RAW DATA VS FEATURE EXTRACTION.

Advantages. · No domain specific knowledge required

· Attempt to capture discriminant information in the data

· Lower dimensionality and complexity

Disadvantages. · Highly redundant, large dimensional spaces

· Unknown discriminability

· Domain specific knowledge required.

$\boxed{\text{MACHINE LEARNING}}$ Set of methods that can automatically detect patterns in data and then use the uncovered patterns to predict future data or to perform other kinds of decision making under uncertainty.

It is used when: { There is a pattern
We cannot pin it down mathematically
We have data on it

TYPES:

① PREDICTIVE or SUPERVISED LEARNING: labelled data set, $D = \{(x_i, y_i)\}_{i=1}^{N}$ training set, we seek to find a mapping from x to y:
regression, classification ...
prediction of quantity YES/NO target

② DESCRIPTIVE or UNSUPERVISED LEARNING: given a data set $\{x_i\}_{i=1}^{N}$ we seek to find out more about their structure: density estimation, clustering, dimensionality reduction...

③ REINFORCEMENT LEARNING: given an external system upon which we exert control action a and receive percepts p, a reward signal r indicating good performance, find mapping from $P \to A$ that maximizes some long-term measure of r.

ML 1:

SIMULATE EXPLOTATION DATA:

Given a data set D, we divide it in 3 sets $D = D_{train} \cup D_{test} \cup D_{validation}$

we use D_{train} to learn different models.

we use $D_{validation}$ to decide which model better adapts the problem (less accuracy)

we use D_{test} to compute a performance of the selected method.

we apply the selected method to new data and hope it predicts correctly.

ML 2 · ABOUT DATA:

HANDLING MISSING DATA.

Do missing data encode any kind of information?

YES → Convert it to a meaningful number or create its own category

NO → Is missing data set small and randomly distributed?

YES → Remove sample with missing data.

NO → Check the histogram: is the distribution of data simple?

YES → Use regressor to infer a plausible value.

NO → Gaussian: use the mean

Non-gaussian: median or randomly draw a sample from the empirical distribution

INPUTING: process of replacing missing data with another value.

Ways of dealing with missing data:

1) Deletion: Remove data sample with missing values

Pair-wise deletion: keep all possible data in each analysis.

2) Single substitution: Replace with mean / median (INPUTING)

Dummy variable (new variable indicating the value is missing) and input missing data with mean.

Regression inputation: regression on available data to input values.

3) Model-based methods: Multiple inputation: by sampling independent values from a distribution.

CATEGORICAL DATA: ONE-HOT ENCODING / DUMMY VARIABLES:

Encode one feature into k-1 new features, where k is the amount of values the original feature has.

If we add k new feature we create a metric where all the distances between categories (it is always 1)

More features does not necessarily traduce in more accuracy!

NUMERICAL DATA PREPROCESSING: NORMALIZATION TECHNIQUES:

Standarization creates a metric with regular distance computation.

$$z = \frac{x - \mu_x}{\sigma_x}$$

MinMaxScaler: $$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

All normalization techniques are influenced by OUTLIERS ⇒ It is a good practice to detect outliers and potentially remove them.

observation which deviates so much from the other observations as to arous suspicious that it was generated by a different mechanism

Approaches.

1) Statistical description: infer parameters of the generating distribution and detect points with small probability of belonging to it
2) Geometric considerations: convex hulls to detect points on the external boundary of the dataset
3) Distance based: distance to neighbors: K-NN.

*) Percentile prc of the data as outlier and normalize according:

$$z = \frac{x - prc(x, \theta)}{prc(x, 100-\theta) - prc(x, \theta)},$$ θ is the percentile we consider as outlier.

FEATURE HASHING: encoding using a hashing function that given the input category returns the value of the index where the one is located.
(if the category is not in that position we conclude it was not on the list)

Hashing functions must be

1) Random: randomly distributes data among all posible indexes
2) Consistent: constant and well defined

when two different samples are assigned to the same position we have a COLLISION.

If data is not uniformly distributed, we must design a function that generates uniformly distributed indexes from data values.

In general, the scheme for hashing data with long character strings, is to break the input into a sequence of small units (bits, words...) and combine all the units b[1],..., b[m] sequentially.
When using bcn numbers we have to create the hashing function that distributes all the values equiprobably.

Hashing advantages:

- Vectors will usually be very sparse: we can store them efficiently
- Increasing complexity of hashing function may prevent collisions. (bloom filters)
- Don't need to prepare dictionaries or structures (real time online friendly)
- Distribution of the hashed data tends to be uniform.

Hashing disadvantages:

- Metric in original space disappears (Locality Sensitive Hashing)
- We have to set in advance the dimensionality of the embedding space.

* The more irrelevant information we add to the data set the worse accuracy we get. We are in fact introducing noise to the model.

ML 3: PERFORMANCE MEASURES:

MODEL SELECTION.

way of comparing models by simulating the explotation stage: splitting the dat set in TRAINING, VALIDATION and TEST sets.

In order to actually get representative results it is advisable to do this process many times in a way that every point will be used for testing purposes: CROSS-VALIDATION. techniques.

LEAVE ONE OUT (LOO): maximizes the amount of data used for training + produces good estimation of validation error

i) Take 1 sample of the data set, x_i

ii) Train the classifier with all the data except x_i: $X_{train} = \{X\} \setminus \{x_i\}$

iii) Test the classifier on x_i and store result

iv) Repeat the process for all samples

v) Array output with all the results ready for the computation of a performance metric.

K-FOLD:

i) Split the data set in k disjoint subsets with same cardinality:
$\{X\} = S_1 \cup \cdots \cup S_k$ where $S_i \subset \{X\}$, $S_i \cap S_j = \emptyset$, $i \neq j$, $|S_i| \sim |S_j|$ $\forall i,j$

ii) Select one S_i as test set

iii) Train the classifier in all except S_i. $X_{train} = \{X\} \setminus S_i$

iv) Test the trained classifier with S_i and store each result for every sample in the subset

v) Repeat with each subset

vi) Array output with all individual results ready for the comp. of p. metric.

ML 3.

OPERATING POINT: precise threshold we select for a specific application by controlling the true positive rate (recall) vs the false positive rate (1- specificity)

This is a way to control the amount of true and false positives.

Consider a BINARY CLASSIFIER, that stores probability of belonging to class A, i.e. $P(x \in A)$; then obviously $1 - P\{x \in A\}$ is probab. of belonging to class B.

The decision of belonging to A is given by $P\{x \in A\} > thr = 0'5$ ↑ usually

Imagine we lower the threshold:

$P(x \in A) > thr$, $thr < 0'5$, A: positive class

$\Rightarrow \nearrow FP$

$P(x \in B) < thr < 0'5$, $\Rightarrow \searrow FN$

Recall the confusion matrix and the partial metrics.

$$\text{Sensitivity / Recall} = \frac{\text{True Positive}}{\text{Real Positives}} = \frac{TP}{TP+FN} \nearrow$$

we have more positives predicted, then since the real are fixed TP ↑

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{Real negatives}} = \frac{TN}{TN+FP} \searrow$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{Predicted Positive}} = \frac{TP}{TP+FP} \searrow, \quad NPR = \frac{TN}{TN+FN} \nearrow$$

ML 4 SUPERVISED LEARNING

MACHINE LEARNING ALGORITHM COMPONENTS:

1) MODEL CLASS/HYPOTHESIS SPACE: family of mathematical models used. Target decision boundary from this space (linear or non-linear models)

2) PROBLEM MODEL: formalizes and encodes the desired properties of the solution → in general an optimization problem, minimization of an error function (predicted model - target model)

In classification, the ideal error function / cost function is the 0-1 loss
1 when incorrectly classify a training sample
0 otherwise (correct)

3) LEARNING ALGORITHM: optimization/search method or algorithm that given a model class fits it to the training data according to the error function. (min error or max probable model)

ML 3

RECEIVER OPERATING CHARACTERISTIC: ROC plot of the operating points by varying the threshold value (precission vs recall)

(4) MACHINE LEARNING PROBLEM: $\begin{cases} \text{Model class} \\ \text{Cost function} \\ \text{Learning algorithm} \end{cases}$

~~LEA~~

MODEL CLASS: Linear models $H(w_0, w_1) = \hat{y} = w_1 x + w_0$, w_0, w_1 param.

COST FUNCTION: mean least squares: $\mathcal{L}(y_i, \hat{y}_i) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$; $\hat{y}$ prediction

The problem of learning is modellized as:

$$\min_{w_0, w_1} \frac{1}{N}\sum_{i=0}^{N}(y_i - (w_1 x_i + w_0))^2$$

In order to solve the optimization problem let's consider the Gradient Descent algorithm,

i) $w^0 = (w_0^0, w_1^0)$

ii) $w^t = w^{t-1} + \eta \Delta w$ with $\begin{cases} \eta: \text{Learning rate} \\ \Delta w = -\nabla_w \mathcal{L} \end{cases}$

Matricial form: Extended data $\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$, $\hat{y} = \sum_{i=1}^{d} w_i x^i + w_0 = \tilde{x}^T w$

where, $x = (x_1, \dots, x_N)^T$, $N = \#$ data points

Loss matrix: $\mathcal{L}(y_i, \hat{y}_i) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N}(y - \hat{y})^T(y - \hat{y})$

Then, the opt. problem becomes: $\min_{w} \frac{1}{N}(y - \tilde{x}^T w)^T(y - \tilde{x}^T w)$

Note $\nabla_w \mathcal{L} = -\frac{2}{N}\tilde{x}(y - \tilde{x}^T w)$

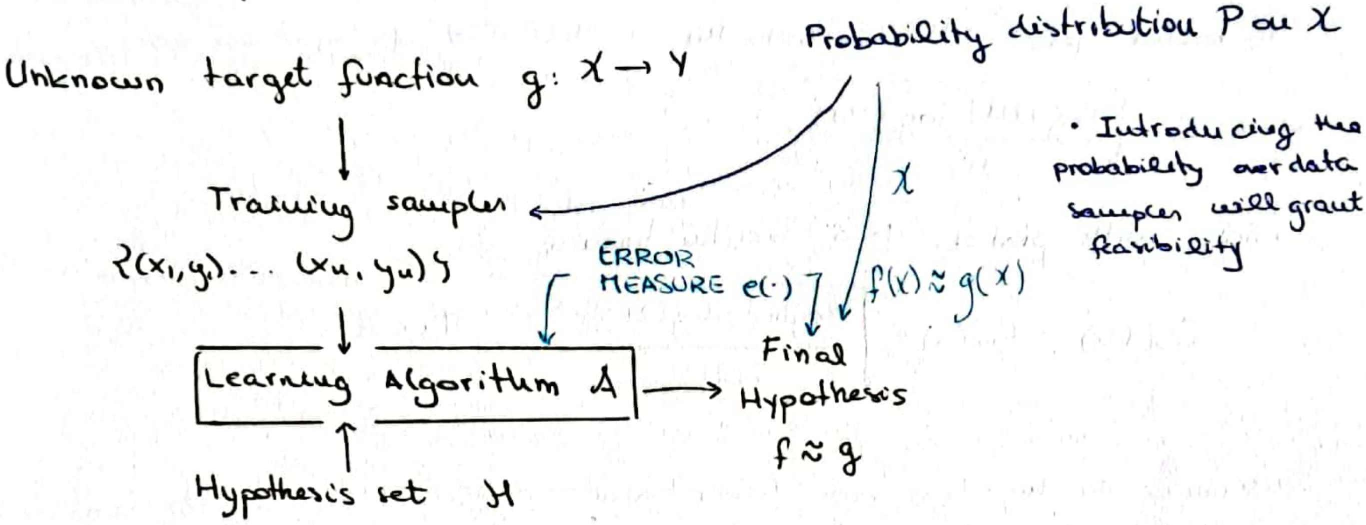
LEAST SQUARES LINEAR REGRESSION:

we have an extreme point $\Leftrightarrow$ $\nabla_w \mathcal{L} = 0 \Leftrightarrow w = (\tilde{x}\tilde{x}^T)^{-1}\tilde{x}y$.

ML: 5 FEASIBILITY OF THE LEARNING PROBLEM

A Machine learning problem consists of $\begin{cases} \text{Model class} \\ \text{Cost function} \\ \text{Learning algorithm} \end{cases}$

The structure followed is:



Given a model with an unkown function of probability, when training a learning model it would be possible that the training set represents an unprobable situation, misleading the training process. However, although any configuration is possible, not all of them are equally probable. Then, in a big sample, large N, ϑ (the supposed probability distribution) is probably close to μ (the real one) within ε:

HOEFFDING INEQUALITY: $P[|\vartheta-\mu| > \varepsilon] \leq 2 \cdot e^{-2\varepsilon^2 N/2}$

ϑ: in sample error = freq. of hypothesis getting it wrong : $E_{in}(h)$
μ: out of sample error = expected error : $E_{out}(h)$

$$P(|E_{out} - E_{in}| > \varepsilon) \leq 2e^{-N\varepsilon^2/2}$$

LEARNING: process of assuming that the in sample error is an indicator of the out of sample error.
We choose the hypothesis with the smaller out of sample error.

Hoeffding doesn't apply to multiple bins!

Using the union bound $P(A_1 \cup \dots \cup A_n) = P(A_1) + \dots + P(A_n)$ we get:

- $P[\max_{h \in H} |E_{in}(h) - E_{out}(h)| > \varepsilon] \leq \sum_{h \in H} P[|E_{in}(h) - E_{out}(h)| > \varepsilon] \leq \underbrace{2|H|e^{-2\varepsilon^2 N}}_{\delta}$ ⊛

The bound gets WORSE with the COMPLEXITY of hypothesis space.

$$⊛ \Rightarrow \varepsilon = \sqrt{\frac{\log|H| + \log(2/\delta)}{2N}}$$

Then, with probab. $1-\delta$, over the training set:

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{\log|H| + \log(2/\delta)}{2N}}, \quad \forall h \in H$$

We can use the hypothesis with lower bound ⇒ MODEL SELECTION.

We can get N: number of training set if we seek for ex. δ, and generalized error G_E:

$$N = \frac{\log|H| + \log(2/\delta)}{2(G_E)^2}$$

average of point wise errors ⊠

- ERROR MEASURES on the final Hypothesis: "$h \approx g$": $E(h, g)$

Different point wise definitions: Squared error: $e(h(x), g(x)) = (h(x) - g(x))^2$

Binary error: $e(h(x), g(x)) = [\![h(x) \neq g(x)]\!]$.

⊠ In sample error: $E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N} e(h(x_n), g(x_n))$

Out of sample error: $E_{out}(h) = \mathbb{E}_x[e(h(x), g(x))]$

The error measure should be specified by the user/client.

ERROR CONCEPTS: CONFUSION MATRIX

	Gold Standard Positive	Gold Standard Negative	
Positive	TP	FP	→ Precision
Negative	FN	TN	→ Negative Predictive Value.
	↓ Sensitivity (Recall)	↓ Specificity	

....

Metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Partial metrics:

Column-wise: $\text{Sensitivity} = \frac{TP}{\text{Real Positives}} = \frac{TP}{TP + FN}$

$\text{Specificity} = \frac{TN}{\text{Real Negatives}} = \frac{TN}{TN + FP}$

Row-wise: $\text{Precision} = \frac{TP}{\text{Predicted Positives}} = \frac{TP}{TP + FP}$

$NPV = \frac{TN}{\text{Predicted Negatives}} = \frac{TN}{TN + FN}.$

Another metrics:

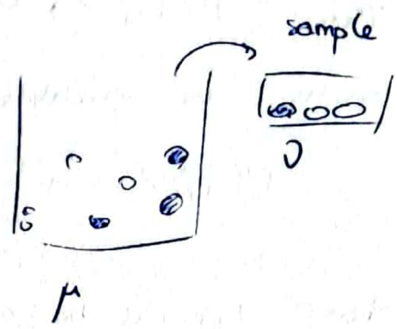
$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 2 \frac{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} = \frac{2TP}{TP + FN + TP + FP}$$

$TPR = \text{recall} = \frac{TP}{TP+FN}$, $FPR = 1 - \text{specificity} = \frac{FP}{TN+FP}$

Then, for a perfect operational/classification we have $FN = FP = 0 \Rightarrow \begin{cases} TPR = 1 \\ FPR = 0 \end{cases}$

M.L 5:

Example considered:



Learning: unknown function $g: \mathcal{X} \to \mathcal{Y}$

Each marble is a point $x \in X$

- Hypothesis right: $h(x) = g(x)$
- " wrong: $h(x) \neq g(x)$

$\mathcal{D}$: in sample error = frequency of the hypothesis getting it wrong: $E_{in}(h)$
μ: out of sample error = expected error: $E_{out}(h)$

Then $P(|E_{out} - E_{in}| > \varepsilon) \le 2e^{-\frac{N\varepsilon^2}{2}}$

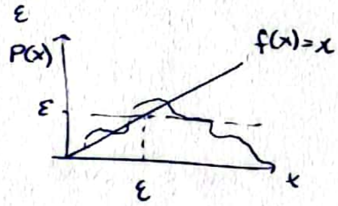
we seek to bound the training error, $P\left(\left|\frac{1}{N}\sum_{i=1}^{n} x_i\right| > \varepsilon\right) \le 2 \mathcal{E} e^{-\frac{N\varepsilon^2}{2}}$

Markov inequality: x positive random variable, then $P(x \ge \varepsilon) \le \frac{\mathbb{E}[x]}{\varepsilon}$

Proof:

$$\mathbb{E}(x) = \int_0^\infty x P(x)\,dx = \int_0^\varepsilon x P(x) dx + \int_\varepsilon^\infty x P(x)\,dx \ge \int_\varepsilon^\infty x P(x)\,dx \ge \int_\varepsilon \varepsilon P(x)\,dx$$

$$\Rightarrow \mathbb{E}(x) \ge \varepsilon \int_\varepsilon^\infty P(x)dx = \varepsilon P(x \ge \varepsilon) \Rightarrow P(x \ge \varepsilon) \le \frac{\mathbb{E}(x)}{\varepsilon}$$



$\square$

Chevisev inequality: $P(|x-\mu| \ge \varepsilon) \le \frac{Var(x)}{\varepsilon^2}$, $\mu = \mathbb{E}(x)$

Proof:

$$|x-\mu| \ge \varepsilon \Leftrightarrow (x-\mu)^2 \ge \varepsilon^2 \xRightarrow{\text{Markov}} P((x-\mu)^2 \ge \varepsilon^2) \le \frac{\mathbb{E}[(x-\mu)^2]}{\varepsilon^2} = \frac{Var(x)}{\varepsilon^2}$$

Then, $P(|x-\mu| \ge \varepsilon) \le \frac{Var(x)}{\varepsilon^2}$

$\square$

Chernof bound: $P(|x-\mu| \ge \varepsilon) \le \min_\lambda \mathbb{E}(e^{\lambda\|x-\mu\|}) e^{-\lambda\varepsilon}$, $\lambda > 0$.

Proof:

since $\lambda > 0$, $|x-\mu| \ge \varepsilon \Leftrightarrow \lambda|x-\mu| \ge \lambda\varepsilon \Rightarrow e^{\lambda|x-\mu|} \ge e^{\lambda\varepsilon}$

Then if it holds $\forall \lambda$, it holds for the minimum. Applying Markov:

$$P(e^{\lambda|x-\mu|} \ge e^{\lambda\varepsilon}) \le \frac{\mathbb{E}[e^{\lambda\|x-\mu\|}]}{e^{\lambda\varepsilon}} = \mathbb{E}[e^{\lambda\|x-\mu\|}]e^{-\lambda\varepsilon} \Rightarrow$$

$$P(|x-\mu| \ge \varepsilon) \le \min_\lambda \mathbb{E}[e^{\lambda\|x-\mu\|}]e^{-\lambda\varepsilon}, \text{ since } \lambda > 0.$$

$\square$

Then, $P\left(\left|\frac{1}{N}\sum_{i=1}^{N} x_i - \mu\right| \geq \varepsilon\right) \leq 2e^{-N\varepsilon^2/2}$

with some variable change, this is equivalent to

$$P\left(\left|\frac{1}{N}\sum x_i\right| \geq \varepsilon\right) \leq 2e^{-N\varepsilon^2/2}$$

Proof: Let x_i be a Rademacher random variable, $x_i = \begin{cases} 1 & P(x_i=1) = 1/2 \\ -1 & P(x_i=-1) = 1/2 \end{cases}$

Chernof bound: $P(|x| \geq \varepsilon) \leq \min_\lambda \mathbb{E}[e^{\lambda|x|}]e^{-\lambda\varepsilon}$

Similarly, $P\left(\sum_{i=1}^{N} x_i \geq N\varepsilon\right) \leq \min_\lambda \underbrace{\mathbb{E}[e^{\lambda\sum x_i}]}_{\mathbb{E}[\prod e^{\lambda x_i}] = \prod_i \mathbb{E}[e^{\lambda x_i}]} e^{-\lambda N\varepsilon}$ x_i random variable independent

But for any i, we have $\mathbb{E}[e^{\lambda x_i}] = \frac{1}{2}e^{\lambda} + \frac{1}{2}e^{-\lambda} = \sum_i \frac{\lambda^{2i}}{(2i)!}$ $\leq \left\{ (2i)! = \underbrace{1\cdots i}_{i!} \cdot i{+}1 \cdots 2i \geq i!\,2^i \right.$

$$\leq \sum_i \frac{\lambda^{2i}}{i!\,2^i} = \sum_i \frac{(\lambda^2/2)^i}{i!} = e^{\lambda^2/2} \qquad \text{since} \begin{cases} e^{\lambda} = \sum_i \frac{\lambda^i}{i!} \\ e^{-\lambda} = \sum_i \frac{(-\lambda)^i}{i!} \end{cases}$$

Then, $P\left(\frac{1}{N}\sum_{i=1}^{N} x_i \geq \varepsilon\right) \leq \prod_{i=1}^{N} \underbrace{\mathbb{E}[e^{\lambda x_i}]}_{\leq e^{\lambda^2/2}} e^{-\lambda N\varepsilon} \leq \mathbb{E}[e^{\lambda x_i}]^N e^{-\lambda N\varepsilon}$

which does not depend on x_i

$$= e^{N\lambda^2/2} e^{-\lambda N\varepsilon} = e^{N\lambda^2/2 - \lambda N\varepsilon}$$

Note that $\frac{\partial e^{(N\lambda^2/2 - \lambda N\varepsilon)}}{\partial\lambda} = e^{(\ )}[2\lambda N/2 - \varepsilon N] = 0 \Leftrightarrow \lambda = \varepsilon$

Therefore, the minimum for $\lambda > 0$ is given by $\lambda = \varepsilon$, then the Chernof bound is given by:

$$P\left(\frac{1}{N}\sum_{i=1}^{N} x_i \geq \varepsilon\right) \leq e^{-N\varepsilon^2/2}$$

I el 2 d'où sort ?

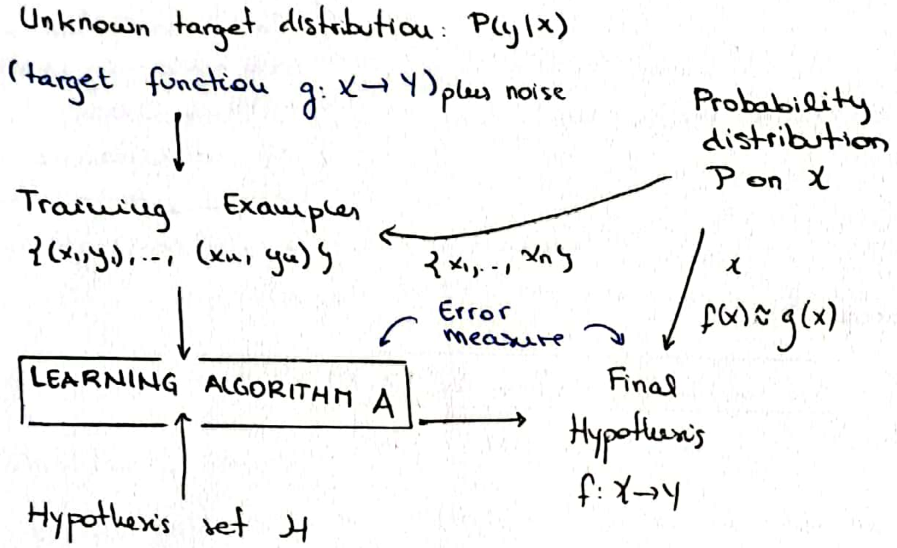
□

Sometimes the "target function" is actually a TARGET DISTRIBUTION: $P(Y|X)$

- x still generated by $P(x)$ [we can use Hoeffding]
- y not deterministic with $x \Rightarrow y$ generated by $P(Y|x)$.

Thus, (x,y) generated by the joint distribution $P(x) \cdot P(y|x) = P(x,y)$

We could consider $g(x) = \mathbb{E}(y|x)$ with noise $y - g(x)$:



- LEARNING is FEASIBLE: Hoeffdings $\Longrightarrow E_{out}(f) \approx E_{in}(f)$

  We also need $E_{out}(f) \approx 0$, i.e. $f \approx g$

We say that a hypothesis set H SHATTERS a set of k points if at least one of the individual hypothesis in the set is able to generate all possible labels over some configuration of k number points.

VAPNIK-CHERVONENKIS DIMENSION: maximum number of points that a classifier can shatter.

Linear model (2D): can shatter until 3 points but not 4:  

$$dim_{vc}(L.M) = 3$$

Circle classifier (2D): can shatter until 4 points: $dim_{vc}(C.M) = 4$

3D Linear classifier: $d_{vc}(\text{linear model}_D) = D+1 \Rightarrow$ In ∞ dimensions we can solve any model with a linear classifier of ∞ dimensions.

$$d_{vc}(L.M._3) = 4$$

FINAL RESULTS:

- $|d_{vc}| \approx 10 \cdot$ free parameters.
- $E_{out}(h) \leq E_{in}(h) + \sqrt{\dfrac{d_{vc}(\log(2n/d_{vc}+1)) + \log(2/\delta)}{2n}}$

Escaneado con CamScanner

LEARNING consists of finding a model such that $E_{out} \to 0$ by satisfying:

1) $E_{in} \to 0$ (by selecting a learning method that minimizes E_{in})

2) $E_{out} \approx E_{in}$. (by considering probabilistics settings + Hoeffding's)

As a consequence: $E_{out} \leq E_{in} + O\left(\sqrt{\frac{C}{N}}\right)$ C: notion of complexity
N: amount of data samples.

* Independently of the data generation process generating we have to match the data complexity and not the model complexity (?)

LEARNING CURVE: curve of training and test errors as the number of training data increases for a given complexity.

(Error rate as: 1 - accuracy over < training / test)

- Note that as the # of training data ↑ ⇒ both errors tend to a value (BIAS)
- Note that when ↘ # of training data ⇒ training error is small but test error is large.
- With small complexity ⇒ training and test error converge with smaller N
⇒ the error of converge is larger !

Difference between BIAS and test error is the VARIANCE

OVERFITTING: increment of test error that occurs when a certain complexity level is attained. However, the training error might be reduced as the complexity increases.

How to cure overfitting: $E_{out} \leq E_{in} + O\left(\sqrt{\frac{C}{N}}\right)$

1) Cross-validation: simulating the out-of-sample error and checking against unseen data several times.

2) Regularization: change learning objective + minimize complexity (adding to the obj function a term that penalizes complexity)

3) Ensemble techniques.