Math 385 Final Project: Active Learning with Model Selection (ALMS)

Michael Zhao

Abstract

This project studies active learning with model selection (ALMS) for regression. The main problem is simple: active learning needs a model to decide what to label, but model selection needs labels to decide the model. I implemented five methods: Passive MS, Normal AL, Sequential AL+MS, Batch AL+MS, and Ensemble Active Learning (EAL). I tested them on the Kaggle California housing dataset and on a synthetic piecewise dataset that I generated. I report MSE/MAE vs label budget, a summary table, and I explain when EAL helps and when it does not.

Introduction

Active learning (AL) tries to reduce test error using fewer labels. It does this by choosing which data points to label next. Model selection (MS) tries to choose the best model class and hyperparameters. In my project, that means polynomial degree and ridge strength.

The problem is that these two steps depend on each other. AL needs a model to score unlabeled points. MS needs labels to pick the model. This is the AL/MS dilemma described by Sugiyama and Rubens (2008). By the paper we introduced in the 'discuession' part, the ALMS model is subjected to the complexity of the data itself

My goal was:

- Implement several ALMS strategies in one codebase.
- Compare them on a real dataset (California housing).
- Build a synthetic dataset where EAL should have a clear advantage.
- Explain the results in a simple way.

Background

Regression and ridge

I use models that are linear in features:

$$\hat{f}(x) = w^\top \phi(x).$$

With labeled data (x_i, y_i), ridge regression fits w by

$$\hat{w} = \arg\min_w \ \|\Phi w - y\|_2^2 + \alpha \|w\|_2^2 \quad \Rightarrow \quad \hat{w} = (\Phi^\top \Phi + \alpha I)^{-1} \Phi^\top y,$$

where Φ is the design matrix and α is the ridge parameter.

Why AL + MS is hard

Active learning wants to choose a sampling distribution (or next query point) that reduces test error fast. But the "best" query depends on the model. If the model is wrong early, the method can keep sampling points that are informative for the wrong model. This can waste labels.

Ensemble Active Learning (EAL): what the paper proposes

Sugiyama and Rubens propose EAL to reduce the risk of committing to one model too early. The key definition is that EAL combines the active-learning objective across candidate models:

$$\widehat{G}^{(\mathrm{EAL})}(p) = \sum_{M \in \mathcal{M}} \widehat{G}_M^{(\mathrm{AL})}(p)\, P(M).$$

In words: we do not optimize AL for one model. We combine across models. This is the main reason EAL can be more stable when MS is uncertain.

Related work

Two ideas show up a lot in active learning for regression. First, the classic "Query by Committee" line of work chooses the next query by maximizing disagreement among a set of models trained on the same labeled data. This is a different scoring rule than the EAL score I used, but it gives the same general motivation: using multiple models can reduce the risk of committing to one wrong hypothesis too early. Second, recent theory work studies active learning for linear regression using leverage-score style sampling. This directly relates to my Normal AL and the building blocks inside my ALMS methods, since my query score is based on ridge/least-squares geometry and tries to pick points that reduce estimation uncertainty faster than random sampling.

Method

Model family and model selection

A model is a pair (d, α):

- d = polynomial degree (controls nonlinearity through feature expansion),
- α = ridge strength (controls regularization).

For a labeled set L, I run cross-validation over all candidates and choose the best (d, α).

Unlabeled pool setup

For each label budget n, I start with a small random labeled set of size `init_n`. All other training points are treated as an unlabeled pool. I simulate labeling by revealing y only when a point is selected.

Five strategies I implemented

Let L be labeled indices and U be the unlabeled pool.

(1) Passive MS. Randomly choose n labeled points. Run CV to pick (d, α). Evaluate on test.

(2) Normal AL. Fix one model (d_0, α_0) and never change it. Repeatedly pick the next point using an informativeness score (in my code: leverage-style score). Then evaluate using the same fixed model.

(3) Sequential AL+MS. Alternate:

$$\text{MS on } L \;\rightarrow\; \text{AL query using current model} \;\rightarrow\; \text{update } L.$$

I re-run MS every `ms_every` steps to reduce runtime. This can drift if early MS picks change a lot.

(4)Batch AL+MS . I first run CV on the warm-start labeled set to pick an initial model (d, α). Then I keep (d, α) fixed and acquire new points greedily one-by-one using the leverage score (recomputed after each acquisition). After reaching the budget, I run CV again on the full labeled set and evaluate the final chosen model.

(5) EAL. For each candidate model $m \in \mathcal{M}$, I compute a pool score $s_m(x)$ for every $x \in U$. In my code, $s_m(x)$ is a ridge-regression variance-reduction style quantity. I then combine scores across models using weights ω_m (I use CV-based weights):

$$s(x) = \sum_{m \in \mathcal{M}} \omega_m s_m(x), \qquad x^\star = \arg\max_{x \in U} s(x).$$

Then I query $x^\star$, move it from U to L, and repeat until the label budget is reached.

How EAL looks as code

This is the structure of the Python loop I used:

```
while len(labeled) < n:
    for model in candidates:
        fit ridge on labeled
        compute score on pool
    combine scores across models
    pick pool point with max combined score
    move it from pool to labeled
```

The important point is: EAL scores each pool point using many models, not just one.

Synthetic dataset design (to show EAL power)

Piecewise + distribution shift

I generated a dataset where different regions prefer different model complexity.

A simple math form is:

$$y = \begin{cases} a_1 x + b_1 + \varepsilon, & x < 0, \\ a_2 x + b_2 + \beta x^2 + \varepsilon, & x \geq 0, \end{cases} \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

So the right model is not purely linear everywhere.

Then I make the task harder in a specific way:

- The unlabeled *pool* under-samples the nonlinear side early.
- The *test* distribution gives more weight to the nonlinear side.

This creates a situation where early MS can easily prefer a low-degree model. If a method then queries mostly for that low-degree model, it can miss the nonlinear region. EAL is designed to avoid that because it keeps multiple models in the loop.

Experiments

Datasets

- **California housing (real):** Kaggle California Housing Prices dataset.
- **Synthetic piecewise (generated):** the piecewise + shift dataset described above.

Protocol

For each method and each label, budget n:

- Start with `init_n` random labeled points.
- Run the method until $|L| = n$.
- If the method includes MS, run CV at the end to choose (d, α).
- Fit ridge on L and evaluate on the test set.

Metrics

I use:

- MSE on the test set as the main metric.
- MAE as a second metric on California.
- AUC-style summary: average error over the budgets (lower is better).

Results and analysis

Synthetic piecewise (EAL advantage)

Figure 1 shows a clean story. EAL is the lowest curve across the whole budget range. It also has a small variability band. That means it is stable across runs.

Sequential AL+MS and Batch AL+MS also do well, but they sit above EAL. This matches the AL/MS dilemma idea: these methods still depend on the model picked early, so they can spend labels in the wrong region. EAL reduces that risk because it uses many candidate models to guide querying.

Passive MS is unstable here. The variability band is huge at low budgets. This happens because random sampling can miss the key nonlinear region early. If it misses it, CV can pick a model that looks good on the easy region but fails on test.

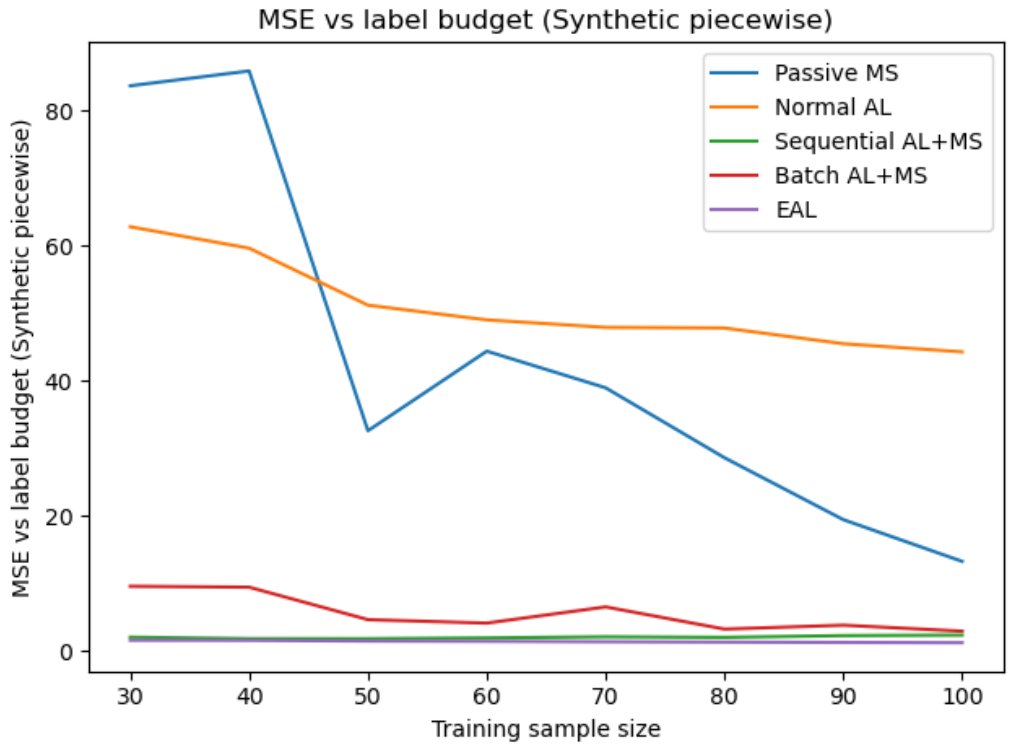


Figure 1: Synthetic piecewise: MSE vs label budget (mean with variability band).

Method	Final MSE @100	AUC MSE	Improvement vs Passive (%)
EAL	1.243 ± 0.039	1.421 ± 0.033	90.6%
Sequential AL+MS	2.371 ± 0.282	2.059 ± 0.085	82.1%
Batch AL+MS	2.948 ± 1.222	5.551 ± 1.046	77.8%
Passive MS	13.272 ± 18.951	43.335 ± 16.135	0.0%
Normal AL	44.265 ± 1.182	50.994 ± 1.117	-233.5%

Table 1: Synthetic piecewise summary. Lower is better for Final MSE and AUC MSE.

Normal AL is the worst here. This is expected. It is locked to one model, so it cannot adapt when the data needs a different complexity.

Table 1 makes the advantage even clearer. EAL has the best final MSE and the best AUC. The AUC result matters because it shows EAL is not only good at the end. It learns faster across the whole curve.

Also, notice the Passive MS standard deviation is very large. That means the dataset is sensitive to which points you label early. This is exactly when active learning can matter.

California housing (passive MS is hard to beat)

Figure 2 shows the opposite situation. Passive MS is the best curve. The gap stays even as we add labels. This suggests two things:

- Model selection is already stable on this dataset.

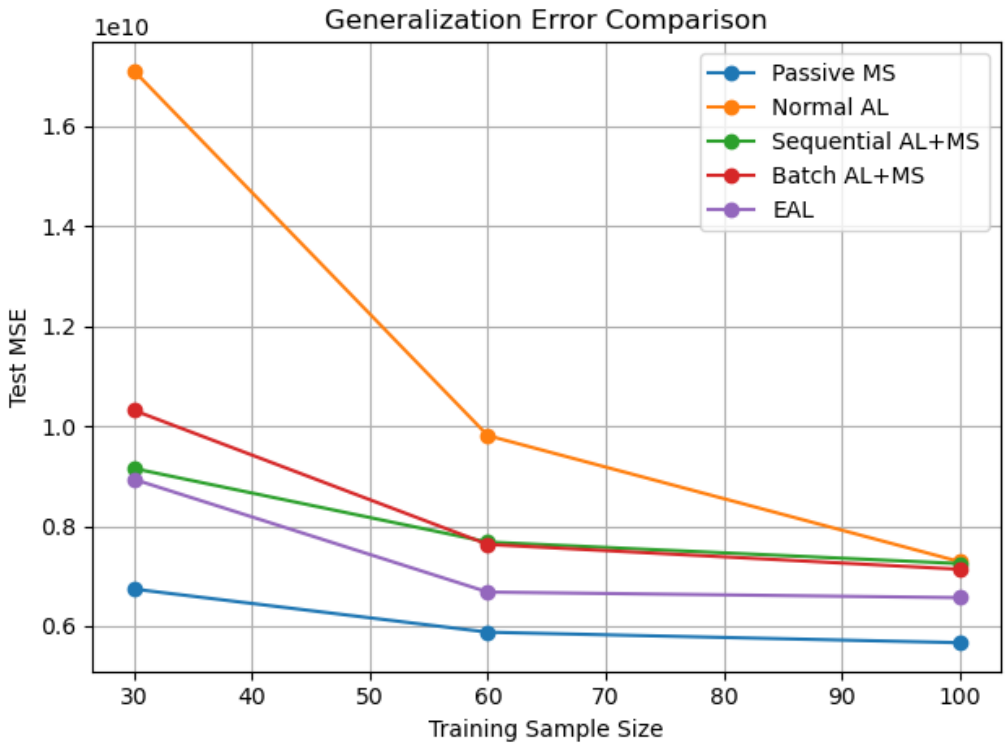


Figure 2: California housing: test MSE vs label budget (mean values).

- Random labeled points are already representative.

In this setting, active querying does not help much. Sometimes it can even hurt. A reasonable explanation is that the query score can over-focus on unusual points. Those points may be "informative" in a narrow way, but not useful for reducing average test error.

What I learned from the two datasets

The synthetic piecewise dataset shows the situation where EAL is most useful: Early model choice is risky, and different models want different data regions. California housing looks more like a case where MS is easy and random sampling is strong. So EAL has less room to improve.

Conclusion

My main conclusion is that ALMS is not automatically better than passive learning. It depends on whether model uncertainty is real early.

EAL works best when the true relationship has a structure that is not captured well by one simple model everywhere. In other words, it helps when the problem is closer to a nonlinear setting. My piecewise dataset is one simple example of that.

A limitation of my implementation is that the model family is still based on polynomial features and ridge. This is a limited way to capture nonlinear behavior. A natural next step is to test stronger nonlinear models, for example:

- Kernel ridge regression,

- Random Fourier Features,
- Small neural networks.

Then we can re-run the same ALMS methods and see if EAL still gives the same stability benefit when the model class is richer. That would also connect more directly to Math 385 topics about approximation, stability, and how model complexity affects generalization.

References

- Masashi Sugiyama and Neil Rubens. "Active Learning with Model Selection in Linear Regression." *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, pp. 518–529, 2008. DOI: 10.1137/1.9781611972788.47.
- Cam Nugent. *California Housing Prices*. Kaggle dataset. (Accessed 2025-12-17.)
- Michael Zhao. *Math 385 Final Project Code Repository*. GitHub: `git@github.com:1367565028/Math-385-F`
- H. S. Seung, M. Opper, and H. Sompolinsky. "Query by Committee." In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, 1992.
- Atsushi Shimizu, Xiaoou Cheng, Christopher Musco, and Jonathan Weare. "Improved Active Learning via Dependent Leverage Score Sampling." *International Conference on Learning Representations (ICLR)*, 2023.