Automated Model Construction through Compositional Grammars

David Duvenaud
Cambridge University
Computational and Biological Learning Lab
April 22, 2013
OUTLINE

- Motivation
- Automated structure discovery in regression
  - Gaussian process regression
  - Structures expressible through kernel composition
  - A massive missing piece
  - grammar & search over models
  - Examples of structures discovered
- Automated structure discovery in matrix models
  - expressing models as matrix decompositions
  - grammar & special cases
  - examples of structures discovered on images
Talk based on two papers:

- **Structure Discovery in Nonparametric Regression through Compositional Kernel Search** [ICML 2013]
  David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, Zoubin Ghahramani

- **Exploiting compositionality to explore a large space of model structures** [UAI 2012]
  Roger B. Grosse, Ruslan Salakhutdinov, William T. Freeman, Joshua B. Tenenbaum
Motivation

- Models today built by hand, or chosen from a fixed set.
  - Fixed set sometimes not that rich
    - Just being nonparametric sometimes isn’t good enough
    - to learn efficiently, need to have a rich prior that can express most of the structure in your data.
  - Building by hand requires expertise, understanding of the dataset.
  - Follows cycle of: propose model, do inference, check model fit
    - Propose new model
    - Do inference
    - Check model fit

- Andrew Gelman asks: How would an AI do statistics?
- It would need a language for describing arbitrarily complicated models, a way to search over those models, and a way of checking model fit.
Finding Structure in GP Regression

(\text{Lin} \times \text{SE} + \text{SE} \times (\text{Per} + \text{RQ}))

\begin{align*}
\text{Lin} \times \text{SE} \\
\text{SE} \times \text{Per} \\
\text{SE} \times \text{RQ} \\
\text{Residuals}
\end{align*}
GAUSSIAN PROCESS REGRESSION

Assume $X, y$ is generated by $y = f(X) + \epsilon_\sigma$
A GP prior over $f$ means that, for any finite set of points $X$,

$$p(f(x)) = \mathcal{N}(\mu(X), K(X, X))$$

where

$$K_{ij} = k(X_i, X_j)$$

is the covariance function or kernel. $k(x, x') = \text{cov} [f(x), f(x')]$
Assume $X, y$ is generated by $y = f(X) + \epsilon_\sigma$

A GP prior over $f$ means that, for any finite set of points $X$,

$$p(f(x)) = \mathcal{N}(\mu(X), K(X, X))$$

where

$$K_{ij} = k(X_i, X_j)$$

is the covariance function or kernel. $k(x, x') = \text{cov} [f(x), f(x')]$

Typically, kernel says that nearby $x_1, x_2$ will have highly correlated function values $f(x_1), f(x_2)$:

$$k_{SE}(x, x') = \exp(-\frac{1}{2\theta}|x-x'|^2)$$
function simple_gp_sample

    % Choose a set of x locations.
    N = 100;
    x = linspace( -2, 2, N);

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros( N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );

    plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
function simple_gp_sample

% Choose a set of x locations.
N = 100;
x = linspace(-2, 2, N);

% Specify the covariance between function values, depending on their location.
for j = 1:N
    for k = 1:N
        sigma(j,k) = covariance( x(j), x(k) );
    end
end

% Specify that the prior mean of f is zero.
mu = zeros(N, 1);

% Sample from a multivariate Gaussian.
f = mvnrnd( mu, sigma );
plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp(-0.5*(x - y)^2);
end
function simple_gp_sample

    % Choose a set of x locations.
    N = 100;
    x = linspace( -2, 2, N);

    % Specify the covariance between function
    % values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros(N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );
    plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
function simple_gp_sample

    % Choose a set of x locations.
    N = 100;
    x = linspace( -2, 2, N );

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros( N, 1 );

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );

    plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
function simple_gp_sample

% Choose a set of x locations.
N = 100;
x = linspace( -2, 2, N);

% Specify the covariance between function
% values, depending on their location.
for j = 1:N
  for k = 1:N
    sigma(j,k) = covariance( x(j), x(k) );
  end
end

% Specify that the prior mean of f is zero.
mu = zeros( N, 1);

% Sample from a multivariate Gaussian.
f = mvnrnd( mu, sigma );

plot(x, f);
end

% Periodic covariance function.
function c = covariance( x, y )
  c = exp( -0.5*( sin(( x - y )*1.5).^2 ));
end
$$f(x^*)|\mathbf{X}, y \sim \mathcal{N}(k(x^*, \mathbf{X})K^{-1}y,$$

$$k(x^*, x^*) - k(x^*, \mathbf{X})K^{-1}k(\mathbf{X}, x^*))$$

With SE kernel:
\[
f(x^*)|\mathbf{X}, \mathbf{y} \sim \mathcal{N}(k(x^*, \mathbf{X})K^{-1}\mathbf{y},
\]
\[
k(x^*, x^*) - k(x^*, \mathbf{X})K^{-1}k(\mathbf{X}, x^*)
\]

With SE kernel:
\[ f(x^*) | X, y \sim \mathcal{N}(k(x^*, X)K^{-1}y, k(x^*, x^*) - k(x^*, X)K^{-1}k(X, x^*)) \]

With SE kernel:
**Conditional Posterior**

\[
f(x^*) | X, y \sim \mathcal{N}(k(x^*, X)K^{-1}y, k(x^*, x^*) - k(x^*, X)K^{-1}k(X, x^*))
\]

With SE kernel:
$f(x^*)|X, y \sim \mathcal{N}(k(x^*, X)K^{-1}y,$

$k(x^*, x^*) - k(x^*, X)K^{-1}k(X, x^*))$

With SE kernel:
Kernel Choice is Important

- Kernel determines almost all the properties of the prior.
- Many different kinds, with very different properties:

Squared-exp (SE)  |  local variation  |  Periodic (PER)  |  repeating structure
Linear (LIN)       |  linear functions |  Rational-quadratic (RQ) |  multi-scale variation
KERNELS CAN BE COMPOSED

- Two main operations: adding, multiplying

<table>
<thead>
<tr>
<th>Lin × Lin</th>
<th>quadratic functions</th>
<th>SE × Per</th>
<th>locally periodic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lin + Per</td>
<td>periodic with trend</td>
<td>SE + Per</td>
<td>periodic with noise</td>
</tr>
</tbody>
</table>
Kernels can be composed

- Can be composed across multiple dimensions

\[
\text{LIN} \times \text{SE} \quad \text{increasing variation} \quad \text{LIN} \times \text{PER} \quad \text{growing amplitude}
\]

\[
\text{SE}_1 \, + \, \text{SE}_2 \quad f_1(x_1) \, + \, f_2(x_2) \quad \text{SE}_1 \, \times \, \text{SE}_2 \quad f(x_1, x_2)
\]
SPECIAL CASES

Bayesian linear regression  \( \text{LIN} \)
Bayesian polynomial regression  \( \text{LIN} \times \text{LIN} \times \ldots \)
Generalized Fourier decomposition  \( \text{PER} + \text{PER} + \ldots \)
Generalized additive models  \( \sum_{d=1}^{D} \text{SE}_d \)
Automatic relevance determination  \( \prod_{d=1}^{D} \text{SE}_d \)
Linear trend with deviations  \( \text{LIN} + \text{SE} \)
Linearly growing amplitude  \( \text{LIN} \times \text{SE} \)
Appropriate kernels are necessary for extrapolation

- SE kernel → basic smoothing.
- Richer kernels means richer structure can be captured.
Kernels are hard to choose

- Given the diversity of priors available, how to choose one?
- Standard GP software packages include many base kernels and means to combine them, but *no default kernel*
- Software can’t choose model for you, you’re the expert (?)
Kernels are hard to construct

- Carl devotes 4 pages of his book to constructing a custom kernel for CO2 data
- requires specialized knowledge, trial and error, and a dataset small and low-dimensional enough that a human can interpret it.
- In practice, most users can’t or won’t make custom kernel, and SE kernel became *de facto* standard kernel through inertia.
RECAP

- GP Regression is a powerful tool
- Kernel choice allows for rich structure to be captured - different kernels express very different model classes
- Composition generates a rich space of models
- Hard & slow to search by hand
- Can kernel specification be automated?
Compositional Structure Search

- Define grammar over kernels:
  - \( K \to K + K \)
  - \( K \to K \times K \)
  - \( K \to \{ \text{SE}, \text{RQ}, \text{LIN}, \text{PER} \} \)

- Search the space of kernels greedily by applying production rules, checking model fit (approximate marginal likelihood).
Compositional Structure Search

No structure

SE

RQ

SE + RQ

PER + RQ

SE + PER + RQ

SE × (PER + RQ)

PER

LIN

PER × RQ
EXAMPLE SEARCH: MAUNA LUA CO₂
**Example Decomposition: Mauna Loa CO₂**

\[
(\text{Lin} \times \text{SE} + \text{SE} \times (\text{Per} + \text{RQ}))
\]

- **Lin × SE**
- **SE × Per**
- **SE × RQ**
- **Residuals**
Suppose functions $f_1, f_2$ are draw from independent GP priors, $f_1 \sim \mathcal{GP}(\mu_1, k_1), f_2 \sim \mathcal{GP}(\mu_2, k_2)$. Then it follows that

$$f := f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2)$$

Sum of kernels is equivalent to sum of functions. Distributivity means we can write compound kernels as sums of products of base kernels:

$$\text{SE} \times (\text{RQ} + \text{LIN}) = \text{SE} \times \text{RQ} + \text{SE} \times \text{LIN}.$$
Example Decomposition: Airline

\[ SE \times (\text{Lin} + \text{Lin} \times (\text{Per} + \text{RQ})) \]

\[ SE \times \text{Lin} \]

\[ SE \times \text{Lin} \times \text{Per} \]

\[ SE \times \text{Lin} \times \text{RQ} \]

Residuals

\[ \text{Residuals} \]
Example: Sunspots
Can express change in covariance:

Periodic changing to SE
Can express change in covariance:

SE changing to linear
Choosing form of kernel is currently done by hand.
Compositions of kernels lead to more interesting priors on functions than typically considered.
A simple grammar specifies all such compositions, and can be searched over automatically.
Composite kernels lead to interpretable decompositions.
Previous work introduced idea of grammar of compositions:

- Exploiting compositionality to explore a large space of model structures [UAI 2012]
  Roger B. Grosse, Ruslan Salakhutdinov, William T. Freeman, Joshua B. Tenenbaum
- Slides that follow are from Roger Grosse
Matrix decomposition

- Cluster assignments
- Cluster centers
- Within-cluster variation

M + G + G

Input matrix
Recursive Matrix decomposition

- Main idea: Matrices can be recursively decomposed
- Example: Co-clustering by clustering cluster assignments.
**Building Blocks**

- **Gaussian (G)**
  - $\lambda_i \sim \text{Gamma}(a, b)$
  - $\nu_j \sim \text{Gamma}(a, b)$
  - $u_{ij} \sim \text{Normal}(0, \lambda_i^{-1} \nu_j^{-1})$ *

  * variance parameters shared between input rows/columns

- **Bernoulli (B)**
  - $p_j \sim \text{Beta}(\alpha, \beta)$
  - $u_{ij} \sim \text{Bernoulli}(p_j)$

- **Multinomial (M)**
  - $\pi \sim \text{Dirichlet}(\alpha)$
  - $u_i \sim \text{Multinomial}(\pi)$

- **Integration (C)**
  - $u_{ij} = \begin{cases} 
  1 & \text{if } i \geq j \\
  0 & \text{otherwise} 
  \end{cases}$
Matrix decomposition: Grammar

Starting symbol: $G$

Production rules:

- **clustering**
  - $G \rightarrow MG + G \mid G M^T + G$
  - $M \rightarrow MG + G$

- **low rank**
  - $G \rightarrow GG + G$

- **binary features**
  - $G \rightarrow BG + G \mid GB^T + G$
  - $B \rightarrow BG + G$
  - $M \rightarrow B$

- **linear dynamics**
  - $G \rightarrow CG + G \mid GC^T + G$

- **sparsity**
  - $G \rightarrow \exp(G) \circ G$
Matrix decomposition: Special cases

\[(MG + G)(GM^T + G) + G\]
Bayesian clustered tensor factorization
(Sutskever et al., 2009)

\[B(GB^T + G) + G\]
binary matrix factorization
(Meeds et al., 2006)

\[B(GG + G)G + G\]
sparse coding
(e.g. Olshausen and Field, 1996)

\[B(G + G) + G\]
binary features
(Griffiths and Ghahramani, 2005)

\[MG + G\]
clustering
(e.g. Kemp et al., 2006)

\[MG + G\]
clustering

\[(exp(GG + G) \circ G)G + G\]
dependent gaussian scale mixture
(e.g. Karklin and Lewicki, 2005)

\[(exp(G) \circ G)G + G\]
linear dynamical system

\[CG + G\]
random walk

\[GG + G\]
low-rank approximation
(Salakhutdinov and Mnih, 2008)

\[G\]
no structure
Evolution of Image Models

Modeling images as linear combinations of uncorrelated basis functions gives a Fourier representation.

Modeling the sparse distribution of the linear reconstruction coefficients gives oriented edges.

Modeling the dependencies in the sparsity pattern gives a high-level texture model.

Bossomaier and Snyder, 1987

Olshausen and Field, 1996

Karklin and Lewicki, 2005
**Application to Natural Image Patches**

Data: 1,000 12x12 patches from 10 blurred and whitened images.

Model 1: Low-rank approximation (PCA).

Model 2: Sparsify coefficients to get sparse coding.

Model 3: Model dependencies between scale variables.
CONCLUSIONS

- Model-building is currently done mostly by hand.
- Grammars over composite structures are a simple way to specify open-ended model classes.
- Composite structures often imply interpretable decompositions of the data.
- Searching over these model classes is a step towards automating statistical analysis.
CONCLUSIONS

- Model-building is currently done mostly by hand.
- Grammars over composite structures are a simple way to specify open-ended model classes.
- Composite structures often imply interpretable decompositions of the data.
- Searching over these model classes is a step towards automating statistical analysis.

Thanks!
• Algorithmic information theory, e.g. Solomonoff induction (Solomonoff, 1964)

• Structure learning in other domains
  • Bayesian networks (e.g. Teyssier and Koller, 2005)
  • Markov random fields (e.g. Lee et al., 2006)

• Learning the form of graph embeddings (Kemp and Tenenbaum, 2008)

• Equation discovery
  • BACON knowledge discovery engine (Langley, Simon, and Bradshaw, 1984)
  • exploiting context-free grammar (Todorovski and Dzeroski, 1997)

• Matrix factorization frameworks
  • Exponential family PCA (Collins et al., 2002)
  • Roweis and Ghahramani (1999)
  • Singh and Gordon (2008)
EXTRAPOLATION
## Multi-D Interpolation

<table>
<thead>
<tr>
<th>Method</th>
<th>bach</th>
<th>concrete</th>
<th>puma</th>
<th>servo</th>
<th>housing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>1.031</td>
<td>0.404</td>
<td>0.641</td>
<td>0.523</td>
<td>0.289</td>
</tr>
<tr>
<td>GAM</td>
<td>1.259</td>
<td>0.149</td>
<td>0.598</td>
<td>0.281</td>
<td>0.161</td>
</tr>
<tr>
<td>HKL</td>
<td>0.199</td>
<td>0.147</td>
<td>0.346</td>
<td>0.199</td>
<td>0.151</td>
</tr>
<tr>
<td>GP SE-ARD</td>
<td>0.045</td>
<td>0.157</td>
<td>0.317</td>
<td>0.126</td>
<td><strong>0.092</strong></td>
</tr>
<tr>
<td>GP Additive</td>
<td>0.045</td>
<td>0.089</td>
<td>0.316</td>
<td>0.110</td>
<td>0.102</td>
</tr>
<tr>
<td>Structure Search</td>
<td>0.044</td>
<td>0.087</td>
<td>0.315</td>
<td>0.102</td>
<td><strong>0.082</strong></td>
</tr>
</tbody>
</table>
1. **Gaussian (G).** Entries are independent Gaussians:

\[ u_{ij} \sim \text{Gaussian}(0, \lambda_i^{-1}\lambda_j^{-1}). \]

This is our most generic component prior, and gives a way of deferring or ignoring structure.\(^1\)

2. **Multinomial (M).** Rows are independent multinomials, with one 1 and the rest 0’s:

\[ \pi \sim \text{Dirichlet}(\alpha) \quad u_i \sim \text{Multinomial}(\pi). \]

This is useful for clustering models, where \( u_i \) determines the cluster assignment for the \( i^{th} \) row.
3. **Bernoulli (B).** Entries are independent Bernoulli's:

\[ \pi_j \sim \text{Beta}(a, b) \quad u_{ij} \sim \text{Bernoulli}(\pi_j). \]

This is useful for binary latent feature models.

4. **Integration matrix (C).** Entries below the diagonal are deterministically 1:

\[ u_{ij} = 1_{i \geq j}. \]

This is useful for modeling temporal structure, as multiplying by this matrix has the effect of cumulatively summing the rows. (Mnemonic: C for “cumulative.”)