02-750 Week 3

Automation of Scientific Research

Aidan Jan

January 29, 2026

Artificial Intelligence

Note that machine learning is a subset of artificial intelligence. Deep learning is a subset of machine learning.

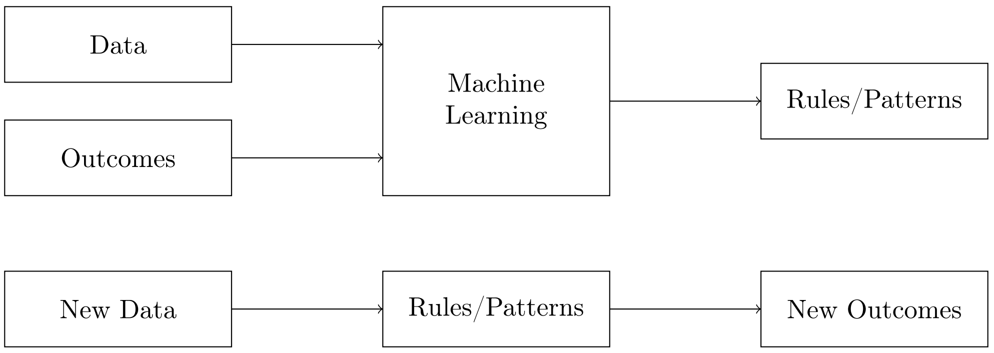
What is Artificial Intelligence (AI)?

"Any task performed by a machine that would have previously been considered to require human intelligence." - Professors Mavin Minsky and John McCarthy (1956)

- "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable." - Professor John McCarthy (2004)
- "A system's ability to adapt and improvise in a new environment, to generalize its knowledge and apply it to unfamiliar scenarios." - AI researcher Français (2020)

What is Machine Learning (ML)?

- A branch of AI
- "Study of computer **algorithms** that allow computer programs to automatically **improve** through **experience**." - Prof. Tom Mitchell (1997)



Types of Machine Learning

- Supervised Learning
  - Input: Training data and labels.
  - Goal: Learn function to map new unlabelled data to labels.
  - For example, classification and regression.

- Semi-Supervised Learning
  - Input: Training data, some of which is labelled.
  - Goal: Learn function to map new unlabelled data to labels and/or learn structure in the data.
- Reinforcement Learning
  - Given a sequence of states and actions with (delayed) rewards, output a policy
  - Policy is a mapping from states to actions that tells you what to do in a given state.
  - Agent and environment interact at discrete time steps. Agent commits ana action on the environment, which changes its state and rewards (either positively or negatively) the agent.
  - For example, training a self-driving car.
- Unsupervised Learning
  - Input: Training data without labels.
  - Goal: Learn structure in the data.
  - For example, K-means Clustering

When to Use Machine Learning

Machine Learning is used when:

- human expertise does not exist (e.g., navigating on Mars)
- humans can't explain their expertise (e.g., speech recognition)
- models must be customized (e.g., personalized medicine)
- models are based on huge amounts of data (e.g., genomics)

Learning isn't always useful:

- There is no need to "learn" to calculate payroll

Designing a Learning System

- Choose the training experience
- Choose exactly what is to be learned (i.e., the *target function*)
- Choose how to represent the target function
- Choose a learning algorithm to infer the target function from the experience

Training vs. Test Distribution

- We generally assume that the training and test examples are independently drawn from the same overall distribution of data
  - We call this "i.i.d" which stands for "independent and identically distributed"
- If examples are not independent, it requires *collective classification.*
- If the test distribution is different, it requires *transfer learning.*

ML in a Nutshell

- Tens of thousands of machine learning algorithms. (Hundreds of new ones every year)
- Every ML algorithm has three components:
  - Representation
  - Optimization
  - Evaluation
- Deep Learning tends to do better than most machine learning algorithms when the amount of data grows large.

Various Function Representations

- Numerical functions
  - Linear Regression
  - Neural Networks
  - Support Vector Machines
- Symbolic functions
  - Decision trees
  - Rules in propositional logic
  - Rules in first-order predicate logic
- Instance-based functions
  - Nearest-neighbor
  - Case-based
- Probabilistic Graphical Models
  - Naïve Bayes
  - Bayesian networks
  - Hidden-Markov Models (HMMs)
  - Probabilistic Context Free Grammars (PCFGs)
  - Markov networks

Various Search / Optimization Algorithms

- Gradient Descent
  - Perceptrons
  - Backpropagation
- Dynamic Programming
  - HMM Learning
  - PCFG Learning
- Divide and Conquer
  - Decision tree induction
  - Rule learning

- Evolutionary Computation
  - Genetic Algorithms (GAs)
  - Genetic Programming (GP)
  - Neuro-evolution

Evaluating ML Models

- Accuracy
- Precision and Recall
- Squared Error
- Likelihood
- Posterior Probability
- Cost / Utility
- Margin
- Entropy
- K-L Divergence
- etc.

Prototypical Active Learning Algorithm

Obtain initial labeled training data $D_L = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, where each $x_i \in \mathcal{X}$ is an input input and $y_i \in \mathfrak{C}$ its corresponding label.

- Repeat, until we hit stopping criterion. (Exit the loop when we run out of money or decide the model is good enough.)
  - Learn model: $h = Learn(D_L); h \in \mathcal{H}$. This function is called the 'Base Learner'; it can be any supervised learning algorithm.
  - Select the next point to label, $x_i \in \mathcal{X}$, according to the data access model. This is the 'Query selection' step. The **data access model** determines which instances are selectable at any given time.
  - Pay "oracle" for x_i's label, $y_i \in \mathfrak{C}$. (The oracle tells us the true label of a selected instance, and is typically a wet lab experiment, a Ph.D. student, data scientists, etc.)
  - Update training data: $D_L = D_L \cup \{(x_i, y_i)\}$
- Output final model: $h = Learn(D_L)$

Data Access Model

Regardless of the data access model, queries are made based on the **expected utility** of each instance. Informally, the utility of an instance is the degree to which it helps us improve the model.

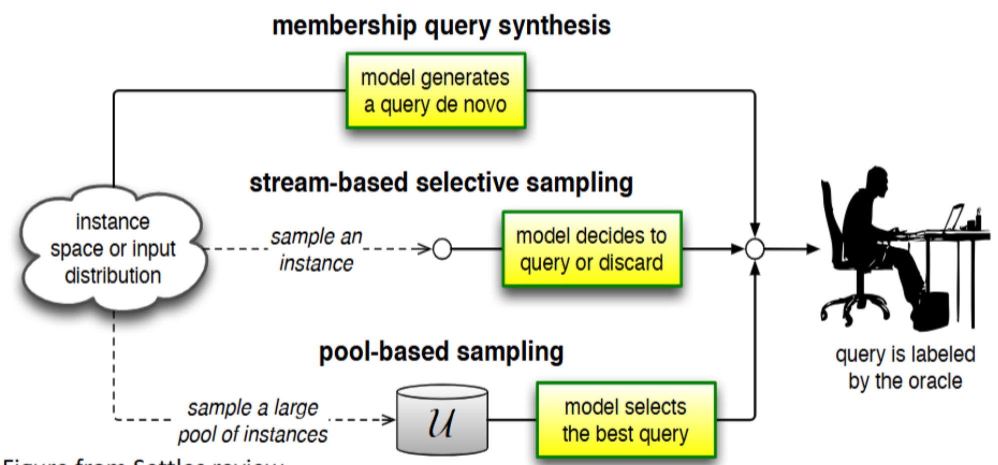

membership query synthesis
model generates a query de novo
stream-based selective sampling
instance space or input distribution
sample an instance
model decides to query or discard
pool-based sampling
sample a large pool of instances
$\mathcal{U}$
model selects the best query
query is labeled by the oracle

Figure from Settles review

Membership Query Synthesis (MQS)

MQS selectors can request **any** point in $\mathcal{X}$ (i.e., instances space)

- In science, $\mathcal{X}$ will often represent a space of possible experiments.

This setting includes queries that the learner generates *de novo*, rather than those sampled from some underlying natural distribution.

Stream-based Selective Sampling

Selective samplers are presented a stream of unlabeled instances

- Examples: automated chemical synthesis and characterization; flow cytometry

Each unlabeled instance is drawn one at a time from the data source

- The learner decides whether to request label, or not
- If no label is request, the instance is 'lost'

Stream-based selective sampling is less common in basic research because we **usually** preserve all the data and/or samples for future access. Stream-based sampling is more relevant to engineering contexts, where the ultimate goal may be to iteratively improve an underlying design in real-time.

Pool-based Sampling

Pool-based samplers have access to a large unlabeled pool $\mathcal{U}$.

- Examples: tissue banks; electronic medical records; unlabeled images.
- Common in biological research when it is easy to obtain data (e.g., images) but it is costly to obtain labels.

Query Selection Methods

- There are many different approaches to query selection
- Some methods are merely *heuristics*. (i.e., they provide no formal guarantees)
- Others do come with one or more formal guarantees (i.e., we can prove theorems about them)
- For example, guarantees with respect to:

- Sample complexity (the number of samples needed to learn a 'good' model)
- Statistical consistency (a guarantee that the model will converge to the 'true' model, given a sufficient amount of training data, even though it is not *iid*)

Heuristic Query Selection Methods

Uncertainty Sampling

Generic uncertainty sampling algorithm (pool-based sampling)

```
Input U := a pool of unlabeled instances and D_L := labeled training data
Output h = learn(D_L)
for i = 1, 2, ... do
    h = Learn(D_L);
    Select x_i in U, the most uncertain instance according to model h
    Query the 'oracle' for x_i's label, y_i
    D_L += {(x_i, y_i)}
    Remove x_i from U
return h = Learn(D_L)
```

Here, the notion of uncertainty is with respect to the model's confidence in x's label.

Aside: Single vs. Batch Mode Learning

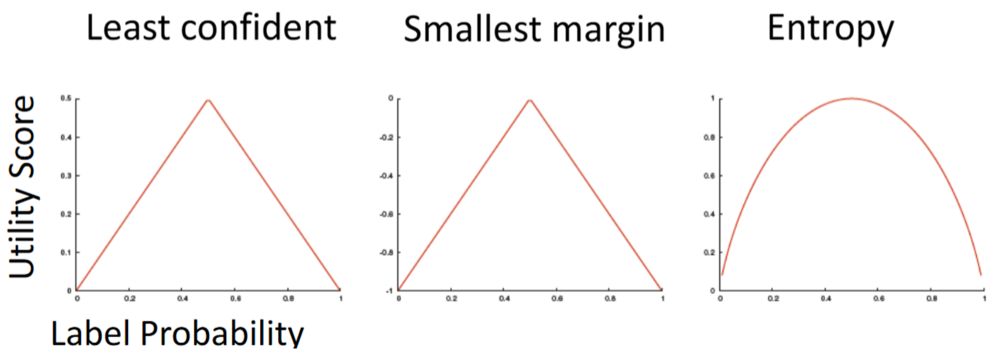
In science, the fixed costs of obtaining labels (i.e., by running experiments) can sometimes be amortized over multiple queries. For example, running 96 experiments at once in a 96-well plate.

- When this is true, the query selection algorithm may be asked to select **batches** of queries for each round, instead of individual experiments.
- Batch-mode query selection is something that can, in principle, be added to most active learning algorithms
- Same algorithm as the one above, but instead of selecting one `x_i`, we select the top k most uncertain instances.

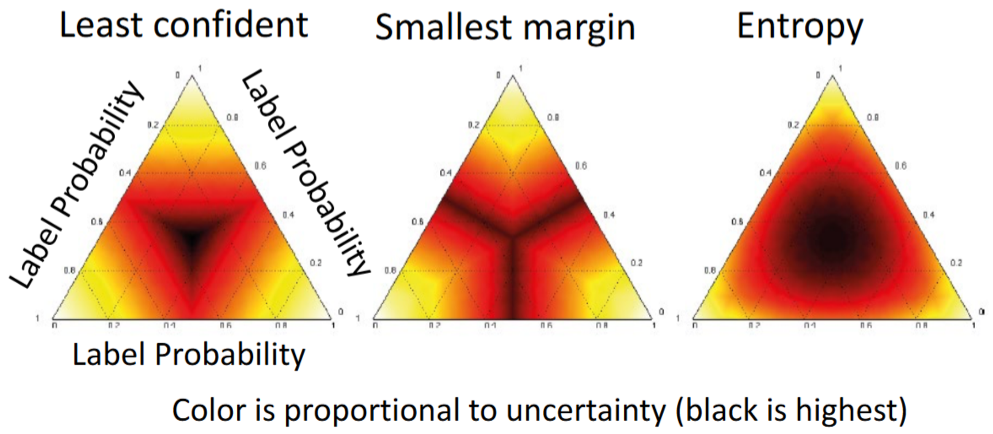
Quantifying Uncertainty: Probabilistic Classifiers

- Least Confident: $x^*_{LC} = \arg\max_{x \in \mathcal{U}} 1 - P(\hat{y}|x)$. Here, $\hat{y} = \arg\max_{y \in \mathfrak{C}} P(y|x)$
- Smallest margin: $x^*_M = \arg\min_{x \in \mathcal{U}} P(\hat{y}_1|x) - P(\hat{y}_2|x)$. Here, $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable class labels, respectively. Small margins mean it is difficult to distinguish between two most-likely labels.
- Entropy: $x^*_H = \arg\max_{x \in \mathcal{U}} - \sum_{y \in \mathfrak{C}} P(y|x) \log P(y|x)$
  - Entropy is a measure over a probability distribution
  - Higher entropy means more uncertainty about the labels
  - Entropy is an information-theoretic measure that represents the amount of information needed to encode a distribution.

Visualizing uncertainty (binary classification)



Visualizing uncertainty (ternary classification)



Here, color is proportional to uncertainty. (Black is highest.) Empirical comparisons of these measures suggest that the best strategy may be application-dependent.

[stop 49]

Query by Committee

Expected Model Change

Minimizing Expected Risk

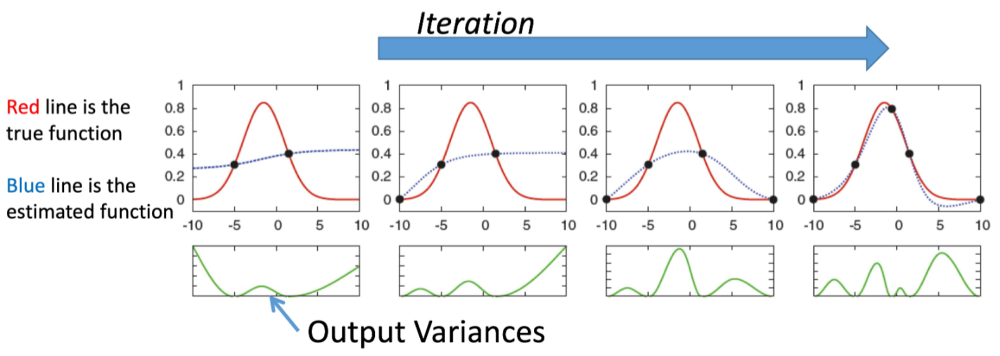
Density-based Sampling

Heuristic Query Selection Methods

Uncertainty Sampling

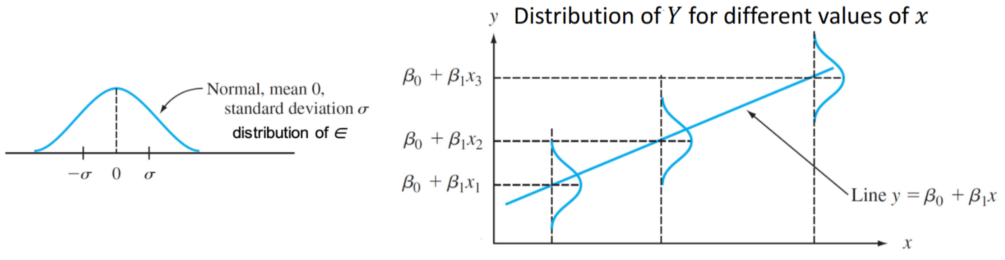
Quantifying Uncertainty for Regression

- For regression problems, uncertainty is often quantified as the **output variance** (i.e., variance of the conditional distribution on y)
- Example: 1D regression model



Consider the predictions made by a simple linear regression model

$$Y = \beta_0 + \beta_1 x + \epsilon$$


Normal, mean 0, standard deviation σ distribution of $\in$
$-\sigma$ 0 σ
Distribution of Y for different values of x
$\beta_0 + \beta_1 x_3$
$\beta_0 + \beta_1 x_2$
$\beta_0 + \beta_1 x_1$
Line $y = \beta_0 + \beta_1 x$

- The values of β_0, β_1, σ^2 will almost never be known.
- We assume the variance (amount of variability) of the distribution of Y values to be the same at each different value of fixed x. (i.e., homogeneity of variance assumption).

Given $D_L = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, the sum of squared vertical deviations from the point to the line is

$$f(b_0, b_1) = \sum_{i=1}^{n} [y_i - (b_0 + b_1 x_i)]^2$$

The point estimates of β_0 and β_1, denoted by $\hat{\beta}_0$ and $\hat{\beta}_1$ are the **least squares estimates** - that is, the values that minimize $f(b_0, b_1)$.

$$b_1 = \hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=i}^{n}(x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$$

$$b_0 = \hat{\beta}_0 = \frac{\sum_{i=1}^{n} y_i - \hat{\beta}_1 \sum_{i=1}^{n} x_i}{n} = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i_1}^{n} y_i}{n}$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}$$

$$\hat{\sigma}^2 = s^2 = \frac{\sum_{i=i}^{n}(y_i - \hat{y}_i)^2}{n-2} = \frac{1}{n-2} \sum_{i=1}^{n} \hat{e}_i^2$$

- The expected value of $\hat{\beta}_1$ is $\mathbb{E}(\hat{\beta}_1) = \hat{\beta}_1$, so $\hat{\beta}_1$ is an **unbiased estimator** of β_1
- The vvariance and standard deviation of $\hat{\beta}_1$ are

$$\mathbb{V}(\hat{\beta}_1) = \hat{\sigma}^2_{\hat{\beta}_1} = s^2_{\hat{\beta}_1} = \frac{\sigma^2}{S_{xx}} \approx \frac{\hat{\sigma}^2}{S_{xx}} = \frac{s^2}{S_{xx}}$$

Consequently,

$$\hat{\sigma}_{\hat{\beta}_1} = s_{\hat{\beta}_1} = \frac{\sigma}{\sqrt{S_{xx}}} \approx \frac{\hat{\sigma}}{\sqrt{S_{xx}}} = \frac{s}{\sqrt{S_{xx}}}$$

- A $100(1-\alpha)$ Cl* for the slope $\hat{\beta}_1$ of the true regression line is

$$\hat{\beta}_1 \pm t_{\frac{\alpha}{2}, n-2} \cdot s_{\hat{\beta}_1}$$

Let $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x^*$, where x^* is **some fixed value of** x

- The expected value of $\hat{Y}$, $\mathbb{E}(\hat{Y}) = \mathbb{E}(\hat{\beta}_0 + \hat{\beta}_1 x^*) = \mu_{\hat{\beta}_0 + \hat{\beta}_1 x^*} = \hat{\beta}_0 + \hat{\beta}_1 x^*$
- Thus, $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x^*$ is an **unbiased estimator** for $\hat{\beta}_0 + \hat{\beta}_1 x^*$

The variance of $\hat{Y}$ is

$$\mathbb{V}(\hat{Y}) = \hat{\sigma}^2_{\hat{Y}} = s^2_{\hat{Y}} = \sigma^2 \left[\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n x_i^2 - \frac{(\sum_{i=1}^n x_i)^2}{n}} \right] = s^2 \left[\frac{1}{n} + \frac{(x^* - \bar{x})^2}{S_{xx}} \right]$$

The standard deviatoin of $\hat{Y}$ is

$$\hat{\sigma}_{\hat{Y}} = s_{\hat{Y}} = s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{S_{xx}}}$$

Uncertainty for Linear Regression in Practice

The standard way of estimating β via ordinary least squares (OLS)

- Given a vector $k \geq b$ observations/samples the OLS estimate is

$$\hat{\beta} = (X'X)^{-1} X' y$$

  - Once we estimate $\hat{\beta}$, the predicted value for any x is: $\hat{y} = x\hat{\beta}$
- It can be shown (via the Gauss-Markov Theorem) that:
  - OLS produces a $\hat{\beta}$ that minimizes $\sum_{i=1}^k (\hat{y}_i - y_i)^2$
  - $\mathbb{E}[\hat{\beta} - \beta_{true}] = 0$ (i.e., OLS is an unbiased estimator)
  - $\mathbb{V}(\hat{\beta}) = \sigma^2 (X'X)^{-1}$
  - $\mathbb{V}(\hat{y}) = \sigma^2 x_u (X'X)^{-1} x_u'$

Quantifying Uncertainty for other Regression Models

- Some models (e.g., some neural networks; Gaussian processes) have closed-form means for calculating the output-variance
- Unfortunately, most models do not.

Suppose that you have a model that doesn't have a closed-form way to compute the output variance. How could you easily **estimate** the variance for a given x?

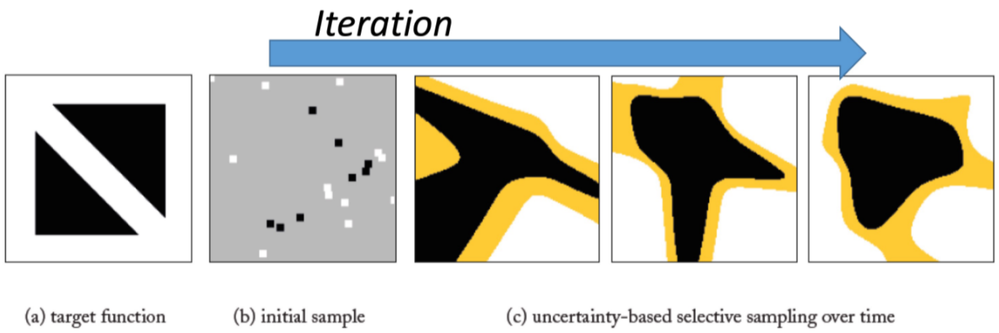
Advantages of Uncertainty Sampling

- Usually easy to implement
- Can be used with most base learners
- Can be applied to all data access models

Limitations of Uncertainty Sampling

Using a **single** model to determine which inputs are uncertain is risky.

- The initial model will be trained on a small set of labeled examples, and may become overconfident (or biased).



- One obvious solution for addressing this limitation is to use multiple models.

Query by Committee (QBC)

Basic idea:

- Maintain a **diverse** set of models $\mathcal{M} = \{h_1, \cdots, h_p\}$
  - Here, 'diverse' means that the models do not always agree with each other.
- For each unlabeled point in the domain/pool/stream: $x \in \mathcal{U}$
  - Use each model to predict x's label: $\hat{y}_1 = h_1(x), \cdots, \hat{y}_p = h_p(x)$
  - Quantify the disagreement amongst the committee's predictions
- Query the point that causes maximal disagreement between the models.

Two key tasks:

1. Constructing and maintaining a diverse committee
2. Quantify the disagreement amongst the committee

Committee Construction and Maintenance

There are many ways to construct and maintain a committee

- The easiest approach is to use an **ensemble** method as the base learner
- Ensemble methods train **multiple models** from a single set of training data
- The training algorithm seeks to ensure that their errors are **not** highly correlated
  - That is, the models exhibit diversity in their 'opinions'

Intuition:

- If the errors made by the ensemble's constituent models are not highly correlated, then a combined prediction (e.g., by voting) will tend to have better **predictive performance** than any single model, on average.
- See *wisdom of the crowds* for a similar phenomenon

One way to construct a committee is **Bagging** ("Bootstrap Aggregation").

- Given a training set, D of size n, Bagging involves creating m new training sets of size n by sampling from D uniformly and with replacement (i.e., bootstrapping)
- You then learn a separate model for each of the m sets, which produces a committee.
- In the context of QBC, the models form the committee.

The second approach is a **Random Forest**

- Random forests combine bagging with **random subspaces** (sometimes called feature bagging)
- In the context of QBC, the trees form the committee.

The third approach is **boosting**.

- Boosting creates a committee in a sequential fashion by training models that handle errors made by previous models.
- In the context of QBC, the models form the committee.

Aside: Weighted Loss Functions

Learning algorithms find parameters that optimize some objective

- e.g., training accuracy: $J_\theta = \frac{1}{n} = \sum_{(x_i,y_i)\in D_L} \mathbb{I}(y_i = h_\theta(x_i))$
- In general, we can always use a **weighted loss function**
  - E.g., weighted training accuracy: $J_\theta = \frac{1}{N} \sum_{(x_i,y_i)\in D_L} w_i \cdot \mathbb{I}(y_i = h_\theta(x_i))$
  - The specific set of weights, $\{w_1, \cdots, w_{|D_L|}\}$, will influence which model is chosen.
- Informally, the weights allows us to declare that some instances are more 'important' to predict correctly than others
  - Boosting assigns large weights to instances that were misclassified by the previous models.

Aside: Ensemble Methods

- Ensemble methods were invented to reduce **overfitting**
- They just happen to be useful in the context of query by committee. (i.e., any ensemble method *can* be used to create and maintain committees.)

Committee Construction and Maintenance

There are many ways to construct and maintain a committee.

- Later this semester, we will discuss Bayesian learning methods
- Bayesian learning methods compute a **posterior distribution** over the parameters of the model, $P(\theta|D_L)$
- Given $P(\theta|D_L)$, we can then sample sets of parameters, each of which represents a distinct model.

Quantifying Committee Disagreement

- Let $\mathcal{M} = \{h_1, \cdots, h_p\}$ be a diverse set of models (i.e., committee members)
- Method 1: **Hard Vote Entropy** (for classification)

$$x^*_{HVE} = \arg\max_{x \in \mathcal{U}} - \sum_{y \in \mathcal{C}} \frac{V(y)}{|\mathcal{M}|} \log \frac{V(y)}{|\mathcal{M}|}$$

  where $V(y) = \sum_{h \in \mathcal{M}} \mathbb{I}(y = h(x))$ is the number of votes for label $y \in \mathcal{C}$ for point $x \in \mathcal{U}$ and $|\mathcal{M}|$ is the size of the committee.

- **Hard vote entropy** selects instances where the vote is closest to being a tie.