Comparison of uncertainty sampling and sensitivity analysis in active learning for feed forward neural networks

Sebastien Meniere

Stellenbosch University
Stellenbosch, South Africa
26484765@sun.ac.za

***Abstract*—This study evaluates three training regimes for one-hidden-layer neural networks under a fixed labeling budget: (i) passive learning with stochastic gradient descent (SGD), (ii) active learning via output sensitivity analysis (SALSA-style), and (iii) active learning via uncertainty sampling (ALUS-style). We consider both classification and function approximation, each across three datasets of varying complexity. For regression, uncertainty is implemented as predictive variance from an ensemble of shallow MLPs; sensitivity is measured via the input–output Jacobian norm and an expected model-change proxy. For classification, we use margin-based uncertainty and the same model-change proxy. We design a unified pool-based active learning framework with consistent model capacity, weight-decay regularization, and a statistically sound evaluation protocol (learning curves and area-under-the-learning-curve, AULC) with repeated seeds. **[results]** summarize the relative label efficiency and final performance across datasets, providing practical guidance on when SALSA- or ALUS-style selection should be preferred over passive sampling.**

INTRODUCTION

Active learning (AL) aims to reduce labeling cost by querying the most informative samples from an unlabeled pool. In continuous function approximation or regression where uncertainty manifests as predictive variance and sensitivity manifests as large local derivatives or expected model change. This report investigates whether two canonical AL families, uncertainty sampling (ALUS) and output sensitivity (SASLA), deliver measurable gains over passive learning (random sampling with SGD) under a controlled neural-network setting.

Research question: Under a shallow MLP with intentional over-parameterization and weight decay, when do SALSA-style sensitivity and ALUS-style uncertainty outperform passive learning for (a) classification and (b) regression? We study label efficiency (performance vs. labels acquired), stability across seeds, and final performance at a fixed budget.

Contributions: (1) A unified AL framework for shallow networks that treats classification and regression consistently via task-appropriate acquisition functions; (2) a regression-specific adaptation of uncertainty (ensemble variance) and sensitivity (Jacobian norm / expected model change) compatible with standard autodiff; (3) a statistically grounded empirical process using repeated seeds, learning curves, and AULC; and (4) a set of compact, reproducible baselines suitable for coursework and future extensions.

BACKGROUND

Passive versus active learning: All models use randomly partitioned sets: train, validation and test of the full dataset. Passive or fixed size learning (FSL) uses the training set, without modification; all points are used to update the parameter weights of the model. Active learning strategies aim to identify and select points from the training data that are most important to generalization. With the intention The active learning explicitly selects the points that are deemed most informative by some metric. The strategies examined here are sensitivity analysis for selective active learning (SASLA) and uncertainty sampling for active learning (ALUS). These two active learning strategies differ in both their initial starting condition and their heuristic for determining which points in the training data are useful for training.

either start with the full training set and select only the points that are most important.

Pool-based active learning iteratively trains a model, scores unlabeled candidates with an acquisition function, and queries the most informative points. For classification, popular scores include least-confidence, margin, and entropy; for regression/function approximation, uncertainty is commonly defined via predictive variance (e.g., Bayesian models, Gaussian processes, or model ensembles), and sensitivity/experimental-design criteria target regions with high curvature or expected error reduction.

Uncertainty-based active learning: Uncertainty sampling for active learning as defined by [ALUS] uses a pool-based approach. The training data D_train is partitioned into a labeled set D_label and an unlabled set $D_unlabeled$ also known as pools. The learner iteratively selects an infromative instance(s) from the $D_unlabeled$ using some utility criterion to incorperate into $D_labeled$. After aquiring the new instance(s) the learner is retrained on the modified $D_labeled$ before repeating the process. The aquisition, retrain process continues until the budget B is exhausted. Algorithm 1 shows the general pool-based active learning process.

a) Classification: Pool-based active learning assumes a small labeled set L and a larger unlabeled pool U. At each round, the learner selects an informative unlabeled instance by a utility criterion, queries its label from an oracle, augments L, prunes from U, and retrains the model; the process continues until a budget B is exhausted (Algorithm 1) . Uncertainty can be measured by entropy, maximum conditional (least confidence), or **margin**; the margin criterion selects instances with the smallest difference between the top two class probabilities $\mathrm{P}_\theta(y_m|x) - \mathrm{P}_\theta(y_n|x)$. The **evidence-based** perspective further distinguishes two kinds of uncertain instances, **conflicting-evidence** versus **insufficient-evidence**, using multiplicative or related rankings of per-class evidences $E^{+1}(x)$ and $E^{-1}(x)$; within the most uncertain set S, conflicting-evidence prefers large $E^{+1}E^{-1}$ and insufficient-evidence prefers small $E^{+1}E^{-1}$.

Sensitivity-based Acquisition: Selective learning aims to focus training on patterns most likely to refine decision boundaries, typically those near the region of uncertainty. Prior work motivates that concentrating training on patterns close to the decision boundary can improves generalization compared to fixed label set training (FLS). SASLA defines pattern informativeness as the norm of the NN output sensitivity vector with respect to small input perturbations: $\iota(p) := \|\tilde{S}_o^{(p)}\|$, with a suggested max-norm over output units for classification. The output–input sensitivity matrix element $S_{oz;ki}^{(p)}$ is given for sigmoidal networks and depends on forward activations and weights, enabling a Jacobian-style computation across the network layers. SASLA executes in epochs; at a subset selection interval (often each epoch), the algorithm computes informativeness for all candidate patterns and retains only those exceeding a scaled average informativeness threshold controlled by a subset selection constant*τ (paper uses α in $[0, 1]$): patterns with informativeness larger than $(1 - \alpha)$ times the average are selected. A conservative α close to one selects more patterns initially; as training progresses, more patterns become uninformative and the training subset shrinks. If $\alpha = 1$, the method reduces to fixed-set learning (FSL) . The SASLA loop alternates between training on the current subset until a termination criterion, recomputing sensitivities for all candidate patterns, and updating the next subset accordingly; the paper recommends selecting a new subset at each epoch and starts from the full candidate set as the initial subset.

Model and Regularization Choices: To isolate acquisition effects, we fix the hypothesis class to a one-hidden-layer MLP with an overestimate of hidden units and apply weight decay (L2) to counter overfitting. This setup is intentionally simple, emphasizing acquisition policy rather than representational power. Early stopping on a validation split and consistent hyperparameters across learners maintain fairness.

IMPLEMENTATION

All implementation for this research was done in *Python 3.11.3*. Shallow neural networks created with *PyTorch 1.8*, data preprocessing performed with *Sklearn 1.2.2*. Experimental results analysis and plotting was done in *Jupyter notebook* using *Pandas*, *Numpy*, *Matplotlib.pyplot*.

Several python class modules were created to facilitate compute tracking, report metrics and results, specify model architecture, strategies and dataset sizes. To ensure determinism, any random selection or random number generation was done using specific set seeds. Ensuring that runs remained consistent and comparable.

The SASLA strategy for classification was implemented following little to no modifications. For regression

METHODOLOGY

EMPIRICAL PROCESS

RESULTS

REFERENCES

All code can be found at on my public git repository at