Rule-Based Classification and Weakly Supervised Learning

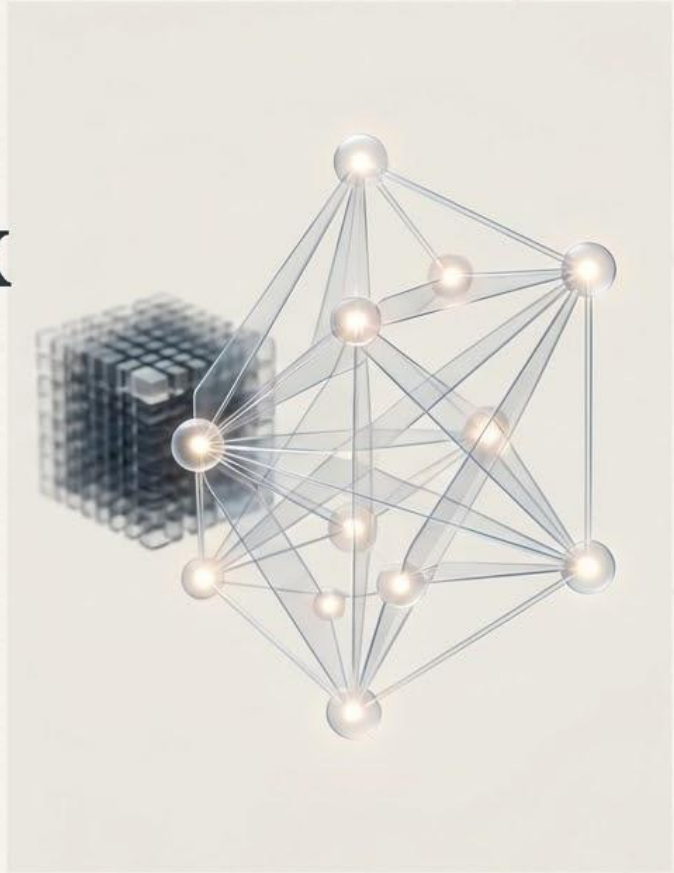
- Duration: 1 Hour
- Target Audience: Data Science/Computer Science students
- Prerequisites: Understanding of basic classification, decision trees, association rule mining

Part 1: Introduction

Beyond the Black Box & Limited Labels

Rule-Based Classification and Weakly Supervised Learning

Context: Part 1 (5 Minutes) - Setting the stage for interpretable models and learning with scarce data.



The Opening Context

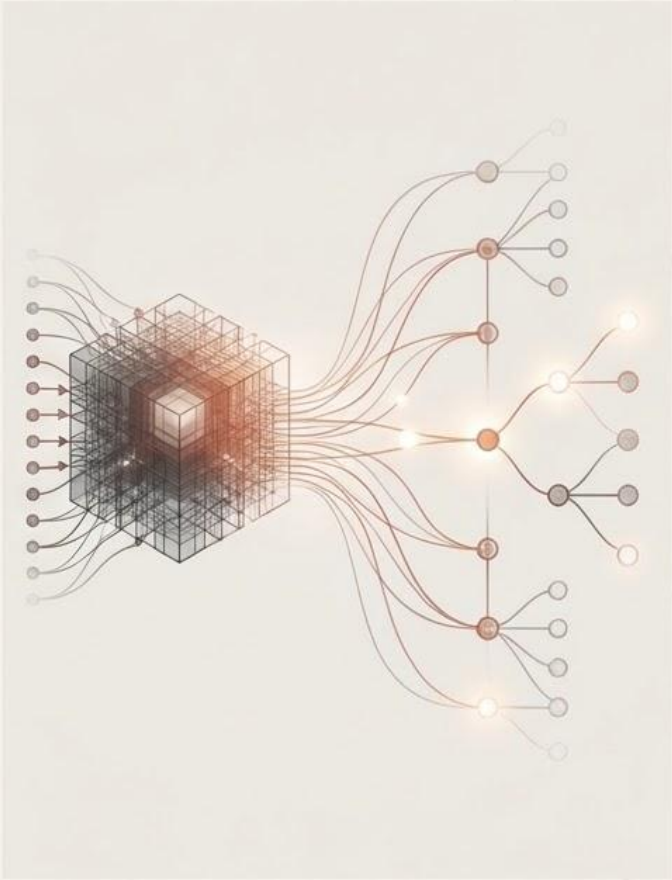
Why Do We Need New Approaches?

The Journey So Far:
We've seen how powerful models like Decision Trees, Bayesian Networks, and Support Vector Machines (SVMs) can accurately classify complex data.

The Interpretability Problem:
What if accuracy isn't enough? What if the business requires highly interpretable, human-readable rules to justify a decision?

The Data Scarcity Problem:
What if we completely lack the massive amounts of explicitly labeled data required to train those traditional models?

Today's Focus:
We are going to explore rule-based classifiers that generate clear, human-understandable knowledge, and weakly supervised methods that can intelligently learn from limited labels. These are essential techniques for modern, real-world AI applications.



Learning Objectives

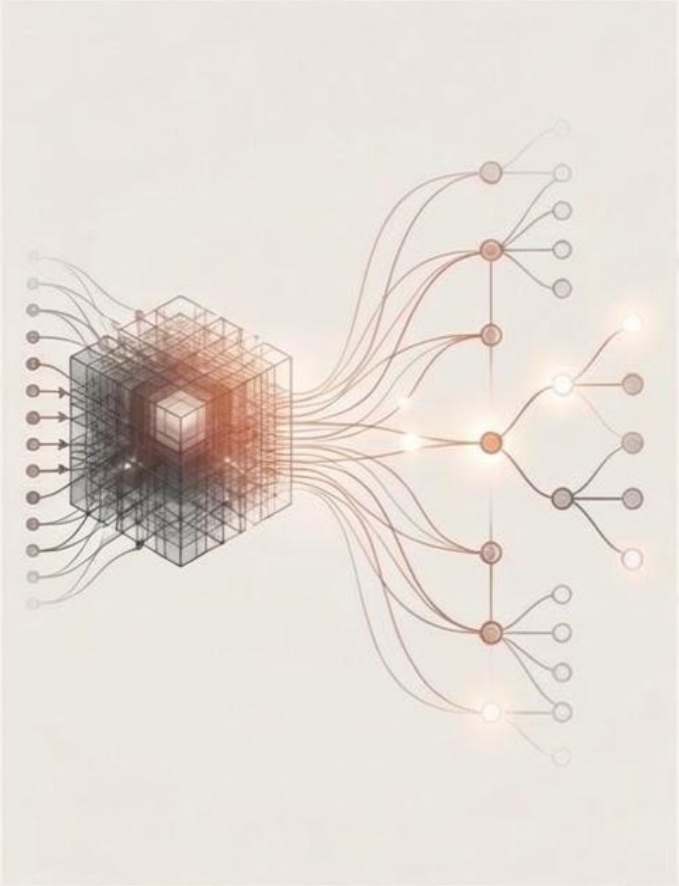
What You Will Learn Today

Extract Rules:
Automatically extract interpretable IF-THEN rules directly from established decision trees.

Understand Algorithms:
Explain how sequential covering algorithms work for direct rule induction.

Master Patterns:
Master the concepts behind associative and discriminative pattern-based classification.

Learn with Less:
Understand the core approaches to weakly supervised learning, specifically semi-supervised, active, and transfer learning.

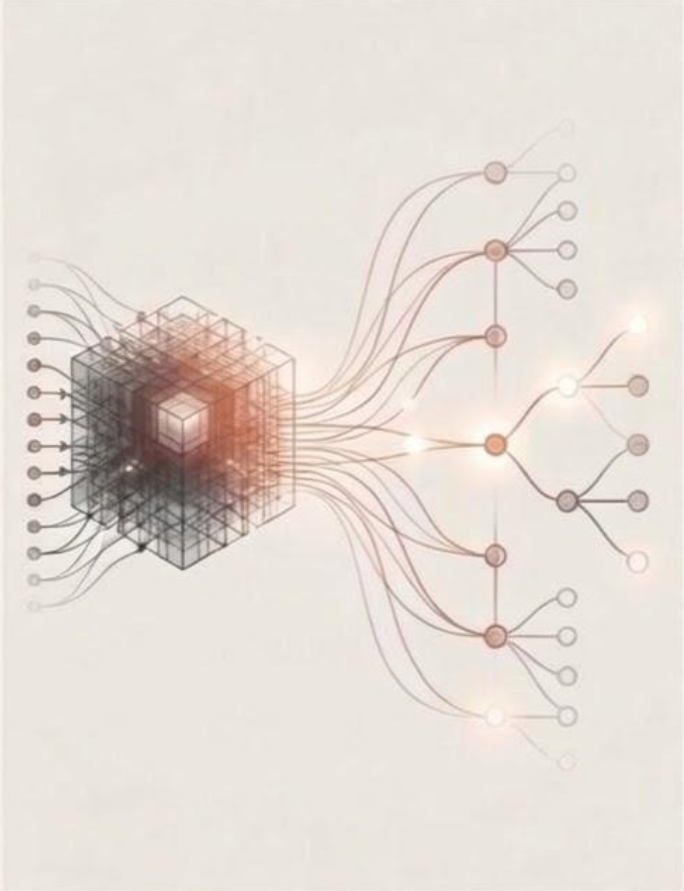


Part 2: Rule-Based Classification Methods

From Branches to Logic

Rule Extraction from Decision Trees

Turning complex trees into modular, human-readable rules.



The Concept and Process

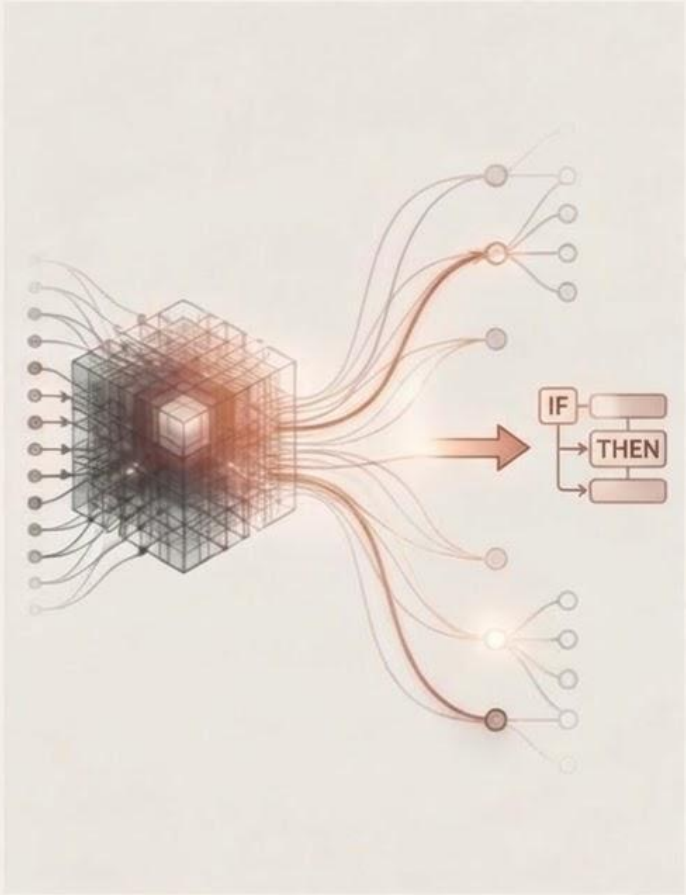
Why Extract Rules?

The Concept:

While decision trees are highly interpretable on their own, extracting the logic into individual rules makes the model even more modular and understandable. Every single path from the root of the tree down to a leaf node represents a distinct rule.

The Extraction Process:

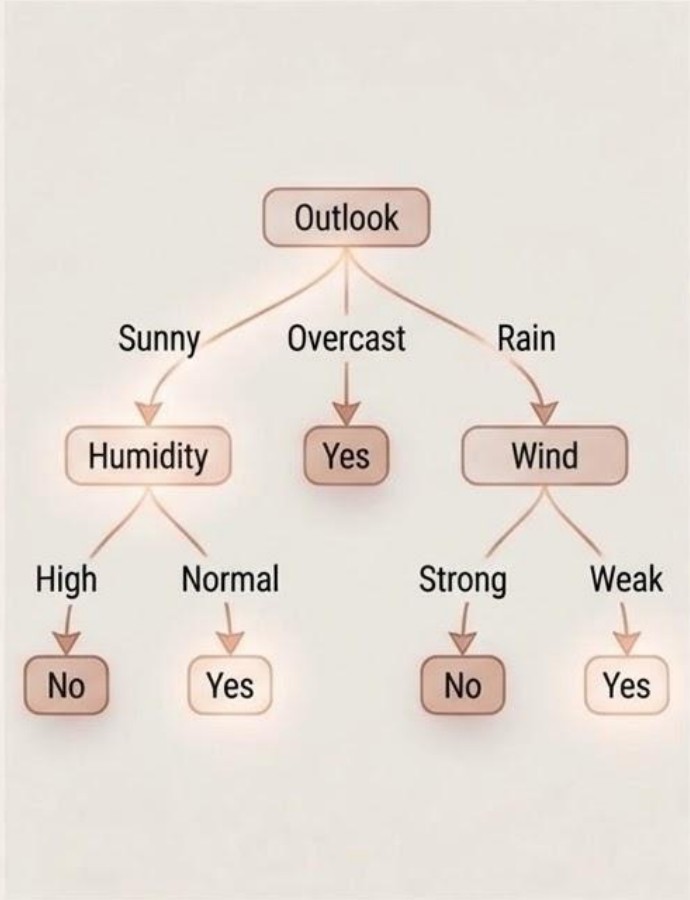
1. For each leaf node, trace the complete path from the root down to that leaf.
2. Combine all the conditions along that specific path to form the rule's antecedent (the “IF” part).
3. Use the leaf’s final class label as the rule's consequent (the “THEN” part).
4. Optionally prune or simplify the resulting rules for better clarity.



A Visual Example

The “Play Tennis” Decision Tree

Here is a classic example of a simple decision tree evaluating whether to play tennis based on the weather:



The Extracted Rules

Translating the Tree

By tracing every path from the root to the leaves in the previous tree, we extract the following modular rules:

IF Outlook = Sunny **AND** Humidity = High **THEN** Play = **No**

IF Outlook = Sunny **AND** Humidity = Normal **THEN** Play = **Yes**

IF Outlook = Overcast **THEN** Play = **Yes**

IF Outlook = Rain **AND** Wind = Strong **THEN** Play = **No**

IF Outlook = Rain **AND** Wind = Weak **THEN** Play = **Yes**

Advantages & The C4.5Rules Algorithm

Why This Matters in Practice

The Advantages:

- Extracted rules are often much simpler to read than visually navigating a massive, full-scale tree.
- Each rule stands independently and is highly interpretable.
- Domain experts can easily read, validate, and sign off on the logic.

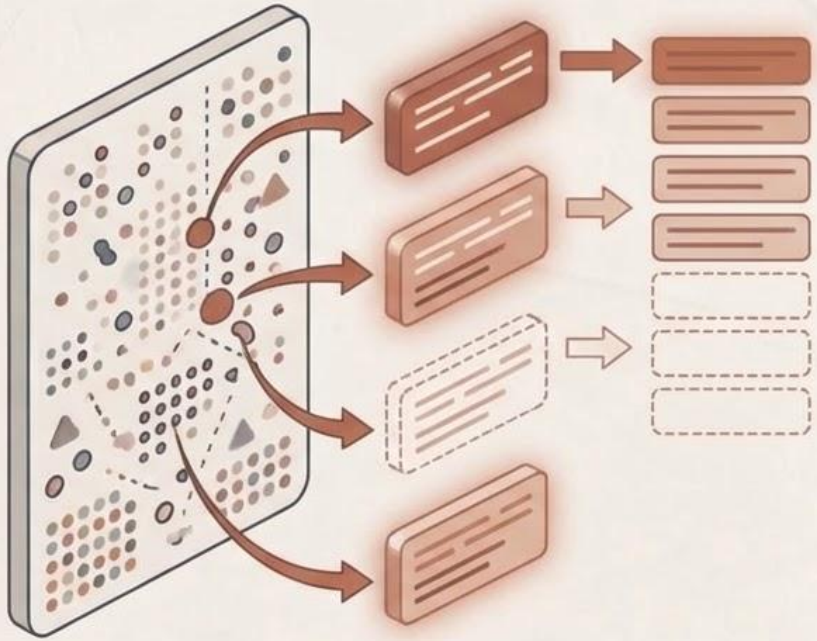
The C4.5Rules Algorithm:
This is the standard method for rule extraction.

1. Build the initial decision tree.
2. Convert every path into a distinct rule.
3. Prune each rule by actively removing conditions that do not improve accuracy.
4. Sort the finalized rules by their estimated accuracy.
5. Process the rules in that specific order for future classification.

The Sequential Covering Algorithm

Learning IF-THEN Rules One at a Time

Part 2 (Continued) - How to **extract rules directly from training data** without building a decision tree first.



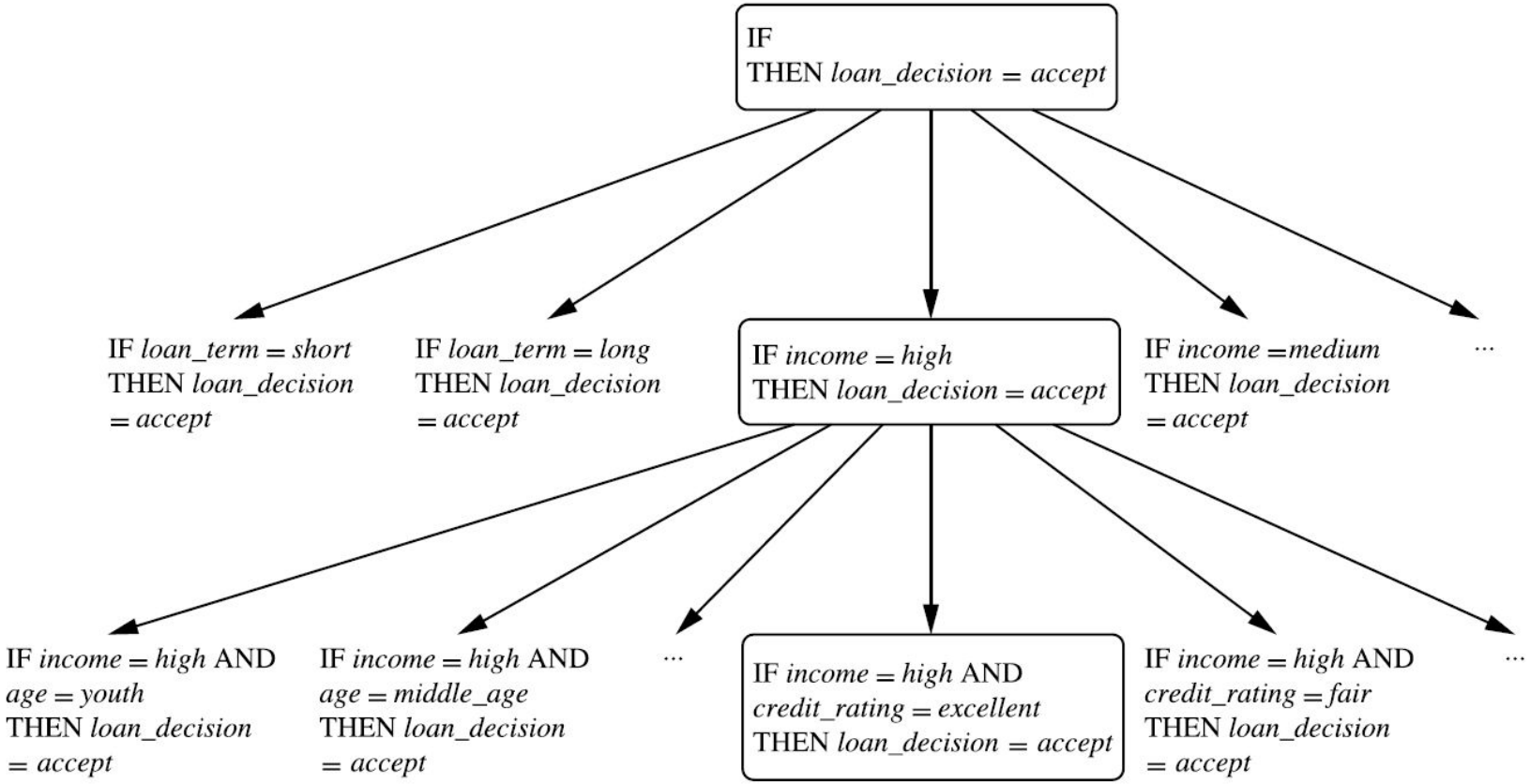


FIGURE 7.11

A general-to-specific search through rule space.

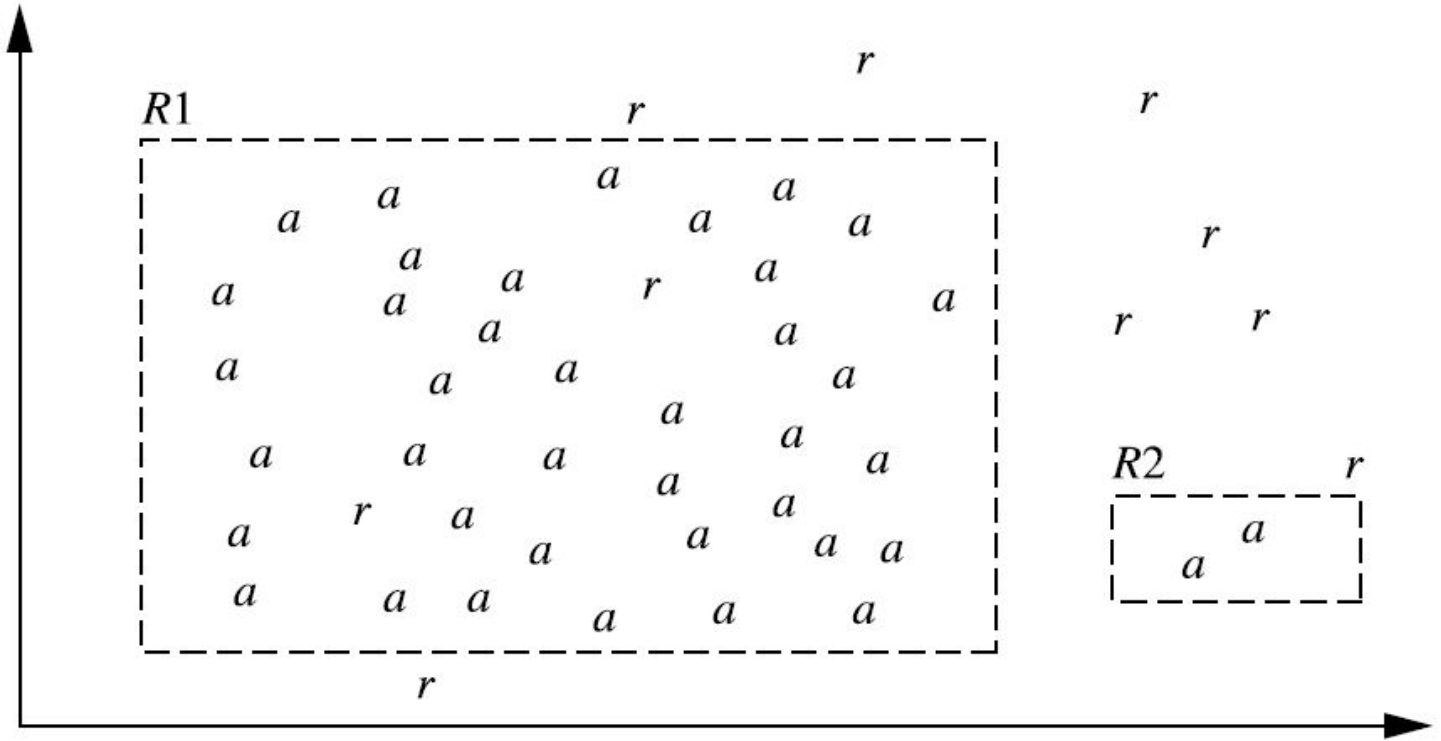


FIGURE 7.12

Rules for the class *loan_decision* = *accept*, showing *accept* (*a*) and *reject* (*r*) tuples.

Example 7.4. Rule accuracy and coverage. Let's go back to our data in Section 6.2, Table 6.1. These are class-labeled tuples from the *AllElectronics* customer database. Our task is to predict whether a customer will buy a computer. Consider rule $R1$, which covers 2 of the 14 tuples. It can correctly classify both tuples. Therefore $coverage(R1) = 2/14 = 14.28\%$ and $accuracy(R1) = 2/2 = 100\%$.

□

Example 7.5. Extracting classification rules from a decision tree. The decision tree of Fig. 6.2 can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree. The rules extracted from Fig. 6.2 are as follows:

$R1$: IF *age* = *youth*	AND *student* = *no*	THEN *buys_computer* = *no*
$R2$: IF *age* = *youth*	AND *student* = *yes*	THEN *buys_computer* = *yes*
$R3$: IF *age* = *middle_aged*		THEN *buys_computer* = *yes*
$R4$: IF *age* = *senior*	AND *credit_rating* = *excellent*	THEN *buys_computer* = *yes*
$R5$: IF *age* = *senior*	AND *credit_rating* = *fair*	THEN *buys_computer* = *no*.

□

Example 7.6. Choosing between two rules based on accuracy. Consider the two rules as illustrated in Fig. 7.12. Both are for the class *loan_decision* = *accept*. We use "*a*" to represent the tuples of class "*accept*" and "*r*" for the tuples of class "*reject*." Rule *R*1 correctly classifies 38 of the 40 tuples it covers. Rule *R*2 covers only two tuples, which it correctly classifies. Their respective accuracies are 95% and 100%. Thus *R*2 has greater accuracy than *R*1, but it is not the better rule because of its small coverage. □

The Core Idea of Sequential Covering

Extract, Remove, Repeat

The Concept:

Instead of building a massive tree, the algorithm learns rules one at a time. Each rule “covers” (explains) a specific subset of the positive training examples. Once covered, those examples are removed, and the process repeats for the remaining data.

The Algorithm Logic:

```
Plaintext
1. Initialize rule set R = empty
2. While training examples remain:
3.   Learn one rule (r) that covers some positive examples
4.   Add r to R
5.   Remove all examples covered by r
6. End While
7. Return R
```

The “Learn-One-Rule” Function

Building the Individual Rule

Algorithms like CN2 and RIPPER rely on a sub-function to build each individual rule step-by-step.

The Process:

Start with an empty rule (covers all examples)

Greedily add specific conditions

e.g., AND Humidity = High

Mathematically improve the rule’s quality

Evaluation Heuristics:

To determine if adding a condition actually improves the rule, the algorithm evaluates candidates using heuristics such as:

- Entropy

- Overall Accuracy

- FOIL Information Gain

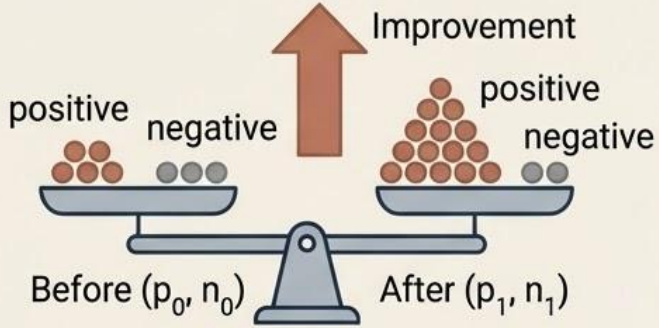
Evaluating Rules with FOIL Gain

The Mathematics of Improvement

FOIL (First-Order Inductive Learner) Information Gain strictly measures how much a new condition improves a rule's ability to isolate positive examples.

The Formula:

$$FOIL\_Gain = p_1 \times \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$



The Variables:

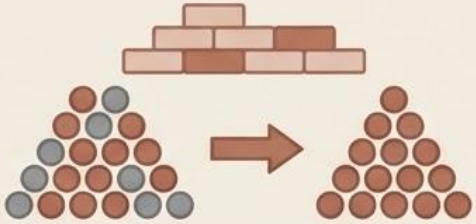
- p_0, n_0: The number of positive and negative examples covered before adding the new condition.
- p_1, n_1: The number of positive and negative examples covered after adding the new condition.

The RIPPER Algorithm

Repeated Incremental Pruning to Produce Error Reduction

RIPPER is one of the most popular and efficient sequential covering algorithms. It builds and refines rules in three distinct phases:

Growth



Greedily add conditions to the rule until it covers absolutely no negative examples (perfectly isolating the positives).

Pruning

Immediately remove the final conditions to optimize the rule's performance on a separate validation set, preventing overfitting.

Optimization

For every rule in the final set, explicitly consider replacements and revisions to ensure the most compact logic possible.

Evaluating Sequential Covering

The Strengths and Weaknesses

The Advantages

- Directly learns highly interpretable, human-readable rules.
- Exceptionally good at handling noisy data (thanks to aggressive pruning).
- Produces highly compact rule sets compared to full decision trees.

The Disadvantages

- Because it uses a ‘greedy search’ to build rules step-by-step, it may miss the mathematically optimal rule set.
- The order in which the rules are generated heavily impacts the final classification, creating strict dependencies.

SLIDE 1: ASSOCIATIVE CLASSIFICATION

Mining for Rules

Combining Association Mining with Classification

Part 2 (Continued) - Leveraging frequent patterns in data to determine class membership.

SLIDE 2: THE CORE CONCEPT

Class Association Rules (CARs)

The Concept: Associative classification bridges the gap between unsupervised pattern discovery and supervised learning. It generates Class Association Rules (CARs) where the consequent of the rule is always a predefined class label.

The Key Insight: Frequent patterns in a dataset are often highly indicative of class membership. If certain items frequently appear together in a specific class, they form a reliable predictive rule.

SLIDE 3: THE CBA ALGORITHM FRAMEWORK

Classification Based on Associations

The CBA framework builds the classifier in four distinct steps:

1. **Rule Generation:** Mine all frequent itemsets (often using the Apriori algorithm) forcing the class label to be the consequent. Generate rules in the form: *itemset* → *class*. Apply strict minimum support and confidence thresholds.
2. **Rule Pruning:** Remove redundant or inferior rules, keeping only the highest confidence rules for each distinct itemset.
3. **Rule Sorting**: Sort the surviving rules strictly by Confidence, then by Support, and finally by Length (ensuring higher confidence rules are evaluated first).
4. **Classifier Building**: Select the minimal set of rules that completely covers the training data, and establish a default rule for any uncovered cases.

SLIDE 4: A PRACTICAL EXAMPLE

From Transactions to Rules

Imagine a dataset of grocery transactions labeled by customer type (Class A or Class B).

Transaction Data:

T1: {Milk, Bread} → Class A
T2: {Milk, Eggs} → Class A
T3: {Bread, Butter} → Class B
T4: {Milk, Bread, Butter} → Class B

Mined Class Association Rules (CARs):

{Milk} → A (support = 2, confidence = 0.67)
{Bread} → B (support = 2, confidence = 0.67)
{Milk, Bread} → B (support = 1, confidence = 0.5)
{Milk, Bread, Butter} → B (support = 1, confidence = 1.0)

SLIDE 5: EVALUATING ASSOCIATIVE CLASSIFICATION

Strengths and Limitations

The Advantages:

- Efficiently leverages existing, highly optimized association mining algorithms (like Apriori or FP-Growth).
- Produces highly interpretable rules that make logical sense to domain experts.
- Often yields surprisingly high classification accuracy on transactional or categorical data.

The Disadvantages:

- Can easily generate massive, overwhelming rule sets if the data is highly dimensional.
- Critically dependent on threshold tuning; setting the minimum support or confidence too low causes a combinatorial explosion.

SLIDE 1: DISCRIMINATIVE FREQUENT PATTERN-BASED CLASSIFICATION

Finding the Differences

Focusing on Patterns That Truly Discriminate

Part 2 (Continued) - Moving beyond basic associations to find the most powerful rules.

SLIDE 2: THE CORE PROBLEM AND SOLUTION

Beyond Simple Association

The Limitation:

- A major flaw of standard Associative Classification is that it generates a massive number of rules that are completely redundant or non-discriminative (they don't actually help separate the classes).

The Solution:

- We must shift the focus strictly to the specific patterns that best discriminate between the different classes, ignoring patterns that are common across the board.

SLIDE 3: KEY APPROACHES TO FINDING PATTERNS

Mining and Generation

- DDPMine (Direct Discriminative Pattern Mining): Actively mines for patterns that appear very frequently in one class, but rarely in others. It evaluates discriminative power using information gain or Fisher scores, often integrating directly with an SVM for the final classification.
- Feature Generation: Treats the discovered discriminative patterns as entirely new features for the dataset. You can then build standard classifiers (like SVMs or logistic regression) on top of this newly created pattern-based feature space.
- Contrast Pattern Mining: Focuses purely on finding patterns that have a significantly different level of support across classes, using ratio-based or statistical tests for selection.

SLIDE 4: MEASURING DISCRIMINATIVE POWER

The Mathematical Metrics

To mathematically prove that a pattern is actually discriminative, the algorithm evaluates it using specific measures:

Measure	Definition
– Growth Rate	The ratio of support across classes: $\frac{support_{class1}}{support_{class2}}$
– Information Gain	The total reduction in entropy after applying the pattern.
– Gini Index	The measurement of impurity reduction.
– Fisher Score	The ratio of between-class variance to within-class variance.

SLIDE 5: THE ADVANTAGES OF DISCRIMINATION

Why This Method Excels

- **Compactness:** Produces much more compact and manageable rule sets compared to standard associative classification.
- **Focus:** Ignores noise and focuses purely on the patterns that truly differentiate the classes.
- **Generalization:** Because it removes redundant and overlapping rules, it often results in significantly better generalization on unseen data.

PART 3: CLASSIFICATION WITH WEAK SUPERVISION

Doing More with Less

Semisupervised Classification

Part 3- Tackling the real-world challenge of scarce labeled data.

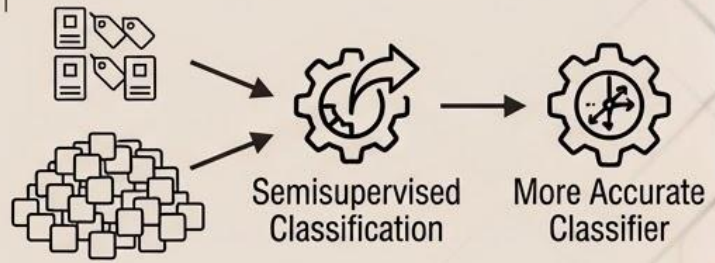
SLIDE 2: THE CORE PROBLEM AND GOAL

The Value of Unlabeled Data

The Problem:

In real-world AI applications, perfectly labeled data is notoriously expensive, time-consuming to create, and often scarce. Conversely, raw, unlabeled data is abundant and cheap.

The Goal:

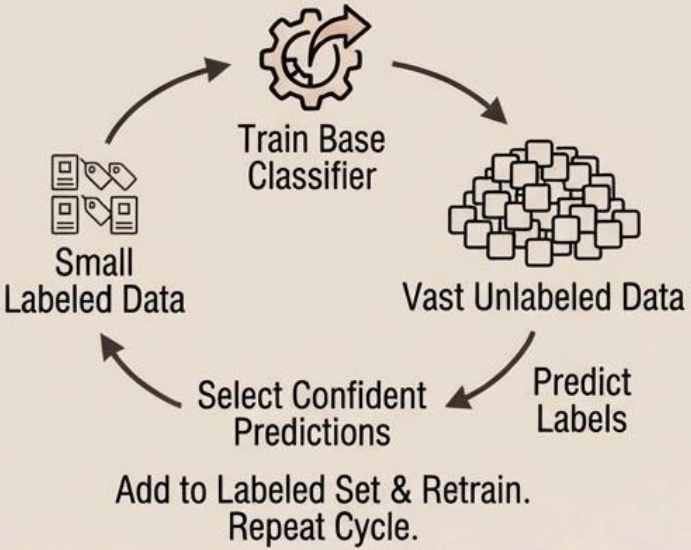


Semisupervised classification attempts to use both a small set of labeled data and a massive pool of unlabeled data to build a more accurate classifier than using the labeled data alone.

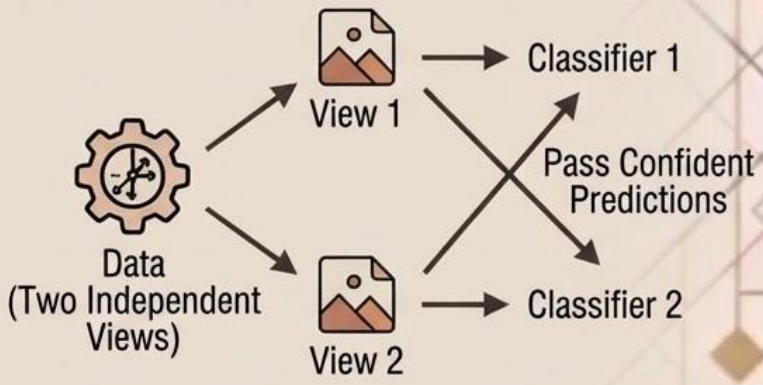
SELF-TRAINING VS. CO-TRAINING

Iterative Learning Strategies

Self-Training



Co-Training

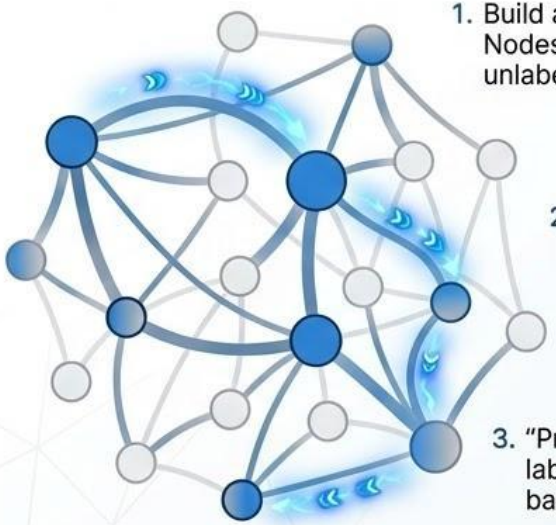


Constraint: This only works well if the two views are truly independent of each other.

Graph-Based and Generative Models

Advanced Semisupervised Techniques

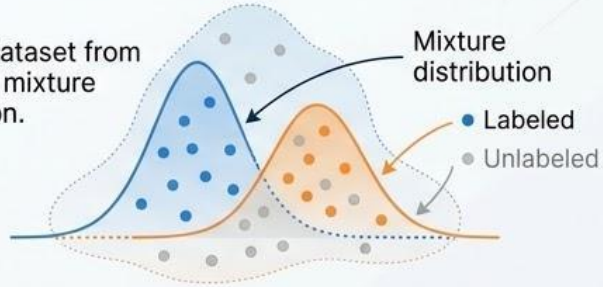
Graph-Based Methods



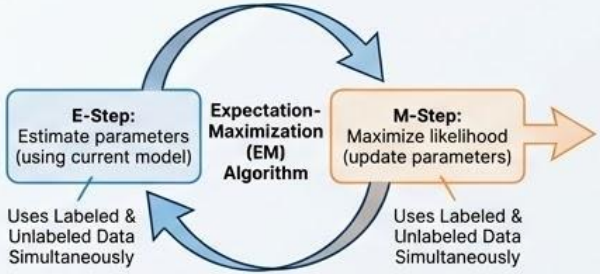
1. Build a mathematical graph: Nodes represent labeled & unlabeled examples.
2. Connect nodes with similarity-weighted edges.
3. "Propagate" known labels to unlabeled nodes based on proximity.

Generative Models

1. Assume dataset from a specific mixture distribution.



2. Utilize EM algorithm to iteratively estimate parameters using all data.



Evaluating Semisupervised Learning

The Trade-Offs of Weak Supervision

The Advantages:

Massively leverages abundant, cheap unlabeled data, drastically reducing data annotation costs.

Can significantly improve overall model accuracy even when starting with just a handful of labels.

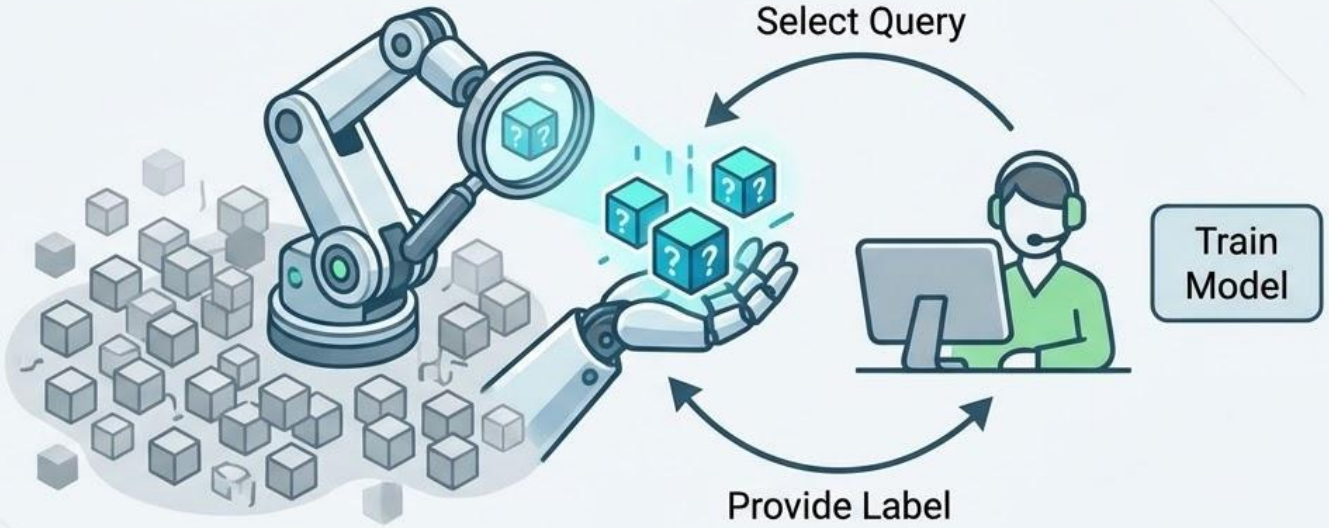
The Disadvantages:

Relies heavily on strong underlying assumptions about the data distribution.

Carries a high risk of “confirmation bias” (especially in self-training), where the model confidently reinforces its own errors over multiple training iterations.

Active Learning

Asking for Help: Minimizing Labeling Effort through Strategic Queries



Part 3 (Continued) - How algorithms proactively choose the data they want to learn from.

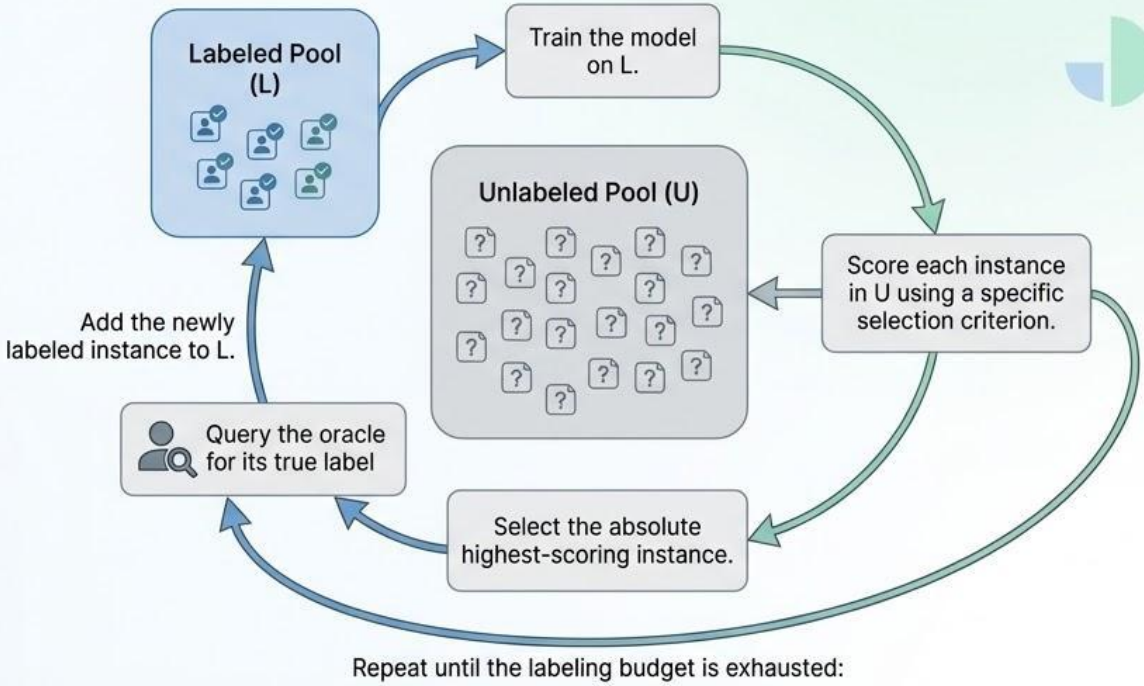
The Core Concept and Framework

Targeted Learning

The Concept:

Instead of randomly labeling data, the algorithm actively selects which specific instances an "oracle" (usually a human expert) should label. The goal is to maximize accuracy while minimizing the overall labeling effort and budget.

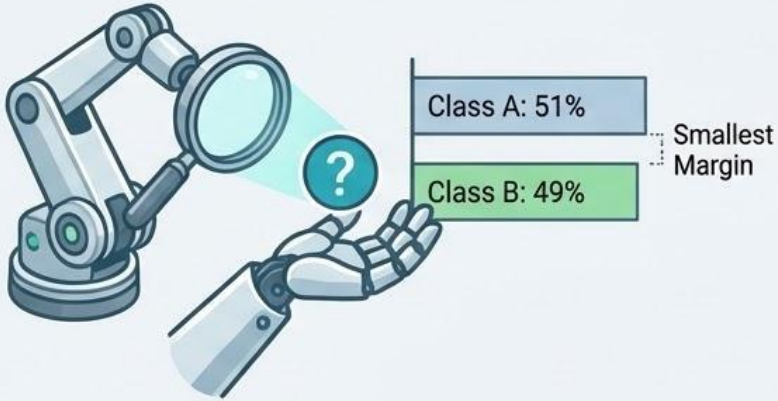
Pool-Based Active Learning Framework:



Core Query Strategies

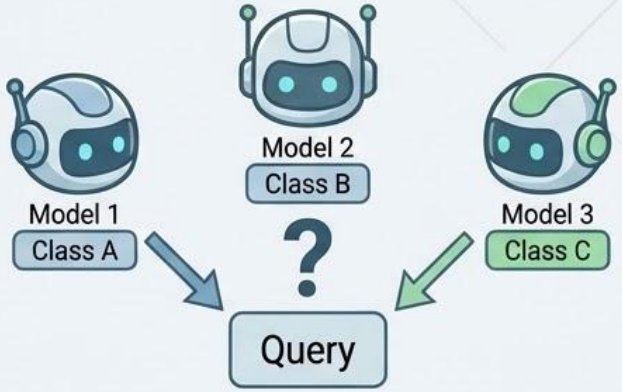
How Does the Model Choose?

Uncertainty Sampling



The model queries instances where it is currently the least confident.
Example: For probabilistic models, this means finding the data points with the smallest margin of probability between the top two predicted classes.

Query-by-Committee

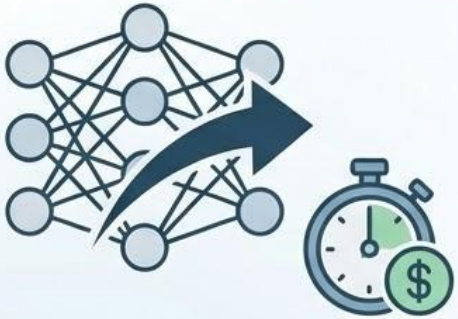


Maintain multiple distinct models (a 'committee'). The algorithm queries the instances where these models disagree with each other the most. Disagreement is mathematically measured using metrics like vote entropy or KL divergence.

Advanced Query Strategies

Optimizing for Impact and Density

Expected Model Change



The algorithm queries instances that would cause the most drastic mathematical change to the current model if their true label were known. (Note: This is often computationally expensive).

Expected Error Reduction

Queries instances that are estimated to minimize the expected future error across the entire remaining unlabeled pool.

Density-Weighted Methods

A hybrid approach that combines uncertainty with representativeness. It specifically prefers to query instances that are located in dense regions of the data, ensuring the model isn't wasting its budget learning about irrelevant outliers.

Evaluating Active Learning

Strategic Data Selection

The Advantages

- Drastically minimizes the raw cost of human labeling.

- Employs highly strategic data selection, focusing human effort only where the model actually needs help.

The Disadvantages

- The heavily iterative process of training, scoring, querying, and retraining can be computationally slow.

- If relying purely on uncertainty sampling without density weighting, the algorithm may waste the labeling budget by repeatedly selecting unrepresentative outliers.

PART 3 (CONCLUSION)

TRANSFER LEARNING

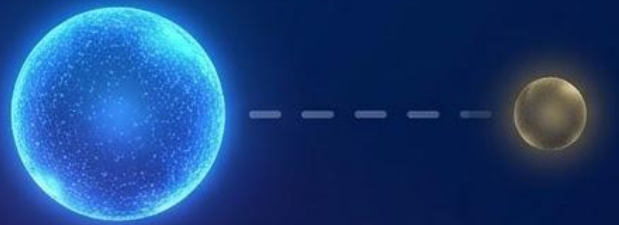
Borrowing Knowledge: Leveraging Data Across Different Domains.

How to solve the problem of limited labeled data by transferring existing knowledge.

The Core Problem and Concept

Don’t Start from Scratch

The Problem



Source Domain
(Abundant Data)

Target Domain
(Limited Data)

In machine learning, we often encounter scenarios where we have an abundance of perfectly labeled data in one specific domain (the "source"), but severely limited data in a new, related domain (the "target").

The Solution

Instead of training a new model from scratch, we use Transfer Learning to extract the knowledge the model learned from the source domain and actively transfer it to the target domain.

Types of Transfer Learning

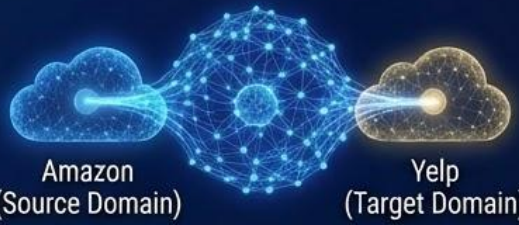
How Knowledge Crosses Over

Depending on the relationship between the tasks and domains, transfer learning falls into three main categories:

Inductive Transfer

The source and target tasks are different, but mathematically related.
Example: A model trained to classify cars is adapted to classify trucks.

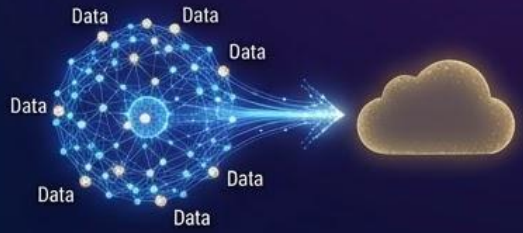
Transductive Transfer



The core task remains exactly the same, but the domains are different.

Example: Transferring a sentiment analysis model trained on Amazon product reviews to evaluate Yelp restaurant reviews.

Unsupervised Transfer



There is absolutely no labeled data in the target domain, so the algorithm must focus strictly on transferring underlying feature representations.

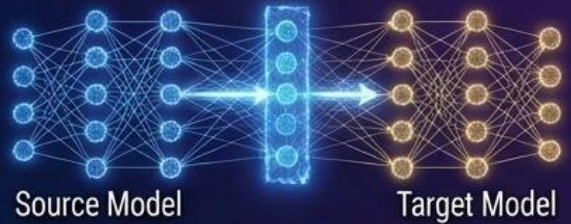
Core Approaches to Transfer

The Mechanisms of Learning

Instance Transfer

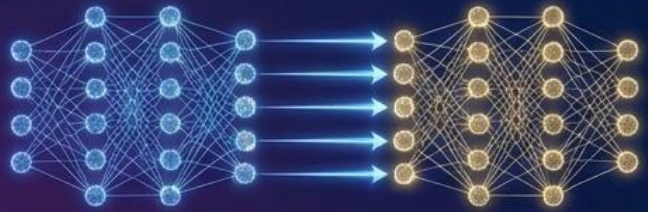
Mathematically re-weighting the source instances so they better match the expected distribution of the target domain.

Feature Representation Transfer



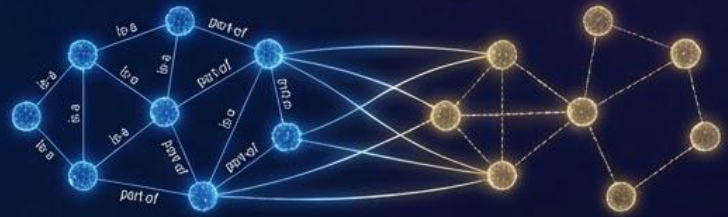
Forcing the model to learn “domain-invariant” features that remain universally true across both datasets.

Parameter Transfer



Directly sharing the underlying model parameters (weights and biases) across the different domains.

Relational Knowledge Transfer



Mapping and transferring the logical relationships between entities, rather than the raw data itself.

Domain Adaptation in Action

A Real-World Example

Domain Adaptation is a special, highly common case of transfer learning where the source and target domains differ, but the specific task remains identical.

The Scenario: Sentiment Analysis

The Source:
A massive database of movie reviews (abundant labeled data).

The Transfer Strategy: The algorithm learns "domain-invariant" sentiment features from the movie reviews—such as the structural use of adjectives like "terrible" or "fantastic"—and applies those universal rules to successfully classify the product reviews.

The Target:
A new database of specific tech product reviews (severely limited labels).

DISTANT SUPERVISION

Automating the Annotation

Generating Training Labels Without Manual Effort

Part 3 (Continued) - How to scale dataset creation using heuristics and external knowledge bases.

The Core Concept and Applications

Zero Manual Labeling

The Concept: Distant supervision completely bypasses the need for human annotators. It generates training labels automatically by applying heuristics, rules, or existing external knowledge bases directly to raw, unlabeled data.

Common Applications:

Relation Extraction

Identifying how entities in a text relate to each other.

Named Entity Recognition (NER)

Automatically tagging people, places, and organizations.

Information Extraction

Pulling structured facts out of unstructured text.

How Distant Supervision Works

From Rules to Models

1. Define Heuristics/Rules

Create logical rules based on domain knowledge. (Example: "If a sentence contains 'was born in' followed by a location, automatically label it as a BIRTH_PLACE relation.")

2. Apply to Unlabeled Data

Run these heuristics across massive datasets to generate a large volume of "noisy" labeled examples.

3. Train the Classifier

Train a machine learning model on this generated data. The model learns the underlying patterns despite the inherent noise in the labels.

4. Refine

Continuously use multiple overlapping heuristics and rely on the model's confidence scores to filter out the most severely noisy labels.

A Practical Example

Relation Extraction in Action

How do we turn existing knowledge into new training data?

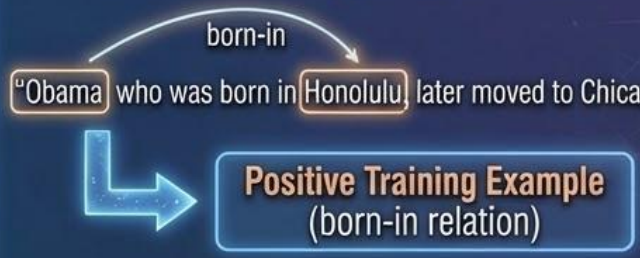
The Knowledge Base (KB) Fact

(Barack Obama, born-in, Honolulu)

The Raw Sentence

"Obama, who was born in Honolulu, later moved to Chicago."

The Distant Supervision Action



The algorithm aligns the known tuple from the knowledge base with the unannotated sentence. Because the entities match, it automatically generates a positive training example for the born-in relation, teaching the model to recognize that linguistic structure.

The Snorkel Framework and Evaluation

Managing the Noise

The Snorkel Framework

A popular system for distant supervision. It requires users to write distinct 'labeling functions', mathematically combines and denoises those functions using a generative model, and then uses that cleaned data to train the final discriminative classifier.

The Advantages

- Requires absolutely zero manual labeling effort.
- Infinitely scalable to massive, web-scale datasets.
- Highly effective at leveraging existing, structured knowledge bases.

The Disadvantages

- Inherently produces noisy and imperfect labels.
- Rigid heuristics may miss subtle nuances or mislabel complex sentences.
- Writing effective, high-quality labeling functions requires deep domain expertise.

The Classification Arsenal

Choosing the Right Strategy for Your Data

Final - Synthesizing the methods and understanding when to deploy each technique.

Method Comparison (Full Supervision)

Rule-Based and Pattern-Based Methods

When you have fully labeled data but need high interpretability or pattern discovery, these methods excel:

Method	Label Requirement	Interpretability	Key Strength
Rule Extraction	Full supervision	Very High	Generates explainable models directly from trees.
Sequential Covering	Full supervision	High	Learns rules directly from the data.
Associative Classification	Full supervision	High	Efficiently leverages frequent itemset patterns.
Discriminative Pattern	Full supervision	Medium	Focuses strictly on class differences, reducing noise.

Method Comparison (Weak Supervision)

Learning with Limited Labels

When perfectly labeled data is scarce or non-existent, these methods allow you to scale your models:

Method	Label Requirement	Interpretability	Key Strength
Semi-Supervised	Few labels + unlabeled	Varies	Massively leverages cheap, unlabeled data.
Active Learning	Few labels (strategic)	Varies	Minimizes the financial cost of human labeling.
Transfer Learning	Source labeled, target few	Varies	Borrows cross-domain knowledge to solve new tasks.
Distant Supervision	No manual labels	Varies	Automates labeling using heuristics and knowledge bases.

Strategic Selection Guide

When to Use Which Method

Choosing the right algorithm depends entirely on your data availability and your business constraints:

Your Scenario	Recommended Method
You need highly interpretable rules from an existing tree.	**Rule Extraction**
You need direct, human-readable rule learning.	**Sequential Covering**
You have abundant data and need pattern-based accuracy.	**Associative Classification**
Class discrimination is critical to the task.	**Discriminative Patterns**
You have very few labels, but massive amounts of unlabeled data.	**Semi-Supervised Learning**
Human labeling is too expensive; you must be strategic.	**Active Learning**
A related domain already has abundant labels you can use.	**Transfer Learning**
You have a large unlabeled corpus and an existing Knowledge Base.	**Distant Supervision**

Key Insights and Takeaways

The Big Picture

Interpretability Matters

Rule-based methods provide the transparency and explainability that are essential for many real-world, high-stakes AI applications.

Bridging the Gap

Pattern-based classification successfully bridges the gap between unsupervised association mining and supervised classification.

The Reality of Data

Weak supervision is not just an academic exercise; it is absolutely essential for real-world scalability where data is messy and labels are expensive.

The Ultimate Trade-Off

The core trade-off between label quality and label quantity will always drive your method selection. However, combining these approaches (e.g., using distant supervision to generate a dataset, then training an interpretable rule-based classifier) often yields the best results.