Probabilistic Abductive Logic Programming using Dirichlet Priors

Calin Rares Turliuc, Luke Dickens, Alessandra Russo and Krysia Broda

Imperial College London

prepared for PLP 2015 `http://stoics.org.uk/plp/plp2015/`

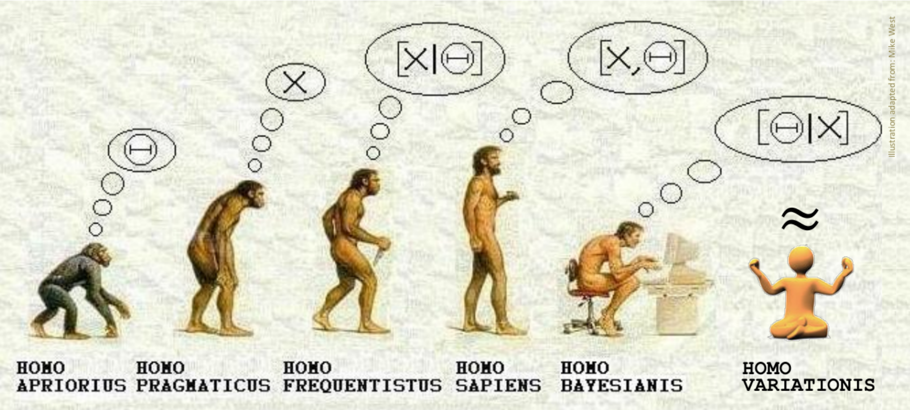


Figure: Credit: from Kay H. Brodersen's presentation on Variational Bayesian Inference.

Why probabilistic programming?

Goal: Generalized inference for a class of probabilistic models.

Probabilistic program = "generative story", execution = inference.

The traditional way:

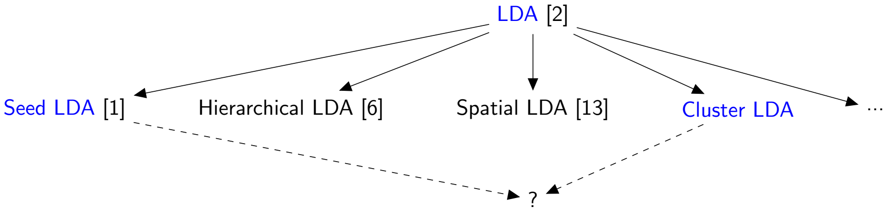
realization = model development + derivation of inference algo. + implementation + execution

Using probabilistic programming:

realization = model development + execution

⇒ Less effort, less time, more models!

E.g. Latent Dirichlet Allocation (LDA) and variations



Why another probabilistic programming language?

Probabilistic Programming

- tendency towards Universal Probabilistic Programming
- typically based on functional programming (Scheme for Church [5], Clojure for Anglican [11], etc.)
- inference in some discrete models, e.g. LDA, can be slow and doesn't always converge

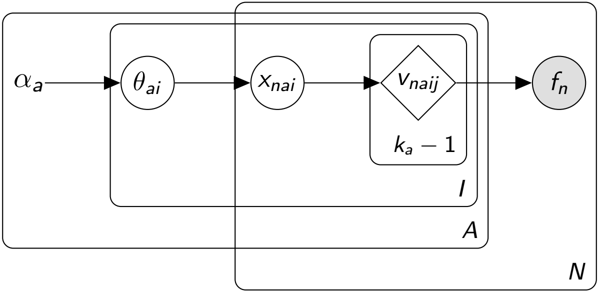
Probabilistic Logic

- [7] describes generalized inference for discrete models with Dirichlet priors, but no probabilistic programming
- typical probabilistic logic programming languages don't support Dirichlet priors
- Markov logic has an inherent limitation: no recursive definitions.

$\Rightarrow$ `peircebayes`

peircebayes (PB)

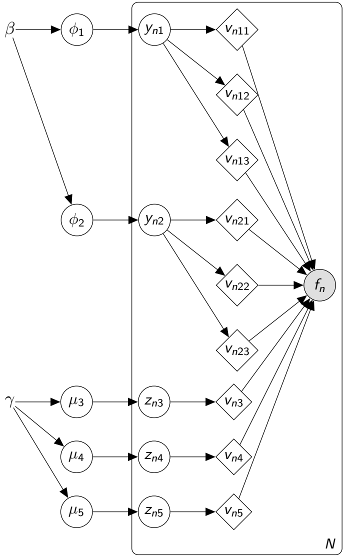
– a probabilistic abductive logic programming language designed for inference in probabilistic models with categorical variables and Dirichlet priors.



- α are parameters of Dirichlet distributions
- $\theta \sim \text{Dirichlet}(\alpha)$ and $x \sim \text{Cat}(\theta)$
- v are boolean encodings of x (i.e. annotated disjunction compilation)
- f are outputs of boolean functions *Bool* of v. The functions are computed using abduction.

Latent Dirichlet Allocation (LDA)

- LDA is well studied model for topic modelling [2].
- Corpus = set of D documents.
- A document is a bag of words.
- V is the size of the vocabulary, # of unique words in the corpus.
- Each word is sampled from one of T topics.
- A topic is a categorical distribution over V categories.
- A document is a mixture of topics.
- Task: infer topics and their mixtures per document.



LDA in peircebayes

```
observe(d(1), [ (w(1), 4), (w(4), 2) ]).
observe(d(2), [ (w(3), 1), (w(4), 5) ]).
observe(d(3), [ (w(1), 4), (w(2), 2) ]).

pb_dirichlet(1.0, mu, 2, 3).
pb_dirichlet(1.0, phi, 4, 2).

generate(Doc, Token) :-
    Topic in 1..2,
    mu(Topic, Doc),
    phi(Token, Topic).

pb_plate(
  [observe(d(Doc), TokenL),
    member((w(Token), Count), TokenL)],
    Count,
    [generate(Doc, Token)]).
```

LDA Experiment

Usenet comp.* dataset (only test documents):

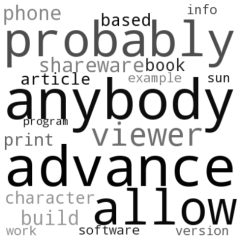
- 1911 documents
- 35850 unique words
- average document length: ~ 108 words
- 10 topics, priors: $\beta = 50/T$, $\gamma = 0.1$

LDA with seed constraints

```
observe(d(1), [ (w(1), 4), (w(4), 2) ]). % ...

pb_dirichlet(1.0, mu, 2, 3).
pb_dirichlet(1.0, phi, 4, 2).

seed_naf(Token) :- seed(Token, _).
seed(1, [1]). seed(4, [2]).

pb_plate(
  [observe(d(Doc), TokenL), member((w(Token), Count), TokenL),
     \+ seed_naf(Token)  ],
  Count,
  [Topic in 1..2, mu(Topic,Doc), phi(Token,Topic)]).
pb_plate(
  [observe(d(Doc), TokenL), member((w(Token), Count), TokenL),
     seed_naf(Token)  ],
  Count,
  [ seed(Token, TopicL), member(Topic, TopicL),
     mu(Topic,Doc), phi(Token,Topic) ]).
```

Seed LDA Experiment

- experiment inspired by [1]
- 4777 documents
- 27206 unique words (tokenize + WordNet lemmatizer)
- average document length: ~ 72 words
- 20 topics, priors: $\beta = 50/T$, $\gamma = 0.01$
- 2 topics are seeded: (hardware, machine, memory, cpu), and (software, program, version, shareware)



Cluster LDA Experiment

Data - all arXiv abstracts in 2007 in five categories.

Partition 25 topics in 5 clusters of 5 topics. The generative story of a word is: choose cluster, choose topic from cluster, choose word from topic.

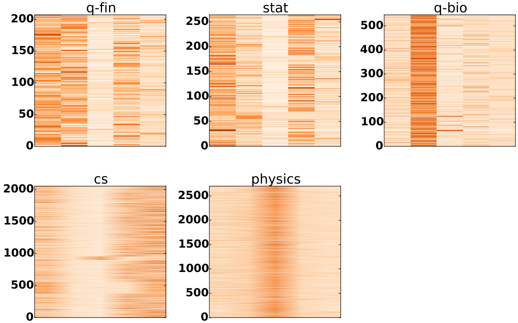
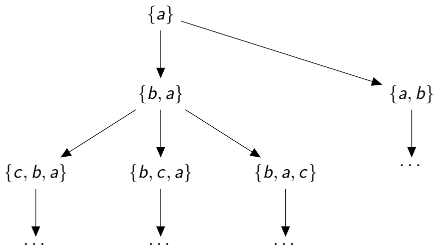


Figure: Cluster mixture for each category (x - topic clusters, y - documents, darker colour - higher probability).

Repeated Insertion Model (RIM) [4]



- Assume a totally ordered set S of N items.
- A preference is a permutation of S.
- A preference profile is a distribution over all the permutations of S.
- Assume K preference profiles.
- Observed data: A list of permutations of S.
- Task: Infer the parameters of the K distribution profiles and their mixture.

RIM Experiment

- 5000 persons surveyed for preference over 10 Sushi ingredients [8]
- experiment inspired by [9], $K = 6$ preference profiles, priors: $50/K$ for preference profile mixtures and 0.1 for all categorical distributions

$\pi_1 = 0.155$	$\pi_2 = 0.194$	$\pi_3 = 0.134$	$\pi_4 = 0.194$	$\pi_5 = 0.197$	$\pi_6 = 0.126$
fatty tuna	fatty tuna	fatty tuna	fatty tuna	fatty tuna	fatty tuna
shrimp	tuna	sea eel	sea urchin	tuna	shrimp
salmon roe	shrimp	tuna	salmon roe	shrimp	tuna
sea eel	squid	shrimp	shrimp	squid	sea eel
squid	egg	squid	sea eel	sea eel	squid
tuna	tuna roll	tuna roll	tuna	tuna roll	salmon roe
tuna roll	sea eel	salmon roe	tuna roll	salmon roe	tuna roll
sea urchin	cucumb. roll	sea urchin	squid	sea urchin	sea urchin
egg	salmon roe	egg	egg	cucumb. roll	egg
cucumb. roll	sea urchin	cucumb. roll	cucumb. roll	egg	cucumb. roll

Table: Mixture parameters and modes of preference profiles on the Sushi dataset.

Summary and Conclusions

We introduce `peircebayes` – a probabilistic abductive logic programming language tailored for inference in discrete models with Dirichlet priors.

We show 4 probabilistic models and experiments in `peircebayes`:

- ☑ vanilla LDA.
- ☑ LDA with seed constraints.
- ☑ cluster LDA.
- ☑ the repeated insertion model (RIM).

Future work:

- ☐ explore other models that fit the paradigm, e.g. Bayesian prevalence model [10], citation influence model [3].
- ☐ explore inductive learning in `peircebayes`.
- ☐ develop a non-parametric `peircebayes` based on work in Dirichlet Processes [12].
- ☐ improve scalability.

Thank You!

Online resources on `peircebayes`: `http://raresct.github.io`.

Towards PILP (Hypotheses)

- We can encode a PILP task in PB, assuming we can explicitly enumerate all the clauses in the language bias.
- Then, a probability distribution over the hypotheses $P(H)$ is specified with sample space $\Omega(H)$ as the superset of the set of clauses.
- Each clause is characterized by a Bernoulli distribution (with Beta prior) and the meaning of the binary sample space is that the clause is included or excluded from the hypothesis.
- We assume that all the random variables governing clause in/exclusion are independent.
- The probability of a hypothesis is then defined as the product of the probabilities of individual clauses.
- Linking back to the PB model, a hypothesis is a realization of x.

Towards PILP (Examples)

- The possible set of examples is constructed from a superset of a set of atoms, where positive examples are included in the set and negative examples are excluded.
- This is the sample space $\Omega(E)$ of $P(E)$.
- Each atom is endowed with a Bernoulli distribution over it being a positive or negative example.
- These random variables are independent, i.e. the probability of a set of examples is the product of the probabilities of individual examples.
- Each hypothesis maps to a set of examples, i.e. there is a surjective function $\models : \Omega(H) \rightarrow \Omega(E)$.
- Linking back to the PB model, assuming a set of examples e is the i-th observation, then the solutions of $Bool_i$ are the set $\models^{-1}(e)$.

Simple PILP Example in PB

```
p(g,m). p(g,d). p(h,t). p(h,m). p(t,e). p(n,e).
f(n). f(e). f(h). f(m).

pb_dirichlet(1.0, theta, 2, 8).

d(X,Y) :- p(X,Y), theta(2, 1).
d(X,Y) :- p(Y,X), theta(2, 2).
d(X,Y) :- f(X), theta(2, 3).
d(X,Y) :- f(Y), theta(2, 4).
d(X,Y) :- p(X,Y), f(X), theta(2, 5).
d(X,Y) :- p(X,Y), f(Y), theta(2, 6).
d(X,Y) :- p(Y,X), f(X), theta(2, 7).
d(X,Y) :- p(Y,X), f(Y), theta(2, 8).

pb_plate([], 1, [d(m,h), d(e,t), d(e,n), d(m,g),
    \+ d(t,h), \+ d(e,h), \+ d(g,n), \+ d(d,g)]).
```

Simple PILP Example in PB (2)

- *Bool* is $v_1 \wedge v_2 \wedge v_3 \wedge v_4 \wedge v_5$
- ... with an associated meaning of:
- Not Include Clause 3 AND Not Include Clause 8 AND Not Include Clause 4 AND Not Include Clause 2 AND Include Clause 7
- from the perspective of inference, the problem is trivial, and the posteriors are:
- $(1, 2)$ for Clause 7, $(2, 1)$ for Clauses 2,3,4,8 and $(1, 1)$ for Clauses 1,5,6

References I

[1] David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht.
A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic.
In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1171–1177, 2011.

[2] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty.
Latent dirichlet allocation.
Journal of Machine Learning Research, 3:2003, 2003.

[3] Laura Dietz, Steffen Bickel, and Tobias Scheffer.
Unsupervised prediction of citation influences.
In *In Proceedings of the 24th International Conference on Machine Learning*, pages 233–240, 2007.

References II

[4] Jean-Paul Doignon, Aleksandar Pekeč, and Michel Regenwetter.
The repeated insertion model for rankings: Missing link between two subset choice models.
Psychometrika, 69(1):33–54, 2004.

[5] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum.
Church: a language for generative models.
Uncertainty in Artificial Intelligence 2008, 2008.

[6] Thomas L. Griffiths, Michael I. Jordan, Joshua B. Tenenbaum, and David M. Blei.
Hierarchical topic models and the nested chinese restaurant process.
In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 17–24. MIT Press, 2004.

References III

[7] Masakazu Ishihata and Taisuke Sato.
Bayesian inference for statistical abduction using markov chain monte carlo.
In *Proceedings of the 3rd Asian Conference on Machine Learning, ACML 2011, Taoyuan, Taiwan, November 13-15, 2011*, pages 81–96, 2011.

[8] T. Kamishima, H. Kazawa, and S. Akaho.
Supervised ordering - an empirical survey.
In *Data Mining, Fifth IEEE International Conference on*, pages 4 pp.–, Nov 2005.

[9] Tyler Lu and Craig Boutilier.
Effective sampling and learning for mallows models with pairwise-preference data.
Journal of Machine Learning Research, 15:3783–3829, 2014.

References IV

[10] Myle Ott, Claire Cardie, and Jeff Hancock.
Estimating the prevalence of deception in online review communities.
In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 201–210, New York, NY, USA, 2012. ACM.

[11] Brooks Paige and Frank Wood.
A compilation target for probabilistic programming languages.
In *ICML*, 2014.

[12] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei.
Hierarchical dirichlet processes.
Journal of the American Statistical Association, 101, 2004.

References V

[13] Xiaogang Wang and Eric Grimson.
Spatial latent dirichlet allocation.
In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1577–1584. Curran Associates, Inc., 2008.

Cluster LDA

```
pb_dirichlet(10.0, psi, 5, 5769).
pb_dirichlet(10.0, theta1, 5, 5769). % ...
pb_dirichlet(10.0, theta5, 5, 5769).
pb_dirichlet(0.1, phi1, 26834, 5). % ...
pb_dirichlet(0.1, phi5, 26834, 5).
pb_plate(
  [observe(d(Doc), TokenL), member((w(Token), Count), TokenL)],
  Count, [generate(Doc, Token)]).
create_term(Functor, Idx, Cat, Distrib, Term) :-
  number_chars(Idx, LIdx), atom_chars(Functor, LFunctor),
  append(LFunctor, LIdx, LF), atom_chars(F, LF),
  Term =.. [F, Cat, Distrib].
generate(Doc, Token) :-
  Cluster in 1..5, Topic in 1..5, psi(Cluster, Doc),
  create_term(theta, Cluster, Topic, Doc, Term1), pb_call(Term1),
  create_term(phi, Cluster, Token, Topic, Term2), pb_call(Term2).
```

RIM in peircebayes (I)

```
observe([5,0,3,4,6,9,8,1,7,2]).    observe([0,9,6,3,7,2,8,1,5,4]).
% ... 4998 'observe' facts ommited

pb_dirichlet(8.33333333333, pi, 6, 1).
pb_dirichlet(0.1, p2, 2, 6).       pb_dirichlet(0.1, p7, 7, 6).
pb_dirichlet(0.1, p3, 3, 6).       pb_dirichlet(0.1, p8, 8, 6).
pb_dirichlet(0.1, p4, 3, 6).       pb_dirichlet(0.1, p9, 9, 6).
pb_dirichlet(0.1, p5, 5, 6).       pb_dirichlet(0.1, p10, 10, 6).
pb_dirichlet(0.1, p6, 6, 6).

pb_plate( [observe(Sample)], 1,
    [generate([0,1,2,3,4,5,6,7,8,9], Sample)] ).

generate([H|T], Sample):-
    K in 1..6, pi(K, 1), generate(T, Sample, [H], 2, K).
```

RIM in peircebayes (II)

```
generate([], Sample, Sample, _Idx, _K).
generate([ToIns|T], Sample, Ins, Idx, K) :-
    % insert next element at Pos yielding a new list Ins1
    append(_, [ToIns|Rest], Sample),
    insert_rim(Rest, ToIns, Ins, Pos, Ins1),
    % build prob predicate in Pred
    number_chars(Idx, LIdx), append(['p'], LIdx, LF),
    atom_chars(F, LF), Pred =.. [F, Pos, K],
    % call prob predicate and recurse
    pb_call(Pred), Idx1 is Idx+1,
    generate(T, Sample, Ins1, Idx1, K).
insert_rim([], ToIns, Ins, Pos, Ins1) :-
    append(Ins, [ToIns], Ins1), length(Ins1, Pos).
insert_rim([H|_T], ToIns, Ins, Pos, Ins1) :-
    nth1(Pos, Ins, H), nth1(Pos, Ins1, ToIns, Ins).
insert_rim([H|T] , ToIns, Ins, Pos, Ins1) :-
    \+member(H, Ins), insert_rim(T, ToIns, Ins, Pos, Ins1).
```