

# Non-parametric Bayesian Methods in Machine Learning

Dr. Simon Rogers  
School of Computing Science  
University of Glasgow  
simon.rogers@glasgow.ac.uk  
@sdrogers

April 29, 2014

# Outline

- ▶ **FIX ME AT THE END**
- ▶ (My) Bayesian philosophy
- ▶ Gaussian Processes for Regression and Classification
  - ▶ GP preliminaries
  - ▶ Classification (including semi-supervised)
  - ▶ Regression application 1: clinical (dis)-agreement
  - ▶ Regressopn application 2: typing on touch-screens
- ▶ Dirichlet Process flavoured Cluster Models
  - ▶ DP preliminaries
  - ▶ Identifying metabolites
  - ▶ (if time) Cluster models for multiple data views

# About me

- ▶ I'm not a statistician by training (don't ask me to prove anything!).
- ▶ Education:
  - ▶ Undergraduate Degree: Electrical and Electronic Engineering (Bristol)
  - ▶ PhD: Machine Learning Techniques for Microarray Analysis (Bristol)
- ▶ Currently:
  - ▶ Lecturer: Computing Science
  - ▶ Research Interests: Machine Learning and Applied Statistics in Computational Biology and Human-Computer Interaction (HCI)

# Lecture 1: Bayesian Inference

Dr. Simon Rogers  
School of Computing Science  
University of Glasgow  
simon.rogers@glasgow.ac.uk  
@sdrogers

April 29, 2014

# Bayesian Inference

Standard setup:

- ▶ We have some data  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
- ▶ We have a model  $p(\mathbf{X}|\Theta)$
- ▶ We define a prior  $p(\Theta)$

# Bayesian Inference

Standard setup:

- ▶ We have some data  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
- ▶ We have a model  $p(\mathbf{X}|\Theta)$
- ▶ We define a prior  $p(\Theta)$
- ▶ We use Bayes rule (and typically lots of computation) to compute (or estimate) the posterior:

$$p(\Theta|\mathbf{X}) = \frac{p(\mathbf{X}|\Theta)p(\Theta)}{p(\mathbf{X})}$$

# Why be Bayesian?

# Why be Bayesian?

- ▶ Within ML we are often interested in making predictions (predicting  $y_*$  from  $\mathbf{x}_*$ ).
- ▶ Being Bayesian allows us to *average* over uncertainty in parameters when making predictions:

$$p(y_*|\mathbf{x}_*, \mathbf{X}) = \int p(y_*|\mathbf{x}_*, \boldsymbol{\Theta})p(\boldsymbol{\Theta}|\mathbf{X}) d\boldsymbol{\Theta}$$



# Why be Bayesian?

- ▶ Within ML we are often interested in making predictions (predicting  $y_*$  from  $\mathbf{x}_*$ ).
- ▶ Being Bayesian allows us to *average* over uncertainty in parameters when making predictions:

$$p(y_*|\mathbf{x}_*, \mathbf{X}) = \int p(y_*|\mathbf{x}_*, \boldsymbol{\Theta})p(\boldsymbol{\Theta}|\mathbf{X}) d\boldsymbol{\Theta}$$

- ▶ Bayes rule tells us how this uncertainty should change as data appear.

## Lecture 2: Gaussian Process Basics

Dr. Simon Rogers  
School of Computing Science  
University of Glasgow  
simon.rogers@glasgow.ac.uk  
@sdrogers

April 29, 2014

# Gaussian Processes

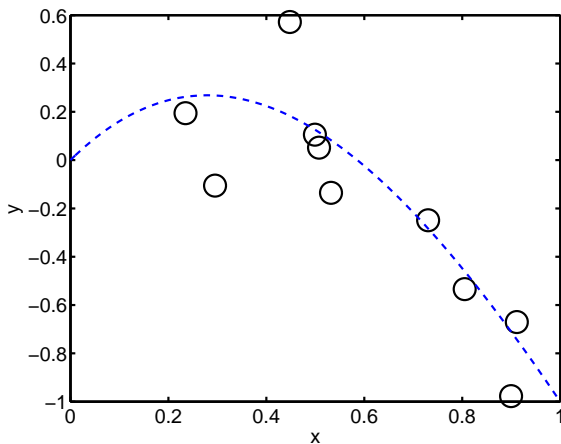


Figure 1 : A familiar problem: learn the underlying function (blue) from the observed data (crosses).

## A parametric approach?

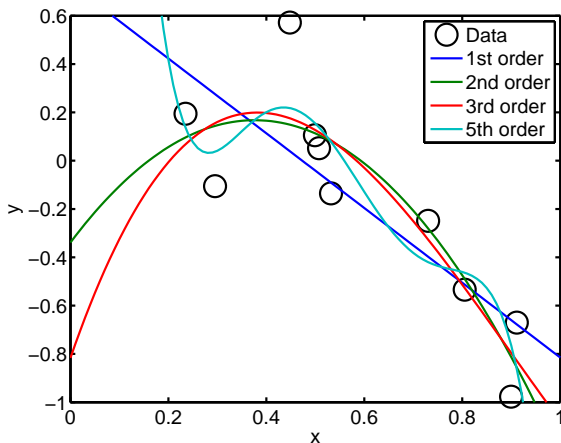


Figure 2 : Polynomials fitted by least squares.

It's easy to under and over-fit. What if we have no idea of the parametric form of the function?

# A non-parametric approach - Gaussian Processes

- ▶ Rather than forcing us to choose a particular parametric form, a Gaussian Process (GP) allows us to place a prior distribution directly on *functions*
- ▶ With a GP prior we can:
  - ▶ Sample functions from the prior
  - ▶ Incorporate data to get a *posterior* distribution over functions
  - ▶ Make predictions

## Visual example – prior

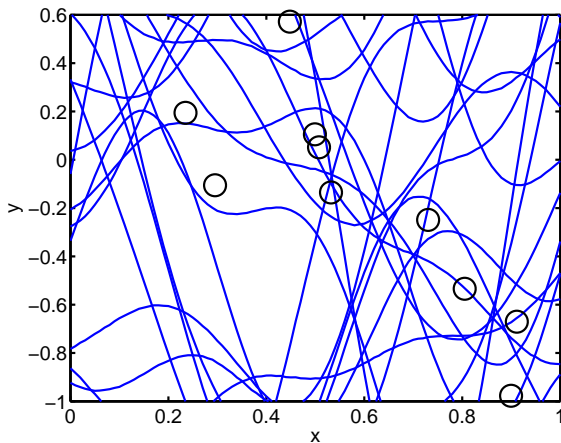


Figure 3 : Some functions drawn from a GP prior.

## Visual example – posterior

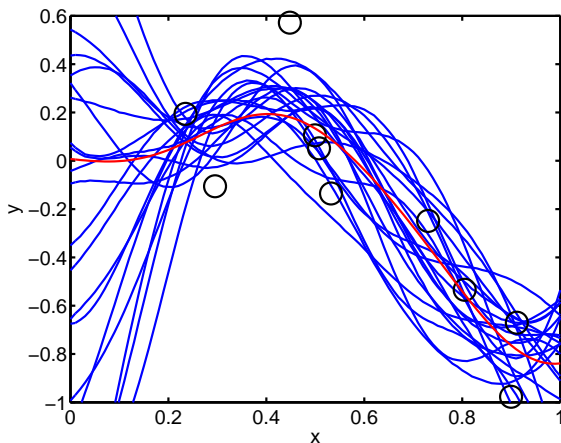


Figure 4 : Some functions drawn from the GP posterior. Posterior mean is shown in red.

## Visual example – predictions

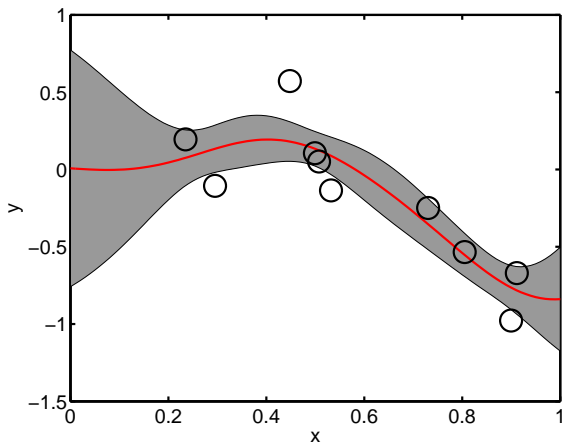


Figure 5 : Predictive mean and standard deviations.



## Some formalities

- ▶ We observe  $N$  training points, each of which consists of a set of features  $\mathbf{x}_n$  and a target  $y_n$ .
- ▶ We can stack all of the  $y_n$  into a vector and  $\mathbf{x}_n$  into a matrix:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$$

# GP definition

- ▶ The GP assumes that the vector of *all possible*  $y_n$  is a draw from a Multi-Variate Gaussian (MVG).
- ▶ We don't observe *all possible* values (if we did, we wouldn't need to make predictions!)

# GP definition

- ▶ The GP assumes that the vector of *all possible*  $y_n$  is a draw from a MVG.
- ▶ We don't observe *all possible* values (if we did, we wouldn't need to make predictions!)
- ▶ But the marginal densities of a MVG are also MVGs so the subset we observe are also a draw from a MVG.

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$$

- ▶ i.e. if we have  $N$  training points, we're dealing with an  $N$ -dimensional MVG

# GP definition

- ▶ The GP assumes that the vector of *all possible*  $y_n$  is a draw from a MVG.
- ▶ We don't observe *all possible* values (if we did, we wouldn't need to make predictions!)
- ▶ But the marginal densities of a MVG are also MVGs so the subset we observe are also a draw from a MVG.

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$$

- ▶ i.e. if we have  $N$  training points, we're dealing with an  $N$ -dimensional MVG
- ▶ With mean  $\boldsymbol{\mu}$  (normally 0) and covariance  $\mathbf{C}$

# GP definition

- ▶ The GP assumes that the vector of *all possible*  $y_n$  is a draw from a MVG.
- ▶ We don't observe *all possible* values (if we did, we wouldn't need to make predictions!)
- ▶ But the marginal densities of a MVG are also MVGs so the subset we observe are also a draw from a MVG.

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$$

- ▶ i.e. if we have  $N$  training points, we're dealing with an  $N$ -dimensional MVG
- ▶ With mean  $\boldsymbol{\mu}$  (normally 0) and covariance  $\mathbf{C}$
- ▶  $\mathbf{x}_n$  looks to have disappeared – we find it inside  $\mathbf{C}$

# GP definition

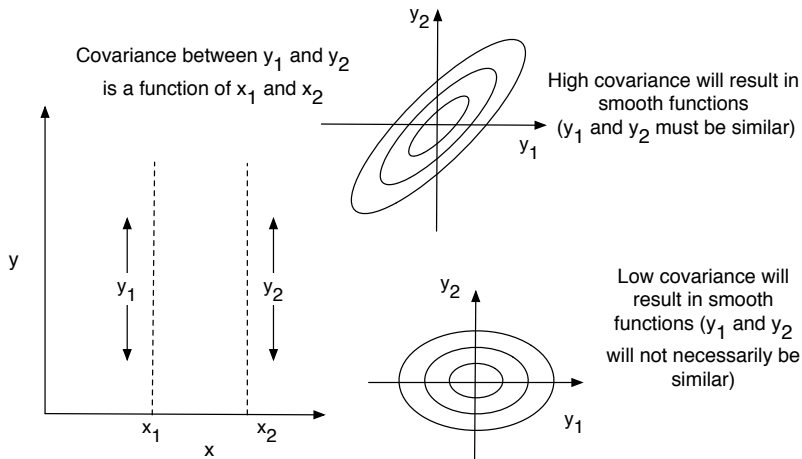


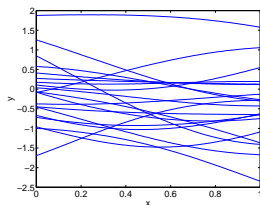
Figure 6 : Schematic of GP prior for two function values.

# Covariance functions

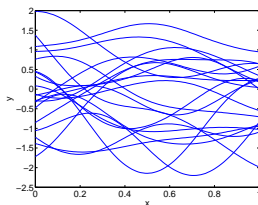
- ▶ By choosing a covariance function, we are making an assumption on the *smoothness* of the regression function.
- ▶ Common choices:
  - ▶ Linear:  $C(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$
  - ▶ RBF:  $C(\mathbf{x}_1, \mathbf{x}_2) = \exp \{ -0.5 \gamma \| \mathbf{x}_1 - \mathbf{x}_2 \|^2 \}$
  - ▶ And many, many more.
- ▶ More details: <http://www.gaussianprocess.org/gpml/>
  - ▶ (Free) book
  - ▶ Code

# Hyper-parameters

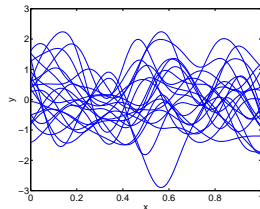
$$C(\mathbf{x}_1, \mathbf{x}_2) = \exp \left\{ -0.5\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2 \right\}$$



(a)  $\gamma = 1$



(b)  $\gamma = 10$



(c)  $\gamma = 100$

**Figure 7 :** Varying hyper-parameters in an RBF covariance varies the smoothness of the function.



# Optimising hyper-parameters

# Making predictions

- ▶ If we assume no observation noise, we can place GP prior directly on  $\mathbf{y}$
- ▶ If we observe  $\mathbf{y}$  and want to predict  $y_*$  for a new observation  $\mathbf{x}_*$ :
  - ▶ Construct joint Density (it's a Gaussian):

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} C(\mathbf{X}, \mathbf{X}) & C(\mathbf{X}, \mathbf{x}_*) \\ C(\mathbf{x}_*, \mathbf{X}) & C(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right)$$

- ▶ And then use standard results for Gaussian conditionals:

$$p(y_* | \mathbf{y}, \mathbf{X}, \mathbf{x}_*) = \mathcal{N}(\mu_*, \sigma_*^2)$$

where

$$\begin{aligned} \mu_* &= C(\mathbf{x}_*, \mathbf{X}) C(\mathbf{X}, \mathbf{X})^{-1} \mathbf{y} \\ \sigma_*^2 &= C(\mathbf{x}_*, \mathbf{x}_*) - C(\mathbf{x}_*, \mathbf{X}) C(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{x}_*) \end{aligned}$$

<http://orion.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>

## Noise-free example

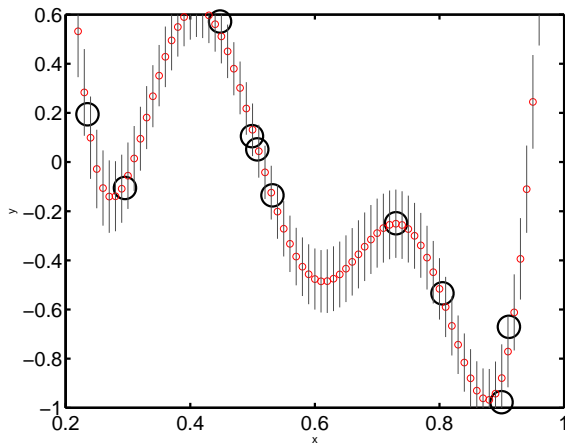


Figure 8 :  $\mu_* \pm \sigma_*$  for a noise-free GP at lots of test points

## Predictions with noise

- ▶ Assuming Gaussian observation noise, we introduce a set of latent variables,  $f_n$  and place the GP prior on these.

$$y_n \sim \mathcal{N}(f_n, \sigma^2), \quad \mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$

## Predictions with noise

- ▶ Assuming Gaussian observation noise, we introduce a set of latent variables,  $f_n$  and place the GP prior on these.

$$y_n \sim \mathcal{N}(f_n, \sigma^2), \quad \mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$

- ▶ Because both terms are Gaussian, the noise can be pushed into the covariance function:

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} C(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & C(\mathbf{X}, \mathbf{x}_*) \\ C(\mathbf{x}_*, \mathbf{X}) & C(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right)$$

## Predictions with noise

- Assuming Gaussian observation noise, we introduce a set of latent variables,  $f_n$  and place the GP prior on these.

$$y_n \sim \mathcal{N}(f_n, \sigma^2), \quad \mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$

- Because both terms are Gaussian, the noise can be pushed into the covariance function:

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} C(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & C(\mathbf{X}, \mathbf{x}_*) \\ C(\mathbf{x}_*, \mathbf{X}) & C(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right)$$

- And, as before (except now predicting  $f_*$ ):

$$p(f_* | \mathbf{y}, \mathbf{X}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(\mu_*, \sigma_*^2)$$

where

$$\begin{aligned} \mu_* &= C(\mathbf{x}_*, \mathbf{X}) [C(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} \\ \sigma_*^2 &= C(\mathbf{x}_*, \mathbf{x}_*) - C(\mathbf{x}_*, \mathbf{X}) [C(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}]^{-1} K(\mathbf{X}, \mathbf{x}_*) \end{aligned}$$

## Example with noise

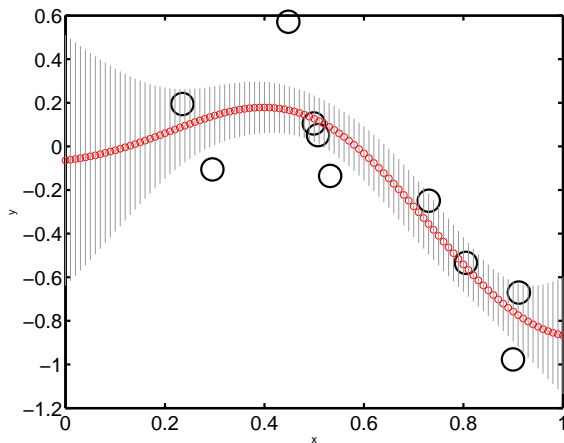


Figure 9 :  $\mu_* \pm \sigma_*$  at lots of test points when observation noise is included.

## Lecture 3: Application: Touchscreen typing

Dr. Simon Rogers  
School of Computing Science  
University of Glasgow  
simon.rogers@glasgow.ac.uk  
@sdrogers

April 29, 2014



# Typing on touchscreens

- ▶ Most people have smartphones
- ▶ Most smartphones have touchscreens
- ▶ Touchscreens are small
- ▶ Keyboards on touchscreens are small
- ▶ Typing on them is hard!
  - ▶ ...but people type on them a lot

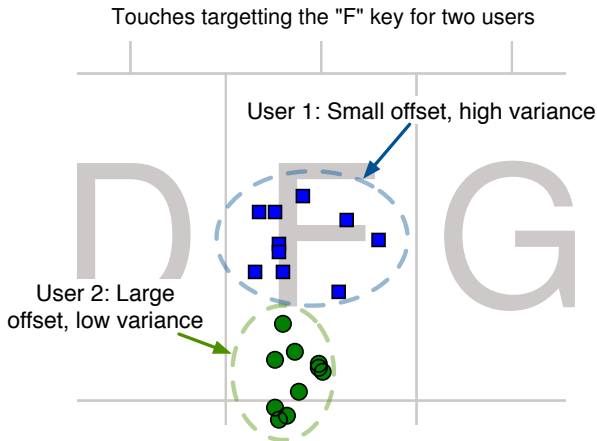
# Background 1: Why is it hard?

- ▶ Occlusion of target by finger
- ▶ 'fat finger' problem
- ▶ Small targets
- ▶ Demo: <http://bit.ly/1nBws97>

# Background 1: Why is it hard?

- ▶ Occlusion of target by finger
- ▶ 'fat finger' problem
- ▶ Small targets
- ▶ Demo: <http://bit.ly/1nBws97>
- ▶ Quite a bit of work in this area:
  - ▶ Add some

## Background 2: All users are different



**Figure 10 :** Touches recorded by two users aiming for the 'F' key. User 2 has high bias and low variance, user 1 has low bias and high variance.

## Background 3: Current systems (maybe?)

- ▶ Touch is boxed into nearest key.
- ▶ Key ID is passed to a Statistical Language Model (SLM).
- ▶ SLM is made up of probabilities of observing certain character strings (from large text corpora).
- ▶ SLM can swap characters to make the character string more likely.
  - ▶ e.g. 'HELLP' → 'HELLO'

# Our idea

- ▶ There is a lot of uncertainty present in touch (bias and variance)
- ▶ Boxing a touch into a key is probably bad
- ▶ Why can't we pass a *distribution* to the SLM?
  - ▶ Pass the uncertainty onwards
  - ▶ Being Bayesian!

# Our idea

- ▶ There is a lot of uncertainty present in touch (bias and variance)
- ▶ Boxing a touch into a key is probably bad
- ▶ Why can't we pass a *distribution* to the SLM?
  - ▶ Pass the uncertainty onwards
  - ▶ Being Bayesian!
- ▶ Can use a user specific GP regression model to predict target from input touch.

# Our idea



**Figure 11 :** Train GPs to predict the intended touch from an input touch. The flexibility of GPs means that the mean and covariance of the offset can vary across the keyboard.



# Our idea



**Figure 11 :** Train GPs to predict the intended touch from an input touch. The flexibility of GPs means that the mean and covariance of the offset can vary across the keyboard.

# Our idea



Figure 11 : Train GPs to predict the intended touch from an input touch. The flexibility of GPs means that the mean and covariance of the offset can vary across the keyboard.

# Our idea

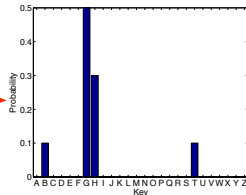


Figure 11 : Train GPs to predict the intended touch from an input touch. The flexibility of GPs means that the mean and covariance of the offset can vary across the keyboard.

# Our idea



Integrate  
predictive  
Gaussian over  
keys to obtain  
distribution



BAB  $\rightarrow$  BAG

Combine  
probabilities with  
those from  
language model

Figure 12 : The complete system

# The model

- ▶ We use independent GP regressions for predicting  $x$  and  $y$  offsets.
- ▶ Training data:
  - ▶ Each user typed phrases provided to them.
  - ▶ Data: the  $x, y$  location of the recorded touch. Target: the center of the intended key minus the touch (i.e. the offset).

# The model

- ▶ We use independent GP regressions for predicting  $x$  and  $y$  offsets.
- ▶ Training data:
  - ▶ Each user typed phrases provided to them.
  - ▶ Data: the  $x, y$  location of the recorded touch. Target: the center of the intended key minus the touch (i.e. the offset).
- ▶ Used a GP with zero mean and a composite covariance:

$$C(\mathbf{x}_1, \mathbf{x}_2) = a\mathbf{x}_1^T \mathbf{x}_2 + (1 - a) \exp\{-\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|^2\}$$

# The model

- ▶ We use independent GP regressions for predicting  $x$  and  $y$  offsets.
- ▶ Training data:
  - ▶ Each user typed phrases provided to them.
  - ▶ Data: the  $x, y$  location of the recorded touch. Target: the center of the intended key minus the touch (i.e. the offset).
- ▶ Used a GP with zero mean and a composite covariance:

$$C(\mathbf{x}_1, \mathbf{x}_2) = a\mathbf{x}_1^T\mathbf{x}_2 + (1 - a)\exp\{-\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|^2\}$$

- ▶ 10 participants, each did  $3 \times 45$  minute sessions, typing whilst sitting, standing and walking. [more details in paper]

# The model

- ▶ We use independent GP regressions for predicting  $x$  and  $y$  offsets.
- ▶ Training data:
  - ▶ Each user typed phrases provided to them.
  - ▶ Data: the  $x, y$  location of the recorded touch. Target: the center of the intended key minus the touch (i.e. the offset).
- ▶ Used a GP with zero mean and a composite covariance:

$$C(\mathbf{x}_1, \mathbf{x}_2) = a \mathbf{x}_1^T \mathbf{x}_2 + (1 - a) \exp\{-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2\}$$

- ▶ 10 participants, each did  $3 \times 45$  minute sessions, typing whilst sitting, standing and walking. [more details in paper]
- ▶ Compared:
  - ▶ GPtype (our system), Swiftkey (commercial Android keyboard), GP only (just offset, no SLM), baseline (boxing, no SLM).



# Results

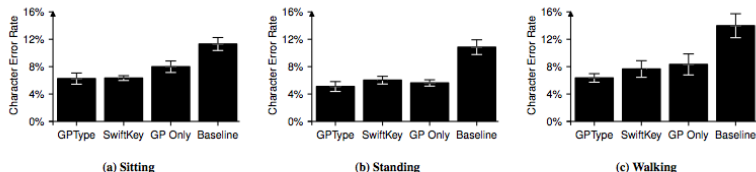


Figure 4. Character error rates for the two keyboards we evaluated, separated by mobility condition (Study 2). Plots show mean and standard error across all participants. The baseline method represents the literal keys touched, while GP Only shows the keys hit after the mean GP offset is applied.

## Figure 13 : Results of GPType experiment

- ▶ GPType marginally (stat sig) better than Swiftkey.
  - ▶ A **lot** of people work on SwiftKey
- ▶ Baseline awful!

# Conclusions

- ▶ GP regression is key to the approach: we make no parametric assumptions (what would they be?)
- ▶ ...and get probabilistic predictions
- ▶ ...that can be fed to the SLM – (un)certainity is passed to the SLM
- ▶ Performance is promising

# Conclusions

- ▶ GP regression is key to the approach: we make no parametric assumptions (what would they be?)
- ▶ ...and get probabilistic predictions
- ▶ ...that can be fed to the SLM – (un)certainity is passed to the SLM
- ▶ Performance is promising
- ▶ Also in the paper:
  - ▶ Using *pressure* to allow users *explicit* control of variance.  
Allows users to switch SLM off (when typing slang / names etc) or rely on it more for words they think it will get right (humans are quite good at predicting auto-correct errors).

# Conclusions

- ▶ GP regression is key to the approach: we make no parametric assumptions (what would they be?)
- ▶ ...and get probabilistic predictions
- ▶ ...that can be fed to the SLM – (un)certainity is passed to the SLM
- ▶ Performance is promising
- ▶ Also in the paper:
  - ▶ Using *pressure* to allow users *explicit* control of variance. Allows users to switch SLM off (when typing slang / names etc) or rely on it more for words they think it will get right (humans are quite good at predicting auto-correct errors).
- ▶ More info:
  - ▶ <http://www.youtube.com/watch?v=1lQI5gV5l74>
  - ▶ <http://pokristensson.com/pubs/WeirEtAlCHI2014.pdf>
  - ▶ Acknowledgements: Daryl Weir, Per Ola Kristensson, Keith Vertanen, Henning Pohl

## Lecture 4: Application: Clinical Ratings

Dr. Simon Rogers  
School of Computing Science  
University of Glasgow  
simon.rogers@glasgow.ac.uk  
@sdrogers

April 29, 2014

# Clinicians disagree in AandE