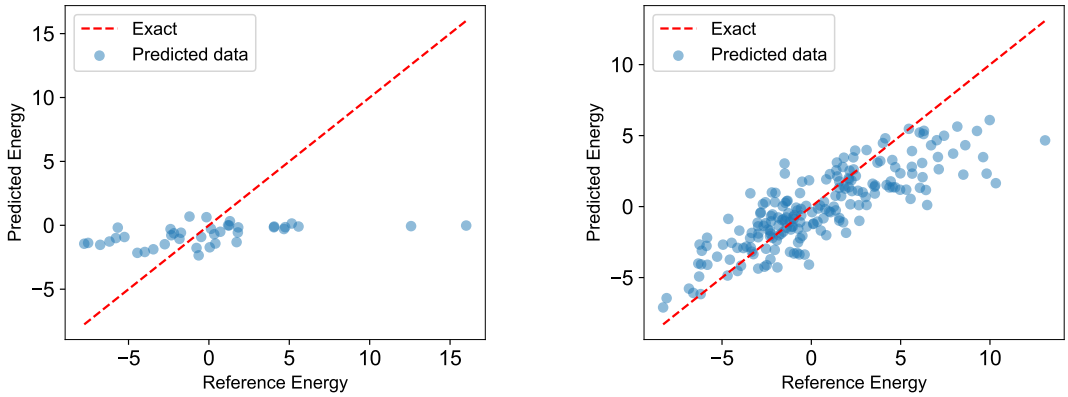
Lecture 11

Active Learning for Chemical Data Efficiency

Contents

We start this lecture by revisiting Lab 9-2, where we trained a Gaussian process regressor (GPR) to learn the potential energy surface (PES) of the ethanol molecule. At the end of that lab, we plotted the predicted energy distribution shown in Fig. 1a. As you may recall, the model only captured the mean of the y values rather than the true PES. I concluded the class by pointing out that the training had failed. Upon revisiting it, I realized the main issue: we only used 200 data points!



(a) GPR predictions with 200 data points.

(b) GPR predictions with 1000 data points.

Figure 1: GPR predictions of the energies of the ethanol molecule with different configurations. Energy shifted by the mean value.

What happens if we increase the number of training points to 1000? The result is shown in Fig. 1b. The prediction improves significantly. However, we still notice some bias in the sparse regions (where data points are few) while the dense regions are well captured.

Now, what if we keep adding more points only in the dense regions? Will this reduce the error in the sparse regions? Obviously not. Clearly, not all data points are equally informative for improving the model. This observation motivates a smarter approach: instead of adding points randomly, we want to *select the most informative points to label next.*

This is exactly the idea behind **active learning**: by choosing new data points strategically–based on model uncertainty or coverage of the input space–we can train models more efficiently and achieve good performance across the entire dataset with fewer expensive labels. This is the central focus of today's lecture.

1 Active Learning Workflow

1.1 A Gentle Introduction to ML Performance

Before diving into active learning, let's first review how we typically evaluate the performance of a machine learning (ML) model. Two key aspects are considered: (1) *prediction uncertainty (error)* and (2) *generalizability*. The performance of an ML model is primarily influenced by two factors: (1) *model complexity* and (2) *data quality*.

Model Complexity Model complexity determines how sophisticated a function the model can represent. Key factors include:

1. The type of model (e.g., simple linear regression, deep neural network such as a fully connected FNN).
2. The number of parameters.
3. The choice of hyperparameters.

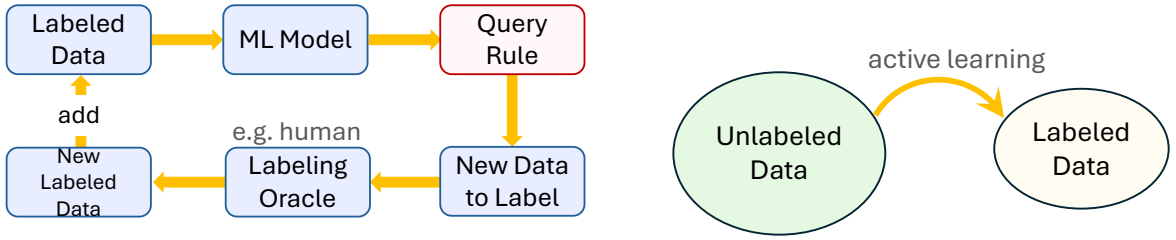
Data Quality Data quality depends on:

1. The amount of data.
2. The level of noise.
3. The choice of representation.
4. Whether the dataset spans a sufficiently diverse region of the input space.

There is no universally optimal ML model: an overly complex model can easily overfit a simple dataset, while an overly simple model may fail to capture latent patterns in a large or complex dataset. Achieving good performance therefore requires matching the model to the dataset.

By pairing the right model with the right data, we can improve **data efficiency**, which measures how effectively a model can learn from a given amount of data — or, more precisely, how much a model can learn *per unit of data*. Data efficiency depends on both model complexity and data quality.

Beyond selecting an appropriate model, data efficiency can also be improved iteratively by



(a) Active Learning Loop. (b) Choosing new data to label.

Figure 2: Active learning loop and labeling data.

selectively adding new data points to the training set. In *supervised learning* [1], where labeled data are used to learn the mapping between features and labels, this strategy is known as **active learning**.

The central idea of active learning is to choose the most **informative and representative** new samples to add to the training dataset, thereby maintaining or improving model performance while using as few labeled data points as possible. This is especially important when data acquisition is costly, a common situation in chemistry research. For instance, obtaining a single experimental data point may take days or weeks, and a quantum-level evaluation of a point on a potential energy surface (PES) can require significant computational time.

1.2 The workflow

A typical active learning loop is illustrated in Fig. 2a. Two pools of data are involved

- Labeled data.
- Unlabeled data.

Starting from an initial pool of labeled data, an ML model is trained. Based on the model's performance, a **query rule** is defined to select which new data points should be labeled and added to the training dataset. A labeling oracle—such as experiments or density functional theory (DFT) calculations—is used to provide the labels. Once the newly labeled data are added, the model is retrained, and the process repeats until the model performance is satisfactory or computational resources are exhausted. Here, we use the previously studied case of learning a PES as an example to illustrate this loop.

[1] For unsupervised learning, similar strategies exist to enhance data efficiency, such as active sampling.

Example: Active Learning Loop for Learning a PES

For the task of learning the potential energy surface (PES), **labeled data** consist of configurations with corresponding internal energies, while **unlabeled data** (the majority) are configurations whose energies are unknown.

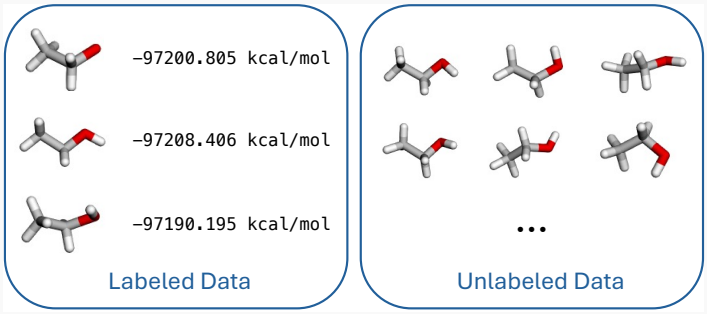


Figure 3: Labeled and unlabeled data for PES learning.

During the active learning loop, the labeled data are fed to an ML model, such as a feedforward neural network (FNN) or Gaussian process regression (GPR). The model's output, e.g., training error or predictive uncertainty, is then used to define a **query rule** (explained later).

Using the query rule, new configurations are selected from the pool of unlabeled data, and their energies are obtained through DFT calculations, converting them into labeled data.

Finally, the newly labeled data are added to the training dataset, and the model is retrained. This loop is repeated until the desired performance is achieved or computational resources are exhausted.

2 Query Strategies

The active learning loop itself is conceptually simple; the central challenge lies in defining an effective *query rule*. In this section, we introduce several commonly used query strategies, which can be broadly divided into two categories:

- **Uncertainty-based sampling**: selecting data points that are expected to be most informative for improving the model.
- **Diversity-based sampling**: selecting data points that improve the representativeness and coverage of the dataset.

2.1 Uncertainty sampling

The idea of uncertainty sampling is to query new data points where the current model is *least confident*, because labeling these points is expected to reduce model error most efficiently.

Formally, the next data point to label is selected based on a model-defined **uncertainty** measure

$\mathcal{U}(\mathbf{x})$, where $\mathbf{x}$ denotes a data point. The selection rule is

$$\mathbf{x}_{\text{next}} = \arg\max_{\mathbf{x}} \mathcal{U}(\mathbf{x}). \tag{1}$$

The specific definition of $\mathcal{U}(\mathbf{x})$ depends on the learning task and the model class. In the following, we review some commonly used definitions of $\mathcal{U}(\mathbf{x})$ for the methods we have encountered.

2.1.1 Gaussian Process Regression (GPR)

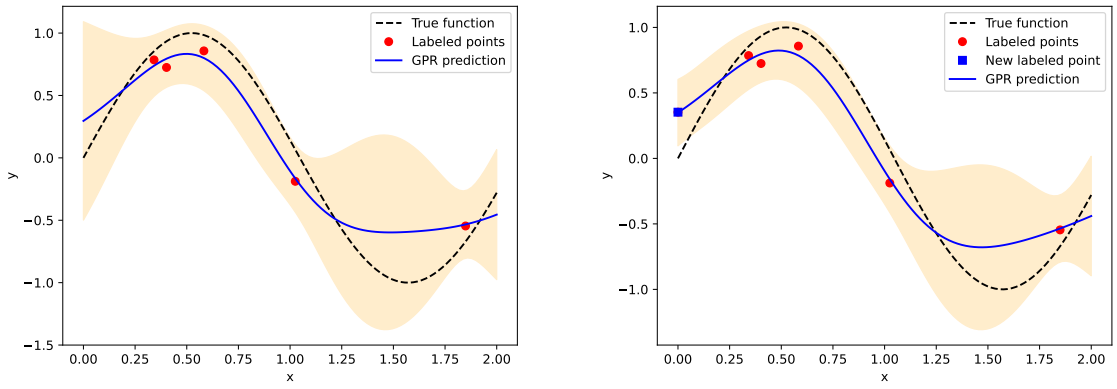
For Gaussian process regression, the uncertainty measure is intrinsic to the model. The label is treated as a conditional distribution

$$p(y \mid \mathbf{x}) \sim \mathcal{N}\big(\mu(\mathbf{x}), \sigma^2(\mathbf{x})\big), \tag{2}$$

where $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ are the predictive mean and variance, respectively. The predictive variance $\sigma^2(\mathbf{x})$ is therefore taken as the uncertainty measure.

With GPR, uncertainty sampling reduces to selecting the point where $\sigma^2(\mathbf{x})$ is maximized. This strategy is often referred to as *expected information gain*.

An example of using the variance as the uncertainty measure to choose the next data point to label is shown in Fig. 4.



(a) GPR predictions before adding a point from active learning.

(b) GPR predictions after adding a point from active learning.

Figure 4: Active learning to select the next point for GPR predictions.

2.1.2 Neural-Network-based Regression

Unlike GPR, where predictive uncertainty is intrinsic, neural networks require additional strategies to estimate uncertainty. Several approaches exist for evaluating the uncertainty of a model's prediction on a given data point.

Query-by-Committee.

In this approach, we train M different neural network models on the same dataset. Given a new unlabeled data point $\mathbf{x}$, each model produces a prediction $\hat{y}_m(\mathbf{x})$. The uncertainty at $\mathbf{x}$ is then evaluated by the variance of the M predictions:

$$\sigma^2(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\left(\hat{y}_m(\mathbf{x}) - \bar{y}_M(\mathbf{x})\right)^2, \tag{3}$$

where

$$\bar{y}_M(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\hat{y}_m(\mathbf{x})$$

is the average prediction over the committee. An illustration is shown in Fig. 5.

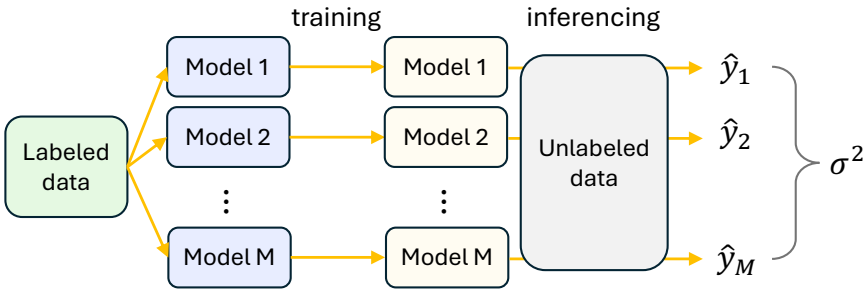


Figure 5: Query-by-committee.

A common question is how the uncertainty from multiple models can represent the uncertainty of a single model. In most active learning setups, we assume that **information is universal**: a data point that is poorly represented in the dataset should exhibit high uncertainty, independent of the specific model.

The main drawback of query-by-committee is its computational cost, since multiple models must be trained.

Monte Carlo Dropout (MC dropout).

The MC dropout strategy is conceptually similar to the query-by-committee approach. However, instead of training multiple models, we apply **neuron dropout** to a single trained neural network

to generate M stochastic realizations ("imperfect shadows") of the model. The term *Monte Carlo* refers to the random sampling process.

To create an imperfect shadow, after training the model, a subset of neurons is randomly deactivated by setting all weights associated with those neurons to zero. This procedure is illustrated in Fig. 6.

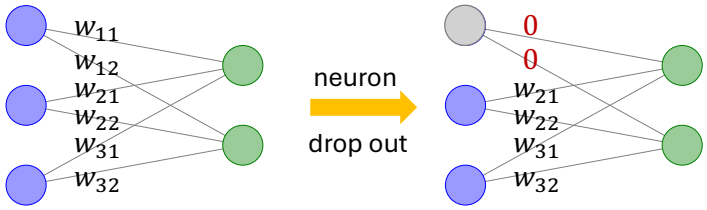


Figure 6: A neuron in the first layer is dropped out by setting all associated weights to zero.

Each stochastic forward pass produces an output that is slightly different from the original model. Points for which the model is least confident are affected the most. The uncertainty of a data point $\mathbf{x}$ is then estimated by the variance of the M outputs:

$$\sigma^2(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{y}_m(\mathbf{x}) - \bar{y}_M(\mathbf{x})\right)^2 , \tag{4}$$

where

$$\bar{y}_M(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m(\mathbf{x})$$

is the average prediction over the M stochastic passes.

If observational noise is present, it can be added to this variance to obtain the total predictive uncertainty.

Thus, both MC dropout and query-by-committee share the same formalism for uncertainty estimation. The key advantage of MC dropout is that only a single model needs to be trained, after which multiple stochastic realizations are obtained by randomly dropping neurons.

2.1.3 Classification

As in supervised learning, regression and classification require different loss functions (e.g., mean squared error for regression, cross-entropy for classification). Correspondingly, different uncertainty metrics are used for classification tasks.

Neural networks for classification typically output class probabilities $p(y \mid \mathbf{x})$ via a softmax or sigmoid activation.

- **Query-by-Committee.** Since each model in the committee is trained independently, we can use the predicted class labels to compute uncertainty. Let v_k denote the number of models that assign $\mathbf{x}$ to class k. The uncertainty is then quantified using *vote entropy*:

$$\mathcal{U}(\mathbf{x}) = -\sum_k \frac{v_k}{M} \log \frac{v_k}{M}. \tag{5}$$

- **MC Dropout.** Each stochastic forward pass of the same network produces a probability vector $p_m(y \mid \mathbf{x})$. The mean predictive probability is

$$\bar{p}_k = \bar{p}(y = k \mid \mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} p_m(y = k \mid \mathbf{x}), \tag{6}$$

and the predictive uncertainty is then quantified by the entropy of this averaged distribution:

$$\mathcal{U}(\mathbf{x}) = -\sum_k \bar{p}_k \log \bar{p}_k. \tag{7}$$

In classification, these entropy-based measures serve as the analog of variance in regression, capturing the model's uncertainty over the predicted class probabilities.

2.2 Diversity Sampling

While uncertainty sampling focuses on querying points about which the model is least confident, it can sometimes select points that are very similar to each other. This occurs because the model is intrinsically uncertain in that region, which can lead to redundant data and inefficient use of labeling resources.

Diversity sampling takes a complementary approach: instead of focusing on model uncertainty, it focuses on the data themselves, explicitly encouraging the selection of points that are *representative of the input space* or *maximally different* from previously labeled points.

Formally, diversity-based sampling selects the next data points $\mathbf{x}$ by optimizing a **diversity measure** $\mathcal{D}(\mathbf{x})$:

$$\mathbf{x}_{\text{next}} = \arg\max_{\mathbf{x}} \mathcal{D}(\mathbf{x}). \tag{8}$$

In the following, we discuss several commonly used formalisms for $\mathcal{D}(\mathbf{x})$.

2.2.1 Max-Min Sampling

Max-min sampling selects points that are as far away as possible from the data we already have.

For each unlabeled point $\mathbf{x}_u$, we compute its distance to the closest labeled point $\mathbf{x}_l$, denoted as $s(\mathbf{x}_u)$:

$$\mathcal{D}(\mathbf{x}_u) = s(\mathbf{x}_u) = \min_{x_l} \text{dist}(\mathbf{x}_u, \mathbf{x}_l). \tag{9}$$

The next data point is then the unlabeled point with the largest $\mathcal{D}(\mathbf{x}_u)$.

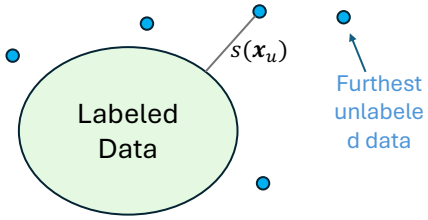


Figure 7: Max-Min Sampling.

For Gaussian process models, the kernel function can be used as a distance metric instead of Euclidean distance.

2.2.2 Clustering-based Sampling

Clustering-based sampling groups data points into clusters and selects representative points from each cluster. We first introduce a common clustering method: **K-means clustering**.

K-means Clustering The algorithm iteratively groups data points around K centroids:

1. Randomly select K points as initial centroids.
2. Assign each data point to the cluster with the closest centroid.
3. Recompute the centroid of each cluster as the mean of its points.
4. Repeat steps 2–3 until the centroids converge.

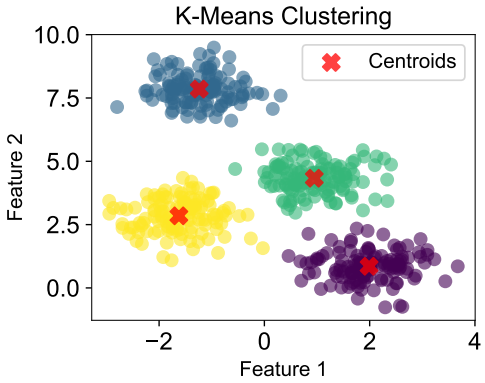


Figure 8: Illustration of K-means clustering with $K = 4$.

Clustering-based Sampling In cluster-based sampling, the points closest to the cluster centroids are considered most representative. Therefore, we first perform clustering (e.g., K-means) on the unlabeled pool and then select the points nearest the cluster centers.

2.3 Combining Diversity with Uncertainty

In practice, the most effective active learning strategies often combine uncertainty and diversity. One simple approach is to rank candidate points by uncertainty and then select a diverse subset from the top-ranked points. This hybrid strategy ensures that selected points are both

- informative (high uncertainty);
- representative (high diversity).

3 Lab

In order to focus on the concepts of active learning, we will use toy dataset for this lab. In the first lab, we use active learning to improve Gaussian process regressor (GPR) results. An illustration sample code is also included to produce GPR results.

In the second lab, we build a FNN and use MC dropout combined with Max-Min samplign to improve the FNN performance

Lab 11-1:

`https://colab.research.google.com/drive/1l8BmJYitOAulumcEMAILy6XcDrpOcG49?usp=sharing`

Lab 11-2:

`https://colab.research.google.com/drive/13yTiv8KAqrZHoG26Qyrq0d8WvC3wWAep?usp=sharing`