Compositional Metamodeling Language

fsaad

February 14, 2016

1 Populations, Variables, and Measurement Types

The Metamodeling Language formalizes the notion of a *statistical population* to be a collection of named variables, $\boldsymbol{X} = \{X_1, \dots, X_D\}$. Each variable X_i takes values in a measurement space $\mathcal{X}_i$, known to MML only through a qualitative token called the *measurement type*.

Some possible measurement types and associated spaces $\mathcal{X}$ for population variables are shown in Table 1.1.

Measurement Type	Example Measurement Spaces
Audio	$\{0,1\}^4$, pulse code modulated signals.
Binary	$\{0,1\}$, $\{\text{boy}, \text{girl}\}$, any arbitrary binary labels.
Circular	$(-\pi, \pi]$, wall clock, directional sets that can be "wrapped".
Counts	$\{0,1,2,\dots\}$, frequencies, rates.
Categorical	$\{1,\dots,K\}$, finite unordered sets with arbitrary labels.
Image	URL i.e `http://fsaad.mit.edu/media/mypic.jpg`, or pixel matrices $\{0,1\}^{28\times 28}$.
Numerical	$\mathbb{R}$, $[0,100]$, uncountable sets.
Ordinal	{'small','medium','large'}, finite well-ordered set with arbitrary labels.

Table 1.1: Possible measurement types for the variables in a population. These measurement types and spaces are not comprehensive, and not all are implemented in MML.

Populations in MML are declared using `CREATE POPULATION`, followed by a *population schema* [1].

```
CREATE POPULATION swallows(
    treatment BINARY,
    orientation CIRCULAR(0, 6.28));
```

The variables in the `swallows` population are `treatment` and `orientation`, and their measurement types are `BINARY` and `CIRCULAR`.

A *member* of the population is a measurement $\boldsymbol{x}_i$, which is a multivariate entity with D attributes. Entry x_{ij} takes values in the measurement space $\mathcal{X}_j$ of the jth variable declared in the population schema. Members of a population have neither probabilistic interpretations nor any distributional assumptions.

MML `INCORPORATE` loads measurements into a population from a data source. Every incorporated measurement is implicitly assigned a binary `primary key` (or `pk`) by MML, which is a unique identifier for all measurements across all populations.

```
INCORPORATE INTO swallows OBSERVATIONS swallows.csv
```

[1] The `swallow` population will be used as a running example in this document. This dataset is collected from an ecology experiment that studied the orientation of juvenile barn swallows (birds) under two magnetic field treatments, local and shifted. The research question was whether swallows can use magnetic information for compass orientation [GB04].

The first 5 measurements can be viewed using a SQL query.

```
SELECT * FROM swallows LIMIT 5
```

treatment	orientation
local	5.38
local	2.32
shifted	3.63
local	0.28

Table 1.2: Five measurements from the `swallows` population. The measurement space for the `BINARY` variable `treatment` is {'local', shifted}, and for `orientation` is $[0, 2\pi)$.

2 Schemas for Generative Population Models

In Section 1 it was explained that a population is not associated with any particular model, it is merely a collection of measurements. The contents of a population can be queried using languages such as SQL.

A *generative population model* (GPM) encodes modeling assumptions for the measurement-generating process of variables in populations. From the perspective of MML, a GPM is a *procedure* which can

- `GENERATE` measurements of (a sub-collection of) variables from some population,
- be `GIVEN` measurements of other variables in the population that it needs to generate its outputs,
- use a `PROGRAM`, opaque to MML, that contains modeling commands.

GPMs offer a form of procedural abstraction for probabilistic and non- probabilistic models. MML cannot see inside the `PROGRAM` of a GPM. The behavior of the GPM is only known to MML in terms of the `GENERATE` and `GIVEN` variables.

Instances of GPMs in MML are declared using `CREATE GPM`, followed by a *schema*

```
CREATE GPM treat-bernoulli FOR swallows USING bernoulli(
    GENERATE (treatment)
    GIVEN (NULL)
    PROGRAM (prior beta(1,1))
```

The `treat-bernoulli` GPM is a density estimator for the marginal distribution of the `treatment` variable. As an *instance* of the `bernoulli` GPM[2], `treat-bernoulli` assumes that measurements of `treatment` are observations from an exchangeable sequence of Bernoulli flips with weight θ. The `bernoulli` GPM can only accept one variable in `GENERATE` whose measurement type is `BINARY`.

`GIVEN` is empty since no other variables are needed to generate measurements from this marginal distribution. The `PROGRAM` is a binary opaque to MML. In this instance, `bernoulli` allows modeling commands to specify the prior over weight θ. Every GPM has its own parser for the contents of `PROGRAM`. Another schema that `bernoulli` would accept is

```
CREATE GPM ... USING bernoulli (
    ...
    PROGRAM (maximum-likelihood))
```

MML allows users to `INITIALIZE` multiple independent instances of a GPM with the same schema. This strategy is usually desirable for complex GPMs which have a high degree of uncertainty in their own generative processes.

```
INITIALIZE 2 INSTANCES OF treat-bernoulli
```

[2] A good question is "where did `bernoulli` GPM come from?" It was compiled with MML.

3 Modeling Directives for Generative Population Models

The INCORPORATE command must be used explicitly to load measurements into a GPM[3].

```
INCORPORATE INTO treat-bernoulli OBSERVATIONS (
    SELECT treatment FROM swallows WHERE pk <= 5)
```

Since the GPM generates variable swallows.treatment, every incorporated measurement must have a corresponding primary key in the swallows population. The previous example only incorporated the first five measurements from swallows. treat-bernoulli can reveal which measurements have been incorporated.

```
SELECT pk FROM treat-bernoulli
```

treat-bernoulli.pk
swallows.#1
swallows.#2
swallows.#3
swallows.#4
swallows.#5

Table 3.1: Primary keys of observations incorporated into the treat-bernoulli GPM.

MML also permits UNINCORPORATE of observations[4].

```
UNINCORPORATE FROM treat-bernoulli OBSERVATIONS (
    SELECT treatment FROM swallows WHERE treatment = ’local’)
```

Unincorporating only some variables x_{ij} from a member $\boldsymbol{x_i}$ is supported. A primary key is unincorporated from a GPM once all its variables have been unincorporated.

Multiple GPMs can be created for the same population variable. For instance, the frequency GPM is a much simpler density estimator for treatment. It learns a distribution by counting the empirical frequencies of all its incorporated measurements, and takes no program.

```
CREATE GPM swallow-freq USING frequency(
    GENERATE (treatment) GIVEN (NULL) PROGRAM (NULL)
-- Incorporate all the treatment data.
INCORPORATE INTO swallow-freq OBSERVATIONS (SELECT treatment FROM swallows)
```

An important point. The frequency GPM is applicable to variables of any *measurement type* (Table 1.1), since it only tallies frequencies of occurrences. It can also accept several variables in GENERATE and learns joint distributions by counting joint frequencies of observations.

MML can instruct swallow-freq to UPDATE SCHEMA to also model the orientation variable.

```
UPDATE SCHEMA FOR swallow-freq(GENERATE (orientation))
```

Since the GIVEN and PROGRAM are both NULL, they have been omitted. Without INCORPORATE, however, the swallow-freq GPM has missing data for all the orientation variables.

```
INCORPORATE INTO swallow-freq OBSERVATIONS(
    SELECT orientation FROM swallows WHERE pk <= 10)
```

Only the first 10 observations in swallow-freq have complete data now. The rest have observations only for the treatment variable. Using frequency GPM for NUMERICAL data is rarely a good idea.

[3] This strategy allows GPMs to observe/unobserve and perform other dynamic inference controls, without having to worry about manipulating the data directly in the population and interfering with other GPMs on the same population. The population is just a data store. This strategy also allows different GPMs over the same variables to observe different measurements (subsampling)

[4] Unincorporating an observation with a pk that has not been incorporated will be ignored, or raise, or warn, etc.

4 Inference Directives for Generative Population Models

After incorporating data into a GPM, it can now learn more about its generative process, a process called *learning* or *posterior inference*. MML instructs the GPM to ANALYZE, optionally taking a PROGRAM.

```
ANALYZE swallow-freq(
    TIMEOUT = 100s)
```

After ANALYZE has completed, swallow-freq will recompute the frequency table for outcomes of its variables, which jointly take values in the product space $\{\text{‘local’},\text{‘shifted’}\} \times [0, 2\pi)$. The code (TIMEOUT = 100s) is a program, opaque to MML, which tells the GPM to time out after 100 seconds (ie if the frequency table is very large). More elaborate programs will be encountered next.

5 Review of Metamodeling Directives

MML is a very small language. Its power comes from the fact that it is *extensible*. The key language constructs are

```
CREATE POPULATION <pop-name> (
    <variable> <measurement-type>[, ...])

CREATE GPM <gpm-name> FOR <pop-name> USING <gpm>(
    GENERATE (<variables> [, ...])
    [GIVEN (<variables> [, ...])]
    PROGRAM ([<blob>])
)

INITIALIZE <k> INSTANCES OF <gpm-name>

INCORPORATE INTO <pop|gpm> OBSERVATIONS <data-source>
UNINCORPORATE FROM <pop|gpm> OBSERVATIONS <data-source>

UPDATE SCHEMA FOR <gpm> (
    [GENERATE (<variables> [, ...])]
    [GIVEN (<variables> [, ...])]
    ([<program>])
)

ANALYZE <gpm> ([<program>])
```

The full set of invariants and constraints that govern rules for the MML directives is left for a separate technical document. One quick example is that, for a GPM with GIVEN variables, every incorporated measurement must include fully observed data for those variables. Many other rules relating to primary key constraints, etc, exist.

6 Compositional Metamodeling

The running example of the swallows population will be investigated further. Figure 6.1 shows the data to be modeled. The illustration will proceed by incrementally building more complex probabilistic models using simple primitives in the metamodeling language.

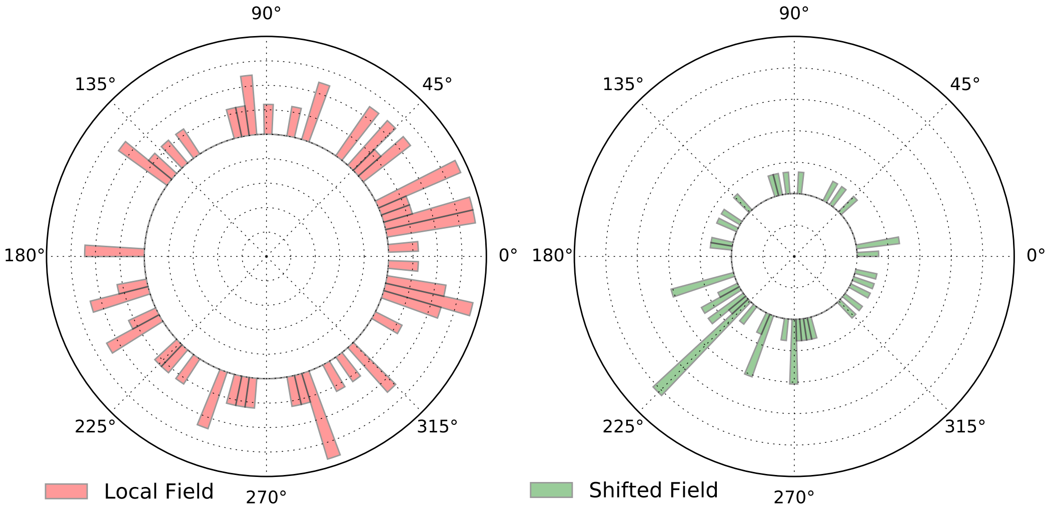

Figure 6.1: Spatial histogram of the `orientation` of birds from the `swallows` population, when the magnetic field (ie `treatment` variable) was local (left figure) and shifted (right figure). The objective of the original study [GB04] was to determine if swallows can use magnetic information for compass orientation.

6.1 Univariate Density Estimation

The `orientation` variable is `CIRCULAR` and takes values in the set $[0, 2\pi)$. One of the most common strategies in statistics is to approximate the distribution of variables using a normal distribution.

```
CREATE GPM ort-norm FOR swallows USING normal(
    GENERATE (orientation)
    PROGRAM (prior nig-normal, hyperparameters learn))
INITIALIZE 6 INSTANCES OF ort-norm
INCORPORATE INTO ort-norm OBSERVATIONS (SELECT orientation FROM swallows)
ANALYZE ort-norm (TIME = 100s)
```

The `normal` GPM type is instructed to use a collapsed normal-inverse- gamma (NIG) normal prior on its parameters (μ, σ^2), and perform hyper- parameter inference. Initializing multiple models is advantageous because learning hyper-parameters can be done only approximately.

Evaluating the posterior distribution after `ANALYZE` can be achieved using a BQL query. Refer to the left plot in Figure 6.2 for the output of the `PLOT` command.

```
CREATE TABLE xvalues AS LINSPACE(0, 6.28, 100)
PLOT(xvalues, ESTIMATE PROBABILITY OF orientation = xvalues FROM ort-norm)
```

An important point to note is that since the measurements in the dummy table `xvalues` have primary keys that do not correspond to any previously incorporated measurement (from the `swallows` population), the `PROBABILITY OF` BQL queries are evaluated assuming unobserved members.

A particular issue with the GPM shown in Figure 6.2 is that probability mass is assigned to points outside the interval $[0, 2\pi)$, effectively ignoring knowledge that `orientation` is `CIRCULAR` and takes values on a bounded interval.

Persisting with assumption of normality, this information can be treated as a *conditioning event*, a programmatic command that the `normal` GPM supports by specifying a *truncation interval*[5]. By using `UPDATE SCHEMA`, all previous incorporated measurements are maintained.

[5] This GPM is implemented fully in `gpmcc`, although conjugacy is lost due to the truncation and inference is slower.

```
UPDATE SCHEMA FOR ort-norm(
    truncate (0, 6.28))
ANALYZE ort-norm (TIME = 100s)
```

The same BQL query can be reused for visual comparison of the effect of the new modeling command on the posterior density, as shown in Figure 6.2.

```
PLOT(xvalues, ESTIMATE PROBABILITY OF orientation = xvalues FROM ort-norm)
```

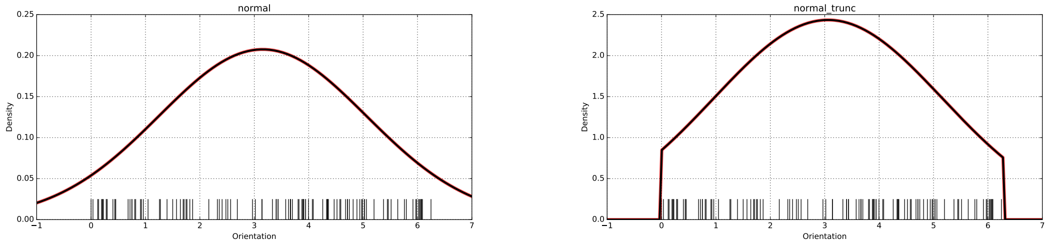

Figure 6.2: Posterior distribution over `orientation` learned by the `ort-norm` GPM before (left) and after (right) updating its schema with `truncate(0, 6.28)`. Observe the value of the posterior density on the y-axis increases by a factor of 10, as probability mass is reassigned to the interval.

6.2 Mixture Modeling of Univariate Densities

Both plots in Figure 6.2 illustrate a poor fit to the data, which can be seen visually. There are many samples at the edges of the interval, and only a few samples in the center. However the posterior mode of the `ort-norm` GPM is forced to the center, and assigns low density to high density regions and vice versa.

To deal with the perceived *multi-modality*, one can *compose* many instances of the truncated `normal` GPM by creating a `dp-mixture` GPM.

```
CREATE GPM ort-norm-mixture FOR swallows USING dp-mixture(
    GENERATE (orientation)
    PROGRAM (
        MODEL orientation AS normal
            (prior nig-normal, hyperparameters learn)))

INITIALIZE 2 INSTANCES OF ort-norm-mixture
INCORPORATE INTO ort-norm-mixture OBSERVATIONS (SELECT orientation FROM swallows)
ANALYZE ort-norm-mixture (
    TRANSITION ALL
    ITERATIONS = 100)
```

The `PROGRAM` of a `dp-mixture` instructs to use the `normal` GPM as the parametric family to mix, with a NIG-normal base measure. Any other commands accepted by the `normal` GPM, such as the truncation range can be used as well. In general, the current implementation of `dp-mixture` allows composition any GPMs which are univariate (one `GENERATE`) and unconditional (no `GIVENS`). Moreover, the GPMs must be of the same *type*. A GPM which relaxes these constraint will be encountered shortly.

Observe that the program in `ANALYZE` contains the directive `TRANSITION ALL`, which tell to Gibbs cycle through **all** its inference kernels, (row assignments, uncollapsed column parameters, column hyper- parameters, CRP α), for 100 sweeps.

As before, the same BQL query can be used for visual evaluation. However, this time individual models are targeted (specified by `USING MODEL n`), rather than aggregating (the default behavior). This query allows visualization of the degree of agreement between independent GPM instances of `ort-norm-mixture`.

```
PLOT(xvalues, ESTIMATE PROBABILITY OF orientation = xvalues FROM ort-norm-mixture
    USING MODEL 1)
PLOT(xvalues, ESTIMATE PROBABILITY OF orientation = xvalues FROM ort-norm-mixture
    USING MODEL 2)
```

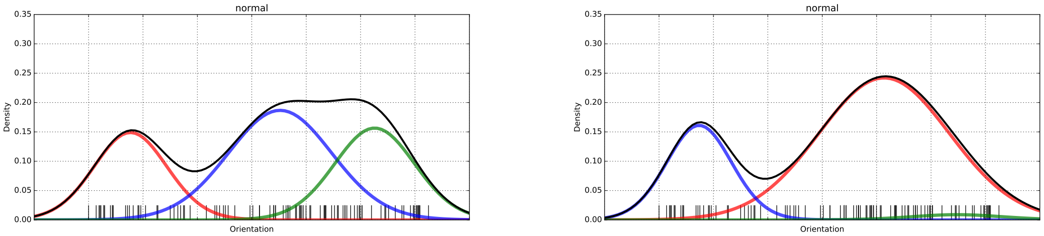

Figure 6.3: Posterior distribution over `orientation` learned by two independent model instances of the `ort-norm-mixture` GPM. Colored lines represent individual instances of the `normal` GPM that live inside the the `dp-mixture` GPM.

7 Beyond Normality and Density Estimation Comparison

Three modeling approaches for the generative process for the `orientation` variable have been explored thus far

- Univariate normal.
- Univariate normal, truncated to the `CIRCULAR` range.
- DP mixture of univariate normals, without truncation.

Are there reasons to consider other GPMs that are not normal and have properties that better match the `orientation` variable? Consider the following possibilities:

- Scaling a distribution on an arbitrary bounded interval to the circular interval $[0, 2\pi)$, such as the standard beta on $(0, 1)$[6].

```
CREATE GPM ort-beta-mixture FOR swallows USING dp-mixture(
    GENERATE (orientation)
    PROGRAM (
        MODEL orientation AS beta
            (scale(0, 2pi), hyperparameters fixed)))
```

- Using a distribution especially appropriate for the `CIRUCLAR` measurement type, such as the Vonmises distribution[7].

```
CREATE GPM ort-vonmises FOR swallows USING vonmises(
    GENERATE (orientation))
```

A common metric, highly flawed, for evaluating the performance of density estimators is segregating data into a *training* set and *testing* set, incorporating and runing analysis on the training set, and then evaluating the *predictive likelihood* on the *test set*. A more Bayesian approach is to observe all the data and then try to compute the *marginal likelihood* of the measurements after analysis. The following program illustrates how MML and BQL can be used to compare these metrics for the four GPMs `ort-normal-mixture`, `ort-trunc-mixture`, `ort-beta-mixture`, and `ort-vonmises`, assuming they have been `CREATED` and `INITIALIZED` with 28 instances each.

[6] Supported by gpmcc.

[7] Available in gpmcc.

The square brackets with multiple GPM names are constructs permitted by MML and BQL, and are syntactic sugar for repeating the same directive or query for each GPM in the list.

```
-- Use first 80 observations for the training set.
INCORPORATE INTO [ort-norm-mixture | ort-trunc-mixture | ort-beta-mixture |
    ort-vonmises] OBSERVATIONS (SELECT orientation FROM swallows WHERE pk <= 80)
-- Allow 100 seconds of analysis.
ANALYZE [ort-norm-mixture | ort-trunc-mixture | ort-beta-mixture | ort-vonmises
    (TIMEOUT=100s)
-- Evaluate predictive likelihood on 20 training samples.
ESTIMATE PROBABILITY OF orientation =
    (SELECT orientation FROM swallows WHERE pk BETWEEN (80, 100))
    FROM [ort-norm-mixture | ort-trunc-mixture | ort-beta-mixture | ort-vonmises]
-- Evaluate the marginal likelihood.
ESTIMATE MARGINAL LIKELIHOOD FROM
    [ort-norm-mixture | ort-trunc-mixture | ort-beta-mixture | ort-vonmises]
```

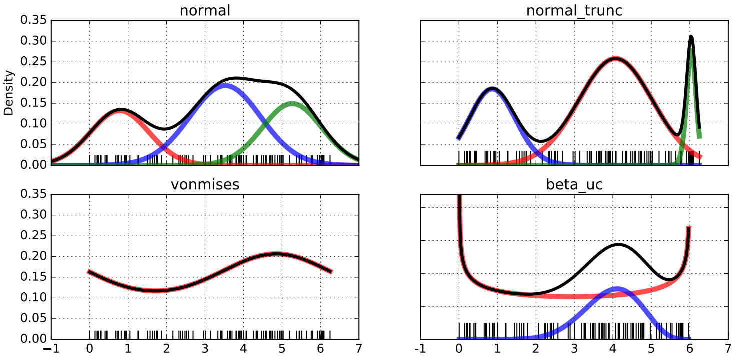

Figure 7.1: Samples from the posterior distribution over `orientation` learned by the four competing GPMs Only the `vonmises` is a single univariate estimator, while the others are `dp-mixture` of their respective densities.

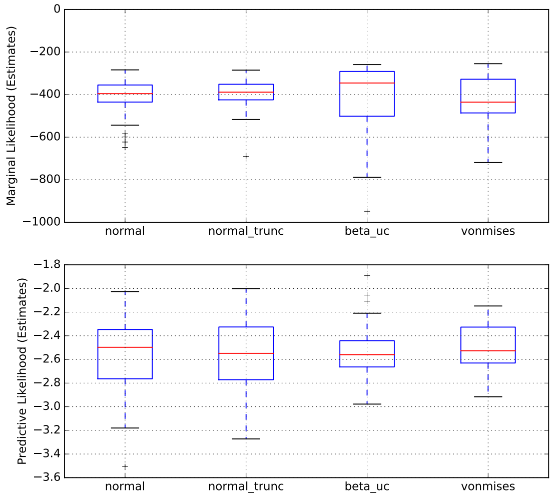

Figure 7.2: Distribution of estimates of the marginal likelihood (top) and predictive likelihood (bottom) on the held out test set. Uncertainty over the predictive likelihood is over the test samples. Uncertainty in the marginal likelihood is over the 28 individual models in each ensemble GPM. Surprisingly, there does not appear to be major differences in the GPM performance using either of these two metrics.

8 Learning Joint Distributions And Evaluating Classification Performance Using a GPM

Recall that the `swallows` population contains the two variables `treatment` and `orientation`. So far, these two variables have been modeled separately. The `crosscat` GPM is a generalization of `dp-mixture` for multiple `GIVEN` variables.

```
CREATE GPM swallows-crosscat FOR swallows USING crosscat(
    GENERATE (treatment, orientation)
    PROGRAM (
        MODEL treatment AS bernoulli
        MODEL orientation AS normal))
```

One method to measure how well `swallows-crosscat` has learned the joint distribution is to ask the GPM to classify unseen `orientation` measurements into either *local* or *shifted* `treatment`[8]. The following MML code sets up and runs the experiment in BQL.

[8] It should be noted that this task is very difficult (see Figure 6.1) and is included only for illustrative purposes.

```
-- Split the population.
CREATE POPULATION swallows-train AS (SELECT * FROM SWALLOWS LIMIT 80)
CREATE POPULATION swallows-test AS (SELECT * FROM SWALLOWS WHERE pk BEWTEEN (80, 100))
-- Incorporate.
INCORPORATE INTO swallows-crosscat
    OBSERVATIONS swallows-train
-- Analyze.
ANALYZE swallows-crosscat (TIMEOUT 100s)
-- Obtain probability of treatment = local for orientations in the test set.
CREATE POPULATION soft-predictions AS
    ESTIMATE PROBABILITY OF (treatment = ’local’)
            GIVEN orientation = (SELECT orientation FROM swallows-test) FROM swallows-crosscat
-- Use a hard classification rule with a threshold of 0.5.
CREATE POPULATION hard-predictions AS
    SELECT (treatment FROM swallows-test) AS true-labels,
    CASE WHEN prob-of-local > 0.5 THEN ’local’ ELSE ’shifted’ END
        AS predicted-labels
    FROM soft-predictions
```

The population `hard-predictions` now contains a 0-1 prediction for `treatment` on all measurements in the test set. There are many metrics for evaluating binary classification. It was in proposed in [Wal06] that the *mutual information* between true labels and predictions is a metric with desirable properties. A new GPM, called `classification-evaluation` based on the `frequency` GPM is ideal for computing this metric.

```
-- Create the GPM for evaluating classification
CREATE GPM classification-evaluator FOR hard-predictions USING frequency(
    GENERATE (true-labels, predicted-labels))
INCORPORATE INTO classification-evaluator OBSERVATIONS hard-predictions
ANALYZE classification-evaluator
-- Compute the mutual information of labels and predictions.
ESTIMATE MUTUAL INFORMATION of true-labels, predicted-labels FROM classification-evaluator
```

To visualize the results, the experiment above was implemented for four GPMs based on `crosscat` using different internal GPMs for `orientation` (`normal`, `normal-trunc` `vonmises` and `beta`) for various thresholds. Figure 8.1 summarizes the results of this experiment.

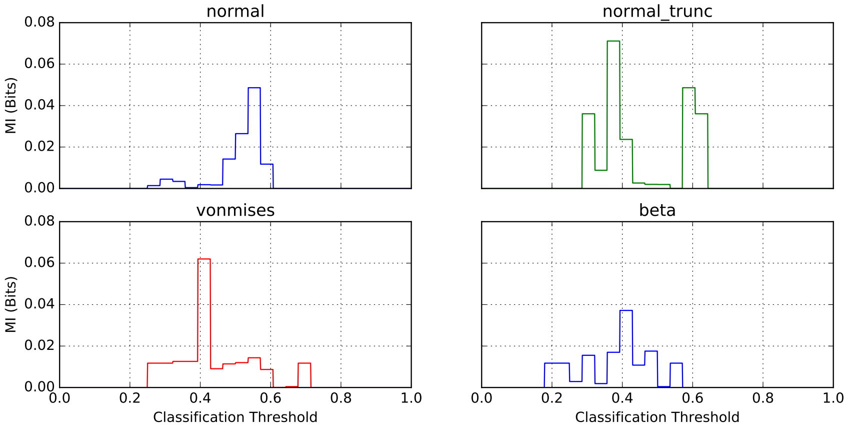

Figure 8.1: Mutual information of true labels and predicted labels of `treatment` computed by the `classification-evaluator` GPM. The test set has 20 observations, and the classification threshold varies between 0 and 1 in increments of .1. The title of each subplot shows the GPM that `crosscat` was instructed to use to model the `orientation` variable. No error bars are shown since only one was test-train split was used.

9 Todo

- Introduce some discriminative models (not density estimators) such as SVM, regression, etc as standalone GPMs.
- Discuss how this MML version easily allows for computationally derived column to be added, modeled, etc with an example.
- Introduce crosscat.
- Introduce compositor network and its CMML language.
- Explain the formalism of hybrid probabilistic-discriminative models

    Unlocalized (imputation only).

    Explicitly localized (no feedback from discriminative to generative GPM).

    Implicitly localized (fully feedback from discriminative to generative GPM).

References

[GB04] Dimitri Giunchi and N. Emilio Baldaccini. Orientation of juvenile barn swallows (hirundo rustica) tested in emlen funnels during autumn migration. *Behavioral Ecology and Sociobiology*, 56(2):124–131, 2004.

[Wal06] H Wallach. Evaluation metrics for hard classifiers. *Cavendish Lab., Univ. Cambridge, Cambridge, online available at http://www. inference. phy. cam. ac. uk/hmw26/, Tech. Rep. Technical Report*, 2006.