Gaussian Processes for NLP

Trevor Cohn & Daniel Beck

CIS, University of Melbourne
DCS, University of Sheffield

ALTA Workshop
26 November 2014

with slides from Daniel Preoţiuc-Pietro, Neil Lawrence, Richard Turner

Gaussian Processes

Brings together several key ideas in one framework

- Bayesian
- kernelised
- non-parametric
- non-linear
- modelling uncertainty

Elegant and powerful framework, with growing popularity in machine learning and application domains.

Gaussian Processes

State of the art for regression

- exact posterior inference
- supports very complex non-linear functions
- elegant model selection

Gaussian Processes

Now mature enough for use in NLP

- support for classification, ranking, etc
- fancy kernels, e.g., text
- sparse approximations for large scale inference
- probabilistic formulation allows incorporation into larger graphical models
- models the prediction uncertainty so that it's propagated through pipelines of probabilistic components

Tutorial Scope

Covers

1. GP fundamentals (1 hour)
   - focus on regression
   - weight space vs. function space view
   - squared exponential kernel
2. NLP applications (1 hour 30)
   - sparse GPs
   - periodic kernels and model selection
   - multi-output GPs
3. Further topics (30 mins)
   - classification and other likelihoods
   - structured kernels

Outline

Gaussian Processes

Regression and classification are fundamental techniques in NLP

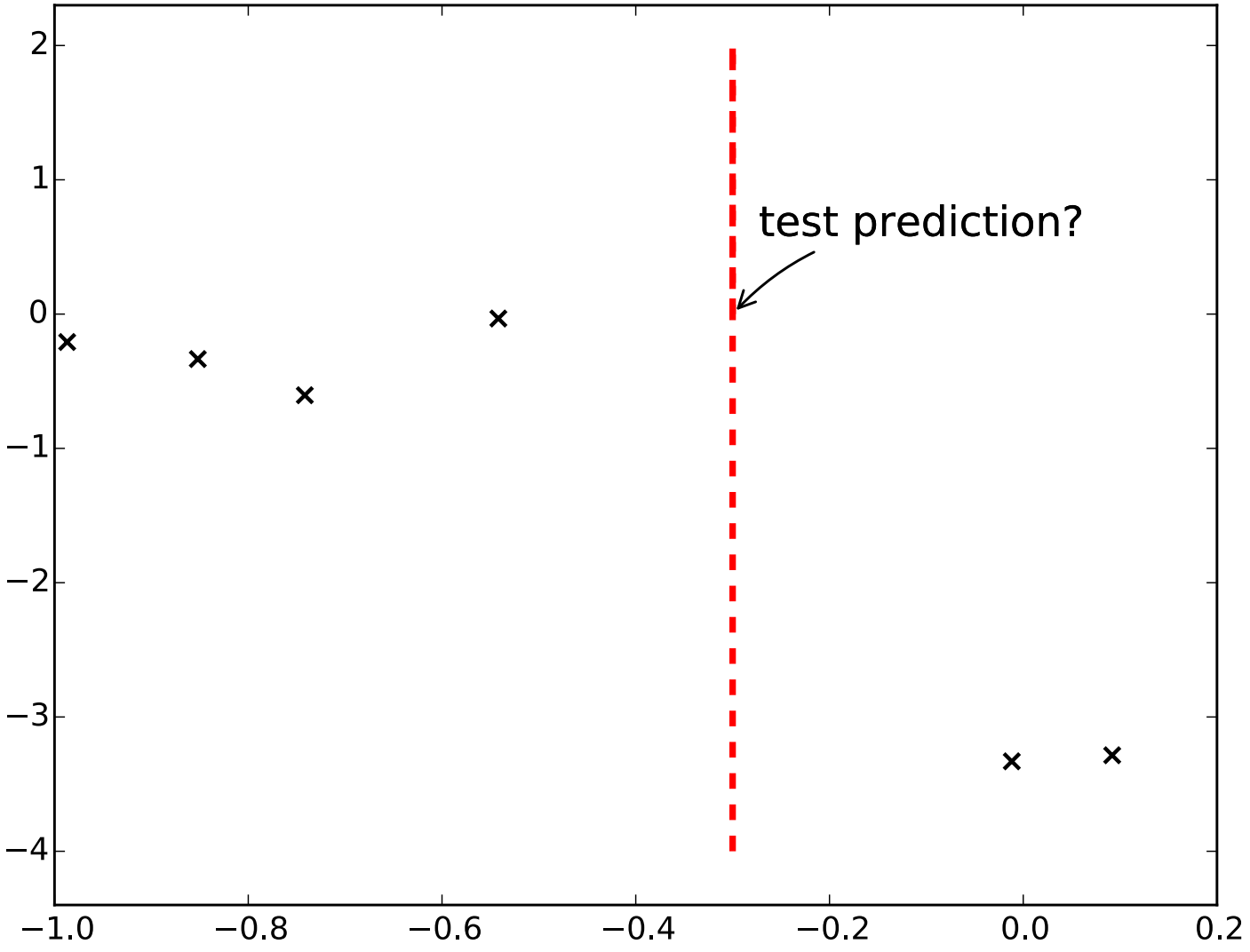
Linear models:

- simple and fast to run
- provide straightforward interpretability
- but very limited in the type of relations they can model

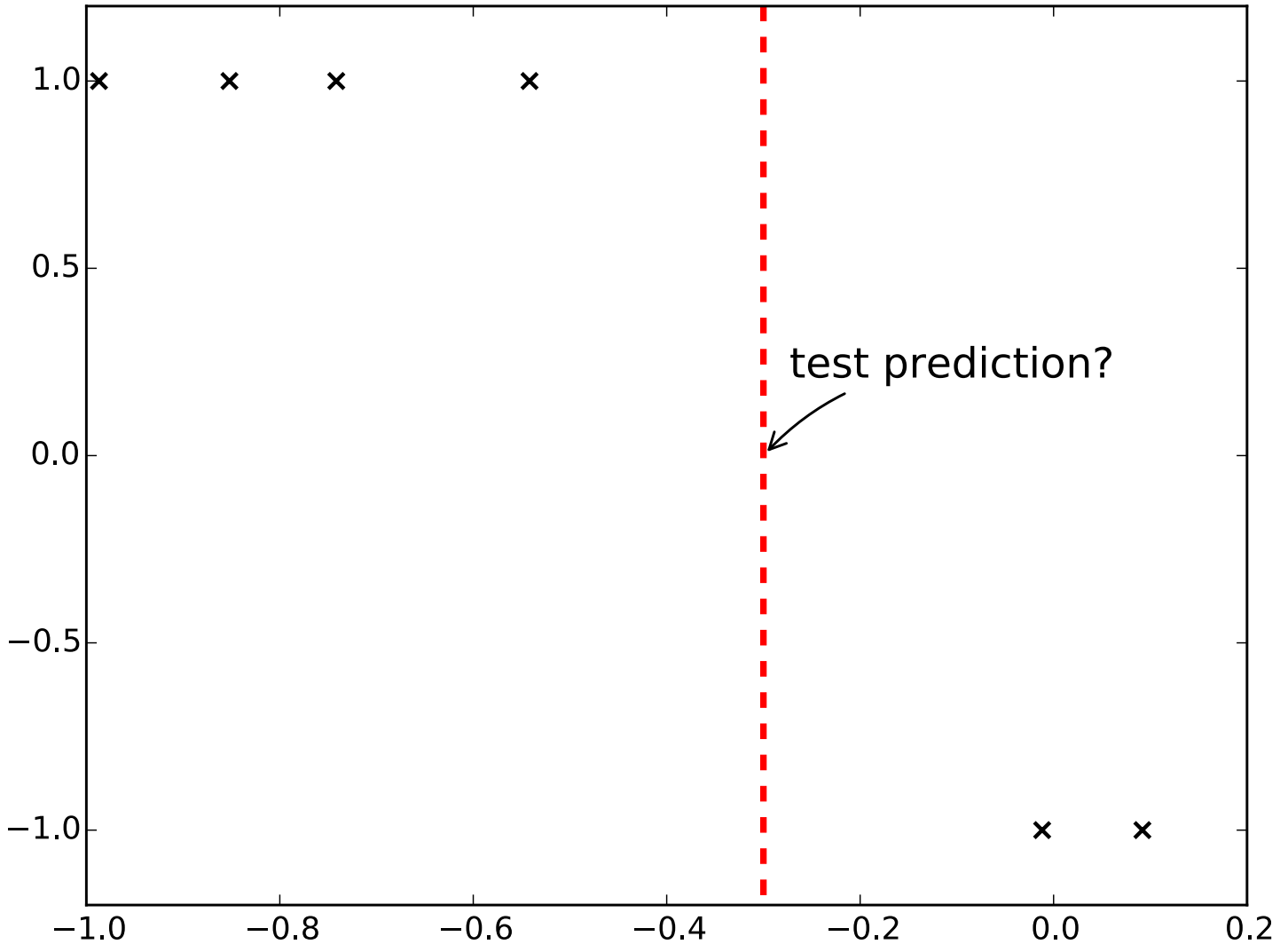
Non-linear models (e.g. SVM):

- better results because they allow to model other relationships
- but high cost of hyperparameter optimisation
- lack of interoperability with downstream processing

Regression



. . . versus classification



The Gaussian (normal) distribution

The Gaussian is probably the most widely used probability density

$$p(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$
$$\triangleq \mathcal{N}(\mu,\sigma^2)$$

The Gaussian (normal) distribution

The Gaussian is probably the most widely used probability density

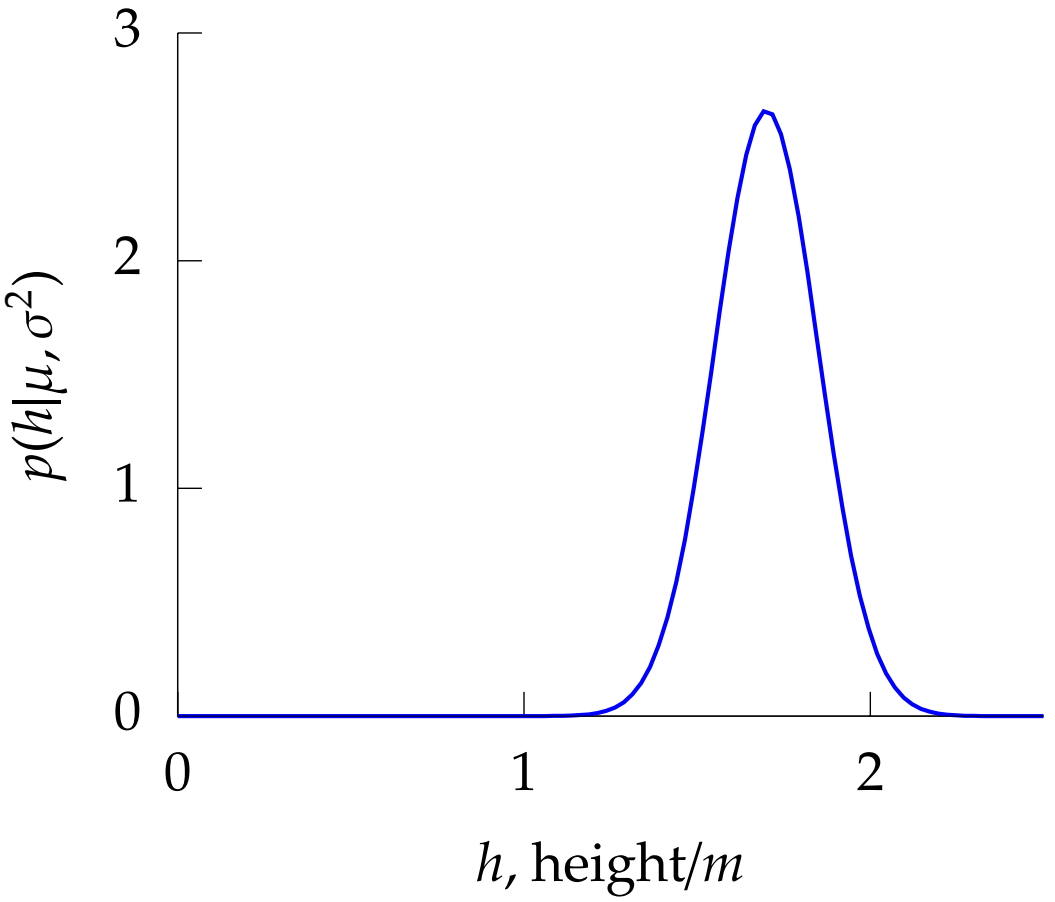
$$p(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$
$$\triangleq \mathcal{N}(\mu,\sigma^2)$$

Multivariate formulation

$$p(\mathbf{y}|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mu)^\top\Sigma^{-1}(\mathbf{y}-\mu)\right)$$

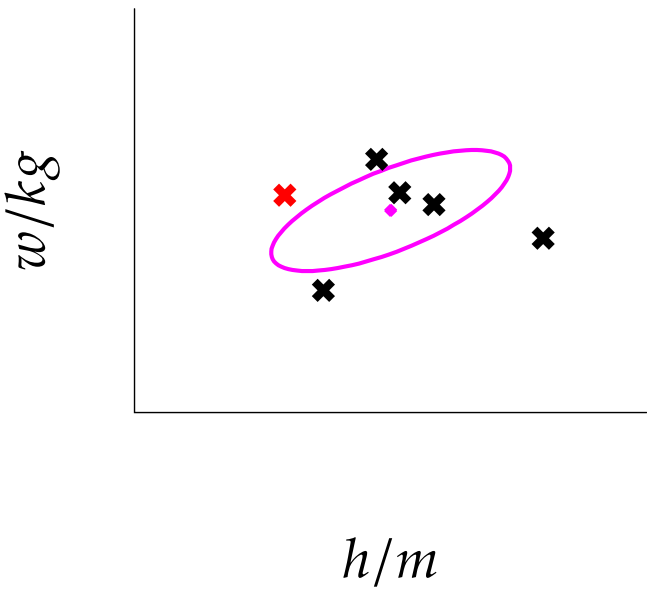
where μ is now a mean vector, and Σ a covariance matrix.

The Gaussian distribution

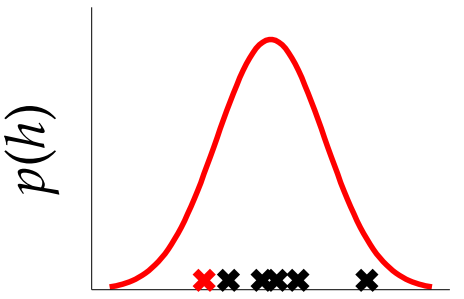


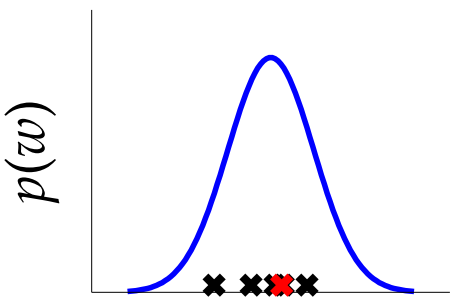
Bivariate Gaussian distribution

Joint Distribution



Marginal Distributions





Gaussian distribution properties

Several very convenient properties:

- Scaling a Gaussian results in a Gaussian
- Sum of Gaussian variables is also Gaussian (generally, central limit theorem)
- Product of two Gaussians is (scaled) Gaussian

For multi-variate Gaussian distributed variables:

- Marginal distribution is also Gaussian
- Conditional distribution is also Gaussian

Gaussian Processes

The Gaussian distribution:

- is a distribution over scalars or vectors

Gaussian Processes

The Gaussian distribution:

- is a distribution over scalars or vectors

The Gaussian process:

- is a distribution over functions

Being Bayesian

Revisiting our regression example:

- what class of functions should we allow?
- given the class what parameter values are sensible?
- e.g., linear should we let $m, c \to \pm\infty$?
- e.g., non-linear should be jagged or smooth?
- e.g., periodic . . .

Usually have some intuitions.

Being Bayesian

Define a prior over the unknowns, $p(\theta)$.

- encodes our intitial intuitions about reasonable values

Observing data gives rise to a likelihood, $p(y|\theta)$.

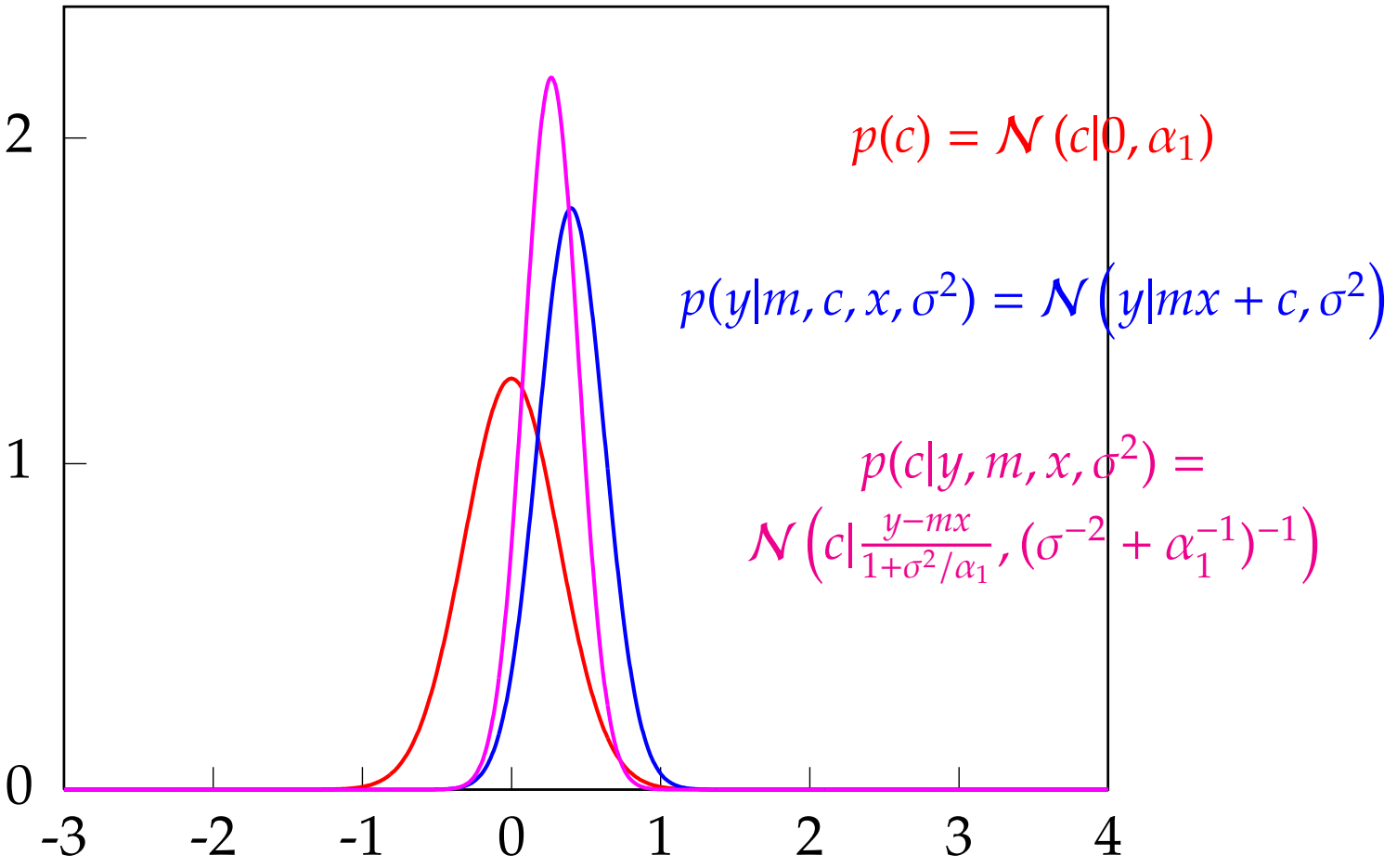
- how well the model fits the training sample

Seek to reason over the posterior

- incorporating the above to form our updated beliefs about the unknowns
- formulated using *Bayes' rule*

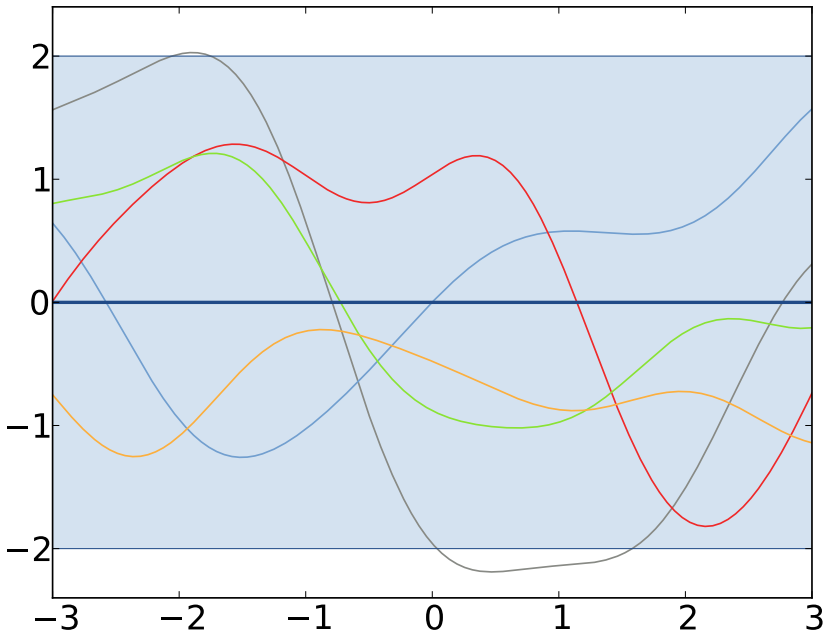
$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

Bayesian update illustrated



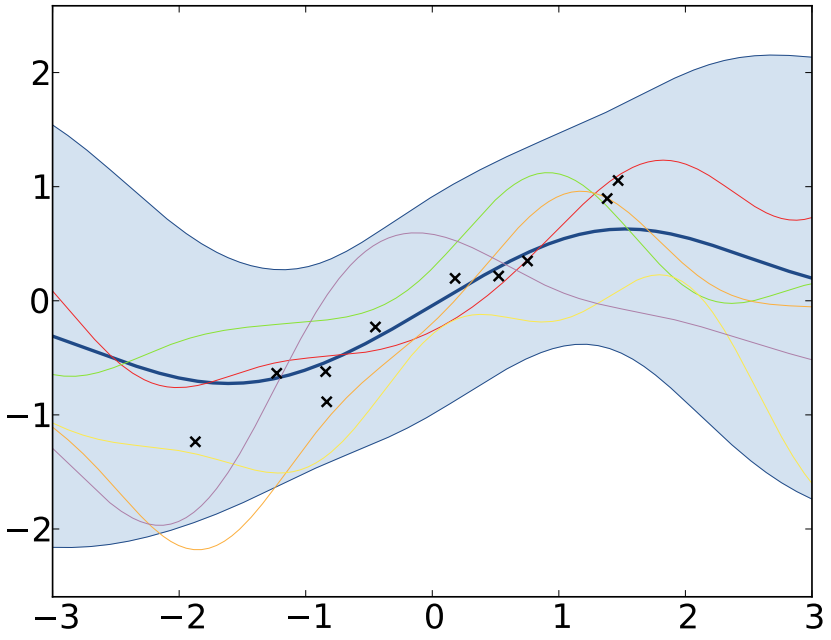
Example of linear regression using prior over intercept parameter, c.

Gaussian Processes prior



- the prior represents our prior beliefs over the functions we expect to have generated our (regression) data
- here, standard choices of 0 mean and 1 variance, and a smoothly varying function

Gaussian Process posterior



- after observing data points posterior now incorporates data likelihood
- the functions must pass close to the observations

Relationship to Linear regression

Linear regression:

$$y = \mathbf{w}^\top \mathbf{x} + \epsilon$$

where w are the learned model parameters.

Gaussian process regression:

$$y = f(\mathbf{x}) + \epsilon$$

where function f is drawn from a Gaussian process prior.

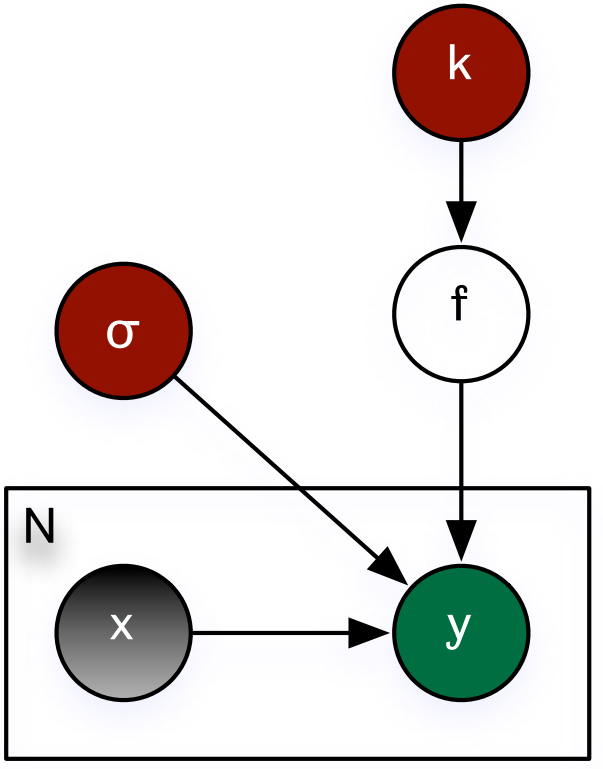
- the $\mathbf{w}$ parameters are gone (in fact, integrated out)
- f is not limited to a parametric form

Graphical model view

$$f \sim \mathcal{GP}(m, k)$$

$$y \sim \mathcal{N}(f(\mathbf{x}), \sigma^2)$$

- $f : \mathcal{R}^D \rightarrow \mathcal{R}$ is a latent function
- y is a noisy realisation of $f(\mathbf{x})$
- k is the covariance function or kernel
- σ^2 is a learned parameter



Formal definition

Definition

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

A GP is fully specified by mean $m(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$ functions:

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

Without loss of generality we can assume $m(\mathbf{x}) = \mathbf{0}$.

Compared to a Gaussian distribution

Gaussian distribution:

- specified by a mean and covariance matrix: $x \sim \mathcal{N}(\mu, \Sigma)$
- the vector index is the position of the random variables x

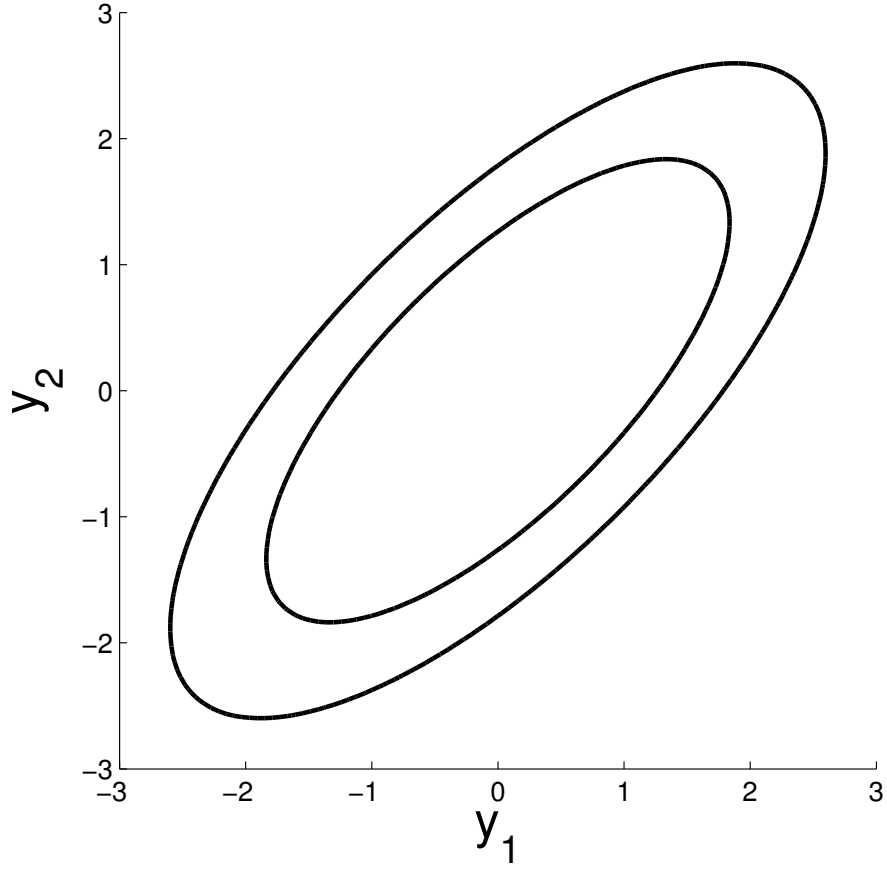
Gaussian process:

- specified by a mean and covariance function: $f \sim \mathcal{GP}(m, k)$
- the index is the argument x of f

The Gaussian Process is an infinite extension of multivariate Gaussian distributions

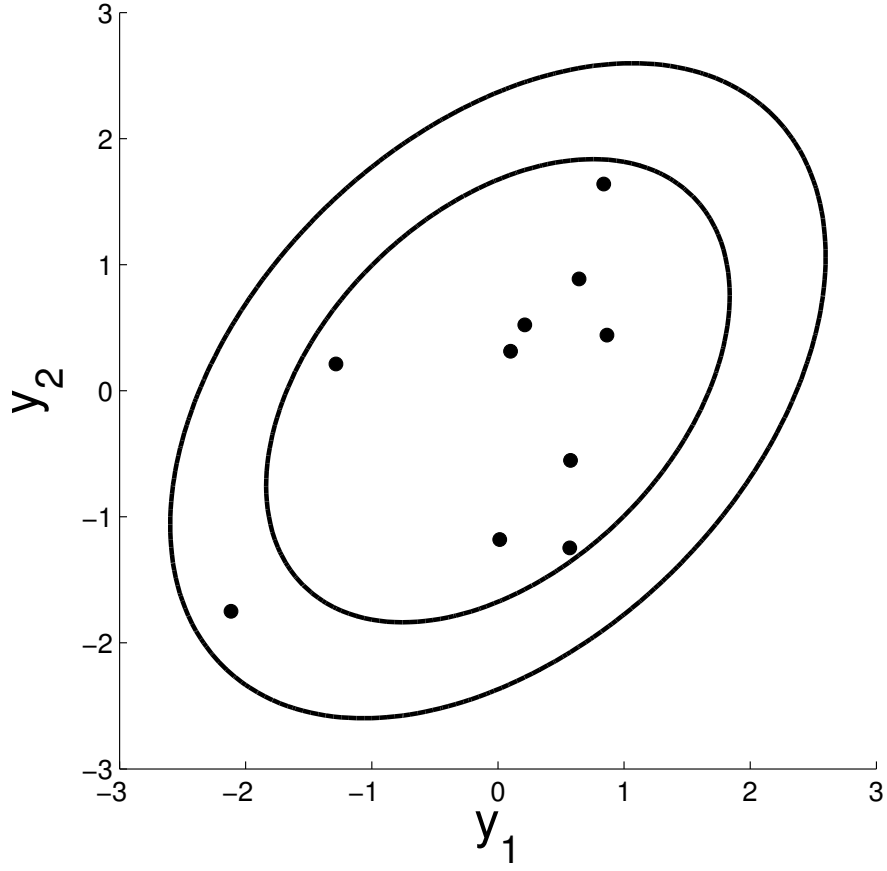
Gaussian distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^\intercal \Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .7 \\ .7 & 1 \end{bmatrix}$$



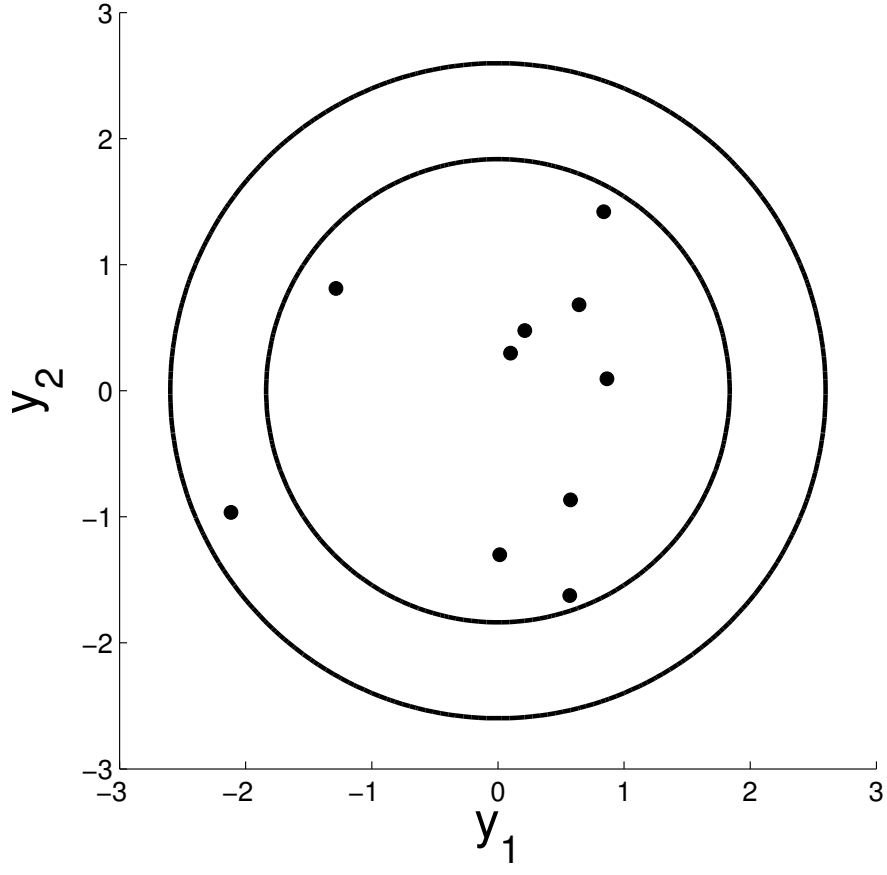
Gaussian distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^\mathsf{T}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .4 \\ .4 & 1 \end{bmatrix}$$



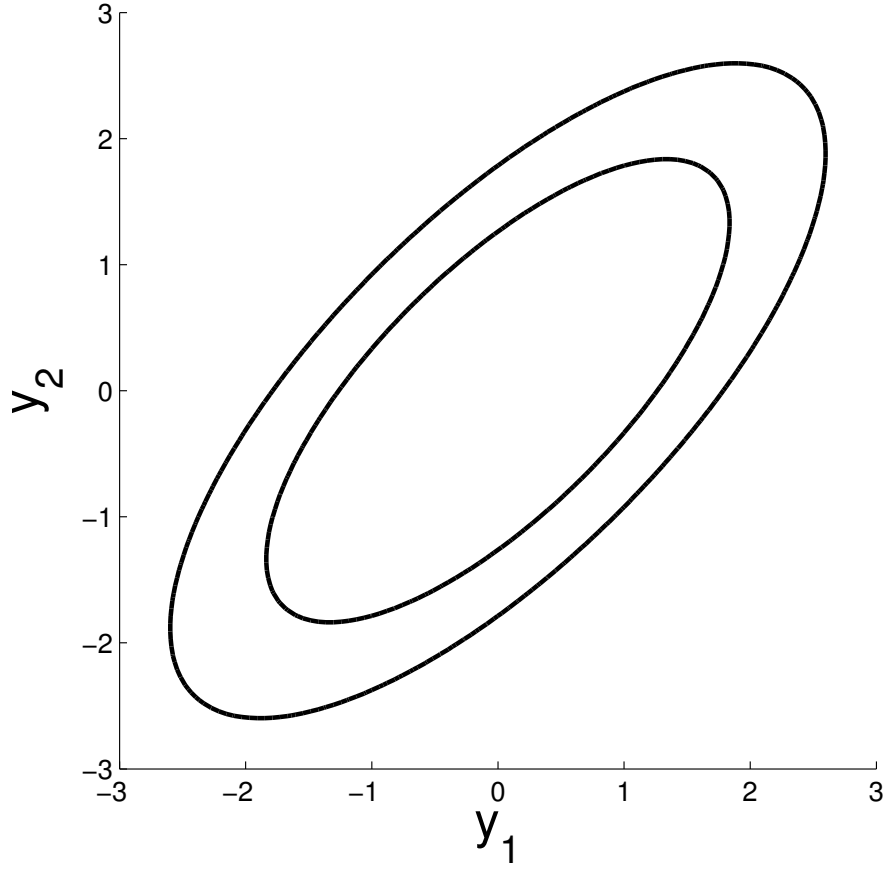
Gaussian distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



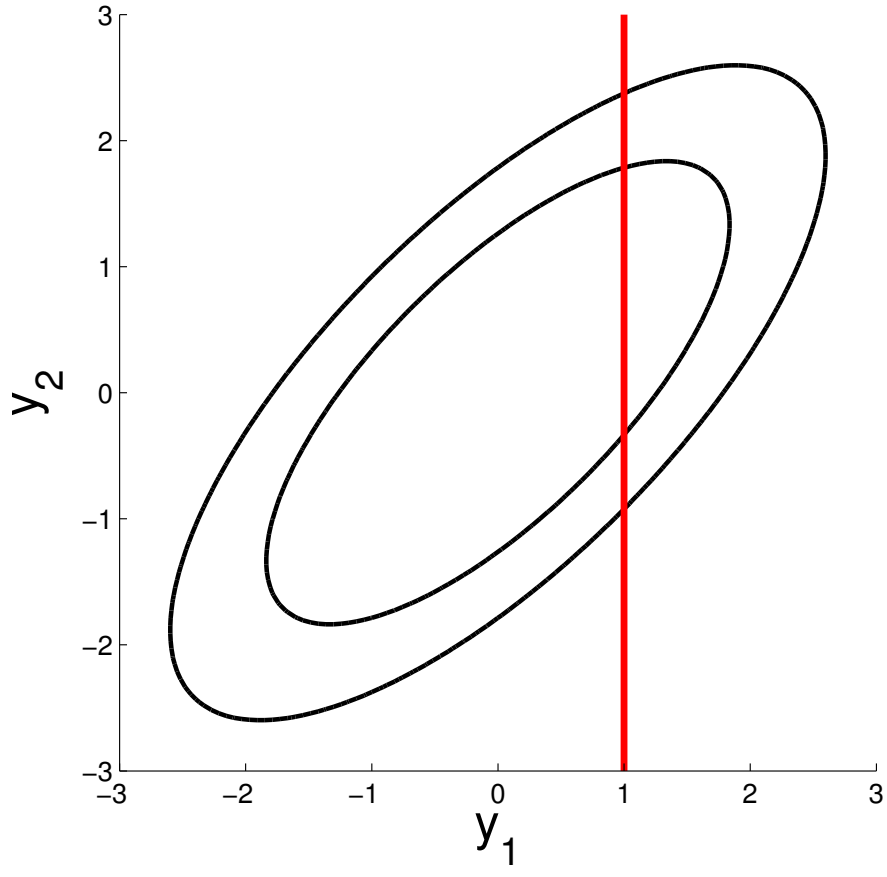
Gaussian distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^\intercal \Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .7 \\ .7 & 1 \end{bmatrix}$$



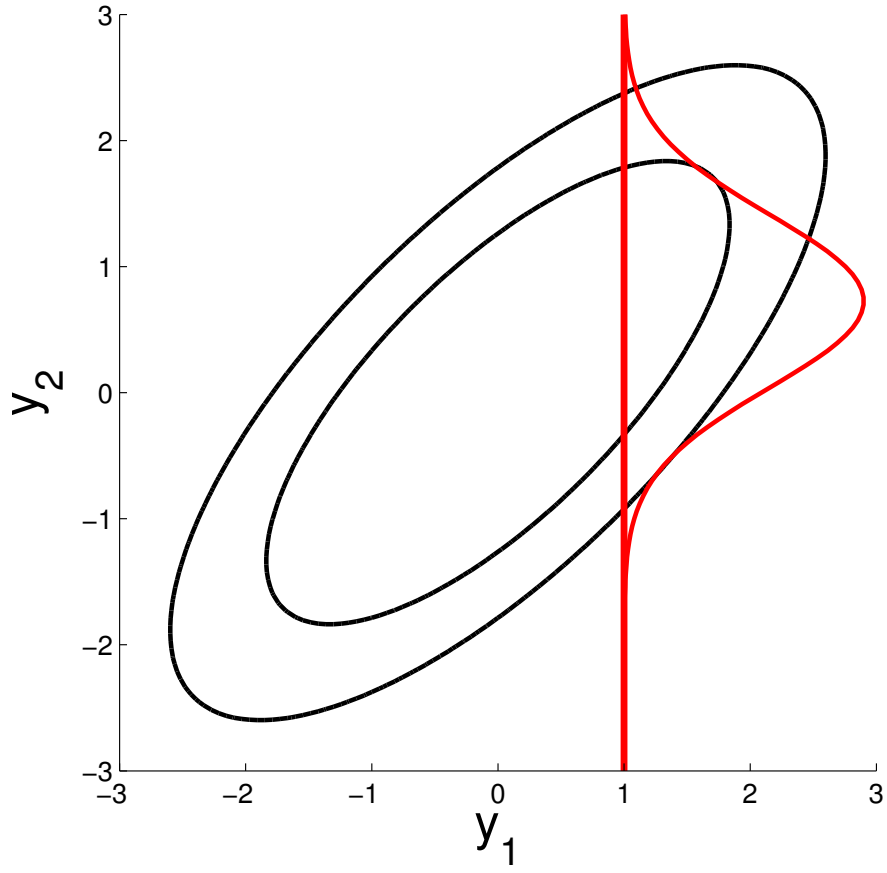
Gaussian distribution

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^\mathsf{T}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .7 \\ .7 & 1 \end{bmatrix}$$



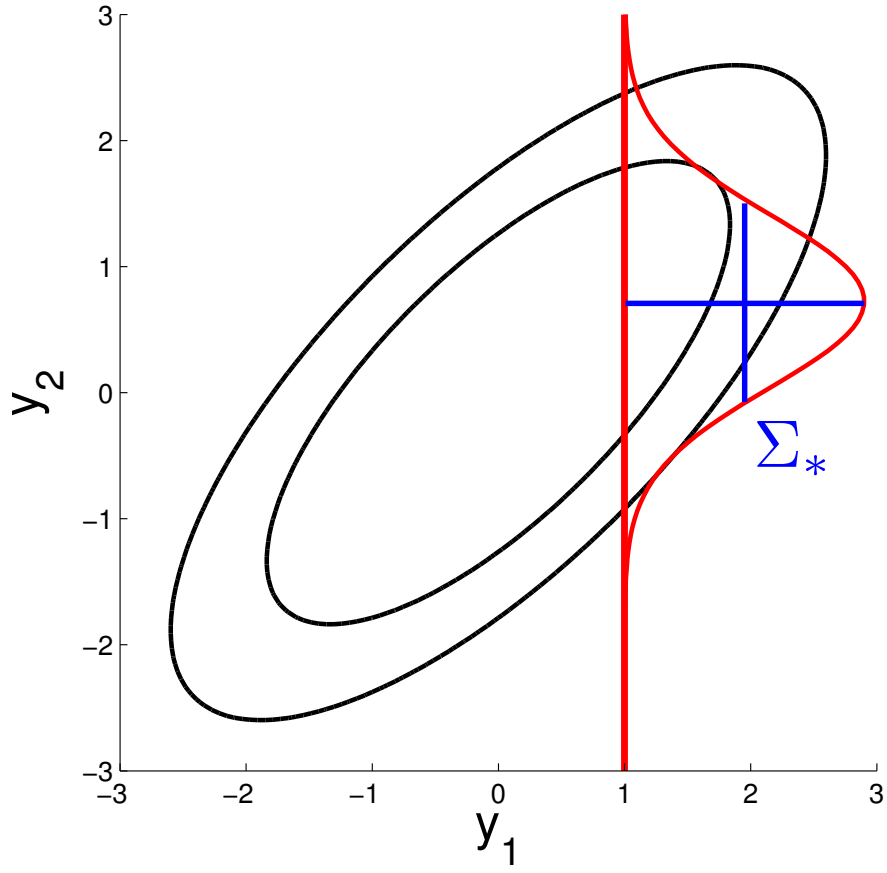
Gaussian distribution

$$p(\mathsf{y}_2|\mathsf{y}_1, \Sigma) \propto \exp\left(-\tfrac{1}{2}(\mathsf{y}_2 - \mu_*){\Sigma_*}^{-1}(\mathsf{y}_2 - \mu_*)\right)$$



Gaussian distribution

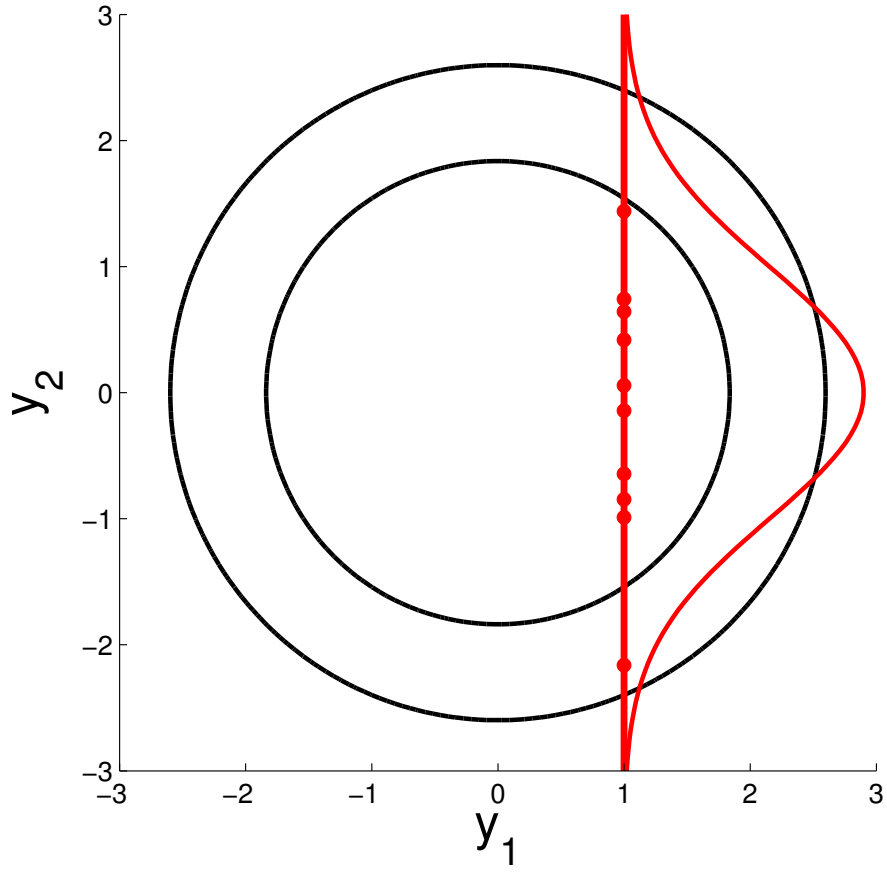
$$p(\mathrm{y}_2|\mathrm{y}_1, \Sigma) \propto \exp\left(-\tfrac{1}{2}(\mathrm{y}_2 - \mu_*){\Sigma_*}^{-1}(\mathrm{y}_2 - \mu_*)\right)$$



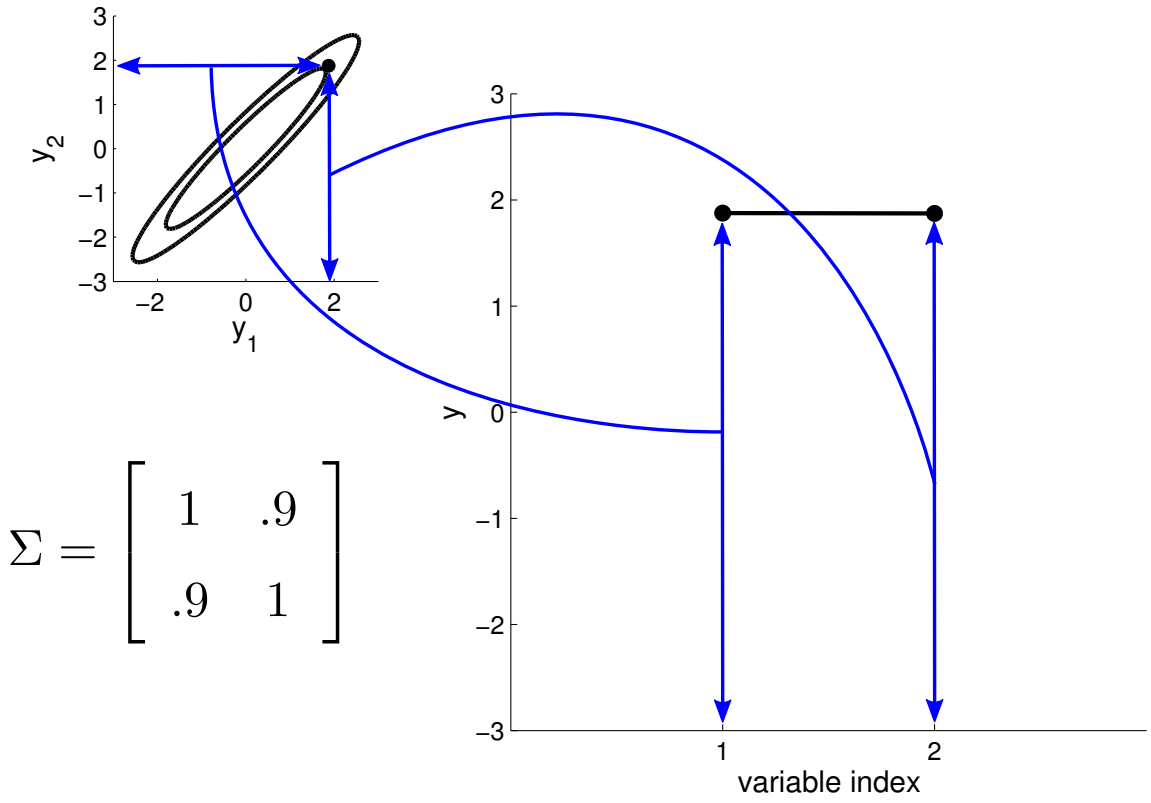
$$\mu_* = \mathrm{W}\mathrm{y}_1$$

Gaussian distribution

$$p(\mathsf{y}_2|\mathsf{y}_1, \Sigma) \propto \exp\left(-\tfrac{1}{2}(\mathsf{y}_2 - \mu_*)\Sigma_*{}^{-1}(\mathsf{y}_2 - \mu_*)\right)$$

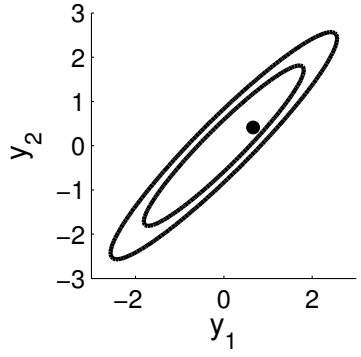


New visualisation

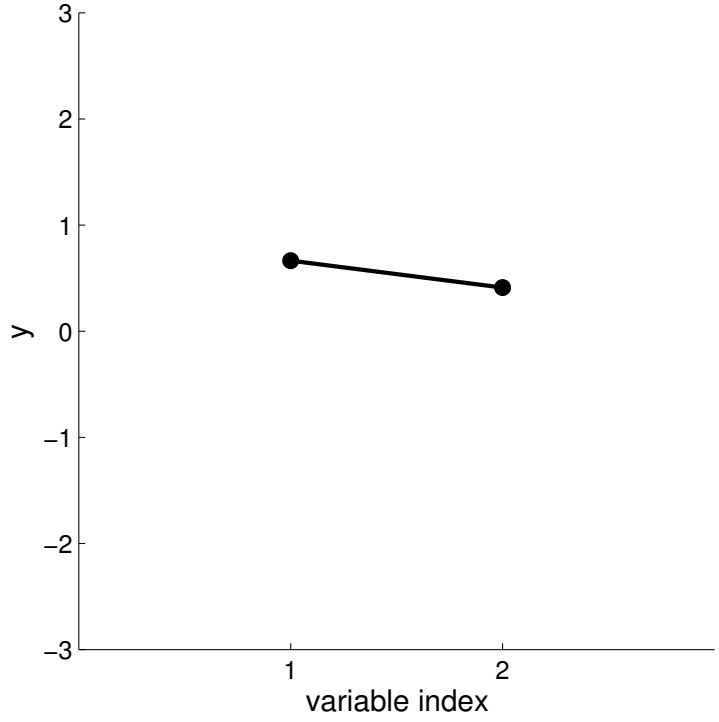


$$\Sigma = \begin{bmatrix} 1 & .9 \\ .9 & 1 \end{bmatrix}$$

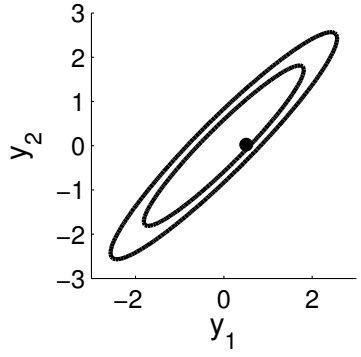
New visualisation



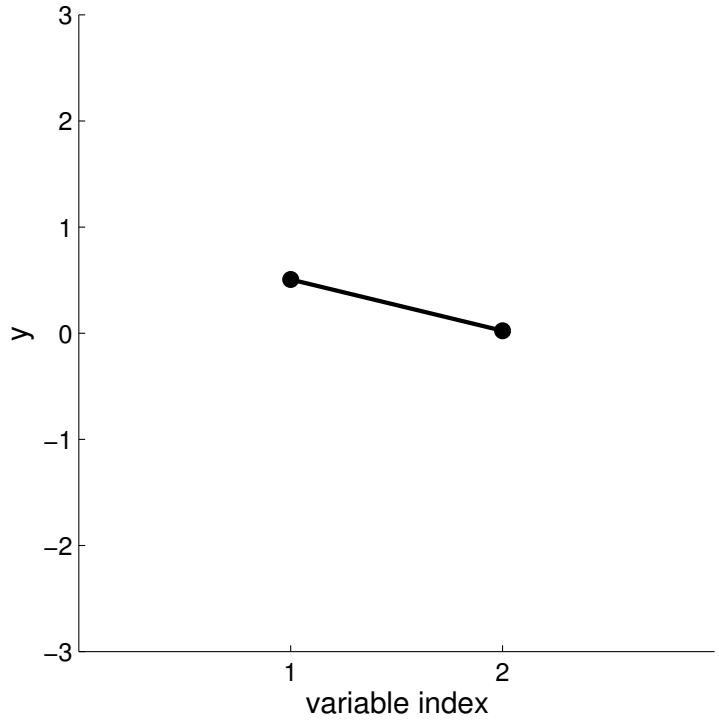
$$\Sigma = \begin{bmatrix} 1 & .9 \\ .9 & 1 \end{bmatrix}$$



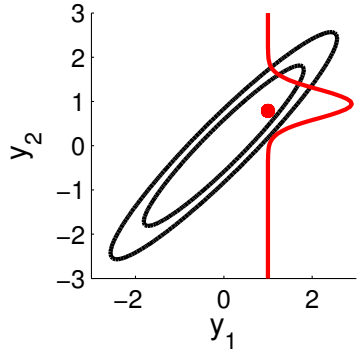
New visualisation



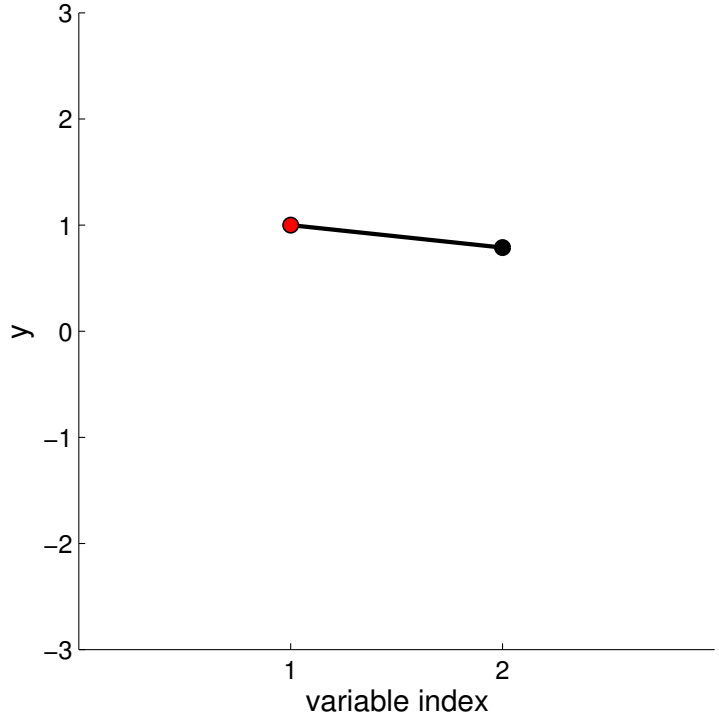
$$\Sigma = \begin{bmatrix} 1 & .9 \\ .9 & 1 \end{bmatrix}$$



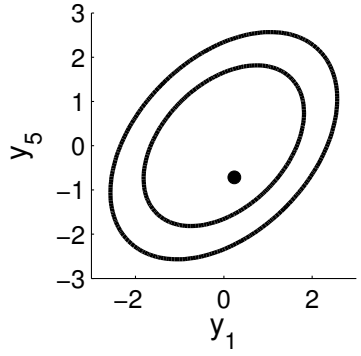
New visualisation

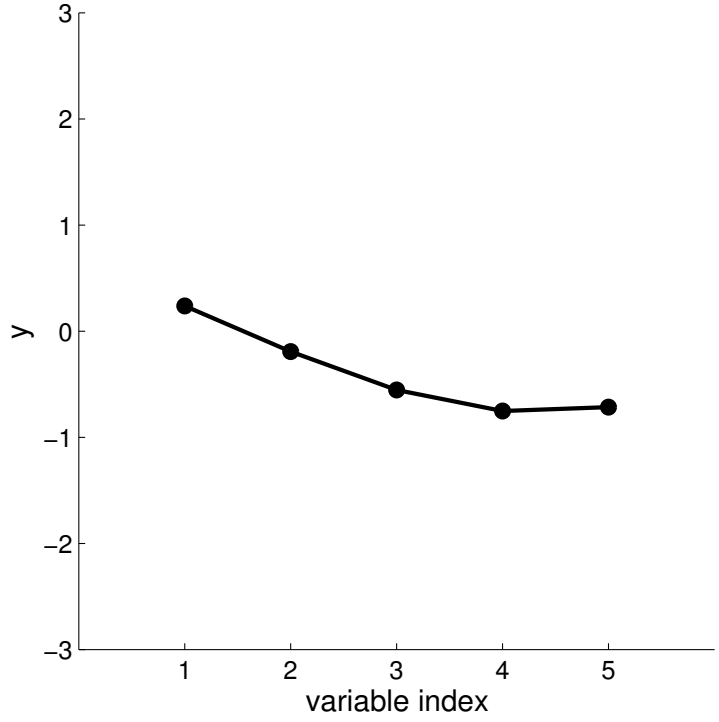


$$\Sigma = \begin{bmatrix} 1 & .9 \\ .9 & 1 \end{bmatrix}$$



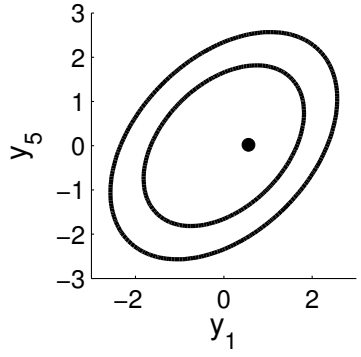
New visualisation

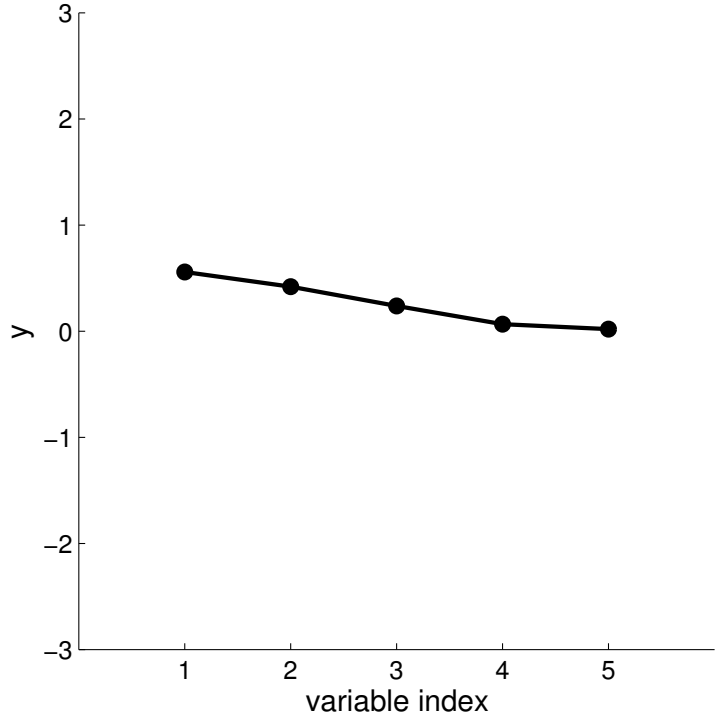




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

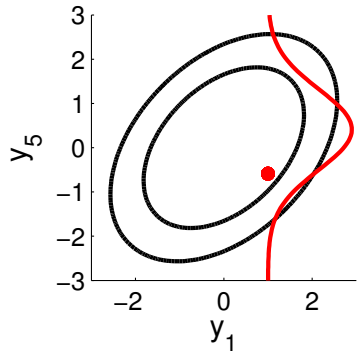
New visualisation

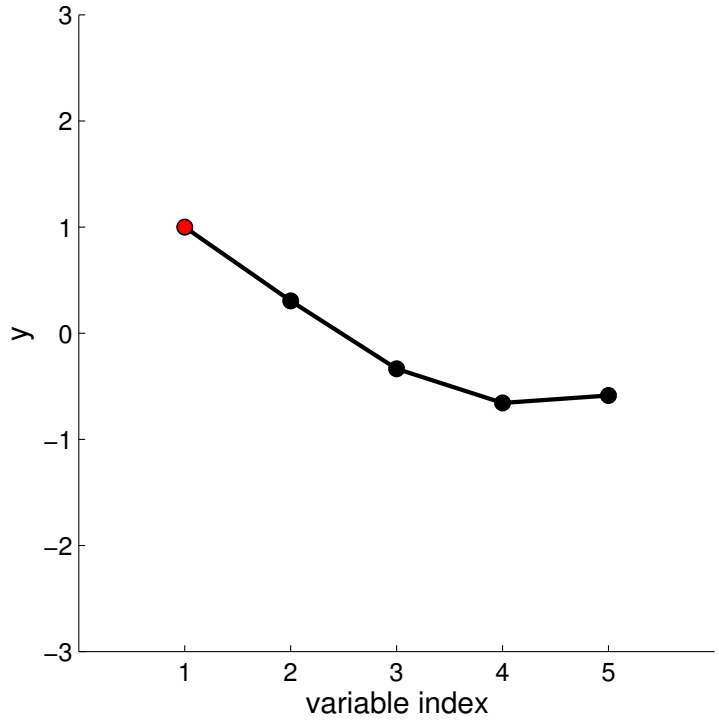




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

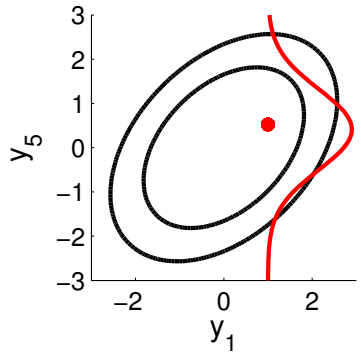
New visualisation

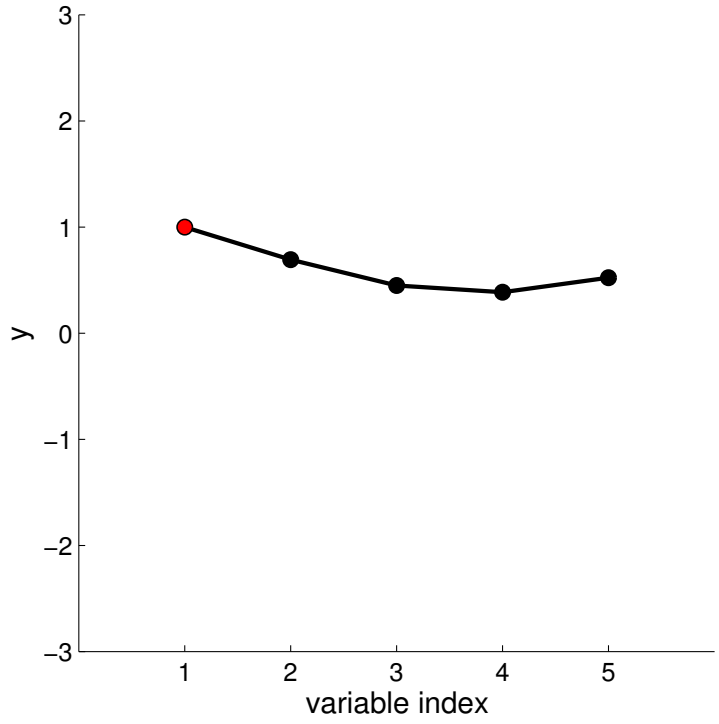




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

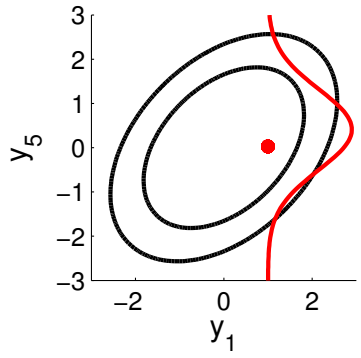
New visualisation

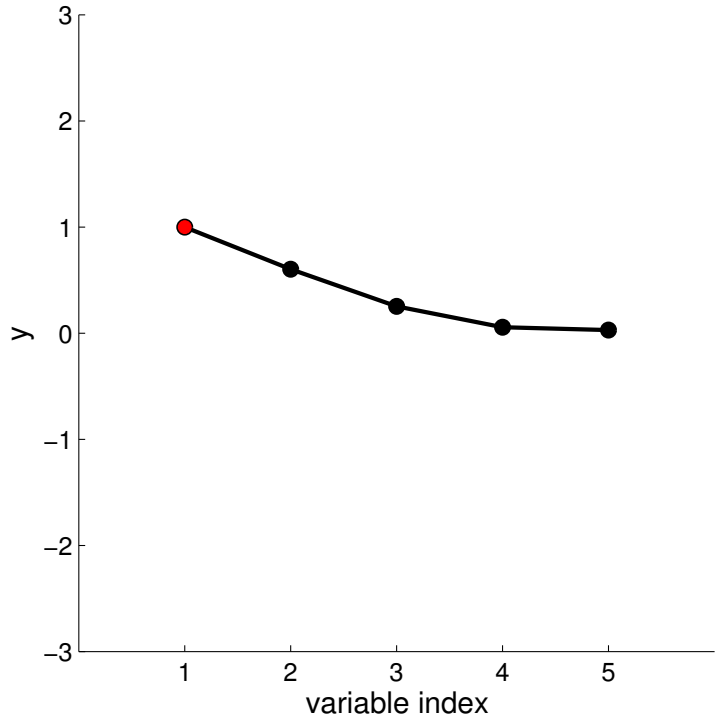




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

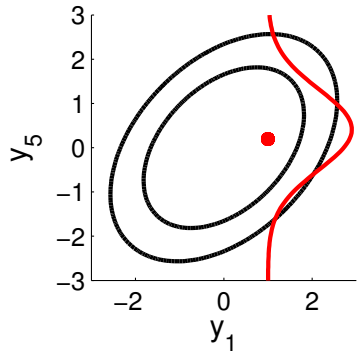
New visualisation

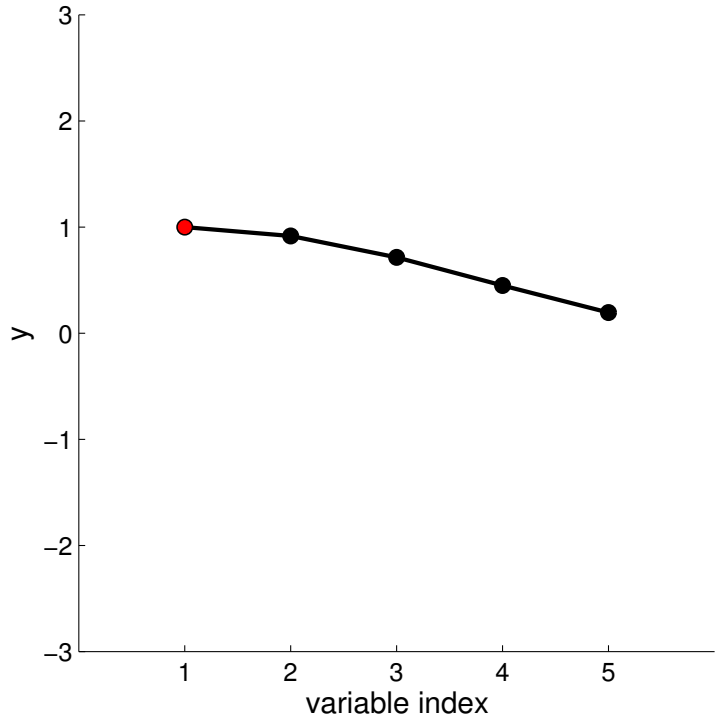




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

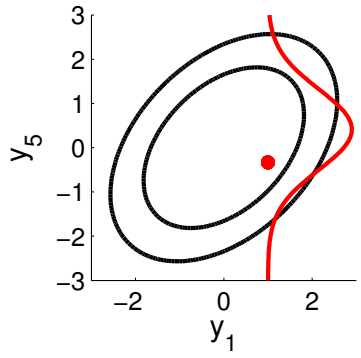
New visualisation

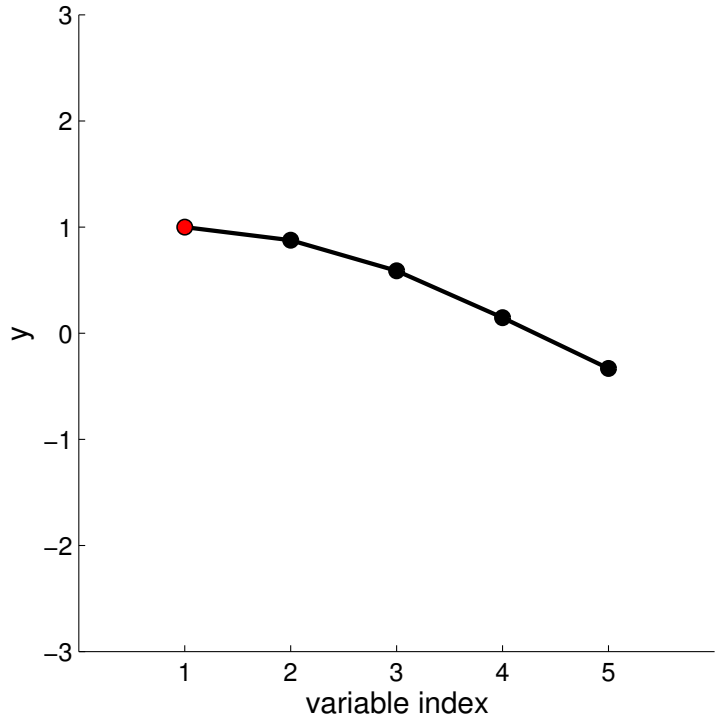




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

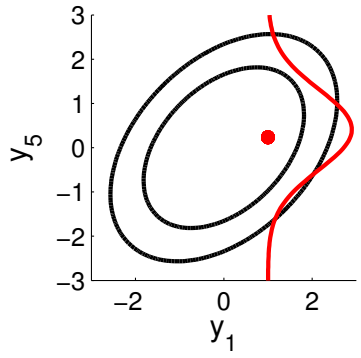
New visualisation

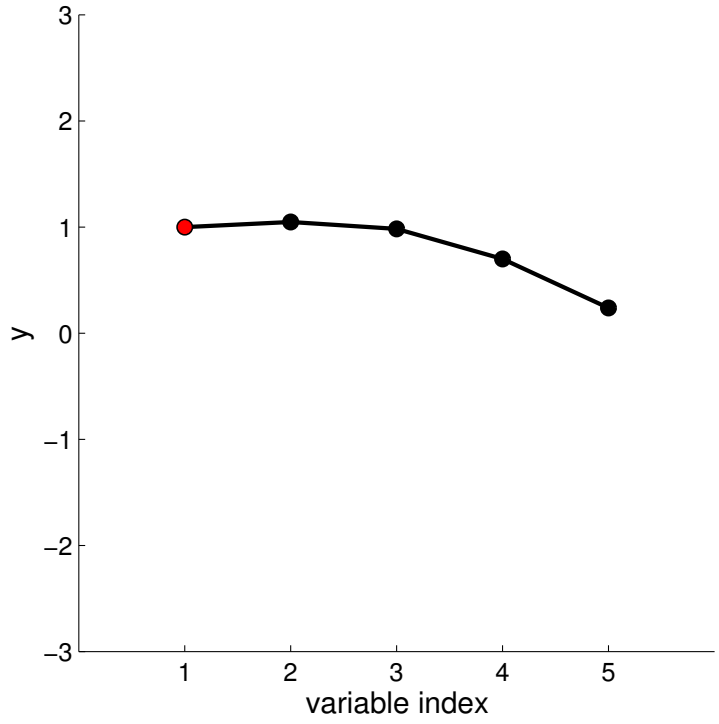




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

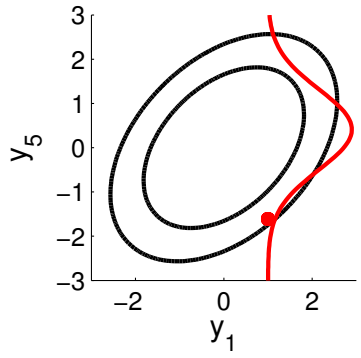
New visualisation

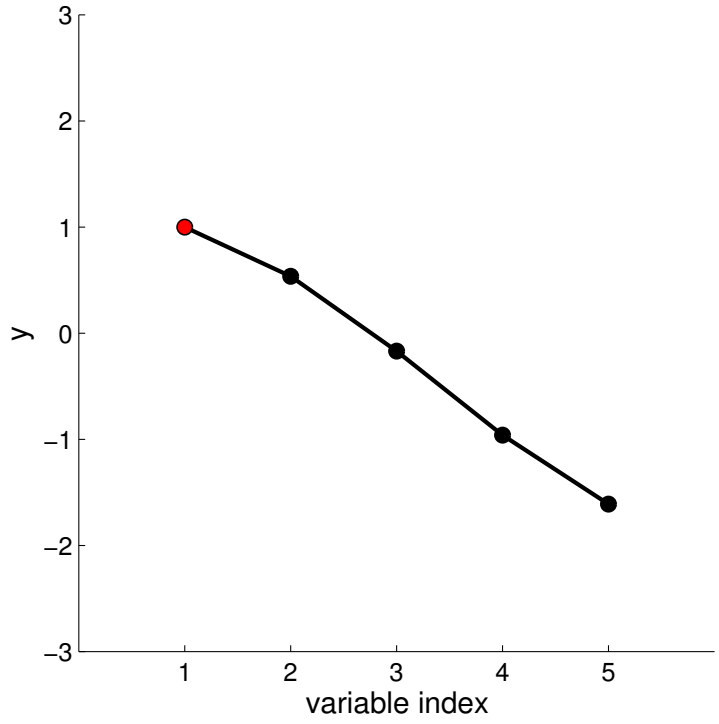




$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

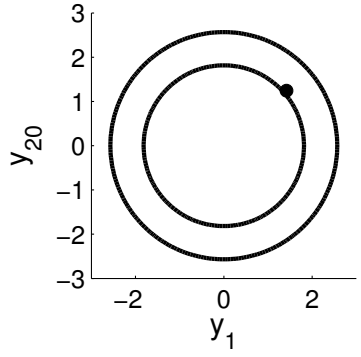
New visualisation

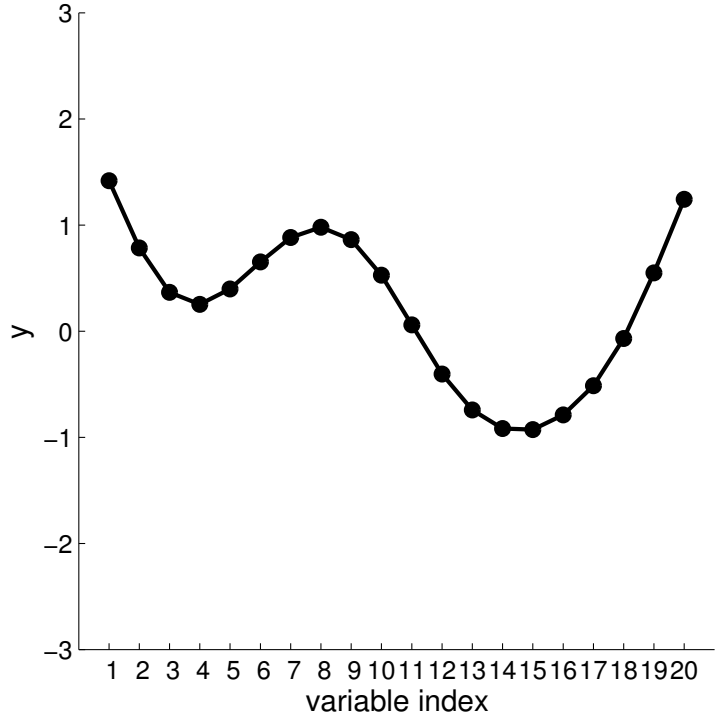




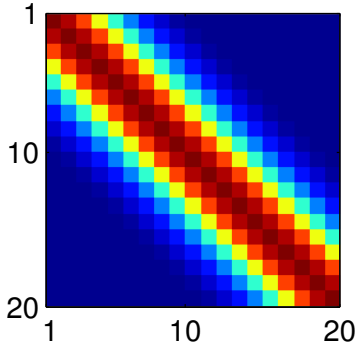
$$\Sigma = \begin{bmatrix} 1 & .9 & .8 & .6 & .4 \\ .9 & 1 & .9 & .8 & .6 \\ .8 & .9 & 1 & .9 & .8 \\ .6 & .8 & .9 & 1 & .9 \\ .4 & .6 & .8 & .9 & 1 \end{bmatrix}$$

New visualisation

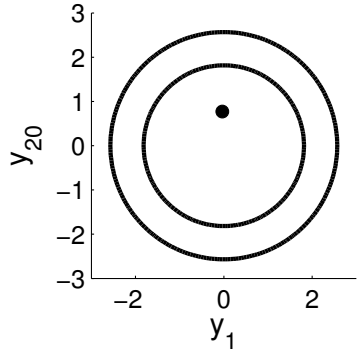


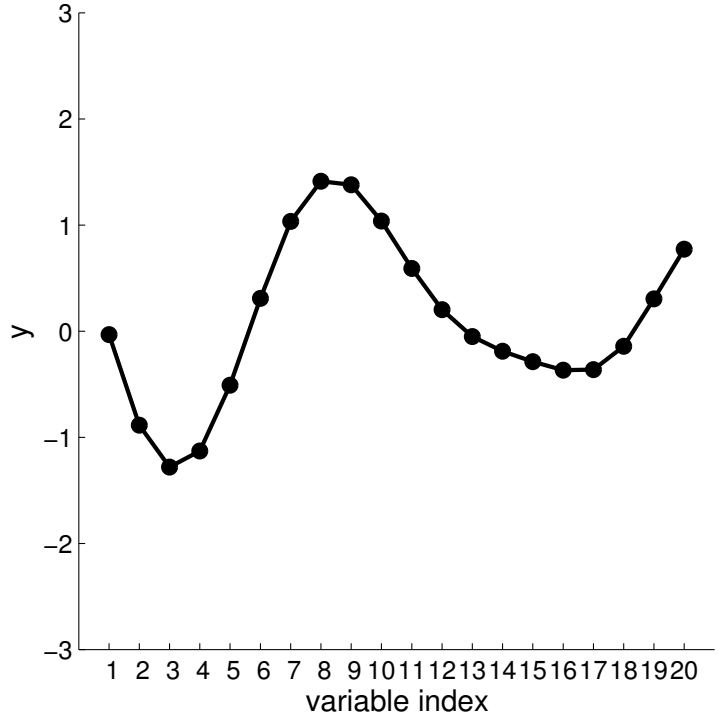


$\Sigma =$

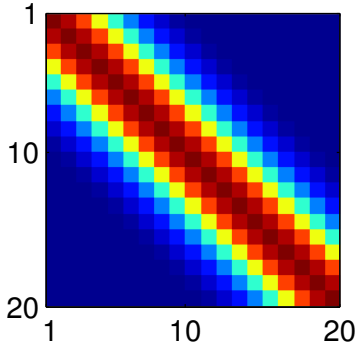


New visualisation

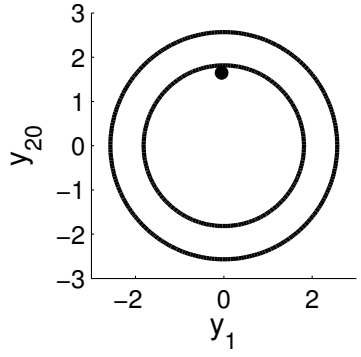


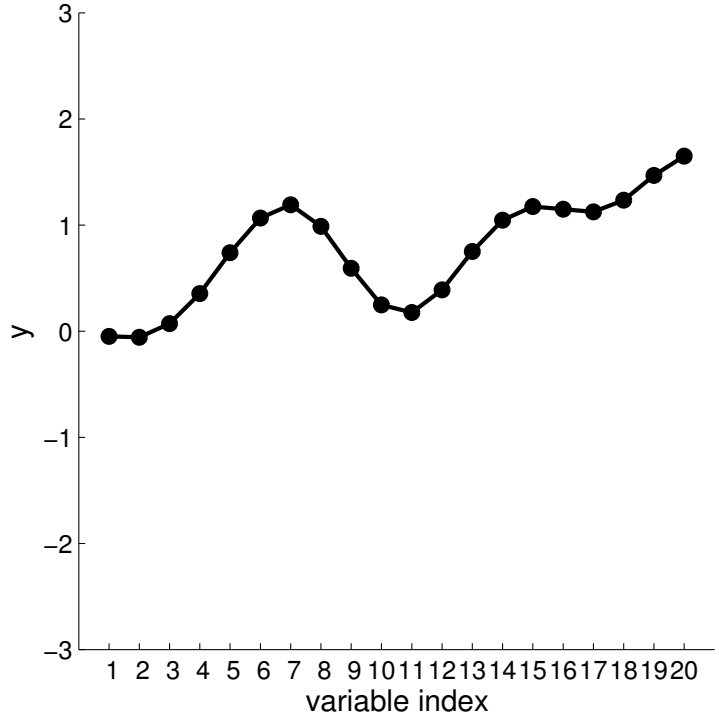


$\Sigma =$

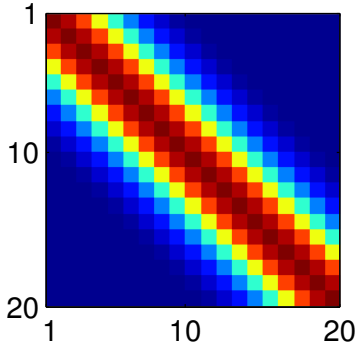


New visualisation

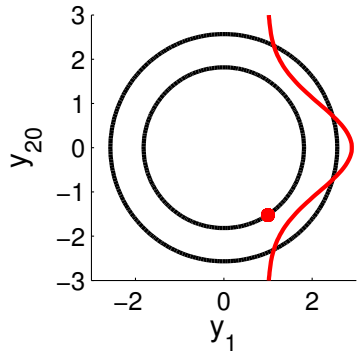


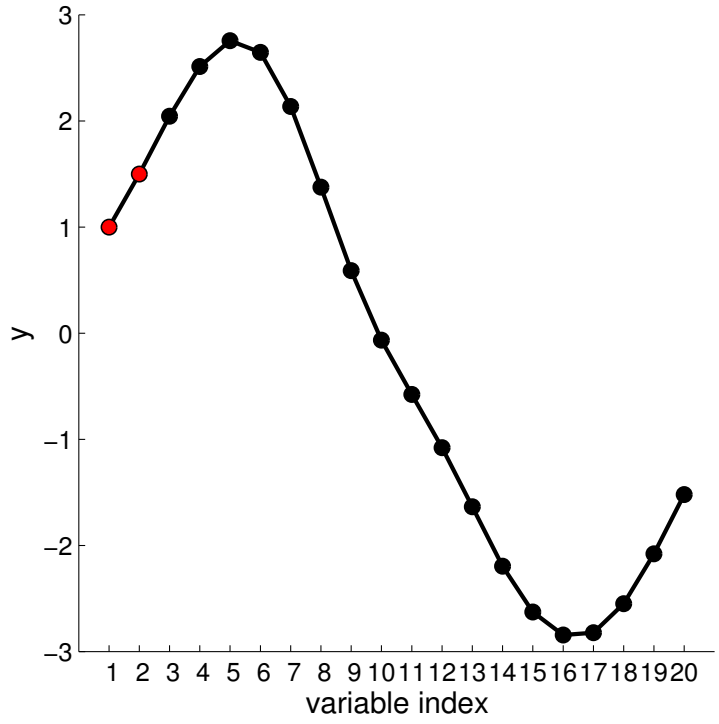


$\Sigma =$

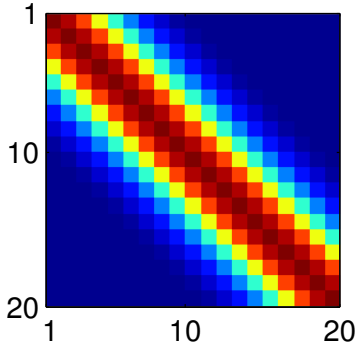


New visualisation

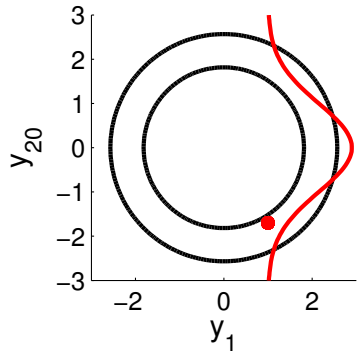


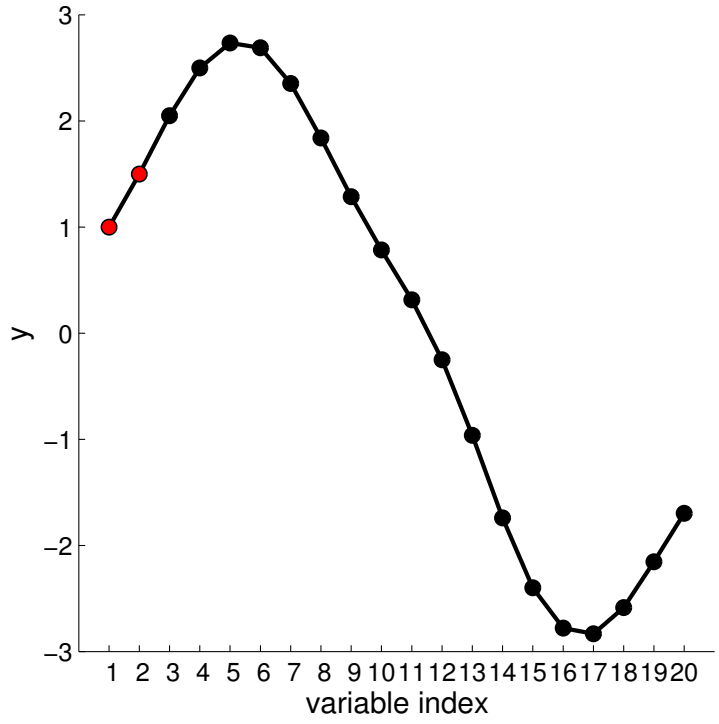


$\Sigma =$

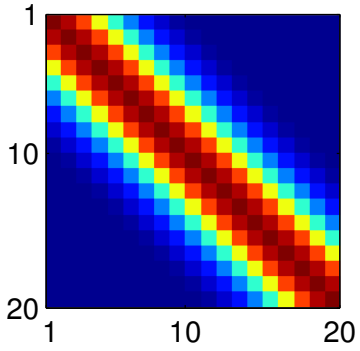


New visualisation

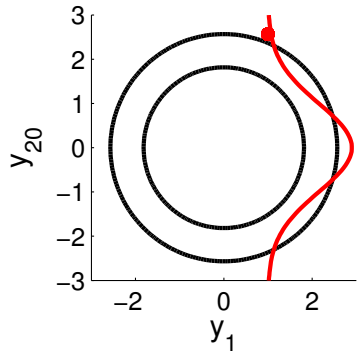


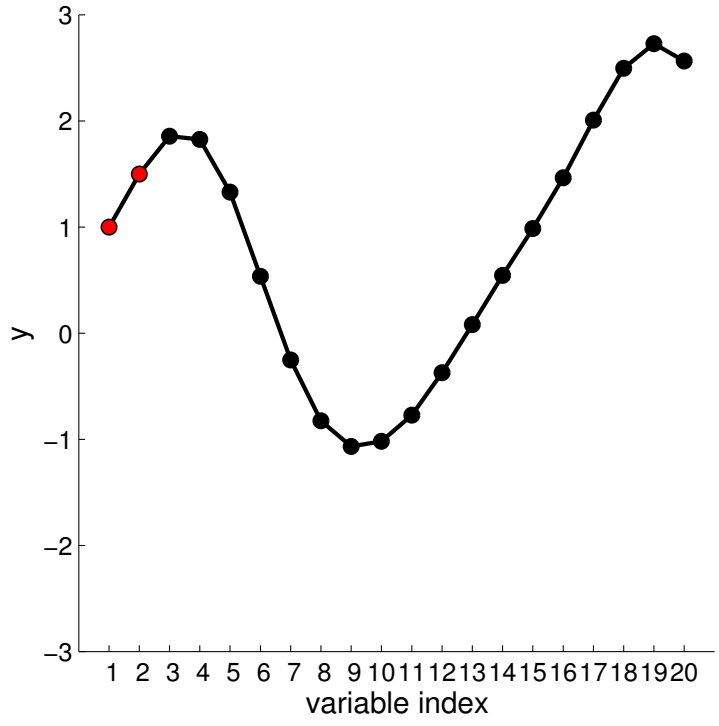


$\Sigma =$

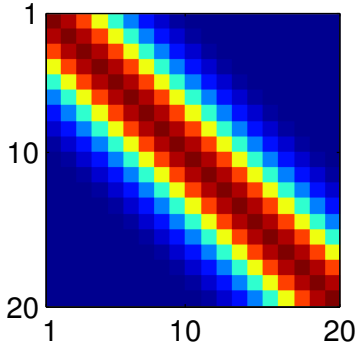


New visualisation



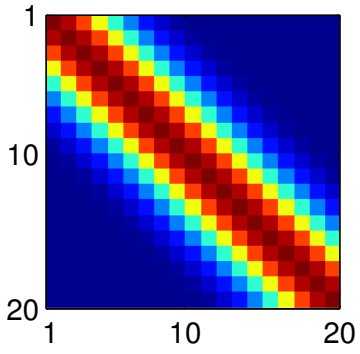


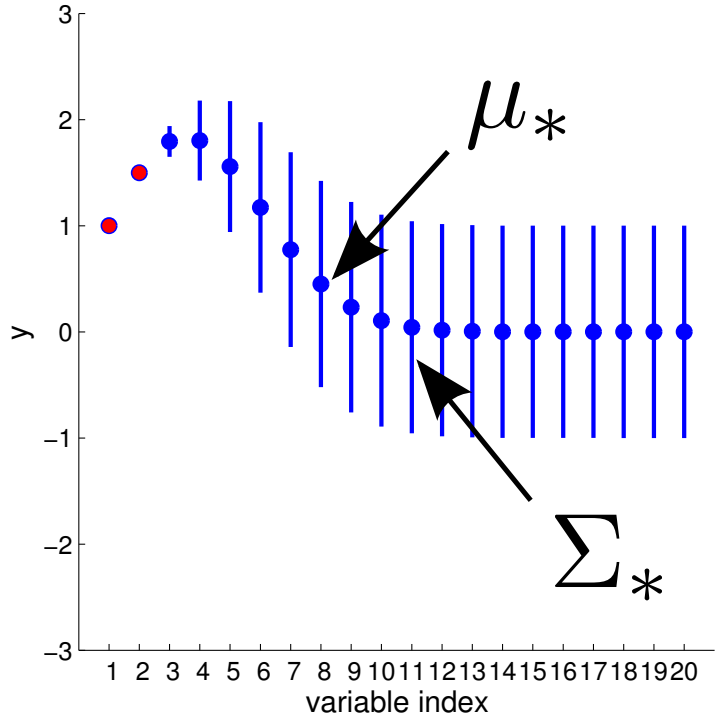
$\Sigma =$



Regression using Gaussians

$\Sigma =$



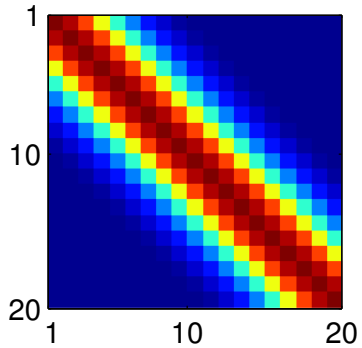


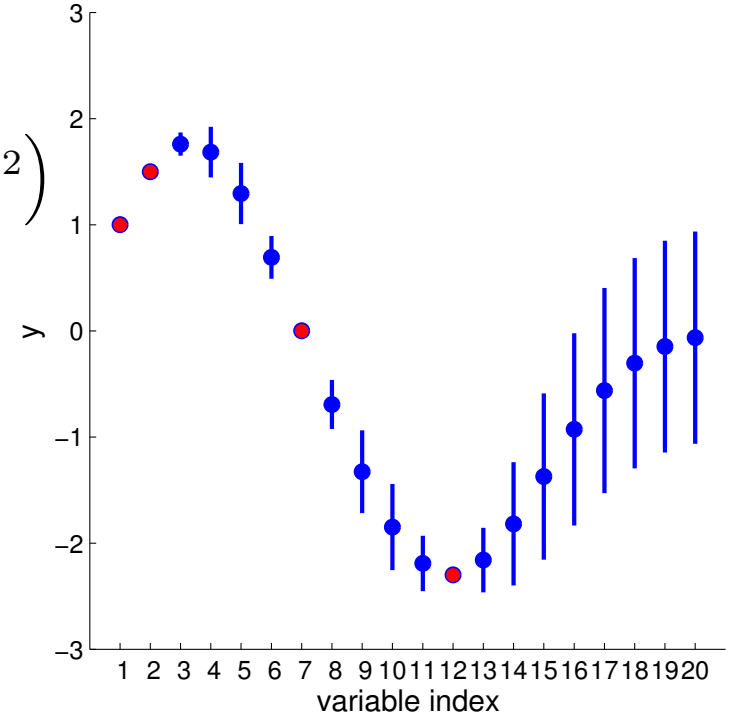
Regression using Gaussians

$$\Sigma(\mathsf{x}_1, \mathsf{x}_2) = \mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) + \mathsf{I}\sigma_{\mathsf{y}}^2$$

$$\mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) = \sigma^2 \exp\left(-\frac{1}{2\mathsf{l}^2}(\mathsf{x}_1 - \mathsf{x}_2)^2\right)$$

$\Sigma =$





Regression: probabilistic inference in function space

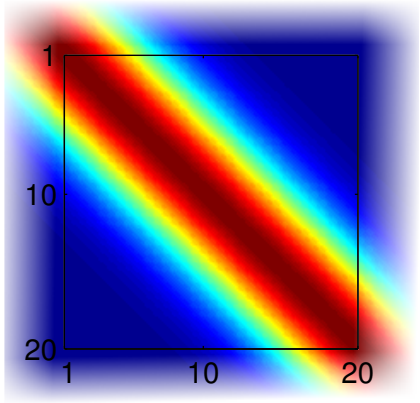
Non-parametric (∞-parametric)

$$p(\mathsf{y}|\theta) = \mathcal{N}(0, \Sigma)$$

$$\Sigma(\mathsf{x}_1, \mathsf{x}_2) = \mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) + \mathsf{I}\sigma_{\mathsf{y}}^2$$

$$\mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) = \sigma^2 \exp\left(-\frac{1}{2l^2}(\mathsf{x}_1 - \mathsf{x}_2)^2\right)$$

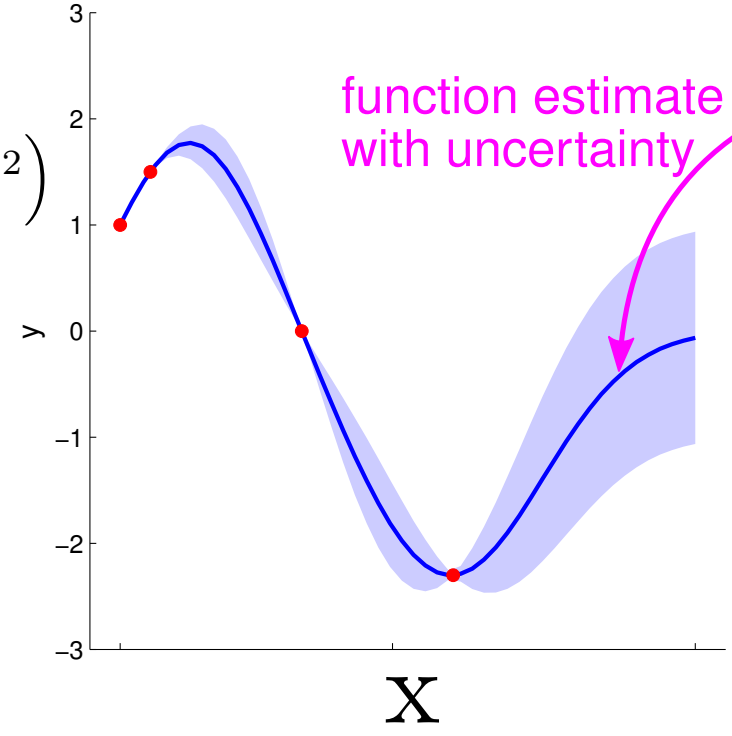
$\Sigma =$



Parametric model

$$\mathsf{y}(\mathsf{x}) = f(\mathsf{x}; \theta) + \sigma_{\mathsf{y}}\epsilon$$

$$\epsilon \sim \mathcal{N}(0, 1)$$



Regression: probabilistic inference in function space

Non-parametric (∞-parametric)

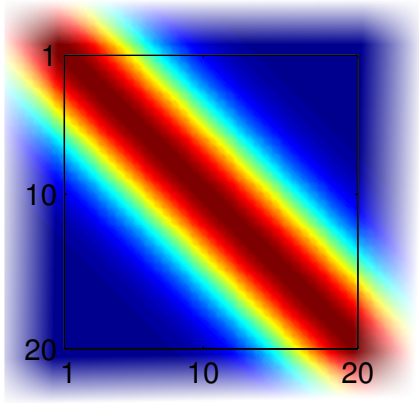
$$p(\mathsf{y}|\theta) = \mathcal{N}(0, \Sigma)$$

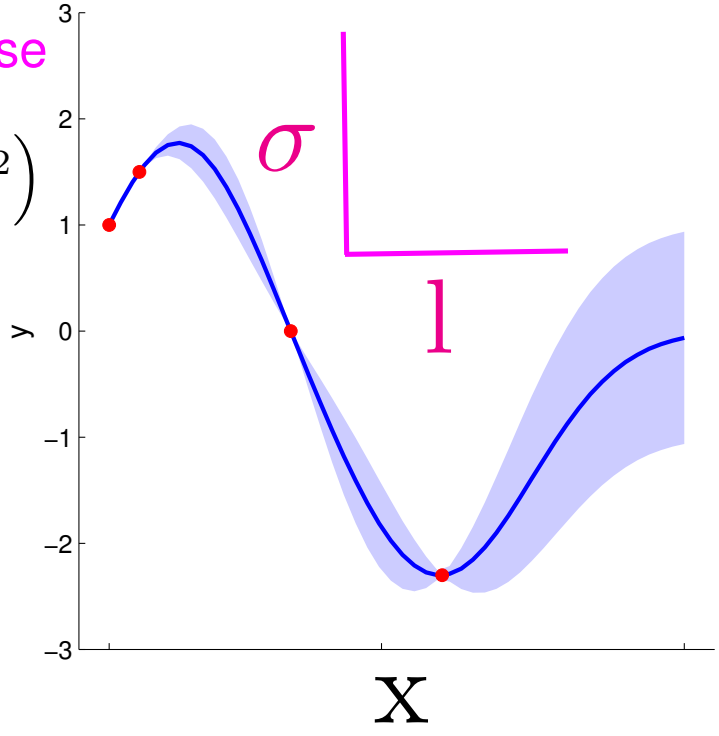
$$\Sigma(\mathsf{x}_1, \mathsf{x}_2) = \mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) + \mathsf{I}\sigma_{\mathsf{y}}^2 \leftarrow \text{noise}$$

$$\mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) = \sigma^2 \exp\left(-\frac{1}{2\mathsf{l}^2}(\mathsf{x}_1 - \mathsf{x}_2)^2\right)$$

vertical-scale horizontal-scale

$\Sigma =$





Parametric model

$$\mathsf{y}(\mathsf{x}) = f(\mathsf{x}; \theta) + \sigma_{\mathsf{y}}\epsilon$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

Squared Exponential (SE) kernel

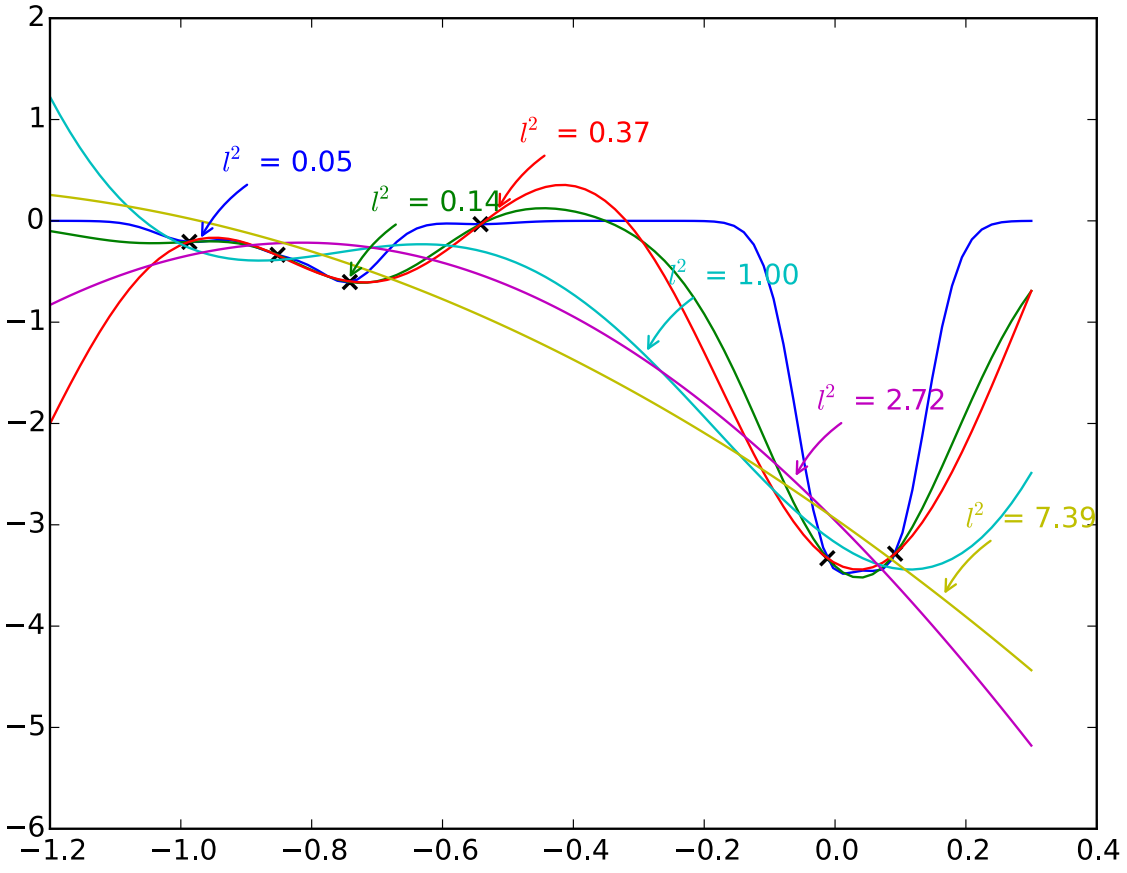
Usual choice for covariance is the Squared Exponential (SE) kernel (a.k.a. RBF, exponentiated quadratic, Gaussian):

$$k(x, x') = \sigma_d^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right)$$

The kernel's parameters $\{l, \sigma_d\}$ are called the GP's hyperparameters.

When l is very high, neighbouring points are very correlated i.e. that feature is less relevant.

How to select the hyperparameters?



Varying lengthscale from very low (0.04) to high (7.4), with noise hyperparameter fixed.

Hyperparameters

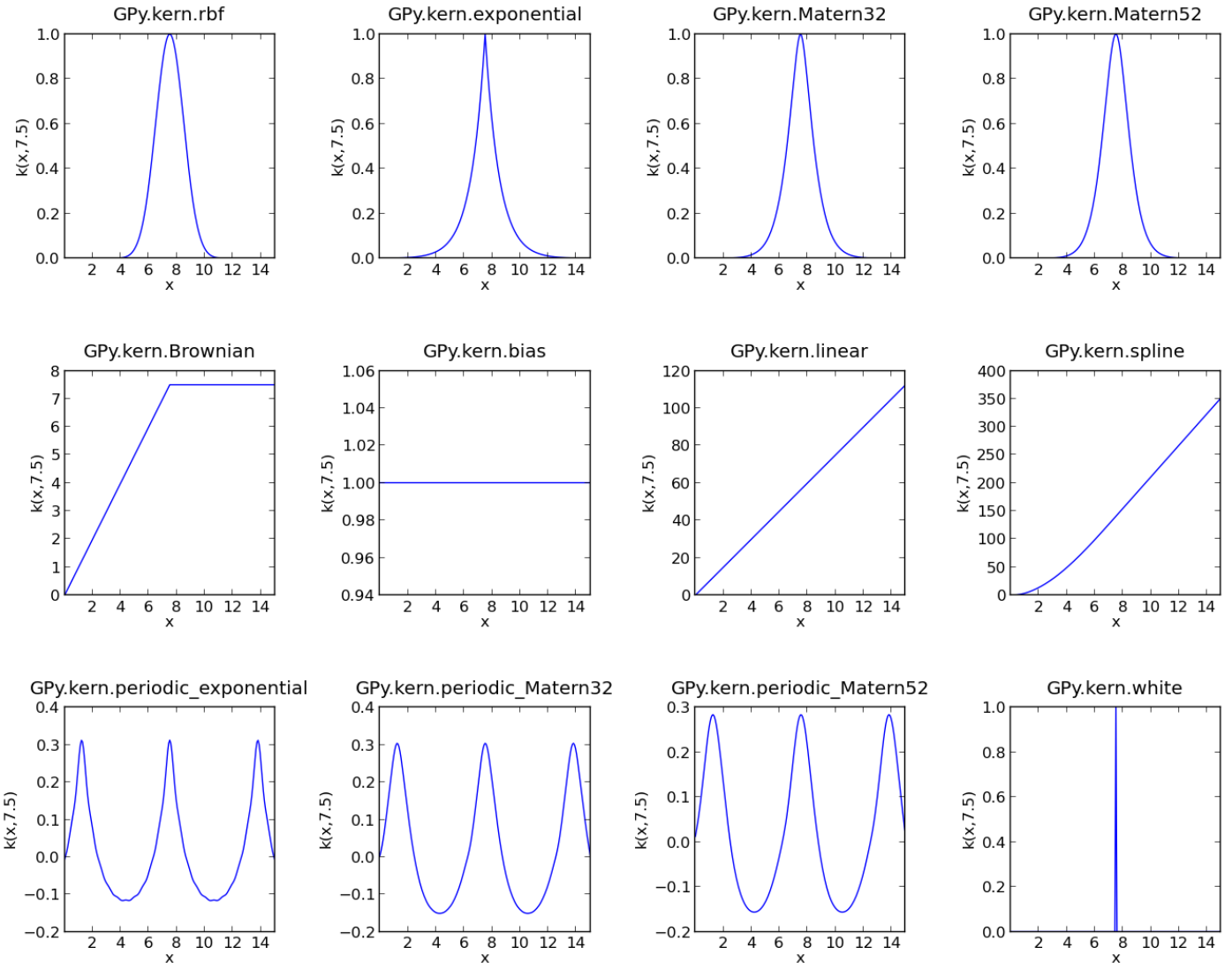
Training a GP means:

- finding the right kernel
- finding values of hyperparameters

The covariance function controls properties of the GP:

- smoothness
- stationarity
- periodicity
- etc.

Other kernels



Inference

Although the GP is an infinite dimensional object, marginalisation property means we only need to work with finite dimensions.

For prediction we need the conditional distribution over test points y_* given observed points $\mathbf{y}$:

$$p(y_*|\mathbf{y}, x_*, X) = \frac{p(\mathbf{y}, y_*|x_*, X)}{p(\mathbf{y}|X)}$$

We'll omit the conditioning on x_*, X for brevity henceforth.

Inference

Consider GP over single extra point

$$p(\mathbf{y}) = \mathcal{N}(0, K + \sigma_n^2 I)$$

$$p(\mathbf{y}, y_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} K + \sigma_n^2 I & K_*^\top \\ K_* & K_{**} + \sigma_n^2 \end{bmatrix}\right)$$

where K are covariance matrices, i.e.,

- $K \in \mathcal{R}^{N \times N}$ are the covar. between training instances
- $K_* \in \mathcal{R}^N$ are the covar. between test and training instances
- $K_{*,*} \in \mathcal{R}$ is the test variance

Inference

Recall property of the multi-variate Gaussian:

Conditional distribution is Gaussian

Leads to the following predictive posterior

$$
\begin{aligned}
p(y_*|\mathbf{y}) &= \mathcal{N}(\mu_*, \Sigma_*) \\
\text{with} \quad \mu_* &= K_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} \\
\Sigma_* &= K_{**} - K_*^\top (K + \sigma_n^2 I)^{-1} K_*
\end{aligned}
$$

- Note that predictive mean is linear in the data; and
- predictive covariance is the prior covariance is reduced by the information the observations give about the function

Model selection

Bayesian formulation seemingly removes the need for training, as the parameters are integrated out.

However, need to select hyperparameter values, e.g.,

- σ_n white noise
- l length (horizontal) scale
- σ_d vertical scale
- ...

How to select appropriate values?

Consider the marginal likelihood, $p(\mathbf{y}|X)$.

Model selection

Marginal likelihood defined as

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

Observe that

- as space of possible functions grows, $p(\mathbf{f}|X)$ diminishes
- poorly fitting functions will score poorly on $p(\mathbf{y}|\mathbf{f})$
- both affected by type of kernel and hyperparameter values

The marginal likelihood balances these opposing forces.

Model selection for training

Marginal likelihood permits analytical solution due to conjugacy

$$p(\mathbf{y}|X) = \int \underbrace{p(\mathbf{y}|\mathbf{f})}_{\text{Gaussian}} \underbrace{p(\mathbf{f}|X)}_{\text{Gaussian}} \, d\mathbf{f}$$
$$= \mathcal{N}(0, K + \sigma_n^2 I)$$

arising from the property that the product of two Gaussians is an unnormalised Gaussian.

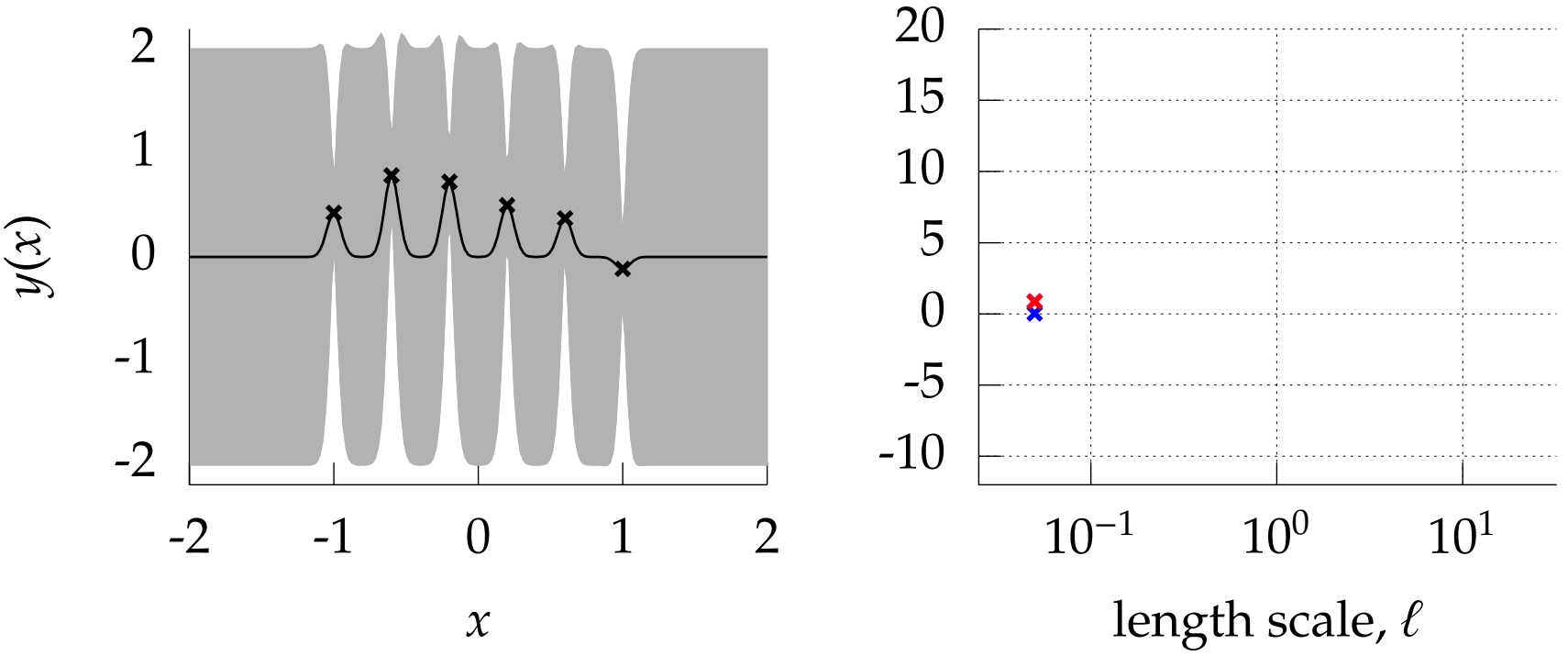
Permits analytical solution

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi$$

Can optimise above with respect to model hyperparameters using gradient ascent, a.k.a. Type II maximum likelihood.

Learning Covariance Parameters

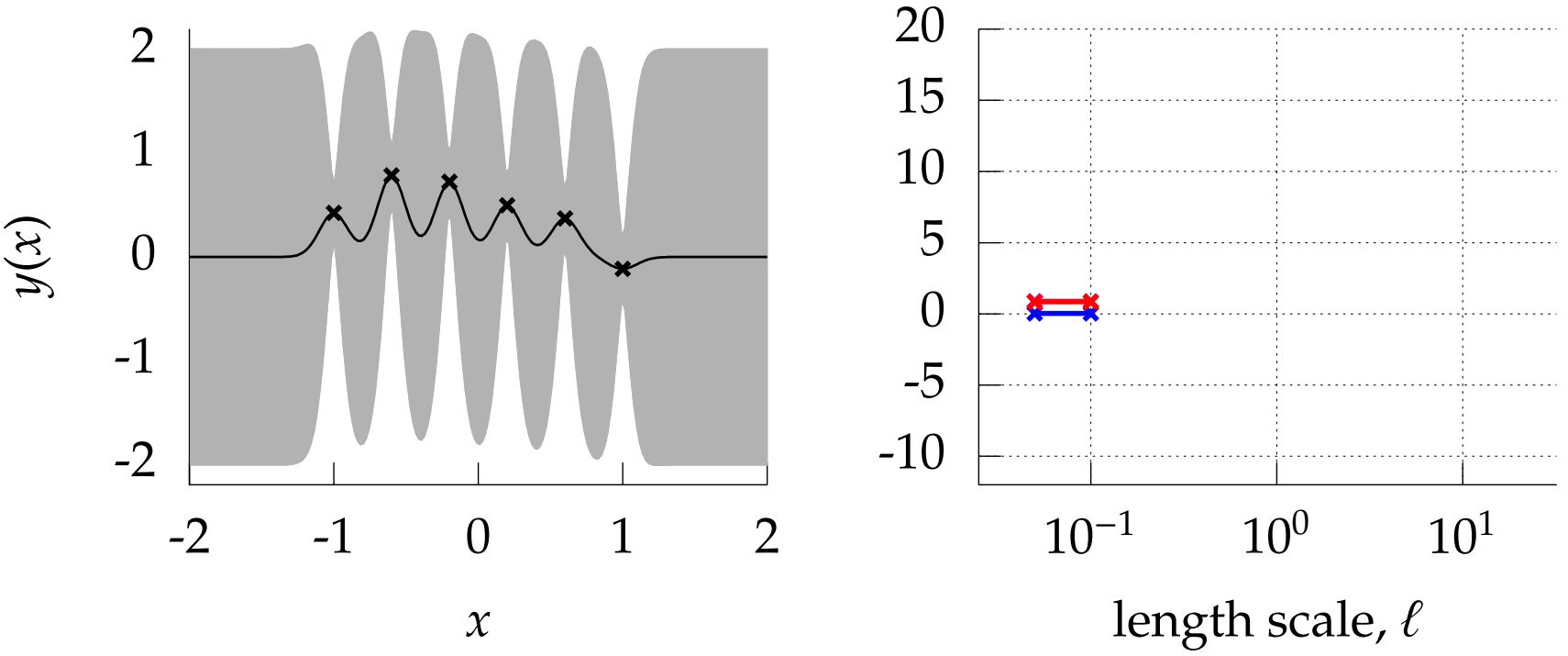
Can we determine length scales and noise levels from the data?



$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

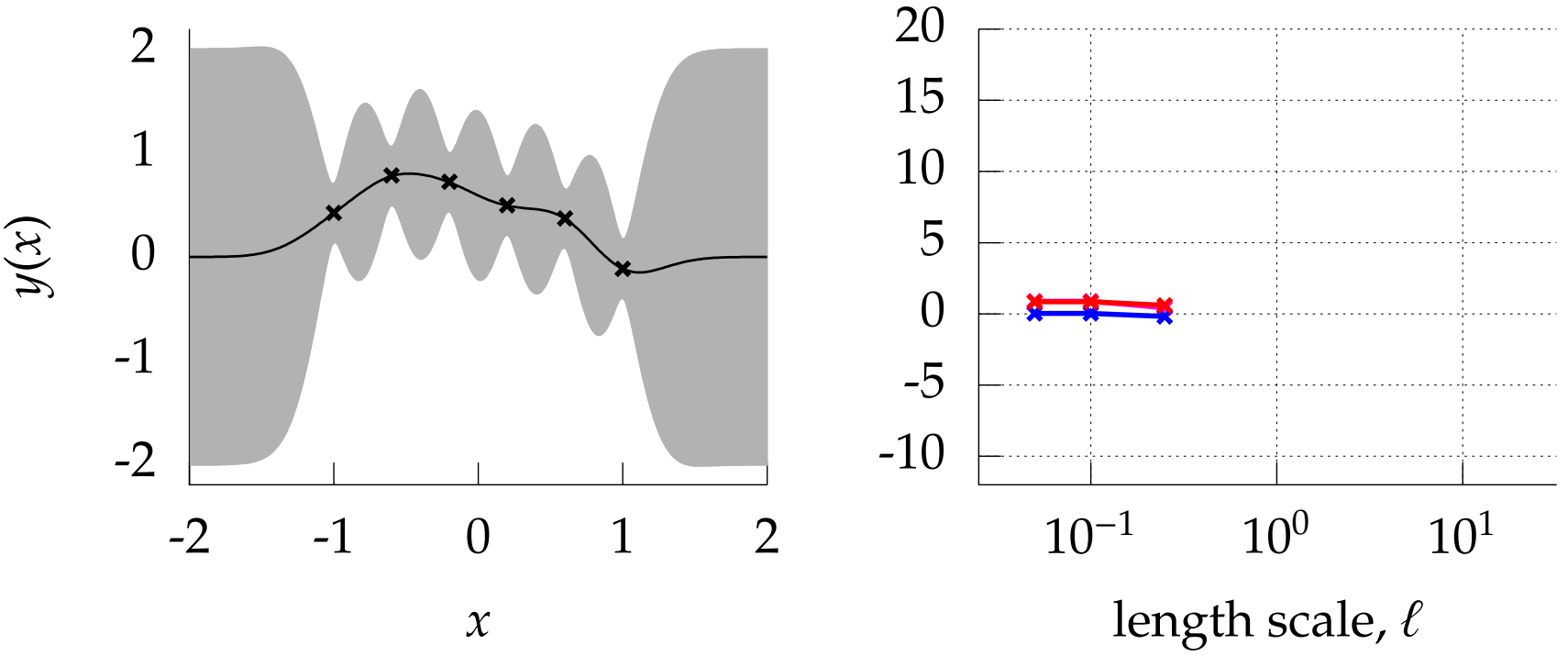
Can we determine length scales and noise levels from the data?



$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

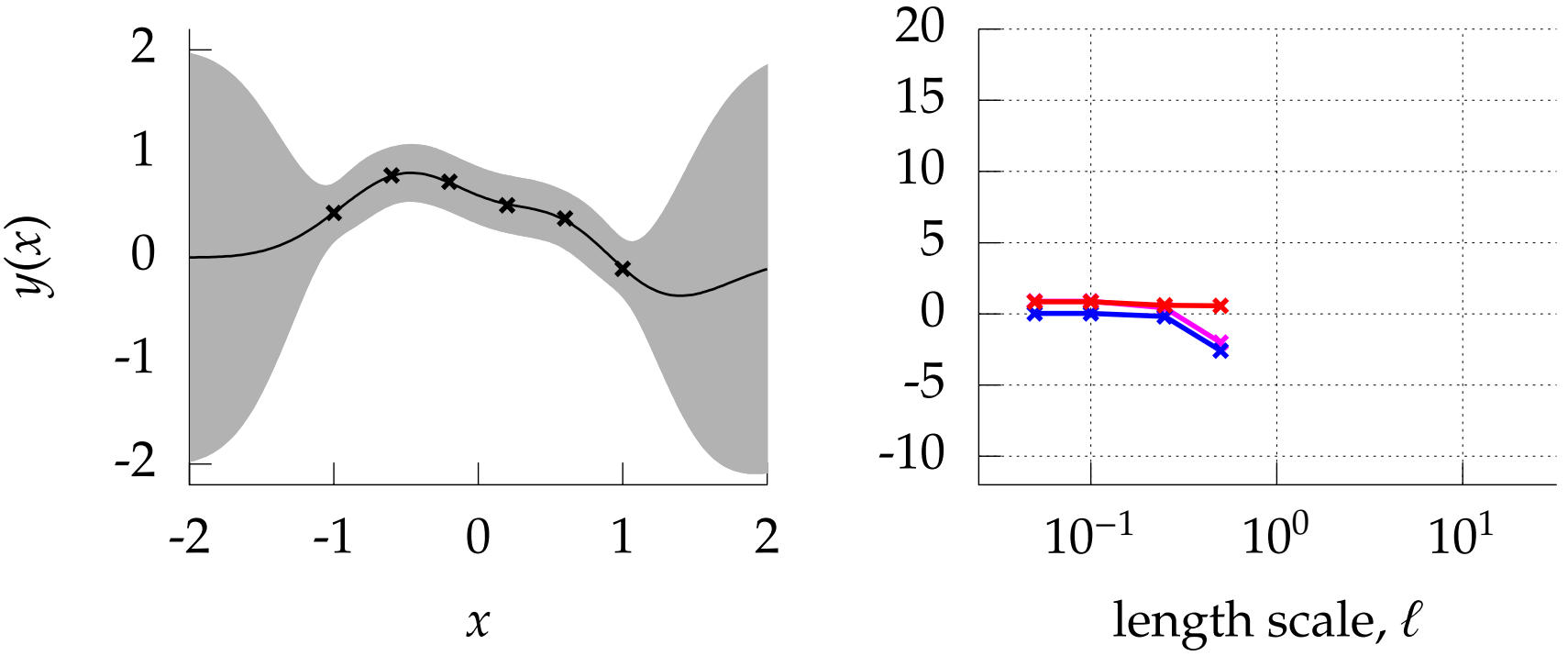
Can we determine length scales and noise levels from the data?



$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

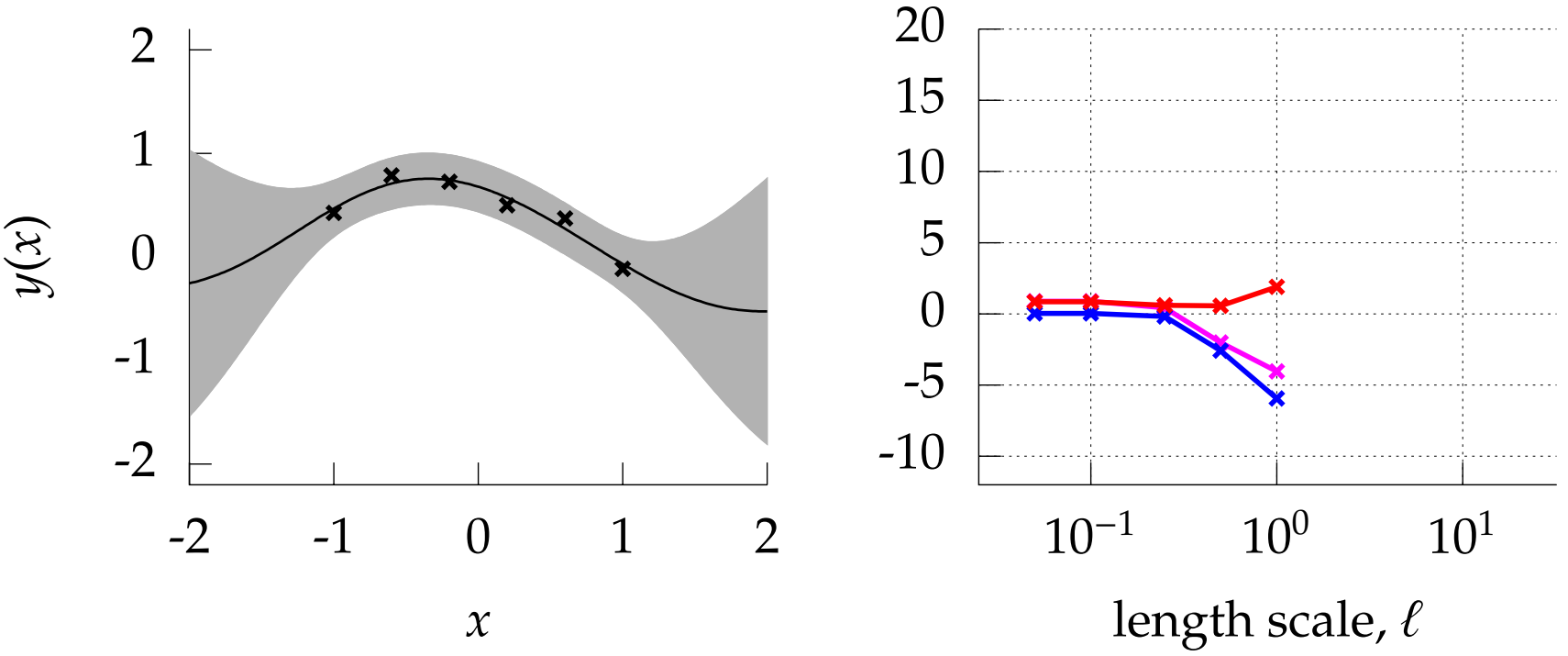
Can we determine length scales and noise levels from the data?



$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

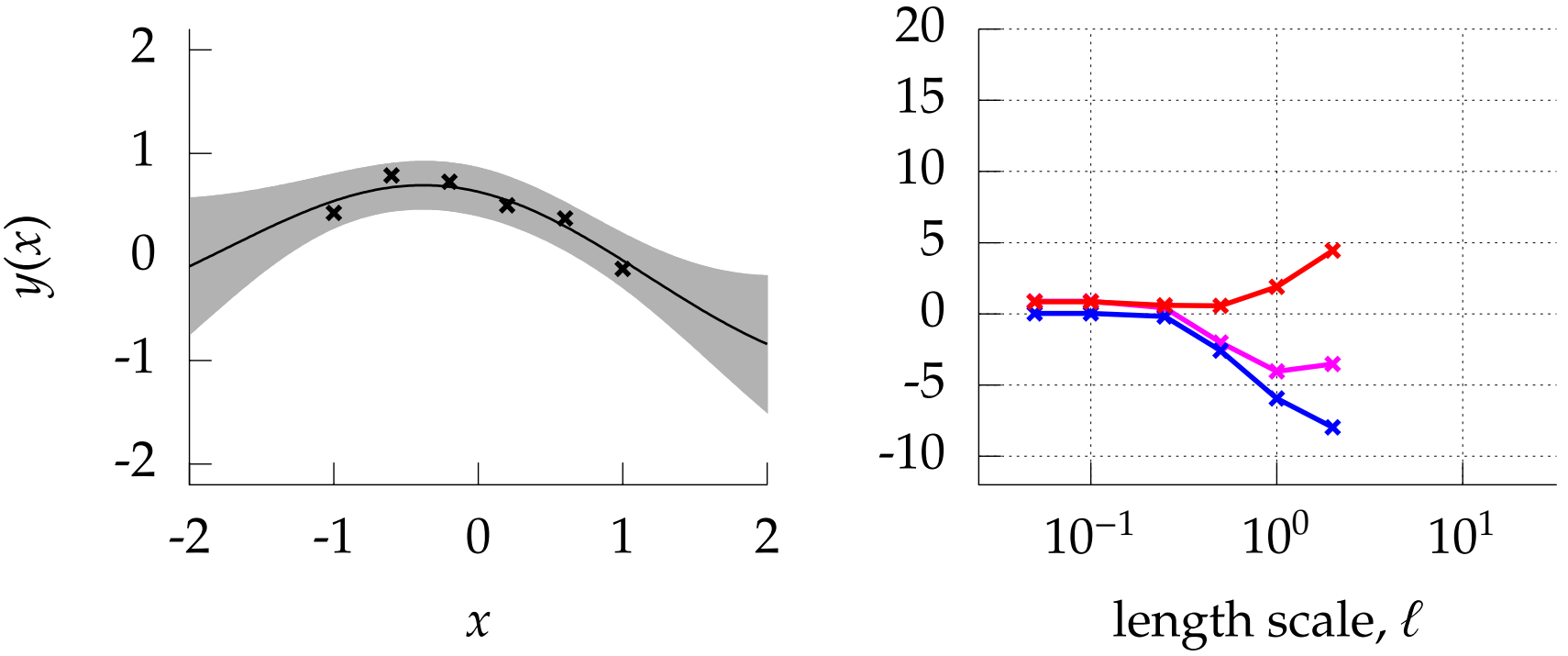
Can we determine length scales and noise levels from the data?



$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

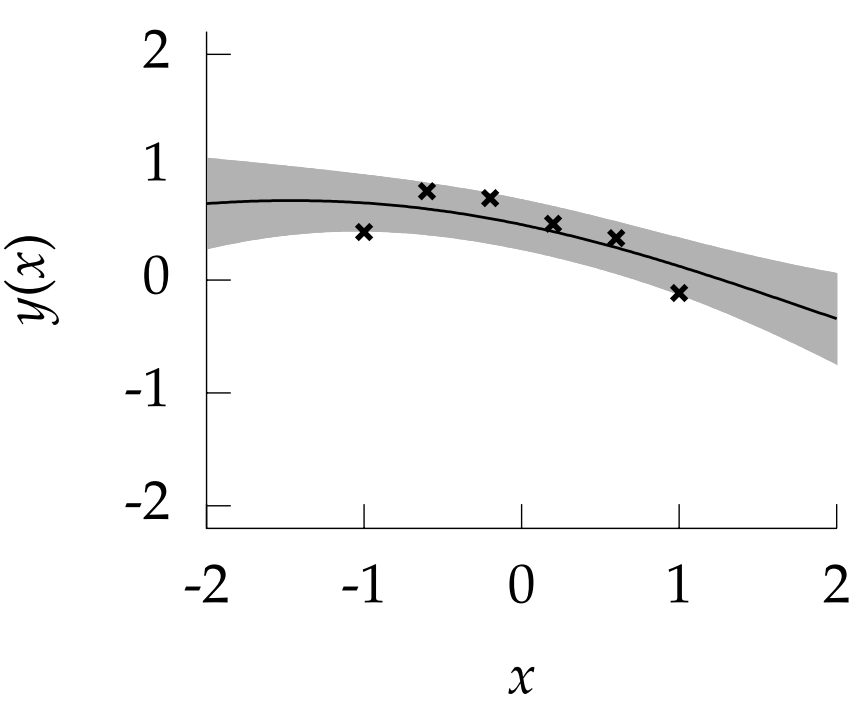
Can we determine length scales and noise levels from the data?

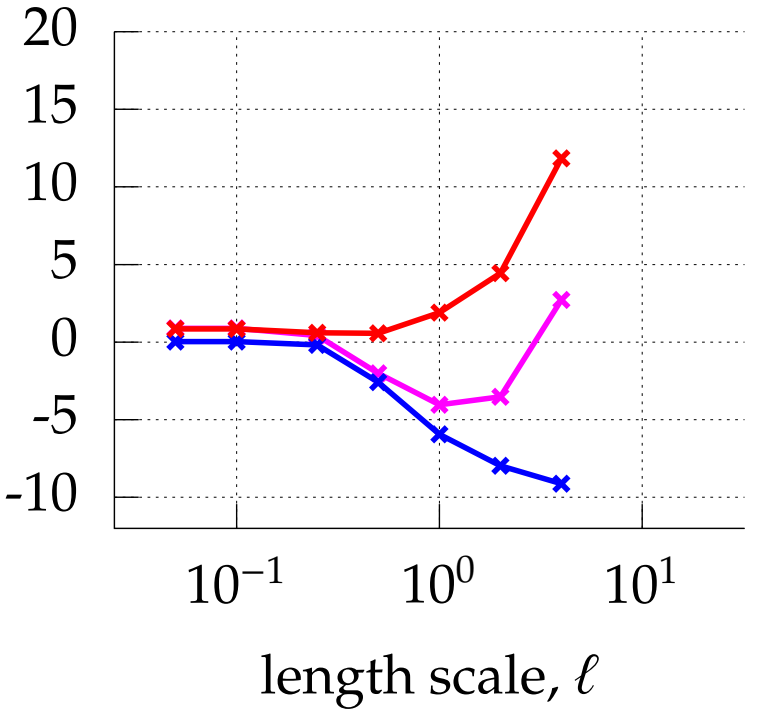


$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

Can we determine length scales and noise levels from the data?

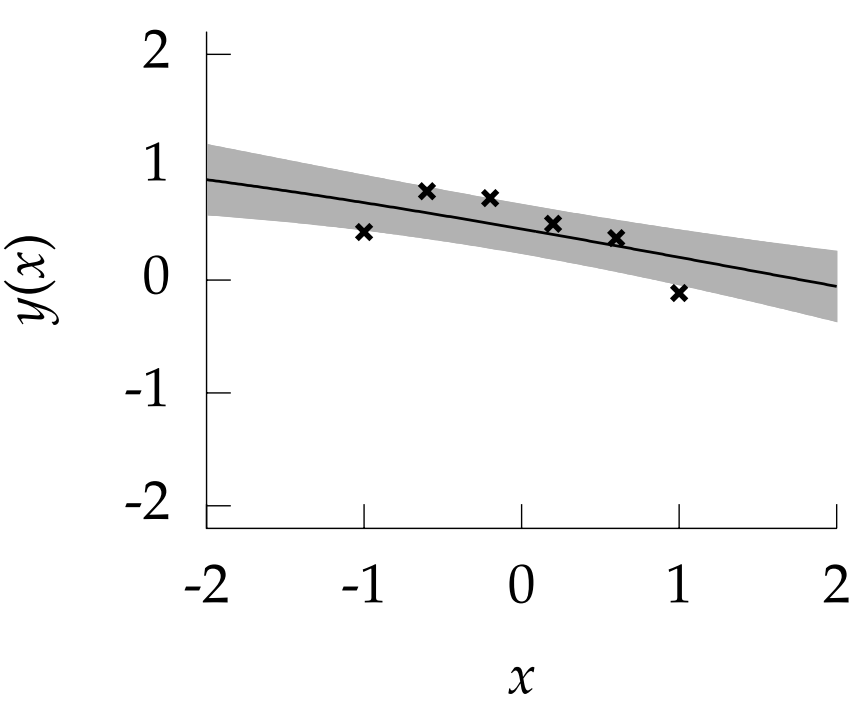


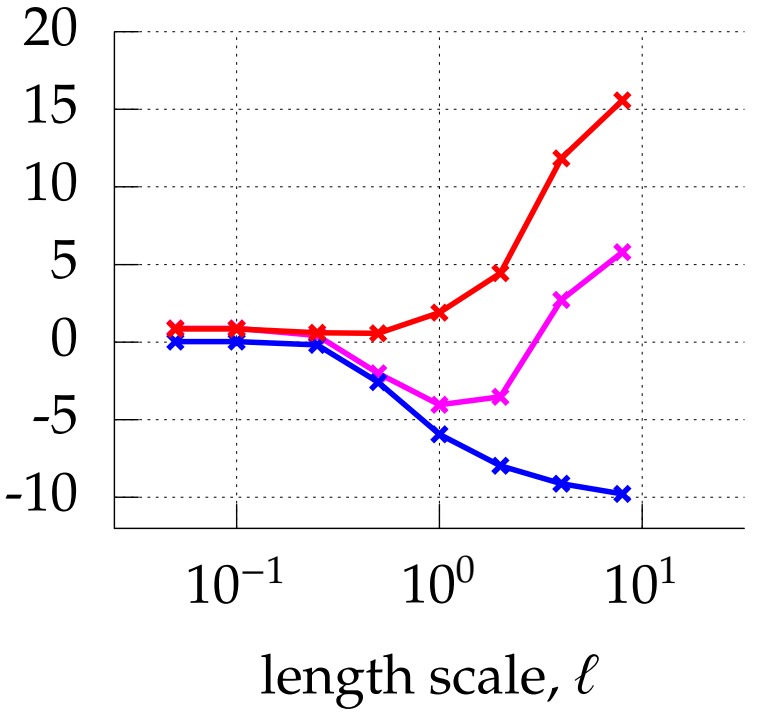


$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

Can we determine length scales and noise levels from the data?

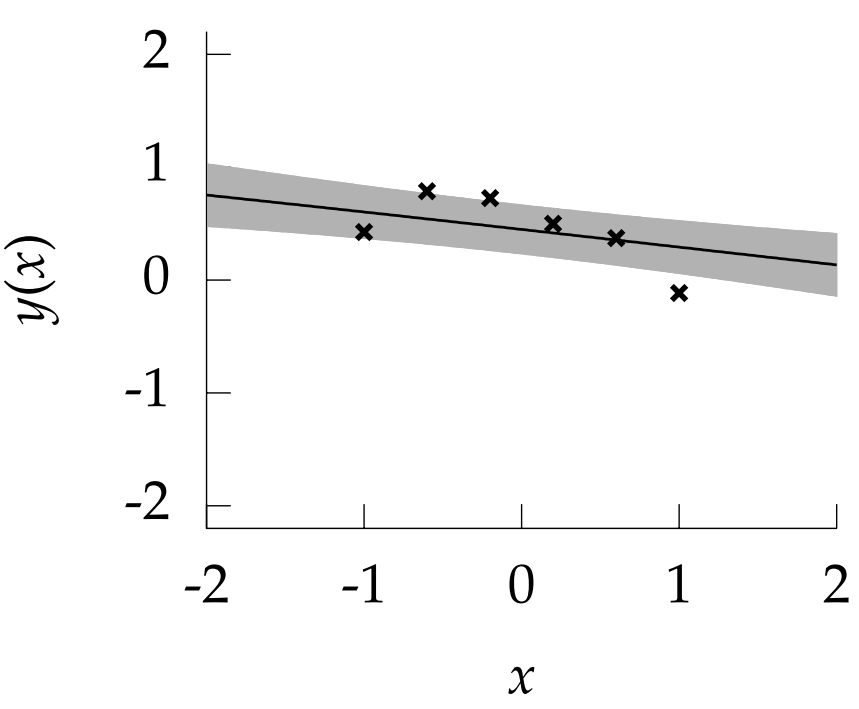


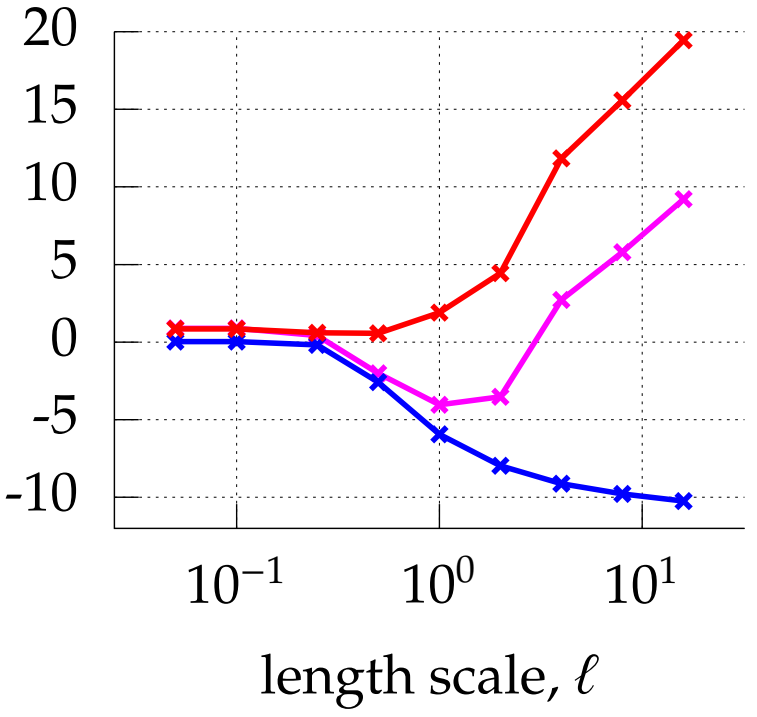


$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Learning Covariance Parameters

Can we determine length scales and noise levels from the data?





$$E(\boldsymbol{\theta}) = \frac{1}{2}\log|\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{2}$$

Beyond GP regression

- Classification
- Ordinal regression
- Count data
- Model selection: kernel and hyperparameter optimisation
- Kernel design
- Extrapolation cf interpolation
- Multi-output GPs for multi-task learning
- Sparse GPs for scaling to large feature and input spaces
- Latent variable models, non-linear probabilistic variant of PCA/CCA et al
- And many others ...

GP limitations

Non-parametric formulation complicates scaling

- $O(N^3)$ time complexity and $O(N^2)$ space complexity
- but ongoing work to bring this down, e.g., $O(NM^2)$ time complexity or even constant in N

Brilliant for regression, but more difficult for other likelihoods

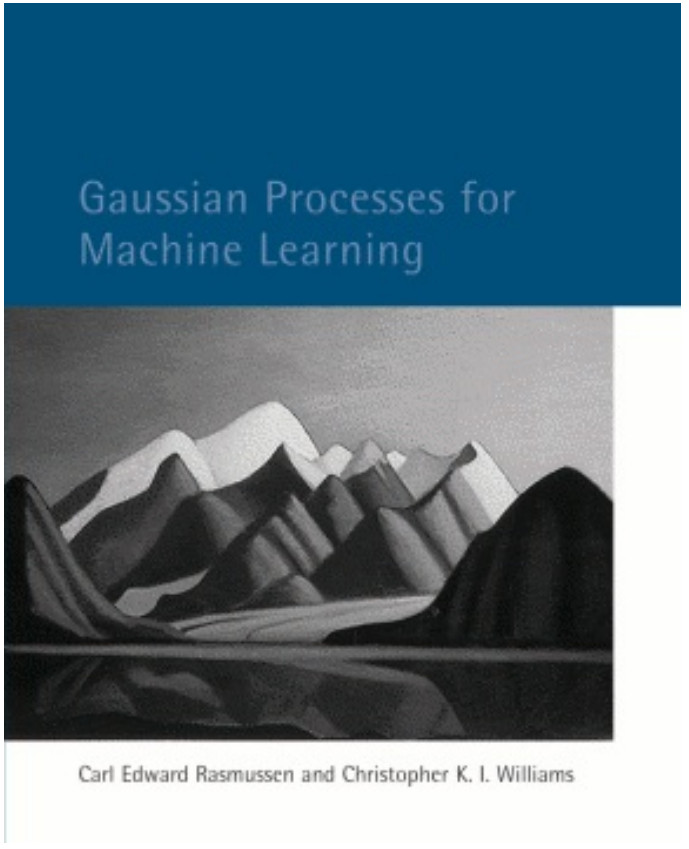
- suite of approximation approaches to deal with inference for non-conjugate configurations

Not as mature as many other frameworks

- but 'coming of age', many big issues have been addressed
- even application to 'big data' scenarios

Resources

Free book:
`http://www.gaussianprocess.org/gpml/chapters/`



Tutorials

- GPs for Natural Language Processing tutorial (ACL 2014)
  `http://goo.gl/18heUk`
- GP Schools in Sheffield and roadshows in Kampala, Pereira, and (future) Nyeri, Melbourne
  `http://ml.dcs.shef.ac.uk/gpss/`
- GP Regression demo
  `http://www.tmpl.fi/gp/`
- Annotated bibliography and other materials
  `http://www.gaussianprocess.org`

Toolkits

- GPML (Matlab)
  `http://www.gaussianprocess.org/gpml/code`
- GPy (Python)
  `https://github.com/SheffieldML/GPy`
- GPstuff (R, Matlab, Octave)
  `http://becs.aalto.fi/en/research/bayes/gpstuff/`

Outline

Outline

Case study: User impact on Twitter

Predicting and characterising user impact on Twitter

- define a user-level impact score
- use user's text and profile information as features to predict the score
- analyse the features which better predict the score
- provide users with 'guidelines' for improving their score

Instance of a text prediction problem

- emphasis on feature analysis and interpretability (specific to social science applications)
- non-linear variation

See Lampos et al. (2014), EACL.

Sparse GPs

Exact inference in a GP

- Memory: $\mathcal{O}(n^2)$
- Time: $\mathcal{O}(n^3)$

Sparse GP approximation

- Memory: $\mathcal{O}(n \cdot m)$
- Time: $\mathcal{O}(n \cdot m^2)$
  where m is selected at runtime.

Typically required when $n > 1000$.

Sparse GPs

Many options for sparse approximations

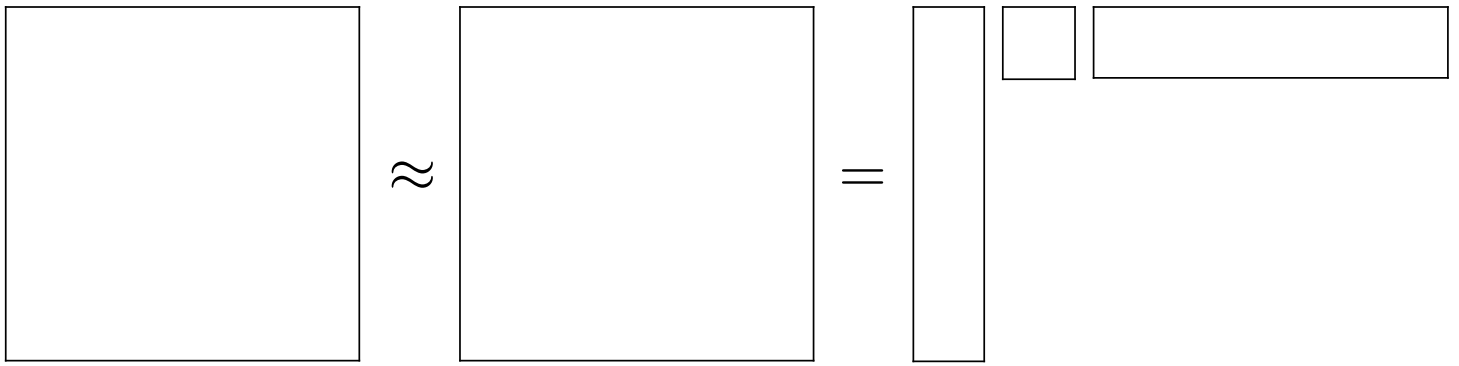
- Based on Inducing Variables
  - Subset of Data (SoD)
  - Subset of Regressors (SoR)
  - Deterministic Training Conditional (DTC)
  - Partially Independent Training Conditional (PITC)
  - Fully Independent Training Conditional (FITC)
- Fast Matrix Vector Multiplication (MVM)
- Variational Methods

See Quiñonero Candela and Rasmussen (2005) for an overview.

Sparse GPs

Sparse approximations where $\mathbf{f}$ are treated as latent variables

- a subset are treated exactly $|\boldsymbol{u}| = m$
- the other are given a computationally cheaper treatment
- avoids large matrix inversions



Hensman, GPSS '13

Inducing points are fixed or learned using greedy search / optimisation

Predicting and characterising user impact

500 million Tweets a day in Twitter

- important and some not so important information
- breaking news from media
- friends
- celebrity self promotion
- marketing
- spam

Can we automatically predict the impact of a user?

Can we automatically identify factors which influence user impact?

Defining user impact

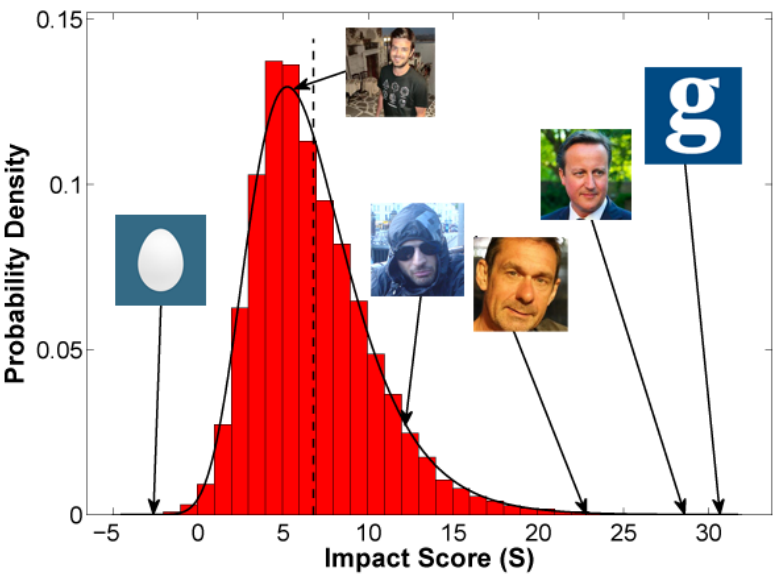
Define impact as a function of network connections

- no. of followers
- no. of followees
- no. of time the account is listed by others

$$\text{Impact} = ln\left(\frac{\text{listings}\cdot\text{followers}^2}{\text{followees}}\right)$$

Dataset

- 38.000 UK users
- all tweets from one year
- 48 million deduplicated messages



User controlled features

Only features under the user's control (e.g. not no. of retweets)

- User features (18)
  extracted from the account profile
  aggregated text features
- Text features (100)
  user's topic distribution
  topics computed using spectral clustering on the word co-occurrence (NPMI) matrix

Models

Regression task

- Gaussian Process regression model
- $n = 38000 \cdot 9/10$, use Sparse GPs with FITC
- Squared Exponential kernel (k-dimensional):

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}_p - \boldsymbol{x}_q)^T D(\boldsymbol{x}_p - \boldsymbol{x}_q)\right) + \sigma_n^2 \delta_{pq}$$

  where $D \in \mathbb{R}^{k \times k}$ is a symmetric matrix.
- if $D_{ARD} = \text{diag}(\boldsymbol{l})^{-2}$:

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{k}\frac{(\boldsymbol{x}_{pd} - \boldsymbol{x}_{pd})^2}{l_d^2}\right) + \sigma_n^2 \delta_{pq}$$

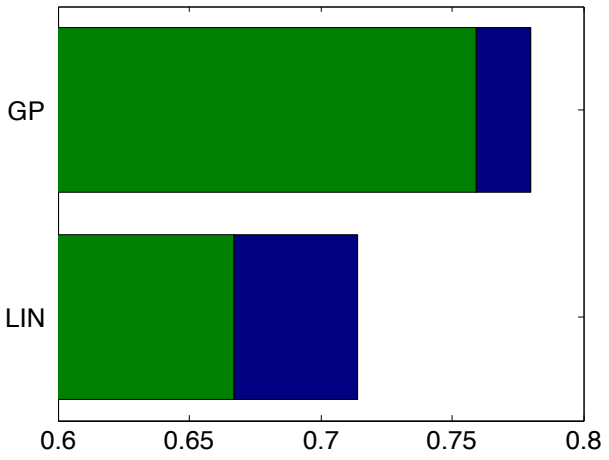
Automatic Relevance Determination (ARD)

- with $D = D_{ARD}$ the kernel is the SE kernel with automatic relevance determination (ARD), with the vector $\boldsymbol{l}$ denoting the characteristic length-scales of each feature
- l_d measures the distance for being uncorrelated along x_d
- $1/l_d^2$ proportional to how relevant a feature is: large length-scales means the covariance becomes independent of that feature value
- sorting by length-scales indicates which features impact the prediction the most
- tuning these parameters in done via Bayesian model selection

Prediction results

Experiments

- 10-fold cross validation
- using predictive mean
- baseline model is ridge regression **(LIN)**
- **Profile features**
- **Text features**



Pearson correlation

Conclusions

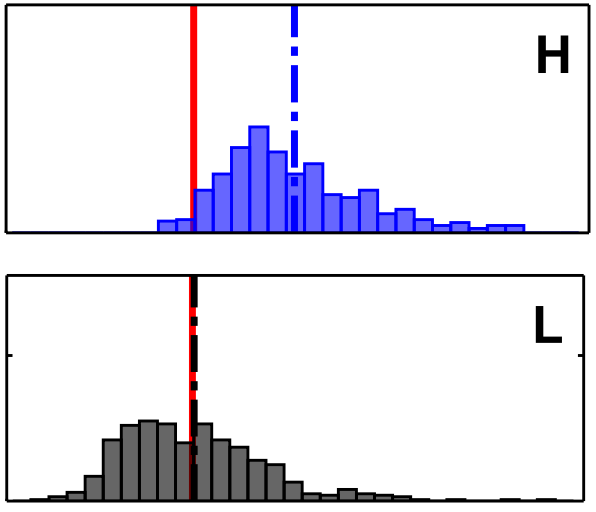
- GPs substantially better than ridge regression
- non-linear GPs with only profile features performs better than linear methods with all features
- GPs outperform SVR
- adding topic features improves all models

Selected features

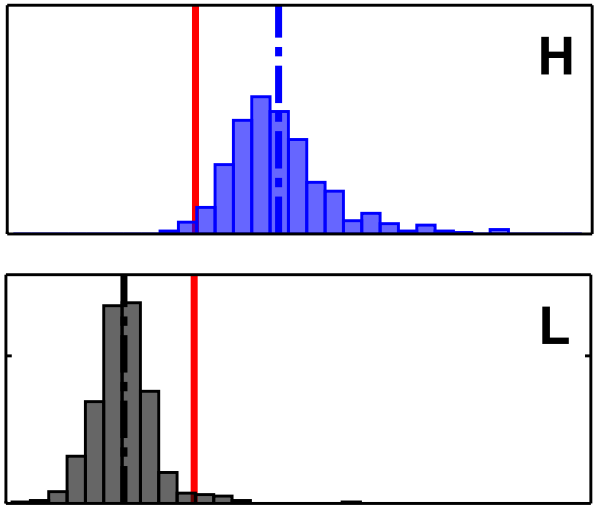
Feature	Importance
Using default profile image	0.73
Total number of tweets (entire history)	1.32
Number of unique @-mentions in tweets	2.31
Number of tweets (in dataset)	3.47
Links ratio in tweets	3.57
T1 (Weather): mph, humidity, barometer, gust, winds	3.73
T2 (Healthcare, Housing): nursing, nurse, rn, registered, bedroom, clinical, #news, estate, #hospital	5.44
T3 (Politics): senate, republican, gop, police, arrested, voters, robbery, democrats, presidential, elections	6.07
Proportion of days with non-zero tweets	6.96
Proportion of tweets with @-replies	7.10

Feature analysis

No. of unique @-mentions

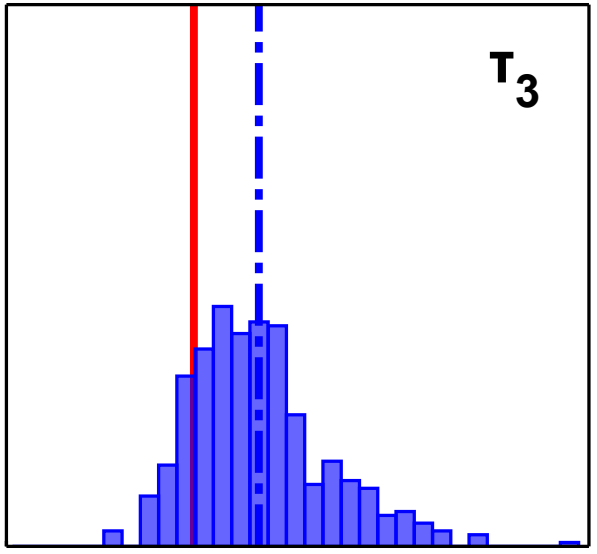


No. of tweets

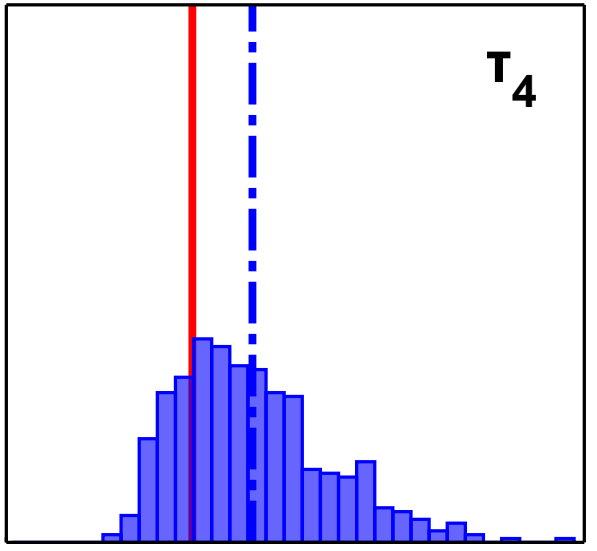


Impact histogram for users with **high (H)** values of this feature as opposed to **low (L)**. **Red line** is the mean impact score.

Feature analysis



damon, potter, #tvd, harry
elena, kate, portman,
pattinson, hermione,
jennifer



senate, republican, gop,
police, arrested, voters,
robbery, democrats,
presidential, elections

Impact histogram for users with **high (H)** values of this feature.
Red line is the mean impact score.

Conclusions

User impact is highly predictable

- user behaviour very informative
- “tips” for improving your impact

GP framework suitable

- non-linear modelling
- ARD feature selection
- sparse GPs allow large scale experiments
- empirical improvements over linear models & SVR

Outline

Case study: Temporal patterns of words

Categorising temporal patterns of hashtags in Twitter

- collect hashtag normalised frequency time series for months
- use models learnt on past frequencies to forecast future frequencies
- identify and group similar temporal patterns
- emphasise periodicities in word frequencies

Instance of a forecasting problem

- emphasis on forecasting (extrapolation)
- different effects modelled by specific kernels

See Preoţiuc-Pietro and Cohn (2013), EMNLP.

Model selection

Although parameter free, we still need to specify to a GP:

- the kernel parameters a.k.a. hyper-parameters θ
- the kernel definition $H_i \in \mathcal{H}$

Training a GP = selecting the kernel and its parameters

Can use only training data (and no validation)

Model selection

Although parameter free, we still need to specify to a GP:

- the kernel parameters a.k.a. hyper-parameters θ
- the kernel definition $H_i \in \mathcal{H}$

Training a GP = selecting the kernel and its parameters

Can use only training data (and no validation)

Use the marginal likelihood to optimise hyperparameters, and use this value to select the kernel

Identifying temporal patterns in word frequencies

Word/hashtag frequencies in Twitter

- very time dependent
- many 'live' only for hours reflecting timely events or memes
- some hashtags are constant over time
- some experience bursts at regular time intervals
- some follow human activity cycles

Can we automatically forecast future hashtag frequencies?

Can we automatically categorise temporal patterns?

Twitter hashtag temporal patterns

Regression task

- Extrapolation: forecast future frequencies
- using predictive mean

Dataset

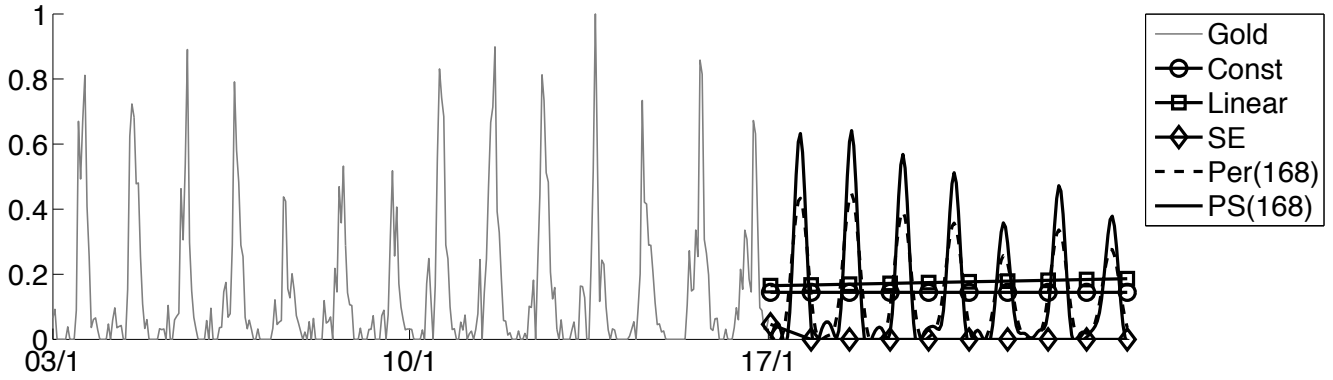
- two months of Twitter Gardenhose (10%)
- first month for training, second month for testing
- 1176 hashtags occurring in both splits
- $\sim$ 6.5 million tweets
- 5456 tweets/hashtag

Kernels

The kernel

- induces the covariance in the response between pairs of data points
- encodes the prior belief on the type of function we aim to learn
- for extrapolation, kernel choice is paramount
- different kernels are suitable for each specific category of temporal patterns: isotropic, smooth, periodic, non-stationary, etc.

Kernels



#goodmorning

	Const	**Linear**	**SE**	**Per**	**PS**
NLML	-41	-34	-176	-180	**-192**
NRMSE	0.213	0.214	0.262	0.119	**0.107**

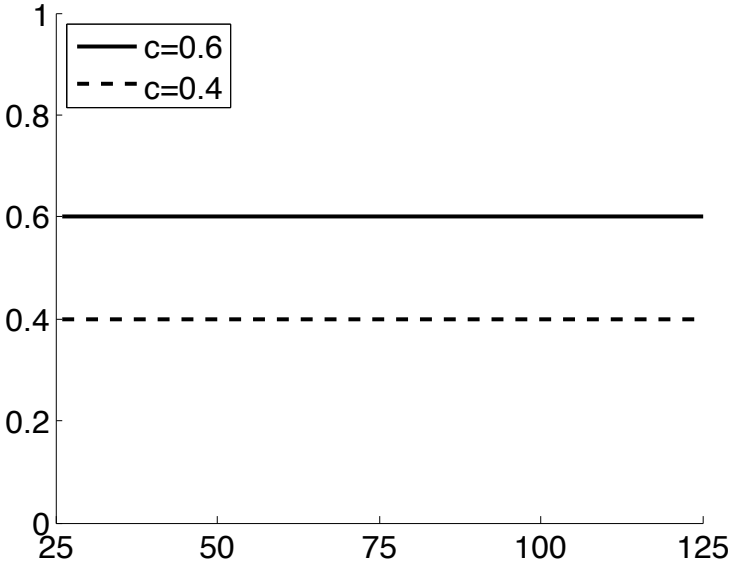
Lower is better

Use Bayesian model selection techniques to choose between kernels

Kernels: Constant

$$k_C(x, x') = c$$

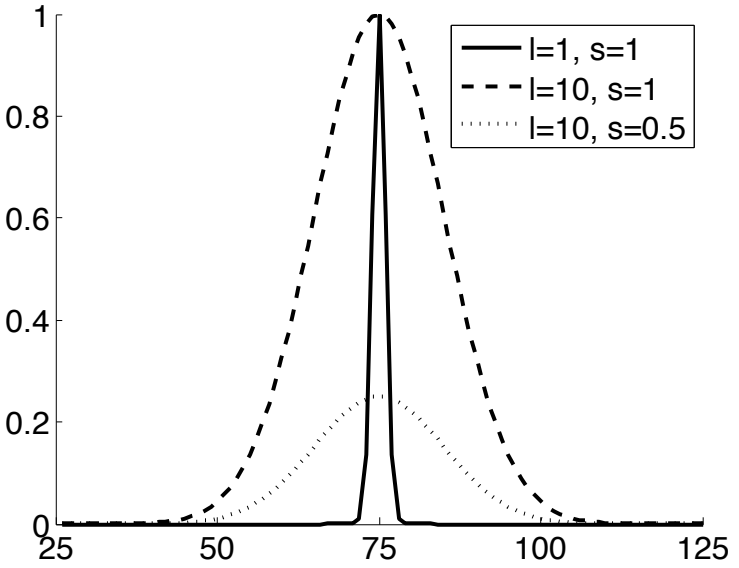
- constant relationship between outputs
- predictive mean is the value c
- assumes signal is modelled by Gaussian noise centred around the value c



Kernels: Squared exponential

$$k_{SE}(x, x') = s^2 \cdot \exp\left(-\frac{(x - x')^2}{2l^2}\right)$$

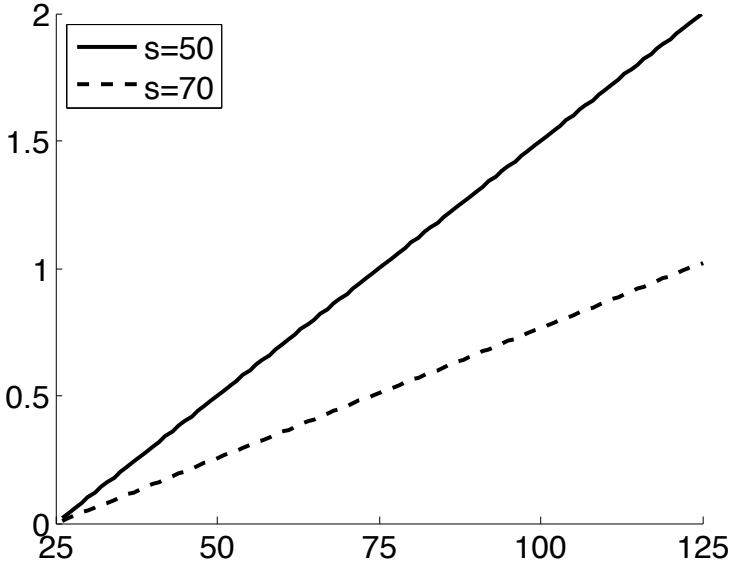
- smooth transition between neighbouring points
- best describes time series with a smooth shape e.g. uni-modal burst with a steady decrease
- predictive variance increases exponentially with distance



Kernels: Linear

$$k_{Lin}(x, x') = \frac{|x \cdot x'| + 1}{s^2}$$

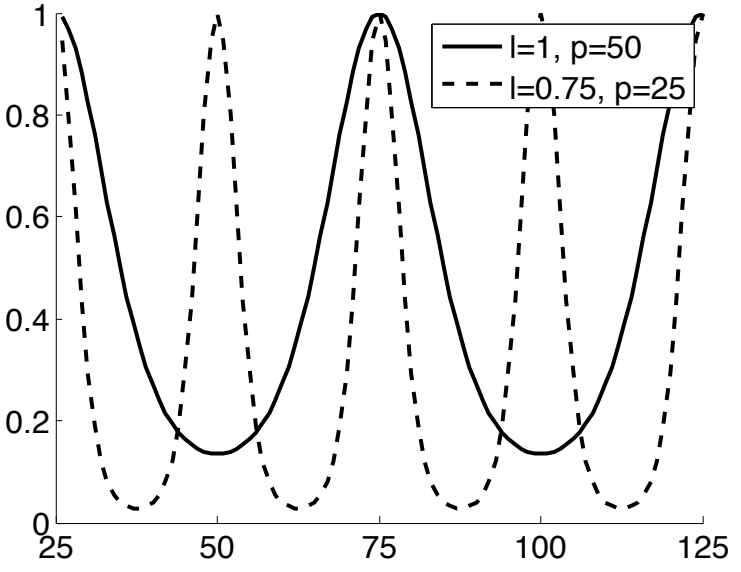
- non-stationary kernel: covariance depends on the data points values, not only on their difference $|t - t'|$
- equivalent to Bayesian linear regression with $\mathcal{N}(0, 1)$ priors on the regression weights and a prior of $\mathcal{N}(0, s^2)$ on the bias



Kernels: Periodic

$$k_{PER}(x, x') = s^2 \cdot \exp \cdot \left(-\frac{2 \sin^2(2\pi(x - x')/p)}{l^2} \right)$$

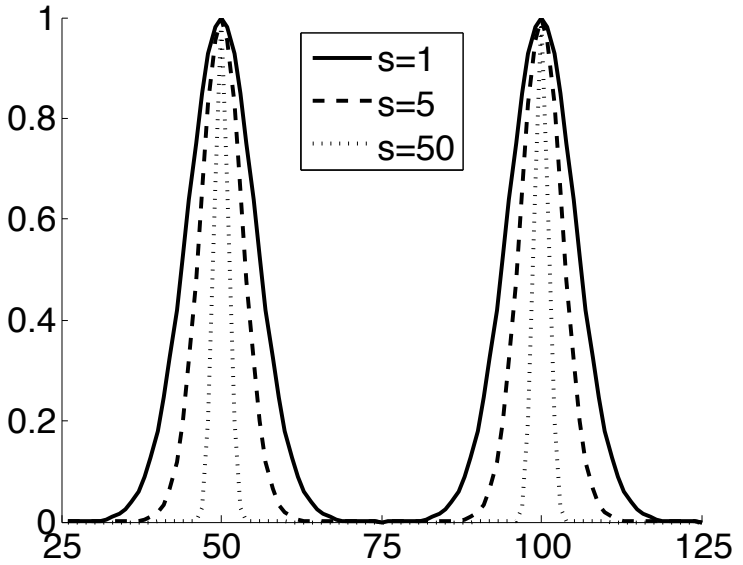
- s and l are characteristic length-scales
- p is the period (distance between consecutive peaks)
- best describes periodic patterns that oscillate smoothly between high and low values



Kernels: Periodic Spikes

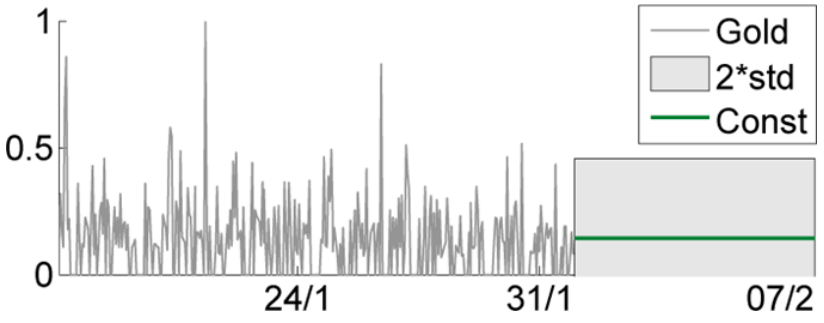
$$k_{PS}(x, x') = \cos\left(\sin\left(\frac{2\pi \cdot (x - x')}{p}\right)\right) \cdot \exp\left(\frac{s\cos(2\pi \cdot (x - x'))}{p} - s\right)$$

- p is the period
- s is a shape parameter controlling the width of the spike
- best describes time series with constant low values, followed by abrupt periodic rise

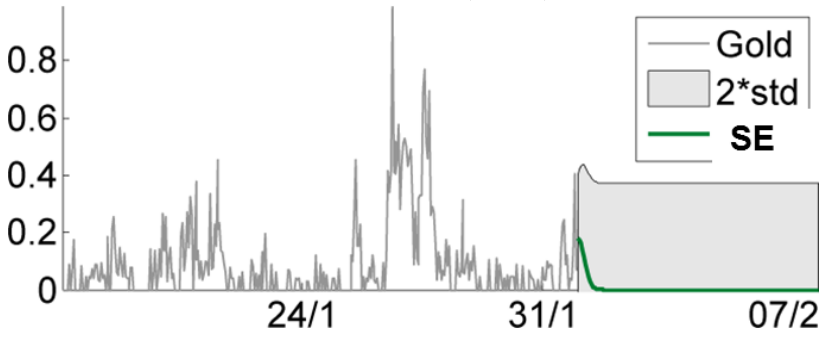


Results: Examples

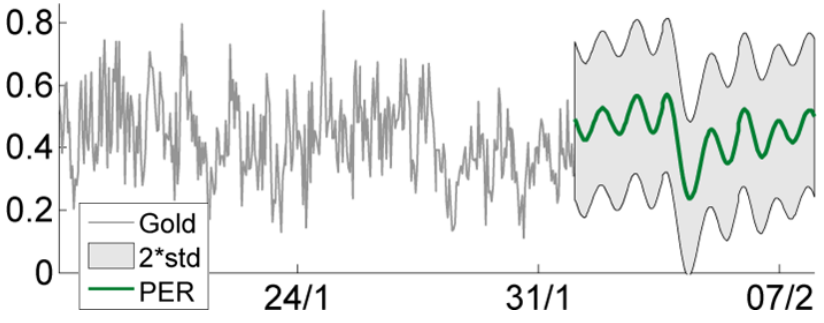
#fyi (Const)



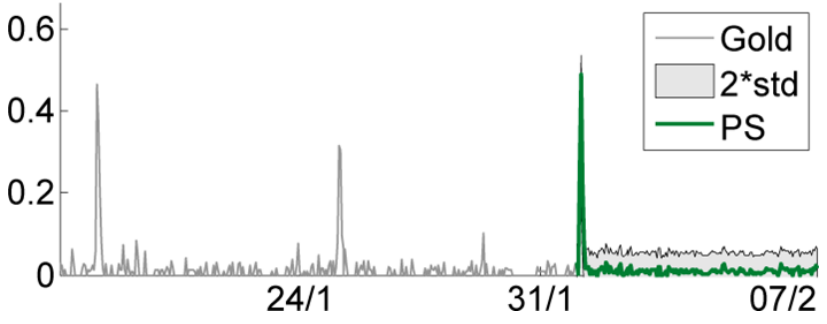
#snow (SE)



#fail (Per)



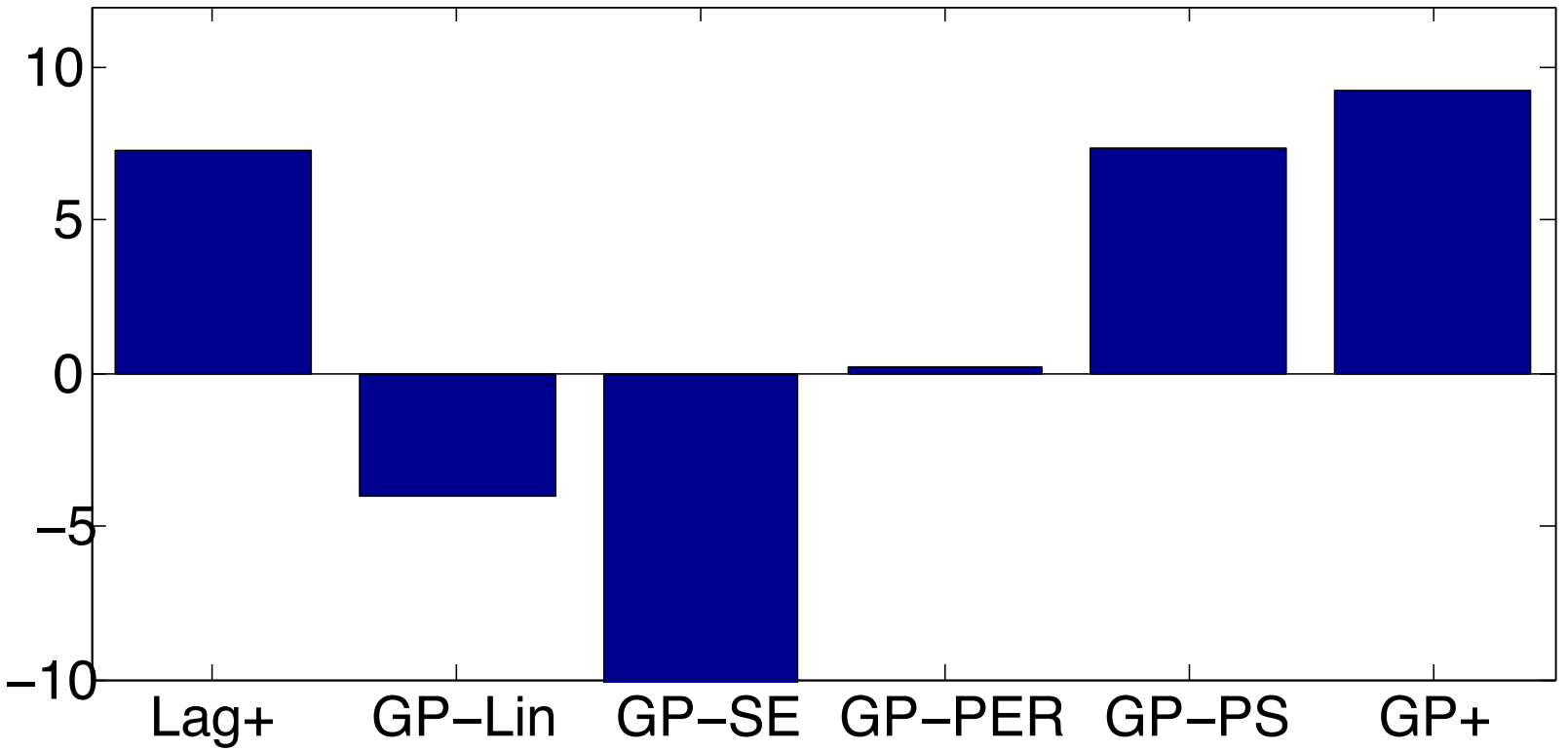
#raw (PS)



Results: Categories

Const	SE	PER	PS
#funny	#2011	#brb	#ff
#lego	#backintheday	#coffee	#followfriday
#likeaboss	#confessionhour	#facebook	#goodnight
#money	#februarywish	#facepalm	#jobs
#nbd	#haiti	#fail	#news
#nf	#makeachange	#love	#nowplaying
#notetoself	#questionsidontlike	#rock	#tgif
#priorities	#savelibraries	#running	#twitterafterdark
#social	#snow	#xbox	#twitteroff
#true	#snowday	#youtube	#ww
49	**268**	**493**	**366**

Results: Forecasting



Application: Text classification

Task

- assign the hashtag of a given tweet based on its text

Methods

- Most frequent (MF)
- Naive Bayes model with empirical prior (NB-E)
- Naive Bayes with GP forecast as prior (NB-P)

	MF	**NB-E**	**NB-P**
Match@1	7.28%	16.04%	**17.39%**
Match@5	19.90%	29.51%	**31.91%**
Match@50	44.92%	59.17%	**60.85%**
MRR	0.144	0.237	**0.252**

Higher is better

Outline

Case study 1: MT Quality Estimation

Manual assessment of translation quality given source and translated texts.

'Quality' can be measured many ways (see Specia et al. (2009))

- subjective scoring (1-5) for fluency, adequacy, perceived effort to correct
- post-editing effort: HTER or time taken
- binary judgements, ranking, . . .

Human judgements are highly subjective, biased, noisy

- typing speed
- experience levels
- expectations from MT

See Cohn and Specia (2013), ACL.

General annotation problem

MT Quality Estimation is an instance of a general annotation problem

- can't rely on single individual
- but many annotators produce different results
- how can we resolve conflicts?

Previous work in MT QE

- averaged responses from several annotators (Callison-Burch et al., 2012); or
- used output of single annotator (Koponen et al., 2012)

More appropriate to model as a multi-task problem.

Multi-task learning

Multi-task learning

- form of transfer learning
- several related tasks sharing the same input data representation
- learn the types, extent of correlations

Compared to domain adaptation

- tasks need not be identical (even regression vs classification)
- no explicit 'target' domain
- several sources of variation besides domain
- no assumptions of data asymmetry

Multi-task learning for MT Quality Estimation

Modelling individual annotators

- each bring own biases
- but correlated decisions with others' annotations
- could even find clusters of common solutions

Here use multi-output GP regression

Multi-task learning for MT Quality Estimation

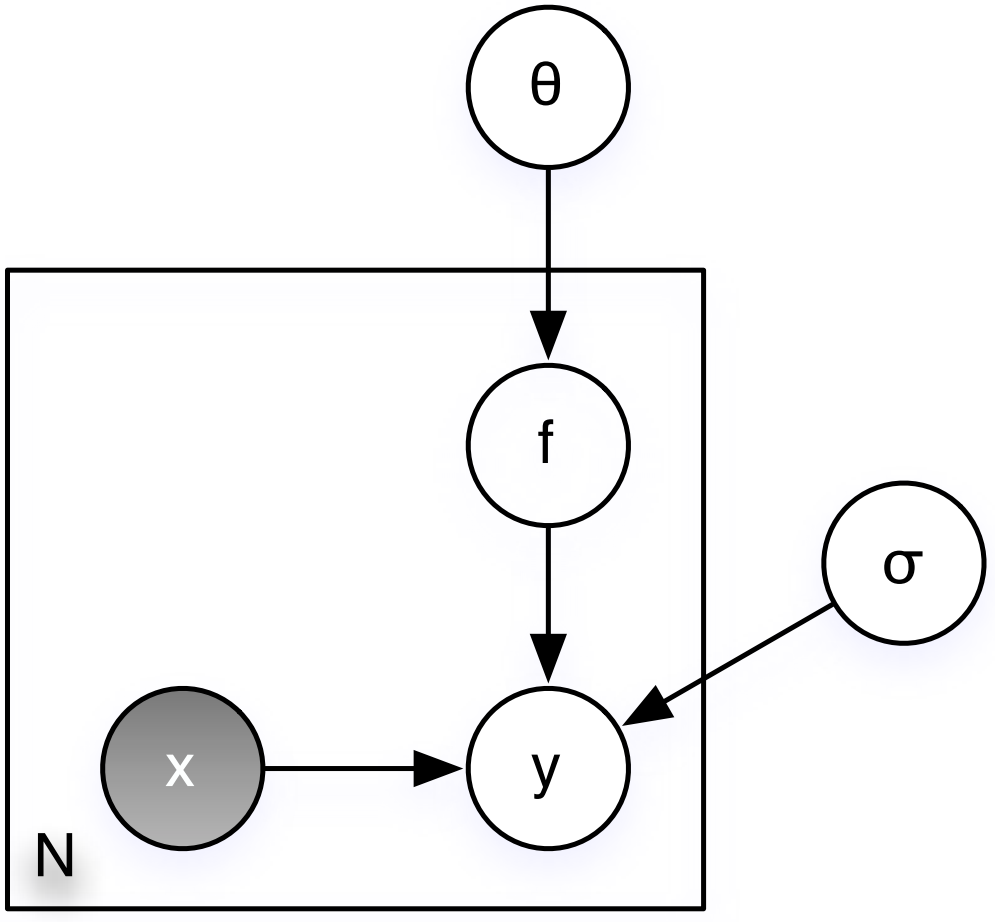
Modelling individual annotators

- each bring own biases
- but correlated decisions with others' annotations
- could even find clusters of common solutions

Here use multi-output GP regression

- joint inference over several translators
- learn degree of inter-task transfer
- learn per-translator noise
- incorporate task meta-data

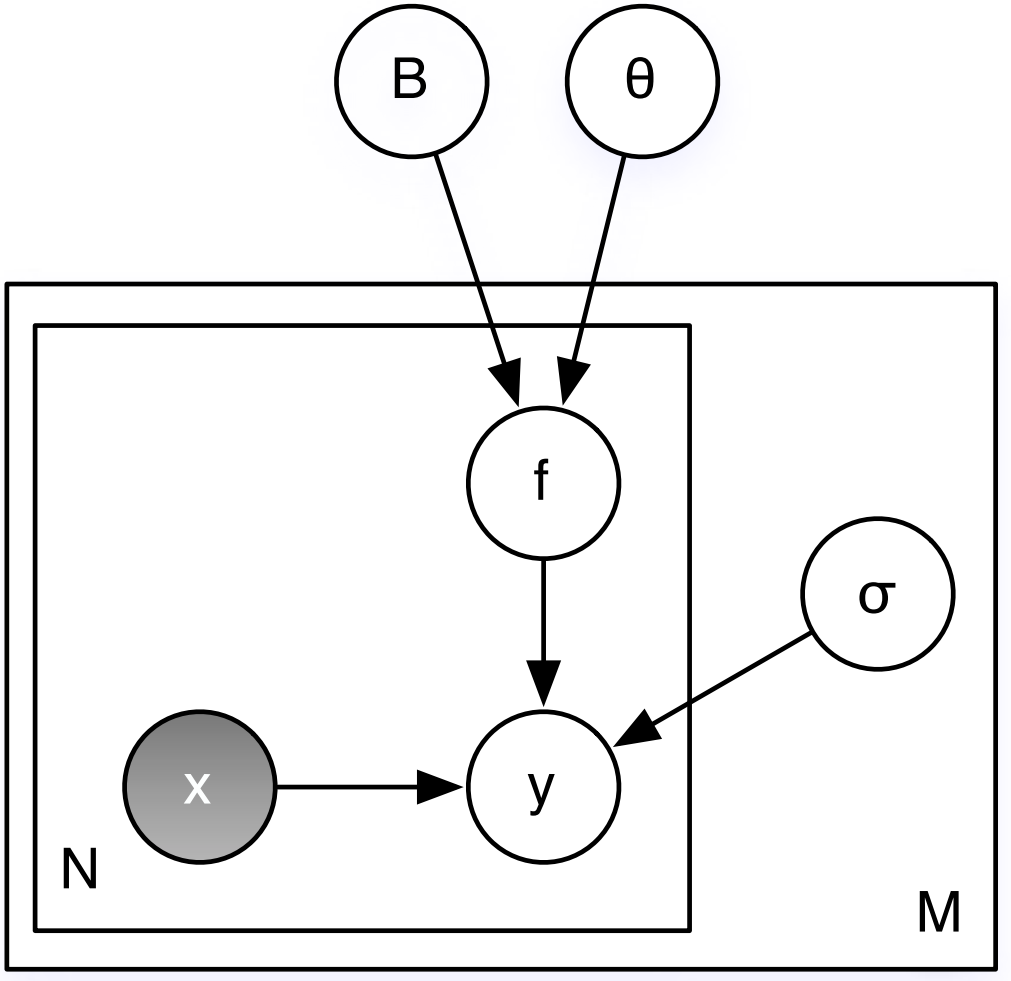
Review: GP Regression



$$\mathbf{f} \sim \mathrm{GP}(\mathbf{0}, \theta)$$

$$y_i \sim \mathrm{N}(f(\mathbf{x}_i), \sigma^2)$$

Multi-task GP Regression



$$\mathbf{f} \sim \mathrm{GP}(\mathbf{0}, \left(B, \theta\right))$$

$$y_{im} \sim \mathrm{N}(f_m(\mathbf{x}_i), \sigma_m^2)$$

See Alvarez et al. (2011).

Multi-task Covariance Kernels

Represent data as $(\mathbf{x}, t, y)$ tuples, where t is a task identifier.
Define a *separable* covariance kernel,

$$K(\mathbf{x}, \mathbf{x}')_{t,t'} = B_{t,t'} k_\theta(\mathbf{x}, \mathbf{x}') + \text{noise}$$

- effectively each input augmented with t, indexing the task of interest
- the coregionalisation matrix, $\mathbf{B} \in \mathcal{R}^{M \times M}$ weights inter-task covariance
- the data kernel k_θ takes data points $\mathbf{x}$ as input
  e.g., squared exponential

Coregionalisation Kernels

Generally $\mathbf{B}$ can be any symmetric positive semi-definite matrix. Some interesting choices

Coregionalisation Kernels

Generally $\mathbf{B}$ can be any symmetric positive semi-definite matrix. Some interesting choices

- $\mathbf{B} = \mathbf{I}$ encodes independent learning

Coregionalisation Kernels

Generally $\mathbf{B}$ can be any symmetric positive semi-definite matrix. Some interesting choices

- $\mathbf{B} = \mathbf{I}$ encodes independent learning
- $\mathbf{B} = \mathbf{1}$ encodes pooled learning

Coregionalisation Kernels

Generally $\mathbf{B}$ can be any symmetric positive semi-definite matrix. Some interesting choices

- $\mathbf{B} = \mathbf{I}$ encodes independent learning
- $\mathbf{B} = \mathbf{1}$ encodes pooled learning
- interpolating the above

Coregionalisation Kernels

Generally $\mathbf{B}$ can be any symmetric positive semi-definite matrix. Some interesting choices

- $\mathbf{B} = \mathbf{I}$ encodes independent learning
- $\mathbf{B} = \mathbf{1}$ encodes pooled learning
- interpolating the above
- full rank $\mathbf{B} = \mathbf{W}\mathbf{W}^\top$, or low rank variants

Coregionalisation Kernels

Generally $\mathbf{B}$ can be any symmetric positive semi-definite matrix. Some interesting choices

- $\mathbf{B} = \mathbf{I}$ encodes independent learning
- $\mathbf{B} = \mathbf{1}$ encodes pooled learning
- interpolating the above
- full rank $\mathbf{B} = \mathbf{W}\mathbf{W}^\top$, or low rank variants

Known as the Intrinsic model of coregionalisation (IMC).

See Alvarez et al. (2011); Bonilla et al. (2008)

Stacking and Kronecker products

Response variables are a matrix

$$\mathbf{Y} = \mathcal{R}^{N \times M}$$

Stacking and Kronecker products

Response variables are a matrix

$$\mathbf{Y} = \mathcal{R}^{N \times M}$$

Represent data in 'stacked' form

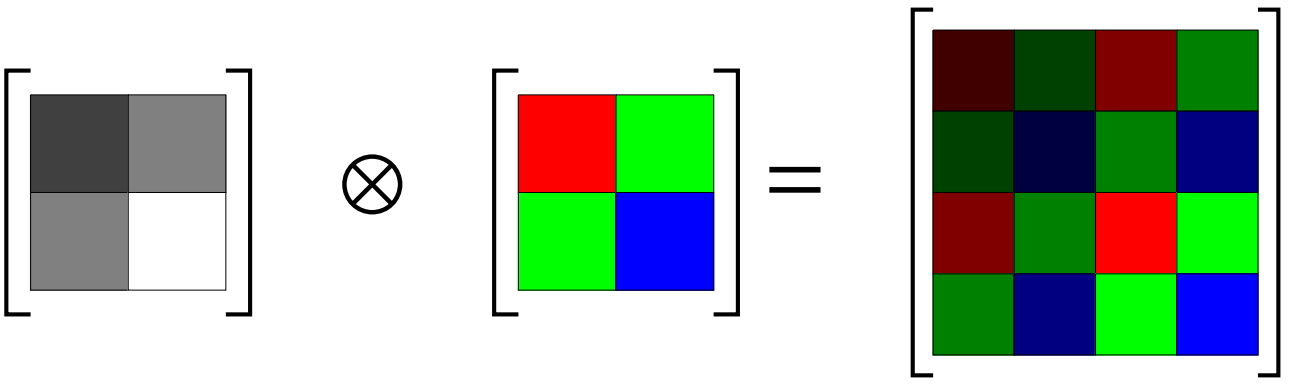
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \\ \vdots \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{N1} \\ \vdots \\ y_{1M} \\ y_{2M} \\ \vdots \\ y_{NM} \end{bmatrix}$$

Kernel a Kronecker product $\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k_{\text{data}}(\mathbf{X}_o, \mathbf{X}_o)$

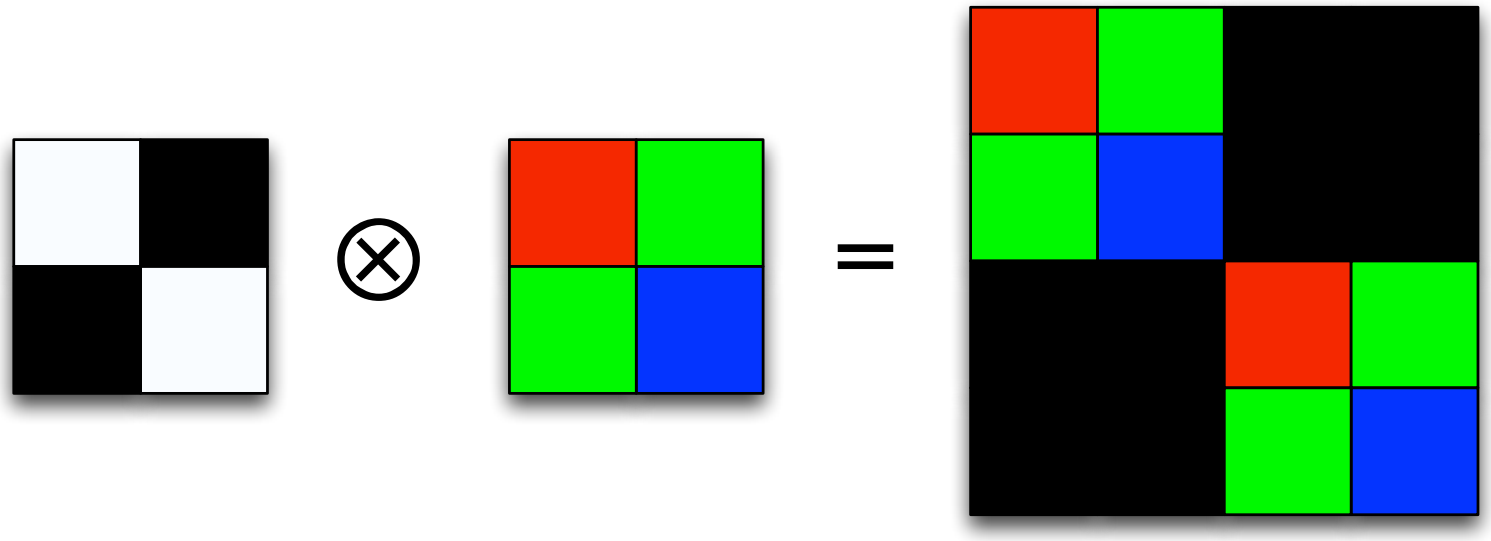
Kronecker product

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \mathbf{K} = \begin{bmatrix} a\mathbf{K} & b\mathbf{K} \\ c\mathbf{K} & d\mathbf{K} \end{bmatrix}$$

Kronecker product

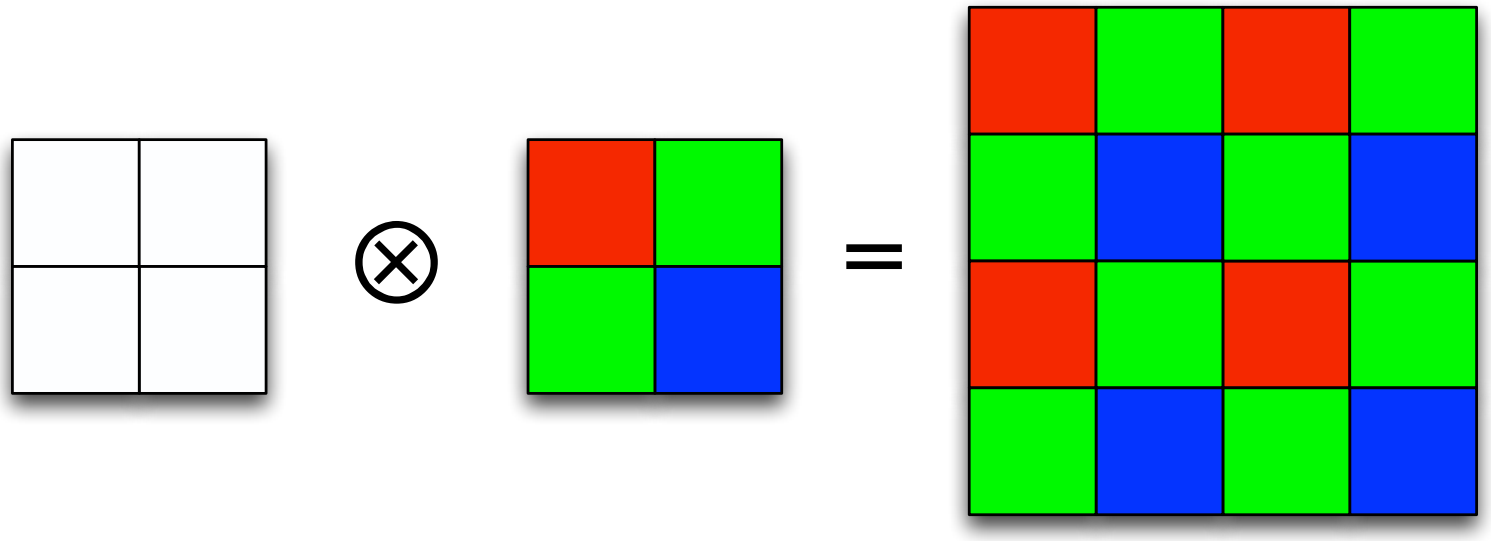


Choices for B: Independent learning



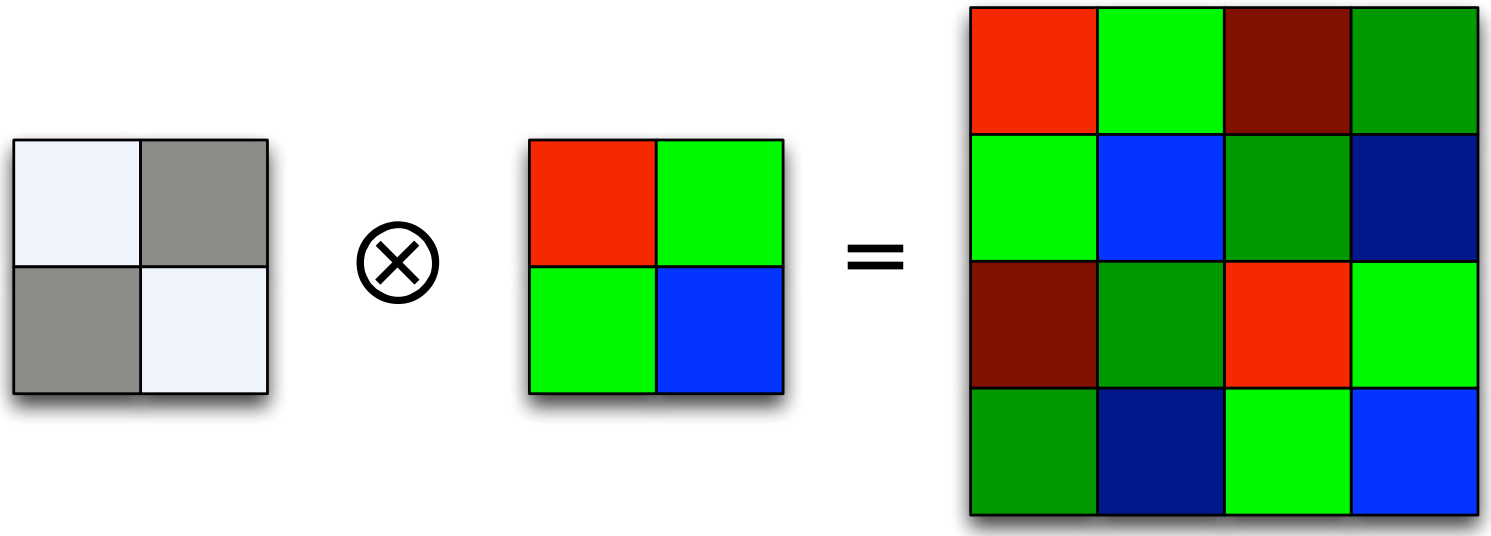
$$\mathbf{B} = \mathbf{I}$$

Choices for *B*: Pooled learning



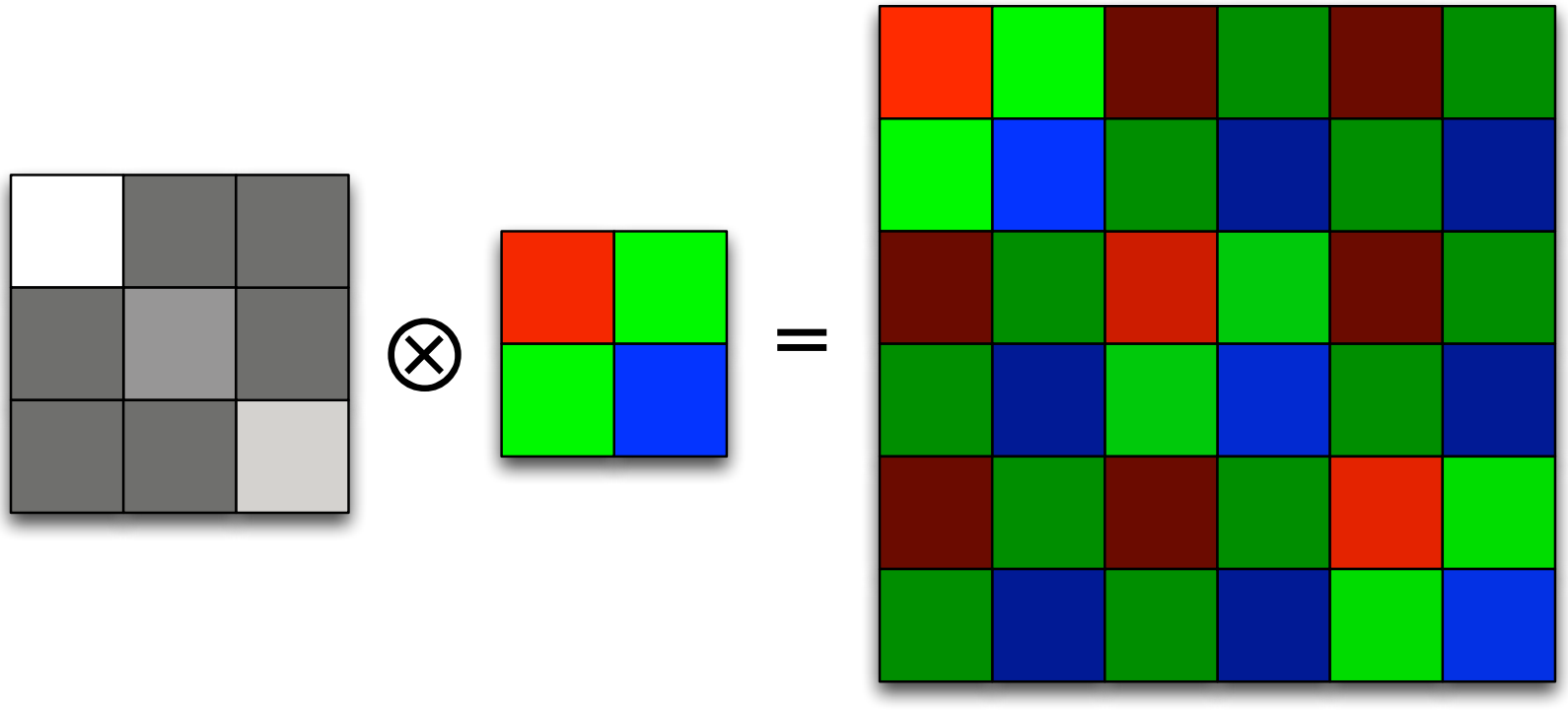
B = 1

Choices for B: Interpolating independent and pooled learning



$$\mathbf{B} = \mathbf{1} + \alpha \mathbf{I}$$

Choices for B: Interpolating independent and pooled learning II



$$\mathbf{B} = \mathbf{1} + \mathrm{diag}(\alpha)$$

Compared to Daumé III (2007)

Feature augmentation approach to multi-task learning. Uses horizontal data stacking:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^{(1)} & \mathbf{X}^{(1)} & \mathbf{0} \\ \mathbf{X}^{(2)} & \mathbf{0} & \mathbf{X}^{(2)} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \end{bmatrix}$$

where $(\mathbf{X}^{(i)}, \mathbf{y}^{(i)})$ are the training data for task i. This expands the feature space by a factor of M.

Equivalent to a multitask kernel

$$k(\mathbf{x}, \mathbf{x}')_{t,t'} = (1 + \delta(t, t'))\, \mathbf{x}^\top \mathbf{x}'$$
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = (\mathbf{1} + \mathbf{I}) \otimes k_{\text{linear}}(\mathbf{X}, \mathbf{X})$$

$\Rightarrow$ A specific choice of $\mathbf{B}$ with a linear data kernel

Compared to Evgeniou et al. (2006)

In the regularisation setting, Evgeniou et al. (2006) show that the kernel

$$K(\mathbf{x}, \mathbf{x}')_{t,t'} = (1 - \lambda + \lambda M \delta(t, t'))\, \mathbf{x}^\top \mathbf{x}'$$

is equivalent to a linear model with regularisation term

$$J(\Theta) = \frac{1}{M} \left(\sum_t \|\theta_t\|^2 + \frac{1-\lambda}{\lambda} \|\theta_t - \frac{1}{M} \sum_{t'} \theta_{t'}\|^2 \right)$$

This regularises each task's parameters θ_t towards the mean parameters over all tasks, $\frac{1}{M} \sum_{t'} \theta_{t'}$.

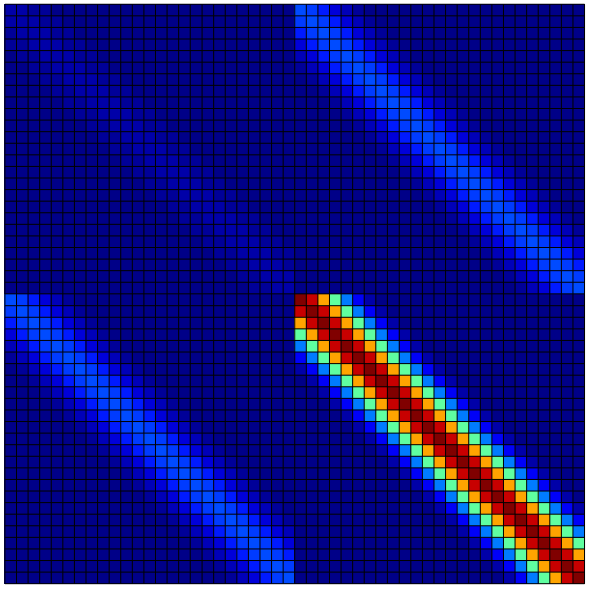
A form of *interpolation* method from before.

ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$

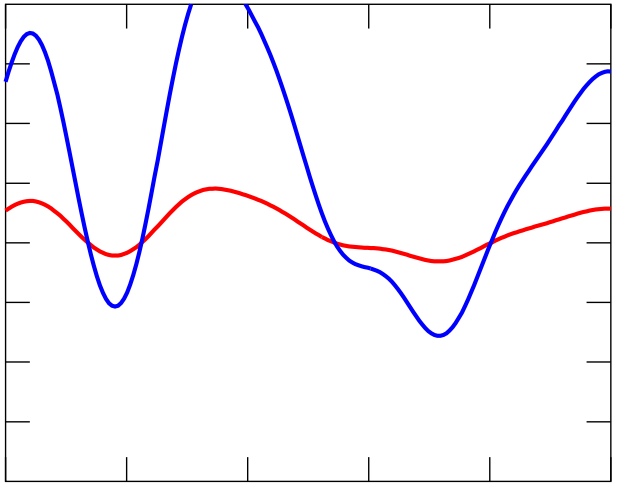


ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$

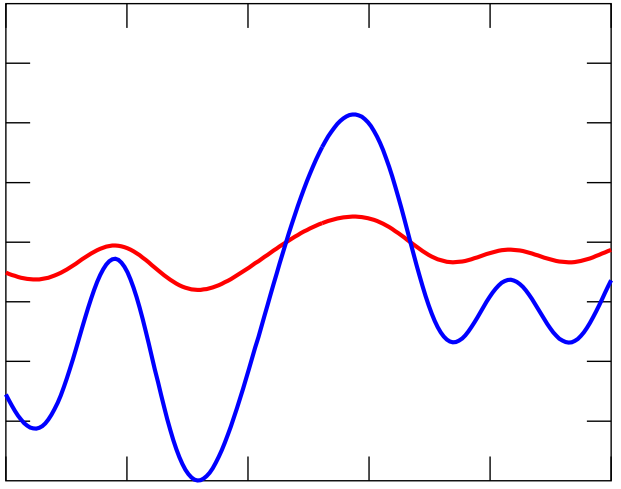


ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$

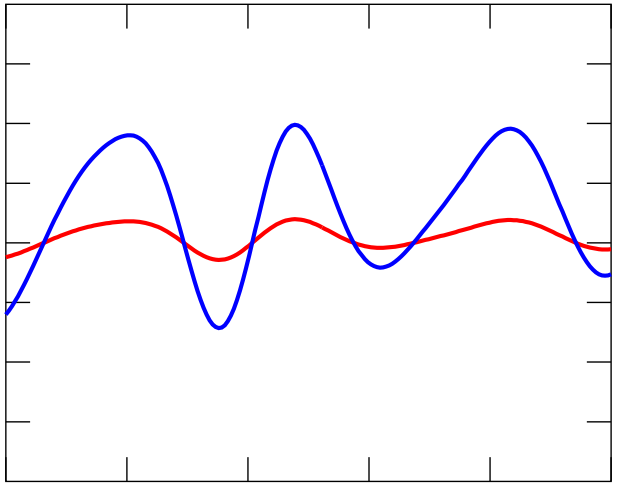


ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

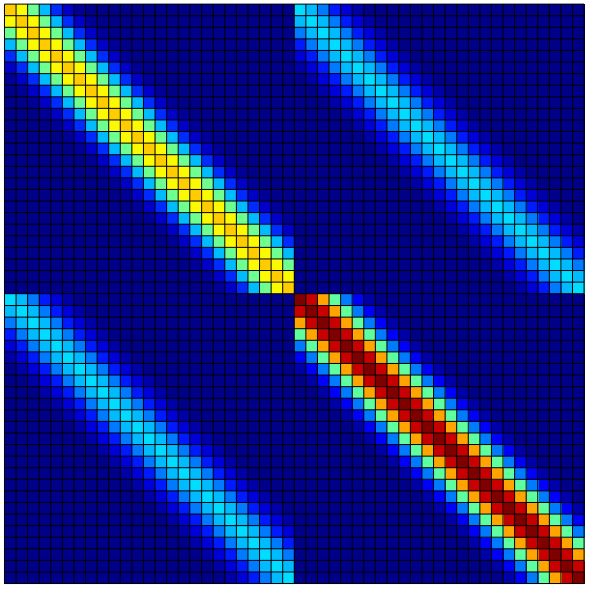
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

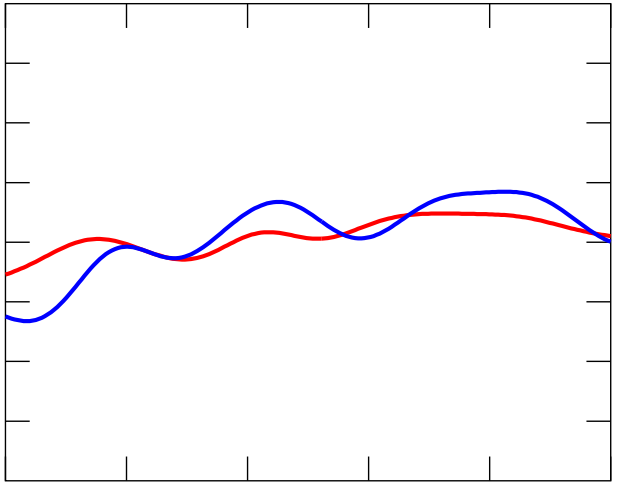
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

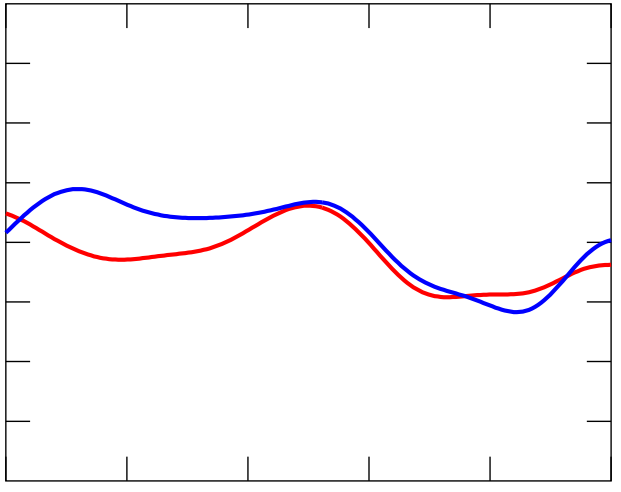
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

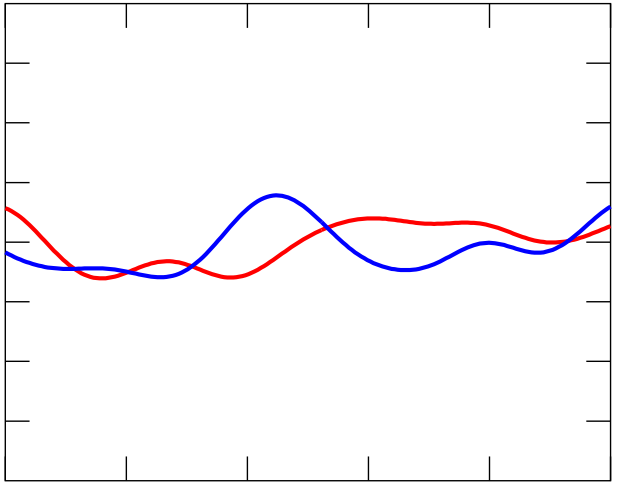
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

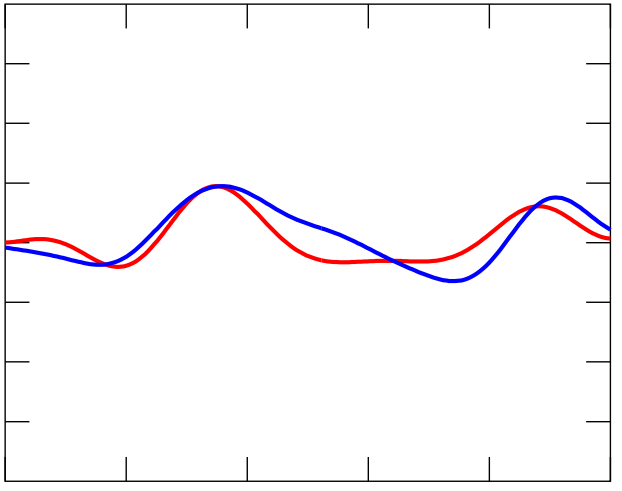
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



ICM samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Experimental setup

Quality Estimation data

- 2k examples of source sentence and MT output
- measuring subjective post-editing (1-5) WMT12
- post-editing time per word, in log seconds WPTP12
- 17 dense features extracted using
  Quest toolkit (Specia et al., 2013)
- using official train/test split, or random assignment

Experimental setup

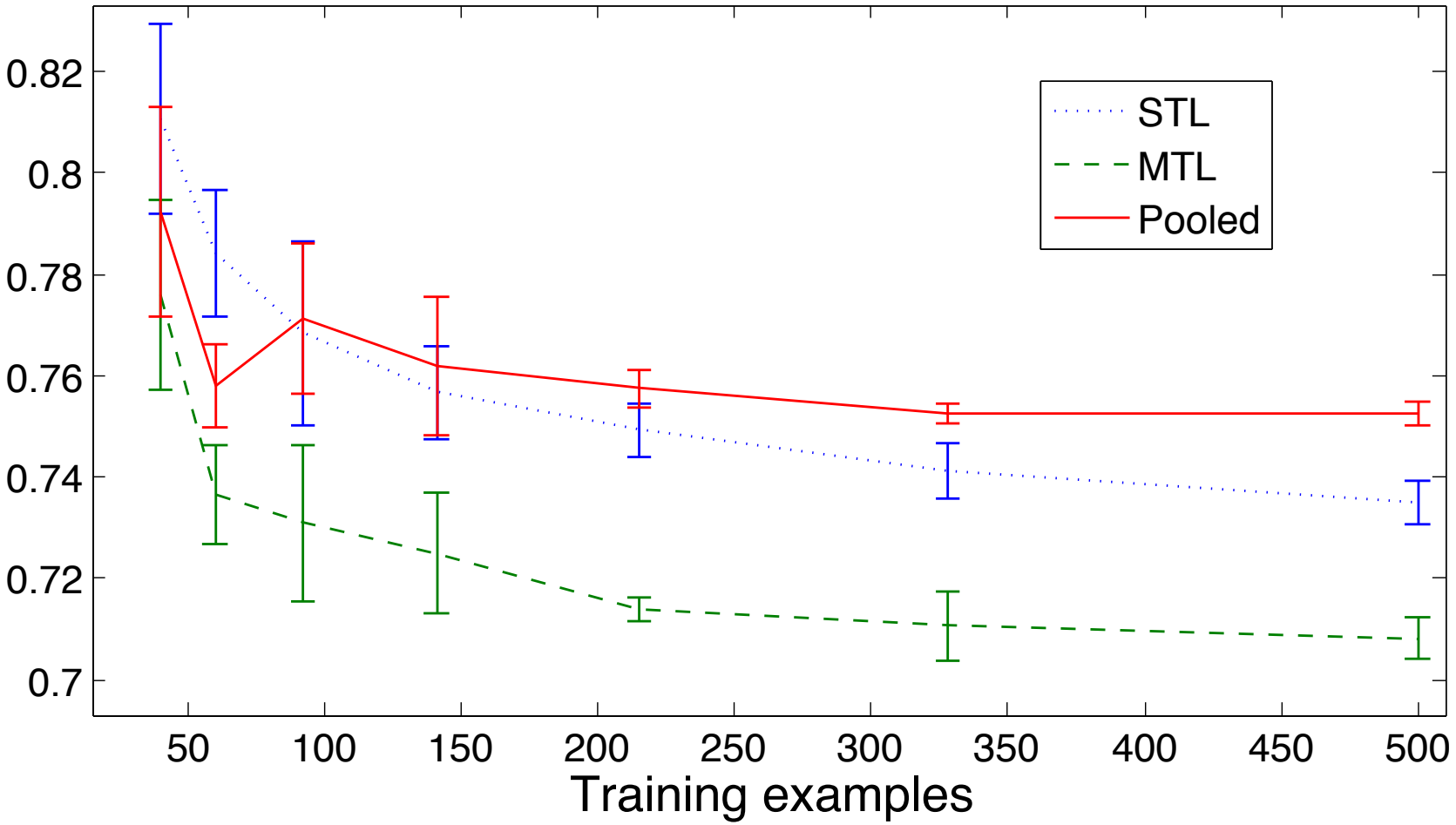
Quality Estimation data

- 2k examples of source sentence and MT output
- measuring subjective post-editing (1-5) WMT12
- post-editing time per word, in log seconds WPTP12
- 17 dense features extracted using Quest toolkit (Specia et al., 2013)
- using official train/test split, or random assignment

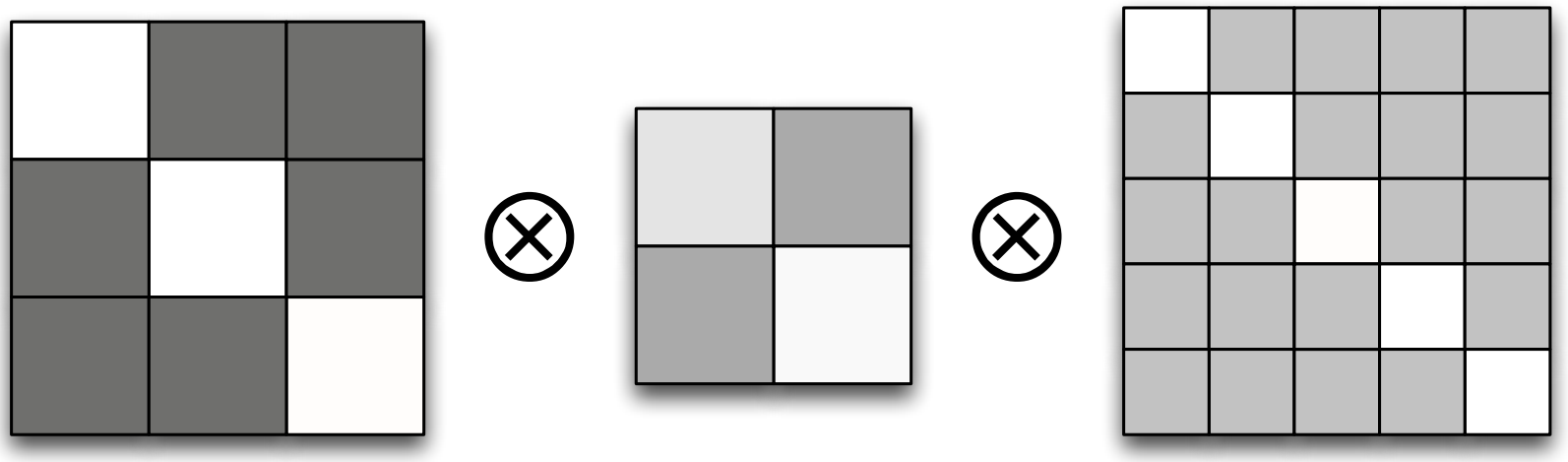
Gaussian Process models

- squared exponential data kernel (RBF)
- hyper-parameter values trained using type II MLE
- consider simple interpolation coregionalisation kernels
- include per-task noise or global tied noise

Results: WMT12 RMSE for 1-5 ratings



Incorporating layers of task metadata

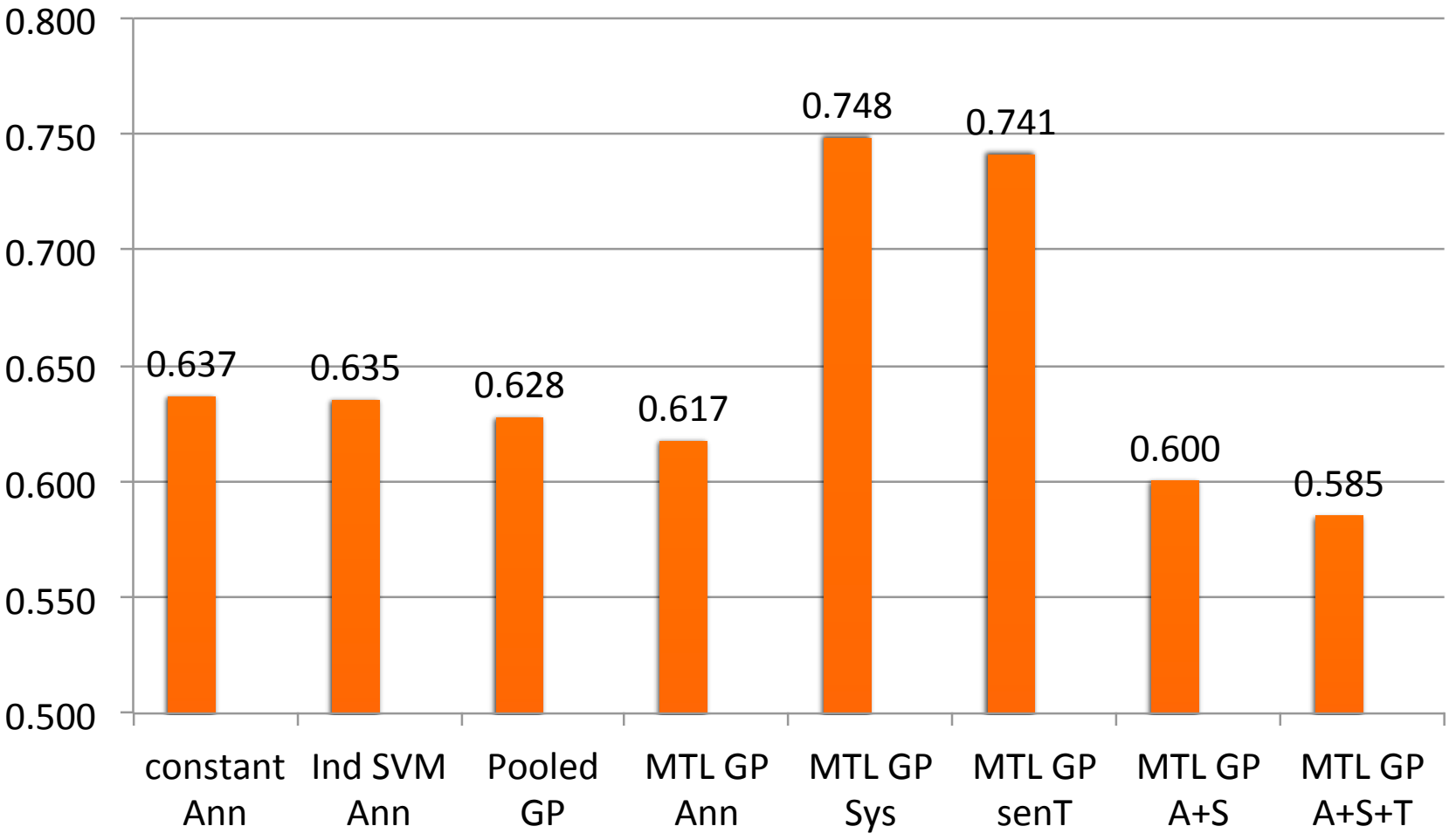


Annotator

System

Source
Sen**T**ence

Results: WPTP12 RMSE post-editing time



Case Study 2: Emotion Analysis

- automatically detect emotions in a text
- fine-grained
- presence of anti-correlations (*sadness* and *joy*)

Headline	**Fear**	**Joy**	**Sadness**
Storms kill, knock out power, cancel flights	82	0	60
Panda cub makes her debut	0	59	0

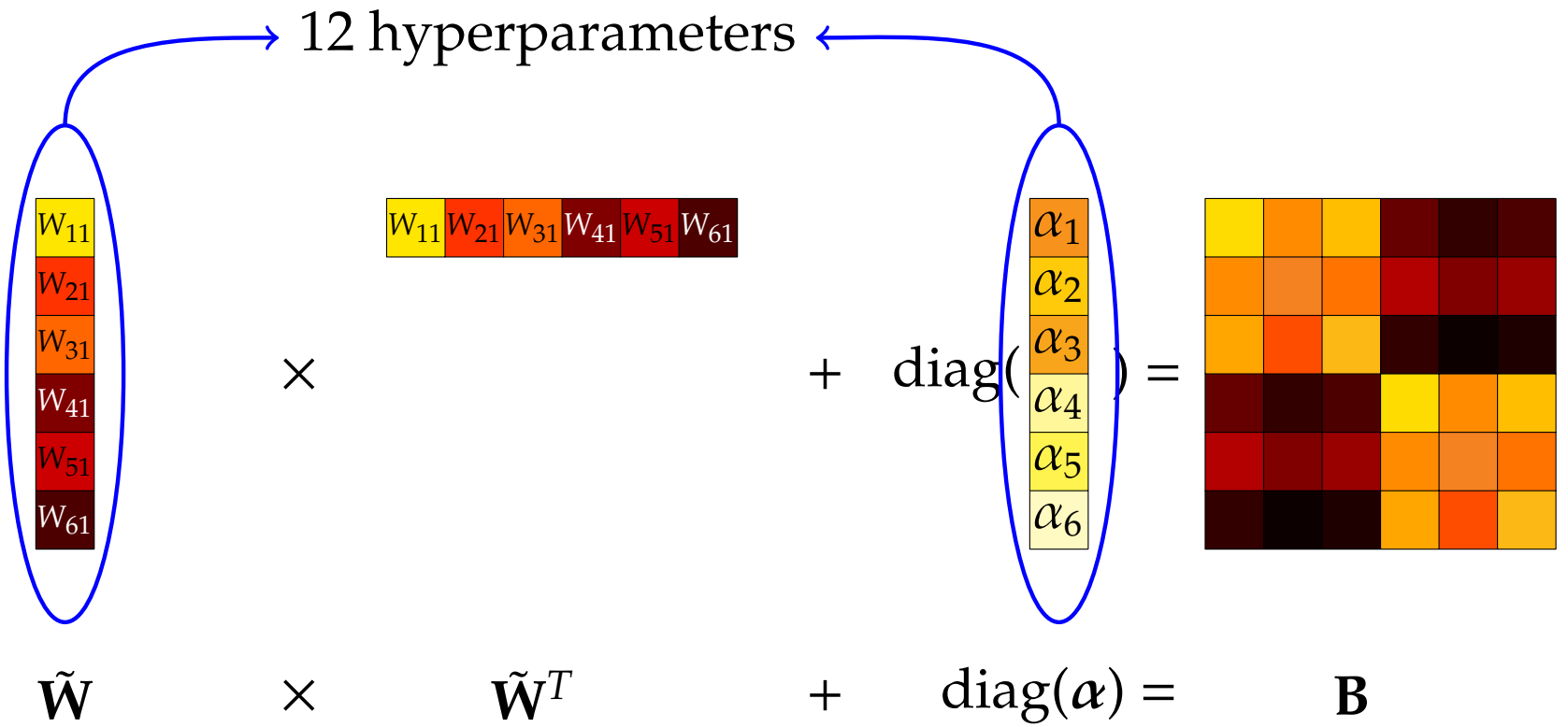
See Beck et al. (2014), EMNLP

Modelling anti-correlations

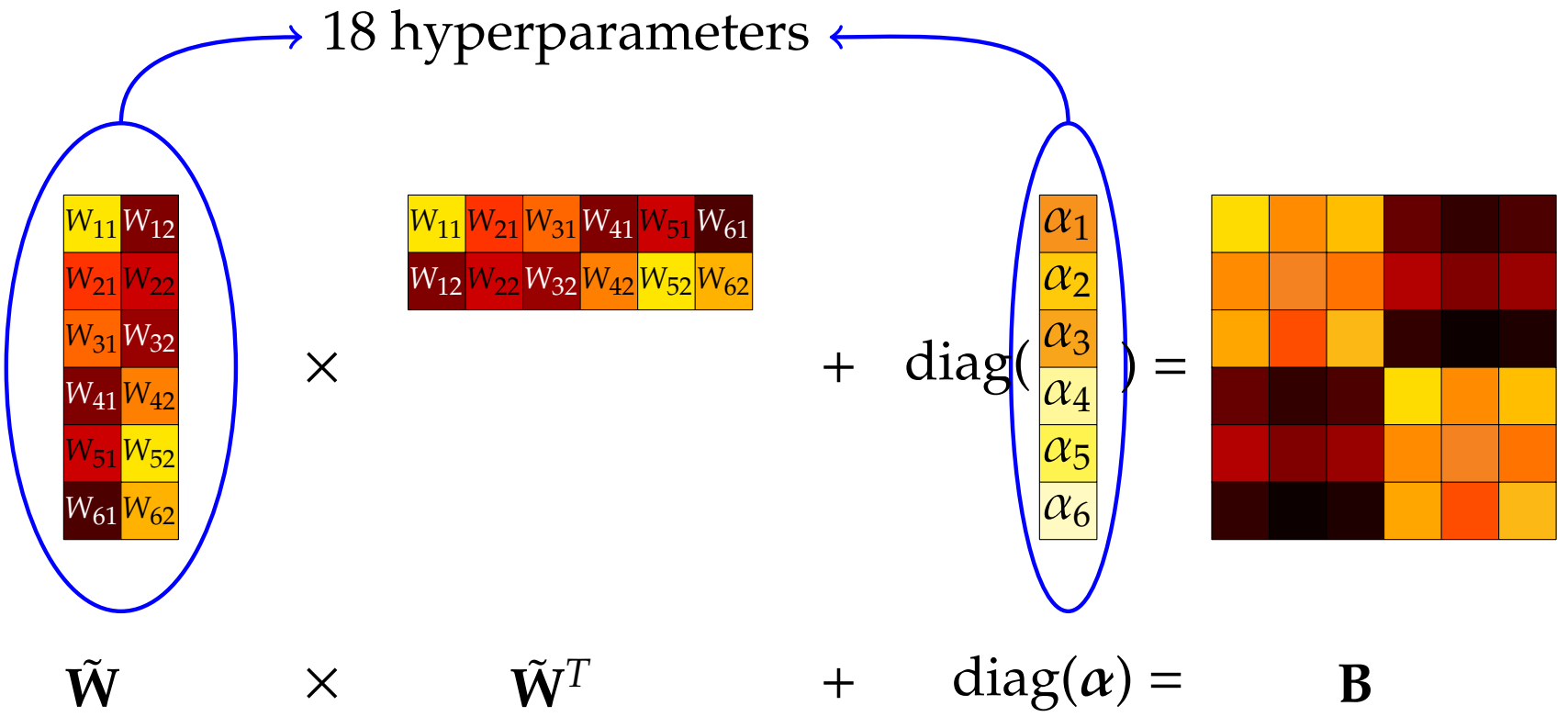
- The coregionalization settings used in Quality Estimation are not suitable for this problem: they assume positive covariances between tasks.
- We need a parameterization that allows B to have negative covariances: we use the incomplete-Cholesky decomposition.

$$\mathbf{B} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^T + \text{diag}(\boldsymbol{\alpha})$$

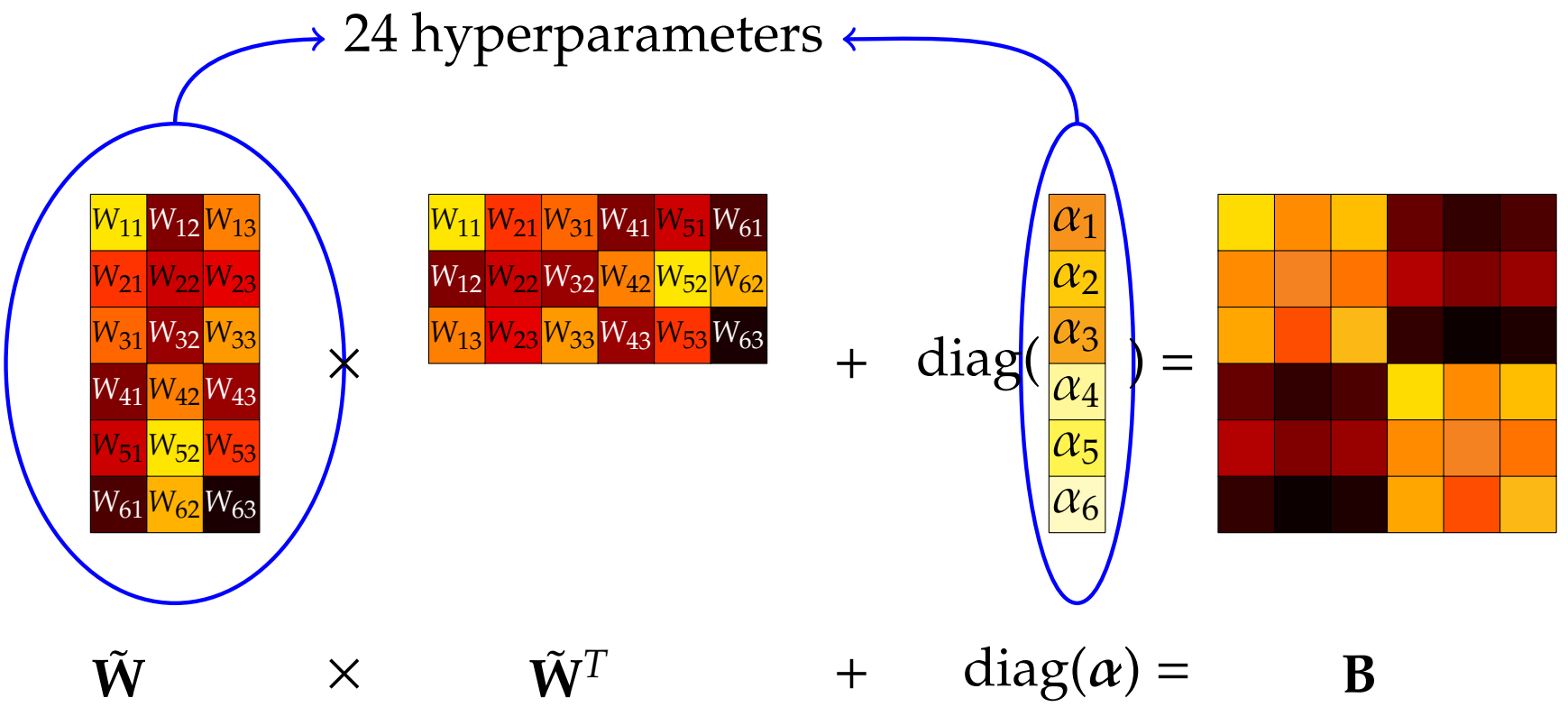
Incomplete-Cholesky model



Incomplete-Cholesky model



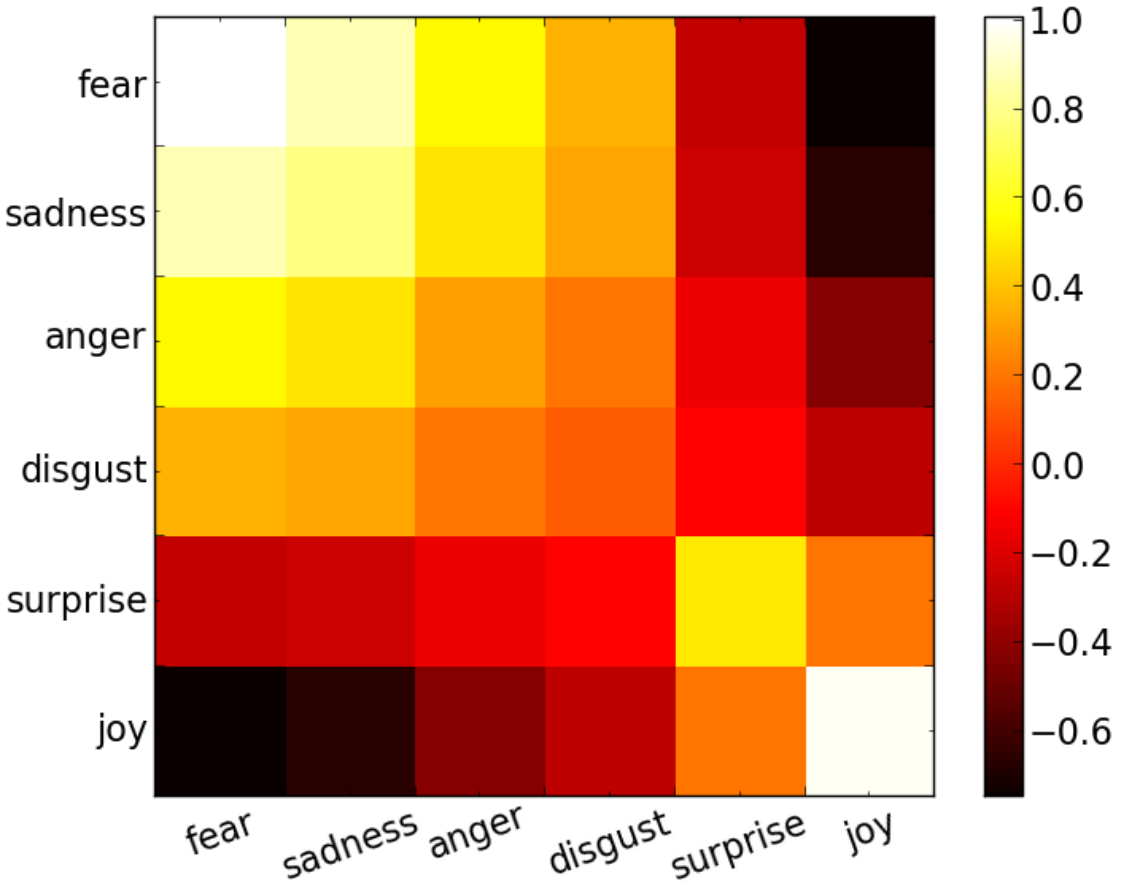
Incomplete-Cholesky model



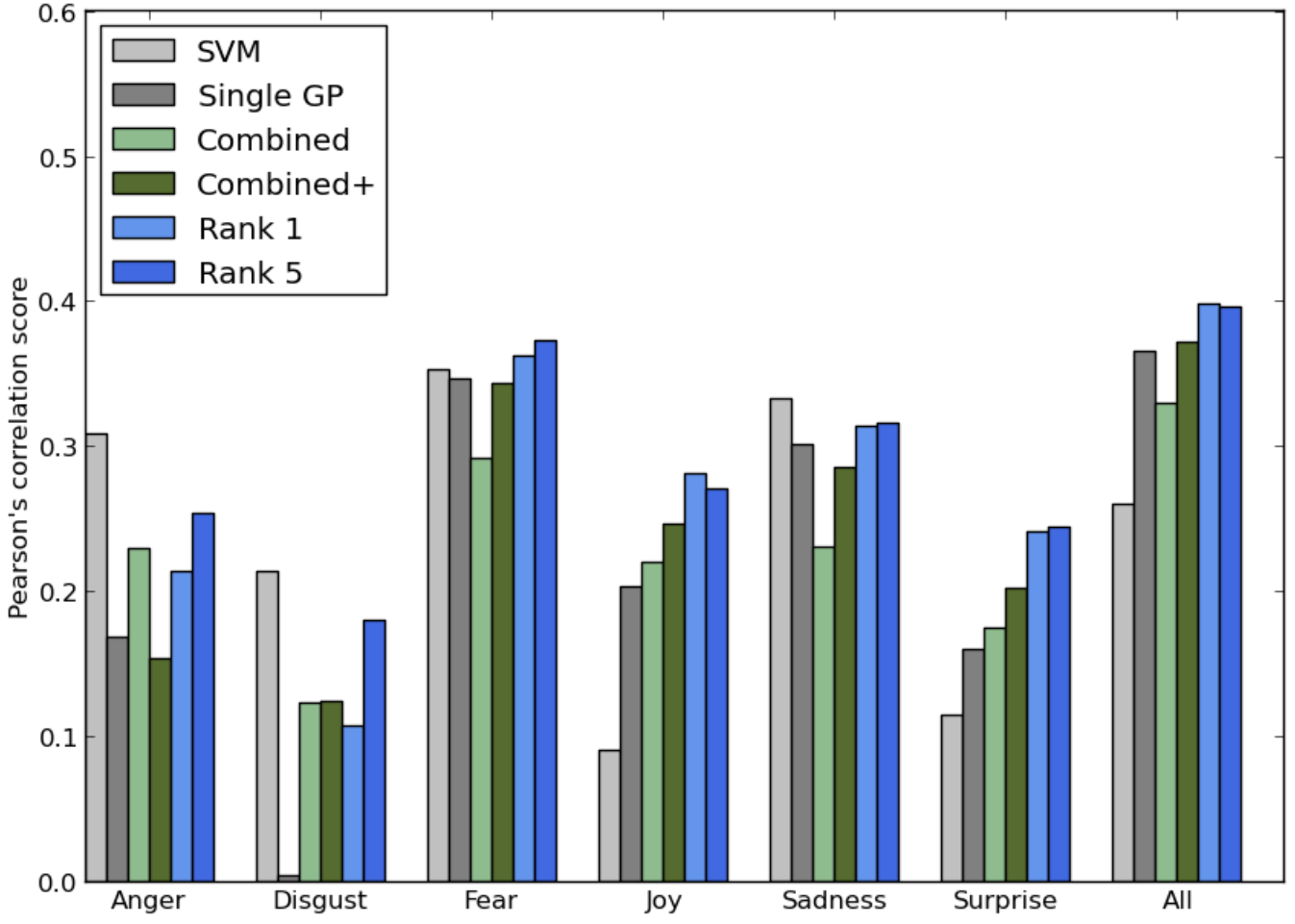
Experimental Setup

- Dataset: SEMEval2007 “Affective Text” Strapparava and Mihalcea (2007);
- 1000 News headlines, each one annotated with 6 scores [0-100], one for emotion;
- 100 sentences for training, 900 for testing;
- Bag-of-words representation as features;

Learned Task Covariances



Prediction Results



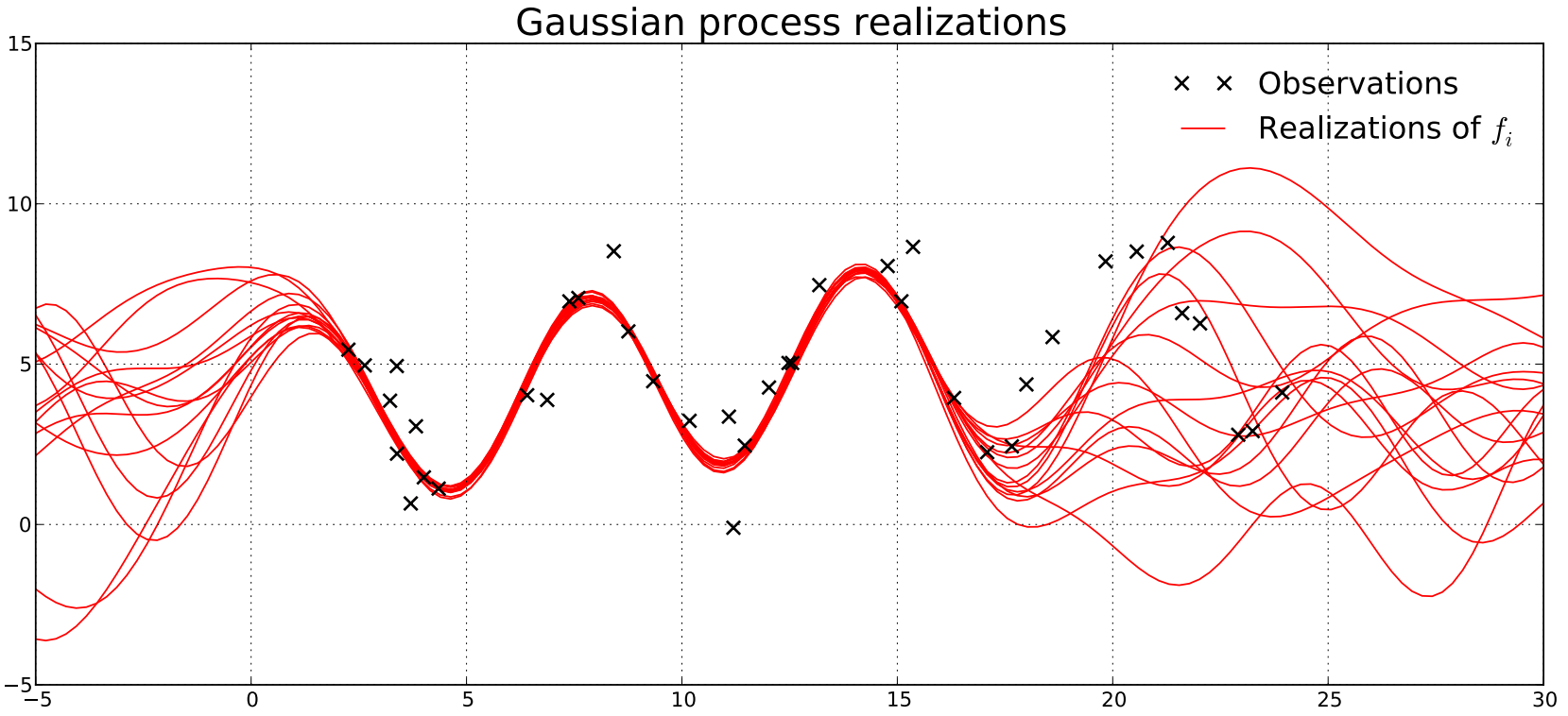
Outline

Outline

Recap: Regression

Observations, y_i, are a noisy version of latent process f_i,

$$y_i = f_i(\mathbf{x}_i) + \epsilon_i, \quad \text{with } \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$



Likelihood models

Analytic solution for Gaussian likelihood *aka* noise

Gaussian (process) prior
$\times$ Gaussian likelihood
$=$ Gaussian posterior

But what about other likelihoods?

- Counts $\mathbf{y} \in \mathcal{N}$
- Classification $\mathbf{y} \in \{C_1, C_2, ..., C_k\}$
- Ordinal regression (ranking) $C_1 < C_2 < ... < C_k$
- ...

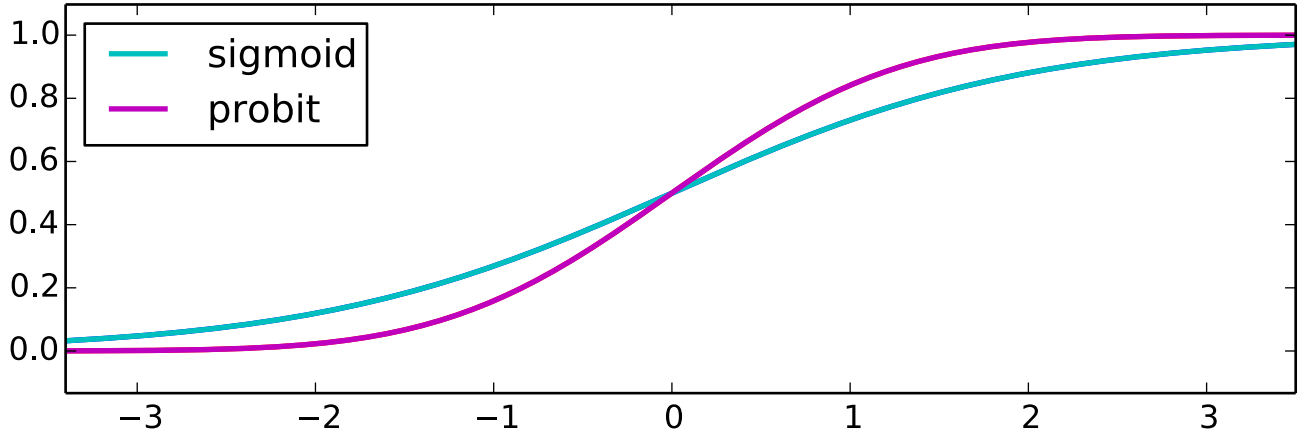
Classification

Binary classification, $y_i \in \{0, 1\}$.

Two popular choices for the likelihood

- Logistic sigmoid: $p(y_i = 1|f_i) = \sigma(f_i) = \frac{1}{1+\exp(-f_i)}$
- Probit function: $p(y_i = 1|f_i) = \Phi(f_i) = \int_{-\infty}^{f_i} \mathcal{N}(z|0, 1)dz$

"Squashing" input from $(-\infty, \infty)$ into range $[0, 1]$



Squashing function

Pass latent function through logistic function to obtain probability, $\pi(x) = p(y_i = 1|f_i)$

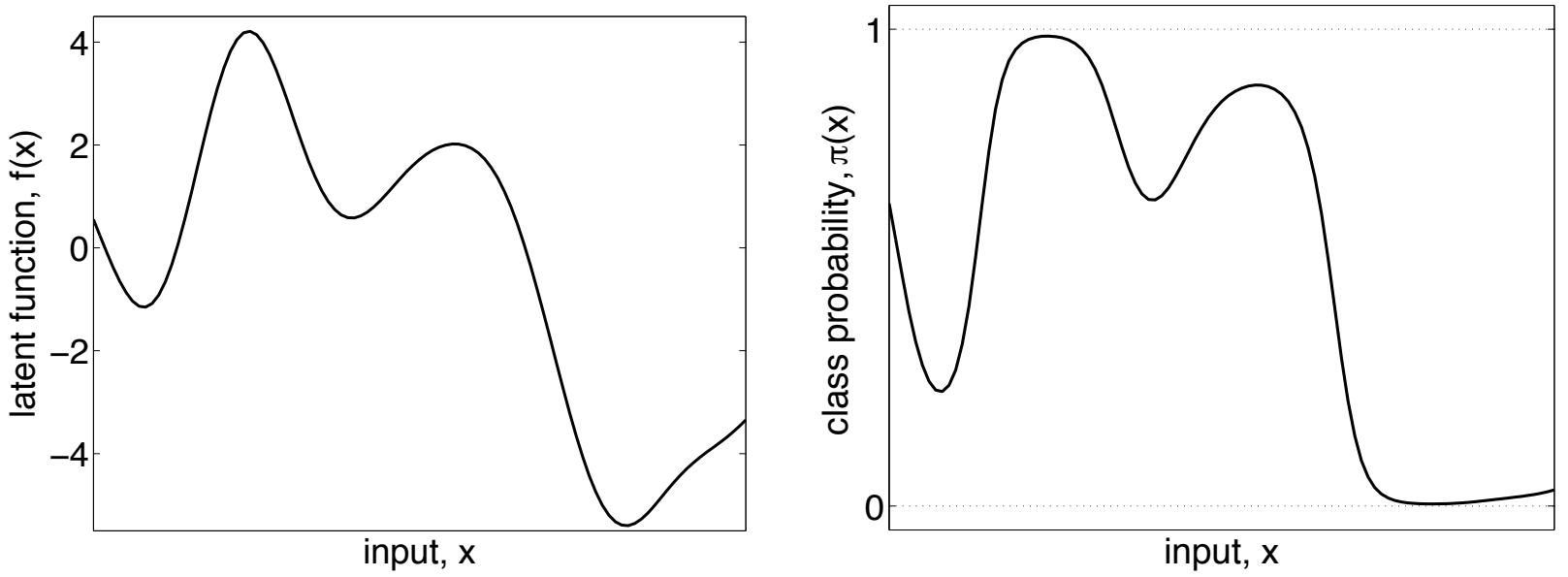


Figure from Rasmussen and Williams (2006)

Inference Challenges: for test case $\mathbf{x}_*$

Distribution over latent function

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

Distribution over classification output

$$p(y_* = 1|X, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|X, \mathbf{y}, \mathbf{x}_*) df_*$$

Problem: likelihood no longer conjugate with prior, so no analytic solution.

Approximate inference

Several inference techniques proposed for non-conjugate likelihoods:

- Laplace approximation
  Williams and Barber (1998)
- Expectation propagation
  Minka (2001)
- Variational inference
  Gibbs and MacKay (2000)
- MCMC
  Neal (1999)

And more, including sparse approaches for large scale application.

Laplace approximation

Approximate non-Gaussian posterior by a Gaussian, centred at the mode

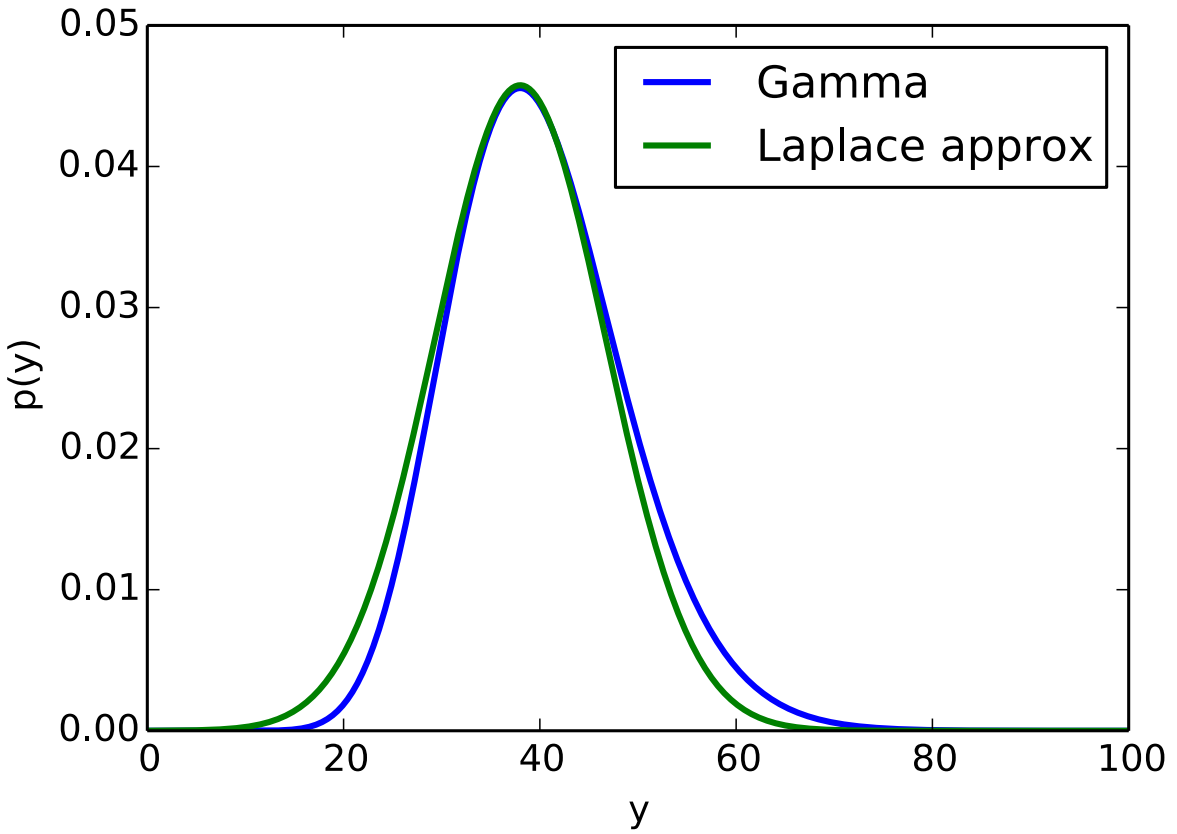


Figure from Rogers and Girolami (2012)

Laplace approximation

Log posterior

$$\Phi(\mathbf{f}) = \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|X) + \text{const}$$

Find the posterior mode, $\hat{\mathbf{f}}$, i.e., MAP estimation, $O(n^3)$.

Then take a second order Taylor series expansion about mode, and fit with a Gaussian

- with mean, $\mu = \hat{\mathbf{f}}$
- and co-variance $\Sigma = (K^{-1} + \nabla\nabla \log p(\mathbf{y}|\mathbf{f}))^{-1}$

Allows for computation of posterior and marginal likelihood, but predictions may still be intractable.

Expectation propagation

Take intractable posterior:

$$p(\mathbf{f}|\mathbf{X},\mathbf{y}) = \frac{1}{Z} p(\mathbf{f}|\mathbf{X}) \prod_{i=1}^{n} p(y_i|f_i)$$

$$Z = \int p(\mathbf{f}|\mathbf{X}) \prod_{i=1}^{n} p(y_i|f_i) d\mathbf{f}$$

Approximation with fully factorised distribution

$$q(\mathbf{f}|\mathbf{X},\mathbf{y}) = \frac{1}{Z_{EP}} p(\mathbf{f}|\mathbf{X}) \prod_{i=1}^{n} t(f_i)$$

Expectation propagation

Approximate posterior defined as

$$q(\mathbf{f}|\mathbf{y}) = \frac{1}{Z_{EP}} p(\mathbf{f}) \prod_{i=1}^{n} t(f_i)$$

where each component assumed to be Gaussian

- $p(y_i|f_i) \approx t_i(f_i) = \tilde{Z}_i \mathcal{N}(f_i|\tilde{\mu}_i, \tilde{\sigma}_i^2)$
- $p(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}(\mathbf{f}|\mathbf{0}, K_{nn})$

Results in Gaussian formulation for $q(\mathbf{f}|\mathbf{y})$

- allows for tractable multiplication, division with Gaussians
- and marginalisation, expectations etc

Expectation propagation

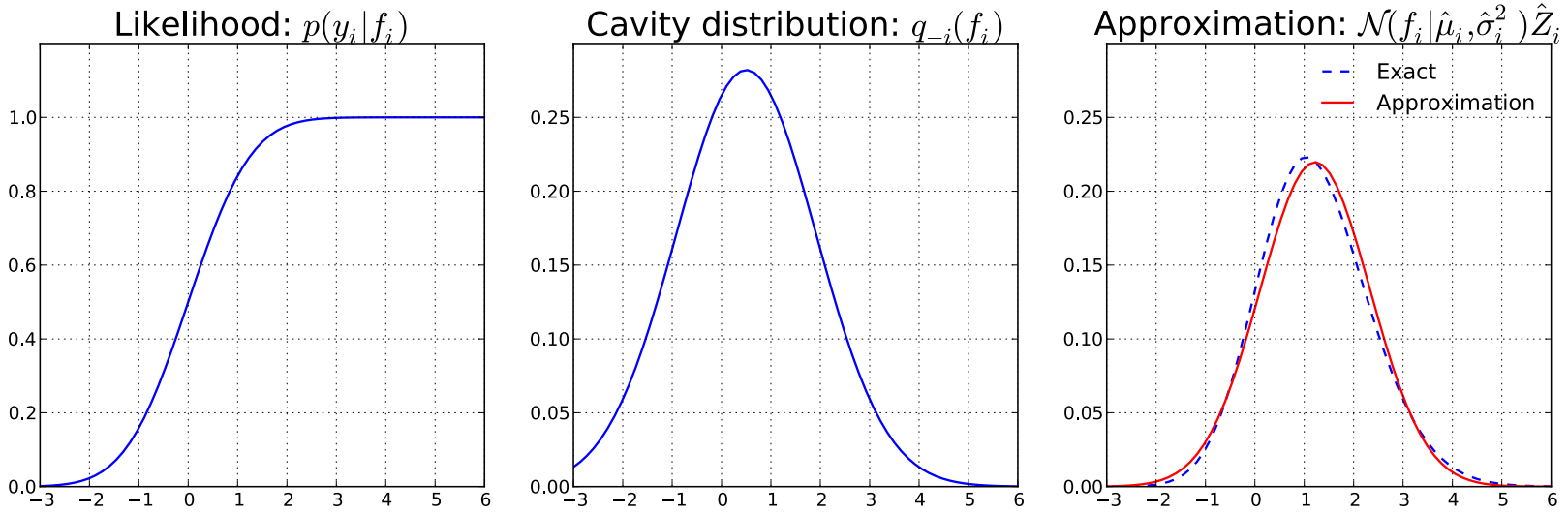
EP algorithm aims to fit $t_i(f_i)$ to the posterior, starting with a guess for q, then iteratively refining as follows

- minimise KL divergence between the true posterior for f_i and the approximation, t_i

$$\min_{t_i} \mathrm{KL}\left(p(y_i|f_i)q_{-i}(f_i) \,||\, t_i(f_i)q_{-i}(f_i)\right)$$

  where $q_{-i}(f_i)$ is the cavity distribution formed by marginalising $q(\mathbf{f})$ over $f_j,\ j \neq i$ then dividing by $t_i(f_i)$.
- key idea: only need accurate approximation for globally feasible f_i
- match moments to update t_i, then update $q(\mathbf{f})$

Expectation propagation



Expectation propagation

No proof of convergence

- but empirically works well
- often more accurate than Laplace approximation

Formulated for many different likelihoods

- complexity $O(n^3)$, dominated by matrix inversion
- sparse EP approximations can reduce this to $O(nm^2)$

See Minka (2001) and Rasmussen and Williams (2006) for further details.

Multi-class classification

Consider multi-class classification, $y \in \{C_1, C_2, \ldots, C_k\}$.

Draw vector of k latent function values for each input

$$\mathbf{f} = \left(f_1^1, \ldots, f_n^1, f_1^2, \ldots, f_n^2, f_1^k, \ldots, f_n^k\right)$$

Formulate clasification probability using soft-max

$$p(y_i = c|\mathbf{f}_i) = \frac{\exp(f_i^c)}{\sum_{c'} \exp(f_i^{c'})}$$

Multi-class classification

Assume k latent processes are uncorrelated, leading to prior covariance $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K)$ where

$$K = \begin{pmatrix} K_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & K_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & K_k \end{pmatrix}$$

is block diagonal $kn \times kn$ with each K_j of size $n \times n$.

Various approximation methods for inference, e.g., Laplace (Williams and Barber, 1998), EP (Kim and Ghahramani, 2006), MCMC (Neal, 1999).

Outline

GPs for Structured Prediction

- GPSC (Altun et al., 2004):
  - Defines a likelihood over label sequences: $p(\mathbf{y}|\mathbf{x})$, with latent variable over full sequences $\mathbf{y}$
  - HMM-inspired kernel, combining features from each observed symbol x_i and label pairs
  - MAP inference for hidden function values, $\mathbf{f}$, and sparsification trick for tractable inference
- GPstruct (Bratières et al., 2013):
  - Base model is a CRF:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{f}) = \frac{\exp \sum_c f(c, \mathbf{x}_c, \mathbf{y}_c)}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp \sum_c f(c, \mathbf{x}_c, \mathbf{y}'_c)}$$

  - Assumes that each potential $f(c, \mathbf{x}_c, \mathbf{y}_c)$ is drawn from a GP
  - Bayesian inference using MCMC (Murray et al., 2010)

Outline

String Kernels

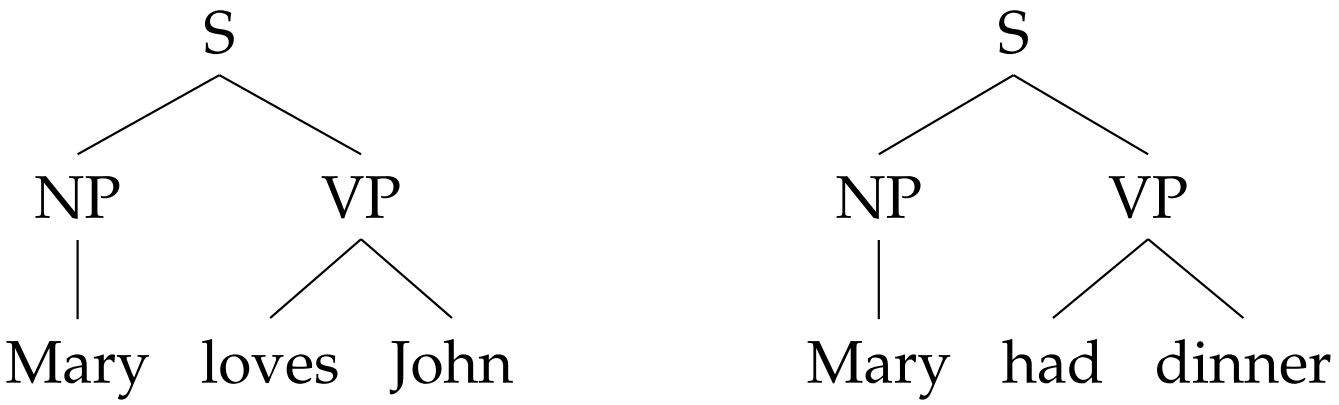
$$k(x, x') = \sum_{s \in \Sigma^*} w_s \phi_s(x) \phi_s(x'),$$

- $\phi_s(x)$: counts of substring s inside x;
- $0 \leq w_s \leq 1$: weight of substring s;
- s can also be a subsequence (containing gaps);

- s = char sequences $\rightarrow$ ngram kernels (Lodhi et al., 2002) (useful for stems);
  k(**ba**r, **ba**t) = 3 (**b**,**a**,**ba**)
- s = word sequences $\rightarrow$ Word Sequence kernels (Cancedda et al., 2003);
  k(**gas** only **injection**, **gas** assisted plastic **injection**) = 3
- Soft matching:
  k(battle, battles) $\neq 0$
  k(battle, combat) $\neq 0$

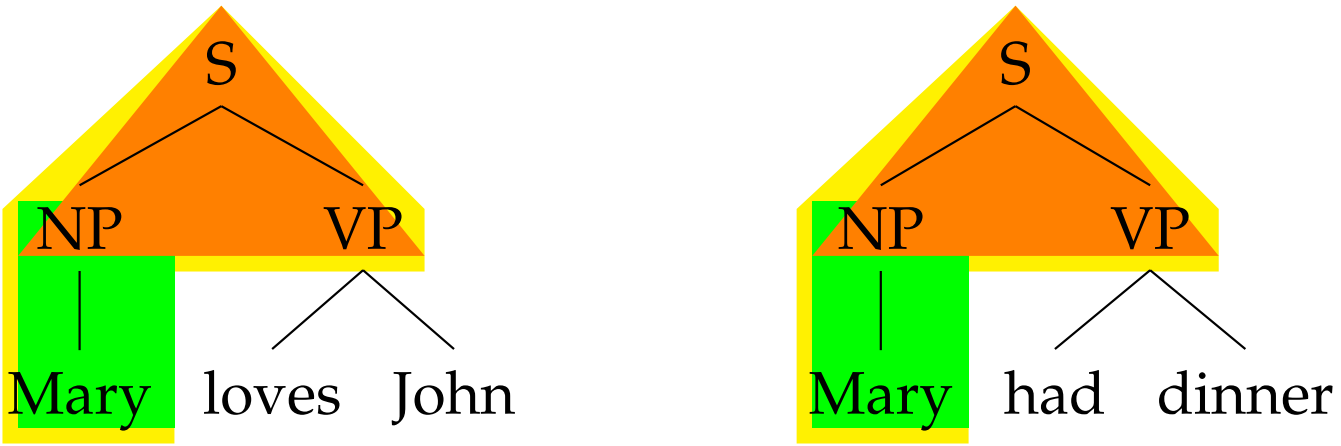
Tree Kernels

- Subset Tree Kernels (Collins and Duffy, 2001)



Tree Kernels

- Subset Tree Kernels (Collins and Duffy, 2001)



- Partial Tree Kernels (Moschitti, 2006): allows "broken" rules, useful for dependency trees;
- Soft matching can also be applied.

Conclusions

Gaussian Processes are a powerful framework

- elegant kernel machines
- probabilistic, quantify uncertainty
- closed form Bayesian inference & model selection

Mature enough for widespread application

- frontier challenges in NLP
- scaling improvements make these applications practical

References I

Altun, Y., Hofmann, T., and Smola, A. J. (2004). Gaussian Process Classification for Segmenting and Annotating Sequences. In *Proceedings of ICML*, page 8, New York, New York, USA. ACM Press.

Alvarez, M. A., Rosasco, L., and Lawrence, N. D. (2011). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266.

Beck, D., Cohn, T., and Specia, L. (2014). Joint Emotion Analysis via Multi-task Gaussian Processes. In *Proceedings of EMNLP*.

Bonilla, E., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian process prediction. NIPS.

Bratières, S., Quadrianto, N., and Ghahramani, Z. (2013). Bayesian Structured Prediction using Gaussian Processes. *arXiv:1307.3846*, pages 1–17.

Callison-Burch, C., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2012). Findings of the 2012 workshop on statistical machine translation.

Cancedda, N., Gaussier, E., Goutte, C., and Renders, J.-M. (2003). Word-Sequence Kernels. *The Journal of Machine Learning Research*, 3:1059–1082.

Cohn, T. and Specia, L. (2013). Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. ACL.

Collins, M. and Duffy, N. (2001). Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems*.

Daumé III, H. (2007). Frustratingly easy domain adaptation. ACL.

References II

Evgeniou, T., Micchelli, C. A., and Pontil, M. (2006). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(1):615.

Gibbs, M. N. and MacKay, D. J. (2000). Variational gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464.

Kim, H.-C. and Ghahramani, Z. (2006). Bayesian gaussian process classification with the em-ep algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):1948–1959.

Koponen, M., Aziz, W., Ramos, L., and Specia, L. (2012). Post-editing time as a measure of cognitive effort. AMTA 2012 Workshop on Post-editing Technology and Practice.

Lampos, V., Aletras, N., Preoţiuc-Pietro, D., and Cohn, T. (2014). Predicting and Characterising User Impact on Twitter. EACL.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2:419–444.

Minka, T. (2001). Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc.

Moschitti, A. (2006). Making Tree Kernels practical for Natural Language Learning. In *EACL*, pages 113–120.

Murray, I., Adams, R. P., and Mackay, D. (2010). Elliptical slice sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 541–548.

References III

Neal, R. (1999). Regression and classification using gaussian process priors. *Bayesian Statistics*, 6.

Preoţiuc-Pietro, D. and Cohn, T. (2013). A temporal model of text periodicities using Gaussian Processes. EMNLP.

Quiñonero Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA.

Rogers, S. and Girolami, M. (2012). *A First Course in Machine Learning*. Chapman & Hall/CRC.

Specia, L., Shah, K., De Souza, J. G., and Cohn, T. (2013). Quest-a translation quality estimation framework. Citeseer.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., and Cristianini, N. (2009). Estimating the Sentence-Level Quality of Machine Translation Systems. pages pp. 28–35, Barcelona, Spain.

Strapparava, C. and Mihalcea, R. (2007). SemEval-2007 Task 14 : Affective Text. In *Proceedings of SEMEVAL*.

Strapparava, C. and Mihalcea, R. (2008). Learning to identify emotions in text. In *Proceedings of the 2008 ACM Symposium on Applied Computing*.

Williams, C. K. and Barber, D. (1998). Bayesian classification with gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1342–1351.