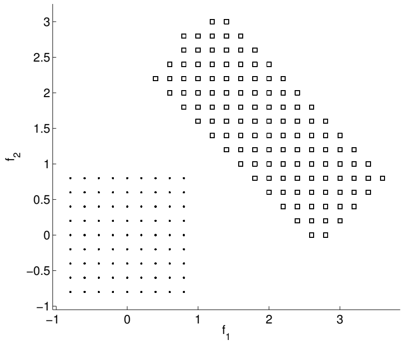
CS188: Exam Practice Session 10 Solutions

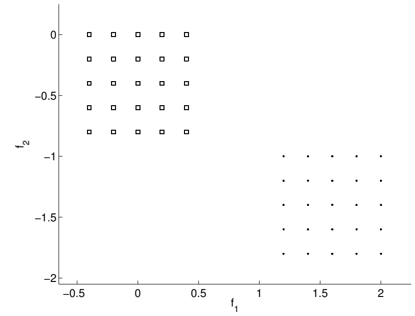
Q1. Naïve Bayes Modeling Assumptions

You are given points from 2 classes, shown as rectangles and dots. For each of the following sets of points, mark if they satisfy all the Naïve Bayes modelling assumptions, or they do not satisfy all the Naïve Bayes modelling assumptions. Note that in (c), 4 rectangles overlap with 4 dots.

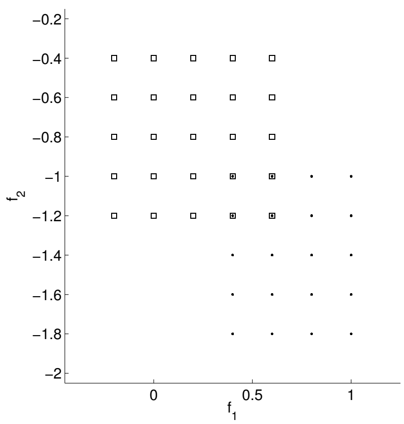
The conditional independence assumptions made by the Naïve Bayes model are that features are conditionally independent when given the class. Features being independent once the class label is known means that for a fixed class the distribution for f_1 cannot depend on f_2, and the other way around. Concretely, for discrete-valued features as shown below, this means each class needs to have a distribution that corresponds to an axis-aligned rectangle. No other assumption is made by the Naïve Bayes model. Note that linear separability is not an assumption of the Naïve Bayes model—what is true is that for a Naïve Bayes model with all binary variables the decision boundary between the two classes is a hyperplane (i.e., it's a linear classifier). That, however, wasn't relevant to the question as the question examined which probability distribution a Naïve Bayes model can represent, not which decision boundaries.



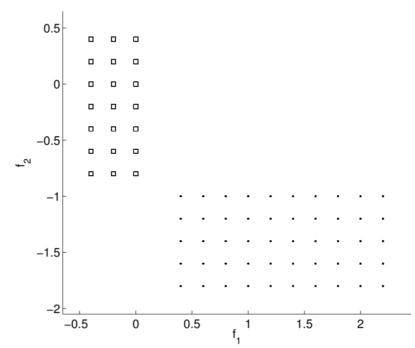
(a) ○ Satisfies ● Does not Satisfy



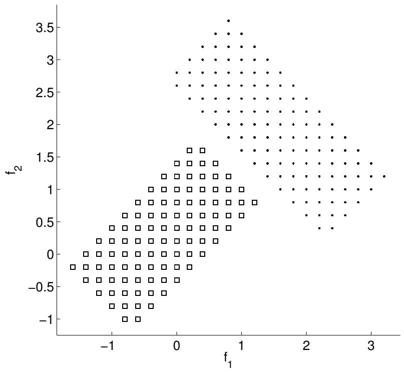
(b) ● Satisfies ○ Does not Satisfy



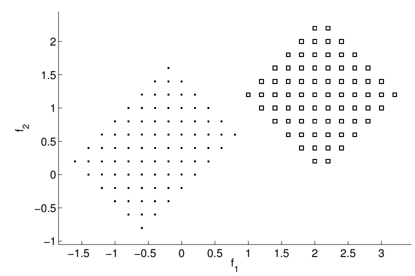
(c) ● Satisfies ○ Does not Satisfy



(d) ● Satisfies ○ Does not Satisfy



(e) ○ Satisfies ● Does not Satisfy

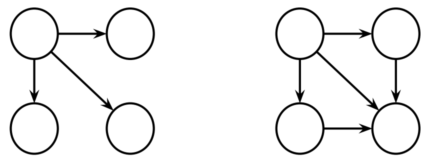


(f) ○ Satisfies ● Does not Satisfy

A note about feature independence: The Naïve Bayes model assumes features are conditionally independent given the class. Why does this result in axis-aligned rectangles for discrete feature distributions? Intuitively, this is because fixing one value is uninformative about the other: within a class, the values of one feature are constant across the other. For instance, the dark square class in (b) has $f_1 \in [-0.5, 0.5]$ and $f_2 \in [-1, 0]$ and fixing one has no impact on the domain of the other. However, when the features of a class are not axis-aligned then fixing one limits the domain of the other, inducing dependence. In (e), fixing $f_2 = 1.5$ restricts f_1 to the two points at the top, whereas fixing $f_2 = 0$ gives a larger domain.

Q2. Model Structure and Laplace Smoothing

We are estimating parameters for a Bayes' net with structure G_A and for a Bayes' net with structure G_B. To estimate the parameters we use Laplace smoothing with $k = 0$ (which is the same as maximum likelihood), $k = 5$, and $k = \infty$.



G_A $\quad$ G_B

Let for a given Bayes' net BN the corresponding joint distribution over all variables in the Bayes' net be P_{BN} then the likelihood of the training data for the Bayes' net BN is given by

$$\prod_{x_i \in \text{Training Set}} P_{BN}(x_i)$$

Let $\mathcal{L}_A^0$ denote the likelihood of the training data for the Bayes' net with structure G_A and parameters learned with Laplace smoothing with $k = 0$.
Let $\mathcal{L}_A^5$ denote the likelihood of the training data for the Bayes' net with structure G_A and parameters learned with Laplace smoothing with $k = 5$.
Let $\mathcal{L}_A^\infty$ denote the likelihood of the training data for the Bayes' net with structure G_A and parameters learned with Laplace smoothing with $k = \infty$.
We similarly define $\mathcal{L}_B^0$, $\mathcal{L}_B^5$, $\mathcal{L}_B^\infty$ for structure G_B.

For a given Bayes' net structure, maximum likelihood parameters would give them maximum likelihood on the training data. As you add more and more smoothing you tend to move away from the MLE and get lesser and lesser likelihood on the training data. Hence parts (a), (b), and (c).

Across models, G_B can represent a larger family of models and hence has a higher MLE estimate than we get with using G_A. Hence, we have (d). In (e), in the case of infinite smoothing, all variables become independent and equally probable to take any value for both G_A and G_B, hence giving equal training likelihood. (f) follows from (a) and (d). A priori, we can not say anything about (g) because the going from $\mathcal{L}_A^0$ to $\mathcal{L}_B^5$ we increase model power (which would increase likelihood) and increase smoothing (which would decrease likelihood) and a priori we dont know which effect would dominate.

For each of the questions below, mark which one is the correct option.

(a) Consider $\mathcal{L}_A^0$ and $\mathcal{L}_A^5$

○ $\mathcal{L}_A^0 \le \mathcal{L}_A^5$ ● $\mathcal{L}_A^0 \ge \mathcal{L}_A^5$ ○ $\mathcal{L}_A^0 = \mathcal{L}_A^5$ ○ Insufficient information to determine the ordering.

(b) Consider $\mathcal{L}_A^5$ and $\mathcal{L}_A^\infty$

○ $\mathcal{L}_A^5 \le \mathcal{L}_A^\infty$ ● $\mathcal{L}_A^5 \ge \mathcal{L}_A^\infty$ ○ $\mathcal{L}_A^5 = \mathcal{L}_A^\infty$ ○ Insufficient information to determine the ordering.

(c) Consider $\mathcal{L}_B^0$ and $\mathcal{L}_B^\infty$

○ $\mathcal{L}_B^0 \le \mathcal{L}_B^\infty$ ● $\mathcal{L}_B^0 \ge \mathcal{L}_B^\infty$ ○ $\mathcal{L}_B^0 = \mathcal{L}_B^\infty$ ○ Insufficient information to determine the ordering.

(d) Consider $\mathcal{L}_A^0$ and $\mathcal{L}_B^0$

● $\mathcal{L}_A^0 \le \mathcal{L}_B^0$ ○ $\mathcal{L}_A^0 \ge \mathcal{L}_B^0$ ○ $\mathcal{L}_A^0 = \mathcal{L}_B^0$ ○ Insufficient information to determine the ordering.

(e) Consider $\mathcal{L}_A^\infty$ and $\mathcal{L}_B^\infty$

○ $\mathcal{L}_A^\infty \le \mathcal{L}_B^\infty$ ○ $\mathcal{L}_A^\infty \ge \mathcal{L}_B^\infty$ ● $\mathcal{L}_A^\infty = \mathcal{L}_B^\infty$ ○ Insufficient information to determine the ordering.

(f) Consider $\mathcal{L}_A^5$ and $\mathcal{L}_B^0$

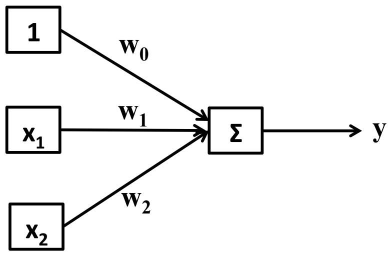
● $\mathcal{L}_A^5 \leq \mathcal{L}_B^0$ ○ $\mathcal{L}_A^5 \geq \mathcal{L}_B^0$ ○ $\mathcal{L}_A^5 = \mathcal{L}_B^0$ ○ Insufficient information to determine the ordering.

(g) Consider $\mathcal{L}_A^0$ and $\mathcal{L}_B^5$

○ $\mathcal{L}_A^0 \leq \mathcal{L}_B^5$ ○ $\mathcal{L}_A^0 \geq \mathcal{L}_B^5$ ○ $\mathcal{L}_A^0 = \mathcal{L}_B^5$ ● Insufficient information to determine the ordering.

Q3. Perceptron

(a) Consider the following perceptron, for which the inputs are the always 1 feature and two binary features $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1\}$. The output $y \in \{0, 1\}$.



$$y = \begin{cases} 1 & \text{if } (w_0 + w_1 \cdot x_1 + w_2 \cdot x_2) > 0 \\ 0 & \text{otherwise} \end{cases}$$

(i) Which one(s) of the following choices for the weight vector $[w_0\ w_1\ w_2]$ can classify y as $y = (x_1 \text{ XOR } x_2)$? XOR is the logical exclusive or operation, which equals to zero when x_1 equals to x_2 and equals to one when x_1 is different from x_2.

○ [1 1 0]
○ [-1.5 1 1]
○ [-2 1 1.5]
○ Any weights that satisfy $(-w_1 - w_2) < w_0 < \min(0, -w_1, -w_2)$.
● No weights can compute the XOR logical relation.

(ii) Which one(s) of the following choices for the weight vector $[w_0\ w_1\ w_2]$ can classify y as $y = (x_1 \text{ AND } x_2)$? Here AND refers to the logical AND operation.

○ [1 1 0]
● [-1.5 1 1]
● [-2 1 1.5]
● Any weights that satisfy $(-w_1 - w_2) < w_0 < \min(0, -w_1, -w_2)$.
○ No weights can compute the logical AND relation.

The truth table for XOR and AND logical operations is:

x_1	x_2	XOR	AND
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0

In order to classify y as $y = x_1$ XOR x_2, we need to have $\begin{cases} w_0 + w_1 \cdot 1 + w_2 \cdot 1 \leq 0 \\ w_0 + w_1 \cdot 1 + w_2 \cdot 0 > 0 \\ w_0 + w_1 \cdot 0 + w_2 \cdot 1 > 0 \\ w_0 + w_1 \cdot 0 + w_2 \cdot 0 \leq 0 \end{cases}$. It is equivalent to $\begin{cases} w_0 \leq -w_1 - w_2 \\ w_0 > -w_1 \\ w_0 > -w_2 \\ w_0 \leq 0 \end{cases}$, which is impossible. The reason is $w_1 > 0$, $w_2 > 0$ and $-w_1 < w_0 \leq -w_1 - w_2$, which makes $w_2 < 0$. Contradiction! So no weights can classify y as x_1 XOR x_2.

Similarly, to classify y as $y = x_1$ AND x_2, we need to have $\begin{cases} w_0 + w_1 \cdot 1 + w_2 \cdot 1 > 0 \\ w_0 + w_1 \cdot 1 + w_2 \cdot 0 \leq 0 \\ w_0 + w_1 \cdot 0 + w_2 \cdot 1 \leq 0 \\ w_0 + w_1 \cdot 0 + w_2 \cdot 0 \leq 0 \end{cases}$. It is equivalent to $\begin{cases} w_0 > -w_1 - w_2 \\ w_0 \leq -w_1 \\ w_0 \leq -w_2 \\ w_0 \leq 0 \end{cases}$, which is $(-w_1 - w_2) < w_0 \leq \min(0, -w_1, -w_2)$. So weight [-1.5 1 1] and [-2 1 1.5] and any weights that satisfy $(-w_1 - w_2) < w_0 \leq \min(0, -w_1, -w_2)$ can be used to classify $y = x_1$ AND x_2.

(b) Consider a multiclass perceptron with initial weights $w_A = [1\ 0\ 0]$, $w_B = [0\ 1\ 0]$ and $w_C = [\ 0\ 0\ 1]$. For prediction, if there is a tie, A is chosen over B over C. The following table gives a sequence of three training examples to be incorporated. When incorporating the second training example, start from the weights obtained from having incorporated the first training example. Similarly, when incorporating the third training example, start from the weights obtained from having incorporated the first training example and the second training example. Fill in the resulting weights in each row.

feature vector	label	w_A	w_B	w_C
		[1 0 0]	[0 1 0]	[0 0 1]
[1 -2 3]	A	[2 -2 3]	[0 1 0]	[-1 2 -2]
[1 1 -2]	B	[2 -2 3]	[1 2 -2]	[-2 1 0]
[1 -1 -4]	B	[2 -2 3]	[1 2 -2]	[-2 1 0]

Initial weights: $w_A = [1\ 0\ 0]$, $w_B = [0\ 1\ 0]$ and $w_C = [0\ 0\ 1]$.

After first training example with feature vector [1 -2 3], the algorithm will predict $y = argmax_y(w_y \cdot f) = C$, while the true label is $y^* = A$, so we update $w_A \leftarrow w_A + f =$ [2 -2 3] and $w_C \leftarrow w_C - f =$ [-1 2 -2].

After second training example with feature vector [1 1 -2], the algorithm will predict $y = argmax_y(w_y \cdot f) = C$, while the true label is $y^* = B$, so $w_B \leftarrow w_B + f =$ [1 2 -2] and $w_C \leftarrow w_C - f =$ [-2 1 0].

After third training example with feature vector [1 -1 -4], the algorithm will predict $y = argmax_y(w_y \cdot f) = B$, while the true label is $y^* = B$, so no weight updating is needed.