PML session 14. Decision making under uncertainty

Probabilistic Machine Learning Reading Group

J.M. Camacho

May 13, 2026

Institute of Mathematical Sciences (ICMAT-CSIC)

Contents

- Statistical decision theory (SDS)
- Influence diagrams
- A/B testing
- Contextual bandits
- Markov decision processes
- Active learning

Contents

Basics

Bayesian inference: updates our beliefs about hidden quantities after observing data $\boldsymbol{X} = \boldsymbol{x}$.

Decision theory: turns these updated beliefs into actions.

Basics of Statistical Decision Theory

A decision maker observes data $\mathbf{x}$ and chooses an action $a \in \mathcal{A}$.

- $h \in \mathcal{H}$: unknown **state of nature**
- $\ell(h, a)$: loss of taking action a when the true state is h
- $a = \delta(\mathbf{x})$: policy or decision procedure

Goal: choose the policy that minimizes expected loss, or **risk**.

$$\delta^* = \arg\min_{\delta} R(\delta) = \arg\min_{\delta} \mathbb{E}[\ell(h, \delta(X))]$$

The risk $R(\delta)$ can be defined from either a frequentist or Bayesian perspective.

Frequentist vs. Bayesian Approach

Frequentist view:

- Treats the state of nature h as fixed but unknown.
- Evaluates a policy by its risk over repeated datasets.

Bayesian view:

- Treats the observed data x as fixed and h as uncertain.
- Uses the posterior $p(h \mid x)$ to compute the posterior expected loss.
- Chooses the action that minimizes this posterior risk.

The **Bayesian approach** is **constructive**: it tells us how to build the optimal policy for a given dataset.

Bayesian Decision Theory

In the Bayesian view, the observed data x is fixed, while the state of nature h is unknown.

Using a prior $\pi(h)$, the posterior is

$$p_\pi(h \mid x) \propto \pi(h)\, p(x \mid h).$$

The **posterior risk** of the policy $a = \delta(x)$ is

$$\rho_\pi(\delta \mid x) = \mathbb{E}_{p_\pi(h|x)}\left[\ell(h, \delta(x))\right] = \int \ell(h, \delta(x))\, p_\pi(h \mid x)\, dh.$$

The optimal policy minimizes the posterior risk:

$$\delta^*(x) = \arg\min_\delta \rho_\pi(\delta \mid x).$$

Single Decision-Making Problems in ML

Classification

The state of nature and the action both correspond to class labels:

$$h = y^* \in \mathcal{Y}, \qquad a = \hat{y} \in \mathcal{Y}.$$

Using zero-one loss,

$$\ell_{01}(y^*, \hat{y}) = \mathbb{I}(y^* \neq \hat{y}).$$

The posterior risk is

$$\rho(\hat{y} \mid x) = P(y^* \neq \hat{y} \mid x) = 1 - p(y^* = \hat{y} \mid x).$$

The optimal policy chooses the most probable label:

$$\delta^*(x) = \arg\max_{y \in \mathcal{Y}} p(y \mid x).$$

Single Decision-Making Problems in ML

Regression

- State of nature: $h \in \mathbb{R}$
- Action: $y \in \mathbb{R}$
- Squared-error loss:

$$\ell_2(h, y) = (h - y)^2$$

The posterior risk is

$$\rho(y \mid x) = \mathbb{E}\left[(h - y)^2 \mid x\right].$$

The optimal action is the posterior mean:

$$\delta^*(x) = \mathbb{E}[h \mid x] = \int h\, p(h \mid x)\, dh.$$

Single Decision-Making Problems in ML

Other examples of single-decision problems in ML include:

- **Parameter estimation**
- **Structured prediction**
- **Fairness-aware decisions**

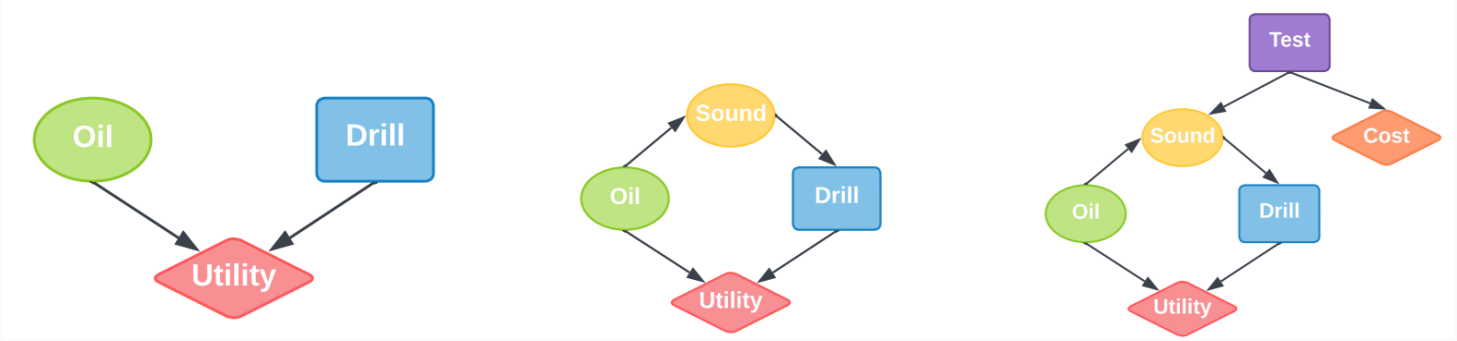
Contents

Influence (Decision) Diagrams

Influence diagrams represent structured, multi-stage decision problems.

They extend probabilistic graphical models with decision and utility nodes.

- **Ovals:** chance nodes, representing random variables.
- **Rectangles:** decision nodes, representing actions.
- **Diamonds:** utility nodes, representing rewards or costs.



Computing the Optimal Policy

Given an influence diagram, the optimal policy can be computed by modifying the variable elimination algorithm (Section 9.5).

Main intuition:

- Work backwards through the decision sequence:

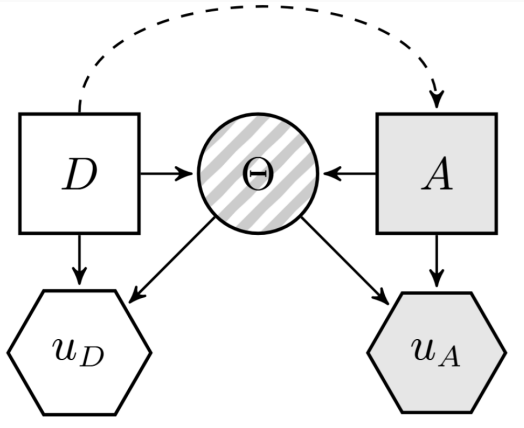
$$D_n, D_{n-1}, \ldots, D_1.$$

- At each step, choose the optimal decision d_i^*, assuming all future decisions are chosen optimally.

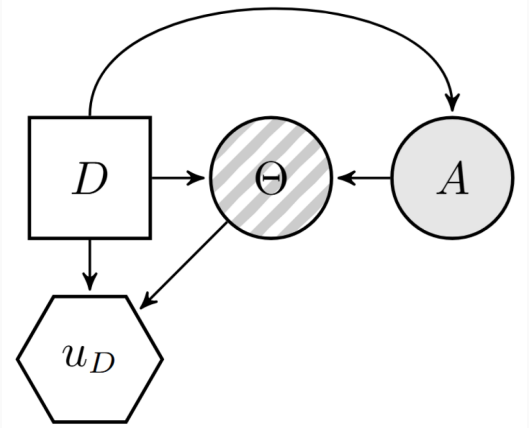
Extension to Several Agents

Influence diagrams can be extended to problems with multiple decision makers.

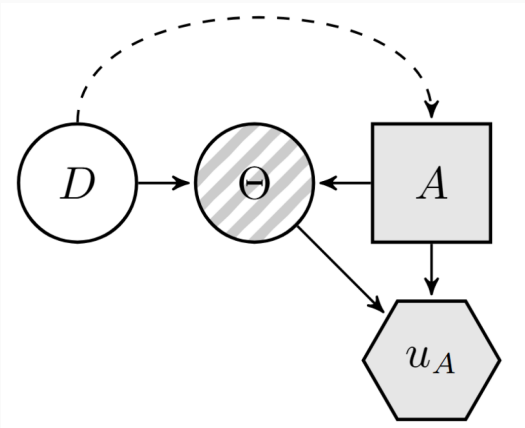
Bi-agent influence diagrams (BAIDs) represent interactions between two agents, such as a defender and an attacker.



Bi-agent influence diagram



Defender's decision problem



Attacker's decision problem

Useful for modelling competitive or adversarial settings, such as **Adversarial Risk Analysis (ARA)**.

Contents

Motivation

We often want to compare two alternatives and decide which one performs better.

- Which product version is likely to sell more?
- Which webpage design leads to more clicks?
- Which drug or treatment is more effective?

In an A/B test:

$$A \equiv \text{control}, \qquad B \equiv \text{treatment}.$$

The alternatives are sometimes called **arms**.

Test-and-Roll Approach

Test phase

1. Randomly assign users to different actions.
2. Measure the accumulated reward for each action.
3. we use a decision rule to select the action that appears best.

Roll phase

4. Deploy the selected action to the remaining population.

Key considerations in A/B Testing

A/B testing involves two main decisions:

1. **Decision rule:** how to choose the better action from noisy test results.
2. **Sample allocation:** how many users to assign to treatment n_1 and control n_0.

Exploration–exploitation trade-off : larger test sizes give more reliable estimates, but reduce the population left for deploying the best action.

Bayesian A/B Testing

The chapter presents a Bayesian decision-theoretic approach to choosing between two versions of a product, webpage, or treatment.

We compare two actions: control $(j = 0)$ and treatment $(j = 1)$.

$$y_j = (y_{1j}, \ldots, y_{n_j j})$$

where y_j is the vector of observed rewards collected from action j during the test phase.

- μ_j: expected reward of action j
- m_j: prior mean reward of action j
- n_j: test sample size for action j

Optimal policy in A/B testing

Optimal policy: choose the action with the larger posterior expected reward:

$$\pi^*(y_1, y_0) = \begin{cases} 1, & \mathbb{E}[\mu_1 \mid y_1] \geq \mathbb{E}[\mu_0 \mid y_0], \\ 0, & \mathbb{E}[\mu_1 \mid y_1] < \mathbb{E}[\mu_0 \mid y_0]. \end{cases}$$

Optimal Sample Size in A/B Testing

The test sizes n_0 and n_1 are chosen to maximize the expected total reward:

$$\mathbb{E}[R] = \mathbb{E}[R_{\text{test}}] + \mathbb{E}_{\pi^*}[R_{\text{roll}}].$$

The expected reward during the test phase is

$$\mathbb{E}[R_{\text{test}}] = n_0 m_0 + n_1 m_1.$$

The expected roll-out reward depends on the decision rule $\pi(y_1, y_0)$:

$$\begin{aligned}\mathbb{E}_{\pi}[R_{\text{roll}}] = \int (N - n_1 - n_0)\,[\pi(y_1, y_0)\mu_1 + (1 - \pi(y_1, y_0))\mu_0] \\ \times p(y_0 \mid \mu_0)\, p(y_1 \mid \mu_1) \\ \times p(\mu_0)\, p(\mu_1)\, dy_0\, dy_1\, d\mu_0\, d\mu_1.\end{aligned}$$

Regret in A/B Testing

Regret measures the expected reward lost compared with an oracle that knows the best action in advance.

$$\mathrm{Regret}(\pi) = \mathbb{E}[R \mid \mathrm{PI}] - \mathbb{E}_{\pi}[R].$$

With perfect information, the oracle always selects the action with the larger true mean reward:

$$\mathbb{E}[R \mid \mathrm{PI}] = N\,\mathbb{E}[\max(\mu_0, \mu_1)]\,,$$

where N is the total population size.

Lower regret means the policy performs closer to the perfect-information oracle.

Contents

From A/B Testing to Contextual Bandits

A/B testing: one-stage decision problem.

- Test actions a_0 and a_1 a fixed number of times.
- Observe rewards y_0 and y_1.
- Deploy the best action to the remaining population.

Bandit setting: decisions are made sequentially (sequential decision problems).

$$a_t \rightarrow r_t \rightarrow \text{update policy} \rightarrow a_{t+1}$$

- Feedback is observed immediately after each action.
- The policy adapts as new rewards are collected.
- Adaptive decisions can increase reward and reduce regret.

Types of Bandit Problems

Multi-armed bandit (MAB):

- At each step, the agent chooses an action $a_t \sim \pi_t$.
- It receives a reward $r_t \sim p_R(r_t \mid a_t)$.
- Goal: quickly identify and exploit the best action.

Contextual bandit:

- Before acting, the agent observes a context $s_t \sim p(s_t \mid s_{1:t-1})$.
- The policy depends on the context: $a_t \sim \pi_t(a_t \mid s_t)$.
- Rewards depend on both context and action:

$$r_t \sim p_R(r_t \mid s_t, a_t), \qquad R(s,a) = \mathbb{E}[R \mid s, a].$$

Each observation $D_t = (s_t, a_t, r_t)$ is used to update the policy and improve future decisions.

Objective in Contextual Bandits

The goal is to maximize the **expected cumulative reward**.

Finite horizon $(T < \infty)$:

$$J = \sum_{t=1}^{T} \mathbb{E}_{p_R(r_t|s_t,a_t)\pi_t(a_t|s_t)p(s_t|s_{1:t-1})}[r_t] = \sum_{t=1}^{T} \mathbb{E}[r_t].$$

Infinite horizon $(T = \infty)$:

$$J = \sum_{t=1}^{\infty} \gamma^{t-1}\mathbb{E}[r_t] = \sum_{t=1}^{\infty} \gamma^{t-1}\mathbb{E}\left[\sum_{a=1}^{K} R_a(s_t, a_t)\right].$$

$$R_a(s_t, a_t) = \begin{cases} R(s_t, a), & a_t = a, \\ 0, & a_t \neq a. \end{cases}$$

The discount factor $0 < \gamma < 1$ prevents the infinite sum from becoming unbounded. Small γ emphasizes immediate rewards; large γ emphasizes long-term rewards.

Exploration–Exploitation Trade-off

The main difficulty in bandit problems is deciding whether to explore or exploit.

- **Exploration:** try different actions to learn the reward function $R(s, a)$.
- **Exploitation:** choose the action currently predicted to give the highest reward.

If the agent exploits too early, it may learn from biased data and get stuck in a negative feedback loop.

Toward the optimal solution to contextual bandits

Let the belief state be the posterior over the reward-model parameters:

$$b_t = p(\theta \mid h_t), \qquad h_t = \{s_{1:t-1}, a_{1:t-1}, r_{1:t-1}\}.$$

The belief state is updated deterministically using Bayes' rule:

$$p(b_t \mid b_{t-1}, a_t, r_t) = \mathbb{I}[b_t = \text{BayesRule}(b_{t-1}, a_t, r_t)] \,.$$

The predictive distribution for the reward is

$$p(r_t \mid b_t) = \int p_R(r_t \mid s_t, a_t; \theta)\, p(\theta \mid b_t)\, d\theta.$$

Computing the exact optimal policy is usually intractable, so practical approximations are used:

- Upper Confidence Bounds (UCB)
- Thompsom sampling

Upper Confidence Bounds (UCB)

UCB principle: optimism in the face of uncertainty.

The agent builds an optimistic reward estimate $\widetilde{R}_t$ such that

$$\widetilde{R}_t(s_t, a) \geq R(s_t, a)$$

with high probability.

Then it chooses the greedy action under this optimistic estimate:

$$a_t = \arg\max_a \widetilde{R}_t(s_t, a).$$

- High uncertainty increases $\widetilde{R}_t$, encouraging exploration.
- As uncertainty decreases, the exploration bonus becomes smaller.
- The policy gradually shifts toward exploitation.

Thompson Sampling

Idea: choose each action with probability equal to the probability that it is optimal.

$$p_a = P(a = a^* \mid s_t, h_t) = \int \mathbb{I}\Big(a = \arg\max_{a'} R(s_t, a'; \theta)\Big)\, p(\theta \mid h_t)\, d\theta.$$

In practice, use a sample-then-exploit strategy:

$$\tilde{\theta}_t \sim p(\theta \mid h_t), \qquad a_t = \arg\max_{a'} R(s_t, a'; \tilde{\theta}_t).$$

- Uncertain posterior $\Rightarrow$ samples vary, encouraging exploration.
- Concentrated posterior $\Rightarrow$ samples agree, leading to exploitation.
- Simple to implement and often effective in practice.

Contents

From Contextual Bandits to MDPs

Contextual bandits assume that actions affect the current reward, but not future states.

Markov decision processes (MDPs) generalize this setting by allowing actions to affect future states.

$$s_t \xrightarrow{a_t} r_t,\ s_{t+1}$$

An MDP is defined by

$$(\mathcal{S}, \mathcal{A}, p, p_R, p_0),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, p is the transition model, p_R is the reward model, and p_0 is the initial state distribution.

$$a_t \sim \pi(a_t \mid s_t), \qquad s_{t+1} \sim p(s_{t+1} \mid s_t, a_t), \qquad r_t \sim p_R(r_t \mid s_t, a_t, s_{t+1}).$$

Value Function and Q-Function

For a fixed policy π, the **value function** or **state-value function** is the expected return when starting in state s and then following π:

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right].$$

The **Q-function** or **action-value function** is the expected return when starting in state s, taking action a, and then following π:

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s,\ a_0 = a \right].$$

Optimal Value Functions

A policy π^* is optimal if it achieves the highest value for every state:

$$V_{\pi^*}(s) \geq V_\pi(s) \qquad \forall s,\ \forall \pi.$$

The optimal value functions are denoted by V_* and Q_*.

Bellman's optimality equations:

$$V_*(s) = \max_a \left[R(s,a) + \gamma\, \mathbb{E}_{p(s'|s,a)} \left[V_*(s') \right] \right]$$

$$Q_*(s,a) = R(s,a) + \gamma\, \mathbb{E}_{p(s'|s,a)} \left[\max_{a'} Q_*(s',a') \right].$$

They express optimal decision making recursively: current reward plus optimal future value.

Deriving the Optimal Policy

Once the optimal value function is known, the optimal policy is obtained by acting greedily.

From the optimal Q-function:

$$\pi^*(s) = \arg\max_a Q_*(s, a).$$

Equivalently, using the optimal state-value function:

$$\pi^*(s) = \arg\max_a \left[R(s, a) + \gamma \, \mathbb{E}_{p(s'|s,a)} \left[V_*(s') \right] \right].$$

- Solving for V_*, Q_*, or π^* is called **policy optimization**.

Planning in MDPs

If the MDP model is known, i.e. $(\mathcal{S}, \mathcal{A}, p, p_R, p_0)$ is known, computing the optimal policy π^* is called **planning**.

If the model is unknown, the problem is tackled using **reinforcement learning** (Ch. 35).

This chapter covers planning methods:

- **Dynamic programming (DP):** value iteration and policy iteration.
- **Linear programming (LP).**

For large or continuous spaces, exact planning becomes intractable, motivating approximate DP and approximate LP.

Contents

Active Learning

Key idea: The agent chooses which data points should be labelled in order to learn efficiently.

- Useful when labels are expensive or time-consuming to obtain.
- Common examples include medical image classification and expert annotation tasks.
- Can achieve high performance with fewer labelled examples than random sampling.

Goal: Maximize learning while minimizing labelling cost.

Active Learning Scenarios

Membership query synthesis

- Generate an arbitrary query $x \sim p(x)$ and ask an oracle for its label, $y = f(x)$.
- In practice, it is difficult to learn good generative models and to access an oracle on demand.

Stream-based selective sampling

- The agent receives a stream of inputs $x_1, x_2, \ldots, x_n$.
- At each step, it decides whether to request a label.

$\Rightarrow$ These scenarios are mostly of theoretical interest.

Active Learning Scenarios

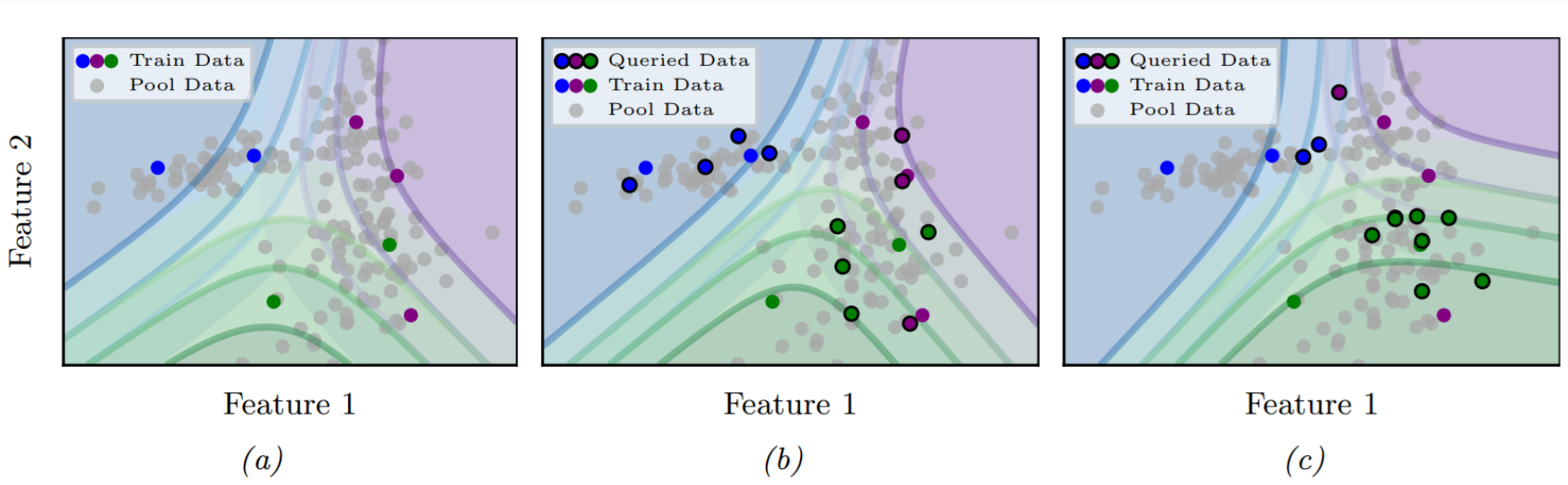
Pool-based sampling

- A pool of unlabelled samples $\mathcal{U}$ is available from the start.
- An acquisition function selects the most informative samples to label.
- The model is updated and the process is repeated.
- Stop when the budget, pool, or target performance is reached.

This is the active learning scenario we consider.

Active learning vs randomly sampled data

Logistic regression applied to a 2-d, 3 class dataset.



a) Initial training data (ITD) $\Rightarrow$ Accuracy: 0.818

b) ITD + 11 randomly sample points $\Rightarrow$ Accuracy:0.848

c) ITD + 11 points with active learning $\Rightarrow$ Accuracy: 0.969

Relationship to Other Sequential Decision Problems

These methods are closely related, but they differ in their objective and in what they query.

Problem	**Goal**	**Action**
Active learning	$\arg\min_f \mathbb{E}_{p(x)}[\ell(f^*(x), f(x))]$	Choose x to obtain its label $y = f^*(x)$
Bayesian optimization	$\arg\max_{x \in \mathcal{X}} f^*(x)$	Choose x to evaluate $f^*(x)$
Contextual bandits	$\arg\max_\pi \mathbb{E}_{p(x)\pi(a\|x)}[R^*(x,a)]$	Choose action a to observe reward $R^*(x,a)$

All three methods aim to make informative decisions using as few queries or actions as possible.

Acquisition Strategies

Acquisition strategies decide which unlabelled points should be queried next.

Active learning heuristics

- **Uncertainty sampling**
- **Query by committee (QBC)**
- **Information-theoretic methods**

Goal: select the most informative samples to label.

Uncertainty Sampling

Idea: query the point for which the current model is most uncertain.

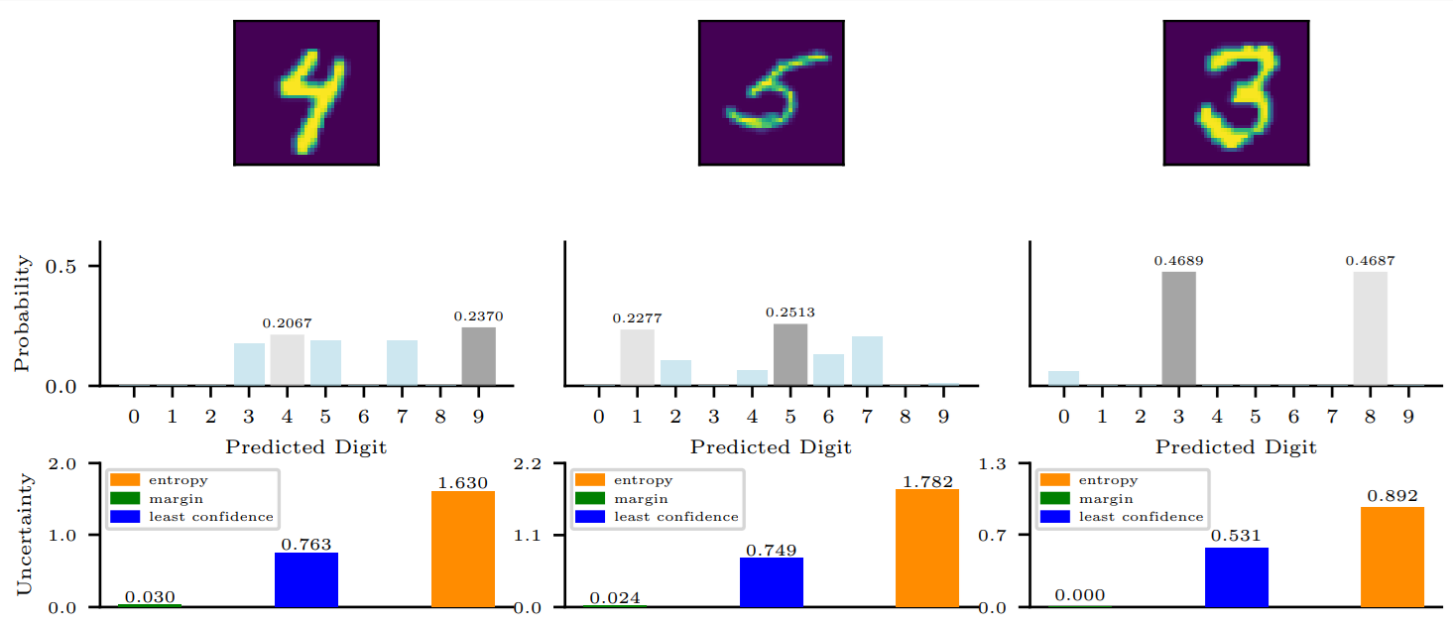
For classification, let $p_n = [p(y = c \mid x_n)]_{c=1}^{C}$ be the predicted class probabilities for an unlabelled point x_n.

Let $p_{(1)}$ and $p_{(2)}$ denote the largest and second-largest predicted class probabilities.

Strategy	**Score**	**Query**
Entropy sampling	$-\sum_c p_c \log p_c$	Highest entropy
Margin sampling	$p_{(1)} - p_{(2)}$	Smallest margin
Least confident	$1 - p_{(1)}$	Lowest confidence

Examples of Uncertainty Sampling Strategies

Different uncertainty criteria may select different data points to query.



Selected data points: Entropy → middle example; Margin → right example; Least confidence → left example.

In practice, **margin sampling** is often found to work best.

Query by Committee (QBC)

Idea: use disagreement between several models as a measure of uncertainty.

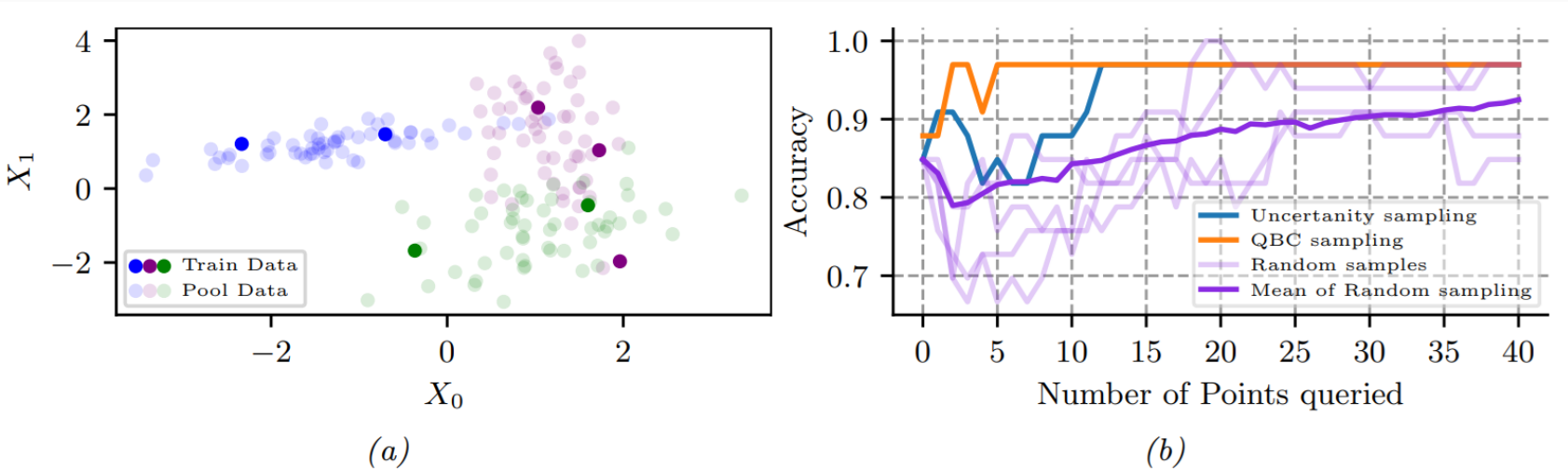
Useful for models such as SVMs, which usually output class predictions rather than probabilities.

$$v_{nc} = \sum_{k=1}^{K} \mathbb{I}(c_n^k = c), \qquad q_{nc} = \frac{v_{nc}}{K}$$

- v_{nc}: votes for class c.
- q_n: empirical distribution of committee votes.
- Query points with margin or entropy sampling as before using q_n.

QBC vs uncertainty sampling

Takeaway: Query by committee (QBC) can often outperform vanilla uncertainty sampling with a single model, reaching high accuracy with fewer queried points.



Information-Theoretic Methods

Idea: query the point whose label gives the most information about the model parameters w.

Bayesian Active Learning by Disagreement (BALD): Selects point that reduces parameter uncertainty.

$$\alpha(\boldsymbol{x}) = \mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}\left[D_{\mathbb{KL}}(p(\boldsymbol{w} \mid \mathcal{D}, \boldsymbol{x}, y) \,\|\, p(\boldsymbol{w} \mid \mathcal{D}))\right] =$$

$$\mathbb{H}(y \mid \boldsymbol{x}, \mathcal{D}) - \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}\left[\mathbb{H}(y \mid \boldsymbol{x}, \boldsymbol{w}, \mathcal{D})\right]$$

Batch Active Learning

Idea: select a batch of unlabelled examples at once, instead of querying one point at a time.

This is useful because retraining the model after every single labelled example can be too slow.

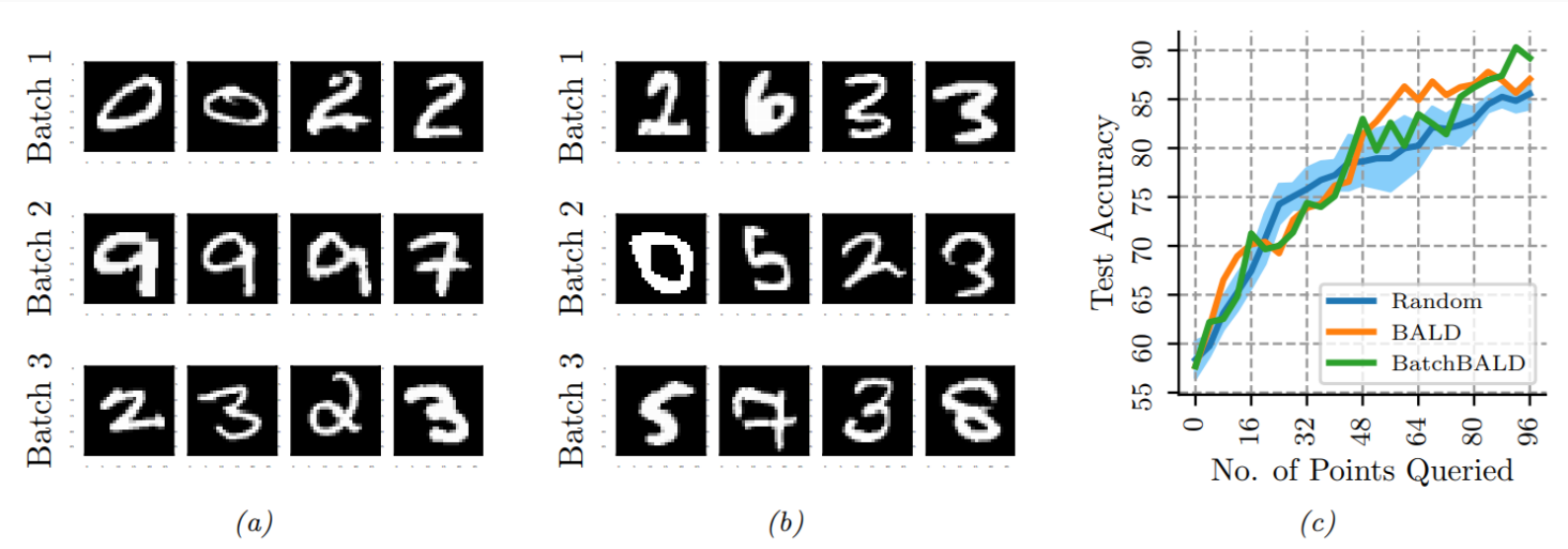
BatchBALD: selects a batch whose labels jointly provide the most information about the model parameters.

$$\alpha_{\text{BBALD}}(\boldsymbol{x}_{1:B}, p(\boldsymbol{w} \mid \mathcal{D})) = \mathbb{H}(\boldsymbol{y}_{1:B} \mid \boldsymbol{x}_{1:B}, \mathcal{D}) - \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}\left[\mathbb{H}(\boldsymbol{y}_{1:B} \mid \boldsymbol{x}_{1:B}, \boldsymbol{w}, \mathcal{D})\right]$$

$\alpha_{\text{BBALD}}(\boldsymbol{x}_{1:B}, p(\boldsymbol{w} \mid \mathcal{D}))$ is intractable in general, so it is usually approximated.

BALD vs. BatchBALD on MNIST

BatchBALD selects more diverse batches by accounting for joint information.



(a): BALD selects individually informative but often redundant examples.
(b): BatchBALD selects more diverse examples within each batch.
(c): BatchBALD results in more efficient learning.

Questions?

Next Session

Causality (Ch. 36)

June 3, 2026

Fátima Sánchez-Cabo