Non-parametric Bayesian Methods in Machine Learning

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

April 30, 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
  - Idenfityting metabolites
  - (if time) Cluster models for multiple data views

About me

- I'm not a statistican 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 30, 2014

Bayesian Inference

Standard setup:

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

Bayesian Inference

Standard setup:

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

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

Why be Bayesian?

Why be Bayesian?

- Within ML we are often interested in making predictions (predicing y_* from $\mathbf{x}_*$).
- Being Bayesian allows us to *average* over uncertainity 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 (predicing y_* from $\mathbf{x}_*$).
- Being Bayesian allows us to *average* over uncertainity 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 30, 2014

Gaussian Processes

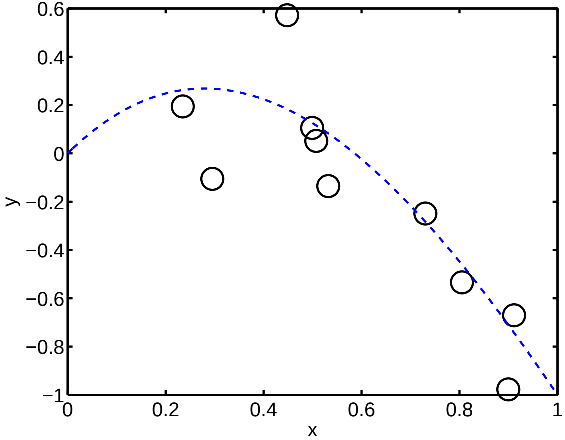


Figure 1 : A familiar problem: learn the underlying function (blue) from the observed data (crosses).

A parametric approach?

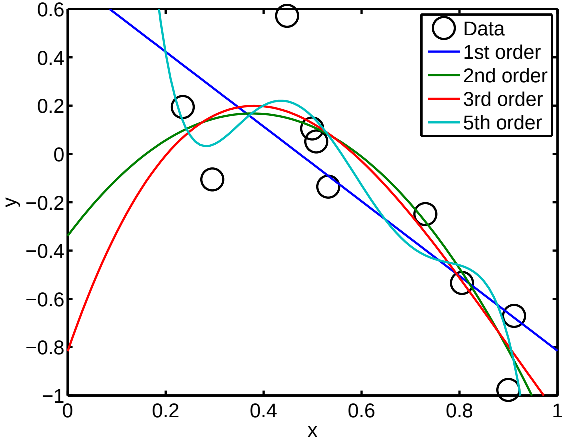


Figure 2 : Polynomials fitted by least squares.

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

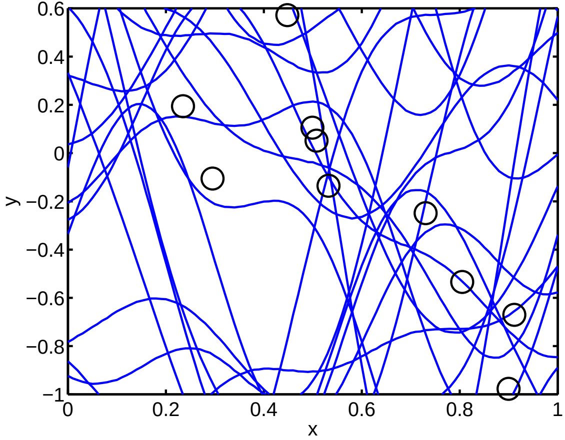


Figure 3 : Some functions drawn from a GP prior.

Visual exmample – posterior

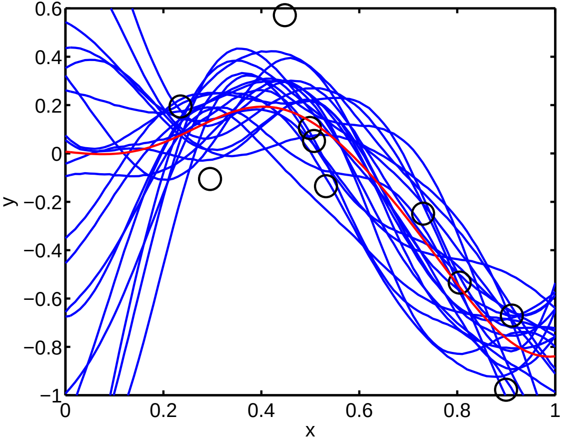


Figure 4 : Some functions drawn from the GP posterior. Posterior mean is shown in red.

Visual example – predictions

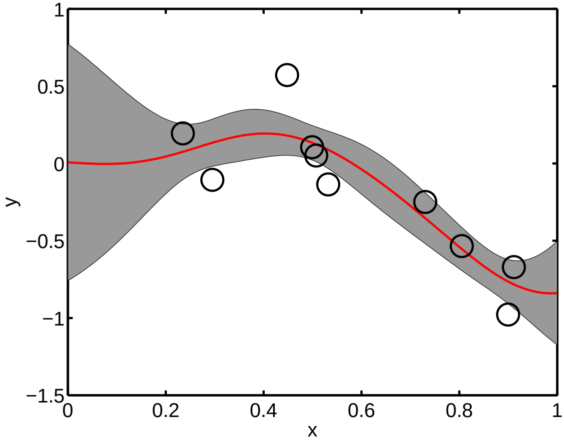


Figure 5 : Predictive mean and standard deviations.

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

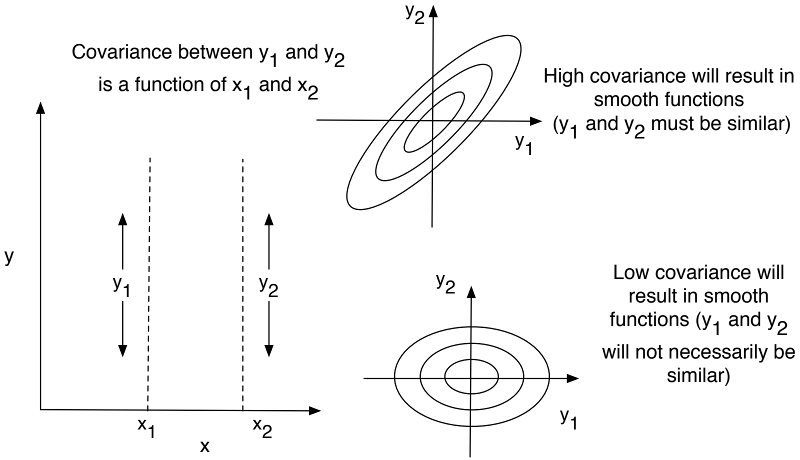


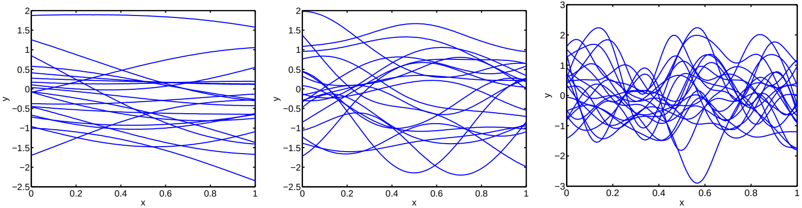
Figure 6 : Schematic of GP prior for two function values.

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

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

  - And then use standard results for Gaussian conditionals:

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

where

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

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

Noise-free example

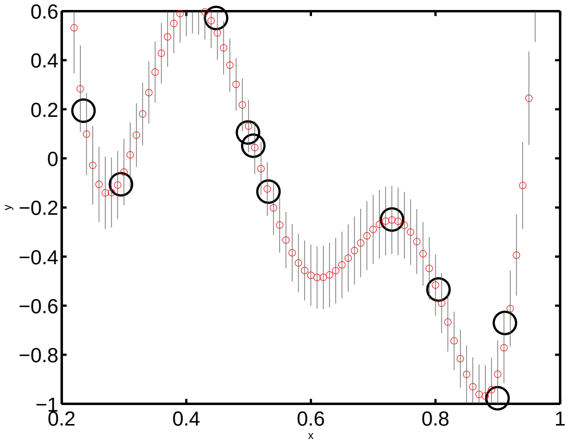


Figure 8 : $\mu_* \pm \sigma_*$ for a noise-free GP at lots of test points

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:

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

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:

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

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

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

where

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

Example with noise

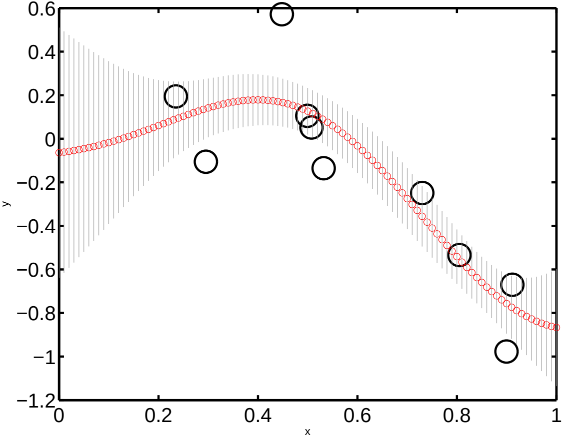


Figure 9 : $\mu_* \pm \sigma_*$ at lots of test points when observation noise is included.

Lecture 3: Application: Touchscreen typing

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

April 30, 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

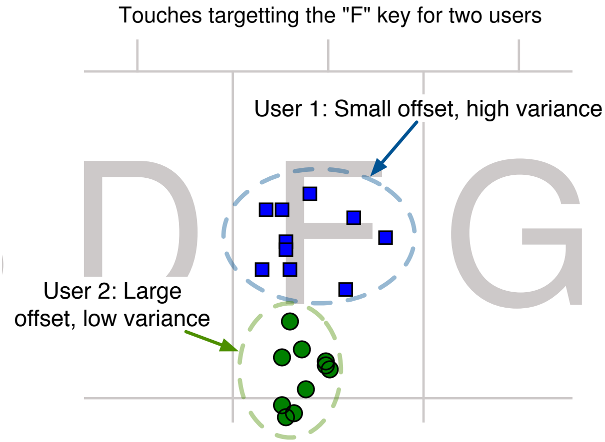


Figure 10 : Touches recorded by two users aiming for the ‘F’ key. User 2 has high bias and low variance, user 1 has low bias and high variance.

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 $\rightarrow$ 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

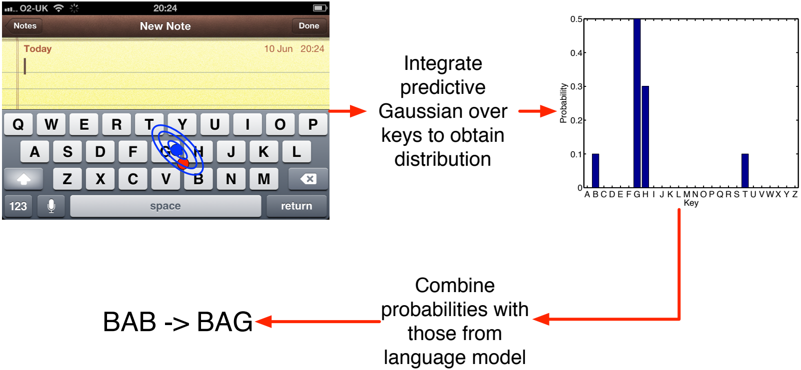


Figure 12 : The complete system

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

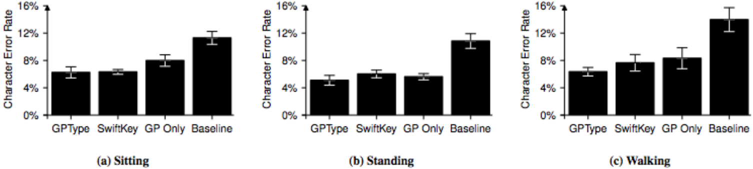


Figure 4. Character error rates for the two keyboards we evaluated, separated by mobility condition (Study 2). Plots show mean and standard error across all participants. The baseline method represents the literal keys touched, while GP Only shows the keys hit after the mean GP offset is applied.

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 tying 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 tying 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: GPs for classification and ordinal regression

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

April 30, 2014

GPs for Classification and ordinal regression

- What if our observation model is non-Gaussian?
  - Classification:

$$P(y_n = 1 | f_n) = \int_{-\infty}^{f_n} \mathcal{N}(z | 0, 1)\ dz = \phi(f_n)$$

  - Logistic Regression:

$$P(y_n = k | f_n) = \phi(b_{k+1}) - \phi(b_k)$$

  - etc
- Analytical inference is no longer possible
- I'll cover how to do inference in these models and extensions with the *auxiliary variable trick*

Binary classification

- Problem setup: we observe N data / target pairs $(\mathbf{x}_n, y_n)$ where $y_n \in \{0, 1\}$
- Place a GP prior on a set of latent variables f_n

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$

- Use the probit likelihood:

$$P(y_n = 1 | f_n) = \phi(f_n) = \int_{-\infty}^{f_n} \mathcal{N}(z | 0, 1)\ dz$$

- Inference in this form is hard

Auxiliary Variable Trick

- Re-write the probit function:

$$\begin{aligned} P(y_n = 1|f_n) &= \int_{-\infty}^{f_n} N(z|0,1)\ dz \\ &= \int_{-\infty}^{0} N(z|-f_n,1)\ dz \\ &= \int_{0}^{\infty} N(z|f_n,1)\ dz \\ &= \int_{-\infty}^{\infty} \delta(z>0)\mathcal{N}(z|f_n,1)\ dz \end{aligned}$$

where $\delta(expr)$ is 1 if *expr* is true, and 0 otherwise.

Auxiliary Variable Trick

- If we define $P(y_n = 1|z_n) = \delta(z_n > 0)$ then we have:

$$P(y_n = 1|f_n) = \int_{-\infty}^{\infty} P(y_n = 1|z_n)p(z_n|f_n)\ dz_n$$

- and could therefore remove the integral to obtain a model including z_n:

$$p(y_n = 1, z_n|f_n) = P(y_n = 1|z_n)p(z_n|f_n)$$

- Doing inference in this model (i.e. with additional variables z_n) is much easier (but still not analytically tractable)
- Note: $P(y_n = 0|z_n) = \delta(z_n < 0)$

Example - Gibbs sampling for binary classification

- An easy way to perform inference in the augmented model is via Gibbs sampling
- Sample $z_n|f_n, y_n$:

$$\begin{aligned} p(z_n|f_n, y_n = 0) &\propto \delta(z_n < 0)\mathcal{N}(z_n|f_n, 1) \\ p(z_n|f_n, y_n = 1) &\propto \delta(z_n < 1)\mathcal{N}(z_n|f_n, 1) \end{aligned}$$

Example - Gibbs sampling for binary classification

- An easy way to perform inference in the augmented model is via Gibbs sampling
- Sample $z_n | f_n, y_n$:

$$\begin{aligned} p(z_n | f_n, y_n = 0) &\propto \delta(z_n < 0)\mathcal{N}(z_n | f_n, 1) \\ p(z_n | f_n, y_n = 1) &\propto \delta(z_n < 1)\mathcal{N}(z_n | f_n, 1) \end{aligned}$$

- Sample $\mathbf{f} | \mathbf{z}, \mathbf{C}$

$$p(\mathbf{f} | \mathbf{z}, \mathbf{C}) = \mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f)$$

  where

$$\boldsymbol{\Sigma}_f = \left(\mathbf{I} + \mathbf{C}^{-1}\right)^{-1}, \quad \boldsymbol{\mu}_f = \boldsymbol{\Sigma}_f^{-1}\mathbf{z}$$

- Repeat ad infinitum

Example - Gibbs sampling for binary classification

- To make predictions:
  - At each sampling step, do a (noise-free) GP regression using the current sample of $\mathbf{f}$ to get a density over f_* (Details in a previous slide).
  - Sample a specific realisation of f_* from this density.
  - Compute $\phi(f_*)$ (or sample a z_* and then record whether it's > 0 or not)
  - Average this value over all Gibbs sampling iterations!

Example - binary classification

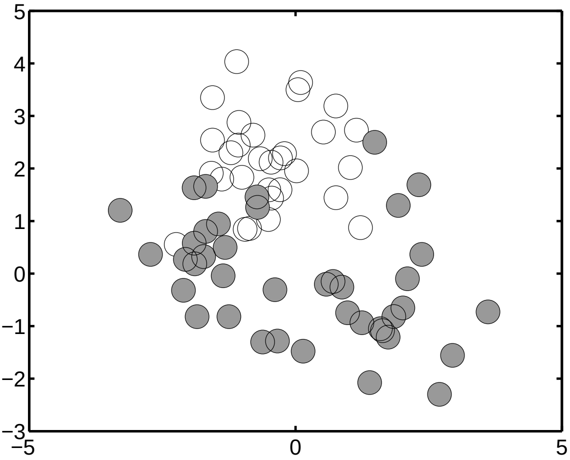


Figure 14 : Some simple classification data

Example - binary classification

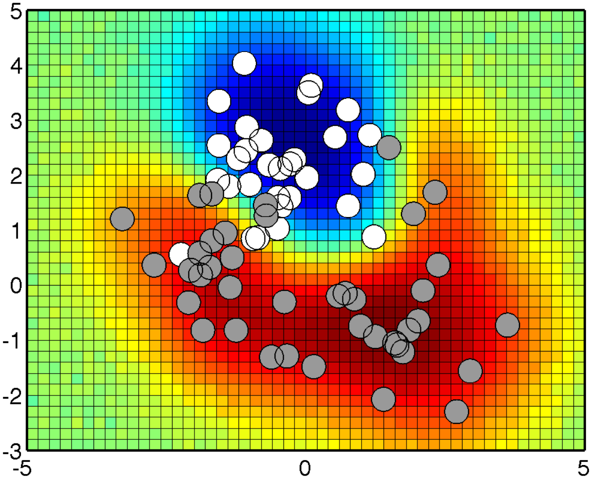


Figure 15 : Predictive probabilities averaged over 1000 Gibbs samples using an RBF covariance. As γ is increased, the model overfits.

Example - binary classification

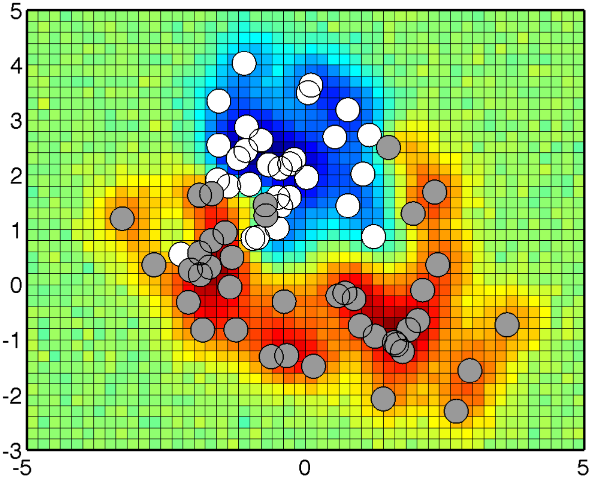


Figure 15 : Predictive probabilities averaged over 1000 Gibbs samples using an RBF covariance. As γ is increased, the model overfits.

Example - binary classification

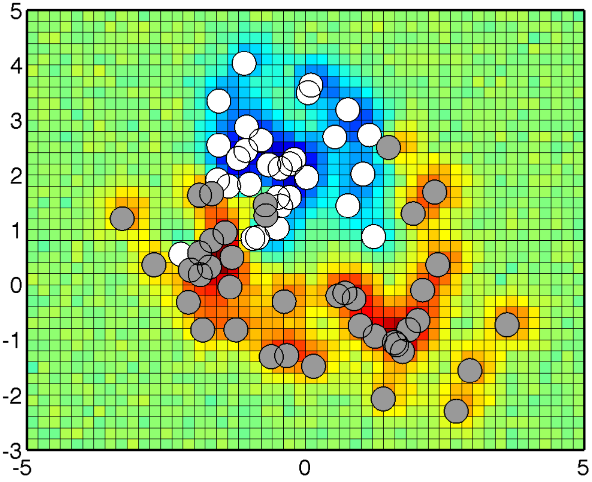


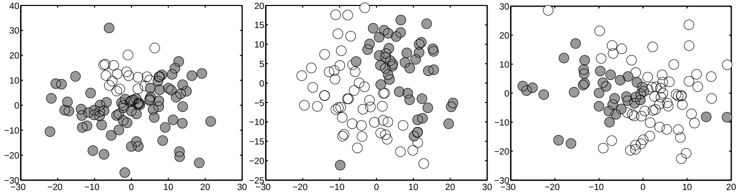
Figure 15 : Predictive probabilities averaged over 1000 Gibbs samples using an RBF covariance. As γ is increased, the model overfits.

Note

- Inference:
  - Gibbs sampling isn't the only option
  - A popular alternative is Variational Bayes

Note 2 – The Generative Process

- Sometimes it's useful to think of the generative process defined by the model.
- In this case, to generate N values of y_n given the associated x_n:
  - Sample $\mathbf{f}$ from a GP with mean $\mathbf{0}$ and Covariance matrix $\mathbf{C}$.
  - For each $n = 1 \ldots N$:
    - Sample $z_n \sim \mathcal{N}(f_n, 1)$
    - If $z_n > 0$ set $y_n = 1$, otherwise $y_n = 0$.
- Some examples:



A more general idea

- Models of this form:
  - $\mathbf{f} \sim GP$
  - $z_n \sim \mathcal{N}(f_n, 1)$
  - $P(y_n|z_n) = \delta(f(z_n))$
- Can be used for more than just binary classification.

A more general idea

- Models of this form:
  - $\mathbf{f} \sim GP$
  - $z_n \sim \mathcal{N}(f_n, 1)$
  - $P(y_n|z_n) = \delta(f(z_n))$
- Can be used for more than just binary classification.
- Ordinal Regression:
  - $P(y_n = k|z_n)$ is now chopped at both ends:

$$P(y_n = k|z_n) = \delta(b_k < z_n < b_{k+1})$$

  - Gibbs distribution for z_n therefore involves a Gaussian truncated at both ends.

A more general idea

- Models of this form:
  - $\mathbf{f} \sim GP$
  - $z_n \sim \mathcal{N}(f_n, 1)$
  - $P(y_n|z_n) = \delta(f(z_n))$
- Can be used for more than just binary classification.
- Ordinal Regression:
  - $P(y_n = k|z_n)$ is now chopped at both ends:

$$P(y_n = k|z_n) = \delta(b_k < z_n < b_{k+1})$$

  - Gibbs distribution for z_n therefore involves a Gaussian truncated at both ends.
- As well as multi-class and semi-supervised classification...

Multi-class classification

- The previous treatment can be extended to multiple classes.
- For a problem with K classes:
  - K GP priors, K N-dimensional latent vectors $\mathbf{f}_k$.
  - $N \times K$ auxiliary variables $z_{nk} \sim \mathcal{N}(f_{nk}, 1)$
  - And:

$$P(y_n = k | z_{n1}, \ldots, z_{nK}) = \delta(z_{nk} > z_{ni} \quad \forall i \neq k)$$

Multi-class classification

- The previous treatment can be extended to multiple classes.
- For a problem with K classes:
  - K GP priors, K N-dimensional latent vectors $\mathbf{f}_k$.
  - $N \times K$ auxiliary variables $z_{nk} \sim \mathcal{N}(f_{nk}, 1)$
  - And:

$$P(y_n = k | z_{n1}, \ldots, z_{nK}) = \delta(z_{nk} > z_{ni} \ \ \forall i \neq k)$$

- Gibbs sampling is similar to the binary case:
  - Only tricky bit is efficently sampling from a K-dimensionl MVG truncated such that the kth element is largest.

Multi-class classification

- The previous treatment can be extended to multiple classes.
- For a problem with K classes:
  - K GP priors, K N-dimensional latent vectors $\mathbf{f}_k$.
  - $N \times K$ auxiliary variables $z_{nk} \sim \mathcal{N}(f_{nk}, 1)$
  - And:

$$P(y_n = k | z_{n1}, \ldots, z_{nK}) = \delta(z_{nk} > z_{ni} \;\; \forall i \neq k)$$

- Gibbs sampling is similar to the binary case:
  - Only tricky bit is efficently sampling from a K-dimensionl MVG truncated such that the kth element is largest.
- Details of a Variational Bayes inference scheme in: Girolami and Rogers 2006

Multi-class Example

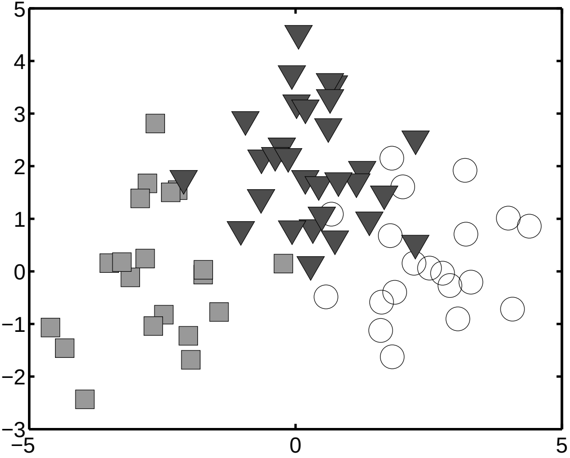


Figure 16 : Multi-class classification example. RBF covariance, $\gamma = 1$.

Multi-class Example

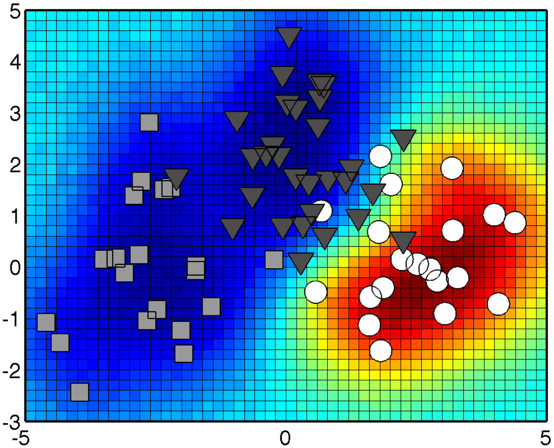


Figure 16 : Multi-class classification example. RBF covariance, $\gamma = 1$.

Multi-class Example

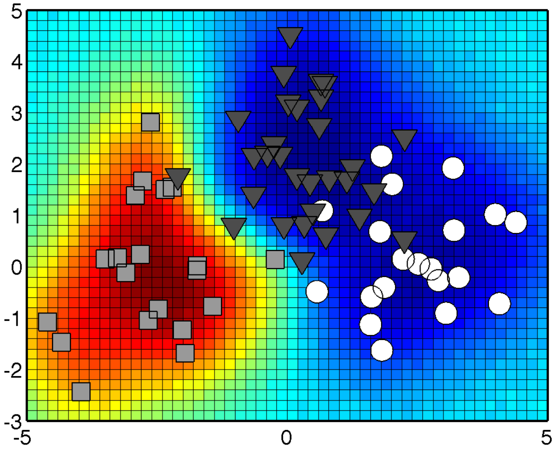


Figure 16 : Multi-class classification example. RBF covariance, $\gamma = 1$.

Multi-class Example

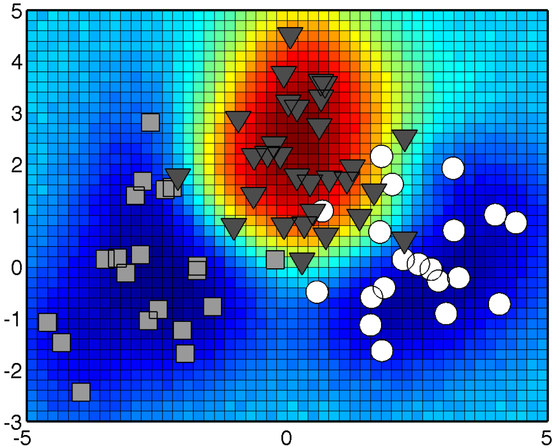


Figure 16 : Multi-class classification example. RBF covariance, $\gamma = 1$.

Semi-supervised Classification

- In some domains, only a subset of data are labeled [e.g. image classification]

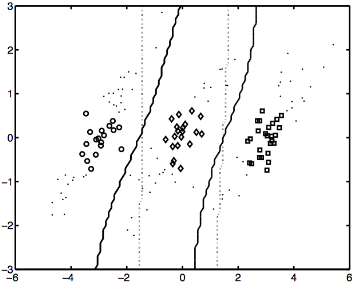


Figure 17 : A toy semi-supervised classification problem.

- Can be overcome using the Null Category Noise Model (NCNM) Lawrence and Jordan 2004

The NCNM

- Going back to binary classification, the auxiliary variable trick can be visualised:

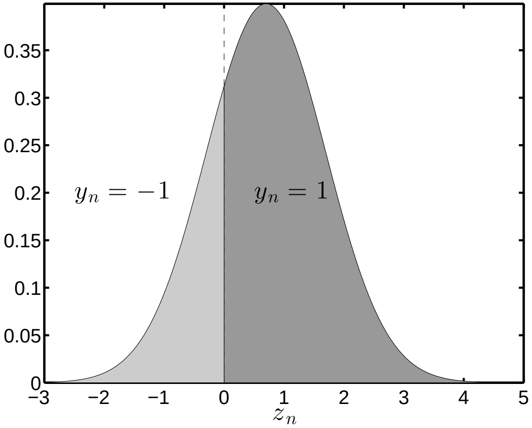


Figure 18 : Visualisation of the auxiliary variable trick. The Gaussian has mean f_n. Note that I'm not calling the classes ± 1.

The NCNM

- To include unlabeled data, we add a third category, for $y_n = 0$:

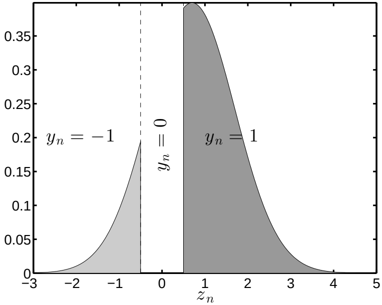


Figure 19 : Visualisation of the NCNM with a null region of width 1.

$$p(y_n|z_n) = \begin{cases} \delta(z_n < -a) & y_n = -1 \\ \delta(z_n > a) & y_n = 1 \\ \delta(z_n > -a) - \delta(z_n > a) & y_n = 0 \end{cases}$$

The NCNM

- The final step is to introduce another set of latent variables.
  - $g_n = 0$ if y_n is observed (i.e. labeled) and $g_n = 1$ otherwise.
- And enforce the constraint that no unlabeled points can exist in the null region:

$$P(y_n = 0 | g_n = 1) = 0$$

The NCNM

- The final step is to introduce another set of latent variables.
  - $g_n = 0$ if y_n is observed (i.e. labeled) and $g_n = 1$ otherwise.
- And enforce the constraint that no unlabeled points can exist in the null region:

$$P(y_n = 0 | g_n = 1) = 0$$

- This has the effect of introducing an empty region around the decision boundary
  - i.e. pushing the decision boundary into regions of empty space

The NCNM

- The final step is to introduce another set of latent variables.
  - $g_n = 0$ if y_n is observed (i.e. labeled) and $g_n = 1$ otherwise.
- And enforce the constraint that no unlabeled points can exist in the null region:

$$P(y_n = 0 | g_n = 1) = 0$$

- This has the effect of introducing an empty region around the decision boundary
  - i.e. pushing the decision boundary into regions of empty space
- Inference:
  - Gibbs sampling is the same as the binary case except $z_n | f_n, g_n = 1$.
  - This is a mixture of two truncated Gaussians – sample the component, and then sample z_n.

NCNM Example

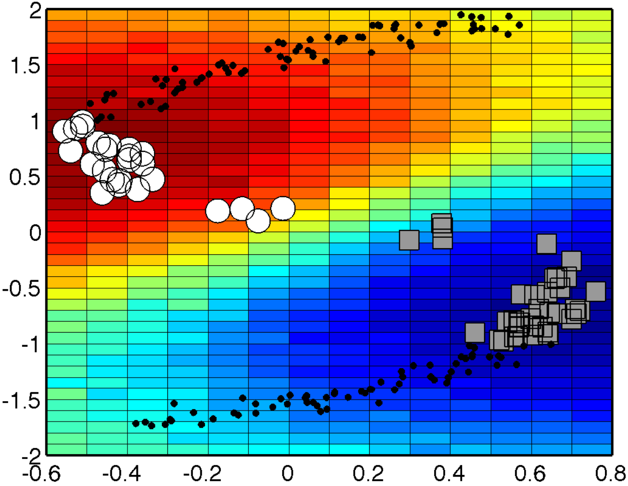


Figure 20 : Standard GP classification (unlabeled data ignored)

NCNM Example

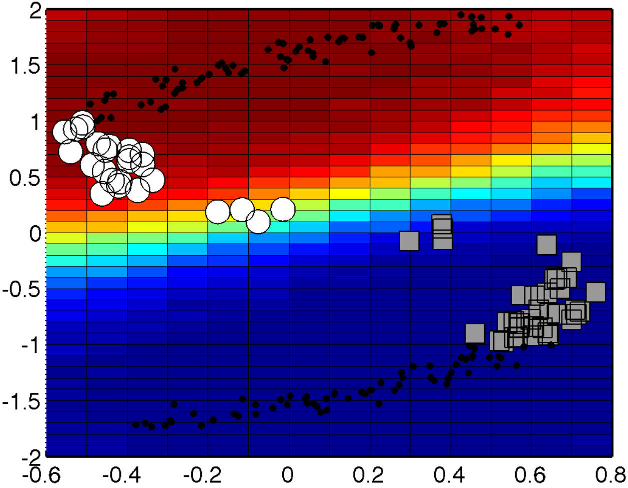
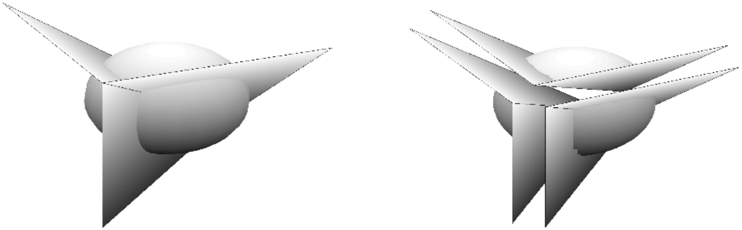


Figure 21 : NCNM GP classification

Multi-class NCNM

- This idea can be extended to the multi-class setting.
- See Rogers and Girolami 2007

$$P(y_n = k | z_{n1}, \ldots, z_{nK}) = \begin{cases} \delta(z_{nk} > z_{ni} + a \quad \forall i \neq k) & y_n > 0 \\ 1 - \sum_j \delta(z_{nj} > z_{ni} + a \quad \forall i \neq j) & y_n = 0 \end{cases}$$



(a) A visualisation of the truncation caused by the standard multi-class probit model

(b) A visualisation of the truncation caused by the multi-class probit model with a null region

Figure 22 : Visualisation of truncation

Summary

- GP priors aren't restricted to regression.
- Analytical solutions aren't possible
- Auxiiliary Variable Trick makes inference (via Gibbs sampling or Variational Bayes) straightforward for:
  - Binary classification
  - Ordinal regression
  - Multi-class classification
  - Semi-supervised classification (binary and mutli-class)
  - As well as others (e.g. binary PCA)

Lecture 5: Application: Clinical Ratings

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

April 30, 2014

Clinicians disagree in AandE

- Patients in Accident and Emergency (A&E) are continually monitored.
  - Heart rate
  - Blood pressure
  - Temperature
  - etc

Clinicians disagree in AandE

- Patients in A&E are continually monitored.
  - Heart rate
  - Blood pressure
  - Temperature
  - etc
- Based on these hourly observations, clinicians (in a Glasgow hospital) give each patient an ordinal rating
  - A (healthy(ish)), B, C, D, E, F (critical)

Clinicians disagree in AandE

- Patients in A&E are continually monitored.
  - Heart rate
  - Blood pressure
  - Temperature
  - etc
- Based on these hourly observations, clinicians (in a Glasgow hospital) give each patient an ordinal rating
  - A (healthy(ish)), B, C, D, E, F (critical)
- These ratings are *subjective*
  - How do clinicians disagree? (variance? bias?)
- More details of this work in Rogers et al 2013 and Rogers et al 2010

Data

- $c = 1 \dots C$ clinicians.
- $p = 1 \dots P$ patients.
- For patient p, we have T_p observations at times $\mathbf{t}_p = [t_{p1}, \dots, t_{pT_p}]^T$.
- $y^p_{\tau c}$ is rating at $t_{p\tau}$ ($\{A, B, C, D, E\}$).
- The model:
  - Patient health is modelled as a GP.
  - Each clinician has their own offset and precision used to *corrupt* the health function (q).
  - q is binned to produce rating.

Model

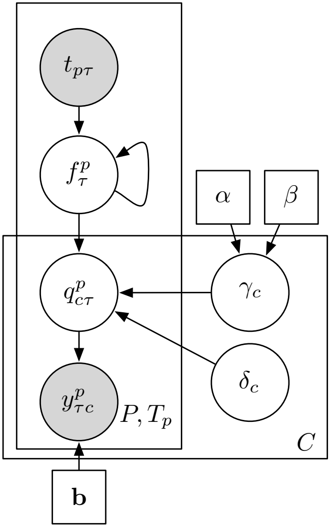


Figure 23 : Plates diagram

$$
\begin{aligned}
\mathbf{f}^p &\sim \mathcal{N}(\mathbf{0}, \mathbf{C}^p) \quad \text{[health]} \\
\delta_c &\sim \mathcal{N}(0,1) \quad \text{[offset]} \\
\gamma_c &\sim \mathcal{G}(\alpha, \beta) \quad \text{[precision]} \\
q^p_{c\tau} &\sim \mathcal{N}(f^p_\tau + \delta_c, \gamma_c^{-1}) \\
P(y^p_{\tau c} = k) &= \delta(b_k < q^p_{c\tau} < b_{k+1})
\end{aligned}
$$

Previously auxiliary variables were $z_n \sim \mathcal{N}(f_n, 1)$. This model adds clinician-specific offsets and precisions: $q^p_{c\tau} \sim \mathcal{N}(f^p_\tau + \delta_c, \gamma_c^{-1})$.

Example data generation

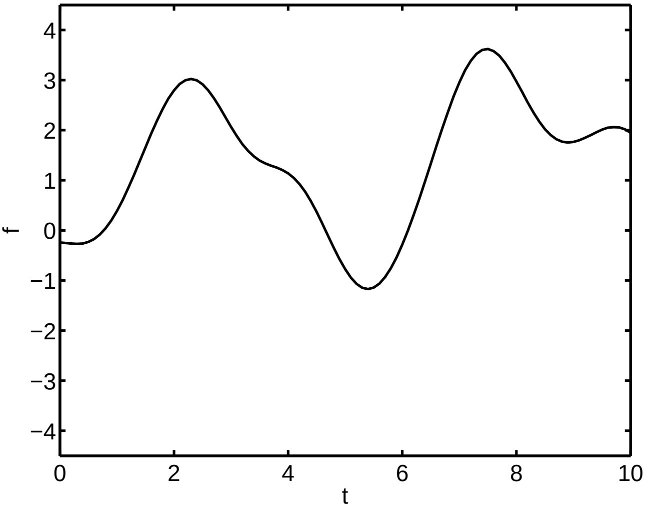


Figure 24 : Example of the generative process described by the model for three clinicans.

Example data generation

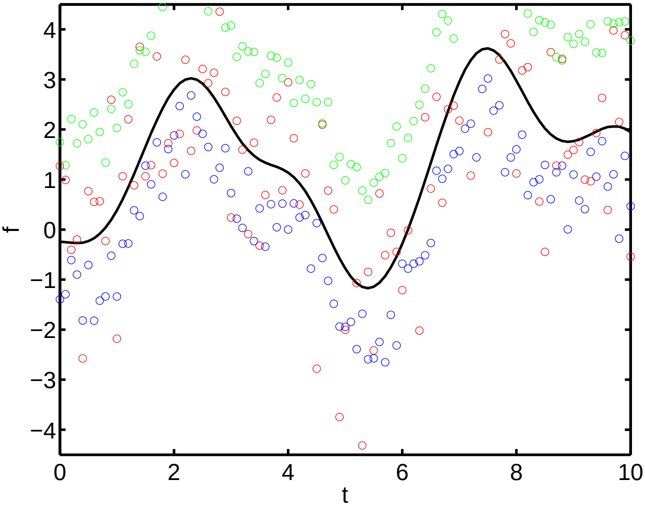


Figure 24 : Example of the generative process described by the model for three clinicans.

Example data generation

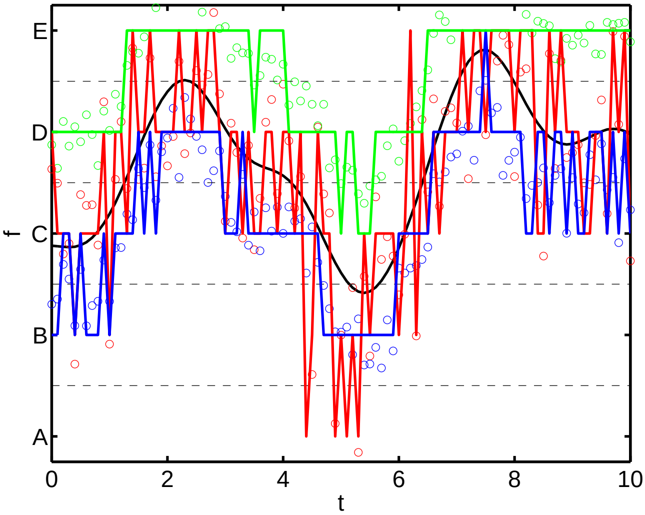


Figure 24 : Example of the generative process described by the model for three clinicans.

Model inference

- Gibbs sampling is straightforward
- We sample:
  - The latent health function for each patient.
  - The auxiliary variables.
  - The offset and precision for each clinician.

Model inference

- Gibbs sampling is straightforward
- We sample:
  - The latent health function for each patient.
  - The auxiliary variables.
  - The offset and precision for each clinician.
- The offset and precision tell us how the clinicians disagree.

Model inference

- Gibbs sampling is straightforward
- We sample:
  - The latent health function for each patient.
  - The auxiliary variables.
  - The offset and precision for each clinician.
- The offset and precision tell us how the clinicians disagree.
- Identifiability: offset for one clinician fixed to 0.

Results

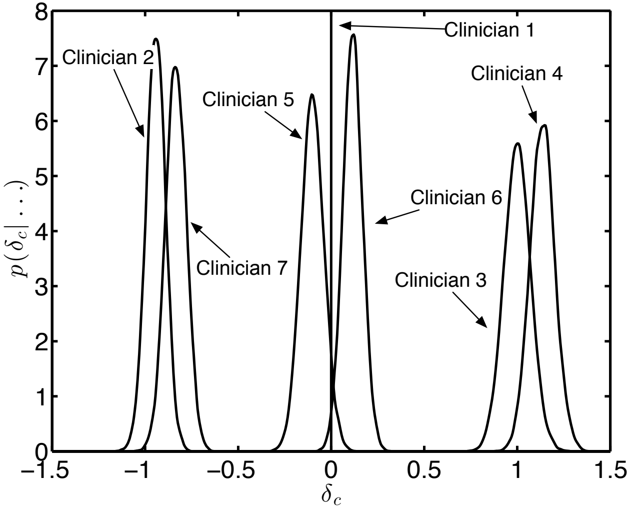
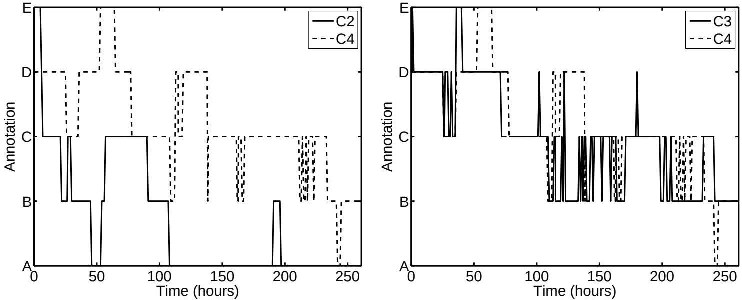


Figure 25 : Marginal offset posteriors

Results



(a) Clinicians 2 and 4 (b) Clinicians 3 and 4

Figure 26 : Inferred offsets make sense on inspection of the data.

Results

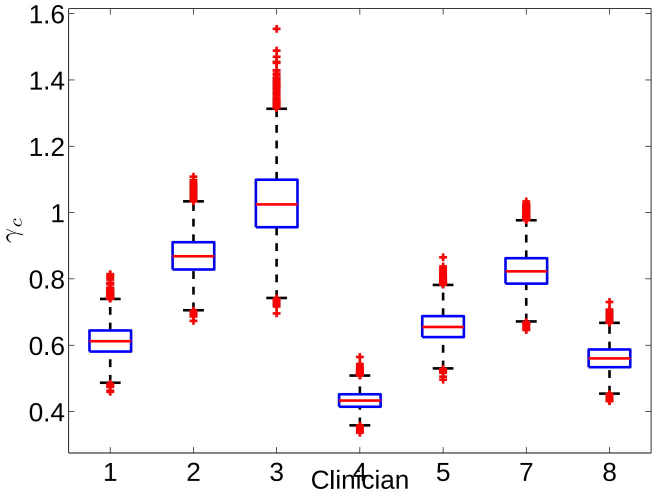
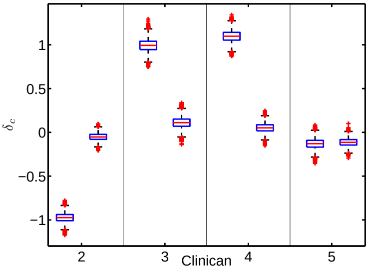


Figure 27 : Marginal precision posteriors. Clinicians 1, 4, and 8 appear to be the least consistent (wrt the majority)

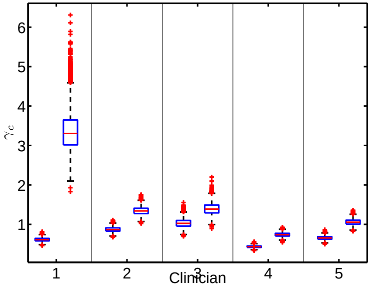
INSIGHT

- After the initial annotation, clinicians went through INSIGHT procedure.
- The goal was to make ratings more consistent.
- If it succeeded, we should see a recuction in offset and increase in precision in the post-INSIGHT data.

Post-INSIGHT results



(a) Offsets

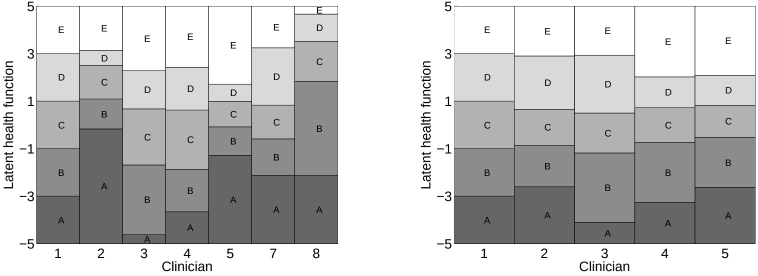


(b) Precisions

Figure 28 : Offsets and precision before and after INSIGHT. Offsets get closer to 0, whilst precision increase suggesting greater agreement amongsth clinicians.

Inferring category boundaries

- So far, it has been assumed that all categories are the same size (i.e. the elements of **b** are equally spaced).
- We can also infer these (with fixed end-points and $\delta_c = 0$).
- Removes uniform prior assumption over categories.



(a) Before INSIGHT (b) After INSIGHT

Figure 29 : Posterior mean cateogory boundaries.

Summary and Conclusions

- Model allows us to:
  - learn something about *how* clinicians disagree and how they rate.
  - assess the effectiveness of the INSIGHT procedure.

Summary and Conclusions

- Model allows us to:
  - learn something about *how* clinicians disagree and how they rate.
  - assess the effectiveness of the INSIGHT procedure.
- GP prior:
  - Flexible
  - Required no parametric assumptions about health function
  - Hyper-parameter (γ) was inferred in the model (could be patient-specific)

Summary and Conclusions

- Model allows us to:
  - learn something about *how* clinicians disagree and how they rate.
  - assess the effectiveness of the INSIGHT procedure.
- GP prior:
  - Flexible
  - Required no parametric assumptions about health function
  - Hyper-parameter (γ) was inferred in the model (could be patient-specific)
- Auxiliary Variable Trick:
  - Not restricted to a standard Gaussian centered on the GP variable.
  - Incorporated offset and precision without causing additional inference challenges.