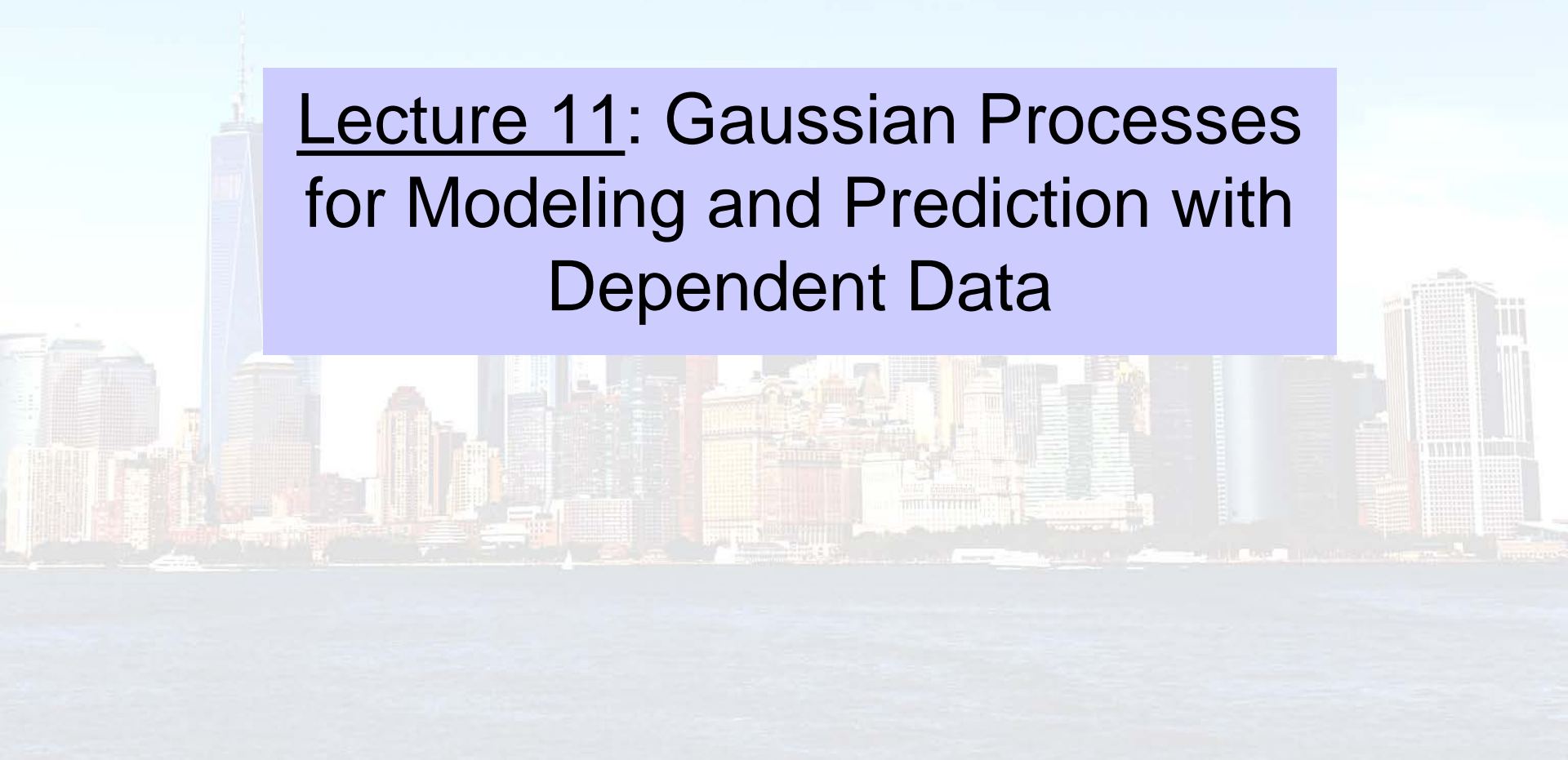


Machine Learning for Cities

CUSP-GX 5003.001, Spring 2022

Lecture 11: Gaussian Processes for Modeling and Prediction with Dependent Data



Why model dependent data?

Typical machine learning assumption: data points are drawn i.i.d. (independently, and identically distributed) from some distribution.

How reasonable is this assumption for urban systems?



Congestion may propagate from an initial traffic event, leading to spatial and temporal dependence.



Environmental monitoring: we expect similar sensor readings if the measurements are close together in space and time.

While dependent data can arise in many contexts (such as structured prediction or correlated data streams), the most common sources are dependence over time (serial autocorrelation) and spatial correlation.

Why model dependent data?

Typical machine learning assumption: data points are drawn i.i.d. (independently, and identically distributed) from some distribution.

How reasonable is this assumption for urban systems?



Congestion may propagate from an initial traffic event, leading to spatial and temporal dependence.



Environmental monitoring: we expect similar sensor readings if the measurements are close together in space and time.

First law of geography: “Everything is related to everything else, but nearby things are more related than more distant things.”

Why model dependent data?

Typical machine learning assumption: data points are drawn i.i.d. (independently, and identically distributed) from some distribution.

How reasonable is this assumption for urban systems?



Congestion may propagate from an initial traffic event, leading to spatial and temporal dependence.



Environmental monitoring: we expect similar sensor readings if the measurements are close together in space and time.

Failing to account for dependence can reduce prediction accuracy, as well as leading us to overestimate the anomalousness of a discovered pattern.

(“Our last 50 air quality readings have been below acceptable levels...”)

Some simple approaches...

Statistical tests for correlation

These just give a yes or no answer to the question, are my data points correlated?

Many different tests for temporal correlation (e.g., Durbin-Watson) and spatial correlation (Moran's I), as well as space-time interaction (Knox and Mantel tests).

Kernel regression

Choose kernel function $K(x_i, x_j)$: decreasing function of distance in space and/or time, for example, $\exp(-||x_i - x_j||^2)$.

Estimate for point x_i is a weighted average of observations at all other points x_j , each weighted by $K(x_i, x_j)$.

What's missing from these simple approaches?

- No underlying probabilistic model of the data
 - Can't provide confidence intervals for predictions.
 - Can't reason about joint probabilities or sample from the joint distribution.
 - Can't accurately estimate anomalousness of a point or set of pts.
- No ability to learn the correlation structure from data.
- Can't work with complex, structured data (limited to space and time).

What are Gaussian Processes?

Gaussian processes (GPs) are an approach for supervised learning (both regression and classification) with dependent data.

They can also be extended to many other tasks such as causal inference, long-range time series forecasting, and change detection.

GPs are a **function approximator**, that is, we want to learn a function $y = f(x)$ from a set of observed data points (x_i, y_i) .

Non-linear: flexibility to fit very complex functions.

Non-parametric: # of parameters increases with # of data points.

Bayesian: assume a **prior distribution** over functions $f(x)$.

Given observed data, compute **posterior distribution** over $f(x)$.

The mean of this distribution gives you a predicted $f(x)$ for any x .

You can also obtain confidence intervals for each prediction.

Univariate Gaussian distribution = distribution over scalars (real #'s)

→ Multivariate Gaussian distribution = distribution over vectors

→ Gaussian **process** = distribution over functions (infinite-dimensional vectors)

Multivariate Gaussian distributions

For more details, see Andrew Moore's tutorial on Gaussians,
<https://www.autonlab.org/media/tutorials/gaussian14.pdf>

Let random variable $\vec{x} = \langle x_1, \dots, x_M \rangle^T$. Then we define $\vec{x} \sim \text{Gaussian}(\vec{\mu}, \Sigma)$, where

the mean vector $\vec{\mu} = \langle \mu_1, \dots, \mu_M \rangle^T$ and the covariance matrix $\Sigma = \begin{bmatrix} \sigma_1^2 & \dots & \sigma_{1,M} \\ \dots & \dots & \dots \\ \sigma_{M,1} & \dots & \sigma_M^2 \end{bmatrix}$,

with the probability density function:

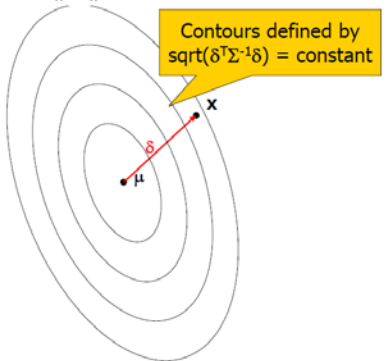
$$p(\vec{x}) = \frac{1}{(2\pi)^{M/2} \|\Sigma\|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1}(\vec{x} - \vec{\mu})\right).$$

Evaluating $p(\mathbf{x})$: Step 3

$$p(\mathbf{x}) = \frac{1}{2\pi \|\Sigma\|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

1. Begin with vector \mathbf{x}
2. Define $\delta = \mathbf{x} - \mu$
3. Count the number of contours crossed of the ellipsoids formed Σ^{-1}

$D = \text{this count} = \text{sqrt}(\delta^T \Sigma^{-1} \delta)$
 = Mahalanobis Distance between \mathbf{x} and μ



Contours defined by $\text{sqrt}(\delta^T \Sigma^{-1} \delta) = \text{constant}$

Copyright © Andrew W. Moore Slide 24

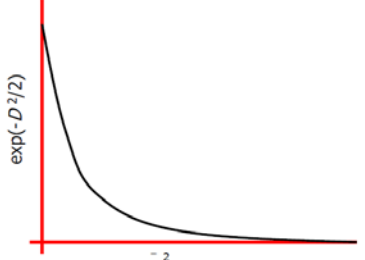
Evaluating $p(\mathbf{x})$: Step 5

$$p(\mathbf{x}) = \frac{1}{2\pi \|\Sigma\|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

1. Begin with vector \mathbf{x}
2. Define $\delta = \mathbf{x} - \mu$
3. Count the number of contours crossed of the ellipsoids formed Σ^{-1}

$D = \text{this count} = \text{sqrt}(\delta^T \Sigma^{-1} \delta)$
 = Mahalanobis Distance between \mathbf{x} and μ

4. Define $w = \exp(-D^2/2)$
5. Multiply w by $\frac{1}{\sqrt{2\pi} \|\Sigma\|^{1/2}}$ to ensure $\int p(\mathbf{x}) d\mathbf{x} = 1$



Copyright © Andrew W. Moore Slide 26

Gaussian Processes: the math

A **stochastic process** is a collection of random variables, $\{f(x): x \in X\}$, indexed by elements from some set X , known as the index set.

A **Gaussian process (GP)** is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution.

In particular, a collection of random variables $\{f(x) : x \in X\}$ is said to be drawn from a Gaussian process with mean function $m(\cdot)$ and covariance function $k(\cdot, \cdot)$ if for any finite set of elements $x_1, \dots, x_M \in X$, the associated finite set of random variables $f(x_1), \dots, f(x_M)$ have probability distribution, $\langle f(x_1), \dots, f(x_M) \rangle^T \sim \text{Gaussian}(\vec{\mu}, K)$, where $\vec{\mu} = \langle m(x_1), \dots, m(x_M) \rangle^T$ and

$$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_M) \\ \dots & \dots & \dots \\ k(x_M, x_1) & \dots & k(x_M, x_M) \end{bmatrix}.$$

We represent this as $f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$.

Gaussian Processes: an example

Let's consider the zero-mean Gaussian process $f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))$, where $k(\cdot, \cdot)$ is the RBF kernel, $k_{RBF}(x, x') = \exp(-\frac{1}{2\tau^2} \|x - x'\|^2)$. Note that τ is the bandwidth, and controls how quickly the covariance between points x and x' decreases as a function of the distance between points.

What do random functions sampled from this Gaussian process look like?

- Function values should be distributed around zero.
- Nearby points have high covariance, since $k_{RBF} \rightarrow 1$ when $\|x - x'\| \rightarrow 0$.
- Points that are far apart have low covariance, since $k_{RBF} \rightarrow 0$ when $\|x - x'\| \gg 0$.

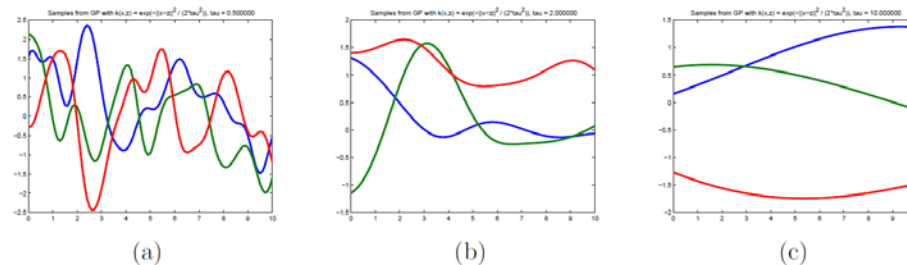


Figure 2: Samples from a zero-mean Gaussian process prior with $k_{SE}(\cdot, \cdot)$ covariance function, using (a) $\tau = 0.5$, (b) $\tau = 2$, and (c) $\tau = 10$. Note that as the bandwidth parameter τ increases, then points which are farther away will have higher correlations than before, and hence the sampled functions tend to be smoother overall.

Gaussian Process Regression

Given a set of M training examples, $(X, Y) = \{(x_i, y_i)\}$ for $i=1..M$,
and a set of M^* test examples, $(X^*, Y^*) = \{(x_i^*, y_i^*)\}$ for $i=1..M^*$.

Assume $y_i = f(x_i) + \varepsilon_i$, where ε_i are independent noise variables drawn from $N(0, \sigma^2)$, and $f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))$, where $k(\cdot, \cdot)$ is the kernel (e.g., RBF).

We can now compute the conditional distribution of the unobserved outputs Y^* given the observed outputs Y and all inputs (X and X^*).

This can be computed in closed form by specifying the joint distribution:
 $[Y \ Y^*] \mid X, X^* \sim N(0, K + \sigma^2 I)$, where K can be decomposed as:

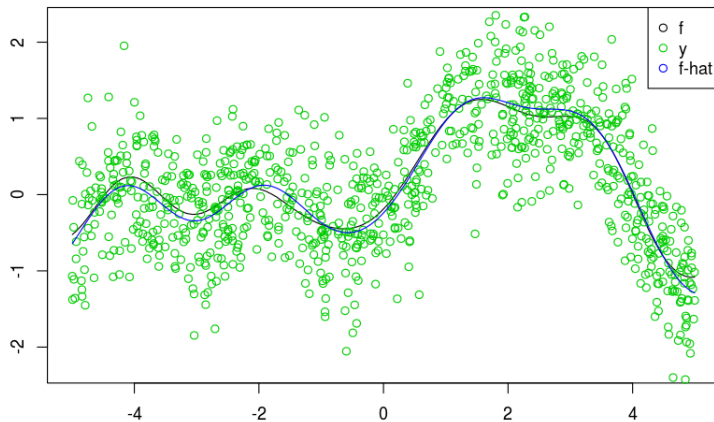
$$K = \begin{bmatrix} K(X, X) & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}.$$

Using properties of multivariate Gaussians, we can compute the conditional distribution $Y^* \mid Y, X, X^* \sim N(\bar{\mu}, \bar{K})$, where:

$$\bar{\mu} = K(X^*, X)(K(X, X) + \sigma^2 I)^{-1}Y$$

$$\bar{K} = K(X^*, X^*) - K(X^*, X)(K(X, X) + \sigma^2 I)^{-1}K(X, X^*)$$

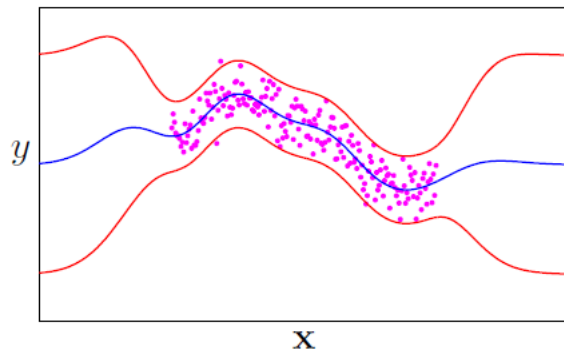
Gaussian Process Regression



(figure by Seth Flaxman)

Here's an example of fitting the true function $f(x)$ with the mean function $\bar{\mu}$ estimated from the GP.

With lots of data, GPs can very accurately estimate a complex, non-linear function of x .



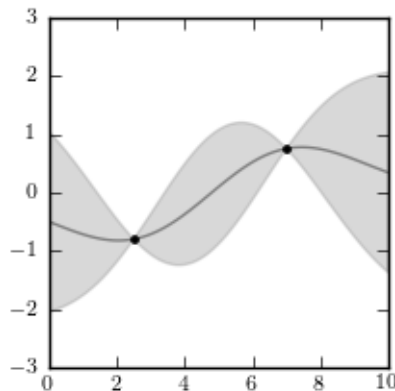
(figure by Zoubin Ghahramani)

We can also sample from the posterior predictive distribution $N(\bar{\mu}, \bar{K})$, and use the samples to compute a 95% confidence interval for prediction.

Computation is exact but expensive: $O(N^3)$ training, $O(N^2)$ test.
But various approximations can be used to speed things up.

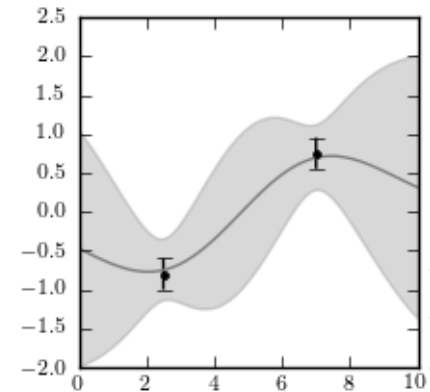
Learning the kernel parameters

Even for the simple case of an RBF kernel, we have two free parameters, kernel bandwidth τ and noise variance σ^2 . Choice of parameters matters a lot!

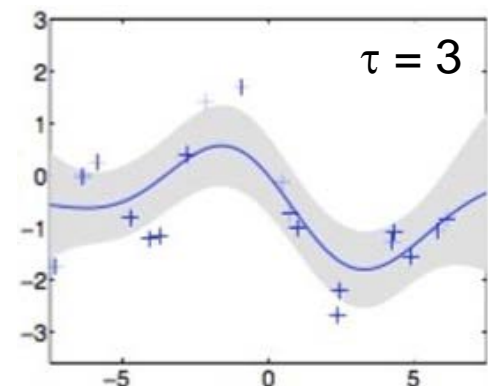
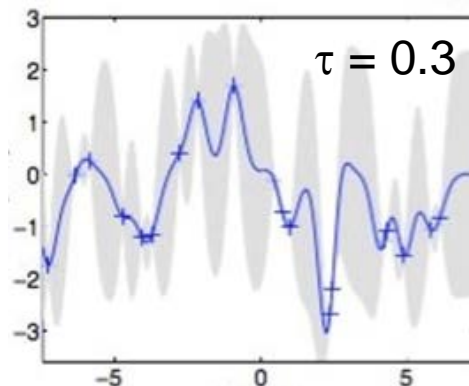
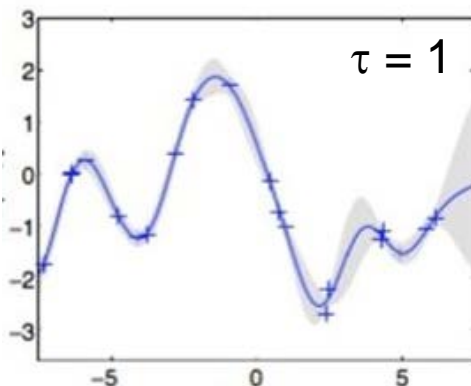


With $\sigma^2 = 0$, the GP fits every training data point exactly, assuming no noise.

With $\sigma^2 > 0$, we have uncertainty in our predictions at the training data points as well.



Overly large or small values of τ will over-smooth or under-smooth the data.



Learning the kernel parameters

Even for the simple case of an RBF kernel, we have two free parameters, kernel bandwidth τ and noise variance σ^2 . Choice of parameters matters a lot!

Typical approach: choose parameter values θ, σ^2 to optimize the **marginal likelihood** of the training data:

$$\Pr(y_1, \dots, y_M \mid x_1, \dots, x_M, \theta, \sigma^2) = \int \Pr(y_1, \dots, y_M \mid f, \sigma^2) \Pr(f \mid x_1, \dots, x_M, \theta) df$$

Good news: in the case of GP regression, we can obtain a closed form expression for the marginal likelihood.

$$Y \mid X, \theta, \sigma^2 \sim N(0, K_\theta + \sigma^2 I)$$

$$\ln \Pr(Y \mid X, \theta, \sigma^2) = -\frac{1}{2} \ln \det(K_\theta + \sigma^2 I) - \frac{1}{2} \hat{y}^T (K_\theta + \sigma^2 I)^{-1} \hat{y} + C$$

This can be directly optimized as a function of θ and σ^2 , or estimated using Bayesian sampling methods.

Classification with GPs

Given a set of M training examples, $(X, Y) = \{(x_i, y_i)\}$ for $i=1..M$, and a set of M^* test examples, $(X^*, Y^*) = \{(x_i^*, y_i^*)\}$ for $i=1..M^*$, where all $y_i = +/- 1$.

$$\text{Assume } \Pr(y_i = 1 \mid x_i, f) = \frac{\exp(f(x_i))}{1 + \exp(f(x_i))},$$

where $f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))$, and $k(\cdot, \cdot)$ is the kernel (e.g., RBF).

Can compute **class probabilities** for a given test data record as follows:

$$\Pr(y_i^* = 1 \mid x_i^*, X, Y, \theta) = \int \Pr(y_i^* = 1 \mid x_i^*, f) \Pr(f \mid X, Y, \theta) df$$

Classification with GPs is **harder** than regression!

For regression: GP prior + Gaussian likelihood = GP posterior.

Everything remains analytically tractable, e.g., closed form solutions for our predictions and the marginal likelihood of the data.

For classification: we have a non-Gaussian likelihood and have to do approximate rather than exact inference.

Options: Approximate non-Gaussian likelihood with Gaussian (aka Laplace approximation), or do Bayesian sampling (such as expectation propagation). See R&W, Ch. 3, for details.

Where are we now?

Gaussian processes are a useful way to **model** and **predict** with dependent data (e.g., spatial and temporal).

Can obtain **prediction intervals** rather than just point estimates, and can **learn** dependence structure from data.

Recent GP methods address computational challenges, both for performing GP regression in faster than $O(N^3)$ time without losing accuracy, and better and more efficient approximations for GP classification.

Additionally, we can learn flexible, interpretable kernels such as **spectral mixture kernels** (Wilson & Adams, 2013). These provide rich information about correlation structure, such as trends and cyclic patterns on multiple scales.

GPs have been extended to many other machine learning tasks, such as change point detection and causal inference.

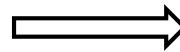
Example application 1

(from Flaxman et al., 2015a)

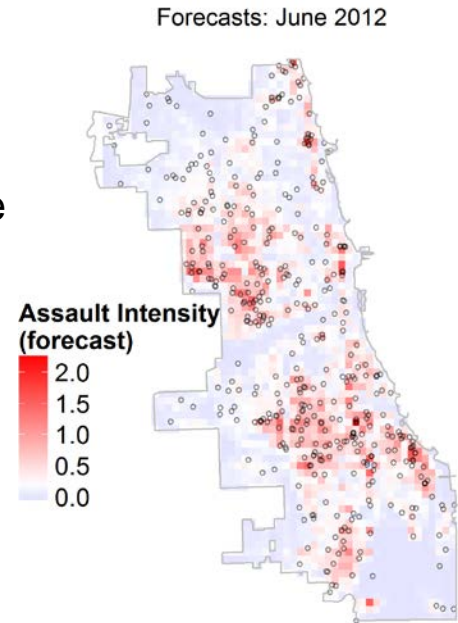
Long-term, local-area **crime forecasting** in Chicago. We applied a new, scalable GP model to 10 years of geocoded, date-stamped crime reports.



New methods for very fast inference and learning with non-Gaussian likelihoods (count data) and interpretable (spectral mixture) kernels.



Approximations plus exploiting grid structure for fast matrix operations.



$n = 233,088$ reported incidents of assault.

After discretization ($\frac{1}{2}$ mile \times $\frac{1}{2}$ mile \times week):
1.6 million observations in total, much too large for standard GP formulations.

Very accurate, small-area crime forecasts
up to 12 months in advance: scalability and accuracy higher than previous state of the art.

Example application 1

(from Flaxman et al., 2015a)

Long-term, local-area **crime forecasting** in Chicago. We applied a new, scalable GP model to 10 years of geocoded, date-stamped crime reports.

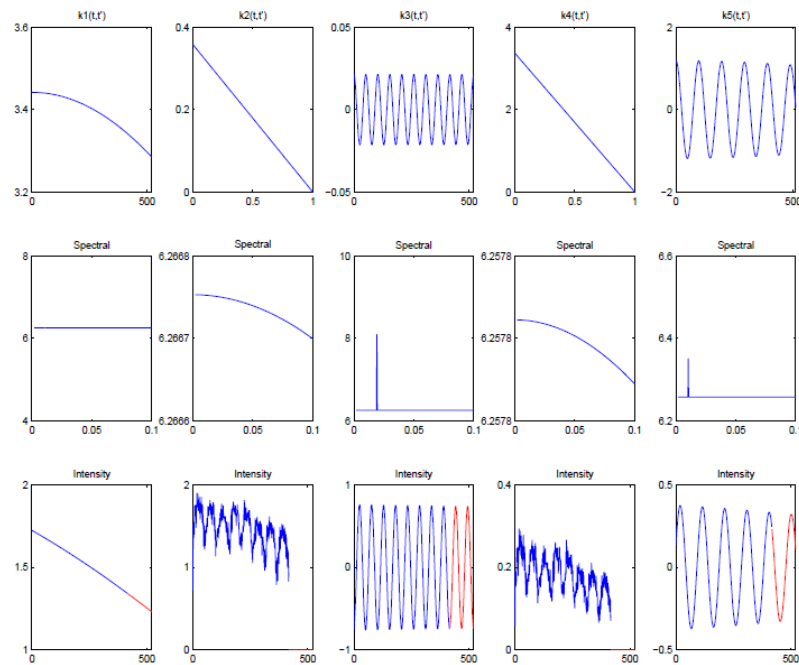


Figure 4. The five spectral mixture components with highest weights learned by our model are shown as a covariance (top) and spectral density (middle). In the bottom row, time series predictions were made on the dataset (ignoring space) using only that component. Red indicates out-of-sample forecasts.

Learning an interpretable kernel tells us a lot about the correlation structure in space and time.

Component 1 picks up long-term trend of decreasing crime.

Component 3 picks up yearly periodic (seasonal) trend.

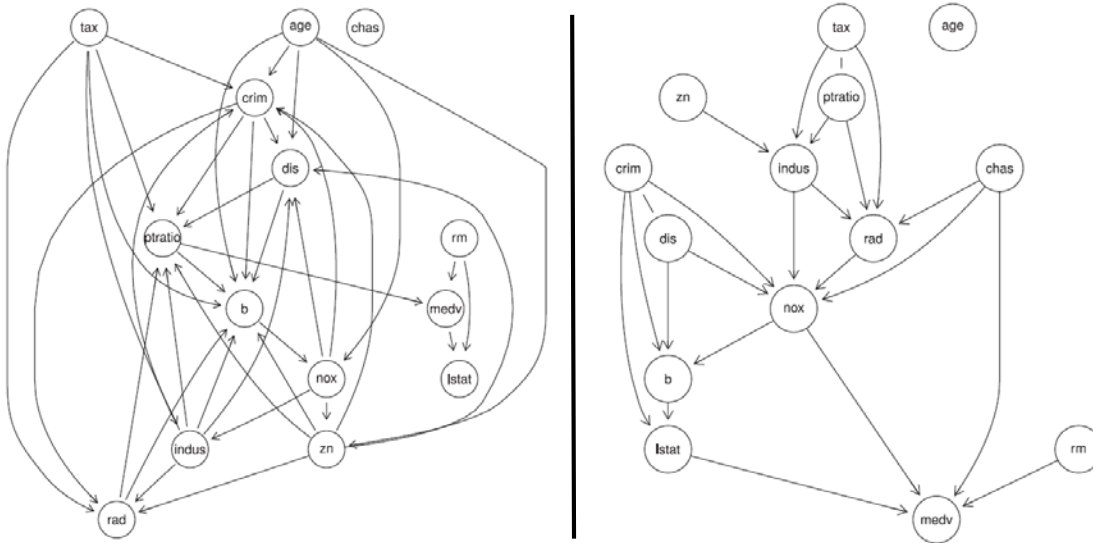
Components 2 and 4 represent correlation on short time-scales (no effect on long-term forecasts).

Component 5 picks up another, more subtle periodic trend.

Example application 2

(from Flaxman et al., 2015b)

Causal inference in non-iid data, using pre-whitening with GPs to remove non-causal relationships resulting from spatial and temporal dependencies.



Causal inference on Boston housing data.

Left: result of PC algorithm.

Right: PC after using our GP inference method.

Note many fewer edges than the previous graph.

Causes of median house value (medv):
Percent of lower SES in the population (lstat), number of rooms (rm), whether located on the Charles River (chas), and pollution as measured by nitric oxide concentration (nox).

Also note the edge from industrial activity (indus) to pollution (nox), while the previous graph had this edge reversed.

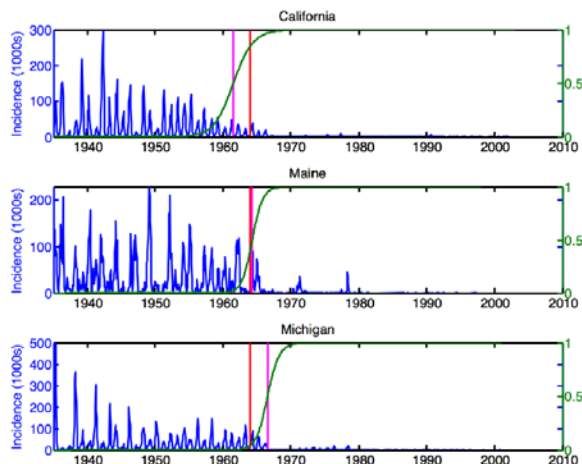
Example application 3

(from Herlands et al., 2016)

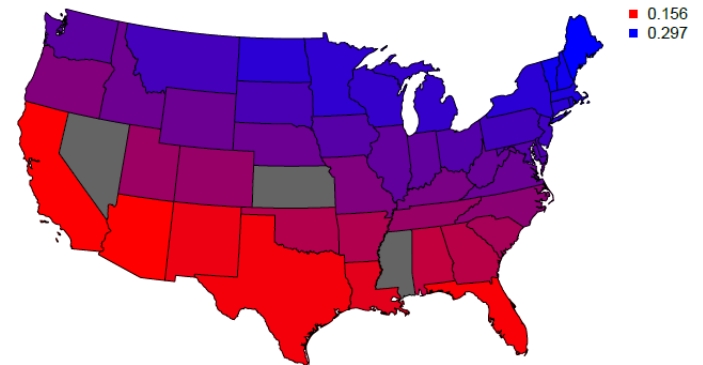
We developed new, scalable GP methods to detect **multidimensional**, **gradual**, and **heterogeneous** changes in the data distribution (“change surface detection”).

We used our approach to model state-level, monthly measles incidence data from 1935 to 2003 (~33K data points in 3 dimensions: long, lat, time).

As expected, we identified a significant change from the introduction of the measles vaccine in 1963. But the impacts of the vaccine were shown to be gradual and heterogeneous: different states had different change points and rates of change from pre-vaccine to post-vaccine.



Change points: mid-1961 to early 1967.



Rate of change by state varied by 2x (Maine fastest, Arizona slowest).

References

- GPs in Python with sklearn: http://scikit-learn.org/stable/modules/gaussian_process.html
- C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. <http://www.gaussianprocess.org/gpml/chapters/>
- GP website: <http://www.gaussianprocess.org/>
- GPML Matlab code: <http://www.gaussianprocess.org/gpml/code/matlab/doc/>
- State of the art, scalable GP methods (GPyTorch), code by Andrew Gordon Wilson: <https://cims.nyu.edu/~andrewgw/code/>
- Spectral mixture kernels: A.G. Wilson and R.P. Adams. Gaussian process kernels for pattern discovery and extrapolation. *Proc. ICML*, 2013.
- S.R. Flaxman, A.G. Wilson, D.B. Neill, H. Nickisch, and A.J. Smola. Fast Kronecker inference in Gaussian processes with non-Gaussian likelihoods. *Proc. ICML*, 2015a. <http://www.cs.nyu.edu/~neill/papers/icml15.pdf>
- S.R. Flaxman, D.B. Neill, and A.J. Smola. Gaussian processes for independence tests with non-iid data in causal inference. *ACM TIST*, 2015b. <http://www.cs.nyu.edu/~neill/papers/TIST2015.pdf>
- W. Herlands, et al. Scalable Gaussian processes for characterizing multidimensional change surfaces. *Proc. AISTATS*, 2016. <http://www.cs.nyu.edu/~neill/papers/aistats16.pdf>