Final exam, ML (MDS + BDMA + MIRI)

21st of June, 2023

- In all your answers, conciseness and clarity will be most appreciated.
- All questions have equal weight.
- Exam duration: **2h**

Question 1

Describe the machine learning technique known as *bagging*. What are its key components? What are its main advantages?

Bagging is a ensemble learning technique whose name comes from *bootstrap + aggregation*.

The idea is to generate a diverse set of (base) models by training them from a series of *copies* of the original data. The final model's predictions are computed by building a consesus from each base models' predictions (average in regression tasks, majority vote in classification tasks). The way to generate the base models is by creating bootstrap copies of the original data by sampling with replacement and training a different model from each of these copies. The technique is ideal if the base models have a strong tendency to overfit and so the pool of generated models will be diverse and not too correlated. In addition, we get validation sets "for free" for each trained model by considering those samples that were left out from the copy that they were trained on (these validation sets are used to compute the so-called *out-of-bag validation error*, which can be used for adjusting hyper-parameters w/o the need of cross-validation or other resampling methods).

Among its main advantages we find that it can be *parallelized* and it can *reduce the variance* of the base models.

Question 2

Describe *Bayes rule* in the context of classification. Give examples of classifiers based on this rule.

Classifiers that make use of *Bayes rule* are the ones known as *generative classifiers*. We have seen several examples of such classifiers in class, examples are *QDA* with their variants *LDA* and *RDA*; and *Naive Bayes* classifiers as well.

In general, in a classification setting, we want to model the posterior probability $P(Y|X = x)$, where x stands for an observed object and Y stands for the random variable that can take values from the discrete set of class labels. Assume in this answer this set is given by class labels $\{c_k\}_{k=1,\dots,K}$ for some natural number $K > 1$.

With *generative classifiers* we estimate posterior probabilities $P(Y = c_k|X = x)$ by invoking *Bayes rule*:

$$P(Y = c_k|X = x) = \frac{P(X = x|Y = c_k) \times P(Y = c_k)}{P(X = x)}$$

We seek to find the class c_k that maximizes $P(Y = c_k|X = x)$ for a given x, therefore we can drop the denominator since it is constant w.r.t. the class c_k, and so:

$$\arg\max_k P(Y = c_k|X = x) = \arg\max_k P(X = x|Y = c_k) \times P(Y = c_k)$$

As a result, generative classifiers obtain class scores for all classes c_k by computing for each class c_k

$$s_k = P(X = x|Y = c_k) \times P(Y = c_k).$$

The class prediction is the class with maximum score. If one wants a probability distribution for $P(Y|X = x)$, we need to normalize the scores, i.e. $P(Y = c_k|X = x) = \frac{s_k}{\sum_{k'} s_{k'}}$.

The term $P(Y = c_k)$ are the *class priors* and are estimated using the proportion of examples of each class; the term $P(X = x|Y = c_k)$ are the class-conditional distributions (distribution of observations for each class).

Each different *generative classifier* makes different assumptions on the class-conditionals. For example, in *QDA* these are assumed to by multivariate Gaussians. Additionally, in Gaussian Naive Bayes, the multivariate gaussians are assumed to have diagonal covariance matrices, so essentially assuming class-conditional independence between pairs of variables.

Question 3

Consider the following univariate regression dataset with three input examples:

$$D = \{(1,1),(2,2),(3,1)\}.$$

a) Compute the mean square *loocv* error of ordinary linear regression.

loocv leaves one example as validation, while training on the remaining ones. Since we have three examples, this means that *loocv* error will be obtained as the average of the three following validation errors:

1) training on first two examples, validating on the third one; in this case, the *least square* solution to training data $\{(1,1),(2,2)\}$ is the $y = x$ line, so prediction on third example $x_3 = 3$ is $\hat{y}_3 = 3$; validation error in this split is $(3-1)^2 = 4$.
2) training on first and third examples, validating on the second one; in this case, the *least squares* solution to training data $\{(1,1),(3,1)\}$ is the flat line $y = 3$, and so prediction on second example is constant $\hat{y}_2 = 3$; validation error in this split is $(3-2)^2 = 1$.
3) training of second and third example, validating on the first one; in this case, the *least squares* solution to training data $\{(2,2),(3,1)\}$ is the line $y = 4 - x$, and so prediction on first example $x_1 = 1$ is $\hat{y}_1 = 4 - 1 = 3$; validation error in this split is $(3-1)^2 = 4$.

So, their average (*loocv* error) is $\frac{4+1+4}{3} = 3$.

b) Compute the mean square training error of ordinary linear regression.

We need to compute the least squares solution to the three points $\{(1,1),(2,2),(3,1)\}$. One way of doing it would be to set up the general equation of a line in 2D with two parameters $y = ax + b$ (a, b are the parameters) and then write the loss function explicitly based on the *mean square error* which is something like (up to constant multipliers) $L(a,b) = 1/2\left\{(1-(a+b))^2 + (2-(2a+b))^2 + (1-(3a+b))^2\right\}$. Then, you could compute the solution for a, b based on normal equations $\frac{\partial L}{\partial a} = 0$ and $\frac{\partial L}{\partial b} = 0$. This results in a 2x2 system of equations which can be easily solved.

A simpler way would be to observe that due to the symmetry of our dataset, the *least squares* solution should be flat and so $a = 0$. So the loss function is simplified to $L(b) = 1/2\left\{(1-b)^2 + (2-b)^2 + (1-b)^2\right\}$. We can directly find the *least squares* solution for b by deriving and equating to zero:

$$\frac{\partial L}{\partial b} = (1-b)(-1) + (2-b)(-1) + (1-b)(-1)$$

This is equal to zero when $3b - 4 = 0$ and so $b = 4/3$.

Actually, an even simpler way would be to realize that the optimum b for a flat (constant) line is given by the *average* of all the y_i values in our dataset, thus $b = (1+2+1)/3 = 4/3$.

This means that the *mean square* training error of the *least squares solution* is:

$$\frac{(1-4/3)^2 + (2-4/3)^2 + (1-4/3)^2}{3} = \frac{(-1/3)^2 + (2/3)^2 + (-1/3)^2}{3} = \frac{6/9}{3} = \frac{2}{9} \approx 0.2222$$

Question 4

Explain the effect of the following operations on bias/variance:

a) Regularizing weights in a logistic regression model

more regularization increases bias, decreases variance

b) k in k-NN model

larger values of k increase bias, decrease variance

c) pruning a decision tree

increases bias, decreases variance

d) increasing the number of hidden units in an MLP neural network

decreases bias, increases variance

Question 5

You are a data analyst and you have to work on a binary classification problem. Describe resampling methodologies suitable in the following scenarios, include in each case the steps you would take in order to produce a final model:

a) Data is very limited

With very little data, overfitting is a serious potential problem, so we should go with simpler models, perhaps a linear model. Regularization becomes important, so I would opt to do *leave one out cross-validation* to adjust regularization and any other parameter. With the winning configuration, I would re-train on the whole data, and would not reserve a test set in this case, and use cross-validation as generalization metric instead. This is not ideal, but in this case it seems ok in order to not waste useful data.

b) There is plenty of data and it is cheap to get new labelled data

In this scenario, we can use a single train/validation split and use the training set for all training, and use the validation set for hyper-parameter tuning and model selection. Once we have obtained a final model, we can request new test data in order to assess its generalization ability. If the test performance is not good enough, we can go back to the start, request more data, perhaps consider more complex models to be trained using the larger volume of data (using the same strategy). Once we have considered the more sophisticated models over the larger volume of data, we can assess the final model again over a fresh test set, and if not good enough, repeat. If we circle through these steps too many times, perhaps we can stop and question ourselves whether the prediction task is achievable with the given features, so that we may need to gather new ones or re-think and/or re-engineer new ones.

The previous scenario is in the case that computational time is not an issue. If it is, then we would have to use perhaps more efficient algorithms such as random forests that work on bootstrapped samples or something along those lines.

Another idea would be to use an online method, that adapts to new data as it becomes available.