Final Exam, Machine Learning (MIRI) June 4th, 2020

Question 1. In the lab for random forest, you saw the following call to build a random forest for classification (the task is to predict the type of email: spam/nonspam). Please explain what this code does and what its paramters mean. In the code, `learn` contains the indices of rows of the dataset that correspond to the training set.

```
model.rf <- randomForest(type ~ ., data=spam3[learn,],
    ntree=150, proximity=FALSE,
    sampsize=c(nonspam=800, spam=500), strata=spam3[learn,]$type)
```

Answer: The code builds a random forest model made of 150 trees with the training data. It uses all the variables in spam3 to predict the type (spam/nonspam). It does not compute the proximity (distances) among the rows. It stratifies the sampling of the data according to type. The size of each sample is specified to be 800 non-spam and 500 spam.

Question 2. Explain the difference between *training* error, *validation* error, *test* error, and *generalization* error.

Answer: Training error of a model is the error computed on the training set on which the model was trained on; it is used to find a good fit of the parameters however it is an overly optimistic estimate of generalization performance. Test error is the error of a model computed on data that the model has not seen during its training procedure; typically it is used as an estimate of the generalization error of the model. Validation error is similar to test error in the sense that it estimates predictive error (uses data not seen during training) however its used in the context of a resampling technique, often to do model selection or hyper-parameter tuning. Finally, generalization error (or "true" error) is the overall error that a model makes. It cannot be computed since in most cases we do not have access to all possible data/labels, and it has to be estimated using the test error.

Question 3. Explain what the Bayes error rate is and how it relates to the generalization error of any classifier.

Answer: Bayes error comes from noise in the data when generated. it is also called irreducible error and is the smallest possible generalization error in a classifier, because summed to variance and bias you get the total error.

Question 4. It is said that generative algorithms for supervised learning learn the joint distribution $p(x, y)$ where y is the target and x corresponds to a vector of explanatory variables, and discriminative algorithms learn $p(y|x)$. Please explain what this means.

Answer: Generative models try to model the distribution of the data (the generating mechanism). With $p(x|Y = y)$ and $P(y)$, the bayes rule is used to compute $P(Y = y|X)$ which is the most likely label to have generated that point. Discriminative models try to separate (discriminate) the data.

Discriminative models find boundaries on the data and use those to predict. So formally they learn $p(y|X = x)$.

Question 5. Please explain the difference between a parameter of a model and a hyper-parameter. You may use an example if you want.

Answer: Parameters are the variables which the model itself adjusts when it is trained so that it is able to predict the best it can. For example, this could be the weights of a neural network or the betas of a linear regression. Hyper-parameters are variables which the model does not adjust and expects them to be provided to it (which means they need to be tuned for it with other procedures like CV). Examples of this would be the C and gamma of an SVM or the size of a hidden layer of an MLP .

Question 6. Please explain why different runs of the routine `nnet` for training a multi-layer perceptron may give you different solutions.

Answer: The routine `nnet` uses an iterative method to train networks that depends on the initial setting of weights. This initial setting is set randomly, so with differnt runs of the routine we start at different locations and thus may end up at different local optima in each run, or not converge at all.

Question 7. Please explain the potential danger of not having any type of regularization in a modelling task and the danger of having too much of it.

Answer: Regularization penalizes complex models. So, if we penalize complexity too much (too much regularization) we may end up with an overly simple model that is unable to capture the particularities of our data (underfitting). On the other hand, if we do not penalize at all (no regularization) we may end up with an unnecessarily complex model that captures the noise in our data (overfitting).

Question 8. Please explain the relation between the bias/variance tradeoff and the k of the k-nearest neighbor algorithm.

Answer: A small k will only take into account the few closest nearest neighbors and therefore will exhibit high variance and small bias. In contrast, a large k will have less variance since it takes into account more neighbors but makes the model more complex exhibiting a larger bias.

Question 9. What is the main objective of the resampling techniques that we have seen during the course (e.g. *cross-validation*)?

Answer: The main objective of a resampling technique is to estimate predictive error (generalization error) from a fixed dataset as honestly as possible. These estimates are then typically used for model comparison, model selection, or hyper-parameter tuning.

Question 10. Can you think of a situation where the EM algorithm for clustering is preferable to k-means?

Answer: K-means assumes spherical clusters (diagonal covariance matrix). In that regard, the EM algorithm is more flexible since it allows for clusters with ellipsoid shapes (non-diagonal covariance

matrix). This flexibility comes at the cost of more computational complexity. In the cases where the assumptions of K-means do not hold true, EM might be better.

Question 11. What is the main purpose of the *backpropagation* algorithm in the context of neural networks?

Answer: The main objective of backpropagation is to compute efficiently the gradients of the error with respect of their weights. In neural networks this algorithm is of use because it allows to optimize, by an iterative process, the values estimated for the parameters/weights in the model.