Human-level Concept Learning

——--Bayesian Program Learning

Junlong Liu

2016.1.6

[1]Lake, Brenden M., Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction." *Science* 350.6266 (2015): 1332-1338.

Outline

- **Concept definition and learning methods**
- Common drawbacks of existing approaches
- Why Bayesian Programming?
- Bayesian Program Learning (BPL) details
- BPL experiment results and conclusion

What is concept?

- Definition of concept:
- “the search for and listing of attributes that can be used to distinguish exemplars from non exemplars of various categories” Bruner, Goodnow, & Austin (1967)[1]
- “A concept is thought to be knowledge that facilitates the categorization process” (e.g., Barsalou, 1991, 1992)
- Examples: object classes, events, ideas

[1]Bruner, J., Goodnow, J. J., & Austin, G. A. (1967). A study of thinking. New York: Science Editions.

Deterministic concept learning [1]

- Rule-based
  - A concept is represented as a set of if-else rules, like decision trees, boolean forms
- Exemplar-based
  - A concept composes a set of properties
- Prototype-based
  - A concept is abstracted out the central tendency of examples

[1] https://en.wikipedia.org/wiki/Concept_learning

Probabilistic concept learning [1]

- Bayesian theory
  - Bayes' rule $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$
  - Bayesian inference, A:Hypothesis, B:Evidence
  - Bayesian programming: automate probabilistic reasoning

$$\text{Program}\begin{cases}\text{Description}\begin{cases}\text{Specification}(\pi)\begin{cases}\text{Variables}\\ \text{Decomposition}\\ \text{Forms}\end{cases}\\ \text{Identification (based on } \delta)\end{cases}\\ \text{Question}\end{cases}$$

- Statistical learning
  - Define loss function on samples and train models to minimize the loss, like supervised classification methods

[1]https://en.wikipedia.org/wiki/Concept_learning

Outline

- Concept learning definition and methods
- **Common drawbacks of existing approaches**
- Why Bayesian Programming?
- Bayesian Program Learning details
- Experiment results and conclusion

Common drawbacks of existing methods

- All methods need hand-crafted variable abstraction, or feature engineering to handle unstructured examples, except deep learning
- All is data-hungry, especially statistical learning methods, except those with manual definition efforts[1]
- All might be fragile to adversarial attack, since they generally learn from a finite set of concepts.
- Humans outperform machine systems in: [1]
  - 1, one-shot learning ability
  - 2, ability of creating new concepts …

[1]Lake, Brenden M., Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction." *Science* 350.6266 (2015): 1332-1338.

Outline

- Concept learning definition and methods
- Common drawbacks of existing approaches
- **Why Bayesian Programming?**
- Bayesian Program Learning details
- Experiment results and conclusion

What is Bayesian Programming?

- Lebeltel et al. proposed BP [1] in 2004 at first, to address the problem of uncertainty in robotics.

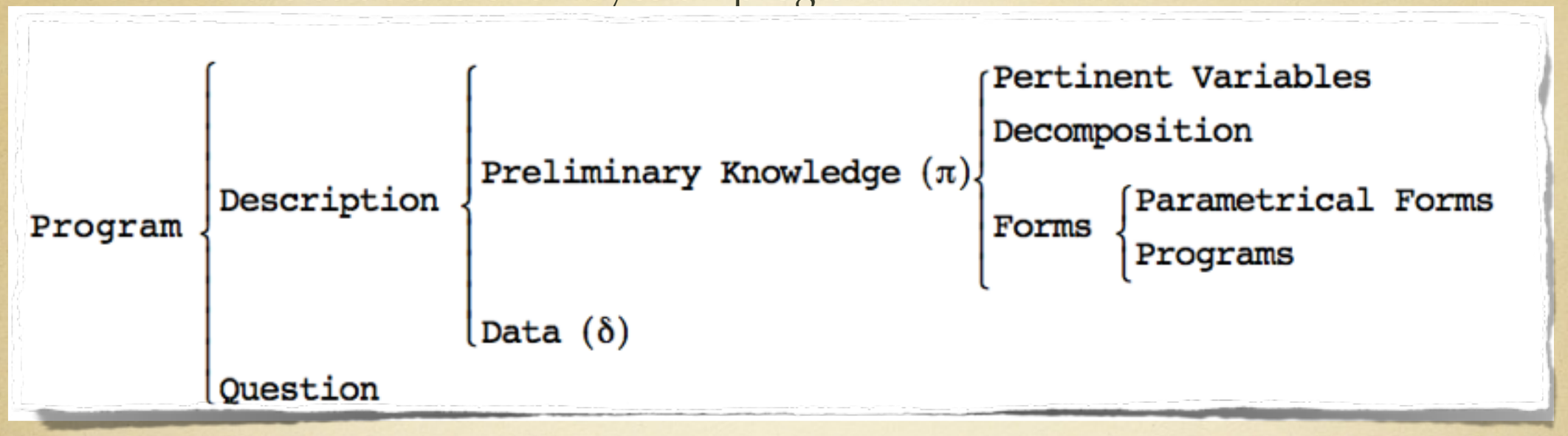
- The goal is to calculate: $P(Y \mid X, Data, Knowledge)$

- BP is a formalism and a methodology to specify probabilistic models and solve problems when all the necessary information is not available.

- BP can restate many classical probabilistic models, for instance, Bayesian Network (BN), Dynamic Bayesian Network (DBN), Bayesian Filters, Hidden Markov Models (HMM), Kalman Filters, Particle Filters, Mixture Models, or Maxim Entropy Models. [2]

[1]Lebeltel O, Bessière P, Diard J, et al. Bayesian robot programming[J]. Autonomous Robots, 2004, 16(1): 49-79.
[2]Diard J, Bessiere P, Mazer E. A survey of probabilistic models using the bayesian programming methodology as a unifying framework[C]//--. 2003: x.

- Below is the structure of a Bayesian program



- The purpose of a "Description" is to specify an effective method to compute joint distribution on a set of variables, namely $P(x^1 \otimes x^2 \otimes \ldots \otimes x^n \mid \delta \otimes \pi)$

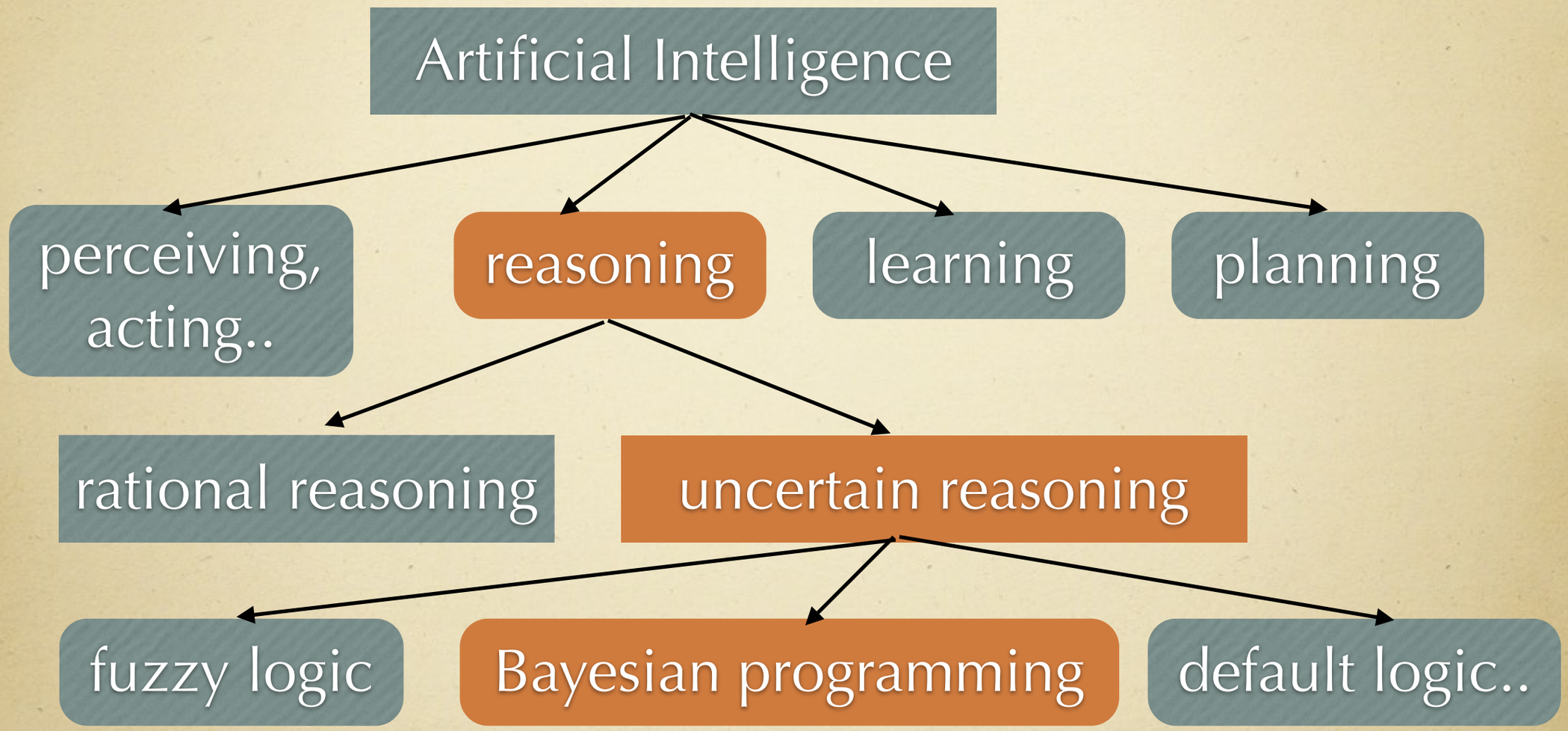
- "Decomposition" partition $\{x^1, x^2, \ldots, x^n\}$ into k subsets, $\{L^1, L^2, \ldots, L^k\}$

$$\mathbf{P}(X^1 \otimes X^2 \otimes \ldots \otimes X^n \mid \delta \otimes \pi)$$
$$= \mathbf{P}(L^1 \mid \delta \otimes \pi) \times \mathbf{P}(L^2 \mid L^1 \otimes \delta \otimes \pi) \times \ldots \times \mathbf{P}(L^k \mid L^{k-1} \otimes \ldots \otimes L^2 \otimes L^1 \otimes \delta \otimes \pi)$$

- "Forms" associate $P(L^1 \mid R^i \otimes \delta \otimes \pi)$ with either a parametric function or another program. where $R^i = \{L^1, \ldots, L^{i-1}\}$

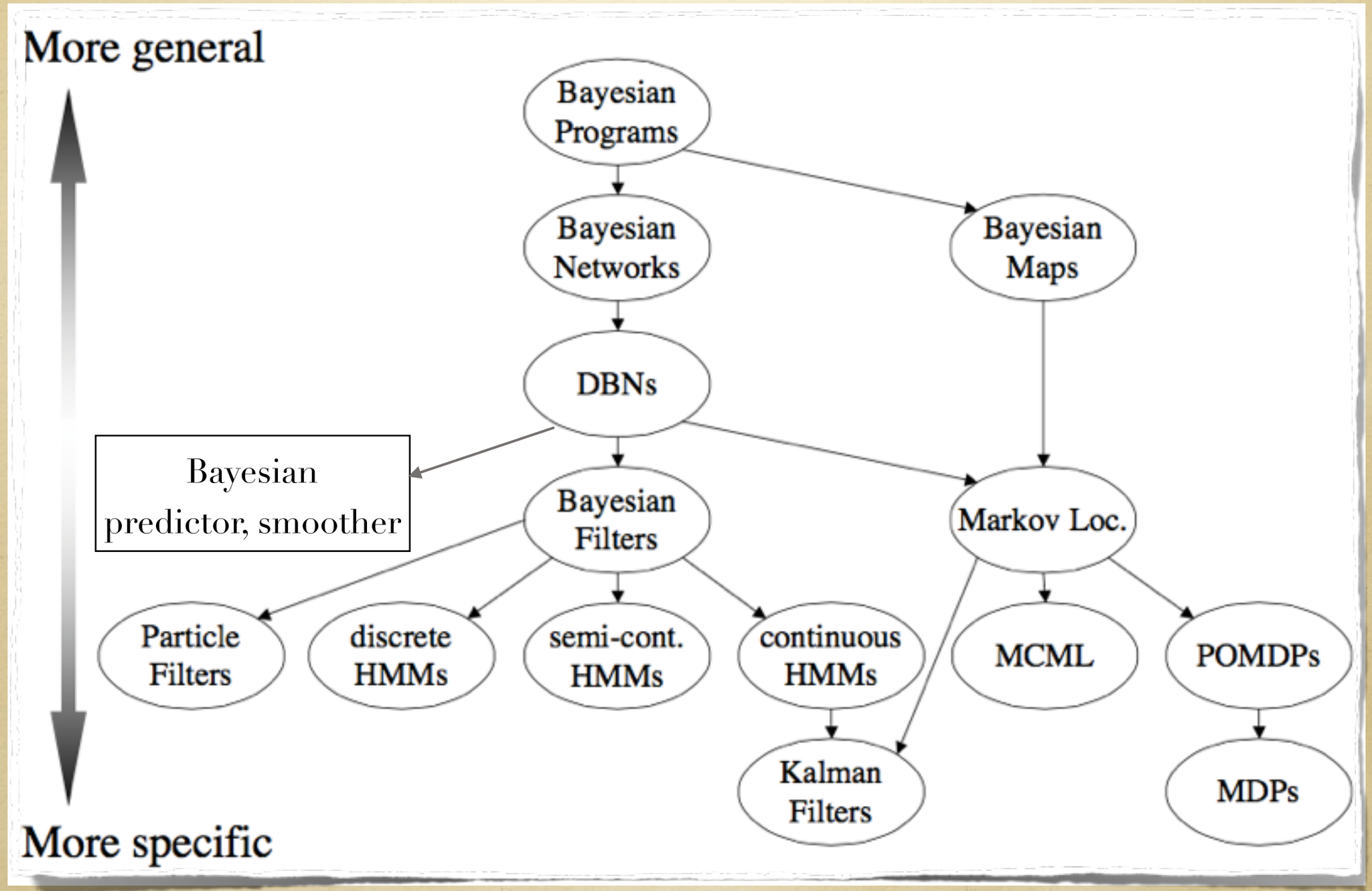
- A "Question" is obtained by divide $\{x^1, x^2, \ldots, x^n\}$ into {searched, unknown, known}, and defined as: $P(Searched \mid Known \otimes \delta \otimes \pi)$

Role of Bayesian Programming in AI



[1]Russell S, Norvig P. Artificial intelligence: a modern approach[J]. 1995.

What Bayesian Programming includes [1]?



[1]Diard J, Bessiere P, Mazer E. A survey of probabilistic models using the bayesian programming methodology as a unifying framework[C]//--. 2003: x.

Why Bayesian Programming?

- Goal: achieve human-level concept learning
  - one-shot classification —> Information is not complete —> uncertain reasoning (fuzzy logic, Bayesian reasoning, default logic…)
  - new concept creation —> various new composition way of common sub-patterns —> Probabilistic generative model
  - Bayesian programming can achieve automatic probabilistic reasoning, also can serve as probabilistic generative model

Outline

- Concept learning definition and methods
- Common drawbacks of existing approaches
- Why Bayesian Programming?
- **Bayesian Program Learning details**
- Experiment results and conclusion

Bayesian Program Learning Details

- **What is Bayesian Program Learning**
  - Key idea and formulation
  - Relation with Bayesian programming
- How does BPL create concept?
- How does BPL learn concept?
  - How to do one-shot classification and generation?

What is BPL?

- Key idea and formulation:
  - Compositionality, Causality, and Learning to learn
    - Decompose rich concepts into simpler primitives, use probabilistic semantics to handle noise and capture abstract casual structure, then construct programs under a Bayesian criterion.
  - Formula:

$$P(\psi) = P(\kappa) \prod_{i=1}^{\kappa} P(S_i) P(R_i | S_1, \ldots, S_{i-1})$$

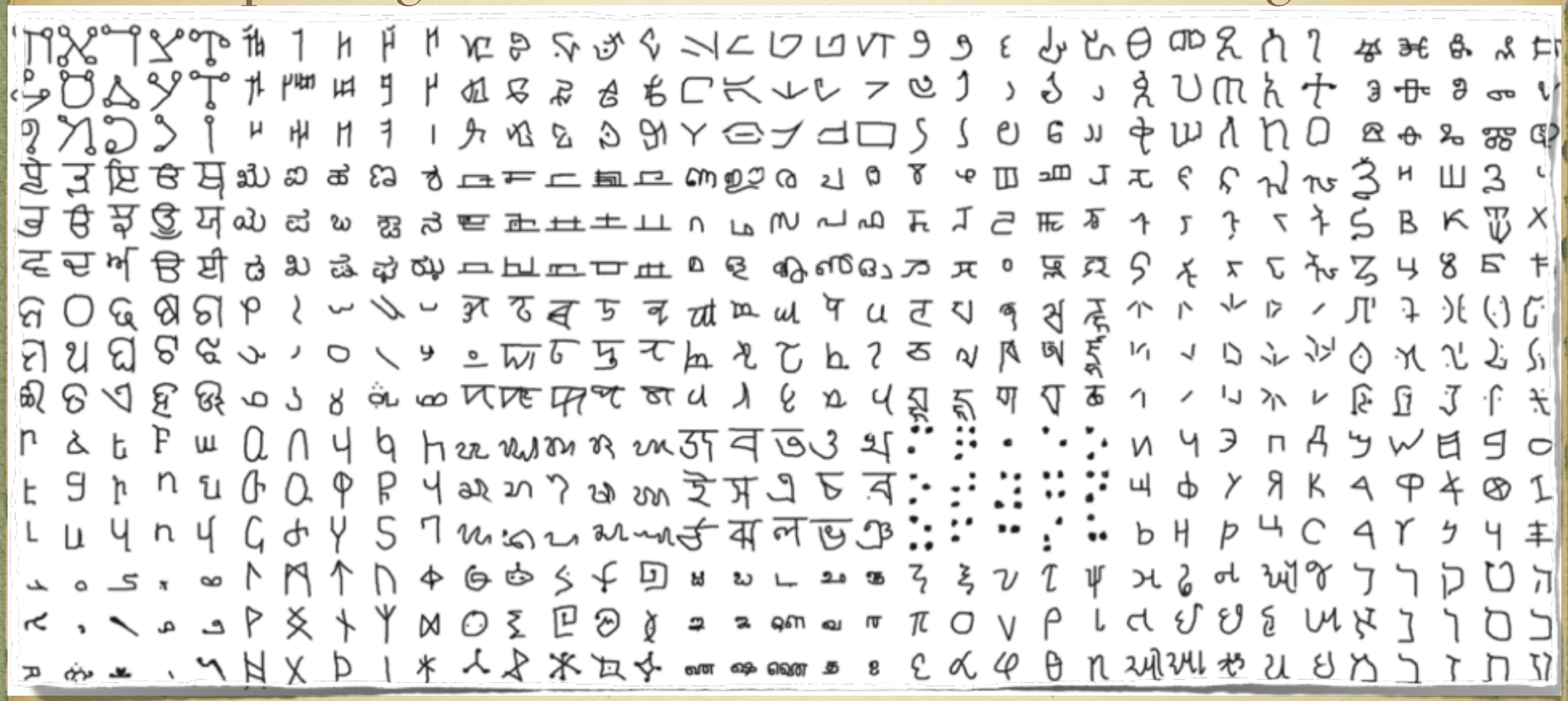
    - ψ is a concept type, k is the number of sub-parts, Si is ith sub-part, Ri is the relation between Si and previous sub-parts. Thus $\psi = (\kappa, S, R)$
- Relation with Bayesian Programming
  - BPL is domain-specific BP

BPL address what kind of concept learning?

- Concepts: handwritten characters from the world's alphabets[1]
- Problem: one-shot classification and generation, concept creation
- Notation:
  - strokes: Si, from pen down to pen up
  - sub-strokes (primitives): sij, separated by brief pause of pen
  - relation: Ri, specifies how Si connects previous strokes
  - concept type: $\psi = (\kappa, S, R)$, is an abstract schema of a concept
  - concept token: $\theta^{(m)}$, is a concrete form of concept type

[1]This dataset contains 1623 characters spanning 50 different alphabets on www.omniglot.com.

Dataset samples , simple visual concepts for comparing human and machine learning



525 (out of 1623) character concepts from omniglot dataset [1], shown with one example each.

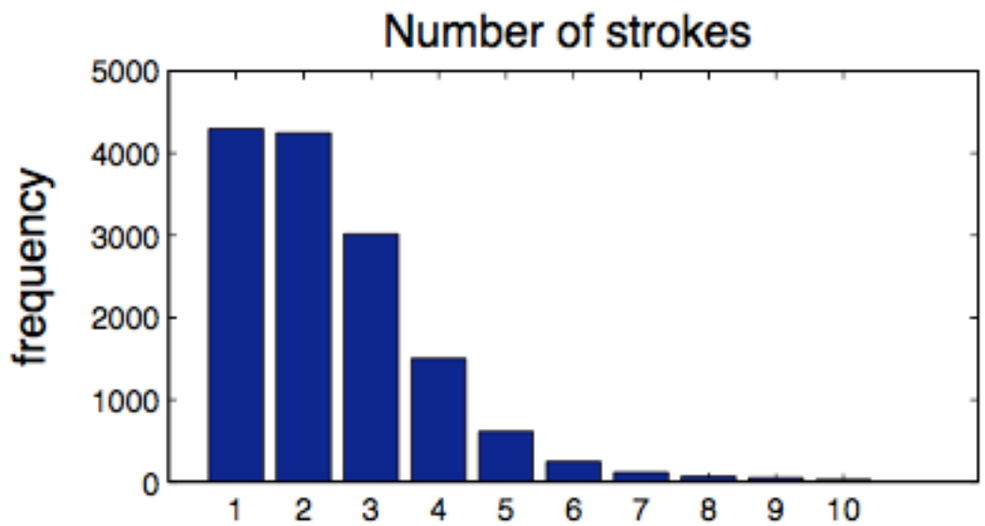
30(/50) alphabets as background set, used to learn BPL conditional distributions.

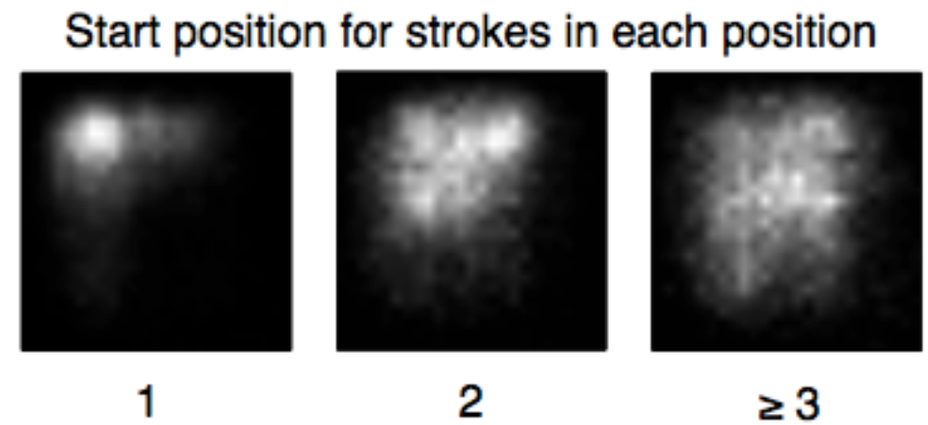
20(/50) other alphabets as evaluation set for one-shot classification and generation.

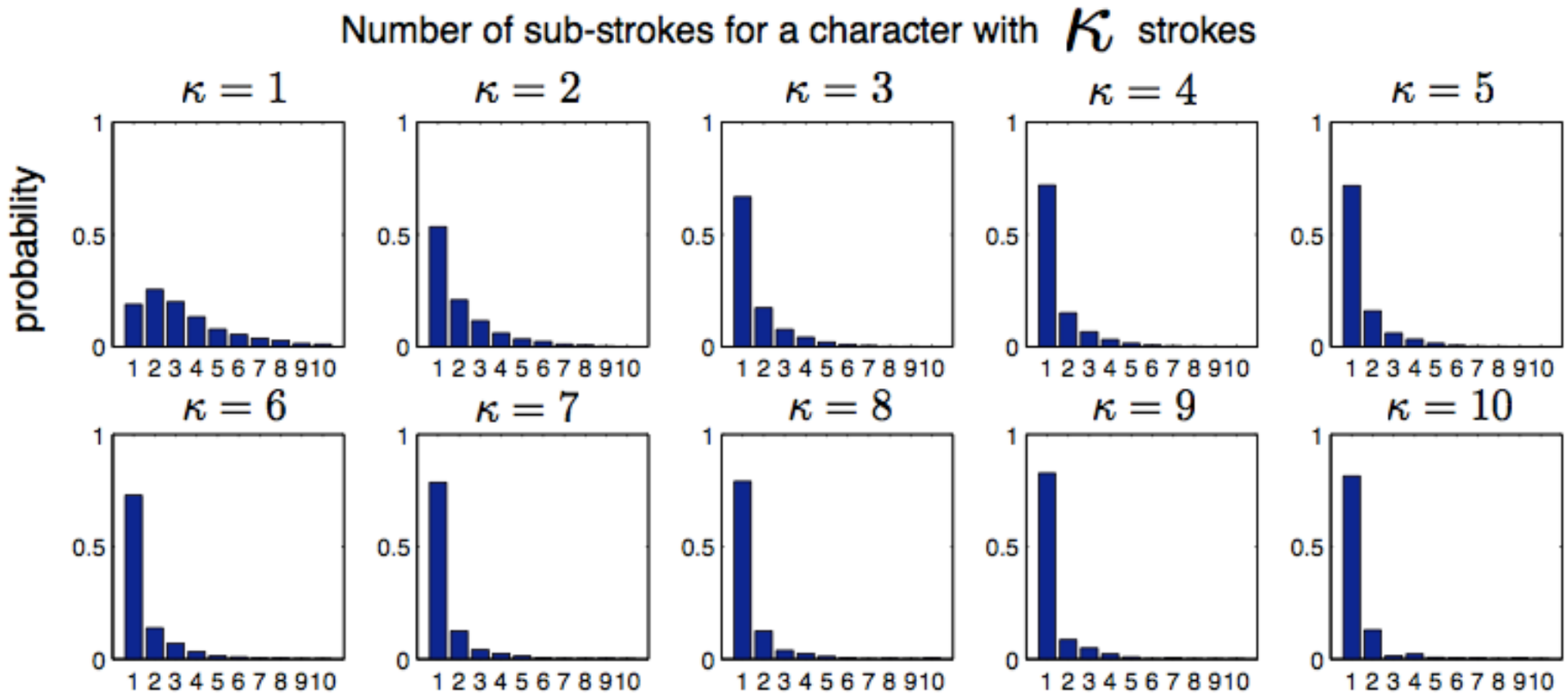
How does BPL generate new concept?

- First, learn primitives, strokes from background datasets, define relations
- Second, do statistics on background datasets (30/50 alphabets), to obtain various distributions, conditional distributions and gaussian noise parameters
- Third, run probabilistic generative program according to statistics learned in second step

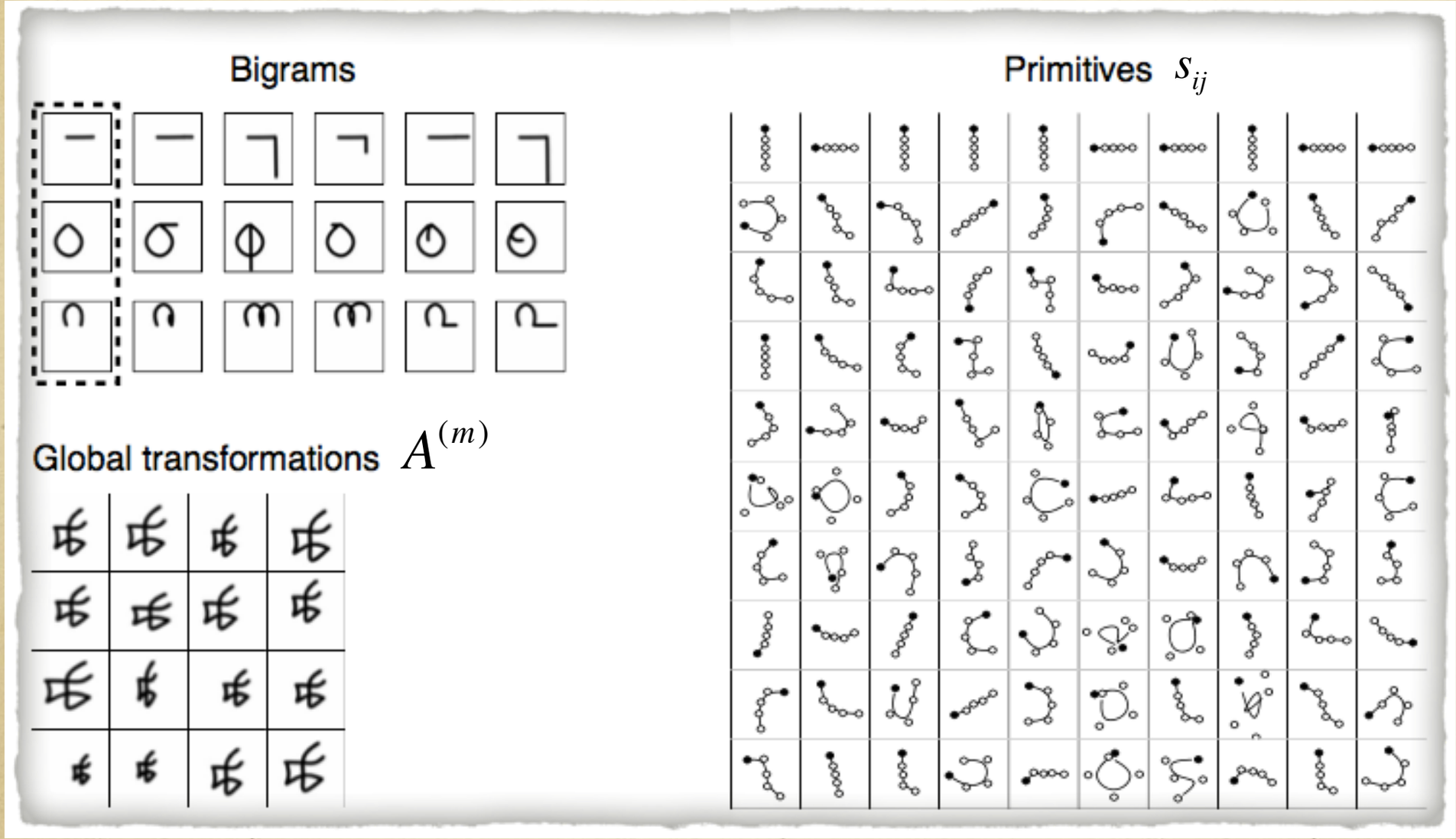
Statistics on background datasets 1







Statistics on background datasets 2



The *Bigrams* illustration show seed primitive in image space (leftmost column) and then the 5 most likely continuations when sampling a second primitive.

Concept generation algorithm

procedure GENERATETYPE
  $\kappa \leftarrow P(\kappa)$ ▷ Sample the number of strokes
  for $i = 1 \ldots \kappa$ **do**
    $n_i \leftarrow P(n_i|\kappa)$ ▷ Sample the number of sub-strokes
    $S_i \leftarrow$ GENERATESTROKE(i, n_i) ▷ Sample stroke
    $\xi_i \leftarrow P(\xi_i)$ ▷ Sample relation to previous strokes
    $R_i \leftarrow P(R_i|\xi_i, S_1, \ldots, S_{i-1})$ ▷ Sample relation details
  end for
  $\psi \leftarrow \{\kappa, R, S\}$
  return @GENERATETOKEN(ψ) ▷ Return program handle
end procedure

P(k), P(ni|k)… are calculated from background datasets

procedure GENERATESTROKE(i,n_i)

$z_{i1} \leftarrow P(z_{i1})$ ▷ Sample the identity of the first sub-stroke

for $j = 2 \ldots n_i$ **do**

$z_{ij} \leftarrow P(z_{ij}|z_{i(j-1)})$ ▷ Sample the identities of the other sub-strokes

end for

for $j = 1 \ldots n_i$ **do**

$x_{ij} \leftarrow P(x_{ij}|z_{ij})$ ▷ Sample a sub-stroke's control points

$y_{ij} \leftarrow P(y_{ij}|z_{ij})$ ▷ Sample a sub-stroke's scale

$s_{ij} \leftarrow \{x_{ij}, y_{ij}, z_{ij}\}$

end for

$S_i \leftarrow \{s_{i1}, \ldots, s_{in_i}\}$ ▷ A complete stroke definition

return S_i

end procedure

P(zij|zi(j-1)) is calculated from background datasets

P(xij|zij), P(yij|zij) follows Gaussian distribution, mean, std are calculated from background datasets

Example of generating new character concept

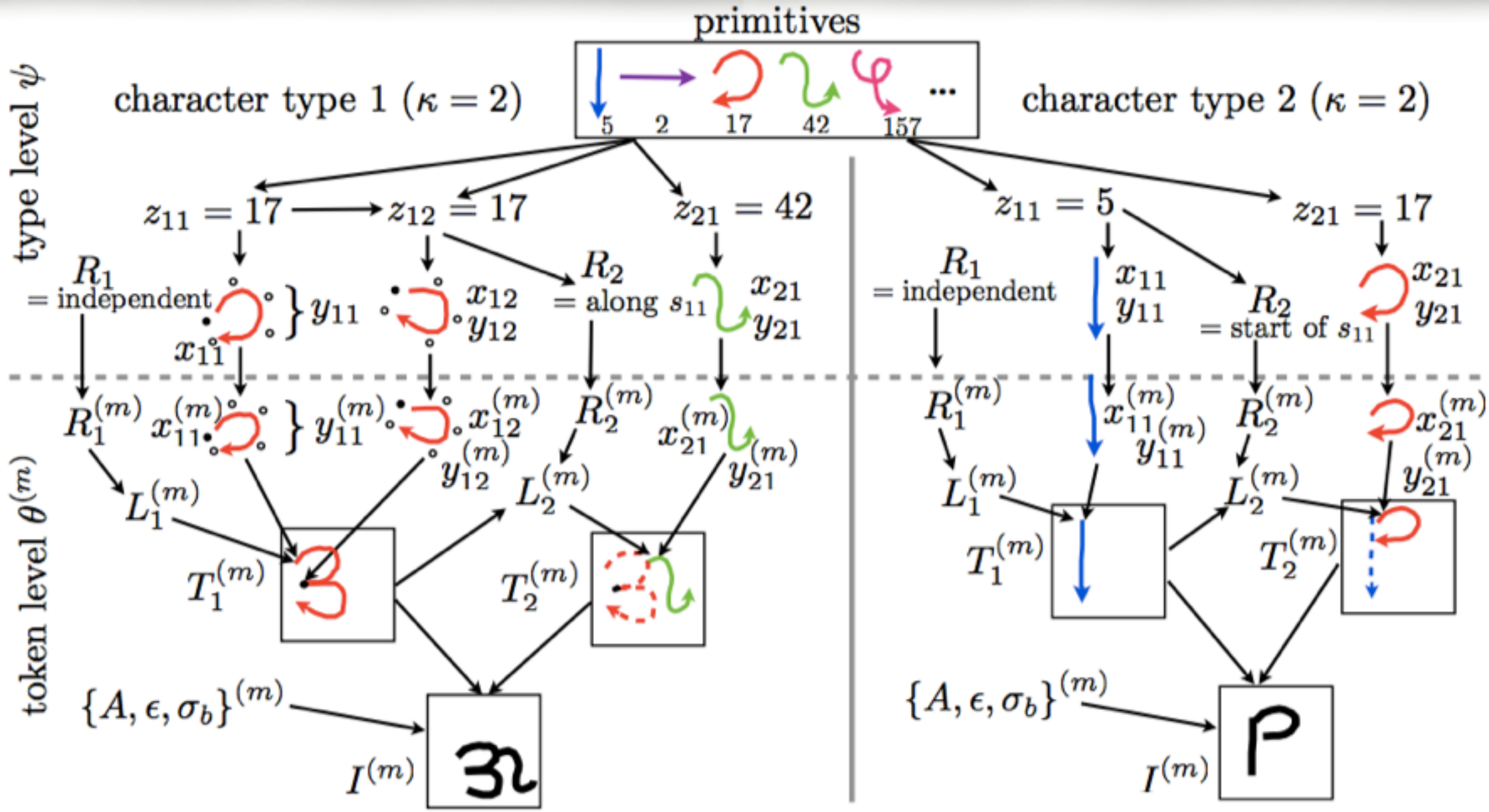


Figure S1: BPL generating two character types. Legend: number of strokes κ, relations R, primitive id z (color-coded to highlight sharing), control points x (open circles), scale y, start locations L, trajectories T, transformation A, noise ϵ and σ_b, and image I.

Note that $P(x^{(m)} \mid x) = \delta(x^{(m)} - x) \propto \mathrm{N}(x, \sigma_x^2), x = [L, R, x, T, A, \varepsilon, \sigma_b]$

Bayesian Program Learning

- What is Bayesian Program Learning
  - Key idea and formulation
  - Relation with Bayesian programming
- How does BPL create concept?
- **How does BPL learn concept?**
  - How to do one-shot classification and generation?

How does BPL learn concept?

- To do one-shot classification:
  - First, generate concept abstraction of train image
  - Second, compute posterior probability of test image belonging to each class and argmax:

$$\arg\max_{c} \log P(I^{(T)} \mid I^{(c)})$$

Bayesian Classification

$I^{(T)}$:test image

$I^{(c)}$:class exemplar, train image

Concept abstraction generation

- First, generate concept tokens and types for train image
  - Extract the character skeleton and generate random parses
  - Search for sub-strokes(primitives) $s_i = (x_i, y_i, z_i)$:

xi,yi:control points
zi:primitive index

$$P(x_i^{(m)}, y_i^{(m)}, z_i) = P(z_i) \prod_{j=1}^{n_i} P(y_{ij}^{(m)} | y_{ij}) P(y_{ij} | z_{ij}) \int P(x_{ij}^{(m)} | x_{ij}) P(x_{ij} | z_{ij}) \, \mathrm{d}x_{ij}$$

  - Then, $\theta^{(m)}, \psi$ for each parse can be specified and choose some best parses by $P(\theta^{(m)} | \psi) P(\psi)$

where $\theta^{(m)} = \{L^{(m)}, x^{(m)}, y^{(m)}, R^{(m)}, A^{(m)}, \varepsilon^{(m)}, \sigma_b^{(m)}\}$

Posterior Probability Computation 1

- Second, compute posterior probability of test image belonging to each class and argmax:
  - Assume K best match parses for test image:

$$\psi^{[1]}, \theta^{(m)[1]}, \ldots, \psi^{[K]}, \theta^{(m)[K]}$$

  - These matches approximate posterior with a discrete distribution:

$$P(\psi, \theta^{(m)} | I^{(m)}) \approx \sum_{i=1}^{K} w_i \delta(\theta^{(m)} - \theta^{(m)[i]}) \delta(\psi - \psi^{[i]})$$

where $\delta(x - x^{[i]}) = P(x \mid x^{[i]})$

Posterior Probability Computation 2

- The matches approximate posterior with a discrete distribution:

$$P(\psi, \theta^{(m)} | I^{(m)}) \approx \sum_{i=1}^{K} w_i \delta(\theta^{(m)} - \theta^{(m)[i]}) \delta(\psi - \psi^{[i]})$$

where $\delta(x - x^{[i]}) = P(x \mid x^{[i]})$ $\quad \theta^{(m)} = \{L^{(m)}, x^{(m)}, y^{(m)}, R^{(m)}, A^{(m)}, \varepsilon^{(m)}, \sigma_b^{(m)}\}$

$\sum_i w_i = 1, w_i$ is proportional to parse score

In details, $w_i \propto P(\theta^{(m)[i]} \mid \psi^{[i]}) P(\psi^{[i]})$

$$P(\theta^{(m)} \mid \psi) = P(L^{(m)} \mid \theta^{(m)}_{\setminus L^{(m)}}, \psi) \prod_i P(R_i^{(m)} \mid R_i) P(y_i^{(m)} \mid y_i) P(x_i^{(m)} \mid x_i) P(A^{(m)}, \varepsilon^{(m)}, \sigma_b^{(m)})$$

$$P(\psi) = P(\kappa) \prod_{i=1}^{\kappa} P(S_i) P(R_i \mid S_1, \ldots, S_{i-1})$$

Posterior Probability Computation 3

To get better approximation, take N sample from: $P(\psi^{[i]} \mid \theta^{(m)[i]})$

$$P(\psi, \theta^{(m)} | I^{(m)}) \approx Q(\psi, \theta^{(m)}, I^{(m)}) = \sum_{i=1}^{K} w_i \delta(\theta^{(m)} - \theta^{(m)[i]}) \frac{1}{N} \sum_{j=1}^{N} \delta(\psi - \psi^{[ij]})$$

Then, log likelihood of test image being class c is:

$$\begin{aligned} \log P(I^{(T)} | I^{(c)}) &\approx \log \int P(I^{(T)} | \theta^{(T)}) P(\theta^{(T)} | \psi) Q(\theta^{(c)}, \psi, I^{(c)}) \, \mathrm{d}\psi \, \mathrm{d}\theta^{(c)} \, \mathrm{d}\theta^{(T)} \\ &\approx \log \sum_{i=1}^{K} w_i \max_{\theta^{(T)}} P(I^{(T)} | \theta^{(T)}) \frac{1}{N} \sum_{j=1}^{N} P(\theta^{(T)} | \psi^{[ij]}), \end{aligned}$$

How does BPL do one-shot generation?

- To do one-shot generation, sampling from this posterior predictive distributions:

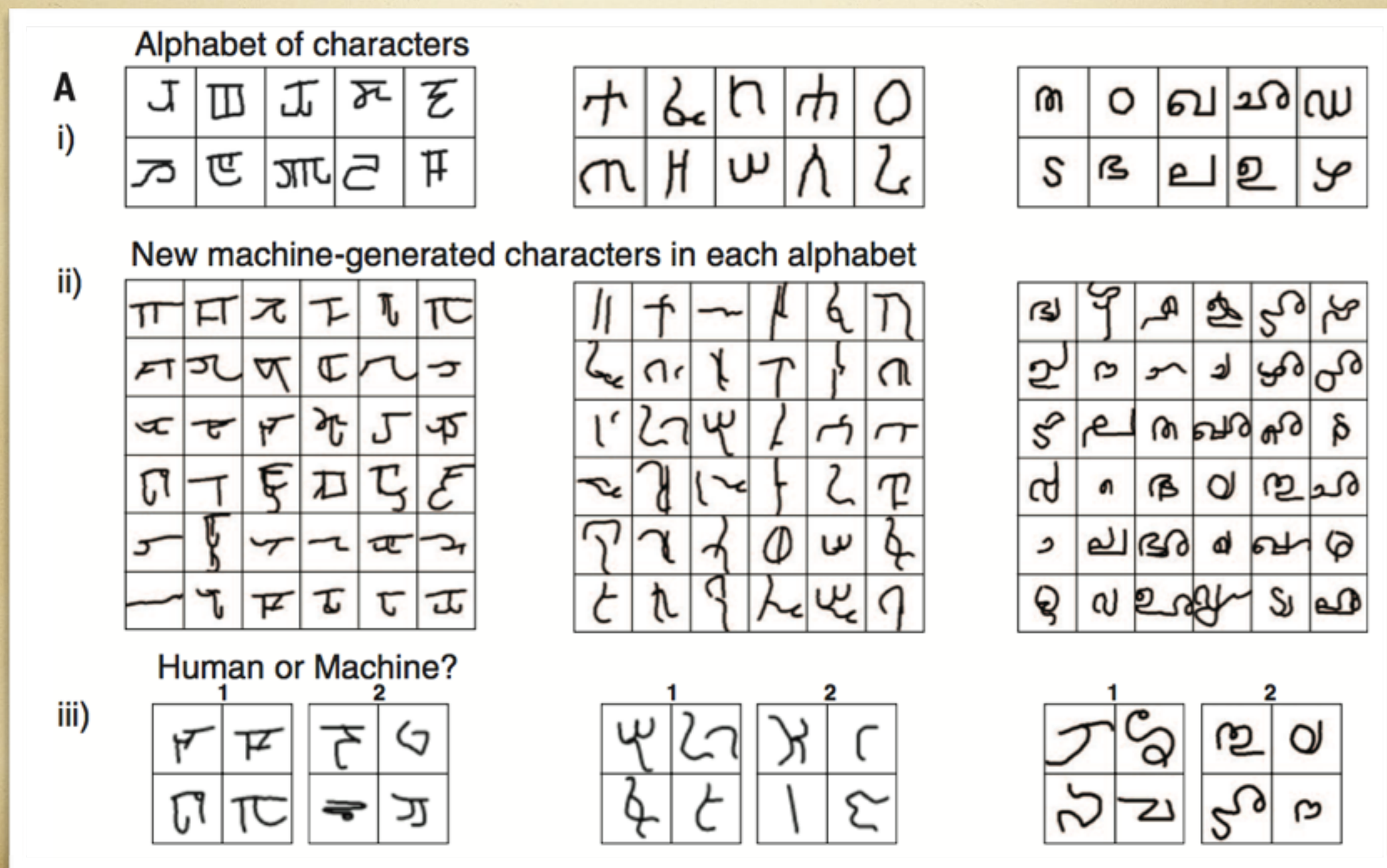
$$
\begin{aligned}
P(I^{(2)}, \theta^{(2)} | I^{(1)}) = & \int P(I^{(2)}, \theta^{(2)} | \theta^{(1)}, \psi) P(\theta^{(1)}, \psi | I^{(1)}) \, \mathrm{d}(\psi, \theta^{(1)}) \\
= & \int P(I^{(2)} | \theta^{(2)}) P(\theta^{(2)} | \psi) P(\theta^{(1)}, \psi | I^{(1)}) \, \mathrm{d}(\psi, \theta^{(1)}) \\
\approx & \int P(I^{(2)} | \theta^{(2)}) P(\theta^{(2)} | \psi) Q(\theta^{(1)}, \psi, I^{(1)}) \, \mathrm{d}(\psi, \theta^{(1)}) \\
= & \sum_{i=1}^{K} \sum_{j=1}^{N} \frac{w_i}{N} P(I^{(2)} | \theta^{(2)}) P(\theta^{(2)} | \psi^{[ij]}).
\end{aligned}
$$

Outline

- Concept learning definition and methods
- Common drawbacks of existing approaches
- Why Bayesian Programming?
- Bayesian Program Learning details
- **Experiment results and conclusion**

1st part

Fig. 7. Generating new concepts. (**A**) Humans and machines were given a novel alphabet (i) and asked to produce new characters for that alphabet. New machine-generated characters are shown in (ii). Human and machine productions can be compared in (iii). The four-character grids in each pair that were generated by the machine are (by row) 1, 1, 2; 1, 2. (**B**) Humans and machines produced new characters without a reference alphabet. The grids that were generated by a machine are 2; 1; 1; 2.



2nd part

Fig. 7. Generating new concepts. (**A**) Humans and machines were given a novel alphabet (i) and asked to produce new characters for that alphabet. New machine-generated characters are shown in (ii). Human and machine productions can be compared in (iii). The four-character grids in each pair that were generated by the machine are (by row) 1, 1, 2; 1, 2. (**B**) Humans and machines produced new characters without a reference alphabet. The grids that were generated by a machine are 2; 1; 1; 2.

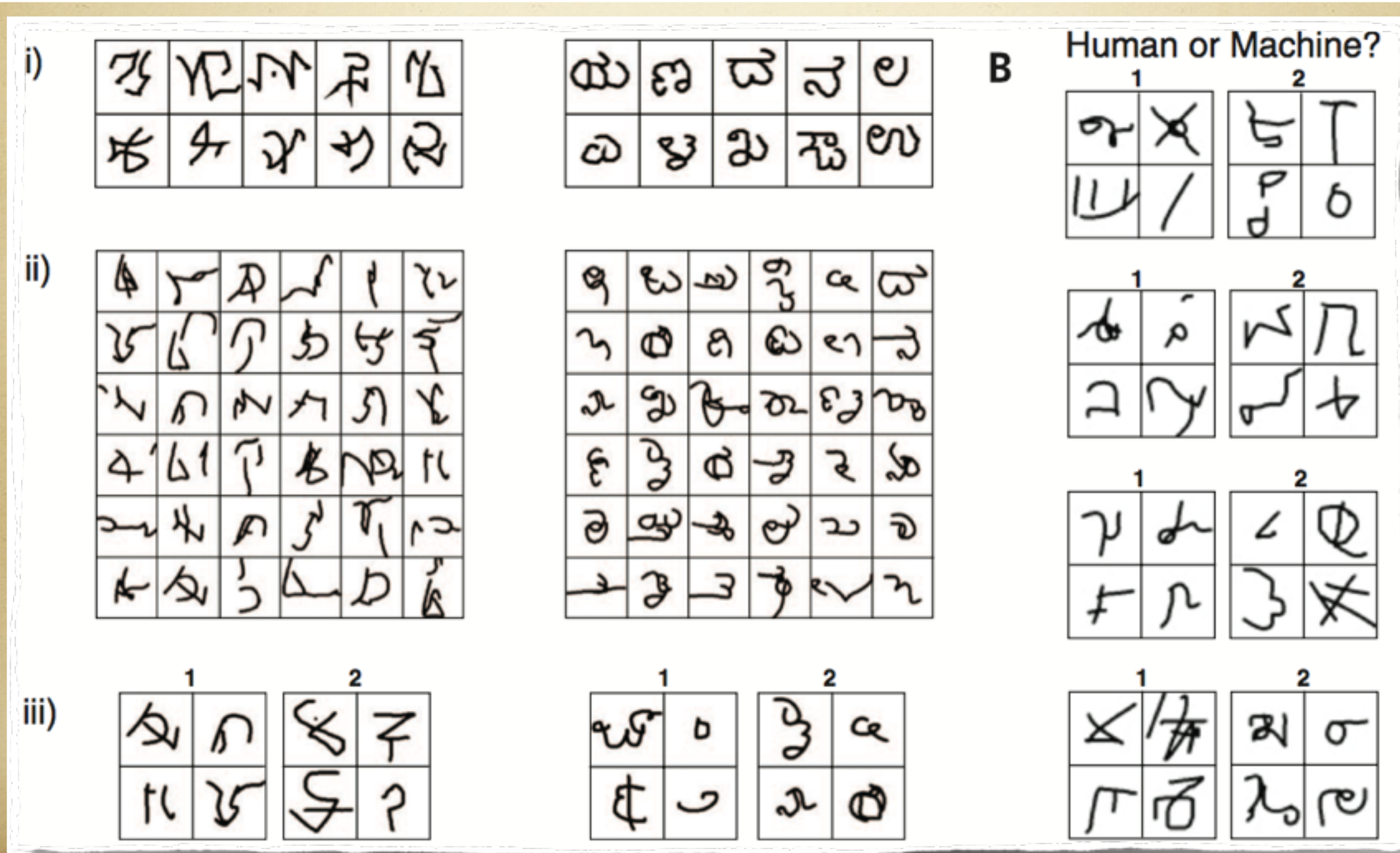


Fig. 5. Generating new exemplars. Humans and machines were given an image of a novel character (top) and asked to produce new exemplars. The nine-character grids in each pair that were generated by a machine are (by row) 1, 2; 2, 1; 1, 1.

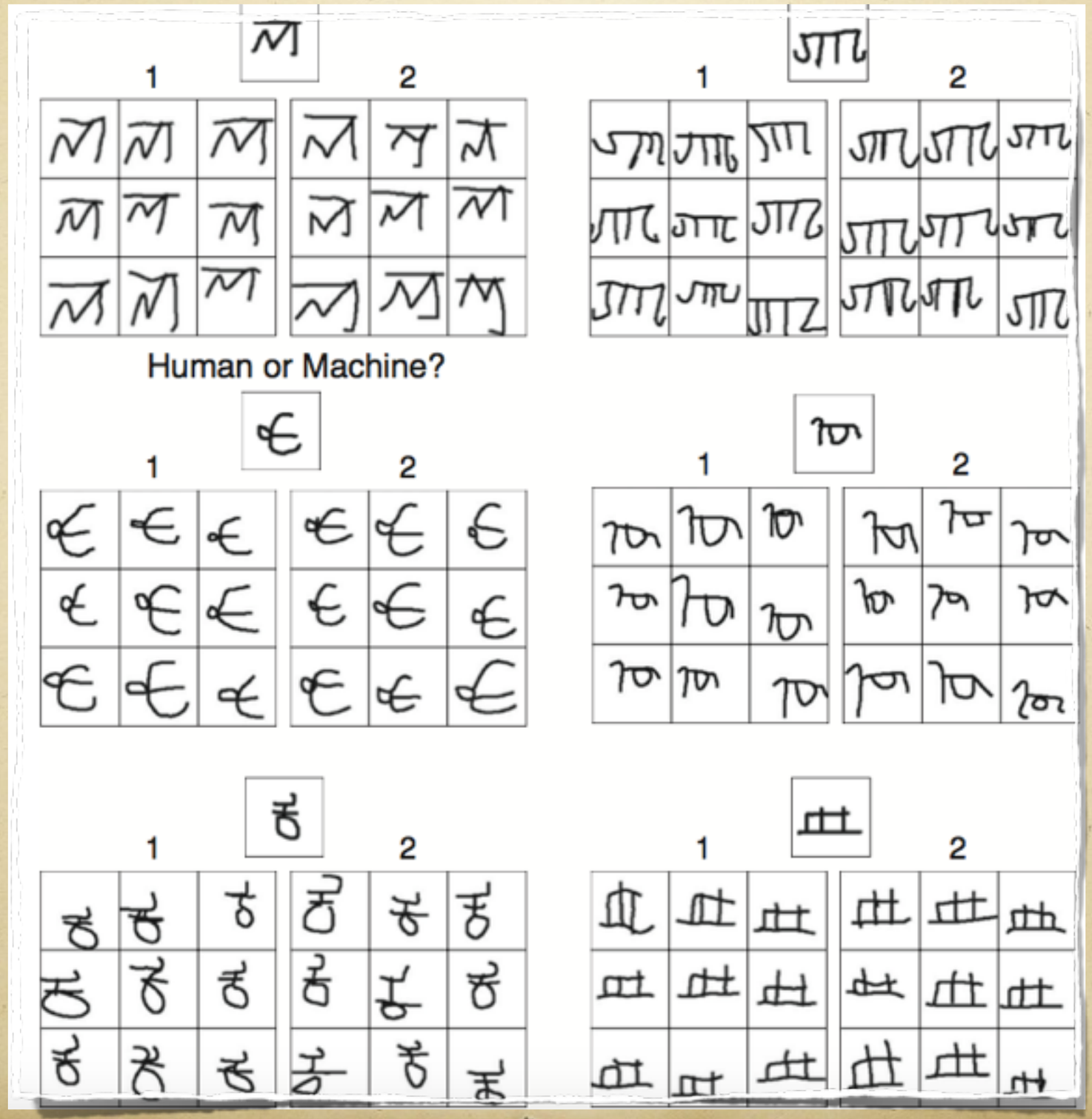
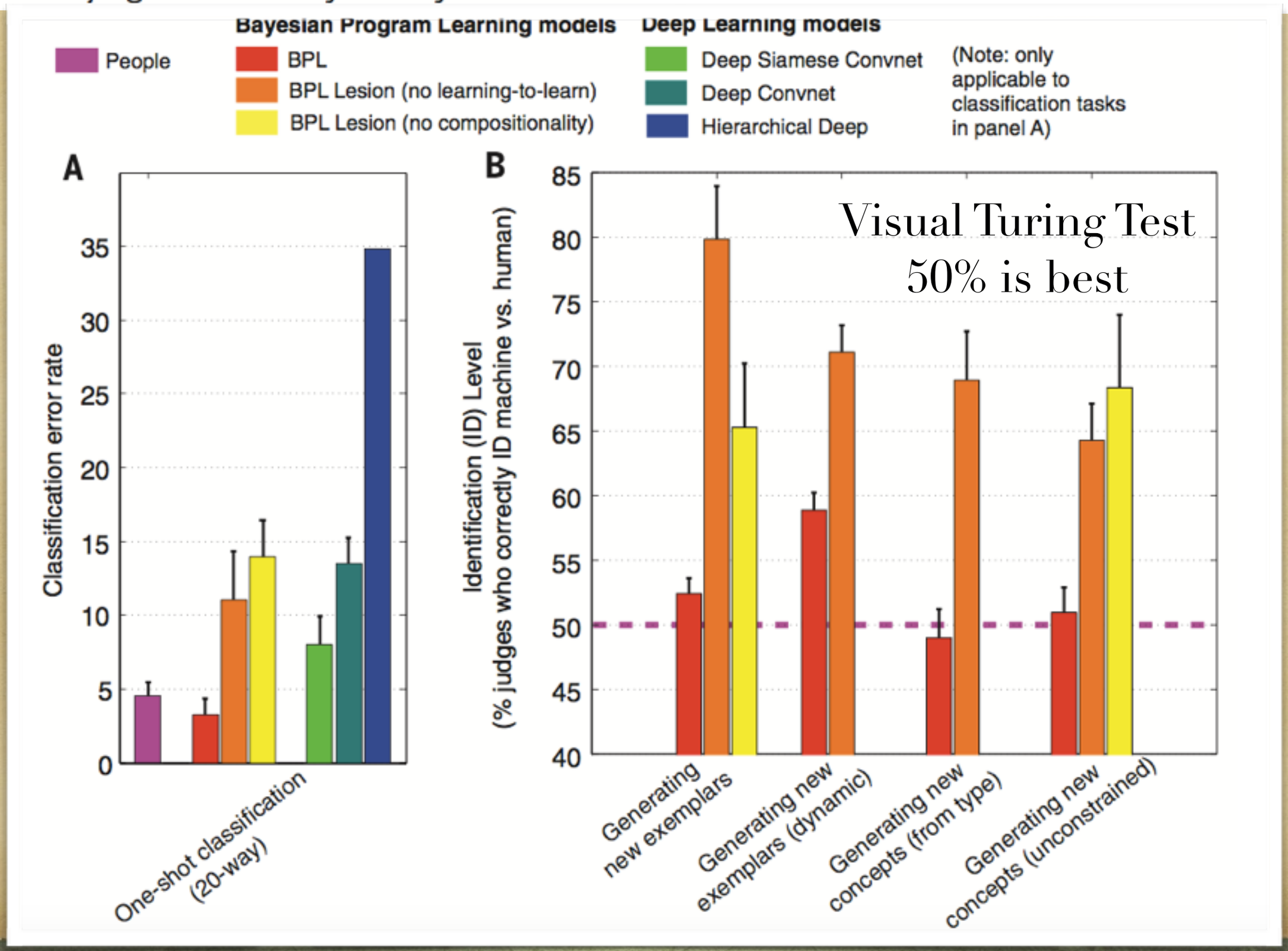


Fig. 6. Human and machine performance was compared on (A) one-shot classification and (B) four generative tasks. The creative outputs for humans and models were compared by the percent of human judges to correctly identify the machine.



Discussion and Conclusion

- BPL can:
  - perform one-shot learning in classification tasks at human-level accuracy
  - fool most judges in visual Turing tests of its more creative abilities
- The principles of compositionality, causality, and learning to learn may be critical in building machines to narrow gaps with human intelligence.
- Human also use concept in planning, explanation, communication,…
- BPL may apply in other symbolic concept learning, what else?

Thank You!