Machine Learning 441 Assignment 3: Active Learning in Neural Networks

KT Mössner
26024284
26024284@sun.ac.za

***Abstract*—**

I. Introduction

Active learning is a supervised machine learning approach in which the learning algorithm has a varied degree of control over the input data that the neural network (NN) is trained on. **TODO: Add github link**

II. Background

A. Active Learning

Active learning is the machine learning technique where the algorithm selects the data that it wants to learn from. This is a dynamic training process where the learning algorithm selects data from a candidate training set. The algorithm selects the most informative instances from the pool of training data that are most likely to provide the greatest improvement in model performance. Active learning has the proposed benefit of reducing training times and better generalisation performance. The algorithm selects the training instances to train the NN on, which leads to a potential reduction in the number of observations required to train the model. Improved generalisation is achieved when the selected data contains enough informative observations to correctly model the problem.

Passive learning follows the approach where models are trained once only on randomly sampled labelled data. Active learning, however, is an iterative process where at each step the model is trained on a labelled subset of the available data, then the most informative data from the unlabelled data pool is selected and the model is retrained on the new dataset containing the additional informative observations. This iterative process is repeated until the models performance stabilises, the dataset becomes exhausted or a specified number of iterations are completed.

The two main approaches to active learning, as outlined in [4], are: *incremental learning* and *selective learning*. Incremental learning begins training on a subset of the training set and during training selects more subsets at specific intervals to add to the training set. The training set that the model is trained on continues to grow as more subsets are selected from the candidate sets. Once a subset has been selected it is removed from the candidate set. During incremental learning the size of the training set grows and the size of the candidate set shrinks after each iteration of the algorithm. At each selection interval of the selective learning process a new training subset is selected from the original candidate set. In contrast to incremental learning, selective learning does not remove the selected subsets from the candidate set. This allows for all observations to be selected at each selection interval. The currently selected subset is used to train the model until a convergence criteria met, such as the training loss stagnating. Once this criteria is met a new training subset is selected from the candidate set and the model continues to train on this new training dataset. This process is repeated until the model has converged or another stopping criterion has been met.

B. Uncertainty Sampling

Uncertainty sampling selects instances for which the current NN is most uncertain how to label [1]. The instances that the NN is unsure about are the ones that lie close to or on the decision boundary for classification models and instances that have high prediction errors for regression models. The uncertainty for classification models can be measured in several methods. The first approach is to use the conditional entropy:

$$x^* = \arg\max_{x^{(i)}\in\mathcal{U}} -\Sigma_{y\in Y} P_\theta(y|x^{(i)})\log(P_\theta(y|x^{(i)}))$$

where $P_\theta(y|x^{(i)})$ is the probability that instance $x^{(i)}$ has label y. This approach selects the samples that have the highest entropy. Another approach is to use the least confidence uncertainty:

$$x^* = \arg\max_{x^{(i)}\in\mathcal{U}} (1 - \max_{y\in Y} P_\theta(y|x^{(i)}))$$

This approach selects samples where the highest class probability is the lowest. The last approach is the margin of confidence sampling:

$$x^* = \arg\min_{x^{(i)}\in\mathcal{U}} (P_\theta(y_m|x^{(i)}) - P_\theta(y_n|x^{(i)}))$$

where, y_m is the most likely label and y_n is the next most likely label for the instance $x^{(i)}$. This last approach measures how uncertain the model is to classify an instance by measuring the margin between the two most likely classes. The smaller the margin is the more uncertain the model is.

Uncertainty sampling for regression is an active learning strategy that selects instances from data that a NN is most uncertain how to predict. Predictive uncertainty for a NN is estimated by using an ensemble of neural networks (Lakshminarayanan et al., 2017). A series of NN regression models are trained with different random parameter initialisations. Each NN generates a prediction for all instances of the unlabelled datasets. The predictions variance across the ensemble of neural networks is used as the measure of regression uncertainty. The instances with the highest predictive variance are chosen for labelling and inclusion into the training set.

Combining uncertainty measures for classification and regression with a pool-based greedy active learning algorithm provides a framework for uncertainty sampling for neural networks. The pool-based greedy active learning algorithm iteratively selects the most uncertain instances and incorporates them into the training set. This process is repeated until some specified stopping criterion is met. This criterion is when a given budget of iterations is exceeded or the NN converges to an optimal solution. The general pool-based active learning algorithm is provided in Algorithm 1 which comes from [3].

Algorithm 1 Pool-Based Active Learning

```
1:  Input: $\mathcal{U}$ – unlabeled data, $\mathcal{L}$ – labeled data, $\theta$ – NN
    model, $B$ – budget
2:  repeat
3:      for all $\langle x^{(i)}, ?\rangle \in \mathcal{U}$ do
4:          compute $uncertainty(x^{(i)}, \theta)$
5:      end for
6:      pick highest uncertainty $x^*$ and query its label $y^*$
7:      $\mathcal{L} \leftarrow \mathcal{L} \cup \{\langle x^*, y^*\rangle\}$
8:      $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\langle x^*, y^*\rangle\}$
9:      Train $\theta$ on $L$
10:     $B \leftarrow B - 1$
11: until $B = 0$
```

Where $\langle x^{(i)}, ?\rangle \in \mathcal{U}$ represents the i-th unlabelled instance from the unlabelled pool $\mathcal{U}$. The algorithm repeatedly samples the most uncertain instances from the unlabelled pool, updates the training set with the uncertain data, removes the instance from the unlabelled pool and retrains the NN on the new training set.

C. Sensitivity Analysis

Sensitivity analysis uses a selective learning approach to continuously select subsets from the original training dataset during training. At each iteration of the selection algorithm, the NN selects the most informative patterns from the candidate training set using the knowledge it has already learned in the training process. One of the key aspects of a select learning algorithm is its ability to decide on what information should be used for training and what information can be overlooked. During the training process, a classification model might become uncertain about a previously correctly classified instance. In this case the selective learning algorithm must be able to identify the instance to add it back to the training subset. The most informative patterns for classification problems are often found near the decision boundaries, referred to as the region of uncertainty.

Sensitivity Analysis Selective Learning Algorithm (SASLA) as proposed by (Engelbrecht, 2001), uses sensitivity analysis to select patterns in the region of a decision boundary. The proximity of patters to decision boundaries are evaluated using first order derivatives of the output units with respect to input units. The patterns that are closest to the decision boundary are determined to be the most informative and are chosen for training. Patterns that have little to no effect on the outputs of a NN are uninformative and patterns that have a strong influence on NN outputs are informative. Pattern informativeness is defined as the sensitivity of the output vector from a NN to small changes in the input vector.

$$\Phi^{(p)} \doteq \|\vec{S}_o^{(p)}\|$$

denotes the informativeness of pattern p, where $\vec{S}_o^{(p)}$ is the output sensitivity vector for pattern p and $\|\cdot\|$ is any suitable norm. The study from [4] suggests the maximum norm be used to compute the pattern informativeness.

$$\Phi_\infty^{(p)} = \|\vec{S}_o^{(p)}\|_\infty = \max_{k=1,\ldots,K}\{|S_{o,k}^{(p)}|\}$$

where $S_{o,k}^{(p)}$ refers to the output sensitivity of a single output unit o_k to changes in the input vector $\vec{z}$.

The output sensitivity vector $\vec{S}_o^{(p)}$ is defined as the norm of the output-input layer sensitivity matrix $S_{oz}^{(p)}$. The norm used to calculate each element k of the output sensitivity vector is the sum-norm.

$$S_{o,k}^{(p)} = \|S_{oz}^{(p)}\|_1 = \Sigma_{i=1}^{I}|S_{oz,ki}^{(p)}|$$

Where I is the total number of input units including the bias unit to the hidden layer. Assuming that sigmoid activation functions are used in the hidden and output layers, each element $S_{oz,ki}^{(p)}$ of the sensitivity matrix is computed using

$$S_{oz,ki}^{(p)} = (1 - o_k^{(p)})o_k^{(p)}\Sigma_{j=1}^{J}w_{kj}(1 - y_j^{(p)})v_{ji}$$

where w_{kj} is the weight between output unit o_k and hidden unit y_j, v_{ij} is the weight between hidden unit y_j and input unit z_i, $o_k^{(p)}$ is the activation value of the output o_k, $y_j^{(p)}$ is the activation of hidden unit y_j, and J is the total number of hidden units including a bias unit to the output layer. A pattern is considered informative if one or more of the output units are sensitive to small perturbations in the input vector. The larger the value of $\Phi_\infty^{(p)}$ the more informative the pattern is.

The SASLA operator $\mathcal{A}$ is defined as

$$\mathcal{A}(D_C, \mathcal{F}(D_T; W)) = \{p \in D_C | \Phi_\infty^{(p)} > \Psi(\vec{\Phi}_\infty)\}$$

where D_C is the candidate training set, W represents the weights of the NN, $\mathcal{F}(D_T; W)$ is the function approximation of the NN which transforms observations from the I-dimensional input space to the K-dimensional output space, using the training patterns in D_T. The vector of $\vec{\Phi}_\infty$ is defined as

$$\vec{\Phi}_\infty = (\Phi_\infty^{(1)}, ..., \Phi_\infty^{(p)}, ..., \Phi_\infty^{(P_C)})$$

The function Ψ implements the rule used to select patterns and is defined as

$$\Psi(\vec{\Phi}_\infty) = (1 - \beta)\bar{\Phi}_\infty$$

where β is the subset selection constant and $\bar{\Phi}_\infty$ is the average pattern informativeness,

$$\bar{\Phi}_\infty = \frac{\Sigma_{p=1}^{P_C} \Phi_\infty^{(p)}}{P_C}$$

where P_C is the total number of patterns in the candidate training set D_C. Patterns are determined to be informative if their informativeness is some factor larger than the average informativeness over all patterns. The subset selection constant $\beta \in [0, 1]$ is used to control the area around the decision boundary which informative patterns may come from. A larger subset selection constant will allow for more informative patterns to be selected. A conservative choice for the subset selection constant is $\beta = 0.9$

The new subset of training data $D_{S_s} = \mathcal{A}(D_C, \mathcal{F}_{NN}(D_T; W))$ is selected at each selection interval τ_s. Set the training set $D_T = D_{S_s}$ for the next training interval.

The general SASLA algorithm is outlined below.

1) Initialise weights and learning parameters.
   Initialise the pattern selection constant, $\beta = 0.9$ as a conservative choice.
   Construct the initial training subset $D_{S_0} = D_C$.
   Let the training set $D_T = D_{S_0}$.

2) Repeat until convergence
   a) Repeat
      Train the NN on the training subset D_T until some termination criterion is met.
   b) Compute the new training subset D_{S_s} for the next subset selection interval τ_s:
      i) For each $p \in D_C$, compute the sensitivity matrix $S_{oz,ki}^{(p)}$ using the equation for sigmoid activation functions.
      ii) Compute the output sensitivity vector $\vec{S}_o^{(p)}$ for each $p \in D_C$.
      iii) Compute the informativeness $\Phi^{(p)}$ of each pattern $p \in D_C$
      iv) Compute the average pattern informativeness $\bar{\Phi}_\infty$.
      v) Apply the operator $\mathcal{A}(D_C, \mathcal{F}_{NN}(D_T; W))$ to find the subset D_{S_s} of most informative patterns. Then, let $D_T = D_{S_s}$.

The convergence criterion that could be considered are: maximum number of epochs reached; mean squared error (MSE) is lower than a specified threshold or when the percentage of correctly classified observations reaches a specified threshold.

III. Implementation

A. Neural Network

The neural network for the classification and regression problems was implemented in Python using the PyTorch library [11]. The architecture of the NN consists of the input layer, one hidden layer and one output layer. The sigmoid activation function was applied to the hidden and output layers in the NN. The number of units chosen for the hidden were obtained by an overestimation of the optimal number of units. The number of units in the output layer is dependent on the problem the NN is solving. A NN for regression problems will consist of a single unit in the output layer whereas for classification problems with one-hot encoded labels, the number of units in the output layer corresponds to the number of classes in the classification problem. Applying the sigmoid activation function to each node in the output layer of classification problems produces a probability for an input belonging to the class corresponding to the output unit. The class label corresponding to the output unit with the highest probability is assigned to the input instance. The loss function and optimisation algorithm used in the NN architecture is the MSE loss and stochastic gradient descent (SGD) optimisation algorithm. For classification with one-hot encoded labels, the MSE loss is computed for each instance as the average of sum of the squared differences between the network outputs and the corresponding elements of the one-hot vector of the true label. The parameters of the SGD algorithm include the learning rate which controls the step size of weight updates, the momentum term which smooths oscillations in the loss function landscape and accelerates convergence, and the weight decay regularisation parameter. Weight decay is a regularisation technique that prevents overfitting by adding a penalty term to the loss function based on the sum of the squared weights of the model. The loss function with weight decay regularisation takes the general form of:

$$Loss(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2$$

Where $\boldsymbol{\theta}$ is the vector of the model's weights, $\lambda \|\boldsymbol{\theta}\|_2^2 = \Sigma_i \boldsymbol{\theta}_i^2$ and $\lambda > 0$ is the weight decay parameter which determines the regularisation strength. Weight decay regularisation penalizes large weights and pushes them towards zero. This manipulates the optimisation algorithm to prefer smaller weights that are more evenly distributed. The weight decay parameter λ is problem dependent and requires cross validated hyper parameter tuning to determine the optimal value. The NN was trained on sequential batches of the training data for each epoch or iteration of the training algorithm. Each batch was randomly selected from the full set of training data. The validation set was used to calculate the validation loss for each epoch.

The implementations for the Passive Learning, Uncertainty Sampling and SASLA learning algorithms are described in the following subsections.

B. Passive Learning

The training algorithm for the Passive Learning approach involves training the NN on the entire training dataset for a fixed set of epochs. At the beginning of each epoch the training data is randomly shuffled and the model is updated on sequential batches from the training data. For each batch, the loss is computed, backpropogation is performed and the NN model weights are updated using the SGD optimisation algorithm. Once the training epochs have been exhausted the final trained NN model is returned along with the models training history for the training and validation losses for each epoch.

A pipeline for the Passive Learning approach was implemented to train the NN for each problem type. The parameters for the pipeline function consists of the NN hyper parameters and two tuples for the training, validation and test splits for the predictor and response variables. Each dataset is unpacked and then converted into tensor objects in preparation for the NN training algorithm. The input and output dimensions for the NN are determined by the shapes of the training data and labels. Two objects for the MSE loss and SGD optimisation algorithms are initialised and passed as parameters to the NN training algorithm. The NN is trained using the training algorithm and the performance of the trained model is evaluated on the test set. The trained NN, the history and the test performance results are returned by the pipeline function.

C. Uncertainty Sampling

The Uncertainty Sampling approach uses the same training algorithm as the Passive Learning approach, by training numerous times on the expanding training set of uncertain instances. For each iteration of the Uncertainty Sampling algorithm a NN is trained on the entire training set. This trained model is then used to determine the instances in the unlabelled candidate pool with the highest classification or prediction uncertainty. The labels for these uncertain instances are then obtained and are added to the labelled training pool for a NN to be trained on in the next iteration of the algorithm. This process is repeated until the budget for the number of iterations of the algorithm is exceeded.

The conditional entropy uncertainty measure was implemented for instances from the classification problems. An ensemble of NN models was used to calculate the predictive uncertainty for instances from the regression problems. The instances that have the highest predictive ensemble variance are the most uncertain instances. A budget for the Uncertainty Sampling algorithm was set to 25. The number of instances to sample for each iteration of the algorithm was determined by limiting the maximum number of samples to consider to 80% of the samples from the training set and dividing this number by 25. The budget and sampling strategy was chosen to prevent the Uncertainty Sampling algorithm from training on the entire training dataset. The initial NN is trained on a randomly sampled subset from the original training set, which becomes the active training set that uncertain instances are added to. Once the budget has been exceeded the training stops and the performance of the final trained model is evaluated on the test set.

D. SASLA

The SASLA approach differs to both the Passive Learning and Uncertainty Sampling approaches by adaptively refining the training subset at each epoch by selecting the instances with the highest informativeness from the candidate training set. As the NN trains it learns which instances have the highest sensitivity in the model outputs with respect to the inputs and learns that these instances are more informative. The SASLA algorithm identifies the instances in the dataset that are close to decision boundaries for classification problems and instances that have substantial output variation for regression problems.

In the beginning of the SASLA algorithm, the entire candidate set of training instances is used as the initial training subset. After each epoch of training, sensitivity analysis on the trained model parameters is conducted to determine the instances with the highest informativeness in the candidate training pool. The informativeness is computed by propagating the instance through the network and measuring the magnitude of the gradient of the outputs with respect to the inputs. This captures how strongly small perturbations in the input influence the network outputs. A threshold, determined by the selection constant β, is applied to select the most informative subset. Instances that have an informativeness greater than $(1-\beta)$ times the average informativeness of the entire candidate set are kept for training in the next epoch. A conservative choice of $\beta = 0.9$ was chosen for the selection constant. The SASLA algorithm terminates when either the MSE loss falls below a specified threshold or the maximum number of epochs have been exhausted. The performance of the final trained model is then evaluated on the test set.

IV. Empirical Process

A. Data Description and Pre-processing

A collection of three classification and three function approximation datasets of varying complexity were collected to evaluate the performance of passive and active learning algorithms. The datasets that were considered for the classification problems are the Breast Cancer Wisconsin Diagnostic dataset [5], the Mobile Price Classification dataset [6] and the Letter Recognition dataset [7]. The datasets considered for the function approximation problems were the Boston Housing Prices dataset [8], the Concrete Compressive Strength dataset [9] and the Abalone Age dataset [10]. Each dataset has a varying degree of complexity, observations, features and data quality issues. The datasets and the pre-processing details are described below. After applying the necessary data pre-processing techniques to each dataset, they were split into 70% training, 10% validation and 20% test sets. The split for each classification class was stratified to preserve equal class distribution in the training, validation and test sets. After splitting the data, the numerical input features were scaled to the range $[-1, 1]$ The target feature for the regression problems were scaled to the range $(0, 1)$, which is the output range of the sigmoid activation function. The target features for the classification problems were one-hot encoded into vectors with a one in the i-th index indicating that the observation belongs to the i-th class. One-hot encoding the target class features allows for sigmoid activation functions on the output nodes and the MSE loss function to be used in training.

1) Breast Cancer Wisconsin: The Breast Cancer Wisconsin dataset consists of 569 instances with 30 numeric features and one categorical target variable. The categorical target variable contains two classes, M for malignant and B for benign tumours. The categorical target variable was first binary encoded, with 1 representing M and 0 representing B, followed by the one-hot encoding transformation. The class distribution for the target variable is 357 instances belonging to class 0 and 212 instances belonging to class 1. The dataset did not consist of any missing values.

2) Mobile Price Classification: The Mobile Price Classification dataset consists of 3000 instances with 20 numeric features and one categorical target variable. The categorical target variable contains four classes indicating the price range for mobile phones. The classes for the price ranges consist of 0 (low cost), 1 (medium cost), 2 (high cost) and 3 (very high cost). The target variables was transformed into a one-hot encoded vector representation. The class distributions for the target variable is 993 instances belonging to class 0, 1,007 instances belonging to class 1, 500 instances belonging to class 2 and 500 instances belong to class 3. The class imbalances were retained to increase the complexity of the classification problem. The dataset did not consist of any missing values.

3) Letter Recognition: The Letter Recognition dataset consists of 20,000 instances with 16 numeric features and one categorical target variable. The categorical target variable consists of 26 classes, each representing a letter of the alphabet. The target variable was encoded into numeric class labels before transforming the target into a one-hot encoded vector representation. The class distributions for the target variable ranges from 734 to 813 instances per class. The dataset did not consist of any missing values.

4) Boston Housing Prices: The Boston Housing Prices dataset consists of 506 instances with 13 numeric features one regression target variable. The target variable contains the median housing price value in the are of Boston. The dataset contained 3.95% missing values across certain features. These missing values were imputed with the average values for the features. The target variable was scaled to the range $(0, 1)$, to match the output range of the sigmoid activation function.

5) Concrete Compressive Strength: The Concrete Compressive Strength dataset consists of 1,030 instances with eight numeric features and one regression target variable. The target variable contains the compressive strength of concrete, measured in megapascals. The target variable was scaled to the range $(0, 1)$. The dataset did not consist of any missing values.

6) Abalone Age: The Abalone Age dataset consists of 4,177 instances with seven numeric features, one categorical feature and one regression target variable. The target variable contains the age of abalones, which is determined by cutting the abalone's shell through a cone, staining it, and counting the number of rings through a microscope. The categorical `Sex` feature of the dataset was transformed into a one-hot encoded vector representation. The target variable was scaled to the range $(0, 1)$. The dataset did not consist of any missing values.

B. Hyper Parameter Tuning

A NN requires an optimal set of hyper parameter values in order to obtain the best performance of the model. These hyper parameters are problem dependent and require extensive tuning in order to determine the optimal values. A grid search cross validation approach was implemented to determine these optimal values. The parameters were searched over a grid of possible values and the results for each parameter combination were evaluated with 5-fold cross validation. The values with the best mean cross validation scores were selected as the optimal set of hyper parameters. The F1 scores from the classification problems and the R-squared scores from the regression problems were used as the cross validation scores. The grid of hyper parameters that required tuning is listed below:

- `n_epochs`: the number of epochs that the NN is trained for.
- `batch_size`: the size of the batches of data trained per each epoch.
- `learning_rate`: the learning rate for the SGD algorithm.
- `momentum`: the momentum parameter for the SGD algorithm.
- `weight_decay`: the λ parameter for the weight decay regularisation approach.

Table I lists the NN architecture and the hyper parameters that were obtained for each problem from the grid search cross validation algorithm. These hyper parameter values were used with the Passive Learning, Uncertainty Sampling and SASLA algorithms as described in Section III to train the NN models for each classification and regression problem.

C. Performance Metrics

The performances for each NN model was evaluated using the test set. The performance metrics that were considered for the classification problems were accuracy, precision, recall and F1 score. The accuracy measures how often the model's predictions are correct overall however this metric is misleading for imbalanced datasets. The precision measures how many of instances predicted to one class actually belong to that class, and recall measures how many of the actual instances of that class were correctly identified by the model. The F1 score is the harmonic mean between the precision and recall metrics to balance their trade-off.

The performance metrics considered for the regression problems were the mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) and the R-squared value. The MSE calculates the average of the squared differences between predicted and actual values, where larger errors are penalize heavily by the squaring function. The RMSE is the square root of MSE, providing an error metric in the same units as the target variable. The MAE measures the average absolute value of the difference between predictions and actual values, and is more robust to outliers. The R-squared value indicates the proportion of variance in the

TABLE I
THE HYPER PARAMETERS FOR PASSIVE LEARNING, UNCERTAINTY SAMPLING AND SASLA EXPERIMENTS

Problem	**NN Architecture**	**Train/ Validation/ Test Split**	**Number of Epochs**	**Batch Size**	**Learning Rate**	**Momentum**	**Weight Decay λ**
Breast Cancer Wisconsin	30-16-2	398/57/114	500	32	0.1	0.8	0.0001
Mobile Price Classification	20-12-4	2100/300/600	200	16	0.1	0.9	0.00001
Letter Recognition	16-24-26	14000/2000/4000	1000	32	0.5	0.9	0.00001
Boston Housing Prices	13-12-1	353/51/102	500	16	0.5	0.8	0.00001
Concrete Compressive Strength	8-8-1	721/103/206	500	16	0.5	0.8	0.00001
Abalone Age	9-16-1	2923/418/836	500	16	0.05	0.9	0.00001

dependent variable that is explained by the NN. The R-squared value ranges from 0 to 1, with values closer to 1 indicating a better model fit and predictive power.

The results for each training algorithm, Passive Learning, Uncertainty Sampling and SASLA, were recorded over multiple independent training runs. Ten independent NN models were trained for each learning approach and each problem dataset. The results from these independent runs were recorded and the averages of these results were used for the non-parametric statistical comparison tests and ranking approaches. The number of independent runs was selected to balance computational cost with the normality assumption, ensuring an accurate comparison of the means.

D. Statistical Tests

The non-parametric statistical comparison tests were all conducted at an alpha level of $\alpha = 0.05$. The Friedman test was used to determine whether a significant difference in the performance of a set of algorithms exists. The F1 scores from the classification problems were used to compare the multiple learning algorithms with the Friedman statistical test. For the regression problems, the R-squared scores were used for the statistical comparison tests. The null and alternative hypotheses for the Friedman test are:

- H_0: All algorithms perform equally on average, the median ranks of their performances are the same and any observed differences are due to random variation.
- H_a: At least one algorithm performs significantly different to the others and not all medians are equal.

If the null hypothesis is rejected then there is a significant difference between the algorithms. If there is a significant difference then the post-hoc pairwise Wilcoxon signed-rank tests were performed to identify which algorithms were significantly different. The results from the Wilcoxon signed-rank tests along with the average ranks for each algorithm were collected and displayed in critical difference plots.

The number of wins, losses and ties were calculated for each algorithm based on each performance metric. These wins, losses and ties were calculated by conducting a systematic pairwise comparison using paired t-tests. Each active learning method was compared against the passive learning benchmark across all datasets and performance measures. For each dataset the performance values from multiple independent runs were subjected to a paired one-tailed t-test. The outcomes of the tests were classified as follows: A win was assigned if the active learning method was statistically significantly better than the benchmark at a significance level of $\alpha = 0.05$; a loss was assigned if the active learning method was significantly worse; and a tie was assigned if there was no statistical significant difference between the two algorithms. This procedure was repeated for all datasets and performance measures, and the counts of each outcomes were aggregated into a final table of performance results.

V. RESULTS & DISCUSSION

The test results for the classification and regression performance of the Passive Learning, Uncertainty Sampling and SASLA algorithms are tabulated in Table II and III respectively. These results show the average performances for each algorithm across all datasets.

The performances of the learning algorithms on the classification datasets do not differ substantially and there was no clear algorithm which consistently performed the best. The results in Table II show that each learning algorithm performed better than the other algorithms for certain datasets according to the average F1 scores. For example, SASLA achieved the best F1 score on the Breast Cancer dataset, while Passive Learning performed best on Mobile Price, and Uncertainty Sampling was superior on the Letter Recognition dataset. This suggests that the benefits of each strategy are dataset-dependent rather than universally applicable. The highlighted F1 scores represent the highest performing algorithm for each dataset.

TABLE II
CLASSIFICATION RESULTS FOR EACH LEARNING STRATEGY

Problem	**Algorithm**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Breast Cancer	Passive	0.9842 ± 0.0035	1.0000 ± 0.0000	0.9571 ± 0.0095	0.9781 ± 0.0049
	Uncertainty	0.9833 ± 0.0026	1.0000 ± 0.0000	0.9548 ± 0.0071	0.9768 ± 0.0037
	SASLA	0.9868 ± 0.0044	1.0000 ± 0.0000	0.9643 ± 0.0119	$\mathbf{0.9818 \pm 0.0062}$
Mobile Price	Passive	0.8053 ± 0.0144	0.8483 ± 0.0156	0.8469 ± 0.0101	$\mathbf{0.8461 \pm 0.0132}$
	Uncertainty	0.8012 ± 0.0093	0.8502 ± 0.0077	0.8444 ± 0.0075	0.8408 ± 0.0113
	SASLA	0.8055 ± 0.0103	0.8531 ± 0.0103	0.8484 ± 0.0074	0.8457 ± 0.0095
Letter Recognition	Passive	0.7567 ± 0.0029	0.7785 ± 0.0038	0.7552 ± 0.0029	0.7530 ± 0.0034
	Uncertainty	0.7636 ± 0.0023	0.7822 ± 0.0039	0.7623 ± 0.0023	$\mathbf{0.7614 \pm 0.0027}$
	SASLA	0.7560 ± 0.0032	0.7784 ± 0.0027	0.7545 ± 0.0032	0.7524 ± 0.0032

Analysing the regression results reveals a similar pattern. No single learning algorithm consistently achieved the best performance across all datasets, as shown in Table III. Passive Learning achieved the highest R-squared value on the Boston Housing and Abalone datasets. Uncertainty Sampling obtained the best R-squared score on the Concrete Compressive Strength dataset, suggesting that it was better suited for the problem. The SASLA algorithm, while competitive, did not clearly outperform the other methods but maintained a performance that was competitive to the other scores. The Uncertainty Sampling algorithm performed the worst on the Abalone Age problem, which was the most complex problem. This suggests that Uncertainty Sampling is better suited to simpler regression tasks instead of complex ones.

TABLE III
REGRESSION RESULTS FOR EACH LEARNING STRATEGY

Problem	Algorithm	MSE	RMSE	MAE	R-squared
Boston Housing	Passive	0.0062 ± 0.0004	0.0789 ± 0.0023	0.0515 ± 0.0019	**0.8279 ± 0.0101**
	Uncertainty	0.0066 ± 0.0003	0.0813 ± 0.0019	0.0548 ± 0.0011	0.8176 ± 0.0089
	SASLA	0.0069 ± 0.0008	0.0828 ± 0.0047	0.0564 ± 0.0050	0.8099 ± 0.0221
Concrete Strength	Passive	0.0070 ± 0.0023	0.0828 ± 0.0117	0.0641 ± 0.0091	0.8250 ± 0.0566
	Uncertainty	0.0062 ± 0.0005	0.0784 ± 0.0029	0.0612 ± 0.0030	**0.8461 ± 0.0117**
	SASLA	0.0063 ± 0.0008	0.0792 ± 0.0048	0.0611 ± 0.0035	0.8424 ± 0.0199
Abalone Age	Passive	0.0065 ± 0.0001	0.0808 ± 0.0006	0.0589 ± 0.0010	**0.5274 ± 0.0067**
	Uncertainty	0.0071 ± 0.0004	0.0841 ± 0.0024	0.0642 ± 0.0043	0.4869 ± 0.0292
	SASLA	0.0065 ± 0.0000	0.0808 ± 0.0003	0.0587 ± 0.0013	0.5268 ± 0.0031

The boxplots in Figure 1 and 2 summarizes the performance metrics over the ten independent runs for each learning algorithm. The performance metrics were averaged across each problem dataset. The classification boxplots in Figure 1 visualise the minor differences between the performances for each learning algorithm on the classification problems datasets.

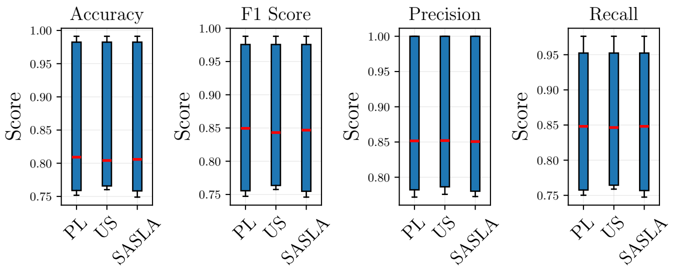

Fig. 1. Classification results boxplots for Passive Learning (PL), Uncertainty Sampling (US) and SASLA.

Noticeable variations across the regression boxplots in Figure 2 can be seen. The R-squared boxplot for Uncertainty Sampling has a longer tail than the other learning algorithms, indicating the results for Uncertainty Sampling contained smaller minimum values. This suggests that the Uncertainty Sampling algorithm has the potential to perform worse than the other learning algorithms for more complex regression problems.

Figure 3 represents the average F1 and R-squared scores for the Uncertainty Sampling as the size of the training set increases. The average scores and their error bars are plotted at for each sampling interval of the Uncertainty Sampling algorithm. The blue plots represent the classification problem datasets and the red plots represent the regression problem datasets. A general trend can be seen with the performance scores increases as the size of the training set increases. The plot in Figure 3b stands out as the F1 score initially decreased when the new uncertain instances were added to the training set before and increased after more than 1,000

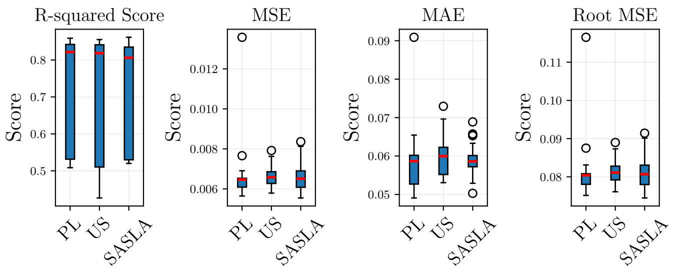

Fig. 2. Regression results boxplots for Passive Learning (PL), Uncertainty Sampling (US) and SASLA.

samples were added to the training set. Figure 3a shows how quickly the Uncertainty Sampling algorithm converged to a stable F1 score. The algorithm required substantially fewer training instances to achieve a level of performance that is comparable to Passive Learning and SASLA approaches. This further highlights the advantage computational advantage of Uncertainty Sampling requiring fewer training instances.

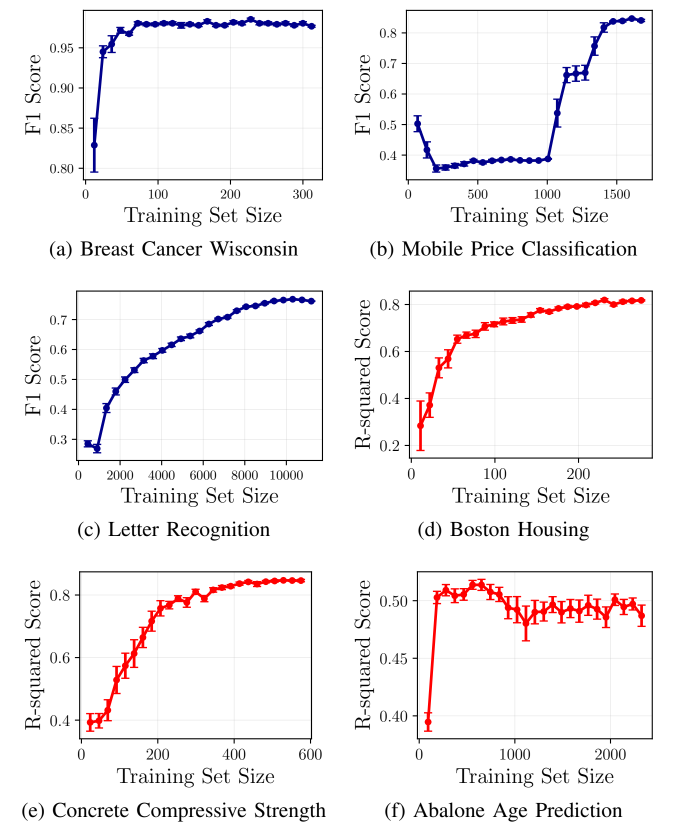

Fig. 3. The Uncertainty Sampling performance results over time as the training subset increases.

The error bars at each selection interval follows a general convergence trend where the average performance scores become more stable as more samples are added to the training set. The results in Figure 3f do not follow this trend which is explained by the complexity of problem. This provides reason to suggest that Uncertainty Sampling does not perform well for complex regression tasks and should instead be considered for classification tasks where resources such as compute and training data are limited.

Figure 4 represents the pattern selection over time for the SASLA algorithm across the six different datasets. Each point in the plots represent the size of the sampled training set from the candidate set that was deemed to have informative patterns by the SASLA selection algorithm. A somewhat general trend can be seen in the plots where the size of the sampled patterns from the candidate set decreases over time for each epoch. This trend however is not consistent among all datasets and some problems sample more data than previous iterations as seen in Figures 4d and 4e.

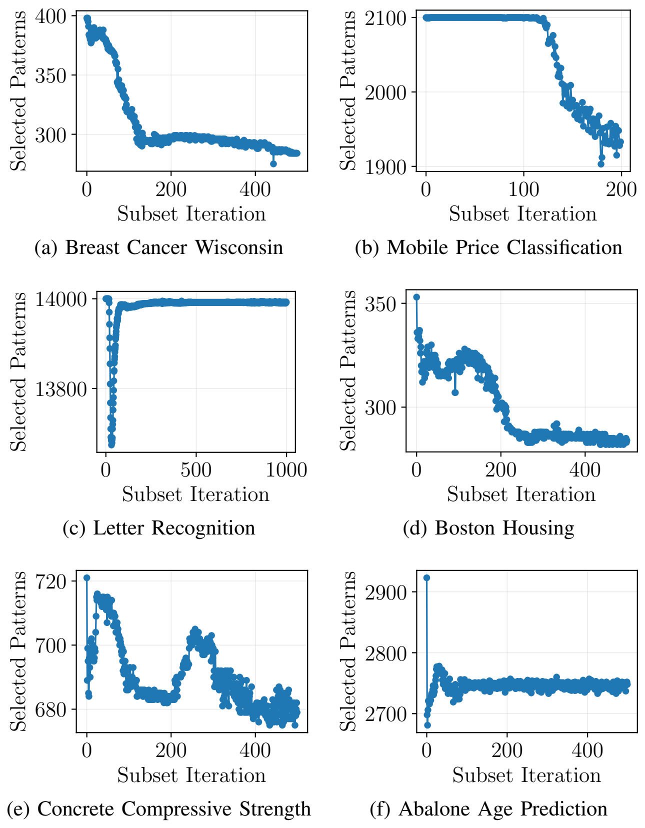

Fig. 4. Pattern selection over time by the SASLA algorithm across the six datasets.

Another noticeable insight to the SASLA algorithm is that it tended to select fewer patterns for the simpler problems than the more complex problems. The pattern selections for the simple problems, shown in Figures 4a and 4d, show an immediate decrease in the number of patterns considered for training. This is a stark contrast to the pattern selection plots in Figures 4c and 4f for the complex problems. The number of selected patterns initially decreased before rapidly increasing again and stabilising after a few iterations. This indicates that the SASLA algorithm could not differentiate between uninformative and informative patterns for complex problems. Instead, the SASLA algorithm identified a large portion of the candidate training set as informative, indicating that NN model is unsure about most of the predictions it generates.

TODO: EXPLAIN STATISTICAL TESTS

TABLE IV
THE NUMBER OF CLASSIFICATION DATASETS ON WHICH UNCERTAINTY SAMPLING AND SASLA SIGNIFICANTLY WIN (W), TIE (T), OR LOSE (L) COMPARED TO THE PASSIVE LEARNING BASELINE

	Accuracy	**F1 Score**	**Precision**	**Recall**
Algorithm	**W/T/L**	**W/T/L**	**W/T/L**	**W/T/L**
Uncertainty	1/2/0	1/2/0	1/2/0	1/2/0
SASLA	0/3/0	0/3/0	0/3/0	0/3/0

TABLE V
THE NUMBER OF REGRESSION DATASETS ON WHICH UNCERTAINTY SAMPLING AND SASLA SIGNIFICANTLY WIN (W), TIE (T), OR LOSE (L) COMPARED TO THE PASSIVE LEARNING BASELINE

	R-squared	**MAE**	**MSE**	**RMSE**
Algorithm	**W/T/L**	**W/T/L**	**W/T/L**	**W/T/L**
Uncertainty	0/1/2	0/1/2	0/1/2	0/1/2
SASLA	0/2/1	0/2/1	0/2/1	0/2/1

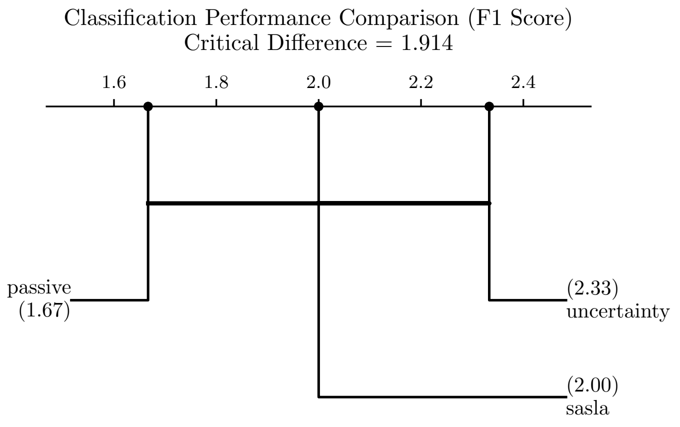

Fig. 5. Critical difference plot for the average ranks of the learning algorithms performance for the classification problems

VI. CONCLUSIONS

REFERENCES

[1] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, New York, NY, USA: Springer-Verlag, 1994, pp. 3–12.

[2] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *arXiv preprint arXiv:1612.01474*, 2017. [Online]. Available: https://arxiv.org/abs/1612.01474

[3] M. Sharma and M. Bilgic, "Evidence-based uncertainty sampling for active learning," *Data Mining and Knowledge Discovery*, vol. 31, no. 1, pp. 164–202, Jan. 2017, doi: 10.1007/s10618-016-0460-3.

[4] A. Engelbrecht, "Sensitivity analysis for selective learning by feedforward neural networks," *Fundamenta Informaticae*, vol. 45, pp. 295–328, Aug. 2001, doi: 10.3233/FUN-2001-45402.

[5] W. Wolberg, O. Mangasarian, N. Street, and W. Street. "Breast Cancer Wisconsin (Diagnostic)," UCI Machine Learning Repository, 1993. [Online]. Available: https://doi.org/10.24432/C5DW2B.

[6] I. Abhishek, "Mobile Price Classification," Kaggle, 2017. [Online]. Available: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification

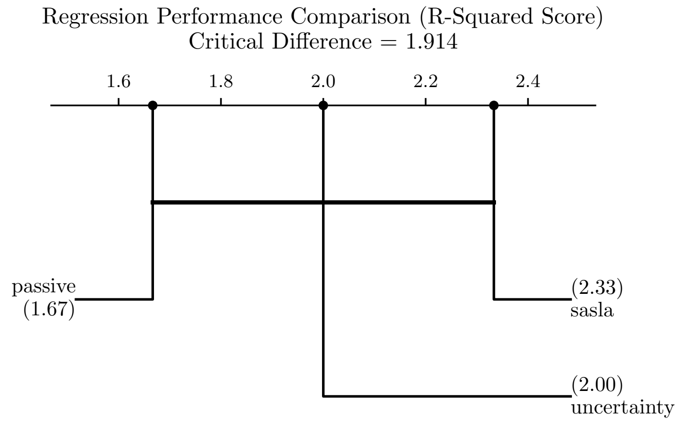

Fig. 6. Critical difference plot for the average ranks of the learning algorithms performance for the regression problems

[7] D. Slate. ”Letter Recognition,” UCI Machine Learning Repository, 1991. [Online]. Available: https://doi.org/10.24432/C5ZP40.

[8] D. Harrison and D. L. Rubinfeld, “Hedonic prices and the demand for clean air,” *Journal of Environmental Economics and Management*, vol. 5, pp. 81–102, 1978.

[9] I. Yeh. ”Concrete Compressive Strength,” UCI Machine Learning Repository, 1998. [Online]. Available: https://doi.org/10.24432/C5PK67.

[10] W. Nash, T. Sellers, S. Talbot, A. Cawthorn, and W. Ford. ”Abalone,” UCI Machine Learning Repository, 1994. [Online]. Available: https://doi.org/10.24432/C55C7W.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, G. Tang, Y. Li, M. Wong, J. Auer, D. Auer, B. Yu, A. Li, K. Xu, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.