MLEARN	*Wednesday, January 18, 2023*
Professors: Eduard Garcia, Josep Vidal, Veronica Vilaplana Duration of exam: 2h 30min Solve the exam in these sheets, do not deliver additional ones	

Exercise 1

For the following statements about units 1, 2 and 3, please mark '**T**' for true and '**F**' for false. Please note that each correct answer will increase the grade by the same amount that each incorrect answer will decrease it. However, the final grade for this exercise will not be negative (i.e. if the number of incorrect answers exceeds the number of correct answers, the grade for this exercise will be set to 0).

1	Reinforcement learning is a type of machine learning in which an agent learns to take decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The agent's goal is to learn a policy that maximizes its expected reward over time.	**V**
2	If you're tested positive for a disease (assume that both *sensitivity* and *specificity* of the test are 99%), you can be sure that the probability that you actually have the disease is higher than the probability that you don't have it (i.e., according to MAP rule, you're sick).	**F**
3	Precision, exactness and accuracy are, in general, synonyms; not for a data scientist, though.	**V**
4	Maximum a Posteriori (MAP) rule minimizes misclassification errors although it may not minimize all types of "costs".	**V**
5	Minimum Bayesian Risk (MBR) minimizes the conditional risk $C(\omega_j\|x)$ and it generates decisions boundaries which are always different of those obtained in MAP.	**F**
6	MAP can be expressed as a particular case of MBR.	**V**
7	When feature vectors X show Gaussian distribution, they can be represented as a hyper-ellipsoid, the main axis of which follows the direction of the eigenvectors of the covariance matrix C.	**V**
8	If C (covariance matrix of input vectors X) is a diagonal matrix, it means that all input features are strongly correlated.	**F**
9	When $C_i = C$ (i.e. covariance matrix is the same for all i classes), decision boundaries between classes are necessarily hyper-quadratic (e.g. hyperellipsoid, hyperboloid, etc.).	**F**
10	If two states are equiprobable ($Pr(\omega_1) = Pr(\omega_2)$) and input feature X is Gaussian, a MAP discriminant would just choose the state whose mean μ_i is closer to the input vector (considering Euclidean or Mahalanobis distance).	**V**
11	The lower the *discriminability* (D') among classes, the larger is the area under the ROC (Receiver Operating Curve).	**F**
12	When defining a threshold for X to distinguish between two classes with the aim to increase detection probability, it also implies reducing probability of false alarms.	**F**
13	Parameters θ computed using a Maximum Likelihood (ML) estimator and obtained through training will always be the same regardless of the partitioning of the data into training and test sets.	**F**
14	An estimator with low bias but high variance will, in general, perform well with training data but may show a bad performance with different test sets.	**V**

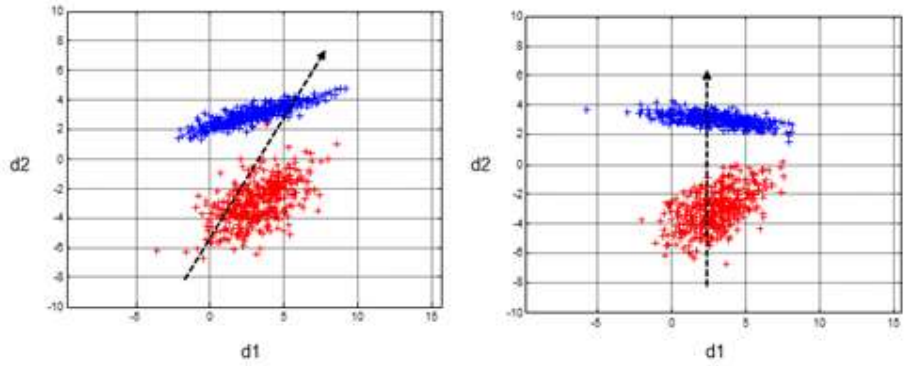
15	Having more features in our data samples (i.e. larger dimension of X vectors) increases the complexity of our algorithms and may entail a worse error probability in some cases.	**V**
16	Principal Component Analysis (PCA) helps reducing the input features by minimizing the error when trying to reconstruct the original vector and, therefore, it guarantees the best possible "discriminability" among classes.	**F**
17	The minimum mean squared error (MSE) trying to reconstruct the original X vector after a feature reduction with PCA depends on the "discarded" eigenvalues, and that's why we chose the largest ones for the PCA transformation.	**V**
18	Multiple Discriminant Analysis (MDA) seeks a transformation on input features such that inter-class distance (as per S_B) is maximized and intra-class dispersion (as per S_C) minimized.	**V**
19	A single transformation on an input vector x using MDA matrices provides a new feature vector z, the dimension of which (d') cannot exceed c - 1, where c is the number of different classes in the dataset.	**V**
20	PCA is a tool that could be used to reduce any number of features from input data.	**V**

Exercise 2

Regarding feature selection and feature reduction, briefly answer the following questions:

a) Give arguments in favor and against having very large input feature vectors (i.e. large number d of independent features).

b) Following are two scatter plots representing the training data of a classification problem with $c = 2$ and $d = 2$ (i.e. two classes and two features). Which one of them can be reduced with PCA while maintaining a good separability? Briefly justify your answer.



Dotted arrow represents the direction of the eigenvector associated with the largest eigenvalue of **S** (scatter matrix of input data).

c) Draw in the previous figures the direction that MDA would use to project the data points to reduce to $d'=1$. Would MDA produce a good separability in both cases?

d) The time of the day turns out to be a relevant data for your classification problem and, hence, your data samples include a field of the form hh:mm:ss. Briefly describe how you would transform this data into meaningful input features to feed your ML model.

e) Another relevant data for your problem is the day of the week (i.e. Monday, Tuesday, …, Sunday). How would you code the day of the week to feed an ML model?

Answers

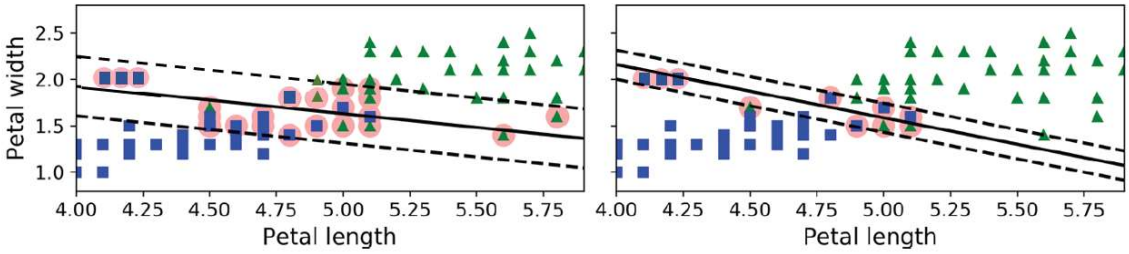
a) Ideally (i.e. for a large enough training set), increasing the number of features improves performance / considering too many features increase the complexity of classifier and makes training more difficult, more time and computer resources.
b) PCA reduces dimension to $d'=1$ by projecting data onto the depicted eigenvector. In the left figure, that results in overlapping classes (bad separability); in the right figure, classes are still separable in 1D.
c) MDA seeks to maximize separability in the reduced space. Left figure: better separability in direction "north-north-west"; right figure: meets the eigenvector.
d) It is important to preserve the periodicity in the representation, so that there is the same distance between the time stamps, let us say, 01:00:00 and 02:00:00 and 12:00:00 and 01:00:00. One possible solution: transform time of the day into number of seconds since 0:00h (or into hours, in decimal numbers); transform to angle (0-24h → 0-360º); one new feature is sin of that angle, and second new feature is cos of that angle.
e) Same comment applies here. One possible solution: one-hot vector (day of the week is coded as seven binary features, all zeros except the bit corresponding to the position of the coded day).

Exercise 3

Support vector machines

We are using a database to classify two types of flowers (iris virginica-squares class and iris versicolor-triangle class) according to the petal width and the petal length. We decide to use linear soft-margin SVM for non-separable classes, where the hyperparameter P [1] and slack variables ξ_i are used to allow for some data to lie on the wrong side of the corresponding margin boundary.

a) For two diferent values of P we obtain these decision boundaries and margins. Explain which one corresponds to the large and the small values of P.



b) The total number of samples is 30 for the squares class and 33 for the triangle class. Build the confusion matrix on each case and decide which is one performs better in training, in terms of the F-score. Can you guess what will be the relative performance on the test database?

c) Is a large value of P generaring an underfit or an overfit classifier? Explain why.

d) Briefly describe a method for selection of P.

Answers:

a) The right plot corresponds to the larger P case. If P is large, the number of vectors inside the margins is lower (we enforce to have smaller values for ξ) and the margins are narrower.

b) In the left case, the confusion matrix is:

Predict / True	Blue	Green
Blue	25	6
Green	4	29

For the right case, the confusion matrix is:

Predict / True	Blue	Green
Blue	27	3
Green	3	30

The right case exhibits a lower number of errors in train, so it might be slightly overfitting. The left case has the potential to perform better in the test database. This can be checked on the F-score as well. The F-score is defined in terms of the precision and the recall:

$$F = 2P \cdot R/(P+R)$$

F=0.8333 for the left plot and F=0.9 for the right plot (if blue class is taken as positive and green as negative classes).

[1] The hyperparameter P is used in the soft-margin SVM optimization:

$$\underset{\xi,\mathbf{w},w_o}{\text{minimize}} \ \tfrac{1}{2}\|\mathbf{w}\|^2 + P\sum_{i=1}^{N}\xi_i \quad \text{subject to} \begin{cases} y_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) \geq 1-\xi_i \\ \xi_i \geq 0 \end{cases} \quad i = 1,\ldots,N$$

c) As the value of P grows, we have more overfitting: more values ξ will be set to zero, hence lower possibility of finding misclassified training samples and lower probability of error.

d) Selection of P can be done using k-fold cross-validation:

1. Define a range of values of P, $\mathcal{P}$
2. Split the training dataset into K subsets.
3. For each value of $\mathcal{P}$, train the classifier with K-1 subsets and evaluate the performance with the validation K subset.
4. Shift the validation subset to another one, and repeat the step 3 up to K times.
5. Average the performance obtained from the K splits, for all values of $\mathcal{P}$ and select the best one.

Exercise 4

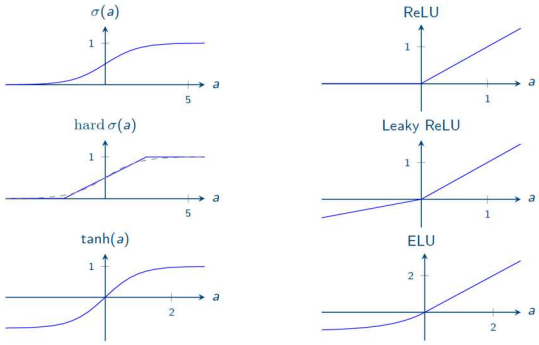
Neural networks

Briefly discuss the following aspects related to a multi-layer neural network (NN) used in a classification problem:

a) Plot three popular activation functions for the neurons.
b) How would you select one activation function?
c) Propose a method to reduce the risk of reaching local minima when learning based on backpropagation gradient descent.
d) How would you identify that the NN underfits the training data? How would you solve the situation?
e) Why regularisation in NN is needed? Briefly describe what is regularization.
f) How would you implement a maximum a posteriori (MAP) or a minimum Bayesian risk (MBR) classifier using a NN?

Answers:

a)



b) The activation function has to be non-linear and continuous. Among the possible ones, the selection has to be done based on validation.

c) Train with different sets of random initial values for the coefficients, and keep the parameters that yield the minimum cost function among them.

d) Underfitting is characterised by a large training error. One way to avoid it is by increasing the number of neurons and use some method to avoid overfitting.

e) Regularisation is needed to avoid overfitting due to a large number of parameters. It consists in adding a penalty function to the cost function $J(\mathbf{w})$ (be it MSE or cross-entropy). The cost function is somehow related to the parameters: in L2 regularization, the cost function is the Euclidean norm of $\mathbf{w}$, in L1 it is the sum of absolute values of $\mathbf{w}$.

f) When the training set is very large and with sufficient number of hidden units, the outputs of the NN are approximately the posterior probabilities $\Pr(\mathbf{w}_i|\mathbf{x})$. From them, one can apply:

$$\omega_{MAP} = \arg\max_{\omega_i} \Pr(\omega_i|\mathbf{x})$$

or

$$\omega_{MRB} = \arg\min_j C(\omega_j|\mathbf{x}) = \arg\min_j \sum_{i=1}^{c} \pi_{ji} \Pr(\omega_i|\mathbf{x})$$

as long as we have costs π associated to each pair of classes.

Exercise 5

We want to solve a machine learning problem using a model *M*. We split the dataset into training, validation and test splits using 60%, 20% and 20% of data, respectively.

We train the model *M* with different settings (but using always the same training, validation and test splits) and we get these results:

	Experiment 1	Experiment 2	Experiment 3
Human level error	1%	1%	1%
Training error	19%	1.1%	1,1%
Validation error	20%	20%	2%
Test error	21%	21%	21%

a) For each experiment, understand the results, describe the problem and propose possible solutions.

b) For each experiment, propose specific solutions for the following models (whenever possible):

1. KNN
2. MLP
3. SVM
4. Decision Tree

Answer:

a)

Experiment1: high training error (and validation and test error): underfitting (high bias)

The model is too simple. Use a more complex model (a different family, a model with more parameters)

Experiment2: low training error but high validation error: overfitting (high variance)

Use a less complex model (a different family with less parameters or same family but decrease the number of parameters in the model), add regularisation, if possible, get more training data

Experiment3: low training and validation error but high test error: overfitting the validation set

Try to get more data that is similar to the test set.

b)

Experiment1:

KNN decrease the value of k

SVM: adjust the value of the regularisation parameter P (very low values tend to produce an underfitted model), use a non-linear SVM (kernel) if you are using a linear SVM, increase the value of sigma if you are using a RBF kernel.

MLP: use a more complex model (more units in the hidden layer, more hidden layers)

Decision Tree: this may happen if the tree is shallow, try to increase the depth of the tree

Experiment2:

KNN: increase the value of k

SVM: decrease the value of the regularisation parameter P, if using kernel-SVM reduce the degree of the polynomial, decrease the value of the sigma in the RBF kernel.

MLP: use a model with less parameters (less hidden layers, less units per layer), add L1 or L2 regularisation, add dropout, use early stopping, use data augmentation.

Decision Trees: pre-prune (control tree growing with hyperparameters like max depth, minimum samples to split an internal node, minimum number of samples to be a leaf node), post-prune, create an ensemble of trees

Experiment3: this is a problem with the data used to train and validate the model

Exercise 6

Ensembles

a) Two popular methods for creating ensembles of classifiers are bagging and boosting. Explain the main similarities and differences between these techniques. Include in your answer a discussion in terms of bias and variance.

b) Some ensemble models use Decision Trees as base predictors. Mention at least two of such models and explain the main differences between them.

Answers:

a)

Bagging

- Obtain different models by training the same model on different training samples
  - Reduce overfitting by averaging out individual predictions (variance reduction)
- In practice: take N bootstrap samples of data, train a model on each bootstrap
  - Higher N: more models, more smoothing (but slower training and prediction)
- Base models should be unstable: different training samples yield different models
  - Eg. deep decision trees or randomized decision trees
- Prediction by averaging predictions of base models
  - Soft voting for classification (possible weighted)
- Can produce uncertainty estimates as well
  - By combining class probabilities of individual models

Boosting

- Combine several weak learners into a strong learner: train predictors sequentially, each trying to correct the previous one.
- Boosting is a bias reduction technique
- Many boosting methods available, the most popular is Adaboost

Adaboost:

- Obtains different models by reweighting the training data every iteration
  - Reduces overfitting by focusing on the 'hard' training examples
  - Increases weights of instances misclassified by the ensemble and viceversa
- Base model should be simple so that different instance weights lead to different models
  - Underfitting models: decision stumps (or very shallow trees)
  - Each is an 'expert' on some parts of the data
- Additive model: predictions at iteration i are sum of base model predictions
- Effect on bias and variance
  - Adaboost reduces bias (and a little variance)
  - Boosting too much will eventually increase variance

b)

- **Bagging of Decision Trees:** the method described in (a), using decision trees as base model. Each tree is trained using a sample drawn with replacement from the training set
- **Random Forest:** similar to Bagging of DT, but uses a random subset of features to train each tree
- **Extremely randomized trees:** considers only one random threshold for random set of features (faster to train)
- **Gradient Boosting:** It is a boosting algorithm that works by sequentially adding a decision tree to the ensemble, trying to fit the new predictor to the residual errors made by the previous predictor (instead of using weights on samples like Adaboost)
- **XGBoost:** a particular implementation of Gradient Boosting (improved and optimized)