Workshop 5: Training models

The aim of this week's workshop is to understand and evaluate the training process of models, in particular stochastic gradient descent, learning curves and convergence.

You will use **Wisconsin Breast Cancer data set (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer)**, which is included in scikit-learn, and we have provided a **starting notebook for this workshop (https://myuni.adelaide.edu.au/courses/71744/files/10644323?wrap=1)** ↓ (https://myuni.adelaide.edu.au/courses/71744/files/10644323/download?download_frd=1) .

1. The notebook reads in the data, and then has some incomplete code to split the data into train/validation/test splits.
   - Fill in the necessary bits so that the ratio for train:validation:test is 60:20:20 (think about how to do this in the two steps provided).
   - After this, a pre-processing pipeline is built and then a function called *sgdfn* is defined. This function creates an SGD classifier and calls it, one epoch at a time, from within a loop using the *partial_fit* method. The SGD classifier needs to use the *warm_start=True* option in order to enable the *partial_fit* method to work in this way. *(Note that an epoch is one pass through all the training data, whereas an iteration is one pass through a batch of training data. A batch is a smaller subset of the training data, although in our case they are the same as we are not splitting this small dataset into smaller batches, since we don't need to.)*
   - Before this loop is executed it is necessary to fit the pre-processing to the training set and then transform the training and validation datasets. You will need to fill in the appropriate calls to do this.
   - Within the loop the predict function is called on the validation data and then accuracy and loss (hinge_loss) are calculated. Fill in the necessary code for these steps.
   - After the loop the accuracy and loss are plotted, and these are the learning curves.
   - Call the function with *nsamp* set to the maximum value for your training set and *learnrate* set to 1e-5.
2. Explore the effect of different learning rates, trying to find the point where the learning rate is too low for it to learn and also the rate where it is too high and becomes unstable. Hint: you will need to explore a large range of values over many orders of magnitude.
3. Explore the effect that different training set sizes (via *nsamp*) has on the learning curves. How does the training set size effect the limits of the learning rate - that is, the values when the learning rate becomes too low or too high? Keep a record of the values for several training set sizes and summarize your findings as a plot or a table.

4. Using nsamp=100, find what you consider is the best learning rate. Then compare this to what happens when you set *learning_rate='optimal'* instead. Do this by taking another argument in your function. Show the learning curves in each case. Can you do better than the 'optimal' setting?
5. Regularisation:
   - Instead of using the function above, create your own pipeline including an SGDClassifier and use this to fit to training data and predict validation data. Calculate the performance metrics (accuracy and loss) for this, but as it is a single call you cannot get a learning curve from this. Record your results as these will be a baseline for the next experiments.
   - The SGDClassifier incorporates L2 regularisation by default. It can be controlled by the parameter *alpha*. Try varying this over many orders to magnitude until you find a value that causes a noticeable change in the results.
   - Make a plot of the accuracy versus alpha value.
   - The type of regularisation used is controlled by the *penalty* parameter. Look at the available options and try the others, making a similar plot for each of them.
   - **Question:** What do you think is the best setting for regularisation?
6. Model Comparison:
   - We complete the standard workflow as usual - choose what you think are the best settings for all of the above parameters.
   - **Question:** Which of the above parameters (i.e. learning rate, type of learning method or regularisation) has the biggest effect on final performance?
   - Measure the performance of this method on the current validation set.
   - Re-train the method on the combined set of training and validation data.
   - Measure the performance of this on the test set.
   - **Question:** How does this performance compare? What did you expect?

Optional Extension

- Early Stopping:
  - Take the pipeline from the start of step 5 and fit it only using 100 training samples before predicting the validation labels. Check that you get an equivalently good results on both training and validation sets.
  - The attribute *n_iter_* in the SGDClassifier can be used to find out how many iterations it performed, so print this out. You should find that it is a lot less than 1000, even though the early stopping parameter is set to False by default. Look at the documentation for the parameters *tol, early_stopping* and *n_iter_no_change* to understand what it is doing.
  - We will now explore the early stopping option, so set up a new SGDClassifier with *early_stopping* set to True and the defaults for *tol* and *n_iter_no_change*. Run this on the cut-down training of 100 samples and see how many iterations it performs.
  - See what happens when you increase or decrease the value of *n_iter_no_change*.

- Also try using *verbose=1* and see what information it shows.
- **Question:** Do you think the early stopping affects the performance?