Probabilistic Machine Learning
(theory and practice)

Charles Sutton
Introduction to Research in Data Science
University of Edinburgh
New methodology

New applications

- New model types
- Inference algorithms (e.g., high dimensional, streaming)
- Approximate learning methods

- Analyzing computer programs
- Data mining
- Exploratory data analysis
- Home energy demand
- Computer security
Syntactic Idioms in Code

... (c != null) {
    try {
        if (c.moveToFirst()) {
            number = c.getString(c.getColumnIndex(phoneColumn));
        }
    } finally {
        c.close();
    }
}
...

(a)

try {
    if ($(Cursor).moveToFirst()) {
        $BODY$
    }
} finally {
    $(Cursor).close();
}

(b) (c) Eclipse JDT's AST for the code in (a). Shaded nodes are objects, successfully mined by aggis.

3. MINING CODE IDIOMS

We will build up step by step. First, we will describe our proposed method for the idiom mining algorithm, which identifies.

3.1 Probabilistic Grammars

A PCFG is defined as $G_{\mathcal{R}} = (\mathcal{N}, \mathcal{S}, \mathcal{R}, P, \mathcal{X})$, where $\mathcal{N}$ is a set of nonterminals, $\mathcal{S}$ is the root of the grammar, $\mathcal{R}$ is a set of productions, $P$ is a set of probabilities, and $\mathcal{X}$ is a set of strings from the known grammar. These distributions will represent trees from the known grammar. These distributions will represent distributions over ASTs which are called probabilistic tree substrings (PTSGs) for which rules from the grammar are used more often, and, crucially, which sets of rules tend to be used contiguously. Our aim is different, because rather than a standard deterministic CFG, Probabilities provide a set of nonparametric Bayesian methods of the trees that yield expressions: $P(\mathcal{X} | \mathcal{R})$. The probability of generating a particular tree $x \in \mathcal{X}$ is the product over all rules that are required to generate $x$,

The probability of the tree $x$ is $P(x | G_{\mathcal{R}}) = \sum_{x' \in \mathcal{X}} P(x' | G_{\mathcal{R}}) P(x' | x'')$, where $x''$ is a particular tree in $\mathcal{X}$.

When of course the grammar of the programming language is already known. We clarify that our aim is to define a distribution over the strings of a context-free language. This encourages the method to avoid identifying idioms. We do this by adding a new parameter to the model (the rule's probability of appearing), and the number of times finite state machines, of method invocations. Although API patterns that are inferred are essentially sequences, or some of the type of pattern that H352 identifies.

Finally, when the maximum size of the model is unbounded. This is exactly different, because nonparametric Bayesian methods require special iteration logic, or a Java library that requires the handling...

...for those included in the idiom. (d) An example of a PTSG rule for a simple expression grammar. See text for more details.

Figure 1: Example of code idiom extraction: (a) A snippet from android.telephony.android.database.Cursor.

Allamanis and Sutton, FSE 2014
Example Idioms

From: Nonparametric Bayesian Tree Substitution Grammar
[Post and Gildea, 2009; Cohn et al, 2010]

channel=connection. createChannel();

(a)

catch (Exception e){
	$(Transaction).failure();
}

(d)

Location.distanceBetween(
	$(Location).getLatitude(),
	$(Location).getLongitude(),
	$...);

(g)

ConnectionFactory factory =
	new ConnectionFactory();

(j)

$kmethodInvoc();

Connection connection =
	factory.newConnection();

(k)

Elements $name=$(Element).

(b)

select($StringLit);

Transaction tx=ConnectionFactory.

getDatabase().beginTx();

(c)

SearchSourceBuilder builder=

getQueryTranslator().build(
	$(ContentIndexQuery));

(f)

LocationManager $name =
	(LocationManager) getSystemService(
	Context.LOCATION_SERVICE);

(i)

try{
	$BODY$
}

finally{
	$(RevWalk).release();
}

Node $name=$methodInvoc();

$BODY$

finally{
	$(Transaction).finish();
}

(h)

while ($(ModelNode) != null){
	if ($(ModelNode) == limit)
		break;

	$(ModelNode)=$(ModelNode)
		.getParentModelNode();
}

(k)

Document doc=Jsoup.connect(URL).

userAgent("Mozilla").

header("Accept","text/html").

get();

(l)

Allamanis and Sutton, FSE 2014
Bayesian Melding

[Poole and Raftery, 2000]

Deterministic simulation \( \tau = f(S) \).

- e.g., \( S \) contains population at time 0 and growth rate \( \tau \) is population at time \( T \)

Two sources of information about \( \tau \)

1. Information about \( S \), which implies information about \( \tau \) via \( f \)
   \[
   p_S(S) \xrightarrow{f} p^\tau(\tau)
   \]

2. Direct measurements of \( \tau \)
   \[
   p_\tau(\tau)
   \]

How to combine? Geometric average.
   \[
   \tilde{p}_\tau(\tau) \propto p^\tau_\tau(\tau)^\alpha p_\tau(\tau)^{1-\alpha}
   \]

How does this yield a distribution on \( S \)?
   \[
   \tilde{p}_S(S) = c_\alpha p_S(S) \left( \frac{p_\tau(f(S))}{p^\tau_\tau(f(S))} \right)^{1-\alpha}
   \]
Latent Bayesian Melding

[Zhong, Goddard, Sutton, NIPS 2015]

Instead, model using a latent variable

\[ p_\tau(\tau) = \int p_\xi(\xi)p(\tau|\xi)d\xi. \]

Ex of latent: Maybe different subpopulations have different growth rates.

Following the standard BM approach intractable, so we take an approximation:

\[ \tilde{p}_S(S) \approx \max_\xi \tilde{p}_{S,\xi}(S, \xi) = \max_\xi c_\alpha p_S(S) \left( \frac{p_\tau(f(S)|\xi)p(\xi)}{p_\tau(f(S))} \right)^{1-\alpha} \]

This yields an integer linear program that we can relax (for our application)
Energy disaggregation (NILM) by integrating short-term and long-term information

Mingjun Zhong, Nigel Goddard, Charles Sutton
School of Informatics, University of Edinburgh, United Kingdom
{mzhong,ngoddard,csutton}@inf.ed.ac.uk

THE IDEAL PROJECT
ENERGY DISAGGREGATION
Mains (Mixture) Disaggregation (Separation) Appliances (Sources)

PROBLEMS TO SOLVE
Given only the mains, we are interested in these questions:
1. Short-term information (fine-grained, e.g., each time point)
   a. Was the appliance on?
   b. How much energy was consumed?
2. Long-term information (coarse-grained, e.g., one day)
   a. What appliances were used?
   b. When the appliances were using?
   c. How many times the appliances were used?
   d. How long the appliances were used?
   e. How much energy in total was used by appliance?

RESULTS
SHORT-TERM INFORMATION

Appliance i
Appliance j
Y_t
Y_t-1
Y_t+1
si,t
si,t-1
si,t+1
sj,t
sj,t-1
sj,t+1

LONG-TERM INFORMATION: SUMMARY STATISTICS

Kettle (5861 day samples)

Days
Counts of Days
Energy (deci-Watt-hours)
Energy Used
Duration (minute)
Duration of 'ON'

Washing Machine (499 day samples)

Days
Nos of Cycles
Counts of Days
Energy (deci-Watt-hours)
Nos of Cycles
Energy Used
Duration (minute)
Nos of Cycles

LATENT BAYESIAN MELDING
Random Variables describing each time point (fine-grained)
Random Variables describing summary statistics (coarse-grained)
Prior distribution
Prior distribution
Deterministic map
Melded Prior distribution

REFERENCES & ACKNOWLEDGEMENT

This work is supported by the Engineering and Physical Sciences Research Council (Grant numbers EP/K002732/1, EP/M008223/1).
Given only the mains, we are interested in these questions:

1. How much energy in total was used by appliance? Was the appliance on? When the appliances were using?

**RESULTS**

**Appliances**

- Short-term and long-term information

This work is supported by the Engineering and Physical Sciences Research Council

---

Model 1

\[ p_S(S) \]

\[ f \]

count how many times appliance turns on in \( S \)

count how much energy appliance uses in \( S \)

Model 2

\[ p_\tau(\tau) = \int p_\xi(\xi)p(\tau|\xi)d\xi. \]
• **Using machine learning to make programming better**
  • ML / NLP for programming languages
  • Combining program analysis with probabilistic machine learning
  • Find patterns in program executions: debugging

• **Using machine learning to make machine learning better**
  • Deep learning: Combining neural networks with prior knowledge
    • “interpretability bias”
  • Learning how to clean data
  • Interactive machine learning
  • Tools for monitoring models over time
  • Unsupervised and weakly supervised learning

• **Using machine learning to make the world better**
  • ML for computer security, NLP, sustainable energy…