Applications of Bayesian Machine Learning

Advanced Statistical Inference

Simone Rossi

Overview

- Bayesian ML is not just about predictions, it is about **decisions under uncertainty**.
- We focus on 3 application patterns:
  1. Uncertainty decomposition (entropy-based)
  2. Bayesian continual learning
  3. Bayesian active learning
- Goal: connect probabilistic modeling to practical choices.
- Emphasis on intuition, equations, and what to do with uncertainty.
- Examples are mostly in classification, where entropy is most interpretable.

Uncertainty decomposition

Why uncertainty matters

- A model can be accurate on average and still fail on critical cases.
- We need to know when to defer, ask for labels, or explore.
- Bayesian models provide **predictive distributions**, not just point estimates.
- Uncertainty is a decision signal, not just a diagnostic.
- Many real systems must trade off accuracy, risk, and cost.

Bayes rule reminder

Given training data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, the posterior is

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})}$$

- The posterior summarizes what we have learned from data.
- Uncertainty about $\boldsymbol{\theta}$ flows to uncertainty in predictions.
- The evidence $p(\mathcal{D})$ normalizes and enables model comparison.

$$\log p(\boldsymbol{\theta} \mid \mathcal{D}) = \log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log p(\mathcal{D})$$

> **Data symbol**
>
> Before, we expressed the posterior in terms of $\boldsymbol{X}$ and $\boldsymbol{y}$. For brevity, we now use $\mathcal{D}$ to denote the full dataset. No change in meaning, just a shorthand.

Predictive distribution

Given training data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, the predictive distribution is

$$p(y_\star \mid \boldsymbol{x}_\star, \mathcal{D}) = \int p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta}$$

- The prediction is a **distribution**, not a point estimate.
- Classification: categorical distribution over classes.
- Regression: predictive density over $y_\star$.
- Plug-in predictions use a single $\boldsymbol{\theta}$; Bayesian predictions integrate uncertainty.

$$p(y_\star \mid \boldsymbol{x}_\star, \mathcal{D}) = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{\theta})\right]$$

Decision-making under uncertainty

- Use the predictive distribution to make decisions.
- Examples:
  - Classification: pick most probable class?
  - Regression: use predictive mean?
- Risk-sensitive decisions: consider uncertainty?

The predictive distribution $p(y_\star \mid \boldsymbol{x}_\star, \mathcal{D})$ encodes all we know about $y_\star$ given data, but there are different types of uncertainty to consider.

Epistemic vs Aleatoric

Epistemic (model)

- Uncertainty about parameters $\boldsymbol{\theta}$
- High when data are scarce or out-of-distribution
- Reducible with more data
- Depends on the model class and prior assumptions
- Shrinks as coverage of the input space improves

Aleatoric (data)

- Noise in the data-generating process (e.g., fog in images, syntax ambiguity in text) or in the data acquisition pipeline (e.g., sensor noise)
- Present even with infinite data
- Often input-dependent (heteroscedastic)
- Cannot be reduced by more data, only modeled
- Can be homoscedastic (constant) or heteroscedastic (varying)

A quick intuition

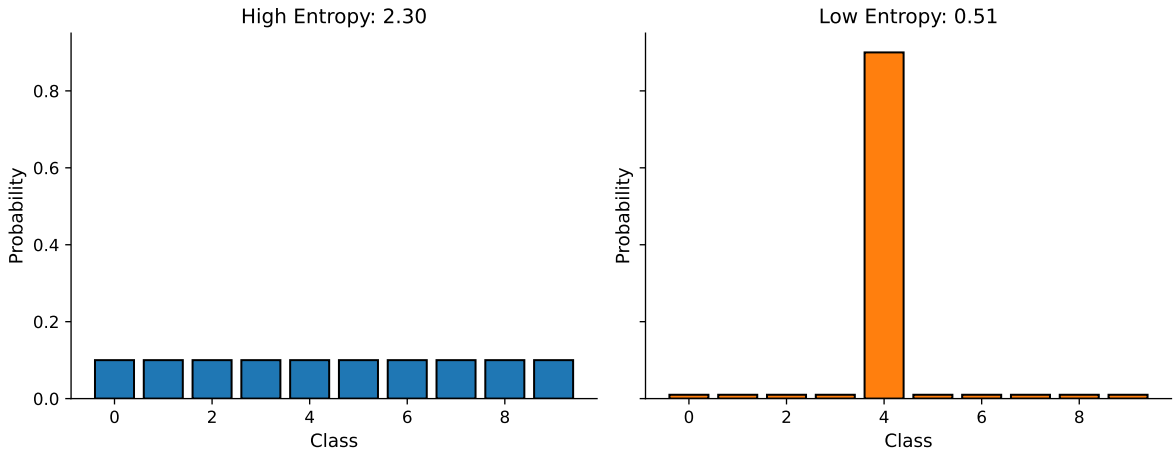
- **Aleatoric**: the input itself is ambiguous (e.g., blurry image).
- **Epistemic**: the input is unfamiliar (e.g., out-of-distribution).
- Both high means the model should abstain or request help.
- High epistemic suggests collecting new data in that region.
- High aleatoric suggests robust decisions rather than more data.

Shannon entropy as uncertainty (classification)

For K classes, predictive entropy is

$$H[y \mid \boldsymbol{x}, \mathcal{D}] = -\sum_{k=1}^{K} p(y = k \mid \boldsymbol{x}, \mathcal{D}) \log p(y = k \mid \boldsymbol{x}, \mathcal{D})$$

- Use $\log_2$ for bits or log for nats.
- Entropy is sensitive to how spread the predictive distribution is.
  - $H = 0$ when the model is fully confident.
  - H is maximal when the distribution is uniform.



Entropy decomposition

For Bayesian classification, predictive entropy decomposes as

$$H[y \mid x, \mathcal{D}] = \mathbb{E}_{p(\theta \mid \mathcal{D})}\left[H[y \mid x, \theta]\right] + I(y, \theta \mid x, \mathcal{D})$$

- **Aleatoric**: expected conditional entropy (left).
- **Epistemic**: mutual information (right).
- Mutual information is zero when all plausible models agree.
- This is the core split used in Bayesian deep learning.

> Key idea
>
> The total uncertainty splits into data noise plus model uncertainty.

Mutual information and epistemic uncertainty

The epistemic component is

$$I(y, \theta \mid x, \mathcal{D}) = H[y \mid x, \mathcal{D}] - \mathbb{E}_{p(\theta \mid \mathcal{D})}\left[H[y \mid x, \theta]\right]$$

In practice:

- "Entropy of the average" minus "average of the entropy".

Two interpretations:

- How much knowing y at $\boldsymbol{x}$ would reduce uncertainty about $\boldsymbol{\theta}$?
  - High if the label would help identify the correct model.
- How much the models disagree on the prediction at $\boldsymbol{x}$?
  - High if different samples $\boldsymbol{\theta}$ give different class predictions

Monte Carlo estimation

Sample $\boldsymbol{\theta}^{(s)} \sim p(\boldsymbol{\theta} \mid \mathcal{D})$ for $s = 1, \ldots, S$:

$$\hat{p}(y = k \mid \boldsymbol{x}, \mathcal{D}) = \frac{1}{S} \sum_{s=1}^{S} p(y = k \mid \boldsymbol{x}, \boldsymbol{\theta}^{(s)})$$

$$\widehat{H}_{\text{pred}} = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k, \quad \widehat{H}_{\text{exp}} = \frac{1}{S} \sum_{s=1}^{S} H[y \mid \boldsymbol{x}, \boldsymbol{\theta}^{(s)}]$$

$$\widehat{I} = \widehat{H}_{\text{pred}} - \widehat{H}_{\text{exp}}$$

- S is the number of posterior samples.
- In practice, S between 10 and 50 is common for BNNs.
- Use log-sum-exp for numerical stability when needed.
- When the true posterior is unknown, use approximation $q(\boldsymbol{\theta})$.

- Explore how epistemic and aleatoric components change with data.
- Notice how epistemic uncertainty shrinks near observed regions.

Interpreting the numbers

- High aleatoric, low epistemic: inherently noisy input.
- High epistemic: data are missing or the input is unfamiliar.
- High both: ambiguous and unfamiliar; safest to defer.
- Low both: confident and reliable predictions.
- Actions differ: collect data for epistemic, redesign sensing for aleatoric.

Why the decomposition matters

- **Calibration**: avoid overconfident mistakes.
- **OOD detection**: flag high epistemic points.
- **Risk-aware decisions**: abstain or lower the stakes.
- **Active learning**: query points that reduce epistemic uncertainty.
- **Model debugging**: identify where uncertainty is poorly captured.
- **Resource allocation**: focus labeling where it matters most.

Bayesian continual learning

What is continual learning?

- Learn from a **sequence of tasks or data batches**.
- Update the model as new data arrive.
- Keep good performance on earlier data without full retraining.
- Data may not be i.i.d. across time.
- The goal is to learn continuously under limited memory.

A motivating example

- A personal assistant learns user preferences over months.
- A robot adapts to new environments every day.
- We want improvement over time without forgetting past skills.
- In practice, data arrive in bursts and the world keeps changing.
- We need models that can adapt without expensive retraining cycles.

Online vs batch learning

- **Batch learning**: train once on a fixed dataset (this is the classic ML setup).
- **Online learning**: update as data arrive over time.
- Continual learning sits between the two: update without full retraining.
- The order of data matters and can bias what is remembered.
- Continual learning focuses on preserving useful knowledge.

Why not just retrain from scratch?

- Data may be too large or private to store.
- We need fast updates for non-stationary environments.
- Memory and compute are limited in many real systems.
- Retraining can be too slow for real-time applications.
- Storing all historical data can violate privacy constraints.

Catastrophic forgetting

Continual learning has a key challenge: **catastrophic forgetting**.

When training only on the current task, the model can **forget** earlier ones.

- New gradients overwrite parameters used for older tasks.
- Performance on past tasks can drop sharply.
- This is a stability-plasticity dilemma.
- The challenge is to keep old skills while learning new ones.

Task settings

Common continual learning scenarios:

- **Task-incremental**: task ID is known at test time.
- **Class-incremental**: new classes appear, no task ID.
- **Domain-incremental**: same labels, data distribution shifts.

Comments

- Class-incremental is typically the most difficult setting, it's also know as open-world learning.
  - Example: learning to recognize traffic signs in different countries over time.
- Domain-incremental focuses on adaptation to distribution shifts without changing the task itself.
  - Example: adapting an autonomous vehicle's perception system to different weather conditions.

Bayesian approach to continual learning

Bayesian inference provides a natural framework for such updates.

> **!** Bayesian update
>
> Remember that Bayes' rule is just an update from one distribution (we call it prior) to another (posterior) using data. And this can be done sequentially.

After observing data from tasks 1 to T, the posterior is

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T}) \propto p(\mathcal{D}_T \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T-1})$$

where $\mathcal{D}_{1:T}$ denotes all data up to task T, and $\mathcal{D}_T$ is the data **only** from task T.

In practice:

- Posterior after task $T-1$ becomes the prior for task T.
- This is **exact**, as long as we can compute the posterior.
- We only need the previous posterior, not all past data.

Posterior as memory

- The posterior summarizes what we have learned so far.
- In continual learning, **memory is stored in the posterior**, without keeping data.
- The update balances **stability** (keep old knowledge) and **plasticity** (learn new data).
- A good posterior is a compact summary of past experience.

Predictive distribution over time

$$p(y_\star \mid \boldsymbol{x}_\star, \mathcal{D}_{1:T}) = \int p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T})\, d\boldsymbol{\theta}$$

- As T grows, the posterior evolves and so do predictions.
- Concept drift appears as changes in the posterior.
- We want predictions that adapt without collapsing.

Challenge:

- Exact $p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T})$ is intractable for neural networks but we can again use **approximate inference** (VI, Laplace, etc.).

Online VI objective

Use the same Variational Inference objective, but now sequentially:

$$q_T = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\theta}) \| p(\boldsymbol{\theta} \mid \mathcal{D}_{1:T})\right)$$

The ELBO becomes

$$\mathcal{L}_T(q) = \mathbb{E}_q[\log p(\mathcal{D}_T \mid \boldsymbol{\theta})] - \mathrm{KL}\left(q(\boldsymbol{\theta}) \| q_{T-1}(\boldsymbol{\theta})\right)$$

- The likelihood term fits **only** new data.
- The prior is now the previous posterior approximation and the KL term keeps the new posterior close to the previous one.

Bayesian active learning

What is active learning?

Traditional supervised learning

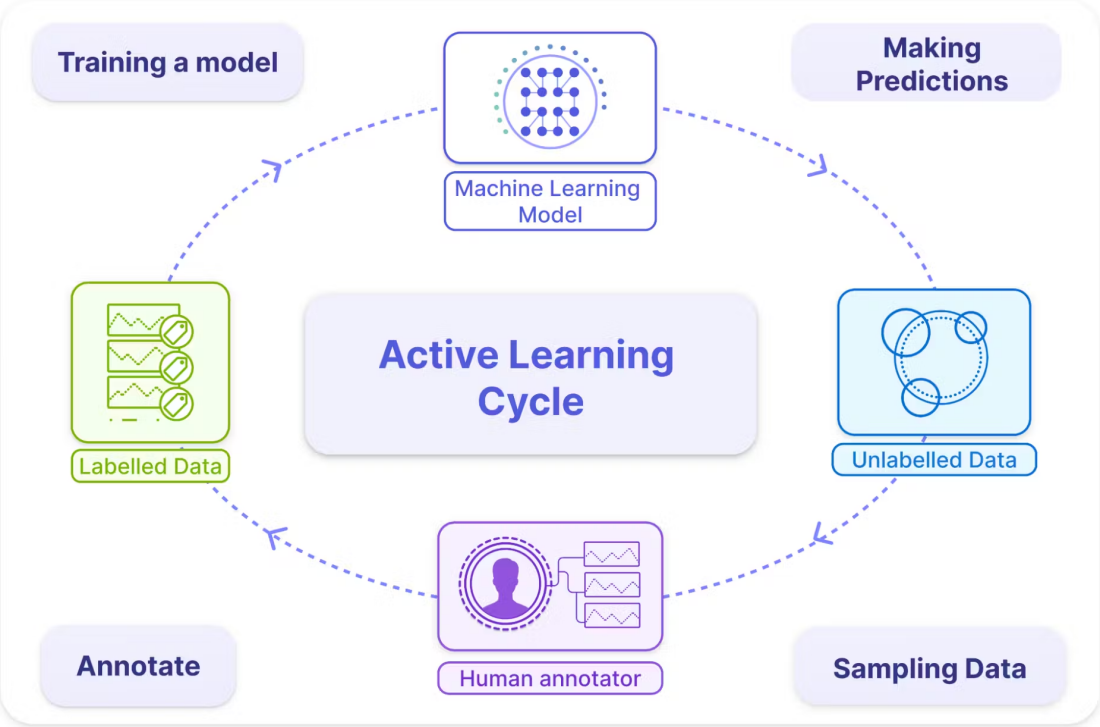
- Collect input features $\boldsymbol{X}$, and manually (or automatically) label them $\boldsymbol{y}$.
- Train a model on the labeled data.

Active learning

- Labels are expensive, but unlabeled data are abundant.
- Collect $\boldsymbol{X}$ first, then **choose which points to label**, based on the model performance.
- Interpret model training as a sequential decision process.

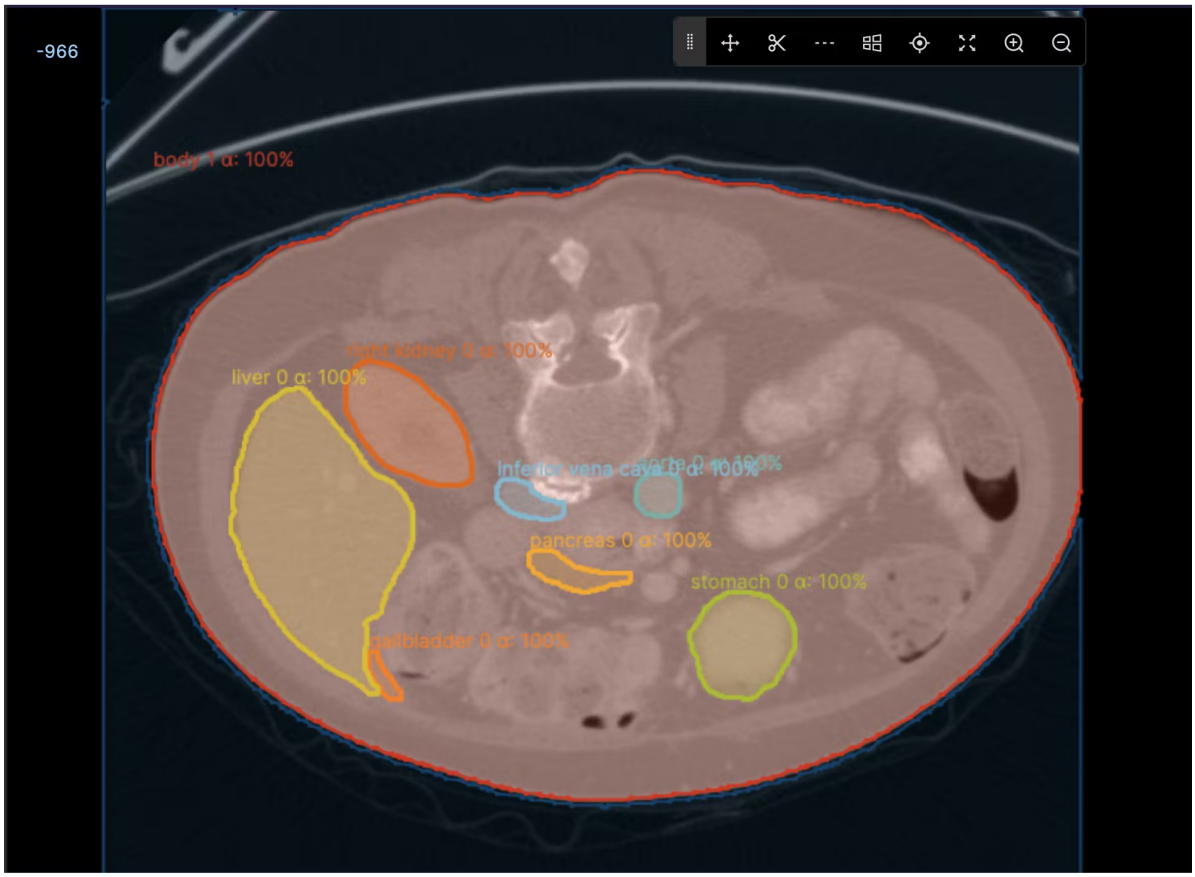
Goal: maximize performance per label cost, by **"smartly" choosing which data to label and to learn from**.

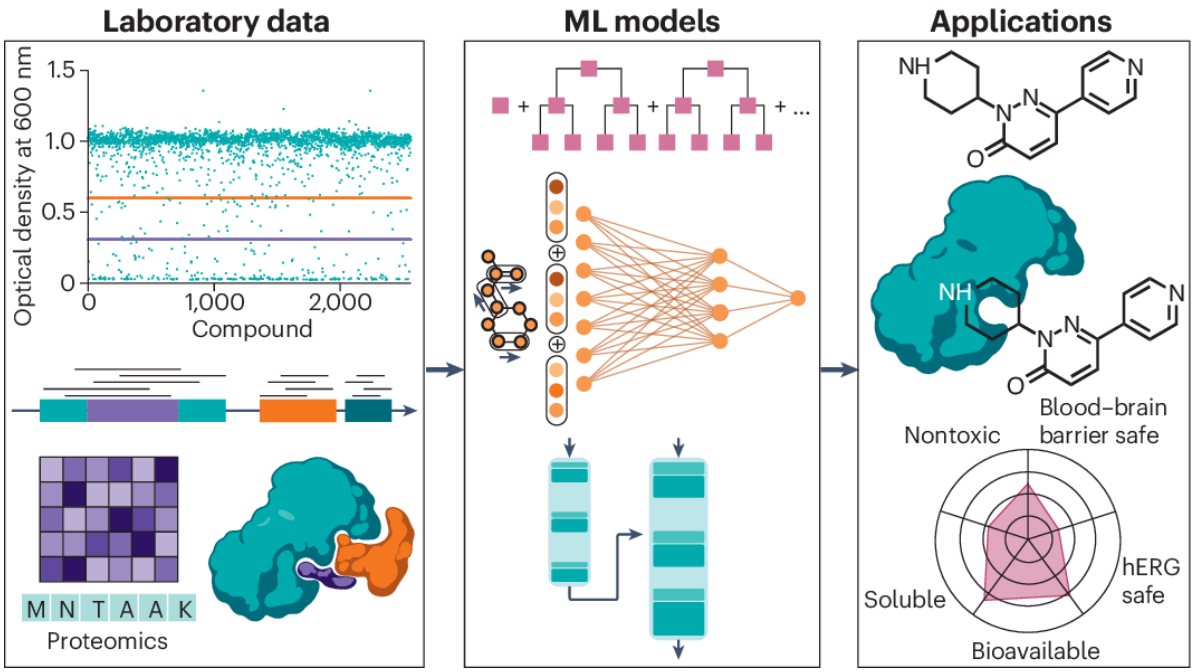
The active learning loop



When is it useful?

- Medical labeling, robotics, scientific experiments (e.g., drug discovery).
- Annotation budgets, slow human experts, or costly sensors.
- Gains are largest when labels are costly and models are uncertain.


-966
body 1 α: 100%
liver 0 α: 100%
pancreas 0 α: 100%
stomach 0 α: 100%



Notation

- Labeled set: $\mathcal{D}_L = \{(x_i, y_i)\}_{i=1}^{N_L}$
- Unlabeled pool: $\mathcal{U} = \{x_j\}_{j=1}^{N_U}$
- Goal: pick a small batch from $\mathcal{U}$ to label next.
- A labeling budget B limits the number of queries.

Active learning loop

Typical pool-based loop:

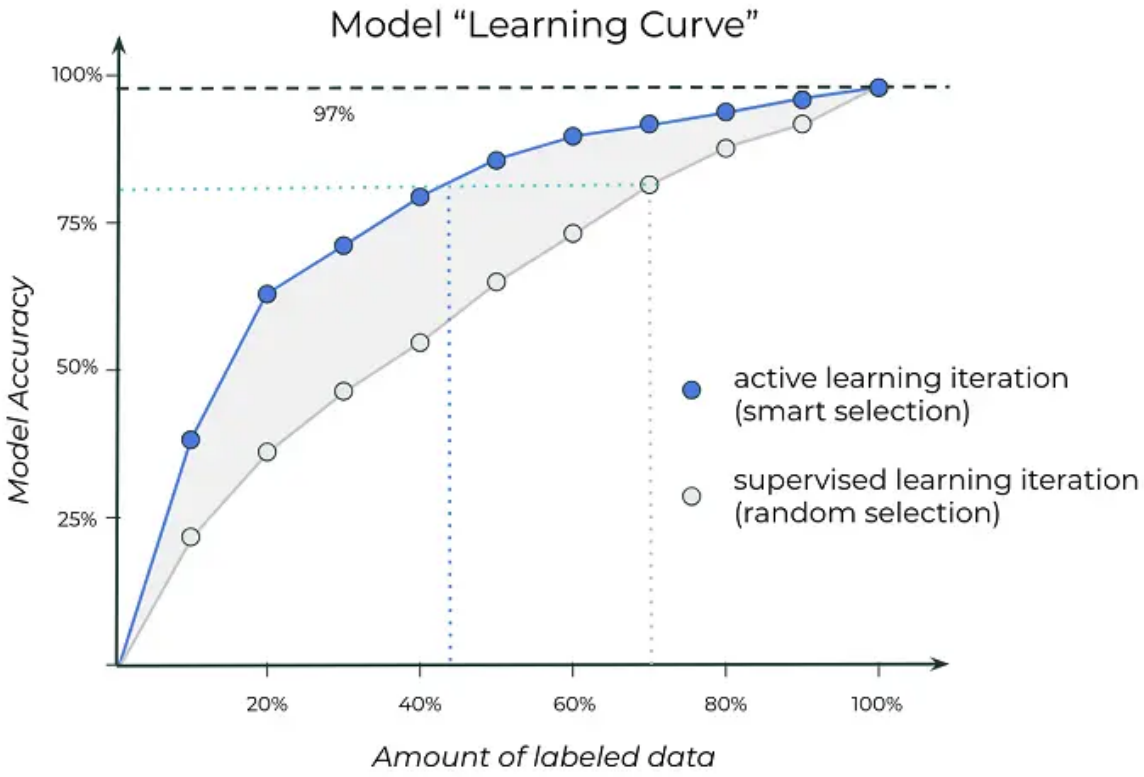
1. Train on labeled set $\mathcal{D}_L$.
2. Score unlabeled pool $\mathcal{U}$ with an acquisition function.
3. Label top points and update the posterior.

- Repeat until the budget is exhausted or performance saturates.
- Retraining can be full or incremental.

Acquisition functions

How to pick which points to label?

1. Randomly (not recommended, but a useful baseline).
2. Use an acquisition function $a(x)$ that scores usefulness of labeling x.



Intuition: Let's use uncertainty to guide labeling!

General acquisition framework

Pick $x_\star$ that maximizes expected utility:

$$x_\star = \arg\max_{x \in \mathcal{U}} a(x)$$

Common families:

- **Uncertainty sampling**: query most uncertain points.
- **Expected error reduction**: reduce future risk.
- **Information gain**: learn most about parameters.
- **Expected improvement**: optimization-focused settings.
- Each criterion encodes a different notion of useful data.

Active learning as information gain

$$a(\boldsymbol{x}) = I(y, \boldsymbol{\theta} \mid \boldsymbol{x}, \mathcal{D}) = H[\boldsymbol{\theta} \mid \mathcal{D}] - \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}[H[\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{x}, y]]$$

- Measures how much labeling $\boldsymbol{x}$ gains information about $\boldsymbol{\theta}$.
- Points that increase our knowledge of the model are preferred.

. . .

But this is the **EPISTEMIC UNCERTAINTY**!

- We want to finds points with high epistemic uncertainty to label. - Labeling such points will reduce uncertainty about the model.
- This connects active learning with Bayesian experimental design.

Why epistemic and not aleatoric?

- If a data point is noisy, it stays uncertain forever, no matter how many times we label it.
- Labeling such points may not improve the model.
- We prefer points that reduce **epistemic** uncertainty.

Image with Low Aleatoric Uncertainty

Image with High Aleatoric Uncertainty

BALD criterion

This acquisition function is called **Bayesian Active Learning by Disagreement (BALD)**.

$$a_{\text{BALD}}(\boldsymbol{x}) = H[y \mid \boldsymbol{x}, \mathcal{D}] - \mathbb{E}_{p(\theta \mid \mathcal{D})}\left[H[y \mid \boldsymbol{x}, \theta]\right]$$

How to estimate it in practice?

- Use a Bayesian inference method to train your neural network.
- Sample from the (approximate) posterior to estimate predictive entropy and expected entropy as before.

Note:

- This is one of the most popular acquisition functions, grounded in solid Bayesian principles.
- You can still do active learning without Bayesian models, but having uncertainty estimates helps a lot!