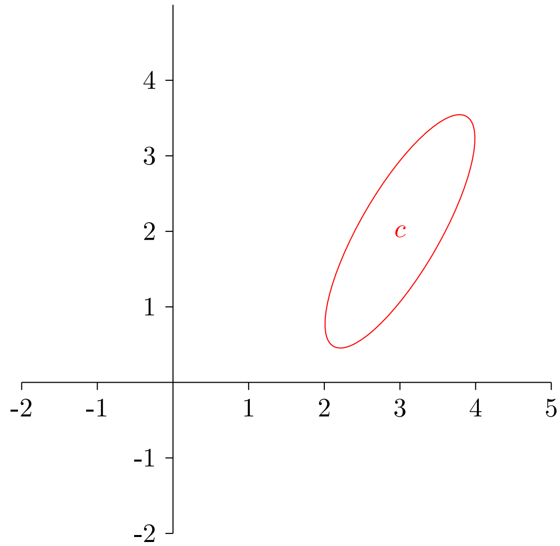
Question 1.

Consider the instance space $X = \mathbb{R}^2$ and a concept $c \subseteq X$ given as



(a) Name some concept classes $\mathcal{C}$ that contain c.

(b) Recall the SVM algorithm and decision trees. What hypothesis classes $\mathcal{H}$ do they work with and how do they internally represent their hypotheses? Would they be appropriate to learn the concept c?

Answer:

(a) The broadest concept class could be the powerset of the instance space 2^X. However, that is an extremely large class without a very nice representation.

Instead, we can consider the concept class of all conic sections, which can be concisely represented by the inequality

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F \leq 0.$$

We could do better still by only considering the concept class of all ellipses. Then, we would have certain limitations imposed on the parameters of the inequality above.

(b) The SVM algorithm searches for a linear separator that maximizes the margin. Hence, its $\mathcal{H}$ are lines (half spaces) which are represented by their slope and a bias (using some finite floating-point representation). The vanilla SVM would not be suitable to learn c as it is not a linearly separable concept. However, we could use some appropriate kernels.

Decision trees split the space into a set of axis-aligned rectangles. They are represented by a specialized binary tree which has indicator functions of instances placed in each of its internal nodes. Decision trees might perform well on learning c, although that is not easily judged and would also be dependant on the tree's depth which is usually the model's hyperparameter.

Question 2.

Consider the generalization algorithm for learning conjunctions.

(a) What is the algorithm's mistake bound? Describe a scenario when we achieve it.

(b) Assume we work on $n = 4$ logical variables. Assume the sequence of examples

$x_1 = (1, 0, 0, 1)$ with label $y_1 = 1$
$x_2 = (1, 1, 0, 0)$ with label $y_2 = 0$
$x_3 = (0, 1, 1, 1)$ with label $y_3 = 1$
$x_4 = (1, 1, 1, 0)$ with label $y_4 = 0$

Write the initial (internal) hypothesis as well as how it will gradually change when processing examples above.

Assuming that the concept class is in fact a set of all conjunctions, can we claim that the final hypothesis describes the target concept?

(c) Adapt the algorithm to learn k-DNF. What is the mistake bound now? Are we still learning efficiently?

(d) How can we use the new algorithm to learn k-clause CNF? What is *improper* learning?

Answer:

(a) MB$= n + 1$.

We can achieve the bound when the concept is the empty set (i.e., the concept is a *tautology*) and we receive examples such that $x_1 = \mathbf{1}$ and $x_i = \mathbf{1} - \mathbf{e}_j$ for all $i > 1$ and some $j \in [n]$.

(b) The hypothesis changes as follows:

$$h_0 = p_1 \wedge \neg p_1 \wedge p_2 \wedge \neg p_2 \wedge p_3 \wedge \neg p_3 \wedge p_4 \wedge \neg p_4$$
$$h_1 = p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4$$
$$h_2 = p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4$$
$$h_3 = p_4$$
$$h_4 = p_4$$

We have made 2 mistakes so far. The mistake bound is $n + 1 = 5$. We can't claim h_4 to be the target concept.

Alternatively, the only other hypothesis we might produce is the empty conjunction, i.e., a tautology. Since, we have seen examples x_2 and x_4 with a negative label, the target concept can't be a tautology. Hence, h_4 is the target concept.

(c) We introduce a new logical variable q_i for each k-disjunction (a clause made up of at most k literals). The initial hypothesis will be a conjunction over all q_i, and it will be updated by the usual procedure.

Labels for each obtained example will be negated. Thus, we will be learning the complementary concept (a negation of a conjunction is a disjunction).

Once the learning is finished, negate the final hypothesis (consequently, negate each q_i meaning that each k-disjunction is turned into a k-conjunction) to obtain the desired k-DNF.

There are

$$A = \sum_{j=0}^{k} \binom{n}{j} 2^j$$

different k-disjunctions. The new mistake bound is $A + 1 \leq \text{poly}(n)$. Additionally, we can evaluate the truth value of each k-disjunction in $\mathcal{O}(n)$, so we are still learning efficiently.

(d) It holds that k-clause CNF $\subseteq$ k-DNF (try "multiplying out"). We can use the algorithm from above to learn k-clause CNF efficiently.

However, the final hypothesis outputted will be a k-DNF, not the actual k-clause CNF, which is called *improper* learning ($\mathcal{C} \neq \mathcal{H}$).