Active Learning in Neural Networks

DT Nicolay 26296918
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
26296918@sun.ac.za

***Abstract*—TODO**
***Index Terms*—TODO**

I. Introduction

The theoretical background of active learning is presented in section 2, while section 3 walks through the implementation of the two active learning approaches with a neural network architecture. Section 4A describes the dataset preparation, followed by the performance criteria in section 4B. Section 4C and 4D outline the experimental design and framework for analysis. Section 5 summarises the results of the various algorithms and discusses their comparative performances with statistical analyses.

II. Background

The active learning paradigm and various approaches to active learning are discussed in this section. We explain how active learning is useful and how it is applied.

A. Passive Learning

Traditional supervised learning approaches perform passive learning during training where the algorithm has no control over which training samples it receives. That is, when techniques such as Stochastic Gradient Descent (SGD) are used, samples of the data are taken at random to update model parameters iteratively. SGD represents a form as passive learning since the model does not *choose* the data points it learns from. This approach is interpretable and easy to implement, especially since there is no overhead required for observation selection.

Mathematical updates during SGD in passive learning can be expressed as:

$$w_{t+1} = w_t - \eta \nabla_w L(x_i, y_i; w_t)$$

where η is the learning rate, (x_i, y_i) is a randomly sampled observation from the training data at iteration t and L represents the loss function.

Random sampling is unbiased, but potentially inefficient in the presence of high noise and variance [1]. Passive learning assumes that the distribution of the training data matches that of the test distribution, and that all the training observations contribute equally. This assumption does not always hold in practice, leading to inefficient computations. More specifically, the algorithm may waste effort on uninformative observations.

B. Active Learning Paradigm

In supervised learning, acquiring labeled training data for a predictive model can be very costly, but acquiring a large amount of unlabeled data is often quite easy. Active learning is a method of obtaining predictive models with high precision at a limited cost through the adaptive selection of samples for labeling [2]. Rather than passively accepting training examples from the teacher, the network is allowed to use its current knowledge about the problem to have some deterministic control over training examples, and to guide the search for informative patterns [3]. This may result in shorter training times due to fewer observations needing to be considered by the algorithm. That is, if the complexity of the observation selection approach does not exceed the reduction in training time achieved by considering fewer observations. Therefore, careful consideration is required to decide on an approach to observation selection.

C. Output Sensitivity Analysis

For this approach, we begin by considering the entire training set and iteratively remove observations that are shown to be less informative. The neural network uses its learned knowledge of the distribution at each selection interval to do so.

The core of this algorithm is based on pattern informativeness. The following definitions and the corresponding mathematical formulation follow the approach introduced by Engelbrecht [3]. An informative pattern is defined as one that as a strong influence of the neural network outputs, whilst an uninformative pattern has a negligible effect. That is, the informativeness of a pattern is the sensitivity of the neural network output vector to small perturbations in the input vector. Denote the informativeness of pattern p as $\Phi^{(p)}$. Then,

$$\Phi^{(p)} = \|\mathbf{S}_o^{(p)}\|, \tag{1}$$

where $\mathbf{S}_o^{(p)}$ is the output sensitivity vector for pattern p (defined in (3)), and $\|\cdot\|$ is any suitable norm. We consider the maximum-norm:

$$\Phi(_\infty p) = \|\mathbf{S}_o^{(p)}\|_\infty = \max_{k=1,\ldots,K} \{\left|S_{o,k}^{(p)}\right|\}, \tag{2}$$

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

The output sensitivity vector is defined as

$$\mathbf{S}_o^{(p)} = \|\mathbf{S}_{oz}^{(p)}\|, \tag{3}$$

where $\mathbf{S}_{oz}^{(p)}$ is the output–input layer sensitivity matrix. Assuming sigmoid activation functions in both the hidden and output layers, each element $S_{oz,ki}^{(p)}$ of the sensitivity matrix is computed as

$$S_{oz,ki}^{(p)} = o_k^{(p)}\big(1 - o_k^{(p)}\big) \sum_{j=1}^{J} w_{kj}\, y_j^{(p)}\big(1 - y_j^{(p)}\big) v_{ji}, \tag{4}$$

where w_{kj} is the weight between output unit o_k and hidden unit y_j, v_{ji} is the weight between hidden unit y_j and input unit z_i, $o_k^{(p)}$ is the activation value of output o_k, $y_j^{(p)}$ is the activation of hidden unit y_j, and J is the total number of hidden units (including the bias unit to the output layer).

Suitable norms for calculating the output sensitivity vector include the sum-norm or the Euclidean norm, i.e.,

$$S_{o,k}^{(p)} = \|S_{oz}^{(p)}\|_1 = \sum_{i=1}^{I} \big|S_{oz,ki}^{(p)}\big|, \tag{5}$$

or

$$S_{o,k}^{(p)} = \|S_{oz}^{(p)}\|_2 = \sqrt{\sum_{i=1}^{I} \big(S_{oz,ki}^{(p)}\big)^2}, \tag{6}$$

where I is the total number of input units (including the bias unit to the hidden layer).

Using Equation 2, a pattern is considered *informative* if one or more of the output units is sensitive to small perturbations in the input vector. The larger the value of $\Phi_1^{(p)}$, the more informative the pattern p is.

To illustrate, assume gradient descent is used to find optimal weight values:

$$\Delta w_{kj},\, \Delta v_{ji} \propto \big(t_k^{(p)} - o_k^{(p)}\big), \tag{7}$$

where t_k is the target value for output unit o_k for pattern p. Each new pattern can be viewed as a perturbation of a previously presented pattern. Let $\Phi_\infty^{(p)} = \big|S_{o,k}^{(p)}\big|$. If $\Phi_\infty^{(p)}$ is large, the output value of o_k changes significantly compared to the previous presentation, making pattern p highly informative. Conversely, if $\Phi_\infty^{(p)}$ is small, no significant change in o_k occurs, and pattern p is an insignificant contributor to the gradient direction, thus uninformative for the learning process.

D. Uncertainty Sampling

Uncertainty sampling, a frequently utilised active learning strategy, selects instances about which the model is uncertain but it does not consider the reasons for why the model is uncertain [4]. There are two reasons that a model may be uncertain about an observation. Considering a classification scenario, *conflicting-evidence uncertainty* is where there is strong conflicting evidence for multiple classes. On the other hand, *insufficient-evidence uncertainty* describes the situation where there is not enough evidence to classify an observation to any class. Uncertain instances often lie close to the decision boundary. Understandably, a model's variance on *conflicting-evidence uncertainty* is higher than *insufficient-evidence uncertainty*.

In contrast with the output sensitivity approach, we begin with an empty set of observations and iteratively include the most uncertain observations. That is, we select an informative instance $\langle x^*, ?\rangle \in U$ and incorporate the new labelled instance $\langle x^*, y\rangle$ into L. More formally, Algorithm 1 was presented by Sharma and Bilgic [4], and describes the process of building the subset.

Algorithm 1 Pool-Based Active Learning

Input: U - unlabeled data, L - labeled data, θ - classification model, B - budget
repeat
  for all $\langle x^{(i)}, ?\rangle_i \in U$ **do**
    compute utility$(x^{(i)}, \theta)$
  end for
  pick highest utility x^* and query its label
  $L \leftarrow L \cup \{\langle x^*, y^*\rangle\}$
  $U \leftarrow U \setminus \{\langle x^*, y^*\rangle\}$
  Train θ on L
  $B = B - 1$
until $B == 0$

This study considers two ways to quantify uncertainty. First, given a probabilistic classifier that outputs a distribution over classes $P(y|x, \mathbf{w})$, the predictive entropy of an observation x is defined as

$$H(x) = -\sum_{c=1}^{C} P(y = c \mid x, \mathbf{w}) \log P(y = c \mid x, \mathbf{w}). \tag{8}$$

This approach has been widely used in literature [5]–[7], however often along with other techniques.

A second approach to measuring uncertainty involves using model ensembles. Instead of relying on a single model's output, construct an ensemble of models (via bootstrap sampling, random initialisations, or subspace sampling), and measure the disagreement among them. That is, for k models, compute

$$\mathrm{Var}_k(P(y \mid x, \mathbf{w}_k)), \tag{9}$$

where $\bar{P}(y \mid x)$ is the mean predictive distribution across the models. Empirically, ensemble methods often provide more reliable uncertainty estimates than single-model entropy, especially in settings with model misspecification or limited data [8].

III. Implementation

A. Neural Network Architecture

The neural network architecture is composed of one hidden layer, of which the number of hidden units is determined through a grid search for each problem. Figure 1 illustrates an overview of the architecture. Each hidden and output

layer both use sigmoid activation functions. Weight decay, specifically L2 regularisation, is considered by experimenting with various penalty terms of $[0.0, 0.001, 0.01, 0.1]$ during the grid search. Momentum values of $[0.0, 0.5, 0.9, 0.95]$ are also considered during the grid search.

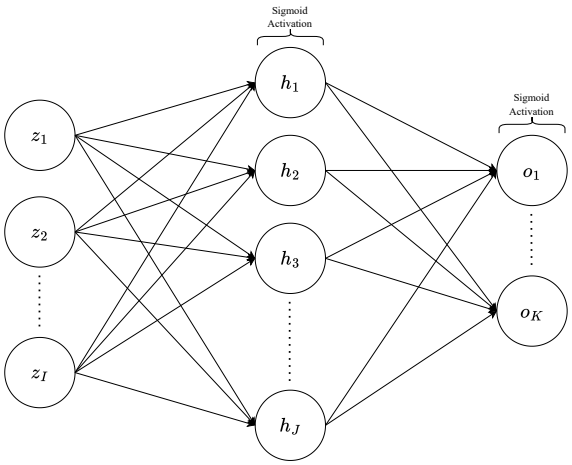

Fig. 1. Neural network architecture. Here, z_i represents input units, h_i represents hidden units, and o_i represents output units.

The grid search is only performed once using the MSE Loss given in Equation 10 in the passive learning setting. Here, N represents the number of training patterns, K represents the number of output units, $t_k^{(i)}$ is the target value for output unit k in pattern i, $o_k^{(i)}$ is the network output of unit k in pattern i, w_j is the weight parameter and λ is the weight decay coefficient. The same parameter values obtained from the passive learning grid search are used in the active learning procedures in order to maintain the focus of the comparison on the algorithms and not the parameters. SGD is used to determine the optimal parameter values.

$$L_{\text{MSE+WD}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(t_k^{(i)} - o_k^{(i)}\right)^2 + \lambda\sum_j w_j^2 \qquad (10)$$

B. Output Sensitivity Algorithm

The output sensitivity active learning implementation follows the background derivations previously described. A conservative pattern selection constant, $\beta = 0.9$ is used. That is, the algorithm only picks the most informative samples, keeping the training subset small at first.

The algorithm operates by repeatedly training the neural network on training subset D_T until a termination criterion on D_T is triggered. In this implementation, a budget of 1000 epochs is used. The exact gradient-based output sensitivity is computed using Equation 4. The pattern informativeness is then computer using both sum-norm and max-norm metrics in Equation 1 and 2. The average pattern informativeness is calculated and multiplied by $(1 - \beta)$ to determine the selection threshold. The patterns with informativeness above this threshold are selected from the candidate set. A safety mechanism was implemented here where we ensure that at least one sample per class is kept. Initially these calculations were performed linearly, however vectorising them results in a massive improvement in performance. A selection interval parameter is used to determine after how many epochs the selection takes place, this is set to one unless otherwise stated.

C. Uncertainty Sampling Algorithm

Both entropy and ensemble based uncertainty sampling were implemented. The Pool-Based Active Learning Framework described in Algorithm 1 is implemented. First, we start with a small labelled set L and large unlabelled pool U. The initial labelled set is defined as the smaller of 10% of the training data or 10 observations. Then, for each query iteration, we computer the utility (uncertainty) for all the samples in U. This is computed using the softmax of the output units,to obtain probabilities for use in the equation Equation 8. We further normalise the raw entropy obtained from Equation 8 by dividing by $\log(C)$, this is to ensure that the entropy is in the range [0, 1]. The highest utility sample is selected and its label queried, then moved from U to L. Now, we simply retrain the model on the updated L set and repeat this process until the budget is exhausted. Here, the budget is defined as the number of remaining samples.

For the ensemble-based approach, three independent neural networks are trained simultaneously with different random weight initialisations to create diversity in their learned representations. Rather than using entropy of a single model's output distribution as the uncertainty metric, we exploit the disagreement among ensemble members. Specifically, for each unlabelled sample $\mathbf{x}_i \in U$, we perform forward passes through all ensemble members to obtain softmax-normalised probability distributions $\{\mathbf{p}_i^{(1)}, \mathbf{p}_i^{(2)}, ..., \mathbf{p}_i^{(n_{ensemble})}\}$. The uncertainty is then quantified as the total predictive variance across the ensemble using Equation 9. Higher variance indicates greater disagreement among models, suggesting the sample lies in an ambiguous region of the feature space where additional labelled data would be most informative. During training, all ensemble members are updated synchronously on the same labelled set L using independent optimisers, and final predictions are made via majority voting across the ensemble. This approach trades increased computational cost (linear in $n_{ensemble}$) for potentially better-calibrated uncertainty estimates, particularly valuable when dealing with limited labelled data where single-model confidence may be poorly calibrated.

D. Software and Hardware Specifications

All experiments were performed on Fedora Linux system with a 13th Gen Intel Core i3-13100 x8 CPU and 16GiB of memory. The code for this paper is available on GitHub [9].

IV. Emperical Process

The datasets along with their respective pre-processing measures are presented in this section, followed by a summary of the performance criteria and the experimental design. Finally, the statistical analysis framework is described.

A. Dataset Selection and Pre-processing

Three classification and regression problems were selected each. One simple, one medium, and one hard problem were selected for each to illustrate how the algorithms apply to varying problem complexities.

TODO

B. Performance Criteria

For the classification tasks, accuracy was the primary metric used for model performance evaluation across all learning strategies. The F1-Score was also recorded and considered for a more comprehensive, balance metric. The precision and recall and also tracked. The MCC, which is a robust metric that accounts for class imbalances, was recorded. In each training iteration, we record the number of training observations seen by the model as it trains. The training set size reduction for the active learning methods were recorded across the epochs. Epochs to convergence was recorded for each method, that is, the number of training epochs required for the model to achieve a stable performance. The convergence rate was computed by calculating the proportion of the trials achieving the convergence threshold.

For the function approximation tasks

C. Experimental Design

D. Statistical Analysis Framework

V. Results & Discussion

A. Dataset Characteristics

B. Control Parameter Optimisation Results

C. Classification Problem Results

D. Function Approximation Results

E. Comparative Analysis

F. Discussion of Findings

VI. Conclusions

TODO

References

[1] M. Mahdavi and R. Jin, "Passive learning with target risk," *Journal of Machine Learning Research*, vol. 30, 02 2013.

[2] H. Hino, "Active learning: Problem settings and recent developments," *ArXiv*, vol. abs/2012.04225, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:227745938

[3] A. Engelbrecht, "Sensitivity analysis for selective learning by feedforward neural networks," *Fundam. Inform.*, vol. 45, pp. 295–328, 08 2001.

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

[5] J. Zhu, H. Wang, T. Yao, and B. Tsou, "Active learning with sampling by uncertainty and density for word sense disambiguation and text classification." 01 2008, pp. 1137–1144.

[6] T. He, S. Zhang, J. Xin, P. Zhao, J. Wu, X. Xian, C. Li, and Z. Cui, "An active learning approach with uncertainty, representativeness, and diversity," *The Scientific World Journal*, vol. 2014, p. 827586, 2014. [Online]. Available: https://doi.org/10.1155/2014/827586

[7] Y. Yang and M. Loog, "Active learning using uncertainty information," 2017. [Online]. Available: https://arxiv.org/abs/1702.08540

[8] T. Yin, G. Panapitiya, E. D. Coda, and E. G. Saldanha, "Evaluating uncertainty-based active learning for accelerating the generalization of molecular property prediction," *Journal of Cheminformatics*, vol. 15, no. 1, p. 105, 2023. [Online]. Available: https://doi.org/10.1186/s13321-023-00753-5

[9] D. Nicolay, "Ml441 assignments," https://github.com/Voltzz9/ml441_assignments, 2025.

[10] R. A. Fisher, "Iris," 1936. [Online]. Available: https://archive.ics.uci.edu/dataset/53/iris

[11] S. Aeberhard and M. Forina, "Wine," 1991. [Online]. Available: https://archive.ics.uci.edu/dataset/109/wine

[12] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017, available at https://github.com/zalandoresearch/fashion-mnist.

[13] R. K. Pace and R. Barry, "Sparse spatial autoregressions," 1990, dataset obtained from Kaggle. [Online]. Available: https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

[14] A. Tsanas and A. Xifara, "Energy Efficiency," UCI Machine Learning Repository, 2012, DOI: https://doi.org/10.24432/C51307.

TABLE I
DATASET DESCRIPTIONS

Dataset	**Missing Values**	**Distribution / Notes**	**Source**
Iris	none	setosa - 33.3% versicolor - 33.3% virginica - 33.3%	[10]
Wine	none	cultivar 0 - 33.1% cultivar 1 - 39.9% cultivar 2 - 27.0%	[11]
Fashion-MNIST	none	10 equally distributed classes	[12]
Sine + Noise	none	synthetic sinusoidal + Gaussian noise	synthetic
California Housing	few	housing variables, spatial data	[13]
Energy Efficiency	none	building energy data, multiple target variables	[14]