ''

$$= 0.2 - (\frac{1}{2} \cdot 0.4 + \frac{1}{2} \cdot 0) = 0$$

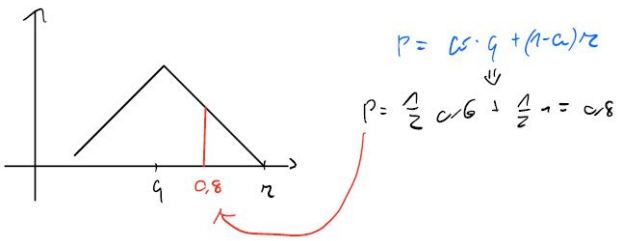


Figure 5.7: Example of domain of K_{NN}

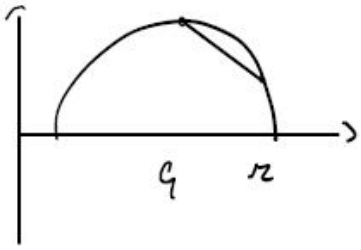


Figure 5.8: Example of domain of K_{NN}

Idea is to replace minimum function with convex combination.

$$\psi(\alpha) = min\ \{\alpha, 1-\alpha\} \qquad \psi(a) \geq \psi(\alpha)$$

$$\begin{cases} \psi_1(\alpha) = 2 \cdot \alpha \cdot (1-\alpha) \longrightarrow GNI \\ \psi_2(\alpha) = -\frac{\alpha}{2} \cdot \ln \alpha - \frac{1-\alpha}{2} \cdot \ln(1-\alpha) \longrightarrow ENTROPY \\ \psi_3(\alpha) = \sqrt{\alpha \cdot (1-\alpha)} \end{cases}$$

All this functions has this shape (concave???)

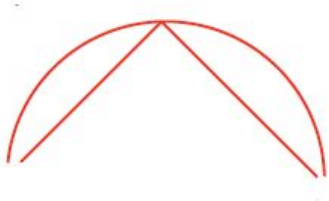


Figure 5.9: Example of domain of K_{NN}

In practise Machine Learning algorithm use GNI or entropy to control the split

5.3 Tree Predictor

- Multi class classification $|Y| > 2 \longrightarrow$ take majority
- Regression $Y = \mathbb{R} \longrightarrow$ take average of labels in S_ℓ

I still take majority among different classes.
Take average of labels in S_ℓ
Unless $\frac{N_\ell^+}{N_\ell} \in 0, 1 \qquad \forall$ leaves ℓ, $\hat{\ell}(h_T) > 0$
Unless leaves are *"pured"*, the training error will be bigger than 0.

In general, i can always write $\hat{\ell}(h_t)$ to 0 by growing enough the tree unless there are x_1 in the Time Series such that $(x_t, y_t)(x_t, y_t')$ with $y_t \neq y_t'$ both occur.

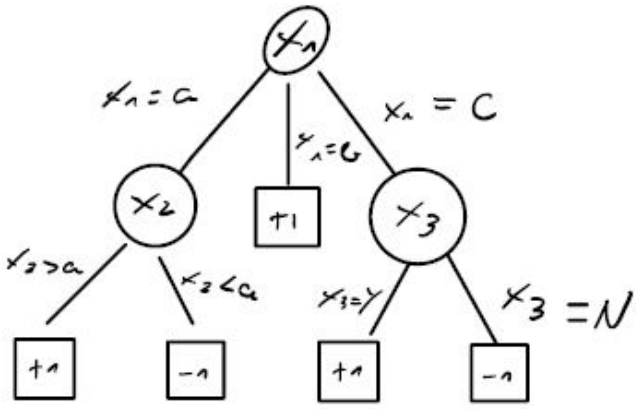


Figure 5.10: Example of domain of K_{NN}

$$if (x_1 = \alpha) \wedge (x_2 =\geq \alpha) \vee (x_1 = b) \vee (x_1 = c) \wedge (x_3 = y)$$
$$\textit{then predict 1}$$
$$\textit{else}$$
$$\textit{then predict -1}$$

— Picture of tree classifier of iris dataset. —
I'm using due attribute at the time.
Each data point is a flower and i can measure how petal and sepal are long.

I can use two attribute and i test this two. I can see the plot of the tree classifier (second one) making test splitting data space into region that has this sort of "blackish" shape (like boxes: blue box, red box, yellow box)
A good exercise in which I want to reconstruct the tree given this picture.

5.4 Statistical model for Machine Learning

To understand Tree classifier, nearest neighbour and other algorithm...
It's important to understand that the only way to have a guideline in which model to choose.

This mathematical model are developed to learning and choose learning algorithm.

Now let start with theoretical model.

- How example (x, y) are generated to create test set and training set?
  We get the dataset but we need to have a mathematical model for this process. (x, y) are drawn from a fixed but unknown probability distribution on the pairs X and Y (X data space, Y label set o label space)

- Why X should be random?
  In general we assumed that not all the x in X are equally likely to be observed. I have some distribution over my data point and this said that I'm most like to get a datapoint to another.

- How much label?
  Often labels are not determined uniquely by their datapoints because labels are given by human that have their subjective thoughts and also natural phenomena. Labels are stochastic phenomena given a datapoint: i will have a distribution.

We're going to write (in capital) (X, Y) since they are random variable drawn from D on $X \cdot Y$ A dataset $(X_1, Y_1)...(X_m, Y_m)$ they are drawn independently from D (distribution on examples)
When I get a training the abstraction of process collecting a training set
D is a joint probability distribution over $X \cdot Y$
where D_x is the marginal over $X \to D_y|x$ (conditional of Y given X).

I can divided my draw in two part. I draw sample and label from conditional.??
Any dataset (training or test) is a random sample (campione casuale) in the statistical sense $\longrightarrow$ so we can use all stastical tools to make inference.

Lecture 6 - 24-03-2020

(X, Y) We random variables drawn iid from D on $X \cdot Y \longrightarrow$ where D is fixed but unknown

Independence does not hold. We do not collect datapoints to an independent process.
Example: identify new article and i want to put categories. The feed is highly depend on what is happening in the world and there are some news highly correlated. Why do we make an assumption that follows reality? Is very convenient in mathematical term. If you assume Independence you can make a lot of process in mathematical term in making the algorithm.
If you have enough data they look independent enough. Statistical learning is not the only way of analyse algorithms —> we will see in linear ML algorithm and at the end you can use both statistical model s

6.1 Bayes Optimal Predictor

$$f^* : X \to Y$$
$$f^*(x) = argmin\, \mathbb{E}\,[\,\ell(y, \hat{y})|X = x\,] \qquad \hat{y} \in Y$$

In general Y given X has distribution $D_y|X = x$
Clearly $\forall\ h \quad X \to Y$

$$\mathbb{E}\,[\,\ell(y, f^*(x))|X = x\,] \leq \mathbb{E}\,[\,\ell(y, h(x)|X = x\,]$$
$$X, Y \qquad \mathbb{E}\,[\,Y|X = x\,] = F(x) \quad \longrightarrow Conditional Expectation$$
$$\mathbb{E}\,[\,\mathbb{E}\,[\,Y|X\,]\,] = \mathbb{E}(Y)$$

Now take Expectation for distribution

$$\mathbb{E}\,[\,\ell(y, f^*(x))\,] \leq [\,\mathbb{E}(\ell(y, h(x))\,]$$

where risk is smaller in f^*
I can look at the quantity before
l_d Bayes risk $\longrightarrow$ Smallest possible risk given a learning problm

$$l_d(f^*) > 0 \qquad because\ y\ are\ still\ stochastic\ given\ X$$

Learning problem can be complem → large risk

6.1.1 Square Loss

$$\ell(y, \hat{y} = (y - \hat{y})^2$$

I want to compute bayes optimal predictor
$\hat{y}, y \in \mathbb{R}$

$$f^*(x) = argmin\, \mathbb{E}\left[\, (y - \hat{y})^2 | X = x \,\right] = \qquad \hat{y} \in \mathbb{R}$$

we use $\mathbb{E}\left[\, X + Y \,\right] = \mathbb{E}[X] + \mathbb{E}[Y] = argmin\, \mathbb{E}\left[\, y^2 + \hat{y}^2 - 2 \cdot y \cdot \hat{y}^2 | X = x \,\right] =$

Dropping y^2 i remove something that is not important for $\hat{y}$

$$= argmin(\mathbb{E}\left[\, y^2 | X = x \,\right] + \hat{y}^2 - 2 \cdot \hat{y} \cdot \mathbb{E}\left[\, y | X = x \,\right]) =$$
$$= argmin(\hat{y}^2 - 2 \cdot \hat{y} \cdot \mathbb{E}\left[\, y | X = x \,\right]) =$$

Expectation is a number, so it's a constant
Assume $\boxdot = y^2$

$$argmin\, \left[\boxdot + \hat{y}^2 + 2 \cdot \hat{y} \cdot \mathbb{E}\left[\, Y | X = x \,\right]\right]$$

where red$G(\hat{y})$ is equal to the part between [...]

$$\frac{dG(\hat{y})}{d\hat{y}} = 2 \cdot \hat{y} - 2 \cdot \mathbb{E}\left[\, y | X = x \,\right] = 0 \quad \longrightarrow \quad \textit{So setting derivative to 0}$$

Suppose we have a learning domain

Figure 6.1: Example of domain of K_{NN}

$G'(\hat{y}) = \hat{y}^2 - 2 \cdot b \cdot \hat{y}$

$$\hat{y} = \mathbb{E}\left[\, y | X = x \,\right] \qquad f^*(x) = \mathbb{E}\left[\, y | X = x \,\right]$$

Square loss is nice because expected prediction is ...
In order to predict the best possibile we have to estimate the value given data point.

$$\mathbb{E}\left[(y - f^*(x))^2 | X = x\right] =$$
$$= \mathbb{E}\left[(y - \mathbb{E}\left[y|X = x\right])^2 | X = x\right] = Var\left[Y|X = x\right]$$

6.1.2 Zero-one loss for binary classification

$Y = \{-1, 1\}$

$$\ell(y, \hat{y}) = I\{\hat{y} \neq y\} \qquad I_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

If $\hat{y} \neq y$ true, indicator function will give us 1, otherwise it will give 0

$$D \quad on \qquad X \cdot Y \qquad D_x^* \quad D_{y|x} = D$$

$$D_x \qquad \eta : X \longrightarrow [0, 1] \qquad \eta = \mathbb{P}(y = 1 | X = x)$$

$$D \rightsquigarrow (D_x, \eta) \quad \longrightarrow \quad \textit{Distribution 0-1 loss}$$

$X \backsim D_x \quad \longrightarrow \quad$ *Where* $\backsim$ *mean "draw from" and* D_x *is marginal distribution*

$Y = 1 \qquad$ *with probability* $\eta(x)$

$$D_{y|x} = \{\eta(x), 1 - \eta(x)\}$$

Suppose we have a learning domain

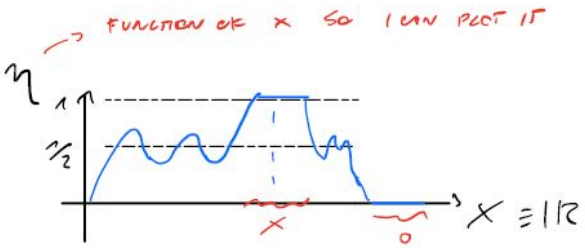


Figure 6.2: Example of domain of K_{NN}

where η is a function of x, so i can plot it
η will te me $Prob(x) =$
η tells me a lot how hard is learning problem in the domain
$\eta(x)$ is not necessary continous

Figure 6.3: Example of domain of K_{NN}

$\eta(x) \in \{0, 1\}$ y is always determined by x

How to get f^* from the graph?

$$f^+ : X \to \{-1, 1\}$$
$$Y = \{-1, +1\}$$

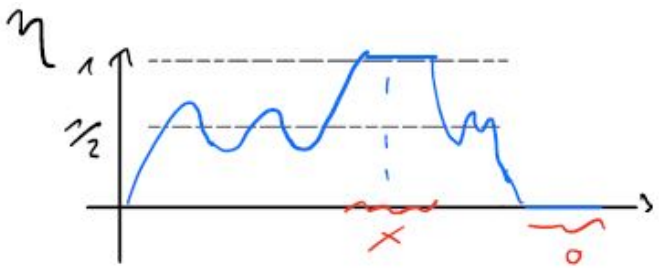


Figure 6.4: Example of domain of K_{NN}

==========
MANCA ROBAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
==========

$$f^*(x) = argmin\, \mathbb{E}\,[\,\ell(y, \hat{y})|X = x\,] = \qquad \longrightarrow \hat{y} \in \{-1, +1\}$$
$$= argmin\, \mathbb{E}\,[\,I\{\hat{y} = 1\} \cdot I\{Y = -1\} + I\{\hat{y} = -1\} \cdot I\{y = 1\}\,|\,X = x\,] =$$

we are splitting wrong cases

$$= argmin\,(\,I\{\hat{y} = 1\} \cdot \mathbb{E}\,[\,I\{Y = -1\}|\,X = x\,] + I\{\hat{y} = -1\} \cdot \mathbb{E}\,[\,I\{y = 1\}\,|\,X = x\,]\,) = \quad ※$$

We know that:

$$\mathbb{E}\,[\,I\{y = -1\}\,|\,X = x\,] = 1 \cdot \mathbb{P}(\hat{y} = -1|X = x) + 0 \cdot \mathbb{P}(y = 1|X = x) =$$
$$\mathbb{P}(x = -1|X = x) = 1 - \eta(x)$$

$$※ = argmin\,(\,I\{\hat{y} = 1\} \cdot (1 - \eta(x)) + I\{\hat{y} = -1\} \cdot (\eta(x)\,)$$

where Blue colored $I\{...\} = 1°$ and Orange $I\{...\} = 2°$

I have to choose -1 or +1 so we will **remove one of the two (1° or 2°)**
It depend on $\eta(x)$:

- If $\eta(x) < \frac{1}{2} \quad \longrightarrow \quad$ kill 1°
- Else $\eta(x) \geq \frac{1}{2} \quad \longrightarrow \quad$ kill 2°

$$f^*(x) = \begin{cases} +1 & if\, \eta(x) \geq \frac{1}{2} \\ -1 & if\, \eta(x) < \frac{1}{2} \end{cases}$$

6.2 Bayes Risk

$$\mathbb{E}\left[\, I\{y \neq f^*(x)\} \,|\, X = x \,\right] = \mathbb{P}(y \neq f^*(x) | X = x)$$

$$\eta(x) \geq \frac{1}{2} \quad \Rightarrow \quad \hat{y} = 1 \quad \Rightarrow \quad \mathbb{P}(y \neq 1 | X = x) = 1 - \eta(x)$$

$$\eta(x) < \frac{1}{2} \quad \Rightarrow \quad \hat{y} = -1 \quad \Rightarrow \quad \mathbb{P}(y \neq 1 | X = x) = \eta(x)$$

Conditiona risk for 0-1 loss is:

$$\mathbb{E}\left[\, \ell(y, f^*(x)) \,|\, X = x \,\right] \quad = \quad I\{\eta(x) \geq \frac{1}{2}\} \cdot (1 - \eta(x)) + I\{\eta(x) < \frac{1}{2}\} \cdot \eta(x) =$$
$$= min\left\{\eta(x), 1 - \eta(x)\right\}$$

$$\mathbb{E}\left[\, \ell, f^*(x) \,\right] = \mathbb{E}\left[\, min\left\{\eta(x), 1 - \eta(x)\right\} \,\right]$$

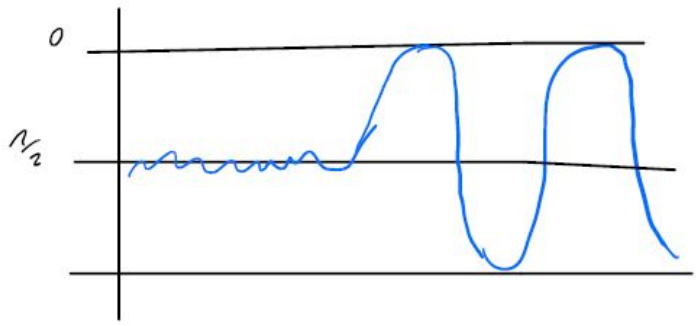


Figure 6.5: Example of domain of K_{NN}

Conditional risk will be high aroun the half so min between the two is around

the half since the labels are random i will get an error near 50%.
My condition risk will be 0 in the region in the bottom since label are going to be deterministic.

Lecture 7 - 30-03-2020

Bounding statistical risk of a predictor
Design a learning algorithm that predict with small statistical risk

$$(D, \ell) \qquad \ell_d(h) = \mathbb{E}\left[\, \ell(y), h(x) \,\right]$$

were D is unknown

$$\ell(y, \hat{y}) \in [0, 1] \quad \forall y, \hat{y} \in Y$$

We cannot compute statistical risk of all predictor.
We assume statistical loss is bounded so between 0 and 1. Not true for all losses (like logarithmic).
Before design a learning algorithm with lowest risk, How can we estimate risk?
We can use test error → way to measure performances of a predictor h. We want to link test error and risk.
Test set $S' = \{(x'_1, y'_1)...(x'_n, y'_n)\}$ is a random sample from D
How can we use this assumption?
Go back to the definition of test error

Sample mean (IT: Media campionaria)

$$\hat{\ell}_s(h) = \frac{1}{n} \cdot \sum_{t=1}^{n} \ell(\hat{y}_t, h(x'_t))$$

i can look at this as a random variable $\ell(y'_t, h(x'_t))$

$$\mathbb{E}\left[\, \ell(y'_t, h(x'_t)) \right] = \ell_D(h) \longrightarrow \textit{risk}$$

Using law of large number (LLN), i know that:

$$\hat{\ell} \longrightarrow \ell_D(h) \qquad as \quad n \to \infty$$

We cannot have a sample of $n = \infty$ so we will introduce another assumption: the Chernoff-Hoffding bound

7.1 Chernoff-Hoffding bound

$$Z_1, ..., Z_n \quad \textit{iid random variable} \qquad \mathbb{E}\left[Z_t\right] = u$$

all drawn for the same distribution

$$t = 1, ..., n \qquad and \qquad 0 \leq Z_t \leq 1 \qquad t = 1, ..., n \quad then \quad \forall \varepsilon > 0$$

$$\mathbb{P}\left(\frac{1}{n}\cdot\sum_{t=1}^{n} z_t > u+\varepsilon\right) \leq e^{-2\,\varepsilon^2\,n} \qquad or \qquad \mathbb{P}\left(\frac{1}{n}\cdot\sum_{t=1}^{n} z_t < u+\varepsilon\right) \leq e^{-2\,\varepsilon^2\,n}$$

as sample size then $\downarrow$

$$Z_t = \ell(Y_t', h(X_t')) \in [0,1]$$

$(X_1', Y_1')...(X_n', Y_N')$ are *iid* therefore,
$\ell\left(Y_t', h\left(X_t'\right)\right) \quad t = 1, ..., n$ are also *iid*
We are using the bound of e to bound the deviation of this.

7.2 Union Bound

Union bound: a collection of event not necessary disjoint, then i know that probability of the union of this event is the at most the sum of the probabilities of individual events

$$A_1, ..., A_n \qquad \mathbb{P}\left(A_1 \cup ... \cup A_n\right) \leq \sum_{t=1}^{n} \mathbb{P}\left(A_t\right)$$

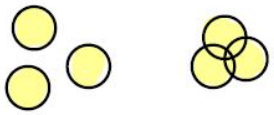


Figure 7.1: Example

that's why $\leq$

$$\mathbb{P}\left(\mid \hat{\ell}_{s'}\left(h\right) - \ell_D\left(h\right) \mid > \varepsilon\right)$$

This is the probability according to the random draw of the test set.

If test error differ from the risk by a number epsilon > 0. I want to bound the probability. This two thing will differ by more than epsilon. How can i use the Chernoff bound?

$$\mid \hat{\ell}_{s'}\left(h\right) - \ell_D\left(h\right) \mid > \varepsilon \quad \Rightarrow \quad \hat{\ell}_{s'}\left(h\right) - \ell_D\left(h\right) > \varepsilon \quad \vee \quad \hat{\ell}_D\left(h\right) - \ell_{s'}\left(h\right) > \varepsilon$$

Figure 7.2: Example

$$A, B \qquad A \Rightarrow B \qquad \mathbb{P}(A) < \mathbb{P}(B)$$

$$\mathbb{P}\left(\mid \hat{\ell}_{s'}(h) - \ell_D(h) \mid > \varepsilon\right) \leq \mathbb{P}\left(\left|\hat{\ell}_{s'}(h) - \ell_D(h)\right|\right) \quad \cup \quad \mathbb{P}\left(\left|\hat{\ell}_D(h) - \ell_{s'}(h)\right|\right) \leq$$

$$\leq \mathbb{P}\left(\hat{\ell}_{s'} > \ell_D(h) + \varepsilon\right) + \mathbb{P}\left(\hat{\ell}_{s'} < \ell_D(h) - \varepsilon\right) \quad \leq \quad 2 \cdot e^{-2\varepsilon^2 n} \quad \Rightarrow \textit{we call it } \delta$$

$$\varepsilon = \sqrt{\frac{1}{2 \cdot n} \ln \frac{2}{\delta}}$$

The two events are disjoint

This mean that probability of this deviation is at least delta!

$$\mid \hat{\ell}_{s'}(h) - \ell_D(h) \mid \leq \sqrt{\frac{1}{2 \cdot n} \ln \frac{2}{\delta}} \qquad \textit{with probability at least } 1 - \delta$$

Test error of true estimate is going to be good for this value (δ)
Confidence interval for risk at confidence level 1-delta.

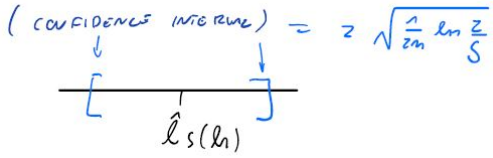


Figure 7.3: Example

I want to take $\delta = 0,05$ so that $1 - \delta$ is 95%. So test error is going to be an estimate of the true risk which is precise that depend on how big is the test set (n).
As n grows I can pin down the position of the true risk.
This is how we can use probability to make sense of what we do in practise.
If we take a predictor h we can compute the risk error estimate.

We can measure how accurate is our risk error estimate.
Test error is an estimate of risk for a given predictor (h).

$$\mathbb{E}\left[\ell\left(Y_t', h\left(X_t'\right)\right)\right] = \ell_D\left(h\right)$$

h is fixed with respect to S' $\longrightarrow$ h does not depend on the test set. So learning algorithm which produce h not have access to test set.
If we use test set we break down this equation.

Now, how to **build a good algorithm?**
Training set $S = \{(x_1, y_1) \ldots (x_m, y_m)\}$ random sample
A $A\left(S\right) = h$ predictor output by A given S where A is learning algorithm as function of traning set S.
$\forall\, S$ $A\left(S\right) \in H$ $h^* \in H$

$\ell_D\left(h^*\right) = min\, \ell_D\left(h\right)$ $\hat{\ell}_s\left(h^*\right)$ *is closed to* $\ell_D\left(h^*\right)$ $\longrightarrow$ **it is going to have small error**
where $\ell_D\left(h^*\right)$ is the training error of h^*

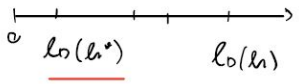


Figure 7.4: Example

This guy $\ell_D\left(h^*\right)$ is closest to 0 since optimum

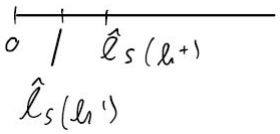


Figure 7.5: Example

In risk we get opt in h^* but in empirical one we could get another h' better than h^+

In order to fix on a concrete algorithm we are going to take the empirical Islam minimiser (ERM) algorithm.
A is ERM on H $(A) = \hat{h} = (\in)\, argmin\, \hat{\ell}_S\left(h\right)$
Once I piack $\hat{h}$ i can look at training error of ERM

$$\hat{\ell}_S\left(\hat{h}\right) of \hat{h} = A(S)$$

where $\hat{\ell}_S$ is the training error

Should $\hat{\ell}_S\left(\hat{h}\right)$ be close to $\ell_D\left(\hat{h}\right)$?
I'm interested in empirical error minimiser and do a trivial decomposition.

$$\begin{aligned}\ell_d\left(\hat{h}\right) = \quad & \ell_D\left(\hat{h}\right) - \ell_d\left(h^*\right) + && \longrightarrow \textbf{Variance error} \Rightarrow \textbf{Overfitting} \\ & + \ell_d\left(h^+\right) - \ell_d\left(f^*\right) + && \longrightarrow \textbf{Bias error} \Rightarrow \textbf{Underfitting} \\ & + \ell_D\left(f^*\right) && \longrightarrow \textbf{Bayes risk} \Rightarrow \textbf{Unavoidable}\end{aligned}$$

Even the best predictor is going to suffer that

$$f^* \textit{ is } \textbf{\textit{Bayes Optimal}} \textit{ for } (D, \ell)$$
$$\forall h \qquad \ell_D\left(h\right) \geq \ell_D\left(f^*\right)$$

If $f^* \notin H$ then $\ell_D\left(h^*\right) > \ell_D(f^*)$

If i pick h^* I will pick some error because we are not close enough to the risk.
We called this component **bias error**.
Bias error is responsible for underfitting (when training and test are close to each but they are both high :( )
Variance error over fitting

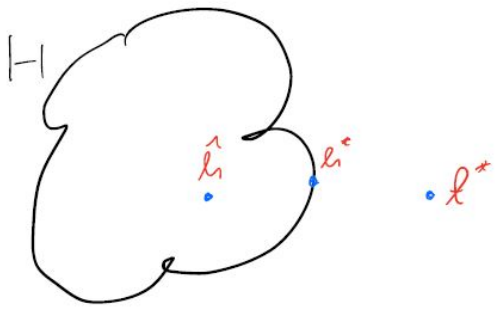


Figure 7.6: Draw of how $\hat{h}$, h^* and f^* are represented

Variance is a random quantity and we want to study this. We can always get risk from training error.

7.3 Studying overfitting of a ERM

We can bound it with probability.
I add and subtract trivial traning error $\hat{\ell}_S(h)$

$$\ell_D\left(\hat{h}\right) - \ell_d(h^*) = \ell_D\left(\hat{h}\right) - \hat{\ell}_S(h) + \hat{\ell}_S\left(\hat{h}\right) - \ell_D(h^*) \leq$$

$$\leq \ell_D\left(\hat{h}\right) - \hat{\ell}_S\left(\hat{h}\right) + \hat{\ell}_S(h^*) - \ell_D(h^*) \leq$$

$$\leq |\,\ell_D\left(\hat{h}\right) - \hat{\ell}_S(h)\,| + |\,\hat{\ell}_S\left(h^+\right) - \ell_D(h^*)\,| \leq$$

$$\leq \quad 2 \cdot max\,|\hat{\ell}_S(h) - \ell_D(h)\,|$$

(no probability here)
Any given $\hat{h}$ **minising** $\hat{\ell}_S(h)$

Now assume we have a large deviation

$$Assume \quad \ell_D\left(\hat{h}\right) - \ell_D(h^*) > \varepsilon \qquad \Rightarrow \qquad max\,|\,\hat{\ell}_S(h) - \ell_D(h)\,| > \frac{\varepsilon}{2}$$

We know $\ell_d\left(\hat{h}\right) - \ell_D(h^*) \quad \leq \quad 2 \cdot max\,|\,\hat{\ell}_S(h) - \ell_D(h)\,| \quad \Rightarrow$

$$\Rightarrow \quad \exists h \in H \qquad |\,\hat{\ell}_S(h) - \ell_D(h)\,| > \frac{3}{2} \qquad \Rightarrow$$

with $|H| < \infty$

$$\Rightarrow U\left(|\,\hat{\ell}_S(h) - \ell_D(h)\,|\right) > \frac{3}{2}$$

$$\mathbb{P}\left(\ell_D\left(\hat{h}\right) - \ell_D(h^*) > \varepsilon\right) \quad \leq \quad \mathbb{P}\left(U\left(|\,\hat{\ell}_S(h) - \ell_D(h)\,|\right) > \frac{3}{2}\right) \quad \leq$$

$$\leq \quad \sum_{h \in H} \mathbb{P}\left(|\,\hat{\ell}_S(h) - \ell_D(h)\,| > \frac{3}{2}\right) \qquad \leq \qquad \sum_{h \in H} 2 \cdot e^{-2\left(\frac{\varepsilon}{2}\right)^2 m} \qquad \leq$$

Union Bound Chernoff. Hoffding bound $(\mathbb{P}(...))$

$$\leq \quad 2 \cdot |H| e^{-\frac{\varepsilon^2}{2} m}$$

Solve for $\varepsilon \qquad 2 \cdot |H| e^{-\frac{\varepsilon^2}{2} m} \quad = \quad \delta$

$$Solve\ for\ \varepsilon \longrightarrow \quad \varepsilon = \sqrt{\frac{2}{m} \cdot \ln \cdot \frac{2|H|}{\delta}}$$

$$\ell_D\left(\hat{h}\right) - \ell_D\left(h^*\right) \quad \leq \quad \sqrt{\frac{2}{m} \cdot \ln \cdot \frac{2|H|}{\delta}}$$

With probability at least $1 - \delta$ with respect to random draw of S.
We want $m >> ln|H|$ $\longrightarrow$ in order to avoid overfitting

Lecture 8 - 31-03-2020

$$|H| < \infty \qquad \hat{h} = argmin\, \hat{\ell}_S(h) \qquad h^* = argmin\, \ell_D(h)$$

minimise risk

Bias-Variance decomposition

$$\ell_d\left(\hat{h}_S\right) = \ell_D\left(\hat{h}_S\right) - \ell_d(h^*) + \longrightarrow \textbf{Variance error} \Rightarrow \textbf{Overfitting}$$

$$+\ell_d\left(h^+\right) - \ell_d(f^*) + \longrightarrow \textbf{Bias error} \Rightarrow \textbf{Underfitting}$$

$$+\ell_D(f^*) \longrightarrow \textbf{Bayes risk} \Rightarrow \textbf{Unavoidable}$$

We state this for all algorithm but we studied for ERM.

$$\ell_D\left(\hat{h}_S\right) \leq \ell_D(h^*) + \sqrt{\frac{2}{m} \ln \frac{2|H|}{\delta}} \qquad \textit{with probability at least } 1-\delta \textit{ over the draw of } S$$

we want **this** to be small when $m >> \ln|H|$

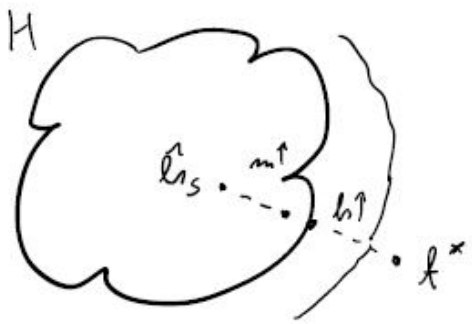


Figure 8.1: Representation of $\hat{h}$, h^* and f^*

Size of model fix, increase m (size of sample) when m is bigger $\longrightarrow$ variance goes down.
When $|H| \longrightarrow$ big model will be closer to optimal

- m grows $\Rightarrow$ variance error goes down (if not overfitting)
- $|H|$ grows $\Rightarrow$ bias error goes down (if not underfitting)

ERM with $|H| < \infty$
A H such that $\forall s\ A(S) \in H$

We controlled this event:

$$\forall h \in H \qquad \hat{\ell}_S(h) - \ell_D(h) \quad \leq \quad \sqrt{\frac{2}{m} \ln \frac{2|H|}{\delta}} \qquad \textit{with probability at least } 1-\delta$$

We assure that training error is a good proxy for the true risk.

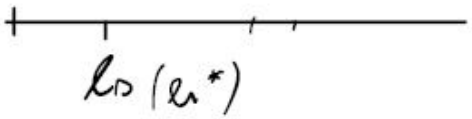


Figure 8.2: Example

If I do the empirical way(for a specific training set) i get something different.

8.1 The problem of estimating risk in practise

Test error is a good estimate for a risk, provided $\frac{1}{\sqrt{n}}$ is small

It's usually good to take a small test set.
80/20 big data, if low data better to look at good estimate...=??
Usually small test set is still ok

Typically we are not given the predictor, instead we start from a learning algorithm. It's true that A has parameter (how many nodes for a tree classifier for example).
In general, if I have parameters then i get a set of different classifier Imaging Algorithm that has parameter θ:

$$A \qquad \{A_\theta : \theta \in \Theta\} \qquad A_\theta(S) = \hat{h}_S$$

$\mathbb{E}[\ell_D(A_\theta(S))]$ **typically averaged over S of size m for fixed m**

In general you may also be interest looking at the best choice of parameter for your algorithm: Suppose now that i want to look at mini sing the risk

with the respect to the choice of the parameter

$\mathbb{E}\left[min\,\ell_D\left(A_\theta\left(S\right)\right)\right]$ i want to choose parameter that minimise risk running algorithm on training set.

I would like to choose best possible value for my parameter k in the k-NN and i want to estimate the risk.
θ can be a set of parameters (so more than 1 parameter), so theta are the **hyper parameters of A**.
In general this parameter are the choices that algorithm that can make: parameter that define a classifier (example nodes and test in the internal nodes and label in the leaves)

Hyper parameters are not determined by the training set $\rightarrow$ chosen before the training set.
I fix k for K_{NN} and I choose an upper bound of number of nodes after the training set is given.
So, some parameters are given after training set is given and some parameter fixed before.
Now the idea is that parameter are determined after training set is given, while hyper parameters are given before training set to get then a predictor.
I have a family of algorithms, so I choose a hyper parameters to get a one algorithm.
Most algorithm are given as family. We have first decide hyper parameters not determined on the training set.
One way to move is to take you dataset and the split that in 3 parts:

- Training set
- Development (or validation) test
- Test sets

There should not be a leak of information between test and train and development.
We get family of algorithm, train with train set and then use dev set to test the parameter and choice the parameters. Once i found the parameter I retrain the algorithm with a part of train and dev set to being then tested.
Development set is like a fake test set $\rightarrow$ it usefull to choose parameters.
Algorithm steps:

1. Train A_θ on training set for each $\theta \in \Theta$ (**grid search**)

2. Find θ such that $\hat{h} = A_{\hat{\theta}}(S)$ minimise developement error
3. Train $A_{\hat{\theta}}$ on training + development set
4. Test resulting predictor on test set

(There's theory about this but it's difficult and we are not going to do that)

It's heuristic and kinda simple to do. This technics work for every learning algorithm.
One parameter: grid on this
Two parameters ecc..
It's quadratic!
Good learning algorithm should have small number of hyperparameters.

8.2 Cross-validation

Solving an easier problem: suppose you have a dataset and what you do is that you can choose training set and test set.
Algorithm A with no hyper parameters and we would like to check how good is the predictor: how good can A be? A is good if the predictor that generate has low risk.
I split dataset in training and test set.

Figure 8.3: Splitting test and training set

$$h = A(A) \qquad \hat{\ell}_{S'}(h) \approx \ell_D(h)$$

I can use Cross-validation! (CV). It helps estimate the risk.

$$\textbf{CV:} \qquad \mathbb{E}\left[\ell_D\left(A\left(S\right)\right)\right]$$

I would like to average ....
How do i do this estimate? Super easy, take my data assuming CV as parameter not of the algorithm A, but intrinsic to cross validation.

Parameter k-fold CV typically $k = 5$ or 10.
What do you do ? If i take a specific training set, i got a ruff estimate of A.
I shouldn't split one but several time.
There are different way but he give us another:
split data in k folds randomly!

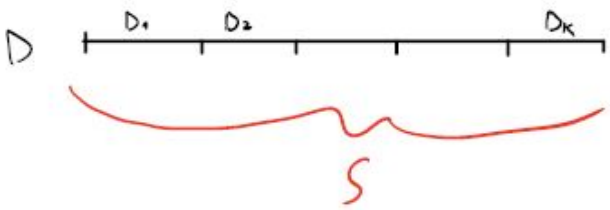


Figure 8.4: K-folds

For each k we define $S^{(k)}$ **take out k-st D**

$$S^{(k)} = S \backslash D_k$$

$$S^{(1)} = D_2 \cup D_3 \cup ... \cup D_k$$
$$S^{(2)} = D_1 \cup D_3 \cup ... \cup D_k$$

For each $k - 1, ...k$ **folds** $S^{(k)}$ training part and D_k test part

$$h_K = A\left(S^{(k)}\right) \qquad \hat{\ell}_{DK}\left(h_K\right) = \frac{k}{m} \sum_{(x,y) \in D_k} \ell\left(y, h_K\left(x\right)\right)$$

Repeat the procedure for $k = 1...k$ get $h_1, ..., h_k$

$$Compute \qquad \frac{1}{k} \cdot \sum_{k=1}^{K} \hat{\ell}_{DK}\left(h_k\right)$$

where CV is $\mathbb{E}\left[\ell_D\left(A\left(S\right)\right)\right]$ and this is called ..... —MANCA — Estimate

It's used a lot!
You get data from internet, then what to do? I want to try an algorithm, try K_{NN} so you do Cross validation and will give you the risk.
In some other cases you get a splitted dataset in training and testing set. You don't use CV since the dataset is already splitted in training and test.

8.3 Nested cross validation

The use of CV to solve the hyper parameters choice problem. If you are given test and training set you can solve splitting in training set and dev set. Suppose not given train and test, you can assign arbitrary .....
Now the idea: you give me a way to optimise splitting training set and test set and then avoiding using a ...
This is what cross validation is doing.

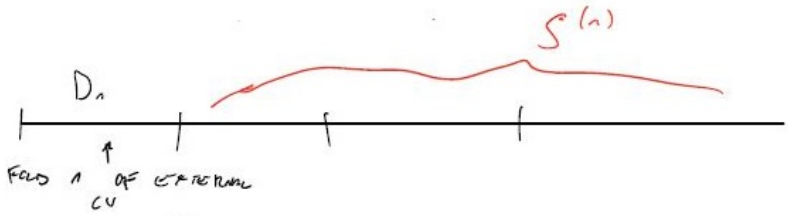


Figure 8.5: Nested Cross Validation

$\{A_\theta : \theta \in \Theta\}$ which i run in each fold?
The idea is **running interval CV for the folds**
I have fold D_1 and $S(1)$ and i perform external validation, then i run internal CV in $S(1)$.
On each training part of the external CV, run a internal CV for each A_θ.
When $\theta \in$ grid (Θ):

- I have a CV-estimate and for each A_θ pick the θ with best CV-estimate
- Run $A_{\hat{\theta}}$ on the entire training part of current external fold

Basically what I'm choosing the best hyper parameter on each fold of the external CV.
External CV is not testing $A_{\hat{\theta}}$ for a given $\hat{\theta} \in \Theta$
I am not measuring a goodness of predictor generate by algorithm for given value of hyper parameters but what I'm estimating is the average risk of the predictor output by learning algorithm when hyper parameters are optimise on the training set. This optimisation on training set is done into a internal CV. To avoid be depend i run and external CV.
Many platform like sklearn allow you to do that in two lines of code. So i can specify the grid, predictor, number of falls for internal and external. It took a bit but that's it in in two lines of code
Cross validation can be done in every order of D_1 or D_2 or D_k. So doesn't matter the order we start fold D2 or D1.

Lecture 9 - 06-04-2020

$\hat{h}$ is ERM predictor

$$\ell_D\left(\hat{h}\right) \leq min\ \ell_D\left(h\right) + \sqrt{\frac{2}{m}\ \ln\ \frac{2\,H}{\delta}} \qquad with\ prob.\ at\ least\ 1-\delta$$

Now we do it with tree predictors

9.1 Tree predictors

$$X = \{0,1\}^d \longrightarrow \textbf{Binary classification}$$

$$h : \{0,1\}^d \longrightarrow \textbf{Binary classification H1}$$

How big is this class?
Take the size of codomain power the domain $\longrightarrow |H| = 2^{2^d}$
Can we have a tree predictor that predict every H in this class?
For every $h : \{0,1\}^d \longleftrightarrow \{-1,1\} \quad \exists T$

We can **build a tree** such that $h_T = h$

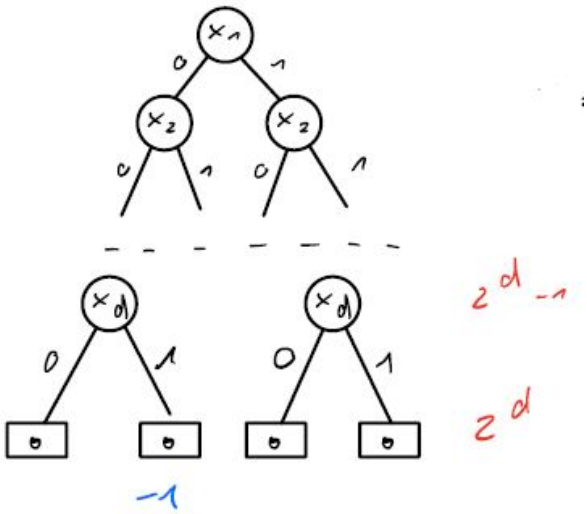


Figure 9.1: Tree building

$X = (0,0,1,...,1) \qquad h\left(x\right) = -1$

$x_1, x_2, x_3, ..., x_d$

I can apply my analisys to this predictors
If I run ERM on H

$$\ell_D\left(\hat{h}\right) \leq min\, \ell_D\left(\hat{h}\right) + \sqrt{\frac{2}{m} 2^d \ln 2 + \ln \frac{2}{\delta}} \qquad \longrightarrow \ln |H| + \ln \frac{2}{\delta}$$

No sense! What we find about training set that we need?
Worst case of overfitting $m >> 2^D = |X| \Rightarrow$ training sample larger

PROBLEM: cannot learn from a class to big (H is too big)
I can control H just limiting the number of nodes.

$H_N \longrightarrow$ tree T with at most N node, $N << 2^D$
$|H_N| = ?$

$|H_N| = (\#\ of\ trees\ with\ \leq N\ nodes) \times (\#\ of\ test\ on\ interval\ nodes\ ) \times (\ \#\ labels\ on\ leaves)$

$$|H_N| = \bigotimes \times d^M \times 2^{N-M}$$

N of which $N - M$ are leaves

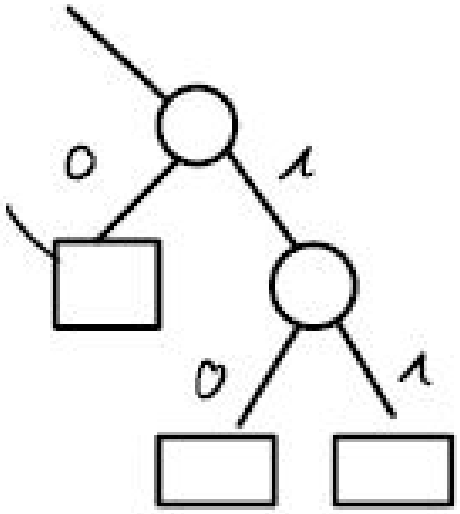


Figure 9.2: Tree with at most N node

$\bigotimes \#\ of\ binary\ trees\ with\ N\ nodes,\ called\ \textbf{Catalan Number}$

9.1.1 Catalan Number

*We are using a binomial *

$$\frac{1}{N}\binom{2N-2}{N-1} \quad \leq \quad \frac{1}{N}\left(e\,\frac{(2N-2)}{N-1}\right)^{N-1} = \frac{1}{N}\,(2e)^{N-1}$$

$$\binom{N}{K} \quad \leq \quad \left(\frac{e\,n}{k}\right)^{k} \qquad \textit{from Stirling approximation}$$

Counting the number of tree structure: a binary tree with exactly N nodes. Catalan counts this number. $\longrightarrow$ **but we need a quantity to interpret easily**. So we compute it in another way.
Now we can rearrange everything.

$$|H_N| \quad \leq \quad \frac{1}{N}\,(2e)^{N-1}\,H^M\,2^{N-M} \quad \leq \quad (2\,e\,d)^N$$

$$d \geq 2 \qquad \leq d^{N-M}$$

where **we ignore** $\frac{1}{N}$ **since we are going to use the** log

ERM on $H_N \quad \hat{h}$

$$\ell_D\left(\hat{h}\right) \leq \min_{\mathbf{h}\in\mathbf{H_N}} \ell_D\left(h\right) + \sqrt{\frac{2}{m}\left(N\cdot(1+\ln(2\cdot d)) + \ln\frac{2}{\delta}\right)}$$

were $N\cdot(1+\ln(2\cdot d)) \quad = \quad \ln(H_N)$

In order to not overfit $m >> N\cdot\ln d$
$N\cdot\ln d << 2^d$ for reasonable value of N
We grow the tree and a some point we stop.

$$\ell_D\left(h\right) \leq \hat{\ell}_S\left(h\right) + \varepsilon \qquad \forall h \in H_N \qquad \textit{with probability at least } 1-\delta$$

remove N **in** H_N **and include** h **on** ε
we remove the N index in H_N adding h on ε

$$\ell_D\left(h\right) \leq \hat{\ell}_S\left(h\right) + \varepsilon_h \qquad \forall h \in H_{\not N}$$

$$W : H \longrightarrow [0,1] \qquad \sum_{h\in H} w\left(h\right) \leq 1$$

How to use this to control over risk?

$$\mathbb{P}\left(\exists h \in H \;:\; |\,\hat{\ell}_S\left(h\right) - \ell_D\left(h\right)\,| \;> \varepsilon_h\right) \quad \leq$$

where $\hat{\ell}_S$ is the prob my training set cases is true

$$\leq \sum_{h \in H} \mathbb{P}\left(\mid \hat{\ell}_S(h) - \ell_D(h) \mid > \varepsilon_h \right) \leq \sum_{h \in H} 2\, e^{-2\, m\, \varepsilon\, h^2} \leq$$

$$\leq \delta \qquad \longrightarrow \textit{since } w(h) \textit{ sum to } 1 \left(\sum_{h \in H} \right)$$

I want to choose $2\, e^{-2\, m\, \varepsilon\, h^2} = \delta\, w(h)$

$$2\, e^{-2\, m\, \varepsilon\, h^2} = \delta\, w(h) \qquad \Leftrightarrow \qquad \textit{— MANCA PARTEEEE —}$$

therefore:

$$\ell_D(h) \leq \hat{\ell}_S(h) + \sqrt{\frac{1}{2\, m} \cdot \left(\ln \frac{1}{w(h)} + ln \frac{2}{\delta} \right)} \quad \textit{w. p. at least } 1 - \delta \quad \forall h \in H$$

Now, instead of using ERM we use

$$\hat{h} = arg \min_{h \in H} \left(\hat{\ell}_S(h) + \sqrt{\frac{1}{2\, m} \cdot \left(\ln \frac{1}{w(h)} + ln \frac{2}{\delta} \right)} \right)$$

where $\sqrt{...}$ term is the penalisation term

Since our class is very large we add this part in order to avoid overfitting. Instead of minimising training error alone i minimise training error + penalisation error.

In order to pick w(h) we are going to use **coding theory**
The idea is I have my trees and i want to encode all tree predictors in H using strings of bits.

$\sigma : H \longrightarrow \{0,1\}^*$ **coding function for trees**
$\forall\, h, h' \in H$ $\sigma(h)$ not a prefix of $\sigma(h')$
$h \neq h'$ where $\sigma(h)$ and $\sigma(h')$ are **string of bits**

σ is called **istantaneous coding function**
Istantaneous coding function has a property called **kraft inequality**

$$\sum_{h \in H} 2^{-|\,\sigma(h)\,|} \leq 1 \qquad w(h) = 2^{-|\,\sigma(h)\,|}$$

I can design $\sigma : H \longrightarrow \{0,1\}^*$ *istantaneous* $|\,\sigma(h)\,|$

$$\ln |H_N| = O\left(N \cdot \ln d\right)$$

number of bits i need = number of node in h

Even if i insist in istantaneous i do not lose ... – MANCA PARTE –

$$|\,\sigma(h)\,| = O\left(N \cdot \ln d\right)$$

Using this σ and $w(h) = 2^{-|\,\sigma(h)\,|}$

$$\ell_D\left(h\right) \leq \hat{\ell}_S\left(h\right) + \sqrt{\frac{1}{2\,m} \cdot \left(c \cdot N \cdot \ln d + \ln \frac{2}{\delta}\right)} \qquad w.\ p.\ at\ least\ 1-\delta$$

where c is a constant

$$\hat{h} = arg \min_{h \in H} \left(\hat{\ell}_S\left(h\right) + \sqrt{\frac{1}{2\,m} \cdot \left(c \cdot N \cdot \ln d + \ln \frac{2}{\delta}\right)} \right)$$

where $m >> N \cdot h \cdot \ln d$
If training set size is very small then you should not run this algorithm.

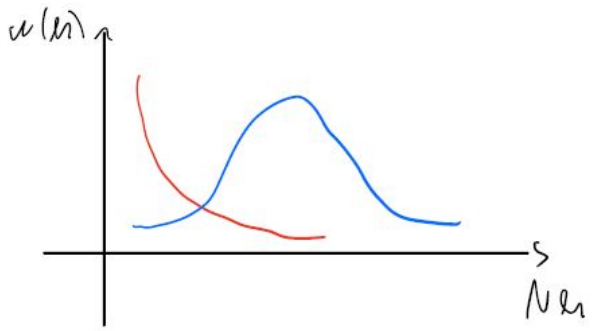


Figure 9.3: Algorithm for tree predictors

This blue curve is an alternative example. We can use Information criterion.

As I increase the number of nodes, N_h decrease so fast. You should take a smaller tree because it gives you a better bound. It's a principle known as Occam Razor (if I have two tree with the same error, if one is smaller than the other than i should pick this one).

Having N^*

Lecture 10 - 07-04-2020

10.1 TO BE DEFINE

10.2 MANCANO 20 MINUTI DI LEZIONE

$$\mathbb{E}[z] = \mathbb{E}[\mathbb{E}[z \mid x]] \qquad \longrightarrow \qquad \mathbb{E}[Z \mid X = x]$$

$$\mathbb{E}[X] = \sum_{t=1}^{m} \mathbb{E}[x \cdot \Pi(At)] \qquad A_1, ..., A_m \text{ portion of sample law of total probability}$$

$x \in \mathbb{R}^d \qquad \mathbb{P}(Y_{\Pi(s,x)} = 1) = \mathbb{E}\left[\Pi Y_{\Pi(s,x)} = 1\right] = \qquad$ Law of total probability

$$= \sum_{t=1}^{m} \mathbb{E}(\Pi\{Y_t = 1\} \cdot \Pi \cdot \{\Pi(s,x) = t\}] \quad =$$

$$= \quad \sum_{t=1}^{m} \mathbb{E}[\mathbb{E}[\Pi\{Y_t = 1\} \cdot \Pi \cdot \{\Pi(s,x) = t\} \mid X_t]] =$$

given the fact that $Y_t \sim \eta(X_t) \Rightarrow$ give me probability

$Y_t = 1$ *and* $\Pi(s,x) = t$ *are independent given* $X_Y \quad (e.g.\ \mathbb{E}[Z\,X] = \mathbb{E}[x] \cdot \mathbb{E}[z]\ )$

$$= \sum_{t=1}^{m} \mathbb{E}[\mathbb{E}[\Pi\{Y_t = 1\} \mid X_t] \cdot \mathbb{E}[\Pi(s,x) = t | Xt]] = \sum_{t=1}^{m} \mathbb{E}[\eta(X_t) \cdot \Pi \cdot \{\Pi(s,x) = t\}] =$$

$$= \quad \mathbb{E}\left[\eta\left(X_{\Pi(s,x)}\right)\right]$$

$$\mathbb{P}(Y_{\Pi(s,x)} | X = x = \mathbb{E}[\eta(X_{\Pi}(s,x))]$$

$$\mathbb{P}(Y_{\Pi(s,x)} = 1, y = -1) = \mathbb{E}\left[\Pi\{Y_{\Pi(s,x)} = 1\} \cdot \Pi\{Y = -1 | X\}\right] =$$

$$= \mathbb{E}\left[\Pi\{Y_{\Pi(s,x)} = 1\} \cdot \Pi\{y = -1\}\right] = \mathbb{E}\left[\mathbb{E}\left[\Pi\{Y_{\Pi(s,x)} = 1\} \cdot \Pi\{y = -1 | X\}\right]\right] =$$

by independence i can split them

$$Y_{\Pi(s,x)} = 1 \qquad y = -1 \qquad \text{which is } 1 - \eta(x) \quad when \quad X = x$$

$$= \mathbb{E}\left[\,\mathbb{E}\left[\,\Pi\{Y_\Pi(s,x)\} = 1|X\,\right]\cdot\mathbb{E}\left[\,\Pi\{y=-1\}|X\,\right]\right] = \mathbb{E}\left[\,\eta_{\Pi(s,x)}\cdot(1-\eta(x))\,\right] =$$

similarly:

$$\mathbb{P}\left(Y_{\Pi(s,x)} = -1, y = 1\right) = \mathbb{E}\left[\,(1-\eta_{\Pi(s,x)})\cdot\eta(x)\,\right]$$

$$\mathbb{E}\left[\,\ell_D(\hat{h}_s)\,\right] = \mathbb{P}\left(Y_{\Pi(s,x)} \neq y\right) = \mathbb{P}\left(Y_{\Pi(s,x)} = 1, y = -1\right) + \mathbb{P}\left(Y_{Pi(s,x)} = -1, y = 1\right)$$
$$= \mathbb{E}\left[\,\eta_{\Pi(s,x)}\cdot(1-eta(x))\right] + \mathbb{E}\left[\,(1-\eta_{\Pi(s,x)})\cdot\eta(x)\,\right]$$

Make assumptions on D_x *and* η**:**

1. $\forall X$ drawn from $D_x \quad \max|X_t| \leq 1$
   Feature values are bounded in $[-1,1]$
   all my points belong to this:
   $$X = [-1,1]^d$$
2. η is such that $\exists c < \infty$:
   $$\eta(x) - \eta(x') \;\leq\; c\cdot\|X - x'\| \quad \forall\, x, x' \in X$$
   It means that η is **Lipschitz** in X $c < \infty \Leftrightarrow \eta$ is continous

using two facts:

$$\eta(x') \leq \eta(x) + c\,||X - x'|| \quad \longrightarrow \quad \textbf{euclidean distance}$$
$$1 - \eta(x') \leq 1 - \eta(x) + c\,||X - x'||$$

$X' = X_{\Pi(s,x)}$

$$\eta(X)\cdot(1-\eta(x')) + (1-\eta(x))\cdot\eta(x') \leq$$
$$\leq \eta(x)\cdot((1-\eta(x)) + \eta(x)\cdot c\,||X - x'|| + (1-\eta(x))\cdot c\,||X - x'|| =$$
$$= 2\cdot\eta(x)\cdot(1-\eta(x)) + c\,||X - x'||$$

$$\mathbb{E}\left[\ell_d\cdot(\hat{h}_s)\right] \leq 2\cdot\mathbb{E}\left[\,\eta(x) - (1-\eta(x))\,\right] + c\cdot(E)\left[\,||X - x_{\Pi(s,x)}||\,\right]$$

where $\leq$ mean at most

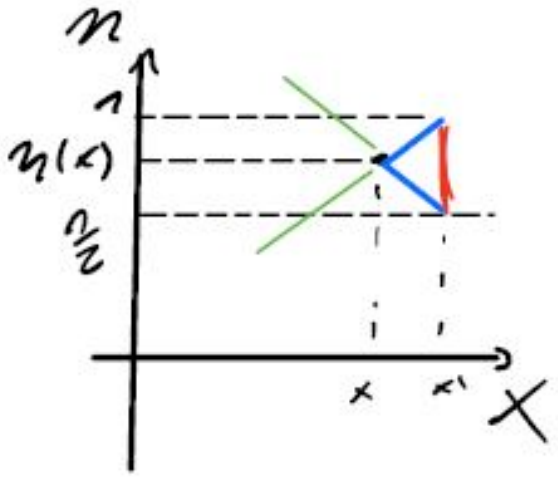


Figure 10.1: Point (2) - where $y = cx + q$ $y = -cx + q$

10.3 Compare risk for zero-one loss

$$\mathbb{E}\left[\min\{\eta(x), 1-\eta(x)\}\right] = \ell_D(f^*)$$

$$\eta(x)\cdot(1-\eta(X)) \;\leq\; \min\{\,\eta(x), 1-\eta(x)\,\} \quad \forall x$$

$$\mathbb{E}\left[\eta(x)\cdot(1-\eta(x)\right] \;\leq\; \ell_D(f^*)$$

$$\mathbb{E}\left[\,\ell_d(\hat{l}_s)\,\right] \leq 2\cdot\ell_D(f^*) + c\cdot\mathbb{E}\left[\,\|X - X_{\Pi(s,x)}\|\,\right] \eta(x)\in\{0,1\}$$

Depends on dimension: **curse of dimensionality**

$\ell_d(f^*) = 0 \iff \min\{\eta(x), 1-\eta(x)\} = 0$ with probability $= 1$
to be true $\eta(x) \in \{0,1\}$

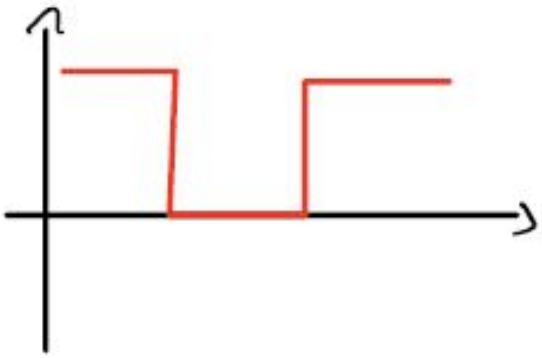


Figure 10.2: Point

Lecture 11 - 20-04-2020

11.1 Analysis of K_{NN}

$$\mathbb{E}\left[\ell_D(\hat{\ell}_s)\right] \leq 2 \cdot \ell_D\left(f^*\right) + c \cdot \mathbb{E}\left[\, \|X = x_{\Pi(s,x)}\| \,\right]$$

At which rate this thing goes down? If number of dimension goes up then a lot of point are far away from X.
So this quantity must depend on the space in which X live.
Some dependence on number of depends and incresaing number of traning points close to X
This expecation is fucniton of random variable X and $X_{\pi(s,x)}$

We are going to use the assumption that:
$|X_t| \leq 1 \qquad \forall$ cordinates $i = 1, ..., d$

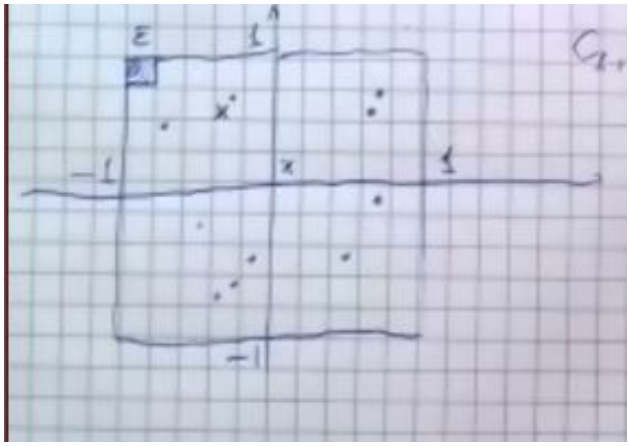


Figure 11.1: Example of domain of K_{NN}

Hyper box in bydimension. All point live in this box and we exploit that. Look at the little suare in which is divided and we assume that we are dividing the box in small boxes of size ε. Now the training points will be a strincle of point distributed in the big square.
Our training points are distribuited in the box (this is our S).
Now we added a point x and given this two things can happned: falls in the square with training points or in a square without training points.
What is going to be the distance $X_{\pi(s,x)}$ in this two cases?
We have c_1 up to c_r How big is this when we have this two cases? (We

lookjing at specific choices of x and s)

$$\|X - X_{s,x}\| \leq \begin{cases} \varepsilon\sqrt{d} & C_i \cup S \neq 0 \\ \sqrt{d} & C_i \cup S = 0 \end{cases}$$

were $X \in C_i$
We have to multiply by the lenght of the cube. Will be $\varepsilon\sqrt{d}$

Figure 11.2: Diagonal length

If things go badly can be very far away like the length of the domain. Lenght is 2 and diagonal is $\sqrt{d}$
if close they are going to be $\varepsilon close$ or far as domain.
We can split that the expression inside the expectation according to the two cases.

$$\mathbb{E}\left[\,\|X - X_{\Pi(s,x)}\|\,\right] \leq$$

$$\mathbb{E}\left[\varepsilon\cdot\sqrt{d}\cdot\sum_{i=1}^{r} I\{X \in C_i\}\cdot I\{C_i \cap S \neq 0\}\right] + 2\cdot\sqrt{d}\cdot\sum_{i=1}^{r} I\{X \in C_i\}\cdot I\{C_i \cap S \neq 0\} =$$

$$= \varepsilon\cdot\sqrt{d}\cdot\mathbb{E}\left[\sum_{t=1}^{r} I\{X \in C_i\} I\cdot\{C_i \cap S \neq 0\}\right] + 2\cdot\sqrt{d}\cdot\sum_{i=1}^{r}\mathbb{E}\left[\,I\{X \in C_1\}\cdot I\{C_1 \cap S \neq 0\}\,\right] \leq$$

I don't care about this one $\sum_{t=1}^{r} I\{X \in C_i\}\cdot I\{C_i \cap S \neq 0\}$
Can be either 0 or 1 (if for some i, X belong to some C_i
So at most 1 the expectation

$$\leq \varepsilon\cdot\sqrt{d} + \square$$

We can bound this square. Are the event I in the summation of the term after $+$. If they are indepednt the product will be the product of the two expectation. If I fix the cube. X and S are independent.
Now the two events are independent
$X \in C_i$ **is independent of** $C_i \cap S \neq$

$$\mathbb{E}\left[\,I\{X \in C_i\}\cdot I\{C_i \cap S \neq 0\}\right] = \mathbb{E}\left[I\{X \in C_i\}\right]\cdot\mathbb{E}\left[I\{C_i\}\right]$$

MANCAAAAAAA 9.26

$$\mathbb{P}\left(C_i \cap S\right) = \left(1 - \mathbb{P}\left(X \in C_1\right)\right)^m \leq \exp(-m \cdot \mathbb{P}(x \in C_1))$$

The probability of the point fall there and will be the probability of falling in the cube.
Probability of Xs to fall in the cube with a m (samples?)
Now use inequality $(1-p)^m \in e^{-pm} \longrightarrow 1 + x \leq e^x$

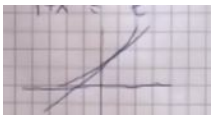


Figure 11.3: Shape of the function

$$\sum_{t=1}^{r} \mathbb{E}\left[\mathbb{P}\left(X \in C_1\right) \cdot \mathbb{P}\left(C_1 \cap S \neq\right)\right] \leq \sum_{i=1}^{r} p_i \cdot e^{-m\,p_i} \leq$$

given that $p_i = \mathbb{P}(X \in C_i)$ I can upper bound this

$$\leq \sum_{t=1}^{r} \left(\max_{0 \leq p \leq 1} p\, e^{-m\,p} \right) \leq r \max_{0 \leq p \leq 1} p\, e^{-m\,p} =$$

where $p\,e^{-m\,p}$ is $F(p)$ it is concave function so i'm going to take first order derivative to maximise it.

$$F'(p) = 0 \Leftrightarrow p = \frac{1}{m} \quad \textit{check!}$$
$$F''(p) \leq 0$$

Check this two condition!

$$= \frac{r}{e\,m}$$

Now get expectation

$$\mathbb{E}\left[\, \|X - X_{\Pi(s,x)}\| \,\right] \leq \varepsilon \cdot \sqrt{d} + \left(2 \cdot \sqrt{d}\right) \frac{r}{e\,m} =$$

I have $(\frac{2}{\varepsilon})^2$ squares. This bring ε in the game

$$\varepsilon \cdot \sqrt{d} + \left(2 \cdot \sqrt{d}\right) \frac{1}{e\,m} \cdot \left(\frac{2}{\varepsilon}\right)^d =$$

$$= \sqrt{d}\left(\varepsilon + \frac{2}{e\,m}\cdot\left(\frac{2}{\varepsilon}\right)^{d}\right)$$

HE MISS THE "c" costant from the start we can choose ε to take them balanced
set $\varepsilon = 2\,m^{\frac{-1}{(d+1)}}$

$$\left(\varepsilon + \frac{2}{e\,m}\right)\cdot\left(\frac{2^{d}}{\varepsilon}\right) \leq 4\,m^{\frac{-1}{(d+1)}} =$$

$$\mathbb{E}\left[\ell_d(\hat{h}_s)\right] \;\leq\; 2\ell_d(f^*) + 4\cdot c\cdot\sqrt{d}\cdot m^{-\frac{1}{d+1}}$$

We have that:
if $m \longrightarrow \infty \quad \ell_D(f^*) \;\leq\; \mathbb{E}\left[\ell_D(\hat{h}_s\right]) \;\leq\; 2\ell_D(f^*)$
I want this smaller than twice risk + some small quantity

$$\mathbb{E}\left[\ell_d(\hat{h}_S)\right] \;\leq\; 2\ell_D(f^*) + \varepsilon$$

How big m ?
Ignore this part since very small $(4\cdot c\cdot\sqrt{d})$

$$m^{-\frac{1}{d+1}} \;\leq\; \varepsilon \Leftrightarrow m \;\geq\; \left(\frac{1}{\varepsilon}\right)^{d} + 1$$

So 1-NN require a training set size exponential "accuracy" $1-\varepsilon$

We show that $1-NN$ can approach twice based risk $2\cdot\ell_D(f^*)$
but it takes a training set exponential in d.

11.1.1 Study of K_{NN}

Maybe we can use the K_{NN}.

$$\mathbb{E}\left[\ell_D(\hat{h}_s)\right] \leq \left(1+\sqrt{\frac{8}{k}}\right)\ell_D(f^*) + 0\;\left(k\,m^{-\frac{1}{d+1}}\right)$$

So is not exponential here.
Learning algorithm A is consistent for a certain loss ℓ
If $\forall\, D$(distribution) of data we have that $A(S_m)$ predictor output by A
Now have the risk of that in $\ell_D(A(S_m))$ and we look at the expectation $\mathbb{E}\left[\ell_D(A(S_m))\right]$

If we give a training set size large ($\lim_{m\to\infty} \mathbb{E}\left[\ell_D(A(S_m))\right] = \ell_D(f^*)$) risk will converge in based risk.

K_{NN} where $K = K_m$ (is a function of training set size). $K_, \to \infty$ as $m \to \infty$. Only way K goes to infinity is sublinearly of training set size. (infinity but so as quicly as m $K_m = O(m)$

For instance $K_m = \sqrt{m}$
Then:

$$\lim_{m\to\infty} \mathbb{E}\left[\ell_D\left(A'\left(S_m\right)\right)\right] = \ell_D(f^*) \qquad \textbf{where } A' \textbf{ is } K_m\textbf{-}NN$$

Increasing the size we will converge to this base risk for any distribution and that's nice.

11.1.2 study of trees

Algorithm that grow tree classifiers can also be made consistent provided two condition:

- The tree keeps growing
- A non-vanishing fraction of traning example is routed to each leaft

Tree has to keep growing but not so fast.
Second point is: suppose you have a certain number of leaves and you can look at the fraction. Each leaf ℓ gets N_ℓ examples. You want that this fraction at any point of time is not going to 0. The fraction of point every leaf receive a split we are reducing the smallest number of examples.
Example keep growing and leaves too and we want that $\frac{N_\ell}{manca}$ this not going to 0. . since not showed the formula.

Given A, how do I know wheter A could be consistent?

$$H_A \equiv \{\ h\ :\ \exists S\ A(S) = h\}$$

S can be any size. If A is ERM then $H_A = H$, so where ERM minimise it.
If $\exists f^* : X \longrightarrow$ such that $f^* \notin H_A$ and $\exists D$ such that f^* is Bayes optimal for some distribution D.
This cannot be consistent because distribution will not be able to generate the Bayes optimal predictor. Maybe is there another predictor f which is

not equal to f^* risk.

What's the intuition?
Every time A is such that H_A is "restricted" in some sense, then A cannot be consistent. (e.g ERM).

Another way of restricting? Could be tree classifiers with at most N nodes (bound number of nodes).
How do i know N is enought to approximate well f^*. I want to converge the risk of f^*.
We can introduce a class of algorithm potentially consistent in which space predictor is not restricted.

11.2 Non-parametric Algorithms

When they are potentially consistent.
What does it mean?
Non-parametric algorithm have the potential of being consistent and do we know if algorithm is parametric or not?
A is non-parametric if:

- the description of $A(S_m)$ grows with m

Your predictor is a function and let's assume i can store in any variable a real number with arbitrary precition.

Any algorithm with bias is incosistent. So ability to converge to base risk is this.
How do i know if i have bias or not? this is where non parametric algorithm came.
Let's consider K_{NN}, how i can describe it? I have to remember distance is maded by training points and if i give you more S the m will increase. So this is parametric.
More training set for tree, then will grow more, even more larger will be ever growing more.
Any algorithm as a give training points is no parametric, while growing with parametric will stop a some point.

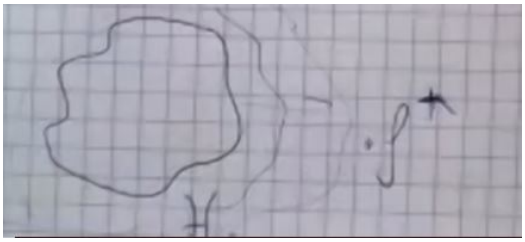


Figure 11.4: Parametric and non parametric growing as training set getting larger

If algorithm is more parametric as i give training points
If a certain point stop growing, f^* will be out and i will grow more.
If algorithm is able to generate — MANCA — Then the algorithm is non-parametric and can be potentially consistent and incluse f^* as it grows.
If set of predictor stops because I'm not enlarging my set of predictor since description of algortim will not depend on training size at some point $\rightarrow$ to be consistent.
If bias vanashes as i increase the S, then i can be consistent. I generating predictor that description depends on how much points i give them.

Parametric is not precise as consistency.
One class of algorithm that has consistency has a predicotr size growing with S growing.
Definition of non parametric is more fuzzy, consistency is precise (we demonstrate that mathematically).

11.2.1 Example of parametric algorithms

Neural network is parametric since i give structure of the network. If i give S small or big S my structure will be the same (will fit better on the training points).
Other example are algorithm with linear classifier in which number of parameter are just the idmension of the space.

Lecture 12 - 21-04-2020

12.1 Non parametrics algorithms

We talk about **consistency**: as the training size grows unbounded the expected risk of algorithms converge to Bayes Risk.

Now we talk about **non parametric algorithm**: the structure of the model is determined by the data.
Structure of the model is fixed, like the structure of a Neural Network but in non parametric algorithm will change structure of the model as the data grows (K_{NN} and tree predictor).
If I live the tree grow unboundenly then we get a non parametric tree, but if we bound the grows then we get a parametric one.

The converve rate of Bayes Risk (in this case doubled) was small. Converge of 1-NN to $2\,\ell_D(f^*)$ is $m^{-\frac{1}{d+1}}$ so we need an esponential in the dimension. And we need this is under Lips assumption of η.
It's possible to converge to Bayes Risk and it's called **No free lunch**.

12.1.1 Theorem: No free lunch

Let a sequenece of number $a_1, a_2 \ldots \in \mathbb{R}$ such that they converge to 0.
Also $\frac{1}{22222222} \geq a_1 \geq a_2 \geq \ldots \; \forall A$ for binary classification $\exists D$ s. t.
$\ell_D(f^*) = 0$ (zero-one loss) so Bayes risk is zero and $\mathbb{E}[\,\ell_D\,(A(S_M))\,] \geq a_m \quad \forall m \geq 1$
Any Bayes Optimal you should be prepared to do so on long period of time.
This means that:

- For specific data distribution D, then A may converge fast to Bayes Risk.
- If η is Lipschitz then it is continous. This mean that we perturb the input by the output doesno't change too much.
- If Bayes Risk is 0 ($\ell_D(f^*) = 0$) function will be discontinous

This result typically people think twice for using consistent algorithm because

I have Bayes risk and some non conssitent algorithm that will converge to some value ($\ell_D(\hat{h}^*)$). Maybe i have Bayes risk and the convergence takes a

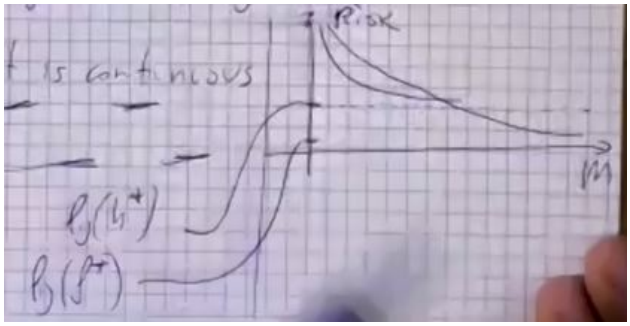


Figure 12.1: Tree building

lot on increasing data points. Before converging was better non parametric (?..)

Picture for binary classification, (similar for other losses)

- Under no assumption on η, the typicall "parametric" converge rate to risk of best model in H (including ERM) is $m^{-\frac{1}{2}}$. (Bias error may be high)
- Under no assumption on η there is no guaranteed convergence to Bayes Risk (in general) and this is **no-free-lunch** that guaranteed me no convergence rate.
- Under Lipshtiz assunption on η the typical non parametric convergence to Bayes Risk is $m^{-\frac{1}{d+1}}$. This is exponentially worst than the parametric convergency rate.

The exponential depencendece on d is called **Curse of dimnsionality**.
But if I assume small number of dimension $\longrightarrow$ K_{NN} is ok if d is small (and η is "easy")
If you have a non parametric algorithm (no Bayes error but may have exponentially infinity training error). I want them to be balanced and avoid bias and variance. We need to introduce a bit of bias in controlled way.
Inserting bias to reducing variance error. So we sacrify a bit to get a better variance error.

It could be good to inject bias in order to reduce the variance error. In practise instead of having 0 training error i want to have a larger training error and hope to reduce overfitting sacrifing a bit in the training error.
I can increase bias in different technics: one is the unsamble methods.

12.2 Highly Parametric Learning Algorithm

12.2.1 Linear Predictors

Our domain is Euclidean space (so we have points of numbers).

$$X \; is \; \mathbb{R}^d \qquad x = (x_1, .., x_d)$$

A linear predictor will be a linear function of the data points.

$$h : \mathbb{R}^d \longrightarrow Y \qquad h\,(x) = f(w^T\, x) \quad w \in \mathbb{R}^d$$

$$f : \mathbb{R} \longrightarrow Y$$

And this is the dot product that is

$$w^T\, x = \sum_{t=1}^{d} w_i x_i = \|w\|\, \|x\| \cos\Theta$$

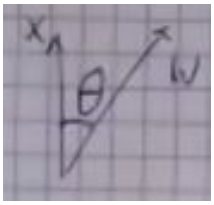


Figure 12.2: Dot product

Suppose we look a regression with square loss.

$$Y = \mathbb{R} \qquad h(x) = w^T\, x \quad w \in \mathbb{R}^d$$

$f^*(x)$ =$\mathbb{E}\,[\,Y|X = x\,]$
Binary classification with zero-one loss $Y = \{-1, 1\}$ We cannot use this since is not a real number but i can do:

$$h(x) = sgn\left(w^T\, x\right) \qquad sgn(x) = \begin{cases} +1 \; if \; z > 0 \\ -1 \; if \; z \leq 0 \end{cases}$$

where sgn is a sign function. Linear classifier.
$\|X\| \cos\Theta$ is the length of the projection of x onto w
Now let's look at this set:

$$\{x \in \mathbb{R}^d : w^T x = c\}$$

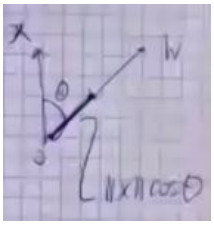


Figure 12.3: Dot product

This is a hyperplane.

$$\|w\|\|x\| \cos \Theta = c \qquad \|x\| \cos \Theta = \frac{c}{\|w\|}$$

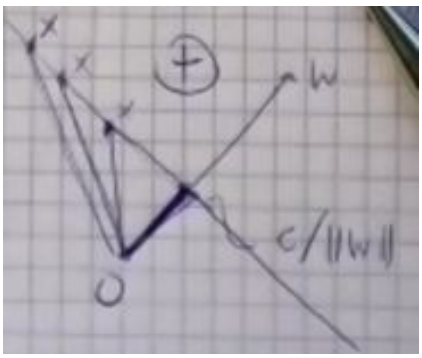


Figure 12.4: Hyperplane

So (w, c) describe an hyperplane.
We can do binary classification using the hyperplane. Any points that lives in the positive half space and the negative. So the hyperplane is splitting in halfs. $H \equiv \{x \in \mathbb{R}^d : w^T x = c\}$

$$H^+ \equiv \{x \in \mathbb{R}^d : w^T x > c\} \qquad \textbf{positive } h_s$$

$$H^- \equiv \{x \in \mathbb{R}^d : w^T x \leq\} \qquad \textbf{negative } h_s$$

$$h(x) = \begin{cases} +1 \; if \; x \in H^+ \\ -1 \; if \; x \notin H^- \end{cases} \qquad h(x) = sgn(w^T - c)$$

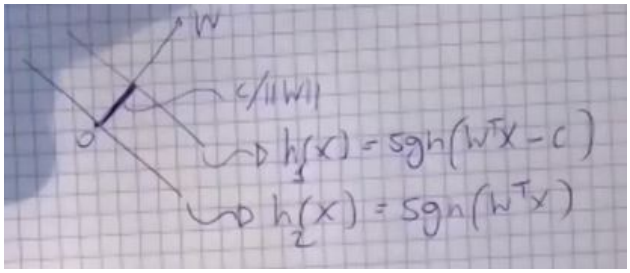


Figure 12.5: Hyperplane

h_1 is non-homogenous linear classifier.
h_2 is homogenous linear classifier. Any homogenous classifier is equivalent to

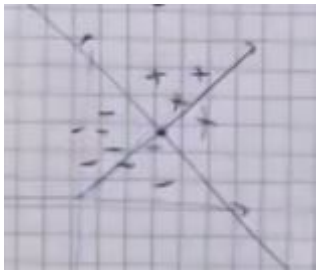


Figure 12.6: Hyperplane

this:

$$\{x \in \mathbb{R}^d : X = c\} \textbf{ is equivalent to } \quad \{x : \mathbb{R}^{d+1} : \nu^T x = 0\}$$

$$\nu = (w_1, .., w_d, -c) \qquad x' = (x_1, ..., x_d, 1)$$

So we added a dimension.

$$w^T x = c \;\Leftrightarrow\; \nu^T x' = 0$$

$$\sum_i w_1 x_1 = c \;\Leftrightarrow\; \sum_i w_1 x_1 - c = 0$$

Rule:
When you learn predictor just add an extra feature to your data points, set it ot 1 and forget about non- homogenous stuff.

One dimensional example

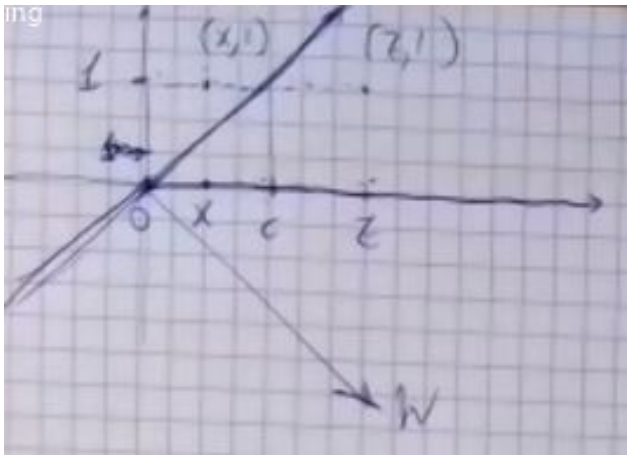


Figure 12.7: Example of one dimensional hyperplane

I have negative (left of $(x, 1)$ and positive point (left of $(z, 1)$ classified

Now i want to learn linear classifier. How can i do it?

$$H_d = \{\ h : \exists w \in \mathbb{R}^d\ h(x) = sgn(w^T x)\ \}$$

Parametric!
We expect high bias a low variance.

$$ERM \qquad \hat{h}_S = arg \min_{h \in H_d} \frac{1}{m} \cdot \sum_{t=1}^{m} I\{h(x_t) \neq y_t\} =$$

$$= \ arg \min_{w \in \mathbb{R}^d} \frac{1}{m} \cdot \sum_{t=1}^{m} I\{\, y_t\, w^T x_t \leq 0\,\}$$

A bad optimisation problem!

FACT:
It is unlikely to find an algorithm that solves ERM for H_d and zero-one loss efficiently.
NP completeness problems!
It's very unlikely to solve this problem.
This problem is called **MinDisagreement**

12.2.2 MinDisagreement

Instance: $(x_1, y_1)...(x_m, y_m) \in \{0,1\}^d \, x \, \{-1, 1\}, \quad k \in \mathbb{N}$
Question: Is there $w \in x \, D^d$
s.t $y_t \, w^T x_t \leq 0$ for at most k indices $t \in \{1, ...m\}$
This is NP-complete!

Lecture 13 - 27-04-2020

13.1 Linear prediction

We had ERM $\hat{h}$

$S = \{(x_1, y_1)...(x_n, y_n)\}$ $\quad x_t \in \mathbb{R}^d$ $\quad y_t \in \{-1, +1\}$ $\quad \ell_t(w) = I\{y_t w^T x_t \leq 0\}$

$$\hat{h}_S = arg \min_{h \in H_D} \frac{1}{m} \sum_{t=1}^{m} I\{y_t w^T x_t\} \leq 0$$

The associated decisio problem is a NP problem so cannot be camputed efficientiy unless $P \equiv NP$
Maybe we can approximate it, so a good solution that goes close to minimise error.
This is called MinDisOpt

13.1.1 MinDisOpt

Instance: $(x_1, y_1)...(x_n, y_n) \in \{0,1\}^d x \{0,1\}$
Solutio:

$w \in Q^D$**minimising the number of indices** $t = 1, ...m$ *s.t.* $h_t w^T x_t \leq 0$

$Opt(S)$ is the smallest number of mislcassified example in S by any linear classifier in H_D
where $\frac{Opt(S)}{m}$ is training error of ERM

Theorem : if $P \not\equiv NP$, then $\forall c > 0$ there are no polytime algorithms (with r. t. the input size $\Theta(m_d)$) that approximately solve every istance S of MinDisOpt with a number of mistakes bounded by $C \cdot Opt(S)$.
If I am able to approximate it correclty this approximation will grow with the size of the dataset.

$\forall A$ **(polytime)** *and* $\forall C \quad \exists S \qquad \hat{\ell}_S\left(A\left(S\right)\right) \geq c \cdot \hat{\ell}_S\left(\hat{h}_S\right)$ **(where** $\hat{h}_S$ **is** ERM**)**

$$Opt(S) = \hat{\ell}_S(\hat{h}_S)$$

This is not related with free lunch theorem (information we need to get base

error for some learning problem). Free lunch: we need arbitrarirally information to get such error. Here is we need a lot of computation to approximate the ERM.

Assume $Opt(S) = 0$ ERM has zero training error on S
$\exists U \in \mathbb{R}^d$ s.t. $\forall t = 1, ...m$ $\quad y_t U^T x_t > 0$ $\quad$ **S is linearly separable**

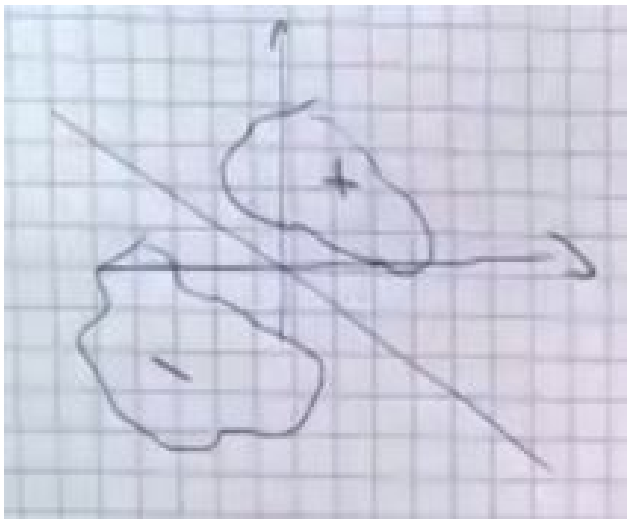


Figure 13.1: Tree building

We can look at the min

$$\min_{t=1,...m} y_t U^T x_t = \gamma(U) > 0$$ **We called this marginn of U on (x_t, y_t)**

We called in this way since $\frac{\gamma(U)}{\|U\|} = \min t y_t \|x_t\| cos(\Theta)$

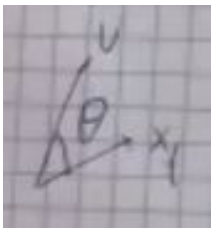


Figure 13.2: Tree building

where Θ is the angle

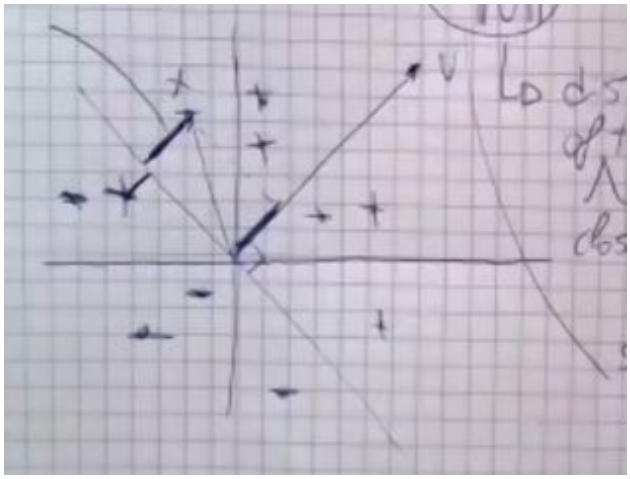


Figure 13.3: Tree building

$where \frac{\gamma(U)}{\|U\|}$ is the distance separating hyperplane on closest training example .

S linearly separable and if i look at the sistem of this linear inequality:

$$\begin{cases} y_t w_T x_t > 0 \\ y_m w_T x_m > 0 \end{cases}$$

We can solve it in polytime using a linear solver. So any package of linear programming, and will be solved in linear time.

This is called **feasibilty problem**. We want a point y that satisfy all my linear inequalities.

Figure 13.4: Feasibilty problem

When $Opt(S) = 0$ is we can implememtn ERM efficiently using LP (Linear programming).

They may overfitting since a lot of bias. When this condition of Opt is no satisfy we cannot do it efficiently. LP algorithm can be complicated so we figure out another family of algorithm.

13.2 The Perception Algorithm

This came from late '50s and was designed for psicology but have a general utility in othe fields.

Perception Algorithm
Input : training set $S = \{(x_t, y_t)...(x_m, y_m)\}$ $x_t \in \mathbb{R}^d$ $y_t \in \{-1, +1\}$
Init: $w = (0, ...0)$
Repcat
read next (x_t, y_t)
If $y_t w^T x_t \leq$ then $w \longleftarrow w + y_t x_t$
Until margin greater than 0 $\gamma(w) > 0$ // w separates S
Output w

We know that $\gamma(w) = \min_t y_t w^T x_t \leq 0$ The question is, will it terminate if S is linearly separable?
If $y_t w^T x_t \leq 0$, then $w \longleftarrow w + y_t x_t$

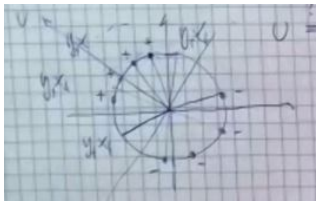


Figure 13.5:

For simplicity our x are in this circle. Some are on the circonference on top left with + sign and some in bottom right with − sign.
All minus flipped to the other side and the we can deal the +.
U is a separating hyperplane, how can i find it?
Maybe i can do something like the average:

$$U = \frac{1}{m} \sum_{t=1}^{m} y_t x_t \quad ?$$

But actually don't take the average of all of them. So do not take average of

all, instead take the one that satisfy $y_t w^T x_t \leq 0$ condition.
$y_t w^T x_t \leq 0$ is a violated consstraint and we want it > 0.
Does $w \longleftarrow w + y_t x_t$ fix it?

$$y_t(w + y_t \cdot x_t)^T x_t = y_t w^T x_t + \|x_t\|^2$$

We are trying to see what happen before and after the updates of w.
SInce $\|x_t\| > 0$ so is positive, the update increase margins, thus going towards fixing violated constraints.

13.2.1 Perception convergence Theorem

dated early 60s On a linearly separable S, perceptron will converge after at most M updates (when they touch in the figure) where:

$$M \leq \left(\min_{U\,:\,\gamma(U)=1} \|U\|^2 \right) \left(\max_{t=1,..m} \|x_t\|^2 \right)$$

Algorithm is not able to do that. ALgorithm keeps looking till he get a violating constraint and then stops. This is bounded by the number of loops.

We said that $\gamma(U) = \min_t y_t U^T x_t > 0$ when U is separator.

$$\forall t \quad y_t U^T x_t \geq \gamma(U) \quad \Leftrightarrow \quad \forall t \quad y_t \left(\frac{U}{\gamma(U)} \right)^T x_t \geq 1$$

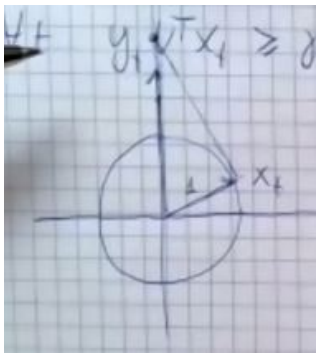


Figure 13.6:

If i rescale U i can make the margin bigger (in particolar > 1)

The shortest $min\|U\|$ s.t. $y_t U^T x_t \geq 1 \quad \forall t$

Proof:
W_m is local variable after M updates, I have zero vector $W_0 = (0, ...0)$
t_M is the index of training example that causes the M-th update.

We want to upper bound M (deriving upper and lower bound
on a certain quantity $\|W\|$ $\|U\|$)
where U is any s.t. $y_t U^T x_t \geq 1 \quad \forall t$

$$\|W_M\|^2 = \|W_{M-1} + y_{tM} x_{tM}\|^2 = \|W_{M-1}\|^2 + \|y_{tM} x_{tM}\|^2 + 2 \cdot y_{tM} W_{M-1}^T x_{tM} \; =$$

$$= \; \|W_{M-1}\|^2 + \|x_{tM}\|^2 + 2 \cdot y_{tM} W_{M-1}^T x_{tM} \; \leq$$

where $y_{tM} W_{M-1}^T x_{tM} \leq 0$

$$\leq \; \|w_{M-1}\|^2 + \|x_{tM}\|^2$$

$$\|W_M\|^2 \leq \|W_0\|^2 + \sum_{i=1}^{M} \|x_t\|^2 \leq M \; \left(\max_t \|x_t\|^2\right)$$

.....

.....

... MANCA ?????

$$\|W_M\| \; \|U\| \; \leq \; \|U\| \sqrt{M} \; \left(\max_t \|x_t\|\right)$$

since $\cos\Theta \in [-1, 1]$

$$\|W_M\| \; \|U\| \geq \|W_M\| \; \|U\| \; \cos\Theta = W_M^T U = \left(W_{M-1} + y_{tM} x_{tM}\right)^T U \; =$$

where last passage is the **Inner product**

$$W_{M-1}^T U + y_{tM} U^T x_{tM} \geq W_{M-1}^T U + 1 \geq W_0^T U + M = M$$

where $y_{tM} U^T x_{tM}$ is ≥ 1

$$M \leq \|W_M\| \; \|U\| \leq \|U\| \sqrt{M} \left(\max_t \|x_T\|\right)$$

$$M \leq \left(\|U\|^2\right) \left(\max_t \|x_t\|^2\right) \qquad \forall U \; : \; \min_t y_t U^t x_t \geq 1$$

$$M = \left(\min_{U \,:\, \gamma(U)=1} \|U\|^2\right) \left(\max_t \|x_t\|^2\right)$$

Some number depends on S
M can be exponential in md when the ball of positive and negative are very closer and the length of U is super long and exponential in D.
If dataset barely separable then perceptron will make a number of mistakes that is exponential in the parameter of the problem. U is a linear separator and has exponential length

Lecture 14 - 28-04-2020

14.1 Linear Regression

Yesterday we look at the problem of emprical risk minismisation for a linear classifier. 0-1 loss is not good: discontinuous jumping from 0 to 1 and it's dififcult to optimise. Maybe with linear regression we are luckier.
Our data point are the form (x, y) $x \in \mathbb{R}^d$ regression, $(\hat{y} - y)^2$ square loss.
We are able to pick a much nicer function and we can optimise it in a easier way.

14.1.1 The problem of linear regression

Instead of picking -1 or 1 we just leave it as it is.

$$h(c) = w^T\ x \qquad w \in \mathbb{R}^d \qquad x = (x_1, ..., x_d, 1)$$

$$\hat{w} = arg\min_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{t=1}^{m} (w^T\ x_t - y_t)^2 \qquad ERM\ for\ \ (x_1, y_1)...(x_m, y_m)$$

How to compute the minimum?
We use the vector v of linear prediction
$v = (w^T x_1, .., w^T x_m)$
and a vector of real valued labels
$y = (y_1, ..., y_m)$ where $v, y \in \mathbb{R}^m$

$$\sum_{t=1}^{m} (w^T x_t - y_t)^2 \ = \ \|v - y\|^2$$

S is a matrix.

$$s^T = [x_1, ..., x_m] \quad d \times m \qquad v = sw = \begin{bmatrix} x_1^t \\ ... \\ x_m^T \end{bmatrix} \begin{bmatrix} w \end{bmatrix}$$

So:

$$\|v - y\|^2 = \|sw - y\|^2$$

$$\hat{w} = arg\min_{w \in \mathbb{R}^d} \|sw - y\|^2 \qquad \textbf{where } sw \textbf{ is the design matrix}$$

$$F(w) = \|sw - y\|^2 \qquad \text{is convex}$$

$$\nabla F(w) = \cancel{2} s^T (sw - y) = 0 \qquad s^T \; sw = s^T y$$

where s^T is $d \times m$ and s is $m \times d$ and $d \neq m$
If $s^T \; s$ invertible (non singular) $\hat{w} = (s^T \; s)^{-1} \; s^T \; y$
And this is called **Least square solutions (OLS)**

We can check $s^T \; s$ is non-singular if $x_1, ..., x_m$ span $\mathbb{R}^d$

$s^T \cdot s$ may not be always invertible. Also Linear regression is high bias solution. ERM may underfit since linear predictor introduce big bias.
$\hat{w} = (s^T \cdot s)^{-1} \cdot s^T \cdot y$ is very instable: can change a lot when the the dataset is perturbed.
This fact is called **instability** : variance error
It is a good model to see what happens and then try more sofisticated model.
Whenever $\hat{w}$ is invertible we have to prove the instability. But there is a easy fix!

14.1.2 Ridge regression

We want to stabilised our solution. If $s^T \cdot s$ non-singular is a problem.

We are gonna change and say something like this:

$$\hat{w} = arg \min_w \|s \cdot w - y\|^2 \quad \rightsquigarrow \hat{w}_\alpha = arg \min_w \left(\|s \, w - y\|^2 + \alpha \cdot \|w\|^2 \right)$$

where α is the **regularisation term.**

$\hat{w}_\alpha \to \hat{w}$ for $\alpha \to 0$
$\hat{w}_\alpha \to (0, ..., 0)$ for $\alpha \to \infty$

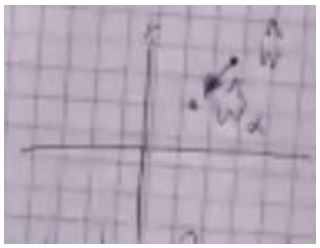


Figure 14.1:

$\hat{w}_\alpha$ has more bias than $\hat{w}$, but also less variance

$$\nabla \left(\|s\, w - y\|^2 + \alpha \, \|w\|^2 \right) \;=\; 2 \, \left(s^T \, s \, w - s^T \, y \right) + 2 \, \alpha \, w = 0$$

$$\left(s^T \, s + \alpha \, I \right) \, w = s^T \, y$$

$$(d \times m) \, (m \times d) \; (d \times d) \; (d \times m) \qquad (d \times m) \; (m \times 1)$$

where I is the identity

$$\hat{w}_\alpha = \left(s^T \, s + \alpha \, I \right)^{-1} \, s^T \, y$$

where $y_1, ..., y_\alpha$ are eigen-values of $s^T \, s$
$y_1, ..., y_\alpha + \alpha > 0$ eigenvalues of $s^T \, s + \alpha I$
In this way we make it positive and semidefinite.
We can always compute the inverse and it is a more stable solution and stable means **do not overfit**.

14.2 Percetron

Now we want to talk about algorithms.
Data here are processed in a sequential fashion one by one.
Each datapoint is processed in costant time $\Theta \, (d)$
(check $y_t \, w^T \leq 0$ and in case $w \leftarrow w + y_t \, x_t$) and the linear model can be stored in $\Theta(d)$ space.
Sequential processing scales well with the number of datapoints.
But also is good at dealing with scenarios where new data are generated at all times.
Several scenario like:

- Sensor data
- Finantial data
- Logs of user

So sequential learning is good when we have lot of data and scenario in which data comes in fits like sensor.
We call it **Online learning**

14.2.1 Online Learning

It is a learning protocol and we can think of it like Batch learning. We have a class H of predictors and a loss function ℓ and we have and algorith that outputs an initial default predictor $h_1 \in H$.

For $t = 1, 2...$
1) Next example (x_t, y_t) is observed
2) The loss $\ell(h_t(x_t), y_t)$ is observed $\qquad (y_t\, w^T\, x_t \leq 0)$
3) The algorithm updates h_t generating h_{t+1} $\qquad (w \leftarrow w + y_t\, x_t)$

The algorithm generates s sequence $h_1, h_2, ...$ of models
It could be that $h_{t+1} = h_t$ occasionally
The update $h_t \to h_{t+1}$ is **local** (it only uses h_t and (x_t, y_t))
This is a batch example in which take the training set and generate a new example.

$$(x_1, y_1) \to A \to h_2$$

$$(x_1, y_1)(x_2, y_2) \to A \to h_3$$

But if I have a non-learning algorithm i can look at the updates:

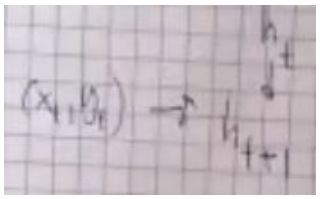


Figure 14.2:

This is a most efficient way and can be done in a costant time. The batch learning usually have single predictor while the online learning uses a sequence of predictors.

How do I evaluate an online learning algorithm A? I cannot use a single model, instead we use a method called **Sequential Risk**.
Suppose that I have $h_1, h_2...$ on some data sequence.

$$\frac{1}{m} \sum_{t=1}^{T} \ell(h_t(x), y_t) \qquad \textit{as a function of } T$$

The loss on the next incoming example.

I would like something like this:

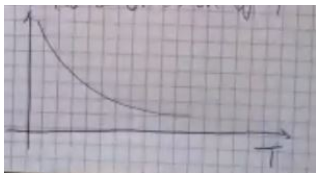


Figure 14.3:

We need to fix the sequence of data: I absorb the example into the loss of the predictor.

$$\ell(h_t(x), y_t) \longrightarrow \ell_t(h_t)$$

I can write the sequential risk of the algorithm:

$$\frac{1}{m}\sum_{t=1}^{T}\ell_t(h_t) - \min_{h\in H}\frac{1}{m}\sum_{t=1}^{T}\ell_t(h)$$

So the sequencial risk of the algorithm - the sequential risk of best predictor in H (up to T).
This is a sequential similar of variance error. $\longrightarrow$ is called **Regret**.

$$h_T^* = arg\min_{h\in H}\frac{1}{T}\sum_t \ell_t(h) \qquad \frac{1}{T}\ell_t(h_t) - \frac{1}{T}\sum_t \ell_t(h_T^*)$$

14.2.2 Online Gradiant Descent (OGD)

It is an example of learning algorithm.
In optimisation we have one dimension and we want to minimise the function i can compute the gradiant in every point.
We start from a point and get the derivative: as I get the derivative I can see if is decreasing or increasing.

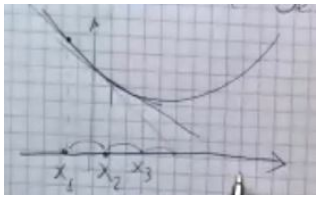


Figure 14.4:

$$f \quad convex \qquad \min_x f(x) \qquad f\ \mathbb{R}^d \to \mathbb{R}$$

$$x_{t+1} = x_t + \eta \nabla f(x_t) \qquad \eta > 0$$

$$w_{t+1} = w_t + \eta\, \nabla \ell_t(w_t)$$

where η is the learning rate.

$h(x) = w^T\, x \qquad \ell_t(w) = \ell(w^T\, x_t, y_t) \qquad for\ istance\ \ \ell(w^T\, x_t, y_t) = (w^T\, x_t - y_t)^2$

Assumption ℓ_t is convex (to do optimisation easily) and differentiable (to compute the gradiant)

Lecture 15 - 04-05-2020

15.1 Regret analysis of OGD

We introduce the **Regret**.

$$\frac{1}{m}\sum_{t=1}^{T}\ell_t(w_t) - \frac{1}{T}\sum_{t=1}^{T}\ell_t(u_t^*)$$

$$(x_1, y_1)...(x_t, y_t) \qquad \ell_t(w) = \left(w^T x_t - y_t\right)^2$$

we build a loss function for example with the square loss.
The important thing is that $\ell_1, \ell_2, ...$ is a sequence of **convex losses**.

In general we define the regret in this way:

$$R_T(u) \;=\; \frac{1}{m}\sum_{t=1}^{T}\ell_t(w_t) - \frac{1}{T}\sum_{t=1}^{T}\ell_t(u_t)$$

The Gradiant descent is one of the simplest algorithm for minimising a convex function. We recall the iteration did by the algorithm:

$$w_{t+1} \leftarrow w_t - \eta_t \nabla f(w_t) \qquad \eta_t > 0 \textbf{ learning rate} \quad f \textit{ convex}$$

$$f : \mathbb{R}^d \to \mathbb{R} \textit{ that's why use the gradiand instead of the derivative}$$

Learning rate can depend on time and we approach the region of the function f where the region is 0. We keep on moving in the X axes in the direction where the function is decreasing.

15.1.1 Projected OGD

2 parameters: $\eta > 0$ and $U > 0$
Initialisation: $w_1 = (0, ..., 0)$
For $t = 1, 2, ...$
1)**Gradiant step**: $w'_{t+1} = w_t - \frac{\eta}{\sqrt{t}} \nabla \ell_t(w_t)$ $(x_t, y_t) \rightsquigarrow \ell_t$

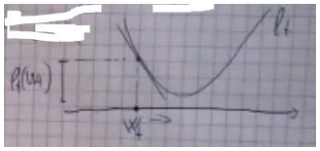


Figure 15.1:

2) Projection step:

Lecture 16 - 05-05-2020

Lecture 17 - 11-05-2020

Lecture 18 - 12-05-2020

18.1 Kernel functions

We use a notion of feature expansion. They are different but somehow they reach something similar. In fact Linear classifier have high bias.
Linear predictor use hyper plane as basic brick to build prediction.

18.1.1 Feature expansion

Given $\phi : \mathbb{R}^d \longrightarrow V$ $\quad$ V is typically $\mathbb{R}^N$ $\quad$ $N >> d$

For example:
$d = 2 \quad N = 6 \quad \phi = \mathbb{R}^2 \longrightarrow \mathbb{R}^6$

$$\phi(x_1, x_2) = (1, x_1^2, x_2^2, x_1, x_2, x_1 x_2)$$

We have a homogenous hyper plane.
$w \in \mathbb{R}^6 \ \{z \in \mathbb{R}^6 : w^T z = 0\}$ $\quad$ $z = \phi(x)$ $\quad$ $x \in \mathbb{R}^2$

$$\forall x \in \mathbb{R}^2 \qquad w^t \phi(x) = w_1 + w_2 x_1^2 + w_3 x_2^2 + w_4 x_1 + w_5 x^2 + w_6 x_1 x_2 = 0$$

$$w^T \phi(x) = 0$$

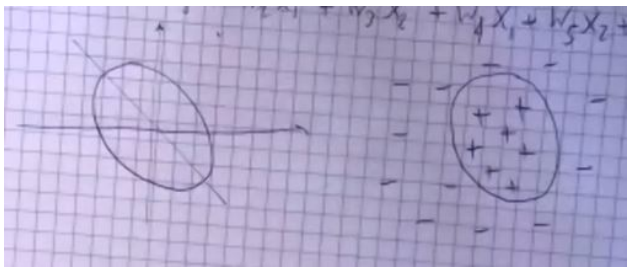


Figure 18.1:

$$\phi : \mathbb{R}^d \longrightarrow \mathbb{R}^N \qquad \Pi_{s=1}^M x_{Vs} \quad v \in \{1, ...d\}^k \quad k = 0, ..., n$$

$$h(x) = sgn(w^T \phi(x)) \qquad w^T \phi(x) = \sum_{i=1}^{N} w_i \phi(x)_i$$

The problem of this feature expansion is the degree of the monomials!

$$N = \sum_{i=0}^{n} |\{1, ...d\}^k| = \sum_{k=0}^{n} d^k = \frac{d^{n+1} - 1}{d - 1} = \Theta(d^n)$$

So it's exponential! But this feature expansion can be implemented in a efficient way.

18.1.2 Kernels implements feature expansion (Efficiently

$w^T\phi(x)$ Perception $w \leftrightarrow w + y_t x_t I\{y_t w^T x_t \leq 0\}$ MANCA quadlcosa

$$w = \sum_{s \in S} y_s x_s \rightsquigarrow \sum_{s \in S} y_s \phi(x_s)$$

where S is a subset of traning set where updates occurred.
Every time i make a mistake i add some of this product of data points.
If I run this using example that are images accourding to some feature expansion map (ϕ), I will get the perceptron after the mapping.

$$w^T\phi(x) = \sum_{s \in S} y_s \phi(x)^T \phi(x_s)$$

It's a inner problem and can have exponentially degree of the component.
Kernels help me compute this inner product $\phi(x)^T\phi(x_s)$ quickly

$$\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^6 \qquad \phi(x_1, x_2) = (1,\ x_1^2,\ x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

$$\phi(x)^T\phi(z) \ = \ 1 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_2z_2 + 2x_1x_2z_1z_2 \ = \ (1 + x^Tz)^2 \ = \ k(x, z)$$

$$w^T\phi(x) = \sum_{s \in S} y_s \ k(x, x_2)$$

$k(x, z)$ implements $\phi(x)^T\phi(z)$ $\forall x, z$ and ϕ defined as before

How to we generalise this?

$$k_n(x, x') = (1 + x^2x')^n$$

This is called polynomial kernel.

I want to check now what is the ϕ for K_n?
I want to compute ϕ s.t. $\phi(x)^T\phi(x') = k_n(x,x') = (1+x^Tx')^n$
We can use Newtons bynomial theorem:

$$(1+x^Tx')^n = \sum_{k=0}^{n} \binom{n}{k} (x^tx')^k$$

$$(x^Tx')^k = \left(\sum_{i=1}^{d} x_i x_i'\right)^k = \sum_{v\in\{1,...d\}^k} \left(\prod_{s=1}^{k} x_{Vs}x'_{Vs}\right)$$

$$\phi(x) = \left(\sqrt{\binom{n}{k}} \prod_{s=1}^{k} x_{Vs}\right) \qquad k=0,...,n \qquad v\in\{1,...d\}^k$$

When I am using polynomial kernel I am implicitely using the feature expansion ...
Can an algorithm work using kernel?
Perceptron works!

$S = 0$
For $t = 1, 2, ...$
1) Get (x_t, y_t)
2) $\hat{y}_t = sgn\left(\sum_{s\in S} y_s\ K(x, x_s)\right)$
3) If $\hat{y}_t \neq y_t \quad S \longleftarrow S\cup\{t\} \qquad w \leftarrow w + y_t\phi(x_t)$

So I am representing y as a sum and not as a vector. In fact, $w = \sum_{s\in S} y_s\phi(x_s)$

ù

18.2 Gaussian Kernel

$$\gamma > 0 \qquad k_\gamma(x,x') = \exp\left(-\frac{1}{2\ \gamma}\ \|x-x'\|^2\right)$$

$$e^{-\frac{1}{2\ \gamma}\ (x-x')^2}$$

I can controll the distribution changing the value of γ

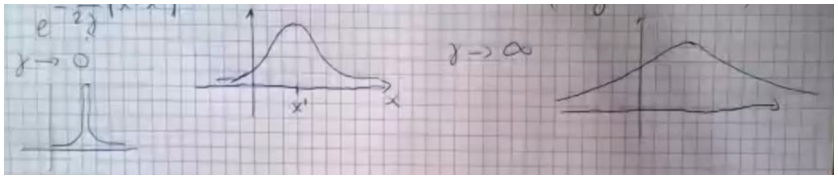


Figure 18.2:

$$\hat{y}_t = sgn\left(\sum_{s\in S} y_s \; K_\gamma(x, x_s)\right)$$

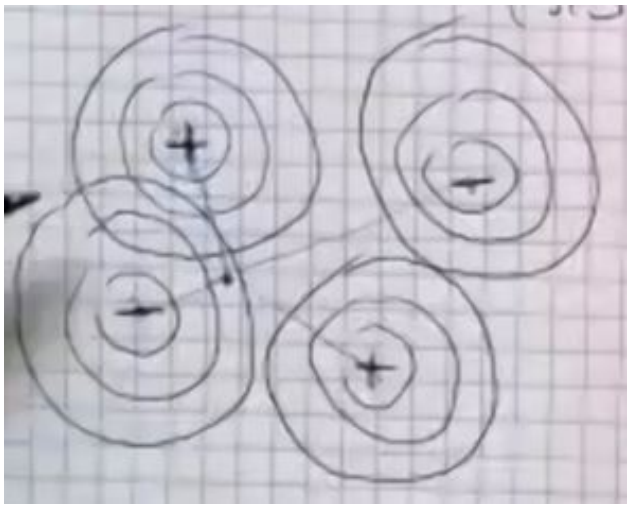


Figure 18.3:

Negative or positive gaussin component looking at the distance.
Now I want to compute: $\phi_\gamma : \mathbb{R}^n \longrightarrow V$

$$\exp\left(-\frac{1}{2\;\gamma}\|x - x'\|^2\right) = \exp\left(-\frac{1}{2\;\gamma}\left(\|x\|^2 + \|x'\|^2\right)\right) \cdot \exp\left(\frac{1}{\gamma}x^T x'\right) =$$

where $e = x + \frac{x^2}{!2}...$

$$= \exp\left(-\frac{1}{2\;\gamma}\|x\|^2\right) \cdot \exp\left(-\frac{1}{2\;\gamma}\|x'\|^2\right) \cdot \sum_{n=0}^{\infty} \frac{1}{n!} \frac{\left(x^T x'\right)^2}{\gamma^n}$$

Gaussian Kernel is a linear combination of infinitely many poly kernels.
The higher I go the small is $n!$. Gaussian kernel mapping into a space that is very large. So large that it has infinitely many dimension. Why? Because each polynomial kernel maps to infinitely dimensions.

ϕ_γ maps $\mathbb{R}^d$ into a space of infinitely many dimensions.

$$\phi_\gamma : \mathbb{R}^d \to V \qquad k_\gamma(x, x') = \phi_\gamma(x)^T \phi_\gamma(x)$$

It maps to infinetely many dimension, so it maps to a function!
$\phi_\gamma(x)$ **is a function.**
In general, when I learn a linear predictor using k_γ
I learn $\sum_s \alpha_s \; k(x_s, \cdot) = f$
$w^T \phi(x)$

$$H_\gamma \equiv \{ \sum_{i=1}^{N} \alpha_i \; k(x_i, \cdot) \;\; : \;\; x_1, ..., x_N \in \mathbb{R}^d, \;\; \alpha_1, ..., \alpha_N \in \mathbb{R}, \;\; N \in \mathbb{N} \}$$

Theorem
$\forall \gamma > 0 \; \forall f : \mathbb{R}^d \to \mathbb{R}$ continous, $\forall \varepsilon > 0$
$\exists g \in H_\gamma$ that approximates f with error ε
We define a function with H. We see the $\cdot$ and this tell us is a function. So we can evaluate every kind of x point in $\cdot$ position.

We are able to get a super parametric algorithm and transform it in a non-parametric algorithm. Parametric algorithms is defined by an arbitrary number of parameter we cannot adapt it for every case.

Gaussian Kernels enable consistency by using feature expansion with infinitely many components.

Bibliography