Introduction to Gaussian Process

CS 778

Chris Tensmeyer
What Topic?

- Machine Learning
- Regression
- Bayesian ML
- Bayesian Regression
- Bayesian Non-parametric
- Gaussian Process (GP)
- GP – Regression
- GP – Regression with Squared Exponential Kernel
Regression

- Predict real value as output
- Learn underlying target function
- Some data samples of target function (with noise)
- E.g. Predict Credit Score
- Estimate model parameters from data points.
  - $y = w \cdot x$, estimate $w$, to minimize loss
Paper Titles

• Gaussian process classification for segmenting and annotating sequences
• Discovering hidden features with Gaussian processes regression
• Active learning with Gaussian processes for object categorization
• Semi-supervised learning via Gaussian processes
• Discriminative Gaussian process latent variable model for classification
• Computation with infinite neural networks
• Fast sparse Gaussian process methods: The informative vector machine
• Approximate Dynamic Programming with Gaussian Processes
Paper Titles

• *Learning to control an octopus arm* with Gaussian process temporal difference measure

• Gaussian Processes and *Reinforcement Learning* for Identification and Control of an Autonomous Blimp

• Gaussian process latent variable models for *visualization of high dimensional data*

• Bayesian *unsupervised signal classification* by Dirichlet process mixtures of Gaussian processes
Bayesian ML

- Probability Theory
  - What makes function a Prob. Dist?

- Estimate \textit{distribution} over model parameters, $\theta$, that generated $D$
  - AKA function or hypothesis

- Important distributions
  - Prior: $p(\theta)$
  - Data Likelihood: $p(D|\theta)$
  - Posterior: $p(\theta|D) = \frac{1}{p(D)} \times p(D|\theta) \times p(\theta)$
  - Posterior Predictive: $p(y|x) = \int p(y|x, \theta)p(\theta|D)d\theta$
    - Let all possible functions (one per $\theta$) vote for the correct value
Running Example

• Predict weight given height
  • Lbs & inches
• Let’s use a linear model with Gaussian Noise
  • 2 parameters: $m, \sigma^2$
  • Let’s assume that $\sigma^2 = 5$
• $Weight \sim N(m \ast height, \sigma^2)$
• $p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$
• Let’s use a prior for $m$
Prior

GENERAL

• \( p(\theta) \)

• Your own bias about what makes a good solution

• Score solutions without seeing data

• Encoded as a distribution over parameters

EXAMPLE

• Let’s get 5 good guesses about what \( m \) is
  • Units are lbs/inch
  • \( p(m = \text{guess}_i) = 0.2 \)

• Other possible \( m \) get 0 prob. under our prior
  • Normally use a prior which assigns everything a non-zero probability.
Data Likelihood

GENERAL

• $p(D|\theta)$

• $\theta$ describes the process of creating $D$

• Assuming $\theta$ reflects reality, out of all the $D'$, how likely is the $D$ we got?

• Maximum Likelihood Estimation

EXAMPLE

• Data: Height -> Weight ($h_i \rightarrow w_i$)
  • 70in -> 150lbs
  • 80in -> 200lbs
  • 60in -> 130lbs

• $p(D|m, \sigma^2) = \prod_{i=1}^{|D|} N(w_i; h_i \ast m, \sigma^2)$

• The $m$ that maximizes $^\wedge$ is the MLE solution.
  • Is it one of our 5 guesses?
  • No. What is it?
  • $m = \frac{686}{298} = 2.302$
Posterior

GENERAL

• \( p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \)

• Bayes Theorem

• Combines Prior with Data Likelihood

• Often find \( \theta \) that maximizes \( p(\theta|D) \)
  • Maximum a posteriori (MAP) estimate
  • Can ignore \( p(D) \) in this case

EXAMPLE

• Plug in each of our guesses for \( m \) into the numerator

• Get denominator by \( \sum p(m = guess_i|D) = 1 \)

• Guess with highest probability is the MAP estimate
Posterior Predictive

**GENERAL**

- \( p(y|x) = \int p(y|x, \theta)p(\theta|D)d\theta \)
- Let all \( \theta \) vote, weighted by posterior probability
  - Bayes Optimal Ensemble
- Rarely tractable integral
  - MLE or MAP is easier
- Really want: \( \text{argmax}_y p(y|x) \)
  - Can also do confidence intervals

**EXAMPLE**

- Given a test point height = 75in, predict weight
  - Using MLE?
  - Using MAP?
- Posterior Predictive
  - Guess a weight and score it
  - Score weight using each of 5 \( m \)
  - Weight scores by \( p(m) \)
  - Iterate until \( \text{argmax}_w p(w|h) \)
Other Regression Methods

• Simple Linear Regression (LR)
  • \( y = w \cdot x \) or in Bayesian Land: \( y \sim N(w \cdot x, \sigma^2) \)
  • Ordinary Least Squares: find \( w \) to minimize \( SSE = \sum(w \cdot x - y_t)^2 \)
    • Equivalent to maximizing \( p(D) = \prod p(y_t|x, w) = \prod N(y_t|w \cdot x; \sigma^2) \)
    • Closed form solution: \( w = (X^T X)^{-1} X^T y \)
• Kernelized Linear Regression (KLR)
  • In LR, replace \( x \) with \( \varphi(x) \), where \( \varphi \) is a non-linear feature transform
  • Kernel trick allows you to do implicit \( \varphi(x) \) for efficiency
  • E.g. Learn coefficients of order N polynomial
Other Regression Methods

• Ridge Regression (RR)
  • Like LR, but with L2 weight penalty to encourage small weights
  • More robust to outliers (less overfit)
  • L2 weight penalty = Gaussian Prior on weights
  \[ w_i \sim N(0, \sigma^2) \Rightarrow p(w) \propto \prod e^{\frac{-w_i^2}{\sigma^2}} \Rightarrow -\log(p(w)) \propto \sum w_i^2 \]
  • Find \( w \) to minimize \( \text{Obj}(w) = \sum (w \cdot x - y_t)^2 + \lambda \sum w_i^2 \)

• Lasso Regression (Lasso)
  • LR with L1 penalty
  • Laplacian Distribution Prior: \( p(w_i) \propto e^{-|w_i|} \)
  • Fewer non-zero weights (sparse)
Other Regression Methods

- **Bayesian Regression (BR)**
  - Like RR, but estimate $\lambda$ from data
  - Prior over the Prior
  - $w_i \sim N(0, \sigma^2)$ but $\sigma^2 \sim \Gamma(\alpha, \beta)$
    - $\alpha, \beta$ are user set

- **Automatic Relevance Determination (ARD)**
  - Like BR, but $w_i \sim N(0, \sigma_i)$ and $\sigma_i \sim \Gamma(\alpha, \beta)$
  - Each weight comes from different prior distribution
  - Good for feature selection
Gaussian Process

- GP = Stochastic Process where Random Variables are Gaussian Distributed

Distribution of f(1)

FIGURE 18.2 Left: 10 samples from the stochastic process $f(x) = \exp(ax) \cos(bx)$ with $a$ and $b$ drawn from Gaussian distributions. Right: The probability distribution of $f(1)$ based on 10,000 samples of $f(x)$.  

Marsland Book
Gaussian Process - MVN

- Generalization of Multi-Variate Normal to $\infty$ dimensions

$$f_x(x_1, \ldots, x_k) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu)\right),$$
Gaussian Process - Background

• First theorized in 1940s
• Used in geostatistic and meteorology in 1970s for time series prediction
  • Under the name *kriging*
**Smooth Functions**

- How would you measure the smoothness of a function?
  - “Smooth” functions is a common prior or bias

- Smoothness: function values close in input space should be similar
  - How similar is $f(1)$ to $f(1.1)$?
  - Compared to $f(1)$ to $f(10)$?

- For any finite # of points, $\{x_i\}$, can fit a MVN to score function smoothness based on training set composed of smooth functions to learn $(\mu, \Sigma)$
  - $\text{Smoothness}(f) := p_{\text{MVN}}(f(x_0), \ldots f(x_i)|\mu, \Sigma)$
Dist. Over Functions

• Functions == infinite lookup table == infinite vector of values
  • Vector entries are value of function at some input

• Problem with our MVN smoothness at finite # of points
  • Functions score well if not smooth between those points

• GP solves this by being non-parametric
  • $\Sigma$ replaced by $k(x, x')$
  • $\mu$ replaced by $m(x)$

• GP is $\infty$ dimensions, but any finite subset of dimension is MVN

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$
Covariance Function

\[ m(x) = \mathbb{E}[f(x)], \]
\[ k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))], \]

- \( k(x_i, x_j) \) yields \( \Sigma \) for MVN over any finite subset \( \{f(x_i)\} \)
- \( \Sigma_{ij} = k(x_i, x_j) \)
- \( \forall \{x_i\}, k(x_i, x_j) \) must make \( \Sigma \) Positive Definite
  - Otherwise MVN is not defined
- Common to subtract out the mean function from train/test points
- Less common to learn mean function from data

Equations from Rasmussen & Williams Book
Covariance Functions

- **Constant**: $K_C(x, x') = C$
- **Linear**: $K_L(x, x') = x^T x'$
- **Gaussian Noise**: $K_{GN}(x, x') = \sigma^2 \delta_{x,x'}$
- **Squared Exponential**: $K_{SE}(x, x') = \exp \left( -\frac{||d||^2}{2l^2} \right)$
- **Ornstein–Uhlenbeck**: $K_{OU}(x, x') = \exp \left( -\frac{||d||}{l} \right)$
- **Matérn**: $K_{Matern}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu||d||}}{l} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu||d||}}{l} \right)$
- **Periodic**: $K_P(x, x') = \exp \left( -\frac{2\sin^2(\frac{d}{2})}{l^2} \right)$
- **Rational Quadratic**: $K_{RQ}(x, x') = (1 + ||d||^2)^{-\alpha}, \quad \alpha \geq 0$

Equations from Wikipedia: Gaussian Process
Covariance Functions

• Learning in GP is finding hyperparameters that fit data well!
  • Many design choices: kernel, fixed, global, local, per-dimension
  • Math is messy
  • Do SGD on hyperparameters to maximize log probability of training data

• Observation Noise
  • Universal hyperparameter $\sigma^2$ is our estimate of how bad we measured the training targets
  • Similar to Bayesian LR
  • Helps prevent overfit
Hyperparameters
Length-Scale

(a), $\ell = 1$

(b), $\ell = 0.3$

(c), $\ell = 3$
Inference/Prediction

• The joint distribution of training targets and test target is MVN

\[ P(t^* | t_N) = \frac{P(t^*, t_N)}{P(t_N)}. \]

• We condition the MVN on the observed training targets

• Resulting conditional distribution is Gaussian

• Use distribution mean/covariance for prediction

\[ P(t^* | t, x, x^*) \propto \mathcal{N} \left( k^T K^{-1} t, k^{**} - k^T K^{-1} k^* \right), \]

Equations from Marsland Book
Conditional distributions

If N-dimensional $\mathbf{x}$ is partitioned as follows

$$
\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}
$$

with sizes

$$
\begin{bmatrix} q \\ (N - q) \end{bmatrix} \times 1
$$

and accordingly $\mathbf{\mu}$ and $\mathbf{\Sigma}$ are partitioned as follows

$$
\mathbf{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}
$$

with sizes

$$
\begin{bmatrix} q \\ (N - q) \end{bmatrix} \times 1
$$

$$
\mathbf{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}
$$

with sizes

$$
\begin{bmatrix} q \\ (N - q) \end{bmatrix} \times q
$$

and $\begin{bmatrix} q \\ (N - q) \end{bmatrix} \times (N - q)$

then, the distribution of $\mathbf{x}_1$ conditional on $\mathbf{x}_2 = \mathbf{a}$ is multivariate normal $(\mathbf{x}_1 | \mathbf{x}_2 = \mathbf{a}) \sim N(\overline{\mathbf{\mu}}, \overline{\mathbf{\Sigma}})$ where

$$
\overline{\mathbf{\mu}} = \mu_1 + \Sigma_{12} \Sigma_{22}^{-1} (\mathbf{a} - \mu_2)
$$

and covariance matrix

$$
\overline{\mathbf{\Sigma}} = \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}. \quad \text{[12]}
$$
## Discussion

<table>
<thead>
<tr>
<th><strong>PROS</strong></th>
<th><strong>CONS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Learned function is smooth but not constrained by form</td>
<td>• Math/Optimization is messy</td>
</tr>
<tr>
<td>• Confidence intervals</td>
<td>• Use lots of approximations</td>
</tr>
<tr>
<td>• Good prior for other models</td>
<td>• NxN Matrix inversion</td>
</tr>
<tr>
<td>• E.g. Bias N-degree polynomials to be smooth</td>
<td>• Numerical Stability issues</td>
</tr>
<tr>
<td>• Can do multiple test points that co-vary appropriately</td>
<td>• Scalability in # instances issues like SVMs</td>
</tr>
<tr>
<td></td>
<td>• High dimensional data is challenging</td>
</tr>
</tbody>
</table>
Implementation

- For given training data \((X, t)\), test data \(x^*\), covariance function \(k()\), and hyperparameters \(\theta = (\sigma_j^2, \sigma_n^2)\):
  - compute the covariance matrix \(K = k(X, X) + \sigma_n I\) for hyperparameters \(\theta\)
  - compute the covariance matrix \(k^* = k(X, x^*)\)
  - compute the covariance matrix \(k^{**} = k(x^*, x^*)\)
  - the mean of the process is \(k^*^T K^{-1} t\)
  - the covariance is \(k^{**} - k^*^T K^{-1} k^*\)

- Only compute \(K^{-1}\) once using numerical stable methods
- Cholesky decomposition: \(K = LL^T\), where \(L\) is lower triangular
- For each test instance, compute covariance vector \(k^*\)
- \(\sigma_n I\) is our observation noise added to raw \(k(X,X)\)
Example

- Using a squared exponential kernel

\[ k(x, x') = \exp(-0.5 \times \|x - x'\|^2) \]

- Data to the right

- Task is to predict \( f(2) \) and \( f(4) \)
  - \( \sigma_n = 0.25 \)

<table>
<thead>
<tr>
<th>X</th>
<th>f(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
</tr>
</tbody>
</table>
Example – Covariance Matrix

- Begin by calculating covariance matrix
- Use \( f(0), f(1), f(3) \) as dimensions of the MVN

\[
\begin{align*}
  k(0,1) &= \exp(-0.5 \ (1 - 0)^2) = e^{-0.5} \\
  k(0,3) &= \exp(-0.5 \ (3 - 0)^2) = e^{-4.5} \\
  k(1,3) &= \exp(-0.5 \ (3 - 1)^2) = e^{-2} \\
  k(x, x) &= \exp(-0.5 \ (x - x)^2) = 1 \\
  \end{align*}
\]

\[
K = \begin{bmatrix}
  1 & e^{-0.5} & e^{-4.5} \\
  e^{-0.5} & 1 & e^{-2} \\
  e^{-4.5} & e^{-2} & 1 \\
\end{bmatrix} + \sigma_n I
\]
Example – Matrix Inversion

\[ K = \begin{bmatrix} 1 & e^{-0.5} & e^{-4.5} \\ e^{-0.5} & 1 & e^{-2} + \sigma_n I \\ e^{-4.5} & e^{-2} & 1 \end{bmatrix} \quad \text{or} \quad K = \begin{bmatrix} 1.25 & 0.607 & 0.011 \\ 0.607 & 1.25 & 0.135 \\ 0.011 & 0.135 & 1.25 \end{bmatrix} \]

Next step, invert \( K \)

\[ K^{-1} = \begin{bmatrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{bmatrix} \]

\[
\begin{array}{lcc}
X & f(X) \\
0 & 0 & -5 \\
1 & 1 & 0 \\
2 & 2 & ? \\
3 & 3 & 5 \\
4 & 4 & ? \\
\end{array}
\]
Example – Multiply by Targets

\[ P(t^*|t, x, x^*) \propto \mathcal{N}(k^T K^{-1} t, k^{**} - k^T K^{-1} k^*) \]

Need \( K^{-1} t \)

\[
K^{-1} = \begin{bmatrix}
1.049 & -0.514 & 0.046 \\
-0.514 & 1.061 & -0.11 \\
0.046 & -0.11 & 0.812
\end{bmatrix}
\]

\[
t = \begin{bmatrix}
-5 \\
0 \\
5
\end{bmatrix}
\]

\[
K^{-1} t = \begin{bmatrix}
-5.013 \\
2.018 \\
3.826
\end{bmatrix}
\]

\[
\begin{array}{cc}
X & f(X) \\
0 & -5 \\
1 & 0 \\
2 & ? \\
3 & 5 \\
4 & ?
\end{array}
\]
Example – Predicting f(2)

\[
P(t^*|t, x, x^*) \propto \mathcal{N}(k^T K^{-1} t, k^{**} - k^T K^{-1} k^*)
\]

Now need \( k^* \) for \( f(2) \)

\[
K^{-1} t = \begin{pmatrix} -5.013 \\ 2.018 \\ 3.826 \end{pmatrix}
\]

- \( k(2,0) = \exp(-0.5 \cdot (2 - 0)^2) = e^{-2} \)
- \( k(2,1) = k(2,3) = e^{-0.5} \)

\[
k^* = \begin{pmatrix} 0.135 \\ 0.607 \\ 0.607 \end{pmatrix} \quad k^{**} = 1 \text{ (no obs noise)}
\]

<table>
<thead>
<tr>
<th>X</th>
<th>f(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
</tr>
</tbody>
</table>
Example – Predicting $f(2)$

$$P(t^* | t, x, x^*) \propto N (k^*^T K^{-1} t, k^{**} - k^*^T K^{-1} k^*)$$

\[
\begin{align*}
    k^* &= 0.607 \\
    k^* &= 0.607 \\
    k^* &= 0.607 \\
    k^* &= 0.607
\end{align*}
\]

\[
\begin{align*}
    K^{-1} &= \begin{pmatrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{pmatrix} \\
    K^{-1} t &= \begin{pmatrix} -5.013 \\ 2.018 \\ 3.826 \end{pmatrix} \\
    \mu &= k^*^T K^{-1} t = 2.866 \\
    k^*^T K^{-1} k^* &= 0.507 \\
    k^* &= 0.550 \\
    0.432
\end{align*}
\]

\[
\begin{align*}
    \sigma^2 &= k^{**} - k^*^T K^{-1} k^* = 1 - 0.55 = 0.45
\end{align*}
\]
**Example – Predicting \( f(4) \)**

\[
P(t^*|t, x, x^*) \propto \mathcal{N}(k^*^T K^{-1} t, k^{**} - k^*^T K^{-1} k^*)
\]

\[
k^* = \begin{bmatrix} 0.011 \\ .607 \end{bmatrix}
\]

\[
K^{-1} = \begin{bmatrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{bmatrix}
\]

\[
K^{-1} t = \begin{bmatrix} -5.013 \\ 2.018 \\ 3.826 \end{bmatrix}
\]

\[
\mu = k^*^T K^{-1} t = 2.341
\]

\[
k^*^T K^{-1} k^* = -0.055 \quad k^* = 0.297
\]

\[
\sigma^2 = k^{**} - k^*^T K^{-1} k^* = 1 - 0.297 = 0.703
\]
Example - Plot

<table>
<thead>
<tr>
<th>X</th>
<th>f(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2.8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2.3</td>
</tr>
</tbody>
</table>
My Experiments

• 1-D synthetic datasets generated with Gaussian noise
  • Linear       Cubic
  • Periodic    Sigmoid   Complex

• 40 instances for training on the interval [-5,5]
  • Additive Gaussian noise with $\sigma = 0.5$

• Test with 1000 evenly spaced instances without noise (MSE)

• Use 3-fold CV on training data to set hyperparameters with grid-search

• Compare with Kernelized Lasso regression and 7-NN with inverse distance weighting
  • Polynomial and Sine function basis
Results - Linear

\[ y = 0.5x - 1 \]

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.009</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.019</td>
</tr>
<tr>
<td>KNN</td>
<td>0.058</td>
</tr>
</tbody>
</table>
Results - Cubic  \( y = 0.02x^3 - 0.1x^2 + 0.5x + 2 \)

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.046</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.037</td>
</tr>
<tr>
<td>KNN</td>
<td>0.101</td>
</tr>
</tbody>
</table>
Results - Periodic  \( y = 2 \sin(1.75x - 1) + 1 \)

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.087</td>
</tr>
<tr>
<td>Lasso</td>
<td>1.444</td>
</tr>
<tr>
<td>KNN</td>
<td>0.350</td>
</tr>
</tbody>
</table>
Results - Sigmoid

\[ y = \frac{4}{1.5 + e^{-(.75x-1)}} + 1 \]

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.034</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.045</td>
</tr>
<tr>
<td>KNN</td>
<td>0.060</td>
</tr>
</tbody>
</table>

**Graph:**
- **Ground Truth**
- **Training Data**
- **GP**
- **Lasso**
- **KNN**
Results - Complex  

\[ y = \text{linear combination of previous functions} \]

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.194</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.574</td>
</tr>
<tr>
<td>KNN</td>
<td>0.327</td>
</tr>
</tbody>
</table>
Real Dataset - Concrete

- **Concrete Slump Test**
  - How fast does concrete flow given its materials?
  - 103 instances; 7 input features; 3 outputs; all real values

- Input Features are kg per cubic meter of concrete
  - Cement, Slag, Fly Ash, Water, SP, Coarse Agg, Fine Agg

- Output Variables are
  - Slump, Flow, 28-day Compressive Strength

- Ran 10-fold CV with our three models on each output variable
## Concrete Results

<table>
<thead>
<tr>
<th>Slump</th>
<th></th>
<th>Flow</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td></td>
<td><strong>Model</strong></td>
<td></td>
</tr>
<tr>
<td>GP</td>
<td>0.103 +/- 0.037</td>
<td>GP</td>
<td>0.091 +/- 0.026</td>
</tr>
<tr>
<td>Lasso</td>
<td><strong>0.072 +/- 0.024</strong></td>
<td>Lasso</td>
<td><strong>0.051 +/- 0.020</strong></td>
</tr>
<tr>
<td>KNN</td>
<td><strong>0.067 +/- 0.040</strong></td>
<td>KNN</td>
<td><strong>0.063 +/- 0.036</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strength</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td></td>
</tr>
<tr>
<td>GP</td>
<td>0.030 +/- 0.015</td>
</tr>
<tr>
<td>Lasso</td>
<td><strong>0.004 +/- 0.002</strong></td>
</tr>
<tr>
<td>KNN</td>
<td>0.008 +/- 0.005</td>
</tr>
</tbody>
</table>
Real Dataset – NASA Airfoil Noise

- How loud does it get in the wind tunnel?
- 1503 instances; 5 input features; 1 output; all real values
- Input Features are wind tunnel conditions
  - Frequency (Hz), Wind Angle, Chord Length, Velocity, Suction Side Displacement (meters)
- Output Variable is “Scaled sound pressure level (decibels)”

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg MSE ± Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.0040 ± 0.0007</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.0165 ± 0.0023</td>
</tr>
<tr>
<td>KNN</td>
<td>0.0049 ± 0.0007</td>
</tr>
</tbody>
</table>
Real Dataset – Energy Efficiency

• What is the heating/cooling load for buildings?

• Simulated data

• 783 instances; 8 input features; 2 outputs;

• Input Features are wind tunnel conditions
  • Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Glazing Area (GA), GA distribution

• Output Variables are heating/cooling load

<table>
<thead>
<tr>
<th>NASA Airfoil Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
</tr>
<tr>
<td>GP</td>
</tr>
<tr>
<td>Lasso</td>
</tr>
<tr>
<td>KNN</td>
</tr>
</tbody>
</table>
Energy Efficiency Results

<table>
<thead>
<tr>
<th>Heating Load</th>
<th>Cooling Load</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td><strong>Avg MSE +- Std</strong></td>
</tr>
<tr>
<td>GP</td>
<td><strong>0.0012</strong> + 0.0002</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.0064 + 0.0015</td>
</tr>
<tr>
<td>KNN</td>
<td>0.0040 + 0.0010</td>
</tr>
</tbody>
</table>
Learning Hyperparameters

• Find hyperparameters to maximize (log) probability of data

$$\log P(t|x, \theta) = -\frac{1}{2} t^T (K + \sigma_n^2 I)^{-1} t - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{N}{2} \log 2\pi$$

• Can use Gradient Descent by computing for

$$Q = (K + \sigma_n^2 I)$$

$$\frac{\partial}{\partial \theta} \log P(t|x, \theta) = \frac{1}{2} t^T Q^{-1} \frac{\partial Q}{\partial \theta} Q^{-1} t - \frac{1}{2} \text{trace} \left( Q^{-1} \frac{\partial Q}{\partial \theta} \right)$$

• is just the elementwise derivative of the covariance matrix which is the derivative of the covariance function

Equations from Marsland Book
Learning Hyperparameters

- For squared exponential covariance function in this form

\[ k(x, x') = \exp(\sigma_f) \exp \left( -\frac{1}{2} \exp(\sigma_l)|x - x'|^2 \right) + \exp(\sigma_n)I \]

\[ = k' + \exp(\sigma_n)I \]

- We get the following derivatives

\[ \frac{\partial k}{\partial \sigma_f} = k' \]

\[ \frac{\partial k}{\partial \sigma_l} = k' \times \left( -\frac{1}{2} \exp(\sigma_l)|x - x'|^2 \right) \]

\[ \frac{\partial k}{\partial \sigma_n} = \exp(\sigma_n)I \]

Equations from Marsland Book
Covariance Function Tradeoffs

\[ k(r) = \exp \left( - \left( \frac{r}{\ell} \right)^\gamma \right) \text{ for } 0 < \gamma \leq 2 \]

\[ r = |x - x'| \]
Classification

- Many regression methods become binary classification by passing output through a sigmoid, Softmax, or other threshold
  - MLP, SVM, Logistic Regression
  - Same with GP, but scarier math

Figures from Rasmussen & Williams Book
Inference is naturally divided into two steps: first computing the distribution of the latent variable corresponding to a test case

$$p(f_* | X, y, x_*) = \int p(f_* | X, x_*, f)p(f | X, y) \, df$$

GP Regression

Intractable!

where

$$p(f | X, y) = p(y | f)p(f | X)/p(y | X)$$

$$p(y_i | f_i) = \sigma(f_i)$$

Vote for class 1

Weight for vote

$$p(y_* = +1 | X, y, x_*) = \int \sigma(f_*)p(f_* | X, y, x_*) \, df_*$$

Equations from Rasmussen & Williams Book
Bells and Whistles

- Multiple Regression – Assume correlation between multiple outputs
- Correlated noise – especially useful for time
- Training set values + derivatives (max or min occurs at...)
- Noise in the x-direction
- Mixture (ensemble) of local GP experts
- Integral approximation with GP
- Covariance over structured inputs (strings, trees, etc)
Bibliography

• Wikipedia: Gaussian Process, Multi-variate Normal, Gamma Distribution

• Scikit-Learn Documentation

• www.gaussianprocess.org
