Mid-term exam, Machine Learning (MDS), April 3rd, 2024

Note: I tried to express what in my view are good answers; which does not mean that in some of the questions there may be different answers that are also plausible. Except problem 2.

Question 1: Given the example of categorical Naive Bayes from the theory slides:

Outlook	*Temperature*	*Humidity*	*Wind*	*Play ball*
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

a) How many parameters uses Naive Bayes for this problem? And with Laplace smoothing?

For the **prior** distribution $p(Y)$ where Y is the class, we use a categorical distribution. We see two possible classes (Yes/No), so we need two parameters p_{Yes}, p_{No}; well really just one free parameter as they need to add up to one.

For the **class-conditional** distribution $p(X = |Y = y)$, we need to model each feature conditional on each possible class independently. So, for each feature X_i, we need to model $p(X_i|Y = Yes)$ and $p(X_i|Y = No)$ using a categorical distribution. Categorical distributions have as many parameters as different values (modalities) the feature can take. Well, since all parameters of the categorical distribution need to add up to one, we could count the number of values *minus one*.

Then, for each feature (here we have four: Outlook, Temperature, Humidity, and Wind) we need one parameter for each value that the feature can take *minus 1*; this for each of the 2 possible class values. So we should count $2 \times \left[\sum_f (nvalues(f) - 1)\right]$, where f stands for each input feature and $nvalues(f)$ stands for the number of values or *modalities* the feature takes.

In this case:

- *Outlook* : contains values Sunny, Overcast, Rain (3 values)
- *Temperature*: contains values Hot, Mild, Cool (3 values)
- *Humidity*: contains values Normal, High (2 values)
- *Wind*: contains values Weak, Strong (2 values)

All together, the cat NB model has $1 + 2 \times [2 + 2 + 1 + 1] = 13$ parameters. Answers that count the bound parameter in each categorical distribution are accepted as well: $2 + 2 \times [3 + 3 + 2 + 2] = 22$.

Laplace smoothing only affects how we estimate these parameters and not how many there are, so number of parameters is the same as before.

b) What steps would you take in order to use Gaussian Naive Bayes for this problem? Do you think it would make sense? Would it need more or less parameters than categorical NB?

Gaussian NB needs numerical features as input, so we would need to transform features into numeric values. Due to the nature of the values we find (they seem to be ordinal, as there is order between levels of the discrete features) an easy way would be to use integers to encode de different levels (in brackets in what follows); we can encode binary features $0/1$:

- *Outlook* : contains values Sunny(1), Overcast(2), Rain(3)
- *Temperature*: contains values Hot(1), Mild(2), Cool(3)
- *Humidity*: contains values Normal(0), High(1)
- *Wind*: contains values Weak(0), Strong(1)

Then technically we could apply Gaussian NB and model each feature using a univariate normal distribution. So, we have to estimate mean and variance for each feature. This way, we would need to estimate, for each of the two classes, 4 means and 4 variances, which totals $1 + 2 \times 8 = 17$ parameters for the **prior** and **class-conditional** distribution.

However, using normal distributions to model these discrete feature does not make much sense as each column does not look normal at all. For example:

- *Outlook* for class *Yes*: { 'Sunny(1)': 2, 'Overcast(2)': 4, 'Rain(3)': 3, 'Sunny(1)': 2}
- *Outlook* for class *No*: {'Sunny(1)': 3, 'Overcast(2)': 0, 'Rain(3)': 2}

If we are to build a histogram with these values, visually we can see that it is far from gaussian.

In the case of temperature:

- *Temp* for class *Yes*: {'Hot(1)': 2, 'Mild(2)': 4, 'Cool(3)': 3}
- *Temp* for class *No*: {'Hot(1)': 2, 'Mild(2)': 2, 'Cool(3)': 1}

In this case things it looks a little better for the *Yes* distribution as it *sort of* looks symmetric around value Mild(2); however the *No* distrubution is again far from being gaussian. Additionally, we have a small amount of data so there is that added problem (estimates will be very poor).

Finally, for the binary features we only have two values possible, so modelling that with a gaussian is a bad idea.

–

There is another possible answer which could apply one-hot encoding to all features resulting in $3 + 3 + 2 + 2 = 12$ binary features, so we would need $1 + 2 \times 2 \times 12 = 49$ parameters (24 means, 24 variances, and 1 for the prior categorical distribution). Which looks like an even worse solution since it does not make sense to model binary data with a gaussian distribution.

Question 2: In an univariate regression problem, find the equation for the best horizontal line that fits a set of input training points using least squares. You can assume your input data is $\{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$ where each $(x_i, y_i) \in \mathbb{R}^2$.

Horizontal lines correspond to equations of the form $y = b$, so they are constant across the whole span of x. Basically we only have one parameter to adjust, which is $b \in \mathbb{R}$. The square error of horizontal line $y = b$ is given by $E(b) = \sum_{i=1}^{n} (y_i - b)^2$. In order to find the value of b that minimizes the error, we take the derivative w.r.t. b and set it equal to 0:

$$\frac{d}{db} E(b) = \frac{d}{db} \sum_{i=1}^{n} (y_i - b)^2 = \sum_i \frac{d}{db} (y_i - b)^2 = \sum_i 2(b - y_i) = 2bn - 2\sum_i y_i$$

Here our normal equation states that the optimum for b has to satisfy that $\sum_i y_i = bn$, equivalently, $b = \frac{1}{n} \sum_i y_i$. So essentially when b is the average of all target values, we see that the square error is minimized (second derivative is positive so all is good).

Question 3: You have access to a set of input finite (labelled) data. Explain in your own words how to:

a) Estimate a model's generalization performance as accurately as possible

- If the model is handed to us and there is no learning or adjustment of parameters to be done, and assuming the adjustment have been done using other data or the model has been found by some other means, then we can use the available data as a "test set" and use it to estimate generalization of the model.
- If there is adjustments of parameters to be made, then a resampling protocol seems necessary
  - If the model has no hyper-parameters, then using loocv error or cross-validation error in general (better if k is large) as and estimate for the generalization error would be acceptable.
  - If hyper-parameters are to be tuned, then it is best to partition the data into training and test sets, do the learning on the training partition (find best hyper-params with cross-validation and re-train final model with the train partition with optimal hyper-params). Then, estimate final model's performance with the test partition.

b) Choose among competing models

- Like before, if the models are already tuned and have not had access to this data, then we can use the available data as "validation data" in order to select the model that minimizes error on the available data.
- If models need to be tuned and/or hyper-params are to be optimized, then a resampling protocol is necessary. Any variant of cross-validation over the available data seems like a good option.

c) Estimate the parameters of a model

This is what learning algorithms do; depending on the type of model we will have some ad-hoc way to estimate its parameters. A common choice is to use a reasonable error function and optimize parameters as to minimize this error function over the available data. Another slightly more sophisticated option is to include a regularization term in the error function and then adjust parameters in order to minimize this regularized error. In any case, estimating parameters involves traversing the space of possible values for the parameters in order to optimize some performance metric.