Website link: <https://intro2ml.pages.doc.ic.ac.uk/autumn2021/>

Evaluation methodology:

**Metrics**

* Text, letter

  Description automatically generated
* Must shuffle data actually [random module/shuffle() for in place True and sample() to create new list]
* For decision tree [max\_depth] must be specified. Hyperparameters: max\_depth, minimum samples required to split an internal node, min\_samples required to be at each new leaf of a node. In this case all the features will be considered [only 7]. Don’t evaluate them based on the accuracy of the training set. Split [training/validation/test] 🡪 this is the required split
* Cross validation: 10 folds [parts] and select 1 for test and train on everything else. Rotate to get test, validation and training splits. Each cross-val step [internally] for optimal entropy loss of 1 fold [lots of computation🡪internally]
* Timeline

  Description automatically generated with medium confidence
* A picture containing text

  Description automatically generated
* A picture containing logo

  Description automatically generated
* Text

  Description automatically generated with low confidence
* Use macro averaged recall
* A picture containing table

  Description automatically generated
* Macro vs micro averaging
* Multi class [one output label among many possible output classes] vs multi label [many output labels]
* Balanced vs imbalanced dataset
* For imbalanced dataset: accuracy follows majority class. Precision goes down for minority class
* If one class completely misclassified 🡪 precision and recall are zero and F1 score is undefined
* Could normalise

**Overfitting and confidence intervals:**

* Mainly for neural networks [too much training time or high model complexity]. Solutions: more simple and use more data
* Text, letter

  Description automatically generated
* Text

  Description automatically generated with medium confidence
* Statistical tests; prefer randomisation test
* Text

  Description automatically generated with medium confidence
* Statistically significant
* P-hacking:

**Neural networks:**

**Table

Description automatically generated**

* Optimised usually by gradient descent
* Linear regression:
  + Gradient descent to optimise a and b
  + Multi-layer NNs.
  + Activation function can be treated as a hyperparameter
  + Activation function for output [sigmoid for multiclass classification]
* Backpropagation and gradient descent:
  + GPUs on google colab and Kaggle
  + A picture containing text, clock, watch, gauge

    Description automatically generatedfor linear regression
  + Classification [binary, multi-class or multi-label]
  + Text

    Description automatically generated
  + Text

    Description automatically generated
  + Table

    Description automatically generated
  + Text, whiteboard

    Description automatically generatedsoftmax
  + Gradient descent: Graphical user interface, text

    Description automatically generated with medium confidence
  + A picture containing graphical user interface

    Description automatically generatedstochastic gradient descent
  + Timeline

    Description automatically generatedmini batched gradient descent
  + Adaptive learning rates [for each parameter] based on extent to which a parameter was updated. ie use Adam or AdaDelta
  + Or apply learning rate decay Text

    Description automatically generated
  + Weight initialisation [from normal distribution with 0 mean and variance of 1]
  + Data normalisation: standardisation or min-max normalisation
  + Diagram

    Description automatically generated Diagram

    Description automatically generated
  + Overfitting in NNs: Can apply early stopping using accuracy vs epochs chart [get optimal pt]. Always store best model when improving on previous performance
  + Regularisation: Penalising model parameters. L1 (encourages sparsity) vs L2 regularisation. Can combine both as well.
  + Dropout: randomly setting some neural activations to zero
  + Coursework: implement a small NN library [1] then use the library to build and train models to predict house prices [2].

**Evolutionary algorithms (Robotics and games):**

* Have no access to gradient as with grad descent eg (RL problem) 🡪 motor commands in robot learning (link between movement and objective function unknown (no mathematical relationship).
* **Initialise population of solutions randomly** (each soln is a genotype). Each genotype expressed to its phenotype. Use all combinations and rank against one another to determine best one (via **fitness function** (blackbox function). Need operators (**selection**, **cross over** (combinations of selected) and **mutation** (combination of parents + small variation)). Note: Want to maximise fitness functions
* Text

  Description automatically generated with medium confidence
* A picture containing text, close, gauge

  Description automatically generatedSample fitness function for mastermind game
* **Quality-diversity optimisation** 🡪 High diversity means large span for descriptor space and high performance means high quality. NOTE: Dimensions = DoF \* parameters
* **MAP-Elites:** grid discretisation (6-D: each in 5 values therefore 56)
* Diagram

  Description automatically generated
* Evolutionary algorithms 🡪 Deep reinforcement learning (for DNNs hyperparameters and weights)

**Exam preparation:**

* Evaluation sheet:
  + Train classifier for binary classification question:
    - Optimise model parameters [optimised automatically based on model training] on training set and evaluate hyperparameter [manually choose] options on validation set.
    - Then evaluate on test set once best model architecture has been chosen
  + Case of having low number of data samples:
    - Cross validation: shuffle and split dataset into N different parts. Rotate which fold is used for testing until all N folds have been used (rotation between test and training + validation parts in each fold)
    - Then average performance across all folds: Overall performance on dataset. In this case did not train and test on the same examples in the same experiment
    - For hyperparameter tuning: At each iteration of the folds use part of training and validation for hyperparameter tuning. Might also perform internal cross validation on training + validation data at each iteration
  + Evaluation metric importance based on task:
    - MSE or RMSE: Regression for predicting amount of rain
    - F1 score: classification [for error class]
    - Accuracy: [identify type of land in aerial photo]
  + If model gets good performance on training but bad on validation:
    - Try regularisation, early stopping, dropout [usually for NNs], getting more training data [all models] or reducing model complexity
  + If bad performance on both training and validation:
    - Reduce regularisation, train for longer and or increase model capacity in case it is overfitting
  + If performs well on validation set:
    - Check if the data has been properly split and maybe examples in validation/test split are still present in training data
  + Note on example NN for images:
    - He used the test data set for number of data points in confidence interval estimation
  + Statistically significant:
    - Use 95% confidence interval. Less than 5 % chance that this performance difference is due to sampling noise and the systems are actually comparable