Introduction to Gaussian Processes

Raquel Urtasun

TTI Chicago

August 2, 2013

Which problems we will be looking at?

In this first lecture:

- Understand GPs from two perspectives:
  - weight view
  - function view
- Applications in Computer Vision

Let's look at the regression problem

Regression Problem

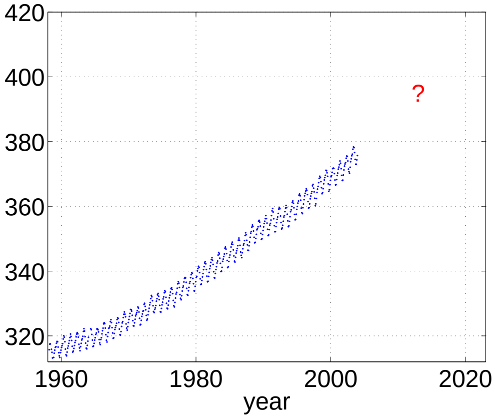


Figure: from C. Rasmussen

Regression Problem

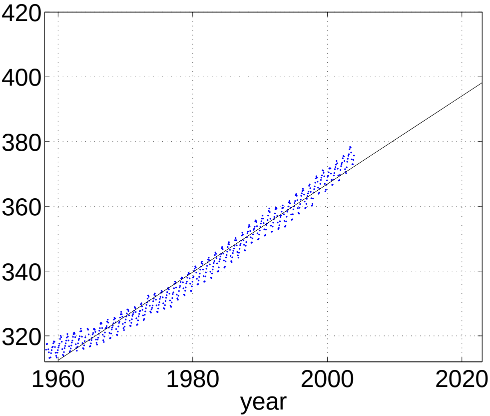


Figure: from C. Rasmussen

Regression Problem

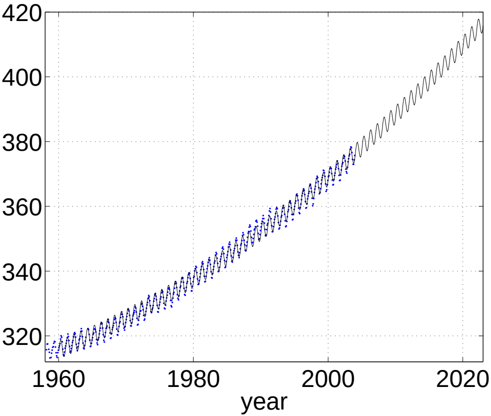


Figure: from C. Rasmussen

Regression Problem

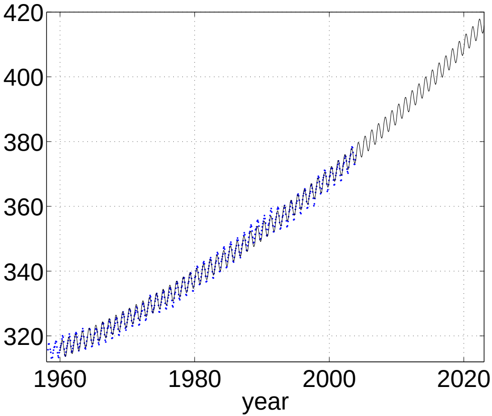


Figure: from C. Rasmussen

What's the difference between this two results?

Questions

- Model Selection
  - how to find out which model to use?
- Model Fitting
  - how do I fit the parameters?
  - what about over fitting?
- Can I trust the predictions, even if I am not sure of the parameters and the model structure?

Supervised Regression

- Assume an underlying process which generates "clean" data.
- **Task:** Recover underlying process from noisy observed data $\{\mathbf{x}^{(i)}, y_i\}$

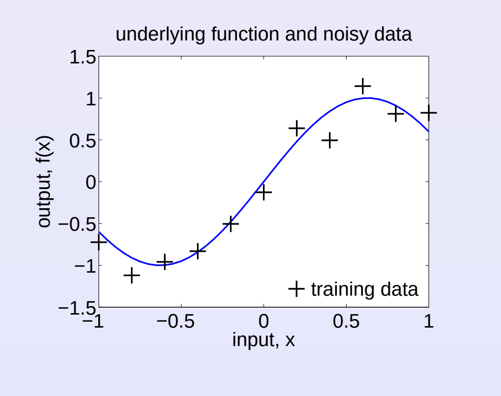


Figure: from H. Wallach

The weight view on GPs

Bayesian Linear Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

with i.i.d. noise $\eta \sim \mathcal{N}(0, \sigma^2)$

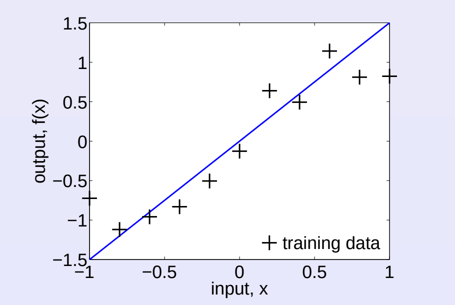


Figure: from H. Wallach

Bayesian Linear Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

with $\eta \sim \mathcal{N}(0, \sigma^2)$

- Likelihood of the parameters is

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{X}, \sigma^2\mathbf{I})$$

Bayesian Linear Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

with $\eta \sim \mathcal{N}(0, \sigma^2)$

- Likelihood of the parameters is

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{X}, \sigma^2\mathbf{I})$$

- Applying the Bayes Theorem we have

$$\underbrace{P(\mathbf{w}|\mathbf{y}, \mathbf{X})}_{posterior} \propto \underbrace{P(\mathbf{y}|\mathbf{X}, \mathbf{w})}_{likelihood} \underbrace{P(\mathbf{w})}_{prior}$$

Bayesian Linear Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \mathbf{x}, \qquad y = f + \eta$$

with $\eta \sim \mathcal{N}(0, \sigma^2)$

- Likelihood of the parameters is

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T \mathbf{X}, \sigma^2 \mathbf{I})$$

- Applying the Bayes Theorem we have

$$\underbrace{P(\mathbf{w}|\mathbf{y}, \mathbf{X})}_{posterior} \propto \underbrace{P(\mathbf{y}|\mathbf{X}, \mathbf{w})}_{likelihood} \underbrace{P(\mathbf{w})}_{prior}$$

- Typically we assume a Gaussian prior over the parameters

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) \propto \mathcal{N}(\mathbf{w}^T \mathbf{X}, \sigma^2 \mathbf{I}) \mathcal{N}(0, \Sigma)$$

Bayesian Linear Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

with $\eta \sim \mathcal{N}(0, \sigma^2)$

- Likelihood of the parameters is

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{X}, \sigma^2\mathbf{I})$$

- Applying the Bayes Theorem we have

$$\underbrace{P(\mathbf{w}|\mathbf{y}, \mathbf{X})}_{posterior} \propto \underbrace{P(\mathbf{y}|\mathbf{X}, \mathbf{w})}_{likelihood} \underbrace{P(\mathbf{w})}_{prior}$$

- Typically we assume a Gaussian prior over the parameters

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) \propto \mathcal{N}(\mathbf{w}^T\mathbf{X}, \sigma^2\mathbf{I})\mathcal{N}(0, \Sigma)$$

- What's the form of the posterior then? Why did we use a Gaussian prior?

Bayesian Linear Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

  with $\eta \sim \mathcal{N}(0, \sigma^2)$

- Likelihood of the parameters is

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T\mathbf{X}, \sigma^2\mathbf{I})$$

- Applying the Bayes Theorem we have

$$\underbrace{P(\mathbf{w}|\mathbf{y}, \mathbf{X})}_{posterior} \propto \underbrace{P(\mathbf{y}|\mathbf{X}, \mathbf{w})}_{likelihood}\underbrace{P(\mathbf{w})}_{prior}$$

- Typically we assume a Gaussian prior over the parameters

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) \propto \mathcal{N}(\mathbf{w}^T\mathbf{X}, \sigma^2\mathbf{I})\mathcal{N}(0, \Sigma)$$

- What's the form of the posterior then? Why did we use a Gaussian prior?

Bayesian Linear Regression

- The posterior is Gaussian!

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{A}^{-1})$$

  with $\mathbf{A} = \Sigma^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X}$

- How can we make predictions?

Bayesian Linear Regression

- The posterior is Gaussian!

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{A}^{-1})$$

with $\mathbf{A} = \Sigma^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X}$

- How can we make predictions?
- The predictive distribution is also Gaussian!

$$\begin{aligned} P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) &= \int P(f^*, \mathbf{w}|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}(\frac{1}{\sigma^2}\mathbf{x}^{*T}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}^{*T}\mathbf{A}^{-1}\mathbf{x}^*) \end{aligned}$$

Bayesian Linear Regression

- The posterior is Gaussian!

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{A}^{-1})$$

  with $\mathbf{A} = \Sigma^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X}$

- How can we make predictions?
- The predictive distribution is also Gaussian!

$$\begin{aligned} P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) &= \int P(f^*, \mathbf{w}|\mathbf{x}^*, \mathbf{X}, \mathbf{y})d\mathbf{w} \\ &= \mathcal{N}(\frac{1}{\sigma^2}\mathbf{x}^{*T}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}^{*T}\mathbf{A}^{-1}\mathbf{x}^*) \end{aligned}$$

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

  with i.i.d. noise $\eta \sim \mathcal{N}(0, \sigma^2)$

- How to model more complex functions?

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x}, \qquad y = f + \eta$$

  with i.i.d. noise $\eta \sim \mathcal{N}(0, \sigma^2)$

- How to model more complex functions?

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}), \qquad y = f + \eta$$

  with $\eta = \sim \mathcal{N}(0, \sigma^2)$ and basis functions $\phi(\mathbf{x})$

- How to model more complex functions?
- $P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})$ can be expressed in terms of inner products of $\phi(\mathbf{x})$

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}), \qquad y = f + \eta$$

  with $\eta = \sim \mathcal{N}(0, \sigma^2)$ and basis functions $\phi(\mathbf{x})$

- How to model more complex functions?
- $P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})$ can be expressed in terms of inner products of $\phi(\mathbf{x})$
- Why is this interesting?

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}), \qquad y = f + \eta$$

with $\eta = \sim \mathcal{N}(0, \sigma^2)$ and basis functions $\phi(\mathbf{x})$

- How to model more complex functions?
- $P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})$ can be expressed in terms of inner products of $\phi(\mathbf{x})$
- Why is this interesting?
- Well we can apply the kernel trick: we do not need to define $\phi(\mathbf{x})$ explicitly

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}), \qquad y = f + \eta$$

  with $\eta = \sim \mathcal{N}(0, \sigma^2)$ and basis functions $\phi(\mathbf{x})$
- How to model more complex functions?
- $P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})$ can be expressed in terms of inner products of $\phi(\mathbf{x})$
- Why is this interesting?
- Well we can apply the kernel trick: we do not need to define $\phi(\mathbf{x})$ explicitly
- How many basis functions should we use?

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}), \qquad y = f + \eta$$

  with $\eta = \sim \mathcal{N}(0, \sigma^2)$ and basis functions $\phi(\mathbf{x})$

- How to model more complex functions?
- $P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})$ can be expressed in terms of inner products of $\phi(\mathbf{x})$
- Why is this interesting?
- Well we can apply the kernel trick: we do not need to define $\phi(\mathbf{x})$ explicitly
- How many basis functions should we use?
- GPs come at our rescue!

Non-linear Bayesian Regression

- The linear regression model is

$$f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}), \qquad y = f + \eta$$

  with $\eta = \sim \mathcal{N}(0, \sigma^2)$ and basis functions $\phi(\mathbf{x})$
- How to model more complex functions?
- $P(f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})$ can be expressed in terms of inner products of $\phi(\mathbf{x})$
- Why is this interesting?
- Well we can apply the kernel trick: we do not need to define $\phi(\mathbf{x})$ explicitly
- How many basis functions should we use?
- GPs come at our rescue!

The function view of GPs

What's a GP?

- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.
- **Definition:** a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.

What's a GP?

- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.
- **Definition:** a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.
- A Gaussian is fully specified by a mean vector μ and a covariance matrix Σ

$$\mathbf{f} = (f_1, \cdots, f_n)^T \sim \mathcal{N}(\mu, \Sigma)$$

(index i)

What's a GP?

- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.
- **Definition:** a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.
- A Gaussian is fully specified by a mean vector μ and a covariance matrix Σ

$$\mathbf{f} = (f_1, \cdots, f_n)^T \sim \mathcal{N}(\mu, \Sigma)$$

(index i)

- A Gaussian process is fully specified by a mean function $m(\mathbf{x})$ and a PSD covariance function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim GP\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

(index $\mathbf{x}$)

What's a GP?

- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.
- **Definition:** a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.
- A Gaussian is fully specified by a mean vector μ and a covariance matrix Σ

$$\mathbf{f} = (f_1, \cdots, f_n)^T \sim \mathcal{N}(\mu, \Sigma)$$

  (index i)

- A Gaussian process is fully specified by a mean function $m(\mathbf{x})$ and a PSD covariance function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim GP\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

  (index $\mathbf{x}$)

- Typically $m(\mathbf{x}) = 0$

What's a GP?

- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.
- **Definition:** a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.
- A Gaussian is fully specified by a mean vector μ and a covariance matrix Σ

$$\mathbf{f} = (f_1, \cdots, f_n)^T \sim \mathcal{N}(\mu, \Sigma)$$

(index i)

- A Gaussian process is fully specified by a mean function $m(\mathbf{x})$ and a PSD covariance function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim GP\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

(index $\mathbf{x}$)

- Typically $m(\mathbf{x}) = 0$

Marginalization property

- Thinking of a GP as a Gaussian distribution with an infinitely long mean vector and an infinite by infinite covariance matrix may seem impractical
- Marginalization property

$$p(x) = \int p(x, y) dy$$

- For Gaussians

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right) \quad \text{then} \quad y^{(1)} \sim \mathcal{N}(\mu_1, \Sigma_{11})$$

- Therefore I only need to consider the points that I observe!

Marginalization property

- Thinking of a GP as a Gaussian distribution with an infinitely long mean vector and an infinite by infinite covariance matrix may seem impractical
- Marginalization property

$$p(x) = \int p(x, y) dy$$

- For Gaussians

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right) \quad \text{then} \quad y^{(1)} \sim \mathcal{N}(\mu_1, \Sigma_{11})$$

- Therefore I only need to consider the points that I observe!
- This is the **consistency** property

Marginalization property

- Thinking of a GP as a Gaussian distribution with an infinitely long mean vector and an infinite by infinite covariance matrix may seem impractical
- Marginalization property

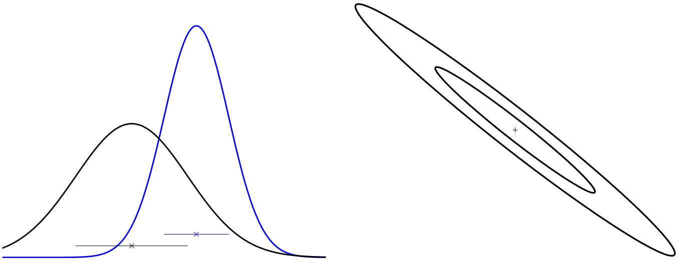
$$p(x) = \int p(x,y)dy$$

- For Gaussians

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right) \quad \text{then} \quad y^{(1)} \sim \mathcal{N}(\mu_1, \Sigma_{11})$$

- Therefore I only need to consider the points that I observe!
- This is the **consistency** property

The Gaussian Distribution



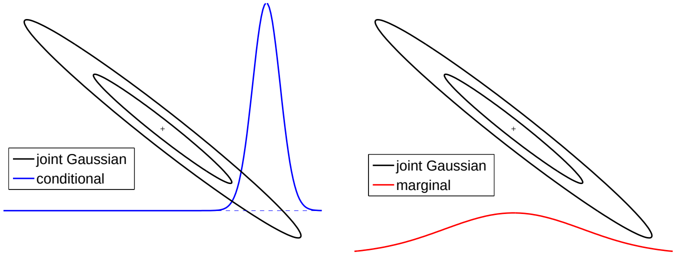
The Gaussian distribution is given by

$$p(\mathbf{x}|\mu, \Sigma) = \mathcal{N}(\mu, \Sigma) = (2\pi)^{-D/2}|\Sigma|^{-1/2} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right)$$

where μ is the mean vector and Σ the covariance matrix.

Figure: from C. Rasmussen

Conditionals and Marginals of a Gaussian



Both the conditionals and the marginals of a joint Gaussian are again Gaussian.

Figure: from C. Rasmussen

GP from Bayesian linear model

The Bayesian linear model is

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \quad \text{with} \quad \mathbf{w} \sim \mathcal{N}(0, \Sigma)$$

- The mean function is
$$\mathbb{E}[f(\mathbf{x})] = \mathbb{E}[\mathbf{w}^T]\mathbf{x} = 0$$
- Covariance is
$$\mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \mathbf{x}^T\,\mathbb{E}[\mathbf{w}\mathbf{w}^T]\mathbf{x}' = \mathbf{x}^T\Sigma\mathbf{x}'$$
- For any set of m basis functions, $\phi(\mathbf{x})$, the corresponding covariance function is
$$K(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = \phi(\mathbf{x}^{(p)})^T\Sigma\phi(\mathbf{x}^{(q)})$$
- Conversely, for every covariance function K, there is a possibly infinite expansion in terms of basis functions

$$K(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = \sum_{i=1}^{\infty} \lambda_i\phi_i(\mathbf{x}^{(p)})^T\phi_i(\mathbf{x}^{(q)})$$

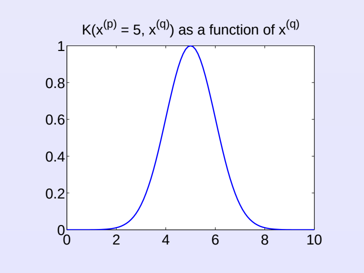
Covariance

- For any set of inputs $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$ we can compute K, which defines a joint distribution over function values

$$f(\mathbf{x}^{(1)}), \cdots, f(\mathbf{x}^{(n)}) \sim \mathcal{N}(0, \mathbf{K})$$

- Therefore, a GP specifies a **distribution over functions**
- Encode the prior knowledge by defining the kernel, which specifies the covariance between pairs of random variables, e.g.,

$$K(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = \exp(\frac{1}{2}||\mathbf{x}^{(p)} - \mathbf{x}^{(q)}||_2^2)$$



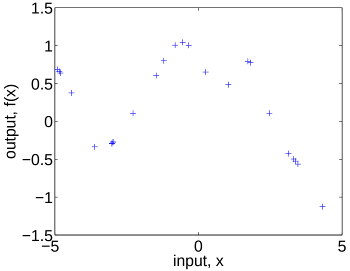
Covariance

- For any set of inputs $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$ we can compute K, which defines a joint distribution over function values

$$f(\mathbf{x}^{(1)}), \cdots, f(\mathbf{x}^{(n)}) \sim \mathcal{N}(0, \mathbf{K})$$

- Therefore, a GP specifies a **distribution over functions**
- Encode the prior knowledge by defining the kernel, which specifies the covariance between pairs of random variables, e.g.,

$$K(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = \exp(\frac{1}{2}||\mathbf{x}^{(p)} - \mathbf{x}^{(q)}||_2^2)$$



Gaussian Process Prior

- Given a set of inputs $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$, we can draw samples $f(\mathbf{x}^{(1)}), \cdots, f(\mathbf{x}^{(n)})$
- Example when using an RBF kernel

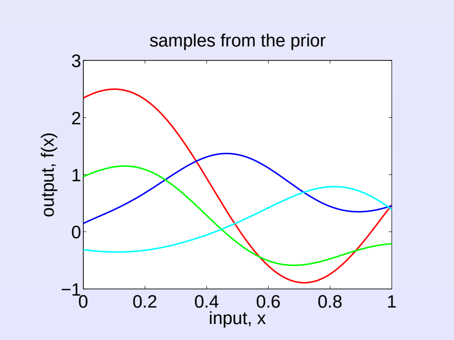


Figure: from H. Wallac

- How can we generate this samples?

Sampling

- Let's do sequential generation

$$p(f_1, \cdots, f_n | \mathbf{x}_1, \cdots, \mathbf{x}_n) = \prod_{i=1}^{n} p(f_i | f_{i-1}, \cdots, f_1, \mathbf{x}_i, \cdots, \mathbf{x}_1)$$

- Each term is again Gaussian since

$$p(x, y) = \mathcal{N}\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \right) \quad \Rightarrow \quad p(x|y) = \mathcal{N}(a + BC^{-1}(y - b), A - BC^{-1}B^T)$$

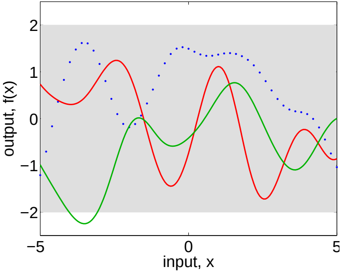


Figure: from C. Rasmussen

Noise free Observations

- Given noise-free training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, f^{(i)})\}$ we want to make predictions f^* about new points $\mathbf{X}^*$
- The GP prior says

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{X}^*) \\ K(\mathbf{X}^*, \mathbf{X}) & K(\mathbf{X}^*, \mathbf{X}^*) \end{bmatrix}\right)$$

- Condition $\{\mathbf{X}^*, \mathbf{f}^*\}$ on the training data $\{\mathbf{X}, \mathbf{f}\}$ to obtain the posterior

Noise free Observations

- Given noise-free training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, f^{(i)})\}$ we want to make predictions f^* about new points $\mathbf{X}^*$
- The GP prior says

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) & K(\mathbf{X},\mathbf{X}^*) \\ K(\mathbf{X}^*,\mathbf{X}) & K(\mathbf{X}^*,\mathbf{X}^*) \end{bmatrix}\right)$$

- Condition $\{\mathbf{X}^*, \mathbf{f}^*\}$ on the training data $\{\mathbf{X}, \mathbf{f}\}$ to obtain the posterior
- This restricts the posterior to contain functions which **agree** with the training data

Noise free Observations

- Given noise-free training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, f^{(i)})\}$ we want to make predictions f^* about new points $\mathbf{X}^*$
- The GP prior says

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{X}^*) \\ K(\mathbf{X}^*, \mathbf{X}) & K(\mathbf{X}^*, \mathbf{X}^*) \end{bmatrix}\right)$$

- Condition $\{\mathbf{X}^*, \mathbf{f}^*\}$ on the training data $\{\mathbf{X}, \mathbf{f}\}$ to obtain the posterior
- This restricts the posterior to contain functions which **agree** with the training data

Noise free Observations

- Given noise-free training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, f^{(i)})\}$ we want to make predictions f^* about new points $\mathbf{X}^*$
- The GP prior says

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) & K(\mathbf{X},\mathbf{X}^*) \\ K(\mathbf{X}^*,\mathbf{X}) & K(\mathbf{X}^*,\mathbf{X}^*) \end{bmatrix}\right)$$

- Condition $\{\mathbf{X}^*, \mathbf{f}^*\}$ on the training data $\{\mathbf{X}, \mathbf{f}\}$ to obtain the posterior
- This restricts the posterior to contain functions which **agree** with the training data
- The posterior is Gaussian $p(f^*|\mathbf{X}^*, \mathbf{X},) = \mathcal{N}(\mu, \mathbf{\Sigma})$ with

$$\begin{aligned} \mu &= K(\mathbf{X},\mathbf{X}^*)K(\mathbf{X},\mathbf{X})^{-1}\mathbf{f} \\ \Sigma &= K(\mathbf{X}^*,\mathbf{X}^*) - K(\mathbf{X},\mathbf{X}^*)K(\mathbf{X},\mathbf{X})^{-1}K(\mathbf{X}^*,\mathbf{X}) \end{aligned}$$

Example of Posterior

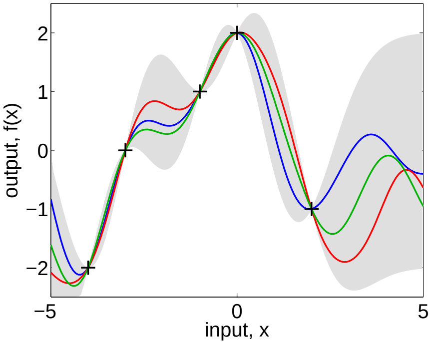


Figure: from C. Rasmussen

- All samples agree with observations
- Highest variance in regions with few training points

How to deal with noise?

- We have noisy observations $\{\mathbf{X}, \mathbf{y}\}$ with

$$\mathbf{y} = \mathbf{f} + \eta \quad \text{with} \quad \eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

- Conditioning on the training data $\{\mathbf{X}, \mathbf{y}\}$ gives a Gaussian predictive distribution $p(\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{y})$

$$\begin{aligned} \mu &= K(\mathbf{X}, \mathbf{X}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} \\ \Sigma &= K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}, \mathbf{X}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}] K(\mathbf{X}^*, \mathbf{X}) \end{aligned}$$

Model Selection: Hyperparameters

- Let's talk about the most employed kernel

$$K(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}) = \exp(-\frac{1}{2\theta^2}||\mathbf{x}^{(p)} - \mathbf{x}^{(q)}||_2^2)$$

- How can we choose θ?

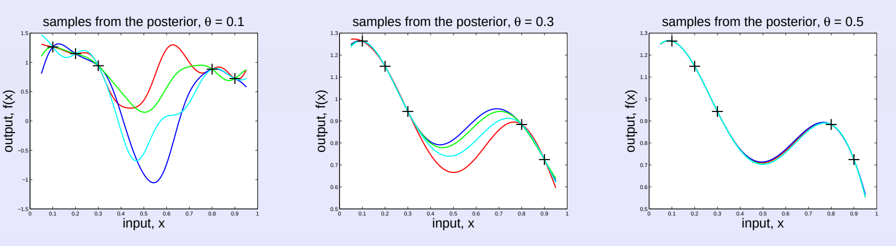


Figure: from H. Wallach

Maximum Likelihood

- If we don't have a prior on $P(\theta)$, the posterior for hyper parameter θ is

$$P(\theta|\mathbf{X}, \mathbf{y}) \propto P(\mathbf{y}|\mathbf{X}, \theta)$$

- In the log domain

$$\log P(\mathbf{y}|\mathbf{X}, \theta) = \underbrace{-\frac{1}{2} \log |K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}|}_{capacity} - \underbrace{\frac{1}{2} \mathbf{y}^T (K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{y}}_{model\ \ fitting} - \frac{n}{2} \log 2\pi$$

- Obtain hyperparameters

$$\underset{\theta}{\operatorname{argmin}} - \log P(\mathbf{y}|\mathbf{X}, \theta)$$

- What if we have $P(\theta)$?

Maximum Likelihood

- If we don't have a prior on $P(\theta)$, the posterior for hyper parameter θ is

$$P(\theta|\mathbf{X},\mathbf{y}) \propto P(\mathbf{y}|\mathbf{X},\theta)$$

- In the log domain

$$\log P(\mathbf{y}|\mathbf{X},\theta) = \underbrace{-\frac{1}{2}\log|K(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I}|}_{capacity} - \underbrace{\frac{1}{2}\mathbf{y}^T(K(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I})^{-1}\mathbf{y}}_{model\ \ fitting} - \frac{n}{2}\log 2\pi$$

- Obtain hyperparameters

$$\operatorname*{argmin}_{\theta} - \log P(\mathbf{y}|\mathbf{X},\theta)$$

- What if we have $P(\theta)$?
- This is not the "right" thing to do a Bayesian would say, as θ should be integrated out

Maximum Likelihood

- If we don't have a prior on $P(\theta)$, the posterior for hyper parameter θ is

$$P(\theta|\mathbf{X}, \mathbf{y}) \propto P(\mathbf{y}|\mathbf{X}, \theta)$$

- In the log domain

$$\log P(\mathbf{y}|\mathbf{X}, \theta) = \underbrace{-\frac{1}{2}\log|K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}|}_{capacity} - \underbrace{\frac{1}{2}\mathbf{y}^T(K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I})^{-1}\mathbf{y}}_{model\ \ fitting} - \frac{n}{2}\log 2\pi$$

- Obtain hyperparameters

$$\underset{\theta}{\operatorname{argmin}} - \log P(\mathbf{y}|\mathbf{X}, \theta)$$

- What if we have $P(\theta)$?
- This is not the "right" thing to do a Bayesian would say, as θ should be integrated out
- A pragmatic is very happy with this

Maximum Likelihood

- If we don't have a prior on $P(\theta)$, the posterior for hyper parameter θ is

$$P(\theta|\mathbf{X}, \mathbf{y}) \propto P(\mathbf{y}|\mathbf{X}, \theta)$$

- In the log domain

$$\log P(\mathbf{y}|\mathbf{X}, \theta) = \underbrace{-\frac{1}{2} \log |K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}|}_{capacity} - \underbrace{\frac{1}{2} \mathbf{y}^T (K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{y}}_{model\ \ fitting} - \frac{n}{2} \log 2\pi$$

- Obtain hyperparameters

$$\underset{\theta}{\operatorname{argmin}} - \log P(\mathbf{y}|\mathbf{X}, \theta)$$

- What if we have $P(\theta)$?
- This is not the "right" thing to do a Bayesian would say, as θ should be integrated out
- A pragmatic is very happy with this

$$\theta^{ML} = 0.3255$$

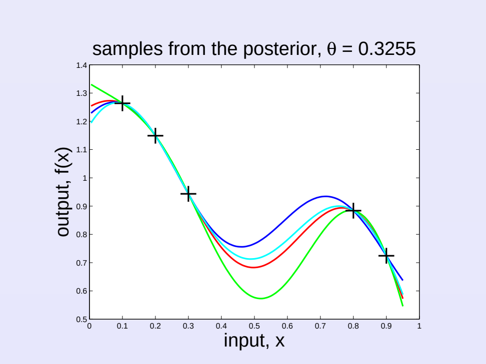


Figure: from H. Wallach

Interpretations

- Recall that the predictive distribution $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is Gaussian with

$$\begin{aligned}\mu(\mathbf{x}^*) &= k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ \Sigma(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]k(\mathbf{x}^*, \mathbf{X})\end{aligned}$$

- Notice that the mean in linear in two forms

$$\mu = \sum_{i=1}^{n} \beta_i \mathbf{y}^{(i)} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$$

- Note that the last one you might have seen e.g., SVMs, Representer theorem

Interpretations

- Recall that the predictive distribution $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is Gaussian with

$$\begin{aligned} \mu(\mathbf{x}^*) &= k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]^{-1}\mathbf{y} \\ \Sigma(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]k(\mathbf{x}^*, \mathbf{X}) \end{aligned}$$

- Notice that the mean in linear in two forms

$$\mu = \sum_{i=1}^{n} \beta_i \mathbf{y}^{(i)} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$$

- Note that the last one you might have seen e.g., SVMs, Representer theorem
- Cool thing is that α has closed form solution, no need to optimize over!

Interpretations

- Recall that the predictive distribution $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is Gaussian with

$$\begin{aligned} \mu(\mathbf{x}^*) &= k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ \Sigma(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]k(\mathbf{x}^*, \mathbf{X}) \end{aligned}$$

- Notice that the mean in linear in two forms

$$\mu = \sum_{i=1}^{n} \beta_i \mathbf{y}^{(i)} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$$

- Note that the last one you might have seen e.g., SVMs, Representer theorem
- Cool thing is that α has closed form solution, no need to optimize over!
- The variance is the difference between the *prior variance* and a term that says how much the data $\mathbf{X}$ has explained.

Interpretations

- Recall that the predictive distribution $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is Gaussian with

$$\begin{aligned}\mu(\mathbf{x}^*) &= k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]^{-1}\mathbf{y} \\ \Sigma(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]k(\mathbf{x}^*, \mathbf{X})\end{aligned}$$

- Notice that the mean in linear in two forms

$$\mu = \sum_{i=1}^{n} \beta_i \mathbf{y}^{(i)} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$$

- Note that the last one you might have seen e.g., SVMs, Representer theorem
- Cool thing is that α has closed form solution, no need to optimize over!
- The variance is the difference between the *prior variance* and a term that says how much the data $\mathbf{X}$ has explained.
- The variance is independent of the observed outputs $\mathbf{y}$

Interpretations

- Recall that the predictive distribution $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is Gaussian with

$$\begin{aligned}\mu(\mathbf{x}^*) &= k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ \Sigma(\mathbf{x}^*) &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{X}, \mathbf{x}^*)[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]k(\mathbf{x}^*, \mathbf{X})\end{aligned}$$

- Notice that the mean in linear in two forms

$$\mu = \sum_{i=1}^{n} \beta_i \mathbf{y}^{(i)} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$$

- Note that the last one you might have seen e.g., SVMs, Representer theorem
- Cool thing is that α has closed form solution, no need to optimize over!
- The variance is the difference between the *prior variance* and a term that says how much the data $\mathbf{X}$ has explained.
- The variance is independent of the observed outputs $\mathbf{y}$

Other Covariances: Periodic smooth

- First map the inputs to $u = (\cos(x), \sin(x))^T$, and then measure distance on the u space.
- Combine with the squared exponential we have

$$k_{periodic}(x, x') = \exp(-2\sin^2(\pi(x - x')/l^2)$$

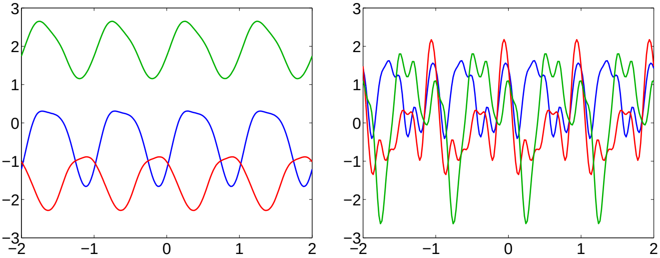


Figure: from C. Rasmussen

Other Covariances: mattern

- Mattern form stationary covariance but not necessarily differentiable
- Complicated function, lazy to write it ;)

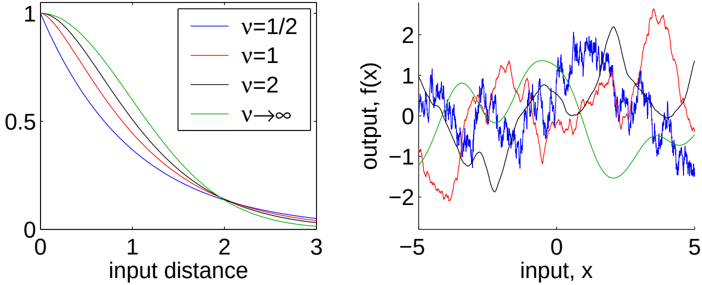


Figure: from C. Rasmussen

- More complex covariances can be created by summing and multiplying covariances

Other Covariances: mattern

- Mattern form stationary covariance but not necessarily differentiable
- Complicated function, lazy to write it ;)

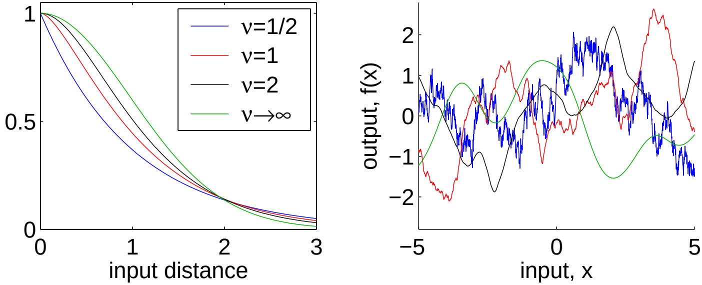


Figure: from C. Rasmussen

- More complex covariances can be created by summing and multiplying covariances

What if you want to do classification?

Binary Gaussian Process Classification

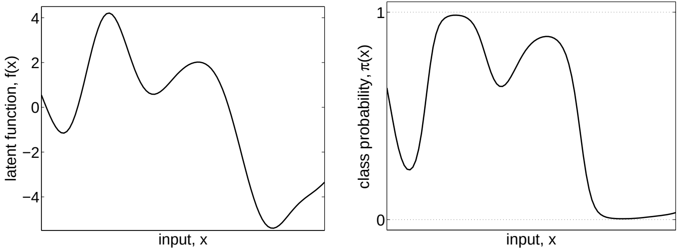
- Simplest thing is to use regression for classification
- More principled is to relate the class probability to the latent function **f** via an additional function

$$p(y = 1|f(x)) = \pi(x) = \psi(f(x))$$

with ψ a sigmoid function such as the **logistic** or **cumulative Gaussian**

- The likelihood is

$$p(y|f) = \prod_{i=1}^{n} p(y_i|f_i) = \prod_{i=1}^{n} \psi(y_i f_i)$$



Houston we have a problem!

- We have a GP prior on the latent function

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, \mathbf{K})$$

- The posterior becomes

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{X}, \mathbf{y})} = \frac{\mathcal{N}(\mathbf{f}|0, \mathbf{K})}{p(\mathbf{X}, \mathbf{y})} \prod_{i=1}^{n} \psi(y_i f_i)$$

- This is non-Gaussian!
- The prediction of the latent function at a new test point is intractable

$$p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{f}, \mathbf{X}, \mathbf{x}_*) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

- Same problem from the predictive class probability

$$p(\mathbf{y}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(\mathbf{y}_*|f_*) p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$$

- Resort to approximations: Laplace, EP, Variational Bounds

Houston we have a problem!

- We have a GP prior on the latent function

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, \mathbf{K})$$

- The posterior becomes

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{X}, \mathbf{y})} = \frac{\mathcal{N}(\mathbf{f}|0, \mathbf{K})}{p(\mathbf{X}, \mathbf{y})} \prod_{i=1}^{n} \psi(y_i f_i)$$

- This is non-Gaussian!
- The prediction of the latent function at a new test point is intractable

$$p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{f}, \mathbf{X}, \mathbf{x}_*) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

- Same problem from the predictive class probability

$$p(\mathbf{y}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(\mathbf{y}_*|f_*) p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$$

- Resort to approximations: Laplace, EP, Variational Bounds
- In practice for the mean prediction, doing GP regression works as well!

Houston we have a problem!

- We have a GP prior on the latent function

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, \mathbf{K})$$

- The posterior becomes

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{X}, \mathbf{y})} = \frac{\mathcal{N}(\mathbf{f}|0, \mathbf{K})}{p(\mathbf{X}, \mathbf{y})} \prod_{i=1}^{n} \psi(y_i f_i)$$

- This is non-Gaussian!
- The prediction of the latent function at a new test point is intractable

$$p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{f}, \mathbf{X}, \mathbf{x}_*) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

- Same problem from the predictive class probability

$$p(\mathbf{y}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(\mathbf{y}_*|f_*) p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$$

- Resort to approximations: Laplace, EP, Variational Bounds
- In practice for the mean prediction, doing GP regression works as well!

Are GPs useful in computer vision?

Applications in Computer Vision

Many applications, we will concentrate on a few if time permits

- Multiple kernel learning: object recognition
- GPs as an optimization tool: weakly supervised segmentation
- Human pose estimation from single images
- Flow estimation
- Fashion show

1) Object Recognition

- **Task:** Given an image $\mathbf{x}$, predict the class of the object present in the image $\mathbf{y} \in \mathcal{Y}$

$y \rightarrow \{car, bus, bicycle\}$

- Although this is a classification task, one can treat the categories as real values and formulate the problem as regression.

1) Object Recognition

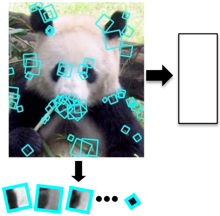
- **Task:** Given an image $\mathbf{x}$, predict the class of the object present in the image $\mathbf{y} \in \mathcal{Y}$

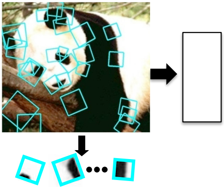
$y \rightarrow \{car, bus, bicycle\}$

- Although this is a classification task, one can treat the categories as real values and formulate the problem as regression.

How do we do Object Recognition?

- Given this two images, we will like to say if they are of the same class.





- Choose a representation for the images
  - Global descriptor of the full image
  - Local features: SIFT, SURF, etc.
- We need to choose a way to compute similarities
  - Histograms of local features (i.e., bags of words), pyramids, etc.
  - Kernels on global descriptors, e.g., RBF
  - $\cdots$

Multiple Kernel Learning (MKL)

- Why do we need to choose a single representation and a single similarity function?
- Which one is the best among all possible ones?
- Multiple kernel learning comes at our rescue, by learning which cues and similarities are more important for the prediction task.

$$\mathbf{K} = \sum_i \alpha_i \mathbf{K}_i$$

Multiple Kernel Learning (MKL)

- Why do we need to choose a single representation and a single similarity function?
- Which one is the best among all possible ones?
- Multiple kernel learning comes at our rescue, by learning which cues and similarities are more important for the prediction task.

$$\mathbf{K} = \sum_i \alpha_i \mathbf{K}_i$$

- This is just hyperparameter learning in GPs! No need for specialized SW!

Multiple Kernel Learning (MKL)

- Why do we need to choose a single representation and a single similarity function?
- Which one is the best among all possible ones?
- Multiple kernel learning comes at our rescue, by learning which cues and similarities are more important for the prediction task.

$$\mathbf{K} = \sum_i \alpha_i \mathbf{K}_i$$

- This is just hyperparameter learning in GPs! No need for specialized SW!

Efficient Learning Using GPs for Multiclass Problems

Supposed we want to emulate a 1-vs-all strategy as $|\mathcal{Y}| > 2$

- We define $\mathbf{y} \in \{-1, 1\}^{|\mathcal{Y}|}$
- We can employ maximum likelihood and learn all the parameters for all classifiers at once

$$\min_{\boldsymbol{\theta}, \boldsymbol{\alpha} > 0} - \sum_i \log p(\mathbf{y}^{(i)} | \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\alpha}) + \gamma_1 ||\boldsymbol{\alpha}||_1 + \gamma_2 ||\boldsymbol{\alpha}||_2$$

  with $\mathbf{y}^{(i)} \in \{-1, 1\}$ each of the individual problems.

- Efficient to do joint learning as we can share the covariance across all classes

Efficient Learning Using GPs for Multiclass Problems

Supposed we want to emulate a 1-vs-all strategy as $|\mathcal{Y}| > 2$

- We define $\mathbf{y} \in \{-1, 1\}^{|\mathcal{Y}|}$
- We can employ maximum likelihood and learn all the parameters for all classifiers at once

$$\min_{\boldsymbol{\theta}, \boldsymbol{\alpha} > 0} - \sum_i \log p(\mathbf{y}^{(i)} | \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\alpha}) + \gamma_1 ||\boldsymbol{\alpha}||_1 + \gamma_2 ||\boldsymbol{\alpha}||_2$$

  with $\mathbf{y}^{(i)} \in \{-1, 1\}$ each of the individual problems.
- Efficient to do joint learning as we can share the covariance across all classes
- What's the difference between θ and α?

Efficient Learning Using GPs for Multiclass Problems

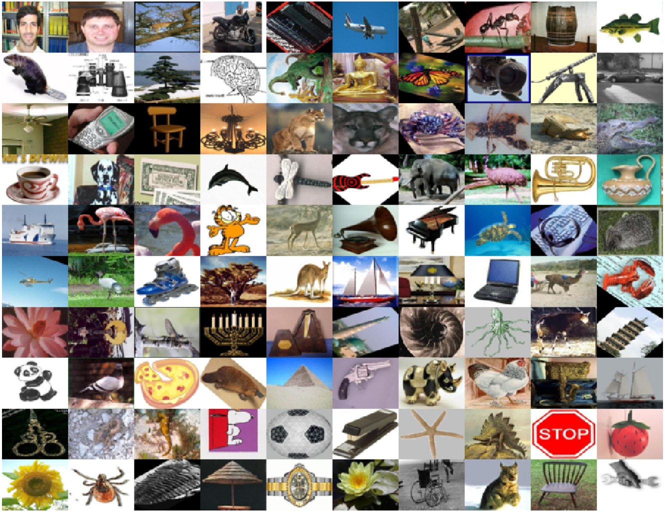
Supposed we want to emulate a 1-vs-all strategy as $|\mathcal{Y}| > 2$

- We define $\mathbf{y} \in \{-1, 1\}^{|\mathcal{Y}|}$
- We can employ maximum likelihood and learn all the parameters for all classifiers at once

$$\min_{\boldsymbol{\theta}, \boldsymbol{\alpha} > 0} - \sum_i \log p(\mathbf{y}^{(i)} | \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\alpha}) + \gamma_1 ||\boldsymbol{\alpha}||_1 + \gamma_2 ||\boldsymbol{\alpha}||_2$$

  with $\mathbf{y}^{(i)} \in \{-1, 1\}$ each of the individual problems.
- Efficient to do joint learning as we can share the covariance across all classes
- What's the difference between θ and α?

Caltech 101 dataset



Results: Caltech 101

[A. Kapoor, K. Graumann, R. Urtasun and T. Darrell, IJCV 2009]

Comparison with SVM kernel combination: kernels based on Geometric Blur (with and without distortion), dense PMK and spatial PMK on SIFT, etc.

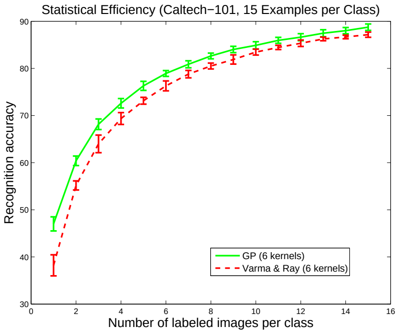


Figure: Average precision.

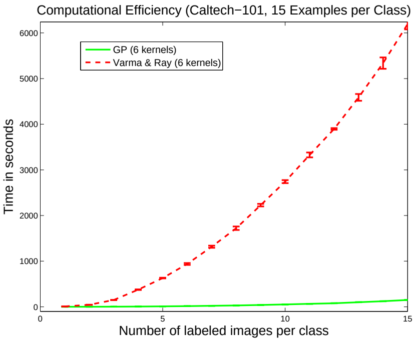


Figure: Time of computation.

Results: Caltech 101

[A. Kapoor, K. Graumann, R. Urtasun and T. Darrell, IJCV 2009]

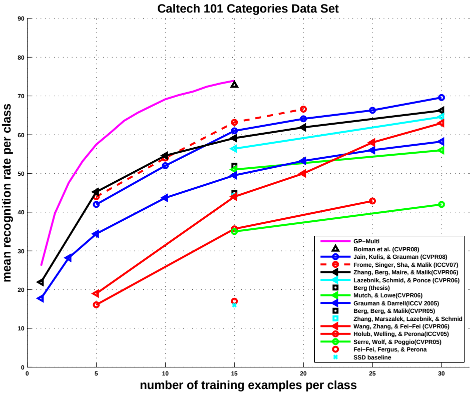


Figure: Comparison with the state of the art as in late 2008.

Other forms of MKL

Convex combination of kernels is too simple (not big boost reported), we need more complex (non-linear) combinations

- **Localized comb.**: (the weighting varies locally) (Christioudias et al. 09)

$$\mathbf{K}^{(v)} = \mathbf{K}_{np}^{(v)} \odot \mathbf{K}_{p}^{(v)}$$

  use structure to define $\mathbf{K}_{np}^{(v)}$, e.g., low-rank

- **Bayesian co-training** (Yu et al. 07)

$$\mathbf{K}_c = \left[\sum_j (\mathbf{K}_j + \sigma_j^2 \mathbf{I})^{-1} \right]^{-1}$$

- **Heteroscedastic Bayesian Co-training**: model noise with full covariance (Christoudias et al. 09)

Check out Mario Christoudias PhD thesis for more details

2) Optimization non-differentiable functions

Supposed you have a function that you want to optimize, but it is **non-differentiable** and also **computationally expensive** to evaluate, you can

- Discretize your space and evaluate discretized values in a grid (combinatorial)
- Randomly sample your parameters
- Utilize GPs to query where to look

GPs as an optimization tool

[N. Srinivas, A. Krause, S. Kakade and M. Seeger, ICML 2010]

Suppose we want to compute $\max f(x)$, we can simply

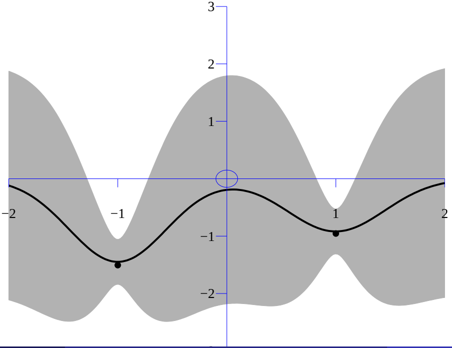
repeat

Choose $\mathbf{x}_t = arg \max_{\mathbf{x} \in D} \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x})$

Evaluate $\mathbf{y}_t = f(\mathbf{x}_t) + \epsilon_t$

Evaluate μ_t and σ_t

until budget reached



GPs as an optimization tool

[N. Srinivas, A. Krause, S. Kakade and M. Seeger, ICML 2010]

Suppose we want to compute $\max f(x)$, we can simply

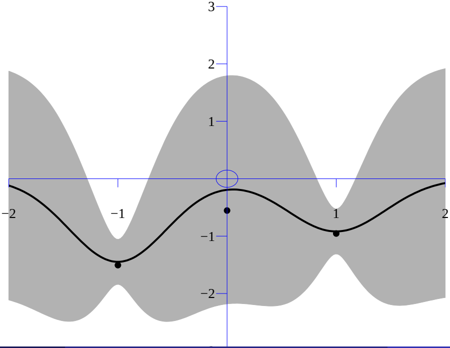
repeat

Choose $\mathbf{x}_t = arg \max_{\mathbf{x} \in D} \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t} \sigma_{t-1}(\mathbf{x})$

Evaluate $\mathbf{y}_t = f(\mathbf{x}_t) + \epsilon_t$

Evaluate μ_t and σ_t

until budget reached

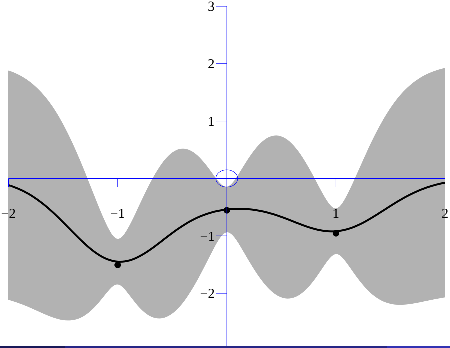


GPs as an optimization tool

[N. Srinivas, A. Krause, S. Kakade and M. Seeger, ICML 2010]

Suppose we want to compute $\max f(x)$, we can simply

repeat
  Choose $\mathbf{x}_t = arg \max_{\mathbf{x} \in D} \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x})$
  Evaluate $\mathbf{y}_t = f(\mathbf{x}_t) + \epsilon_t$
  Evaluate μ_t and σ_t
until budget reached



GPs as an optimization tool in vision

[A. Vezhnevets, V. Ferrari and J. Buhmann, CVPR 2012]

- Image segmentation in the weakly supervised setting, where the only labels are which classes are present in the scene.

$\mathbf{y} \in \{sky, building, tree\}$

- Train based on **expected agreement**, if I partition the dataset on two sets and I train on the first, it should predict the same as if I train on the second.
- This function is sum of indicator functions and thus non-differentiable.

GPs as an optimization tool in vision

[A. Vezhnevets, V. Ferrari and J. Buhmann, CVPR 2012]

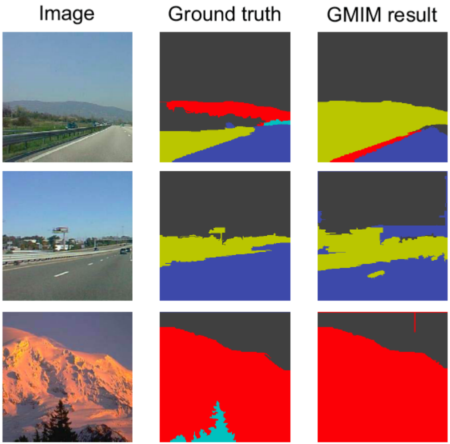
- Image segmentation in the weakly supervised setting, where the only labels are which classes are present in the scene.

$\mathbf{y} \in \{sky, building, tree\}$

- Train based on **expected agreement**, if I partition the dataset on two sets and I train on the first, it should predict the same as if I train on the second.
- This function is sum of indicator functions and thus non-differentiable.

Examples of Good Segmentations and Results

[A. Vezhnevets, V. Ferrari and J. Buhmann, CVPR 2012]



	[Tighe 10]	[Vezhnevets 11]	GMIM
supervision	fulll	weak	weak
average accuracy	29	14	21

3) Discriminative Approaches to Human Pose Estimation

- **Task:** given an image **x**, estimate the 3D location and orientation of the body parts **y**.

- We can treat this problem as a multi-output regression problem, where the input are image features, e.g., BOW, HOG, etc.
- The main challenges are
  - Poor imaging: motion blurred, occlusions, etc.
  - Need of large number of examples to represent all possible poses: represent variations in appearance and in pose.

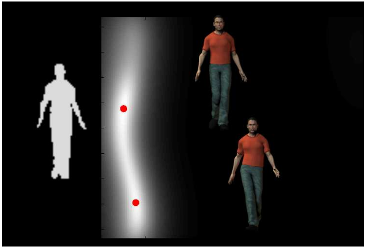
3) Discriminative Approaches to Human Pose Estimation

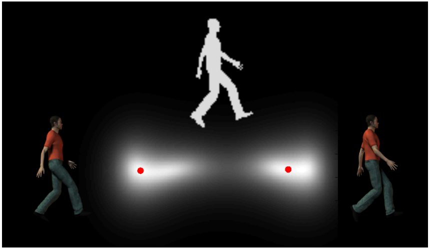
- **Task:** given an image $\mathbf{x}$, estimate the 3D location and orientation of the body parts $\mathbf{y}$.

- We can treat this problem as a multi-output regression problem, where the input are image features, e.g., BOW, HOG, etc.
- The main challenges are
  - Poor imaging: motion blurred, occlusions, etc.
  - Need of large number of examples to represent all possible poses: represent variations in appearance and in pose.

Challenges for GPs

- GP have complexity $\mathcal{O}(n^3)$, with n the number of examples, and cannot deal with large datasets in their standard form.
- This problem can't be solved directly as a regression task, since the mapping is multimodal: an image observation can represent more than one pose.





- Solutions to the first problem exist in the literature, they are called **sparsification** techniques

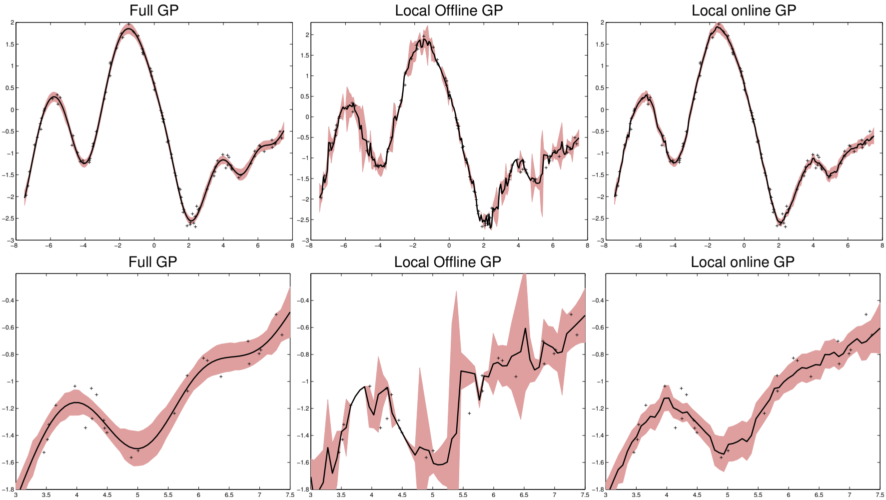
Dealing with multimodal mappings

- We can represent the regression problem as a mixture of experts, where each expert is a local GP.
- The experts should be selected online to avoid the possible boundary problems of clustering.
- Fast solution with up to millions of examples if combined with fast NN retrieval, e.g., LSH.

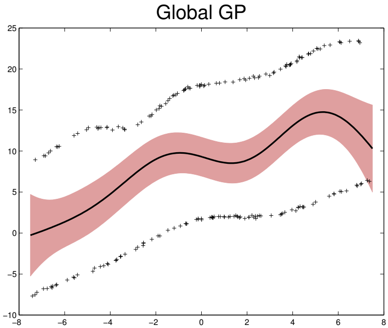
ONLINE: Inference of test point $\mathbf{x}_*$
T: number of experts, S: size of each expert

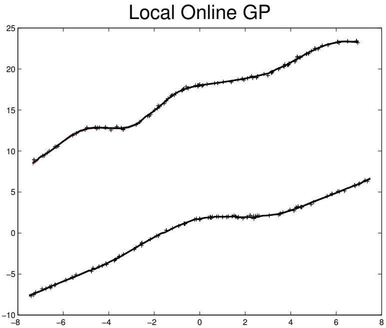
Find NN in $\mathbf{x}$ of $\mathbf{x}_*$
Find Modes in $\mathbf{y}$ of the NN retrieved
for $i = 1 \ldots T$ **do**
    Create a local GP for each mode i
    Retrieve hyper-parameters
    Compute mean μ and variance σ
end for
$p(\mathbf{f}_* | \mathbf{y}) \approx \sum_{i=1}^{T} \pi_i \mathcal{N}(\mu_i, \sigma_i^2)$

Online vs Clustering



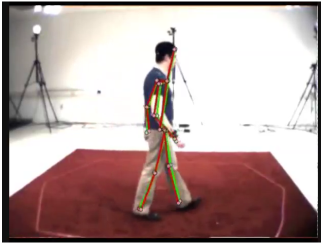
Single GP vs Mixture of Online GPs





Results: Humaneva

[R. Urtasun and T. Darrell, CVPR 2008]



	walk	jog	box	mono.	discrim.	dyn.
Lee et al. I	3.4	–	–	yes	no	no
Lee et al. II	**3.1**	–	–	yes	no	yes
Pope	4.53	4.38	9.43	yes	yes	no
Muendermann et al.	5.31	–	4.54	no	no	yes
Li et al.	–	–	20.0	yes	no	yes
Brubaker et al.	10.4	–	–	yes	no	yes
Our approach	3.27	**3.12**	**3.85**	yes	yes	no

Table: Comparison with state of the art (error in cm).

- Caviat: Oracle has to select the optimal mixture component

4) Flow Estimation with Gaussian Process

- Model a trajectory as a continuous dense flow field from a sparse set of vector sequences using Gaussian Process Regression
- Each velocity component modeled with an independent GP
- The flow can be expressed as

$$\phi(\mathbf{x}) = \mathbf{y}^{(u)}(\mathbf{x})\mathbf{i} + \mathbf{y}^{(v)}(\mathbf{x})\mathbf{j} + \mathbf{y}^{(t)}(\mathbf{x})\mathbf{k} \in \Re^3$$

  where $\mathbf{x} = (u, v, t)$

- Difficulties:
  - How to model a GPRF from different trajectories, which may have different lengths
  - How to handle multiple GPRF models trained from different numbers of trajectories with heterogeneous scales and frame rates

4) Flow Estimation with Gaussian Process

- Model a trajectory as a continuous dense flow field from a sparse set of vector sequences using Gaussian Process Regression
- Each velocity component modeled with an independent GP
- The flow can be expressed as

$$\phi(\mathbf{x}) = \mathbf{y}^{(u)}(\mathbf{x})\mathbf{i} + \mathbf{y}^{(v)}(\mathbf{x})\mathbf{j} + \mathbf{y}^{(t)}(\mathbf{x})\mathbf{k} \in \Re^3$$

  where $\mathbf{x} = (u, v, t)$
- Difficulties:
  - How to model a GPRF from different trajectories, which may have different lengths
  - How to handle multiple GPRF models trained from different numbers of trajectories with heterogeneous scales and frame rates
- Solution: normalize the length of the tracks before modeling with a GP, as well as the number of samples
- Classification based on the likelihood for each class

4) Flow Estimation with Gaussian Process

- Model a trajectory as a continuous dense flow field from a sparse set of vector sequences using Gaussian Process Regression
- Each velocity component modeled with an independent GP
- The flow can be expressed as

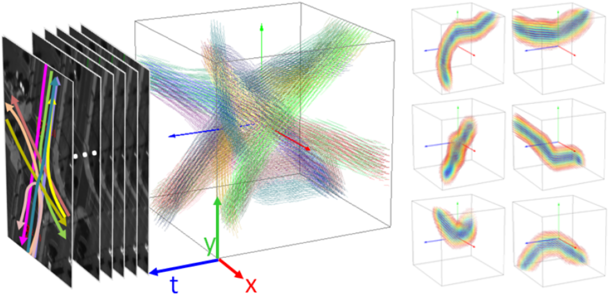
$$\phi(\mathbf{x}) = \mathbf{y}^{(u)}(\mathbf{x})\mathbf{i} + \mathbf{y}^{(v)}(\mathbf{x})\mathbf{j} + \mathbf{y}^{(t)}(\mathbf{x})\mathbf{k} \in \Re^3$$

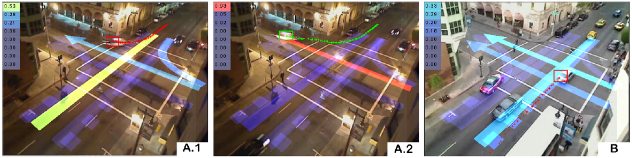
  where $\mathbf{x} = (u, v, t)$

- Difficulties:
  - How to model a GPRF from different trajectories, which may have different lengths
  - How to handle multiple GPRF models trained from different numbers of trajectories with heterogeneous scales and frame rates
- Solution: normalize the length of the tracks before modeling with a GP, as well as the number of samples
- Classification based on the likelihood for each class

Flow Classification and Anomaly Detection

[K. Kim, D. Lee and I. Essa, ICCV 2011]





5) 3D Shape Recovery for Online Shopping

- Interactive system for quickly modelling 3D body shapes from a single image
- Obtain their 3D body shapes so as to try on virtual garments online

5) 3D Shape Recovery for Online Shopping

- Interactive system for quickly modelling 3D body shapes from a single image
- Obtain their 3D body shapes so as to try on virtual garments online
- Interface for users to conveniently extract anthropometric measurements from a single photo, while using readily available scene cues for automatic image rectification

5) 3D Shape Recovery for Online Shopping

- Interactive system for quickly modelling 3D body shapes from a single image
- Obtain their 3D body shapes so as to try on virtual garments online
- Interface for users to conveniently extract anthropometric measurements from a single photo, while using readily available scene cues for automatic image rectification
- GPs to predict the body parameters from input measurements while correcting the aspect ratio ambiguity resulting from photo rectification

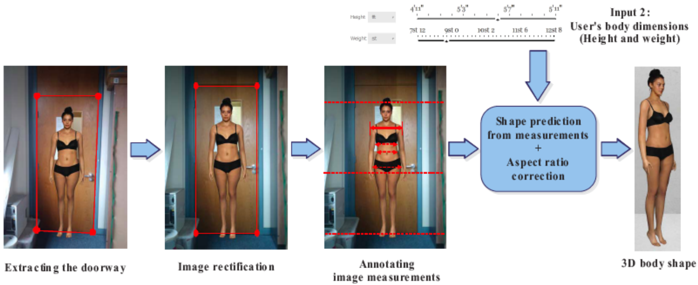
5) 3D Shape Recovery for Online Shopping

- Interactive system for quickly modelling 3D body shapes from a single image
- Obtain their 3D body shapes so as to try on virtual garments online
- Interface for users to conveniently extract anthropometric measurements from a single photo, while using readily available scene cues for automatic image rectification
- GPs to predict the body parameters from input measurements while correcting the aspect ratio ambiguity resulting from photo rectification

Creating the 3D Shape from Single Images

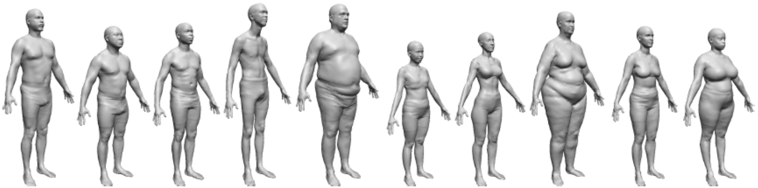
Manually annotate a set of five 2D measurements

- Well-defined by the anthropometric positions, easy to discern and unambiguous to users.
- Good correlations with the corresponding tape measurements and convey enough information for estimating the 3D body shape
- User's effort for annotation should be minimised



The role of the GPs

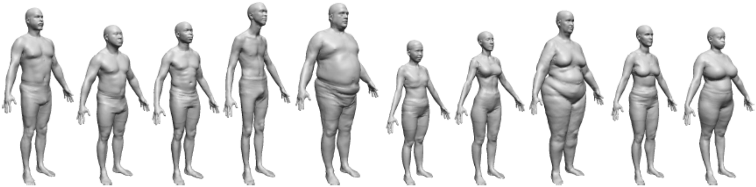
- A body shape estimator is learned to predict the 3D body shape from user's input, including both image measurements and actual measurements.
- Training set is (CAESAR) dataset (Robinette et al. 99), with 2000 bodies.



- Register each 3D instance in the dataset with a 3D morphable human body
- A 3D body is decomposed into a linear combination of body morphs

The role of the GPs

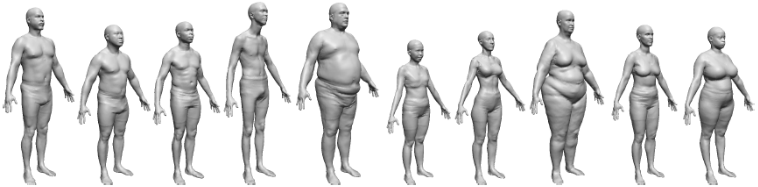
- A body shape estimator is learned to predict the 3D body shape from user's input, including both image measurements and actual measurements.
- Training set is (CAESAR) dataset (Robinette et al. 99), with 2000 bodies.



- Register each 3D instance in the dataset with a 3D morphable human body
- A 3D body is decomposed into a linear combination of body morphs
- Shape-from-measurements estimator can be formulated into a regression problem, $\mathbf{y}$ is the morph parameters and $\mathbf{x}$ is the user specified parameters.
- Multi-output done as independent predictors, each with a GP

The role of the GPs

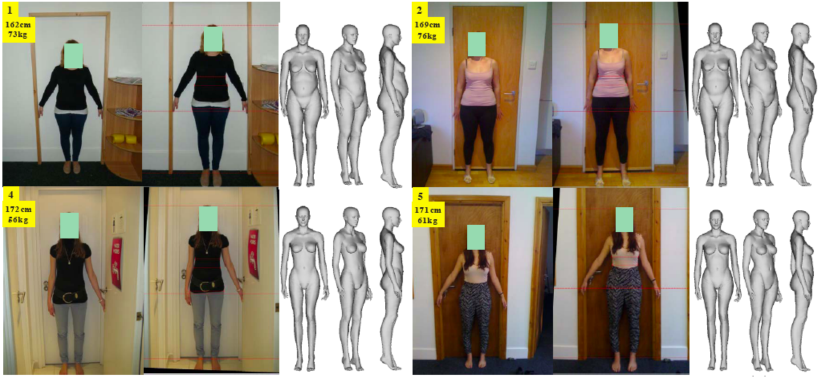
- A body shape estimator is learned to predict the 3D body shape from user's input, including both image measurements and actual measurements.
- Training set is (CAESAR) dataset (Robinette et al. 99), with 2000 bodies.



- Register each 3D instance in the dataset with a 3D morphable human body
- A 3D body is decomposed into a linear combination of body morphs
- Shape-from-measurements estimator can be formulated into a regression problem, $\mathbf{y}$ is the morph parameters and $\mathbf{x}$ is the user specified parameters.
- Multi-output done as independent predictors, each with a GP

Online Shopping

[Y. Chen and D. Robertson and R. Cipolla, BMVC 2011]



	Chest	Waist	Hips	Inner leg length
Error(cm)	1.52 ± 1.36	1.88 ± 1.06	3.10 ± 1.86	0.79 ± 0.90