Introduction to *In-Silico* learning
Modelling Learning

Abstract

This tutorial provides an introduction to learning from a theoretical perspective: this preliminary introduction is therefore required to understand the incoming algorithms and models. The next tutorials will provide more concrete examples providing specific instances of these problems.

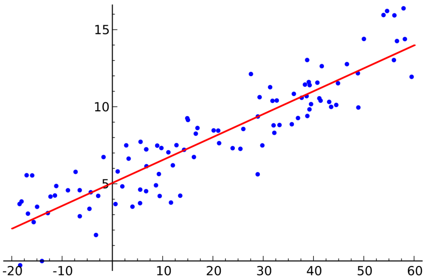
1 Intuition

In this tutorial we prefer using the term *In-Silico* learning to machine learning, because such term does not necessarily include other learning-based algorithms, such as data mining ones. In particular, both activities can be assimilated to a reasoning process, after which the machine can solve specific problems that would require "intelligence" for a human to solve. Therefore, these set of tutorials will use a different umbrella term for englobing two distinct approaches to automate learning [2].

Generally speaking, **machine learning** is the study of systems that improve their behaviour over time with experience: such experience consists in a set of examples E in which we look for regularities or patterns fitting a specific model. In particular, these models focus on learning one specific function approximating the target function that the data implicitly exhibits. Figure 1a sketches a naïve learning approach: using a *linear least-squares (error) estimation*, we can infer a single hypothesis h which, in this case, is a line minimizing the distance from the prediction (line's y value) to the real data value (*loss*). Learning approaches then differ from the different model or function type that is going to be returned as an hypothesis.

Due to the limitations of the first attempts, more advanced symbolic reasoning approaches involving the use of both logic and probabilistic models emerged (*relational learning*[1]): in these approaches, as well as in **data mining** algorithms, we want to provide several different plausible hypotheses h that meet some quality criterion $\mathcal{Q}$ over a sample $D \subseteq E$ of the original dataset. Similarly to the learning algorithms, $\mathcal{Q}$ and the choice of how to sample D from E might differ from algorithm to algorithm but, as we will see in the next section, they can be formalized with one single "mathematical"

[1] We're not going to cover these kind of algorithms now, as they require advance knowledge of probability theory and logic.



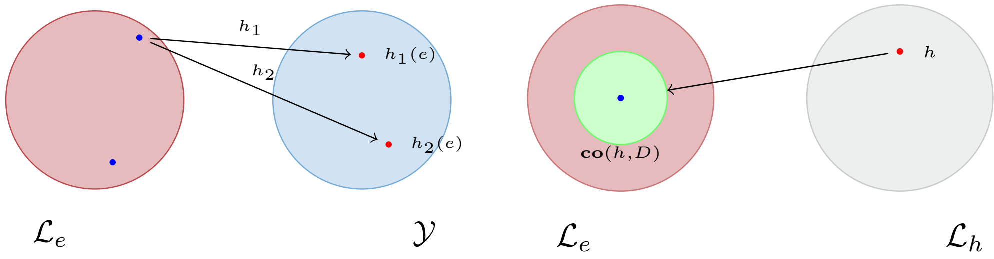
(a) Trying to fit the data (blue dots) to a linear model. The red line represents the outcome of the learning process.

ID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Frequent Item-sets	**Association Rules**
{Diapers, Beer}	{Diapers} → {Beer}
{Bread, Milk}	{Bread } → {Milk}

(b) Using a frequency predicate $\mathcal{Q}$, we can determine the most frequent items over all the possible subset of the data items, thus providing multiple possible hypotheses.

Figure 1: Two examples of different approaches to *In-Silico* learning.



(a) Hypotheses viewed as functions.

(b) The *covers* relation for $\mathcal{Y} = 2$.

Figure 2: Examples of mathematical models that are used as intermediate steps for achieving computational representation.

formulation. An example of two data mining algorithms' outcome is provided in Figure 1b: using two distinct quality criterion over two distinct representations of the items, we can first detect the most frequent subsets, and then extract which are the most likely rules.

1.1 Formalization (*)

We are now going to formalize such distinct approaches in order to outline differences and similarities of the two distinct approaches.

All the records in our dataset E come with a data **schema** $R(X_1, \dots, X_n, Y)$. A schema is a table named R with a set of *attributes* $X_1, \dots, X_n, Y$, where the X_i variables name the different "fields" of the training data, while Y names the target classification name. Each **record** (or tuple, or data point) in E is defined as a mapping (or function) e of each attribute $Z \in \{X_1, \dots, X_n, Y\}$ into a value in $dom(Z)$ [1]: the **training data** is represented by the set $\mathcal{L}_e \supseteq dom(X_1) \times \dots \times dom(X_n)$, while the **target classification classes** are defined in $\mathcal{Y} \supseteq dom(Y)$.

Usually, each data point in E is represented as a pair. $(\mathbf{x}, y) \in \mathcal{L}_e \times \mathcal{Y}$, so to clearly separate the training data from the classes value: from this definition, it follows that each pair-represented data point represents the map $(\mathbf{x}, f(\mathbf{x}))$ of a target function $f : \mathcal{L}_e \to \mathcal{Y}$ that we want to approximate with an hypothesis h inferred from the data [2].

Please note that we are only interested to solve problems when $|\mathcal{Y}| \geq 2$, because for $|\mathcal{Y}| < 2$ the solution to the problem is always trivial. When $|\mathcal{Y}| = 2$, then we're going to solve a **binary classification problem**, while when $|\mathcal{Y}| > 2$ we must deal with a **multiclass classification problem**. For binary classification problems, we are going to use $\mathcal{Y} = \{0, 1\}$: all the examples such that $f(e) = 1$ are *positive*, while the others are called *negative*. This terminology comes from relational learning, where the former examples are the one that are logically true, while the latter are the ones to be considered as false.

Example 1. *Table 1 provides a dataset with the schema:*

$$\textsc{StarCraftReplay}(\texttt{Age}, \texttt{HoursPerWeek}, \dots, \texttt{MaxTimeStamp}, \texttt{LeagueIndex})$$

where we want to predict the XP level of the player (`LeagueIndex`*) from all the remaining variables (*`Age`*,*`HoursPerWeek`*,...,*`MaxTimeStamp`*). We want to predict eight classes, and in fact* $dom(\texttt{LeagueIndex}) = \mathcal{Y} = \{1, 2, 3, 4, 5, 6, 7, 8\}$*. Last, the first row* e *can be represented either as a record with the fields mapping* $e.\texttt{Age} = 27 \dots e.\texttt{MaxTimeStamp} = 127448$, $e.\texttt{LeagueIndex} = 5$ *or as a pair* $((27, 10, 3000, \dots, 127448), 5)$*. This means that the data induces a function* $f(27, 10, 3000, \dots, 127448) = 8$ *that we need to learn using any approach of choice. We're going to use the former notation when accessing to a specific field is relevant (e.g., decision trees) and to the latter case in all the remaining situations.*

Machine Learning As per previous discussion, the aim of machine learning (or data mining) algorithm is to learn one (or multiple possible) **hypotheses**[2] $h \colon \mathcal{L}_e \to \mathcal{Y}$ assigning the training data to one single target classification class (Figure 2a). We can define the set of all the possible hypotheses

[2]Also called "decision function".

Age	HoursPerWeek	TotalHours	APM	SelectByHotkeys	AssignToHotkeys	UniqueHotkeys	MinimapAttacks	MinimapRightClicks	NumberOfPACs	GapBetweenPACs	ActionLatency	ActionsInPAC	TotalMapExplored	WorkersMade	UniqueUnitsMade	ComplexUnitsMade	ComplexAbilityUsed	MaxTimeStamp	LeagueIndex
27	10	3000	143718	0.003515159	0.000219697	5.49e-05	0.000109849	0.000392317	0.004849036	32.6677	40.8673	4.7508	0.000219697	0.0013966	4.71e-05	0	0	127448	5
23	10	5000	129.2322	0.003303812	0.000259462	6.92e-05	0.000294057	0.000432436	0.004307064	32.9194	42.3454	4.8434	0.000380544	0.0011935	8.65e-05	0	0.00020757	57812	5
30	10	200	69.9612	0.001101091	0.00033557	4.19e-05	0.000293624	0.000461409	0.002925755	44.6475	75.3548	4043	0.000230705	0.00074455	6.29e-05	0	0.00018876	95360	4
19	20	400	107.6016	0.001033542	0.000213101	1.07e-05	5.33e-05	0.000543409	0.003782551	29.2203	53.7352	4.9155	0.000202446	0.0004262	7.46e-05	0	0.00038358	93852	3
32	10	500	122.8908	0.001136014	0.000327326	3.85e-05	0	0.001328558	0.002368299	22.6885	62.0813	9374	0.000288817	0.0011745	7.7e-05	0	1.93e-05	51936	3
27	6	70	44457	0.00097839	0.000255232	2.13e-05	0	0	0.002424706	76.4405	98.7719	3.0965	0.000170155	0.00037221	6.38e-05	0	0	94032	2
21	8	240	46.9962	0.000820114	0.000168517	6.74e-05	0	4.49e-05	0.001988496	94.0227	90.5311	4.1017	0.000168517	0.00057296	5.62e-05	0	0	89012	1
17	42	10000	212.6022	0.009039739	0.00067624	5.97e-05	0.001163531	0.001253033	0.004952464	24.6117	41.7671	6.6104	0.000447512	0.0022773	8.95e-05	0.000129281	0.00024862	100556	7
20	14	2708	117.4884	0.002944275	0.000526771	1.88e-05	1.88e-05	0.000413892	0.005399406	52014	46.4321	3.3746	0.000272792	0.0010347	6.58e-05	0.000272792	0.00047033	106308	4
18	24	800	155.9856	0.005053908	0.000524109	9.98e-05	2.5e-05	0.000399321	0.003568933	24.4632	52.1538	6.5664	0.000336927	0.0013103	7.49e-05	0	0	80136	4
16	16	6000	153801	0.001676615	0.000318557	6.71e-05	0	0.000821541	0.003772383	23.4107	48.0711	7.0044	0.000402387	0.0015928	0.000117363	0	1.68e-05	59644	3
26	4	190	79.2948	0.000378539	0.000255102	2.47e-05	1.65e-05	0.000164582	0.00355497	39.6381	65.5	4.2269	0.000156353	0.00075708	5.76e-05	0.000106978	0.00026333	121520	4
18	12	350	67.4754	0.000422522	0.000169009	1.21e-05	2.41e-05	0.000144865	0.002885219	42437	68.0502	4.3222	0.000193153	0.00074847	8.45e-05	0	0.00043459	82836	3
38	6	1000	119.4366	0.004952043	5.21e-05	3.48e-05	8.69e-05	3.48e-05	0.002727968	54.8718	79.2102	6.2293	0.000364887	0.0014943	8.69e-05	0	0	57552	3
16	30	5000	160.4754	0.00425386	0.000431735	2.54e-05	0.000774584	0.000406339	0.004571312	36.2897	46.8889	5.4361	0.000355547	0.0019809	8.89e-05	0	0	78752	5
17	16	1500	81.7722	0.002333484	0.000430449	9.06e-05	0	0.000249207	0.002899864	45.1654	64.75	4.5312	0.000339828	0.001246	0.000135931	0	0.00022655	44140	5
28	8	2000	50.8374	0.000664109	0.00022137	1.84e-05	1.84e-05	0.00014758	0.002656434	45.7902	76.8889	3.5	0.000239817	0.00066411	0.000110685	0	0.00029516	54208	4
20	10	120	160.6464	0.003430344	0.00063363	5.1e-05	0	9.47e-05	0.005535163	28.3636	37.7947	4.7671	0.00021121	0.0027676	7.28e-05	8.74e-05	0.00010196	137304	5
16	14	350	107.9118	0.006701306	0.000706102	6.66e-05	1.33e-05	0.000466294	0.00323741	67.0744	71.3251	4.3786	0.000359712	0.0013323	0.000119904	0.000239808	0	75060	5
26	28	1100	114.7806	0.002629613	0.000287389	7.18e-05	0.000244281	0.000344867	0.003649845	40.4427	59937	4.9961	0.000229911	0.0015806	7.18e-05	0	0	69592	4
21	10	800	115.1274	0.002651148	0.000660037	8.8e-05	9.9e-05	0.000693039	0.004147232	30766	49.4854	4679	0.000330018	0.00082505	8.8e-05	0	0.00051703	90904	5
21	6	500	133.7016	0.004499795	0.000420105	2.8e-05	1.87e-05	7.47e-05	0.003874304	21.4686	50.5253	5.4892	0.000270735	0.0010083	9.34e-05	0	0.00056014	107116	6
18	20	800	99.5088	0.00073402	0.000108298	1.2e-05	2.41e-05	0.00057759	0.002695418	22.7265	68.7321	6.7054	0.000276762	0.0019614	8.42e-05	0	0	83104	5
26	10	500	83.9172	0.002853513	0.000492966	2.84e-05	0	0	0.004607334	51.2082	52.0494	2.8663	0.000293883	0.00065413	8.53e-05	0	0	105484	5
17	14	500	216.6936	0.012494649	0.000481592	0.000107021	5.35e-05	0.000107021	0.004334332	27.9255	45.1605	6.3395	0.000481592	0.001204	0.000133776	0	0	37376	4
23	20	800	129.8598	0.002315078	0.000333769	3.55e-05	7.81e-05	0.001008408	0.003472617	36.3934	57.5215	6.7587	0.000255653	0.00075276	7.81e-05	0.000284059	0.0012286	140816	4
18	70	2520	267.5586	0.027814934	0.00070816	9.84e-05	0	0.000127862	0.005616099	34.6035	40.6025	4.1629	0.00035408	0.00085569	0.000118027	8.85e-05	0.00019671	101672	6
25	6	800	74.1174	0.000875	0.000205357	5.36e-05	1.79e-05	4.46e-05	0.003517857	46.4122	71.8173	3.7944	0.000303571	0.00092857	4.46e-05	0	0	112000	5
25	20	700	101.6796	0.001406194	0.000427198	8.9e-05	0	0.0001246	0.003453186	26.7979	54.5979	5.3144	0.000391598	0.0010324	7.12e-05	0	0.0005696	56180	5
18	10	160	150.5004	0.005666689	0.000631529	0.00010241	0.000375504	0.00040964	0.004625521	30.2222	45.5941	4.9077	0.000238957	0.001007	0.00010241	0	0	58588	6
18	8	250	41.9094	0.000275751	0.000190904	0	0	0.000169693	0.002142372	54.4677	91.3069	3.6881	0.000180299	0.00066817	6.36e-05	0	2.12e-05	94288	3
16	16	1000	128.0784	0.002753792	9.5e-05	2.53e-05	0.000240561	0.000164594	0.003969259	35393	56.4019	5.2711	0.000170925	0.00043048	5.06e-05	0	0	157964	5
16	28	500	161.3466	0.006109907	0.000577482	6.42e-05	0.000327953	0.0007272	0.00413506	34.1209	47.1034	5.8466	0.000342212	0.0013475	6.42e-05	0.000377859	0.00017111	140264	6
22	4	400	90.7686	0.001610454	0.000392794	5.24e-05	0	0.000183304	0.003469676	44.7424	59.5321	4.5698	0.000196397	0.00070703	7.86e-05	0	0.00032733	76376	4
20	14	730	162.0876	0.007596727	0.000571635	0.000105301	0.000255731	0.000752151	0.003986401	23.5303	51034	5.7698	0.000421205	0.0026024	9.03e-05	0	1.5e-05	66476	6
18	6	230	69.5076	0.00017507	0.000204248	0.000116713	5.84e-05	0.000204248	0.001458917	45.0612	112.96	9.14	0.000320962	0.00058357	0.000116713	0	0	34272	1
16	24	300	81.0702	0.001746793	0.000104286	0	2.61e-05	0.000143393	0.00262019	47.48	75.0846	5.0796	0.000182501	0.00080822	7.82e-05	2.61e-05	3.91e-05	76712	3
22	8	300	157332	0.007742665	0.000223472	1.31e-05	0.000131454	0.000985908	0.003799032	33.0417	57.7716	5.8581	0.000381218	0.0018929	7.89e-05	0	0	76072	5
23	4	100	67.6194	0.001565762	0.000398308	5.49e-05	4.12e-05	0.00112625	0.001661905	49.4333	89.9174	7.1322	0.000233491	0.00083782	8.24e-05	9.61e-05	0.0002747	72808	4
25	10	200	75.9642	0.000609665	0.000136864	4.98e-05	0.000124421	0.000323496	0.00209028	45.1018	78.3095	6.5417	0.000186632	0.00093316	7.47e-05	0	0	80372	2
25	12	200	164.1114	0.003372032	0.000403892	6.57e-05	0.000112714	0.001568605	0.004987601	40.4604	43.7815	5.6365	0.000300571	0.0012774	7.51e-05	0.000216035	0.0010896	106464	5
20	14	270	101.7864	0.001453175	0.000538213	3.59e-05	5.38e-05	0.000304987	0.004180122	22569	54.6094	4.2961	0.000269107	0.00096878	7.18e-05	0	0.00021529	55740	4
25	8	100	79128	0.00050077	8.99e-05	2.57e-05	1.93e-05	0.001720596	0.002754237	50.5327	67.1422	5345	0.000179764	0.00043657	3.85e-05	0	6.42e-06	155760	2
22	20	1200	120.7014	0.005495624	0.000239888	3.63e-05	0	2.18e-05	0.003612864	44.8387	60.9175	4.8089	0.00022535	0.00051612	3.63e-05	0	0	137564	4
18	20	800	108.5424	0.002100536	0.000398378	7.24e-05	0	0.000434594	0.003078372	28.4762	61.9294	5.8353	0.000398378	0.0025351	0.000144865	0	0	27612	6
23	2	30	66.7818	0.000685853	0.000168228	2.59e-05	5.18e-05	0.001112894	0.002406957	56.7784	97.9785	4.9355	0.000245872	0.00042704	7.76e-05	0	0	77276	3
18	28	2000	129.1464	0.00263107	0.000599729	3.87e-05	7.74e-05	0.000338557	0.005146063	25.0245	43.0827	4.1917	0.000338557	0.00083188	8.71e-05	0.000174115	0.00048365	103380	6
18	8	600	62.2194	0.001099196	0.000321716	0	4.02e-05	0.000201072	0.002386059	44.3164	90.8989	4.5393	0.000187668	0.00071046	9.38e-05	0	0.00010724	74600	3
20	6	150	72141	0.001237902	0.000393878	4.69e-05	1.88e-05	0.000271963	0.003376097	45.9164	68.8778	3.7528	0.000225073	0.00079713	7.5e-05	0.000121915	0.00035637	106632	3
23	20	100	83.9658	0.002672697	0.000376919	0.000102796	0	0.000171327	0.002912555	44.0476	78.0706	4.6118	0.000342654	0.00071957	0.000102796	0	0	29184	4
17	56	540	199.6854	0.005146297	0.000453133	1.62e-05	3.24e-05	0.0002913	0.004790264	25.3831	33.9054	6.7095	0.000469316	0.0020876	0.000113283	0	1.62e-05	61792	5
29	12	800	66.6738	0.000320895	4.58e-05	2.29e-05	0	0.000194829	0.002292106	38.5126	72.16	5.46	0.000229211	0.0010314	0.000103145	0.000252132	0.00014899	87256	4
27	24	250	73.9722	0.002647278	0.000454884	5.22e-05	7.46e-06	0.000186428	0.002684564	58.8858	88.6889	4.4111	0.000178971	0.00058166	5.97e-05	0.0001044	0.00023863	134100	2
21	8	730	130941	0.004758302	0.00075655	0.000199092	7.96e-05	0.000139364	0.003384566	36.8047	57.2471	5.7765	0.000219001	0.00059728	3.98e-05	0	0	50228	6
27	8	280	55047	0.000642809	0.000126454	0	7.38e-05	9.48e-05	0.002529084	62.8452	83.8667	3.9333	0.000231833	0.00052689	6.32e-05	0	0.00017914	94896	2
16	10	1600	181104	0.00563541	0.000581793	0.000114077	0	0.000456308	0.005007985	20.8037	39.2255	5.6993	0.000342231	0.0017226	9.13e-05	0.000365047	0.001175	87660	6
22	36	1000	98.6694	0.004474848	0.000462915	2.57e-05	0.000257175	0.000360045	0.002391729	52.6957	62.5806	5828	0.000437198	0.0005915	7.72e-05	0	0	38884	4
18	6	50	107.4186	0.003028577	0.000310623	5.18e-05	0.000220025	0.000336509	0.003559226	37.3723	56.0436	5.0873	0.000310623	0.00095776	9.06e-05	0	0	77264	4
0	0	0	189.7404	0.004582151	0.000654593	7.27e-05	7.27e-05	0.000618227	0.006291367	23513	32.5665	4.4451	0.000454579	0.0022183	0.000109099	0	0	54996	8
0	0	0	287.8128	0.029039722	0.001041265	0.000173544	0.000231392	0.000655611	0.005399152	31.6416	36.1143	4.5893	0.000655611	0.0011377	0.000115696	5.78e-05	0	51860	8
0	0	0	294.0996	0.029639771	0.001075709	5.33e-05	0.000302265	0.002373671	0.006294229	16.6393	36.8192	4185	0.000231144	0.00098681	5.33e-05	0	0	112484	8
0	0	0	274.2552	0.018120523	0.001264223	0.000105352	5.27e-05	0.000974505	0.007111252	10.6419	24.3556	4387	0.000368732	0.0011062	7.9e-05	0	0	75936	8
0	0	0	274.3404	0.023131288	0.000739192	7.21e-05	0.000622003	0.003551725	0.005354632	19.1568	36.3098	5.2811	0.000252407	0.00073919	5.41e-05	0	0	110932	8
0	0	0	245.8188	0.010471414	0.000841238	8.01e-05	0.000656967	0.001313934	0.005031406	14.5518	36.7134	7.1943	0.000264389	0.0014742	8.81e-05	4.01e-05	4.81e-05	124816	8
0	0	0	211.0722	0.01304902	0.000940282	0.000104476	0.000365665	0.000908939	0.003719336	19.6169	38.9326	7132	0.000240294	0.00089849	9.4e-05	0	0	95716	8
0	0	0	189.5778	0.007558997	0.000487252	0.00013169	0.000605773	0.000566266	0.005820691	22.0317	36733	4905	0.000368732	0.00053993	6.58e-05	0	0	75936	8
0	0	0	210.5088	0.007973528	0.000867121	6.98e-05	0.00054818	0.000637882	0.006518359	15.7856	30.7156	4.8058	0.000338875	0.00081729	5.98e-05	0	0	100332	8
0	0	0	248.0118	0.014721527	0.00175219	0.000109512	0.000375469	0.000109512	0.004114518	17.4656	34.2357	7.8973	0.000312891	0.0011108	0.000125156	0	0	63920	8
0	0	0	299229	0.026428161	0.00095137	0.000110624	0.000154874	0.000929245	0.005442719	17.0835	33.7398	5.2703	0.000176999	0.00069693	6.64e-05	3.32e-05	1.11e-05	90396	8

Table 1: A subset of the StarCraft II dataset available online at `http://summit.sfu.ca/item/13328`. This table subset provides the following fields: *(i)* the age of each player ($X_1, \dots$), *(ii)* the reported hours spent playing per week, *(iii)* the reported total hours spent playing, *(iv)* the number of continuous actions per minute, *(v)* the number of item selections via hotkeys per time frame, *(vi)* the number of assigned items via hotkeys per time frame, *(vii)* the number of unique and continuous hotkeys used per time frame, *(viii)* the number of continuous attack on a minimap per time frame, *(ix)* the number of continuous right clicks on a minimap per time frame, *(x)* the number of PACs per time frame, *(xi)* the mean duration between PACs, *(xii)* the mean latency from the onset of PACs to their first action, *(xiii)* the mean number of actions within each PAC, *(xiv)* the total number of 24×24 game coordinate grids viewed by player per time frame, *(xv)* the number of battle items trained per time frame, *(xvi)* the number of unique units made per time frame, *(xvii)* the number of complex battle items trained per time frame, *(xviii)* the number of abilities requiring specific targeting instructions used per time frame, *(xix)* Time stamp of game's last recorded event ($\dots, X_{19}$), and *(xx)* the target XP level class (Y).

as[3] $\mathcal{Y}^{\mathcal{L}_e}$. As per previous discussions, the aim of machine learning algorithms is to find an hypothesis h maximizing such loss function, which might be described as the discrepancy between the prediction by h and the expected result. Given a distance[4] measure $\delta_{\mathcal{Y}}$ among the classes of interest, we can define the *least mean squares* loss function as follows:

$$loss^{\delta_{\mathcal{Y}}}_{\text{lms}}(h, E) := \sum_{(\mathbf{x},y)\in E} \delta^2_{\mathcal{Y}}(y, h(\mathbf{x}))$$

For example, if $\mathcal{Y} \subseteq \mathbb{R}^n$ for any $n \in \mathbb{N}$, then we can use $\delta^2_{\mathbb{R}^n}$ as the euclidean distance $\delta^2_{\mathbb{R}^n}(\mathbf{x}, \mathbf{x}') := \|\mathbf{x} - \mathbf{x}'\|$. The training problem can be summarized as follows:

$$ML(loss, D, \mathcal{L}_h) := \arg\min_{h'\in\mathcal{L}_h} loss(h', E)$$

With respect to binary classification problems, in most real world algorithms, we're not necessarily going to learn a discrete $h\colon \mathcal{L}_e \to \{ 0, 1 \}$, but we're going to learn a function approximating such values in $\mathbb{R}$ (e.g., a regression function), so $\tilde{h}\colon \mathcal{L}_e \to [0, 1]$ is learned instead. In these situations, we might define the hypothesis from the regression function as follows:

$$h(\mathbf{x}) := \arg\min_{y\in\mathcal{Y}} \delta_{\mathcal{Y}}(y, \tilde{h}(\mathbf{x}))$$

Data Mining Let us now suppose that we deal with data mining problems, where a possible outcome of the algorithm might be a set of elements $B \subseteq \mathcal{L}_e$. We know from set theory that we can always express such set as an *indicator function* $\mathbf{1}_B\colon \mathcal{L}_e \to \{0, 1\}$ determining whether any element of $\mathcal{L}_e$ is also in B or not[5]. In this scenario we can still model our hypotheses as per previous discussions: for each possible returned set B, we can return an hypothesis h_B which is actually the indicator function $\mathbf{1}_B$. This also implies that each data mining algorithm will retrieve only the *predicted positive* examples.

Contrarily to machine learning algorithms, data mining algorithms are not going to return a single hypothesis, but all such hypotheses that satisfy a quality criterion $\mathcal{Q}$, that often requires B: when such set is not explicitly given but h is returned instead, we can define a cover function $\mathbf{co}(h, D) = \{ x \in D \mid h(x) = 1 \}$ identifying the set of all the items that are covered by the hypothesis h. An intuitive graphical representation of such function is given at Figure 2b. Data mining algorithms are often interested in returning only the hypothesis exhibiting frequent patterns from the data: in order to do so, we can define the *min-frequency* predicate that is going to return only the hypothesis above a given threshold θ:

$$\mathcal{Q}_\theta(h, D) := |\mathbf{co}(h, D)| \geq \theta$$

This means that we can generalize all the possible data mining problems as follows:

$$DM(\mathcal{Q}, D, \mathcal{L}_h) := \{ h \in \mathcal{L}_h \mid \mathcal{Q}(h, D) \text{ holds} \}$$

Albeit it might seem quite inefficient to visit all the possible hypotheses in $\mathcal{L}_h$, we might provide an intuition[6] of how we can efficiently do so by pruning most of the non-valid hypotheses: for binary classification problems, the set of all the possible hypotheses is $2^{\mathcal{L}_e}$, so each possible predicted positive example in $\wp(\mathcal{L}_e)$ represents the outcome of one single hypothesis. Given that there is an isomorphism between $\mathcal{L}_h$ and $\wp(\mathcal{L}_e)$ (Lemma A), we can search directly the hypotheses from the latter space. Let us now also suppose that we want to use the min-frequency predicate as a possible quality criterion: given that all the generalizations of an hypothesis will satisfy such criterion (Lemma 2), then as a dual condition none of the specializations of the hypotheses not-satisfying such criterion will satisfy it [2]. As a result, we can define top-down search algorithms starting from the most general hypothesis and specializing it: we can stop expanding an hypothesis when such hypothesis doesn't meet the quality criterion.

[3] See `https://en.wikipedia.org/wiki/Power_set#Representing_subsets_as_functions`.

[4] See the first tutorial on algorithms and data structure for a formal definition of a distance metric.

[5] $\mathbf{1}_B(\mathbf{x}) := \begin{cases} 1 & \mathbf{x} \in B \\ 0 & \mathbf{x} \notin B \end{cases}$

[6] For the formal proofs see Appendix A.

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Figure 3: Displaying the performance of a learning algorithm.

2 Multiclass Classification from Binary Classification

At this stage, we mainly discussed binary classification problems. This decision is based on the fact that both (1) these classifiers are simple enough to be understood in a few days time, and (2) it is always possible to reduce an arbitrary classification problem to a binary classification problem.

- **one-versus-all**: for each class $y \in \mathcal{Y}$, train a binary (e.g. learning) algorithm so that it will learn whether any $x \in \mathcal{L}_h$ belongs to the class y or not. For each of these algorithms, obtain the approximated hypothesis function $\tilde{h}_y$. Now, we can define the ensemble hypothesis as follows:

$$h_{\mathrm{OvA}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \tilde{h}_y(\mathbf{x})$$

- **one-versus-one**: for each class $y \in \mathcal{Y}$, initialize a binary (e.g., learning) algorithm $\mathcal{A}_{(y_1,y_2)}$ for each distinct pair $(y_1, y_2) \in \mathcal{Y}^2$, infer the binary hypothesis $\mathrm{Im}(h_{(y_1,y_2)}) = \{\, y_1, y_2 \,\}$ and store it in a pool ($|C| = \frac{|\mathcal{Y}|(|\mathcal{Y}|-1)}{2}$). Then, associate $\mathbf{x}$ to the class winning the following maximum vote strategy.

$$h_{\mathrm{OvO}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} |\, \{\, h_{(y_1,y_2)} \in C \mid h_{(y_1,y_2)}(\mathbf{x}) = y \,\} \,|$$

Furthermore, this technique is independent from the specific in-silico model of choice, given that it only deals with the returned hypotheses.

3 Confusion Matrix for Binary Classification

Last, we can use confusion matrices to display the performance of a classification algorithm. In fact, these matrices can be used for both determine how many predicted positive elements are actually true (**precision**), and how many positive elements were actually retrieved even though wrongly classified (**recall**). Based on the matrix in Figure 3, we can define precision and recall as the following metrics:

$$\textbf{Precision} = \frac{|p \cap p'|}{|P|} \qquad \textbf{Recall} = \frac{|p \cap p'|}{|P'|}$$

This confusion matrix can be easily generalized for the multi-class classification problem.

References

[1] Christopher M. Bishop. *Fundamentals of Database Systems*. Pearson, 7th edition, 2015.

[2] Luc De Raedt. *Logical and Relational Learning*. Springer Verlag, Germany, 2008.

A Formal Proofs

Let us consider binary classification problems. One natural way to structure the search space is to employ the generality relation $\preceq$: we can state that h_2 is more general that h_1 if all the examples covered by h_1 are also covered by h_2, that is:

$$h_2 \preceq h_1 \Leftrightarrow \mathbf{co}(h_1, D) \subseteq \mathbf{co}(h_2, D)$$

Now, given that the hypotheses are just indicator functions for sets $B_i \in \wp(\mathcal{L}_e)$, we can express such relations in terms of the predicted positive sets

Lemma 1. *Under the assumption that $h_1 = \mathbf{1}_{B_1}$ and $h_2 = \mathbf{1}_{B_2}$, then for binary decision problems we have that $h_2 \preceq h_1 \Leftrightarrow B_1 \subseteq B_2$.*

Proof. If we assume to know that, for any subset $S \subseteq T$ of T, the intersection $S \cap T$ is equal to S[7] ($\mathbf{T}$), then we can provide the following proof:

$$\begin{aligned} h_2 \preceq h_1 &\Leftrightarrow \mathbf{co}(h_1, D) \subseteq \mathbf{co}(h_2, D) \\ &\Leftrightarrow (B_1 \cap D) \subseteq (B_2 \cap D) \\ &\overset{\mathbf{T}}{\Leftrightarrow} (B_1 \cap D) \cap (B_2 \cap D) = (B_1 \cap D) \\ &\Leftrightarrow (B_1 \cap B_2) \cap D = (B_1) \cap D \\ &\Leftrightarrow B_1 \cap B_2 = B_1 \\ &\Leftrightarrow B_2 \subseteq B_2 \end{aligned}$$

□

We want now to show that min-support can be used as a pruning quality criterion while visiting $\wp(\mathcal{L}_e)$. First, we need to prove that such criterion is anti-monotonic. The definition of anti-monotonicity for h_1, h_2 is given as follows:

$$\forall D \subseteq \mathcal{L}_e.\ (h_1 \preceq h_2) \wedge \mathcal{Q}(h_2, D) \Rightarrow \mathcal{Q}(h_1, D)$$

Lemma 2. *The min-support quality criterion is anti-monotonic with respect to a given set D.*

Proof. By Lemma 1, we can rewrite $h_1 \preceq h_2$ as $B_2 \subseteq B_1$ (**H1**). Also, the min-support quality criterion for h_2 becomes $|B_2 \cap D| \geq \theta$, thus implying that we know a $x \in \mathbb{R}^+$ for which $|B_2 \cap D| = \theta + x$ (**H2**). If we now expand the min-frequency definition for h_1, then we have to prove that $|B_1 \cap D| \geq \theta$. By expanding **H1**, we have that we know a set S for which $|(B_2 \cup S) \cap D| = |(B_2 \cap D) \cup (S \cap D)| \geq \theta$. We can now prove:

$$\begin{aligned} |(B_2 \cup S) \cap D| = |(B_2 \cap D) \cup (S \cap D)| \geq \theta &\Leftrightarrow \exists y \in \mathbb{R}^+. |B_2 \cap D| + y = \theta \\ &\overset{\mathbf{H2}}{\Leftrightarrow} \exists y \in \mathbb{R}^+. \theta + x + y = \theta \\ &\Leftrightarrow \theta \geq \theta \end{aligned}$$

□

[7] See `https://proofwiki.org/wiki/Intersection_with_Subset_is_Subset`