A Gaussian Process View of Multiple Kernel Learning

Raquel Urtasun

TTI Chicago

December 11, 2010
Contents

- What are Gaussian processes?
- Simple MKL
- Heteroscedastic MKL
- Bayesian co-training
- Heteroscedastic Bayesian co-training
- Latent variable models
- Conclusion
The function view of Gaussian processes

**Definition**

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

- Probability Distribution over Functions
- Functions are infinite dimensional.
  - Prior distribution over *instantiations* of the function: finite dimensional objects.
- GPs are consistent.
A (zero mean) Gaussian process likelihood is of the form

\[ p(y|X) = N(y|0, K), \]

where \( K \) is the covariance function or \textit{kernel}.

Covariance samples

\[ \text{Figure: linear kernel, } K = XX^T \]
A (zero mean) Gaussian process likelihood is of the form

\[ p(y|X) = \mathcal{N}(y|0, K), \]

where \( K \) is the covariance function or \textit{kernel}.

Covariance samples

\[ k_{i,j} = \alpha \exp \left( -\frac{1}{2l} \|x_i - x_j\|^2 \right), \text{ with } l = 0.32, \alpha = 1 \]

\textbf{Figure}: RBF kernel, \( k_{i,j} = \alpha \exp \left( -\frac{1}{2l} \|x_i - x_j\|^2 \right), \text{ with } l = 0.32, \alpha = 1 \)
A (zero mean) Gaussian process likelihood is of the form
\[ p(y|X) = N(y|0, K), \]
where \( K \) is the covariance function or *kernel*.

- Covariance samples

**Figure:** RBF kernel, \( k_{i,j} = \alpha \exp \left( -\frac{1}{2l} \| x_i - x_j \|^2 \right) \), with \( l = 1, \alpha = 1 \)
A (zero mean) Gaussian process likelihood is of the form

\[ p(y|X) = N(y|0, K), \]

where \( K \) is the covariance function or kernel.

Covariance samples

Figure: bias 'kernel', \( k_{i,j} = \alpha \), with \( \alpha = 1 \) and
Gaussian processes

- A (zero mean) Gaussian process likelihood is of the form

\[ p(y|X) = N(y|0, K), \]

where \( K \) is the covariance function or kernel.

- Covariance samples

Figure: summed combination of: RBF kernel, \( \alpha = 1, l = 0.3 \); bias kernel, \( \alpha = 1 \); and white noise kernel, \( \beta = 100 \)
Gaussian process regression

**Posterior Distribution over Functions**

- Gaussian processes are often used for regression.
- We are given known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a *posterior* distribution over functions.

![Graph showing examples of WiFi localization and C14 calibration curve.](image)
Gaussian process regression

Posterior Distribution over Functions

- Gaussian processes are often used for regression.
- We are given known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a posterior distribution over functions.

```
Figure: Examples include WiFi localization, C14 callibration curve.
```
Gaussian process regression

Posterior Distribution over Functions

- Gaussian processes are often used for regression.
- We are given known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a posterior distribution over functions.
Gaussian process regression

**Posterior Distribution over Functions**
- Gaussian processes are often used for regression.
- We are given known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a *posterior* distribution over functions.
Gaussian process regression

**Posterior Distribution over Functions**

- Gaussian processes are often used for regression.
- We are given known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a *posterior* distribution over functions.

![Graph showing examples including WiFi localization and C14 calibration curve.](image)
Gaussian process regression

Posterior Distribution over Functions

- Gaussian processes are often used for regression.
- We are given known inputs \( X \) and targets \( Y \).
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a posterior distribution over functions.

![Diagram](image.png)
Gaussian Process Regression

**Posterior Distribution over Functions**

- Gaussian processes are often used for regression.
- We are given known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a *posterior* distribution over functions.

![Example graph showing posterior distribution over functions](image)

Examples include WiFi localization, C14 calibration curve.
Gaussian process regression

Posterior Distribution over Functions

- Gaussian processes are often used for regression.
- We are given known inputs $X$ and targets $Y$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a *posterior* distribution over functions.

![Diagram of Gaussian process regression](image)

Examples include WiFi localization, C14 calibration curve.
Learning in GPs

- Optimize over the hyperparameters of the covariance function.
- Typically done by maximum likelihood, where the negative log likelihood is
  \[ \mathcal{L} = \frac{1}{2} \ln |K| + \frac{1}{2} \mathrm{tr}(K^{-1}YY^T) \]
- This requires solving a non-convex optimization problem
  \[ \hat{\theta} = \arg \min_{\theta} \mathcal{L} - \ln p(\theta) \]
  where \( \theta \) is the set of hyperparameters
- For fixed hyperparameters closed-form solution (invert the kernel matrix!)

\[ \mu(x_*) = k_{*,f}K^{-1}Y \]
\[ \sigma^2(x_*) = k_{*,*} - k_{*,f}K^{-1}k_{f,*} \]
Simple MKL

- Linear combination of kernels

\[ K = \sum_i \alpha_i K^{(i)} \]

with \( K_i \) the individual kernels

- The kernels are typically defined using different kernels or kernels defined over different input spaces, e.g., different features in object recognition

- Learning is simply done by maximum likelihood

\[ \{ \hat{\theta}, \hat{\alpha} \} = \arg \max_{\theta, \alpha} \mathcal{L} - \ln p(\theta, \alpha) \]

- The prior on \( \alpha \) can be a laplacian (i.e., \( \ell_1 \)), a Gaussian (i.e., \( \ell_2 \)) or a combination of both (i.e., elastic nets) [Kapoor et al. 09]
Results in object recognition

- Object recognition benchmark Caltech101
  - 101 object categories
  - Centered objects with one object per image

![Graph showing mean recognition rate per class vs. number of training examples per class for various methods on Caltech101 dataset.](image)

- Results as of end of 2008 [Kapoor et al. 09], later I’ll show updated results.
- State of the art results in activity recognition [Han et al. 09]
Advantages:
- Very simple optimization
- In multiclass scenarios, if the $\{\theta, \alpha\}$ are shared across classes, then we can learn all classifiers at once

Problems:
- $\alpha = [1, \cdots, 1]$ works almost as well as learning the weights
- Limited in their ability to learn from inputs corrupted by complex noise (e.g., occlusion), with incomplete data, and/or whose discriminative properties locally vary across the input space
The product of kernels can be expressed as

\[ K = K^{(1)} \odot \ldots \odot K^{(V)} \]

where \( \odot \) is the Hadamard (element-wise) product.

This makes the covariance matrix more sharp.

Learning is again done as

\[ \hat{\theta} = \arg\max_{\theta} \mathcal{L} - \ln p(\theta) \]
Global vs Local kernel combinations

- These measures are global, and might be too simplistic for real cases.
- Limited in their ability to learn from inputs corrupted by complex noise (e.g., occlusion), with incomplete data, and/or whose discriminative properties locally vary across the input space.
- Local combination approaches have been proposed within SVM-based learning frameworks [Lin et al '07][Gonen and Alpaydin ’08],[Yang et al '09]
Assume a GP prior \( p(f|X) \) with covariance [Christoudias et al. 09a]

\[
\tilde{K} = \sum_{\nu=1}^{V} K^{(\nu)},
\]

where \( K^{(\nu)} \) is defined by the product of two kernels

\[
K^{(\nu)} = K_{np} \circ K_{p}^{(\nu)}
\]

where \( \circ \) is the Hadamard (element-wise) product.
Non-Parametric Covariance Representation

- Full pair-wise weighting

\[
\tilde{K} = \sum_{v=1}^{V} \begin{bmatrix}
  k_{np}^{(v)}(1, 1) & \cdots & k_{np}^{(v)}(1, N) \\
  \vdots & \ddots & \vdots \\
  k_{np}^{(v)}(N, 1) & \cdots & k_{np}^{(v)}(N, N)
\end{bmatrix} \odot \begin{bmatrix}
  k_{p}^{(v)}(x_1, x_1) & \cdots & k_{p}^{(v)}(x_1, x_N) \\
  \vdots & \ddots & \vdots \\
  k_{p}^{(v)}(x_N, x_1) & \cdots & k_{p}^{(v)}(x_N, x_N)
\end{bmatrix}
\]

number of parameters = \(V \cdot \frac{N(N-1)}{2}\)
Assume a low-rank approximation

\[
\tilde{K} = \sum_{v=1}^{V} \left( \begin{bmatrix} G^{(v)^T} \\ \end{bmatrix} \begin{bmatrix} G^{(v)} \\ \end{bmatrix} \right) \odot K_{p}^{(v)}
\]

with \( G^{(v)} \in \mathbb{R}^{m \times N} \)

number of parameters = \( V \cdot m \cdot N \)
Per-Sample Kernel Combination

- Rank-1 approximation

\[
\tilde{K} = \sum_{v=1}^{V} \left( \begin{bmatrix} g_1(1) \\ \vdots \\ g_1(N) \end{bmatrix} \right) \left[ \begin{bmatrix} g_1(1) \\ \cdots \\ g_1(N) \end{bmatrix} \right] \odot K_p^{(v)}
\]

with \( g_1 \in \mathbb{R}^N \).

number of parameters = \( V \cdot N \)

- Indication of how noisy every view is for each datapoint
Component-Wise Kernel Combination

- Assume weighting varies smoothly across input

\[
\tilde{K} = \sum_{\nu=1}^{V} \left( \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} \begin{bmatrix} \alpha_1^{(\nu)} \\ \vdots \\ \alpha_P^{(\nu)} \end{bmatrix} \right) \left( \begin{bmatrix} \alpha_1^{(\nu)} & \cdots & \alpha_P^{(\nu)} \end{bmatrix} \begin{bmatrix} e_1^T \\ \vdots \\ e_N^T \end{bmatrix} \right) \odot K_p^{(\nu)}
\]

with \( e_{i,j} \in \{0, 1\} \) and \( \sum_j e_{i,j} = 1 \).

- Combinatorial problem to solve for \( e_i \)

- Pre-cluster to give component-wise weighting

  \[ \text{number of parameters} = V \cdot P \]
Relation to Conventional Multiple Kernel Learning

- Conventional MKL: Single weight per view

\[
\tilde{K} = \sum_{v=1}^{V} [(\alpha^{(v)})^2] \odot K^{(v)}_p
\]

number of parameters = \(V\)

- Localized MKL [Christoudias et al. 09b]

\[
\tilde{K} = \sum_{v=1}^{V} \left( \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} \begin{bmatrix} \alpha_1^{(v)} \\ \vdots \\ \alpha_P^{(v)} \end{bmatrix} \right) \left( \begin{bmatrix} \alpha^{(v)}_1 & \cdots & \alpha^{(v)}_P \end{bmatrix} \begin{bmatrix} e_1^T \\ \vdots \\ e_N^T \end{bmatrix} \right) \odot K^{(v)}_p
\]

number of parameters = \(V \cdot P\)
Learning is performed by minimizing the negative log posterior

\[ \mathcal{L} = \frac{1}{2} \ln |\bar{K}| + \frac{1}{2} \text{tr}(\bar{K}^{-1}YY^T) + \lambda \sum_i \sum_j \frac{1}{(\alpha_j(i))^2} \]

with respect to the parameters \( \bar{\alpha} = [\alpha^{(1)}, \ldots, \alpha^{(V)}] \).

Efficient learning by optimizing all classifiers at once

Inference is performed using GP mean prediction

\[ \bar{y}_* = \max_c \{ \bar{k}_*^T \bar{K}^{-1} Y^c \} \]

where \( \bar{k}_* \) is the kernel computed between the training and test inputs, and \( Y^c \) are 1-vs.-all binary training labels over class \( c \).
Audio-Visual Dataset

- View-disagreement dataset [Christoudias et al ’08]

- 15 subjects answering a set of yes/no with head gestures and speech
  - head nod and shake gestures
  - ‘yes’ and ‘no’ utterances

- Simulated view-disagreement using background head motion and audio babble noise
Different levels of view disagreement

(0%)

(70%)
Caltech-101 Dataset

- Object recognition benchmark
  - 101 object categories
  - Centered objects
  - One object per image

- Four feature kernels:
  - Geometric blur kernels with/without distortion term [Zhang et al ’06]
  - Pyramid Match Kernel (PMK) [Grauman and Darrell ’05] and Spatial-PMK [Lazebnik et al ’06] on dense SIFT features
Results: Caltech-101

- Comparison to baseline approaches

- Results as of 2009, probably outperformed by now ;)

Raquel Urtasun (TTIC)
Results: Caltech-101

- Performance is stable across cluster number
- Only outperforms in missing data or noise cases

![Graph showing classification accuracy against number of clusters and number of missing samples per class](image)

(number of clusters) (missing data)
Estimating the Full Covariance

- Involves large number of parameters and is prone to over-fitting
- Laplacian regularizer proposes another method for exploiting the locality assumption

\[ L_{lap} = \sum_{v=1}^{V} (g^{(v)})^T L g^{(v)} \]

- Alternatively can assume solution lies close to an initial estimate \( G_0 \)

\[ L_{frob} = \| G - G_0 \|_F^2 \]

- Minimize sum of regularizer and GP objective
Preliminary Results

- Audio-visual dataset with Frobenius regularizer

![Graph showing Correct Classification Rate (CCR) vs. number of labeled examples per class](image)

- 50% View Disagreement
- Audio
- Video
- Late Integration
- MKL Baseline
- Our Approach (P=3)
- Our Approach (P=N)
Bayesian co-training

- Probabilistic co-training algorithm using GPs [Yu et al. 07]
- Main idea: agreement prior used to regularize solution
- **Transductive** setting
- GP prior assumed over each view

\[ f_c = \text{weighted average of } f_1, \cdots, f_V \]

\[
\mu_c = \sigma_c^2 \sum_j \frac{f_j}{\sigma_j^2} \\
\sigma_c^2 = \sum_j \left( \frac{1}{\sigma_j^2} \right)^{-1}
\]

- Marginalizing \( f_i \), we get a GP prior over \( f_c \) that favors view agreement

\[
p(f_c) = \mathcal{N}(0, K_c), \quad K_c = \left[ \sum_j (K_j + \sigma_j^2 I)^{-1} \right]^{-1}
\]
A generalization of [Yu et al. ‘07] to model sample-dependent noise was developed by [Christoudias et al. ‘09b]

Noise is modeled using a full covariance noise matrix $\mathbf{A}_j$

Need to parameterize $\mathbf{A}_j$
Heteroscedastic Noise Model

- Assume simple noise model for each $A_j$
  
  $$A_j = \text{diag}(\sigma^2_{1,j}, \cdots, \sigma^2_{N,j})$$

- To make learning and inference tractable, use a quantized noise model
  
  $$A_j = \text{diag}(E^{(j)} \cdot \phi_j)$$

  with noise variances $\phi_j = [\sigma^2_{1,j}, \cdots, \sigma^2_{P,j}] \in \mathbb{R}^{P \times 1}$, and indicator matrices $E^{(j)} = [e_1^{(j)}, \cdots, e_N^{(j)}]^T$, $e_i^{(j)} \in \{0, 1\}^{P \times 1}$.

- Model assumption: $E_j$ known on labeled data
Model Learning and Inference

- Learning is a two-step process:
  1. Learn kernel hyper-parameters noise variances, $\Phi = \phi_1, \cdots, \phi_V$ using $n$-fold cross-validation on the labeled data
  2. Estimate indicator matrices $E^{(j)}$ on the unlabeled data using Nearest Neighbors (NN) in each view independently

- Inference: $\bar{y}_* = k_c(X_*)^T(\hat{K}_c + \sigma^2 I_N)^{-1}y$

$\hat{K}_c$ is the sub-matrix of $K_c$ defined over the training examples
Latent variable models

- Unsupervised scenario, i.e., the inputs $X$ are unknown. Learning by

$$\{\hat{\theta}, \hat{X}\} = \arg\min_{\theta,X} \frac{1}{2} \ln |K| + \frac{1}{2} \text{tr}(K^{-1}YY^T)$$

- Very related to metric learning with Bregman divergences (i.e., logdet) where the optimization is over $X$ and not over $K$ [Lu et al. 09]

- This can be interpreted as learning the kernel where it is restricted to be generated from a low dimensional space.

- Extensions for multiple views using shared/private latent spaces, where the shared latent space represents all the views, and the private the individual views [Ek et al. 08, Salzmann et al. 10]
Conclusions

- GPs provide a simple and effective framework for MKL
- They allow for more complex combinations than simple linear mixing.
- We show that locality can be exploited to learn localized kernel combinations, which is effective in the presence of noise, or when the properties vary locally.
- Bayesian co-training which imposes agreement in a semi-supervised setting.
- Latent variables can be used to learn rich structures.
- Most of this work done by Mario C. Christoudias during his PhD
- Much more to do in the future!
- See later in the workshop C. Archambeau and F. Bach paper.