AN INTRODUCTION TO GAUSSIAN PROCESSES

Daryl Weir

Computational Interaction Summer School

22/6/15

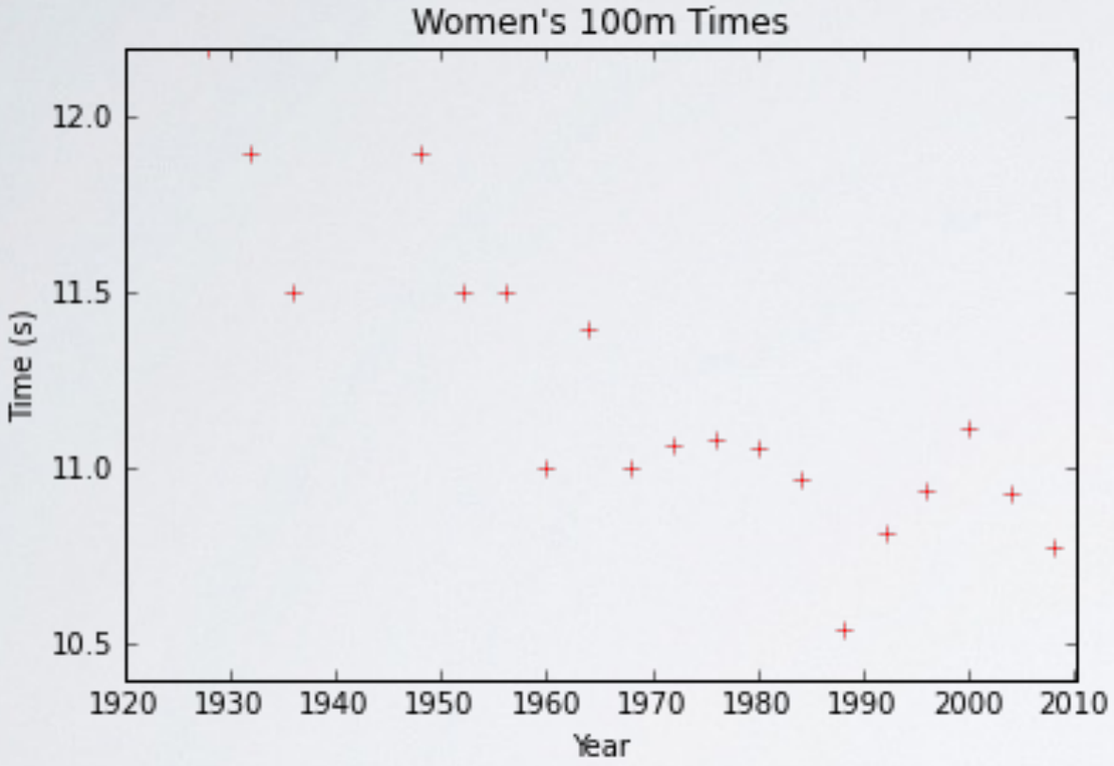
User Interfaces
Aalto University

github.com/darylweir/summerschool2015

OVERVIEW

- Introduction
- Bayesian Regression - a whistle stop tour
- What's a Gaussian process?
- GPs in HCI - touch offsets

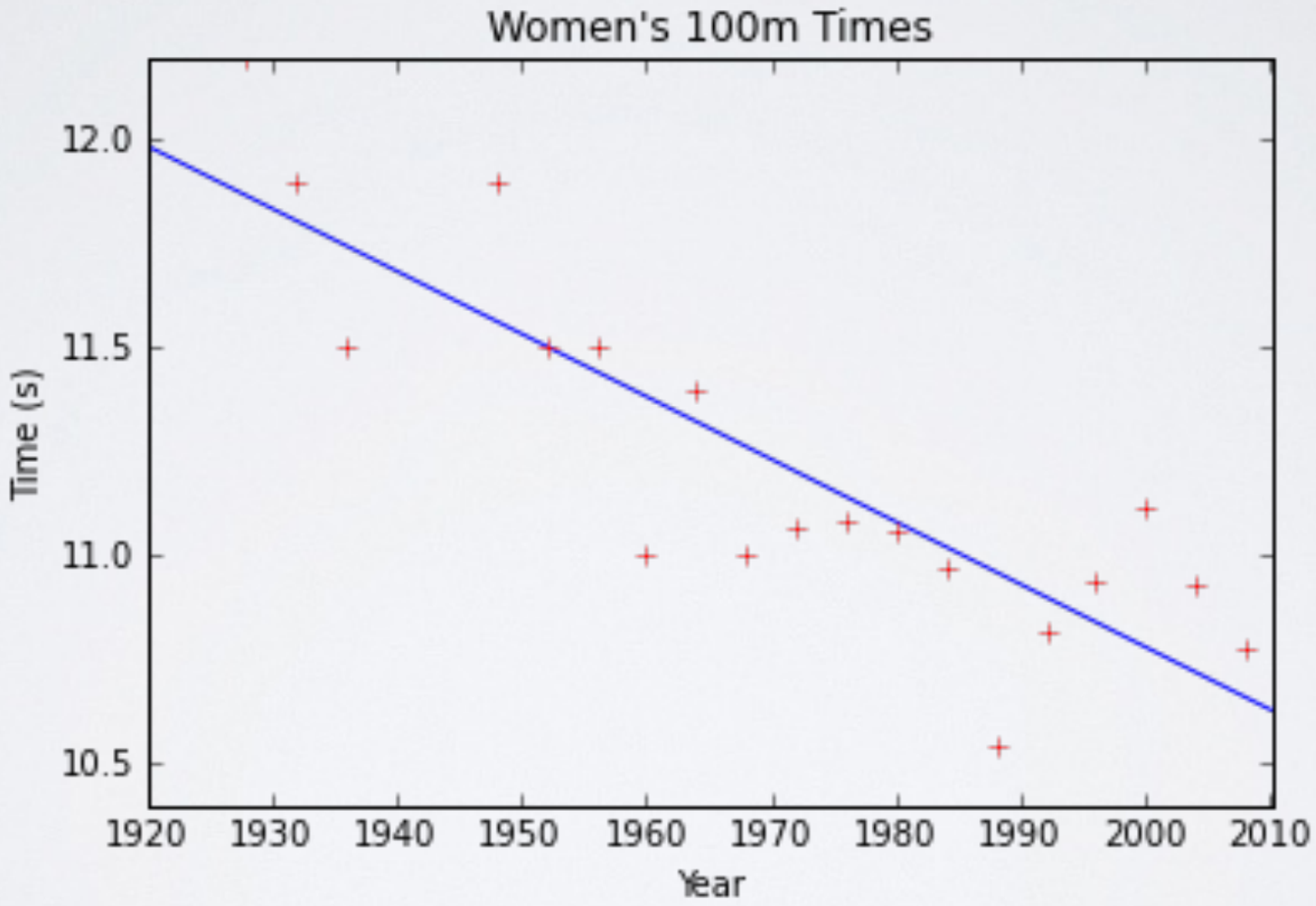
BUT FIRST, THE OLYMPICS



What function might have generated these times?

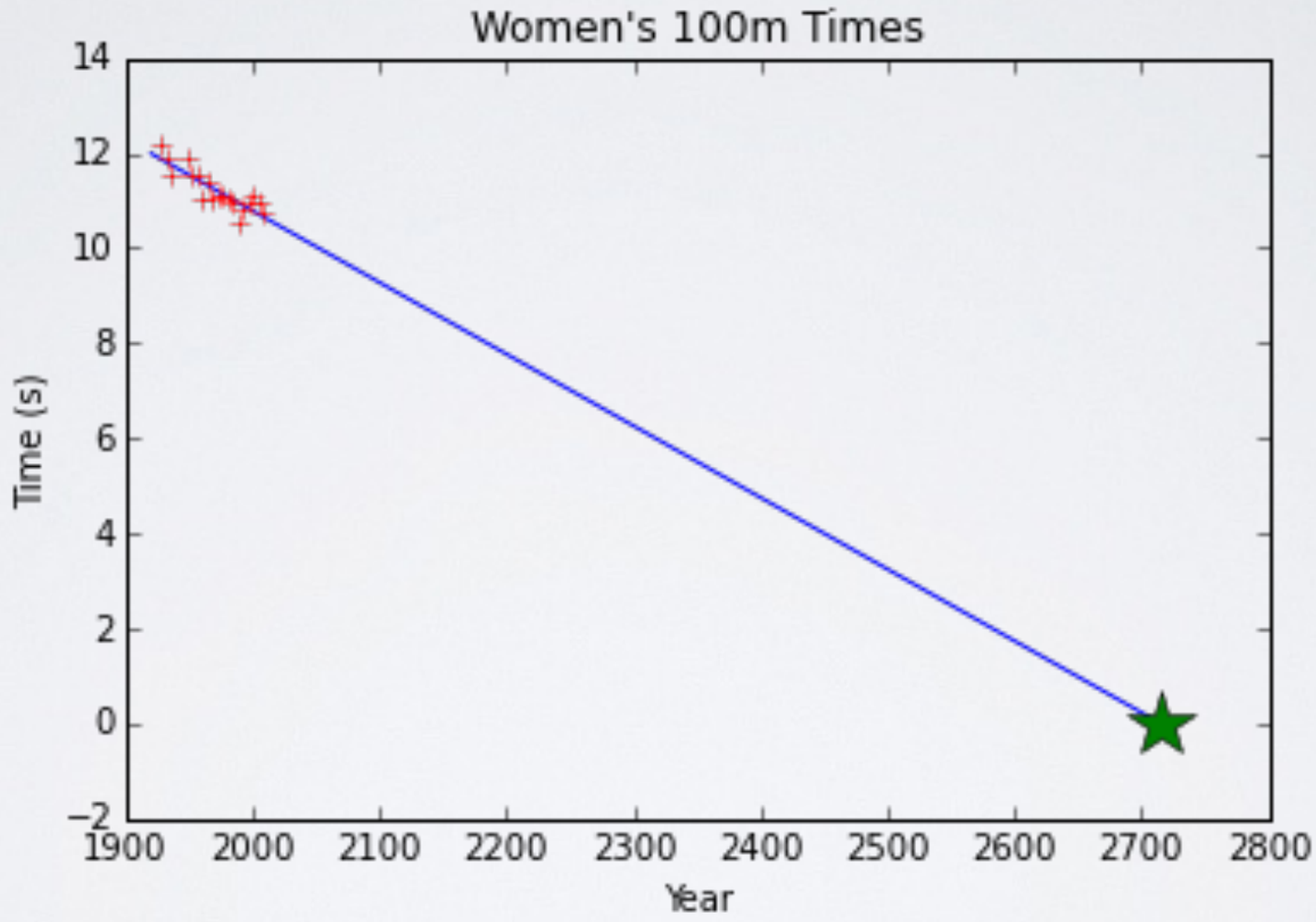
HOW ABOUT A STRAIGHT LINE?

$$y = w_0 + w_1 x$$



Looks okay, I guess…

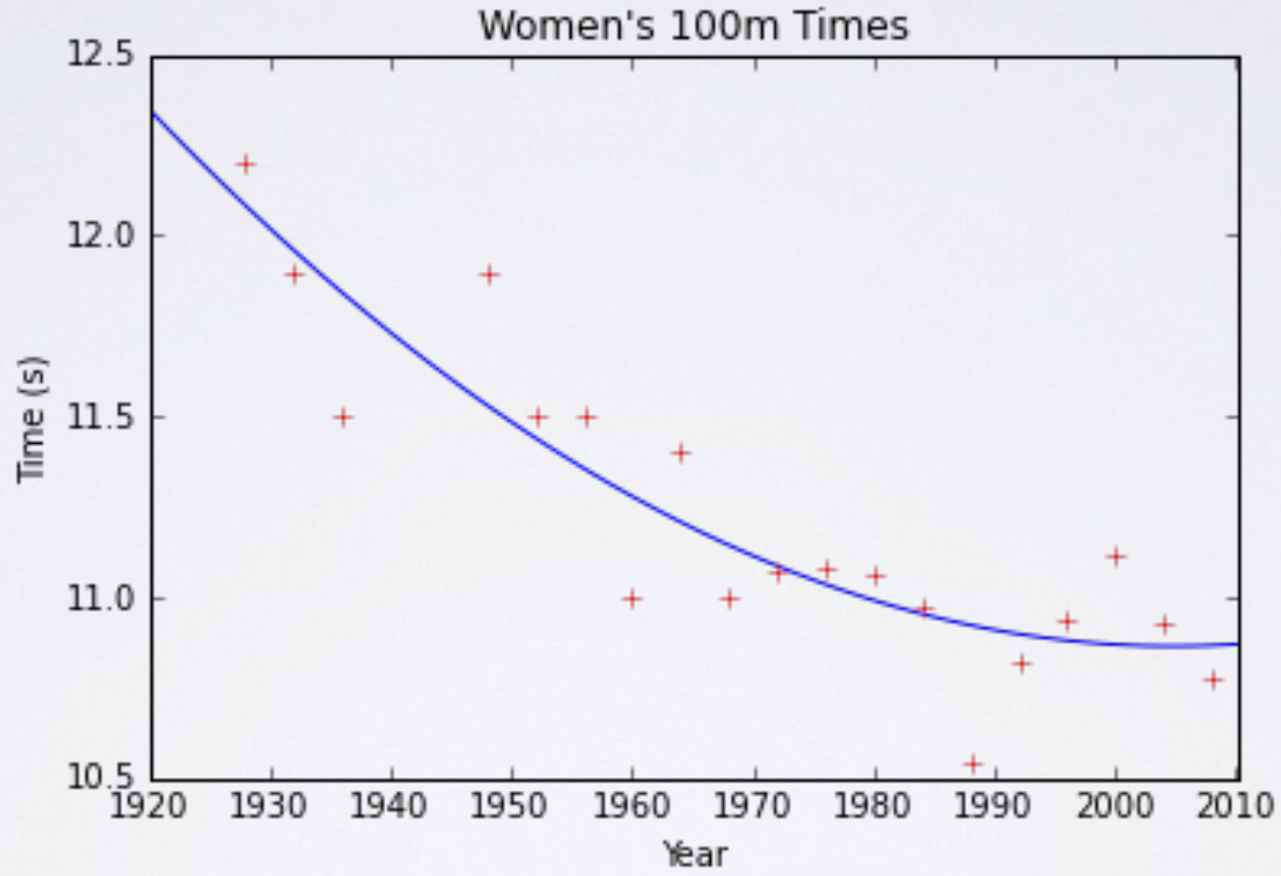
THE YEAR 2715



We probably shouldn't be too confident about this model.

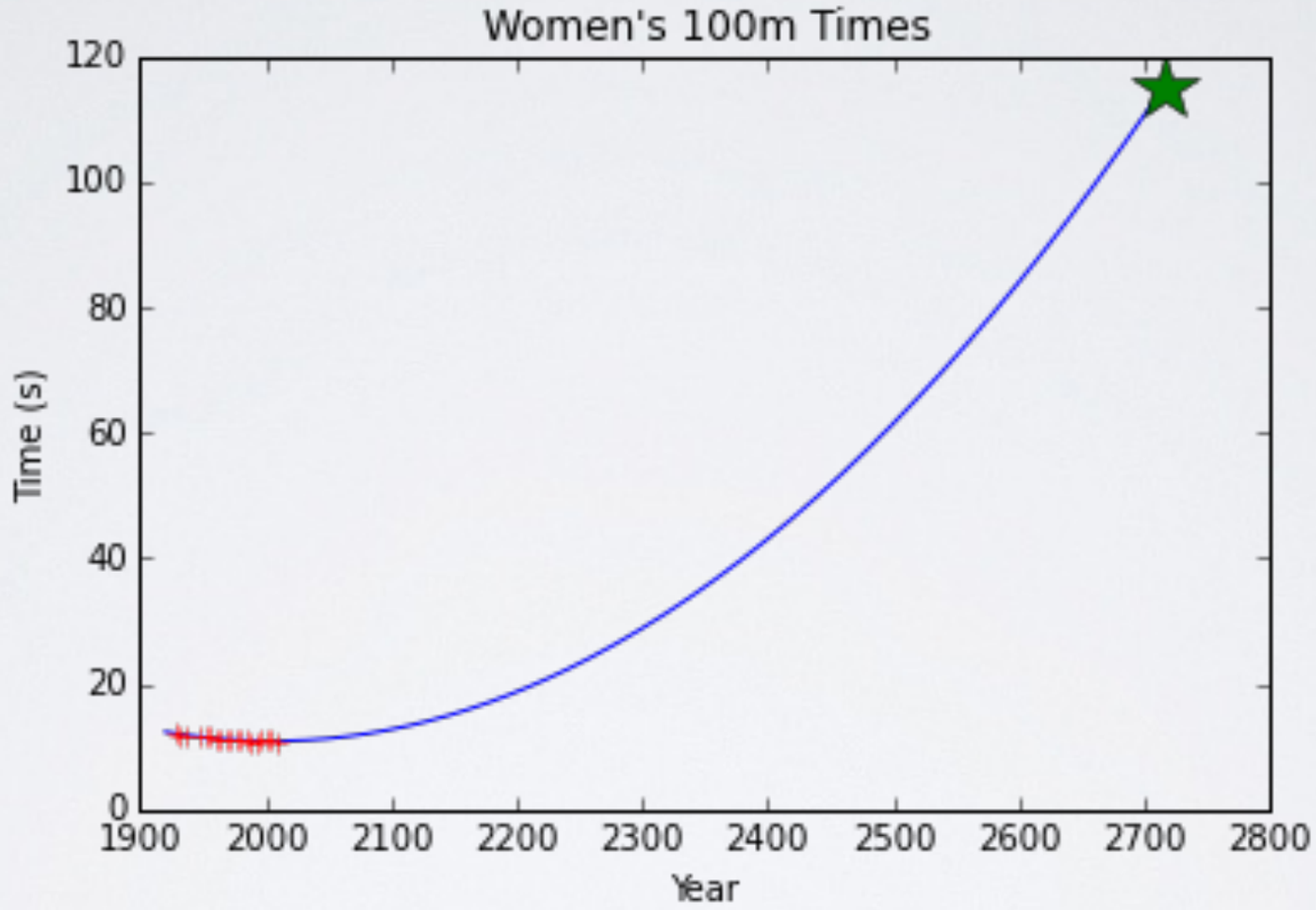
HOW ABOUT A QUADRATIC?

$$y = w_0 + w_1 x + w_2 x^2$$



A little better, maybe?

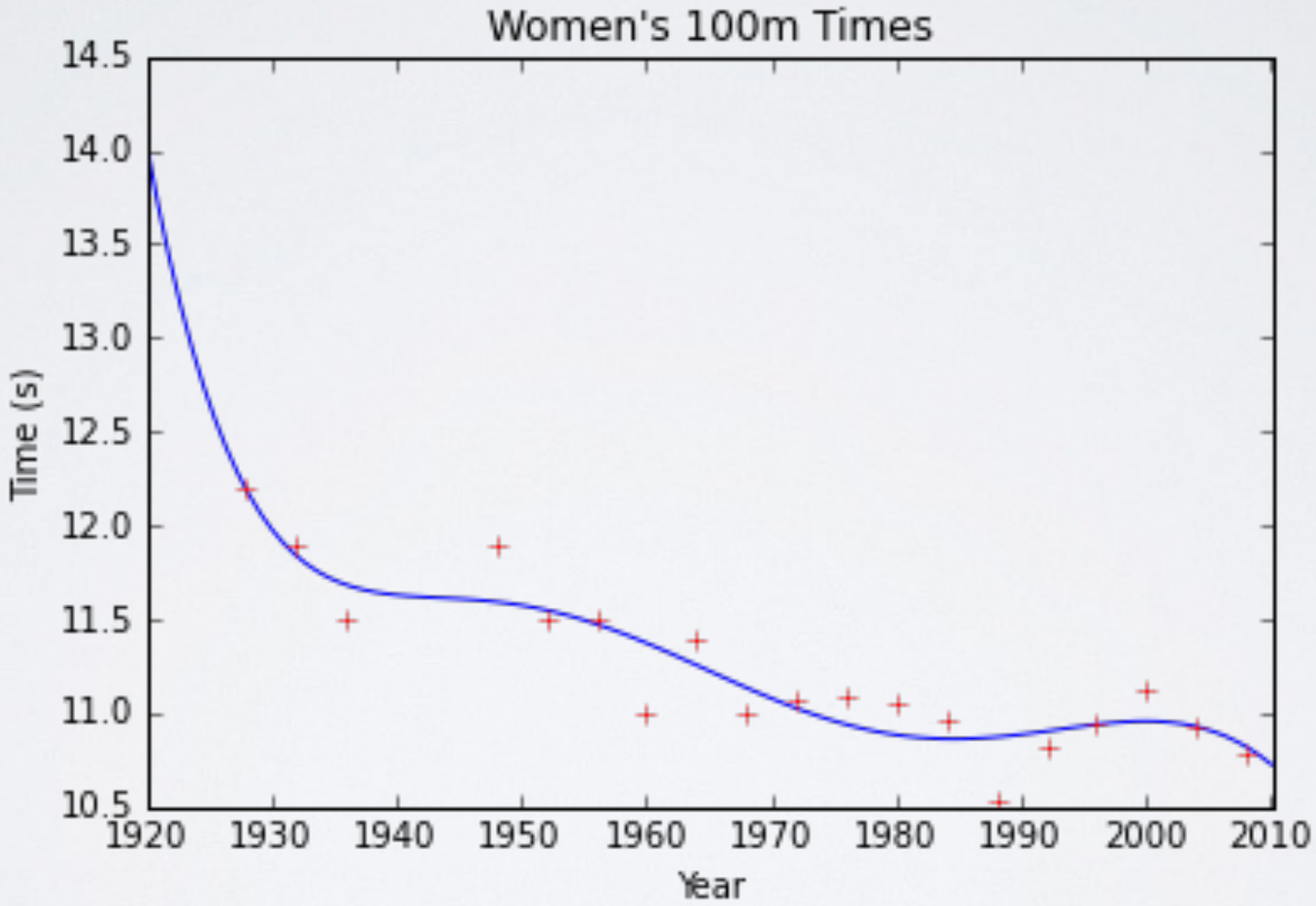
BACK TO THE FUTURE



100m in 118 seconds? Pretty good.

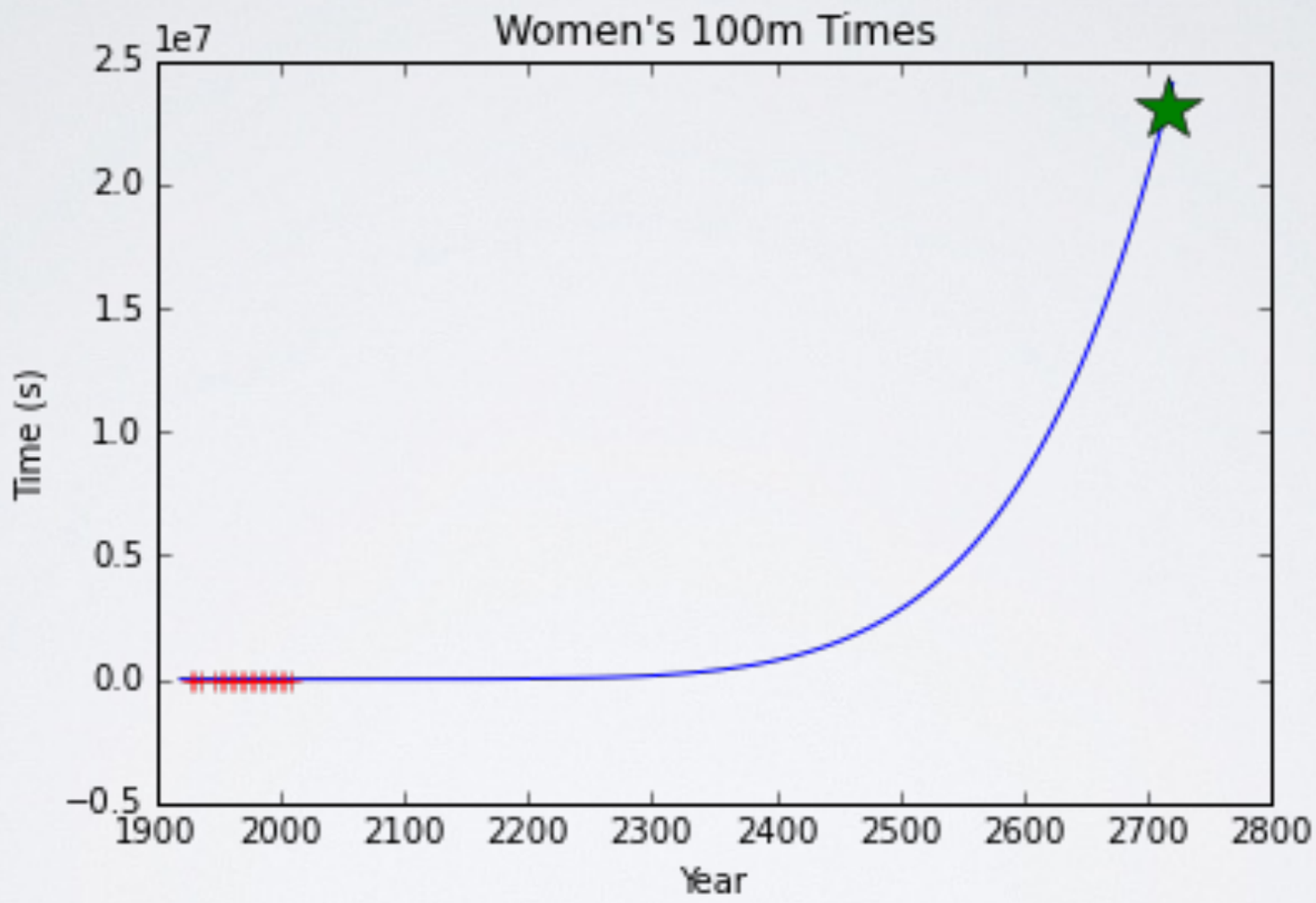
MAYBE 6TH ORDER?

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + w_6 x^6$$



Looking good. Let's check in with 2715 again.

OH.



9 months is a long time.

THE LESSON

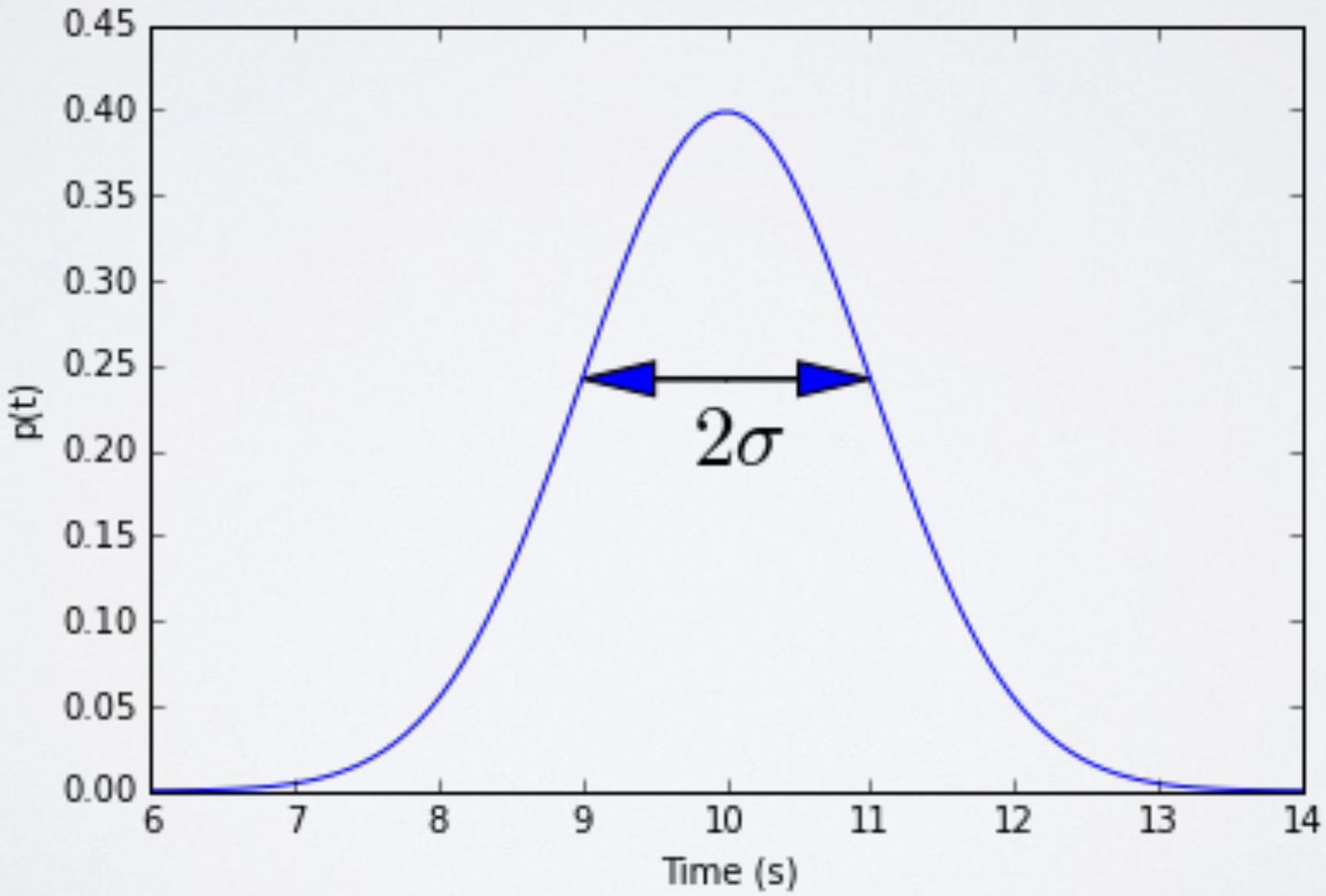
Don’t extrapolate too far from your data!

Also, a few things to note:

- it’d be good to have a confidence level for the predictions - problem 1
- we had to make some assumptions about the form of the fit to get started - problem 2

PROBLEM 1 - CONFIDENCE

Idea: Instead of predicting a single time, predict a **distribution** over times. The variance of this distribution indicates how confident we should be.



LINEAR REGRESSION

Problem Statement

Given N observations $\{\mathbf{x}_i, y_i\}$, construct N equations of form

$$y_i = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + \epsilon_i$$

where $\boldsymbol{\phi}(\mathbf{x}_i)$ is a set of D **basis functions**, $\mathbf{w}$ is a vector of D weights, and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise variable.

e.g. for two dimensional input $\mathbf{x} = (x_1, x_2)$ some basis functions might be $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2^2, x_1^5, \sin x_2]^T$

Linear regression is the task of finding the optimal set of weights to fit these basis functions to the data.

SOLUTION

Defining $\mathbf{y} = [y_1, \ldots, y_N]^T$ and $\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^T \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^T \end{bmatrix}$

a bit of algebra shows that minimising the squared error gives

$$\mathbf{w} = [\mathbf{\Phi}^T \mathbf{\Phi}]^{-1} \mathbf{\Phi}^T \mathbf{y}$$

But this is only a point estimate. How confident should we be?

BAYESIAN LINEAR REGRESSION

From Bayes' rule: $$p(\mathbf{w}|\mathbf{\Phi}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{\Phi}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{\Phi})}$$

Put a Gaussian prior on the weights: $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I})$

The likelihood is Gaussian (since we used Gaussian noise), so the posterior is also Gaussian:

$$p(\mathbf{w}|\mathbf{\Phi}, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{A})$$

$$\mathbf{A} = (\sigma^{-2}\mathbf{\Phi}^{\boldsymbol{T}}\mathbf{\Phi} + \alpha^{-1}\mathbf{I})^{-1} \qquad \boldsymbol{\mu} = \sigma^{-2}\mathbf{A}\mathbf{\Phi}\mathbf{y}$$

PREDICTION

How do we use the posterior on the weights to predict?

Could pick a single value (say, the mean) and use that.

But we're Bayesians now! Let's marginalise.

Define:

$\mathbf{x}_*$ - test point

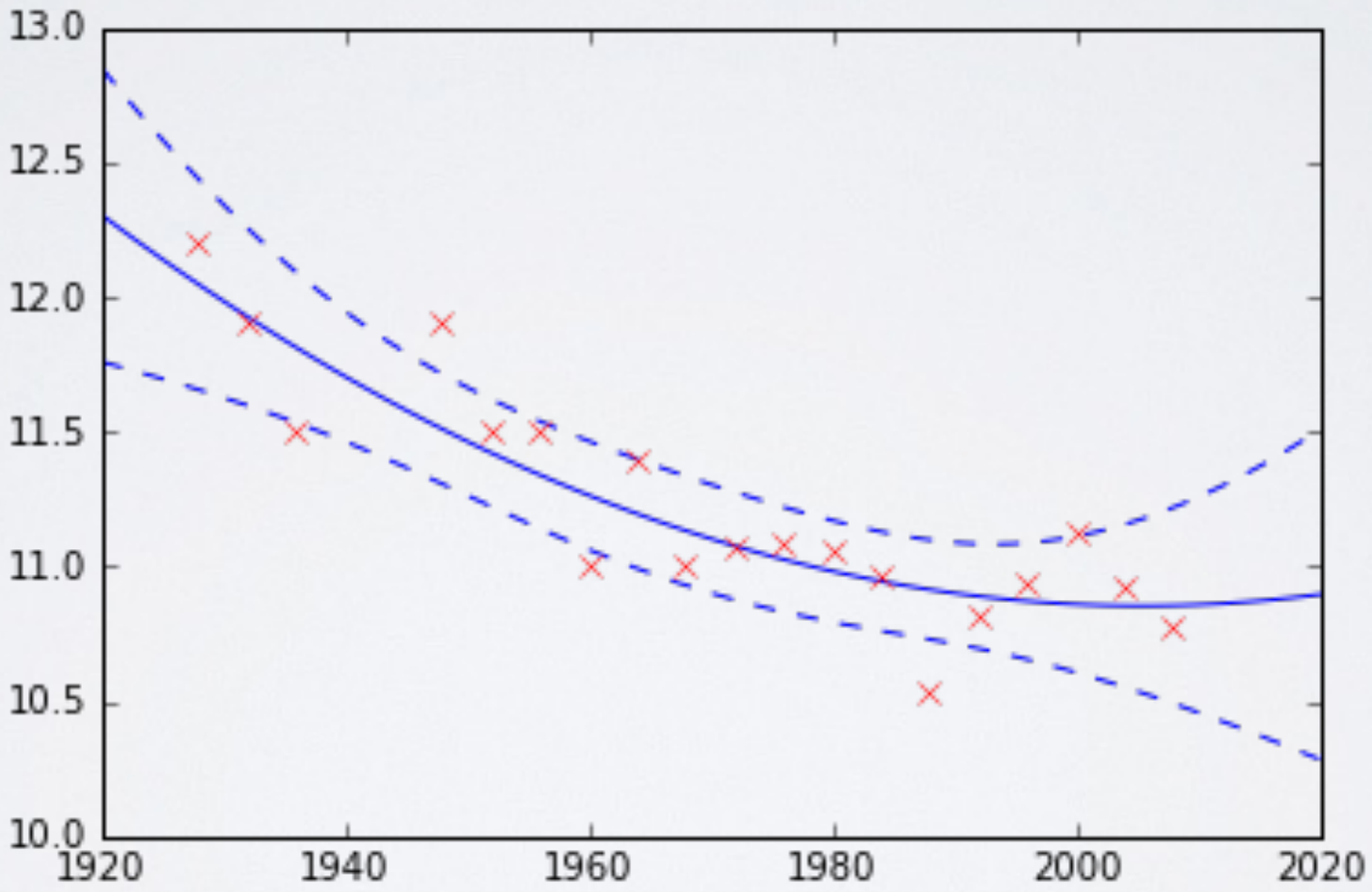
f_* - predicted value

$\boldsymbol{\phi}_*$ - basis functions at test point

$$p(f_*|\boldsymbol{\phi}_*, \boldsymbol{\Phi}, \mathbf{y}) = \int p(f_*|\boldsymbol{\phi}_*, \mathbf{w}) p(\mathbf{w}|\boldsymbol{\Phi}, \mathbf{y}) d\mathbf{w}$$

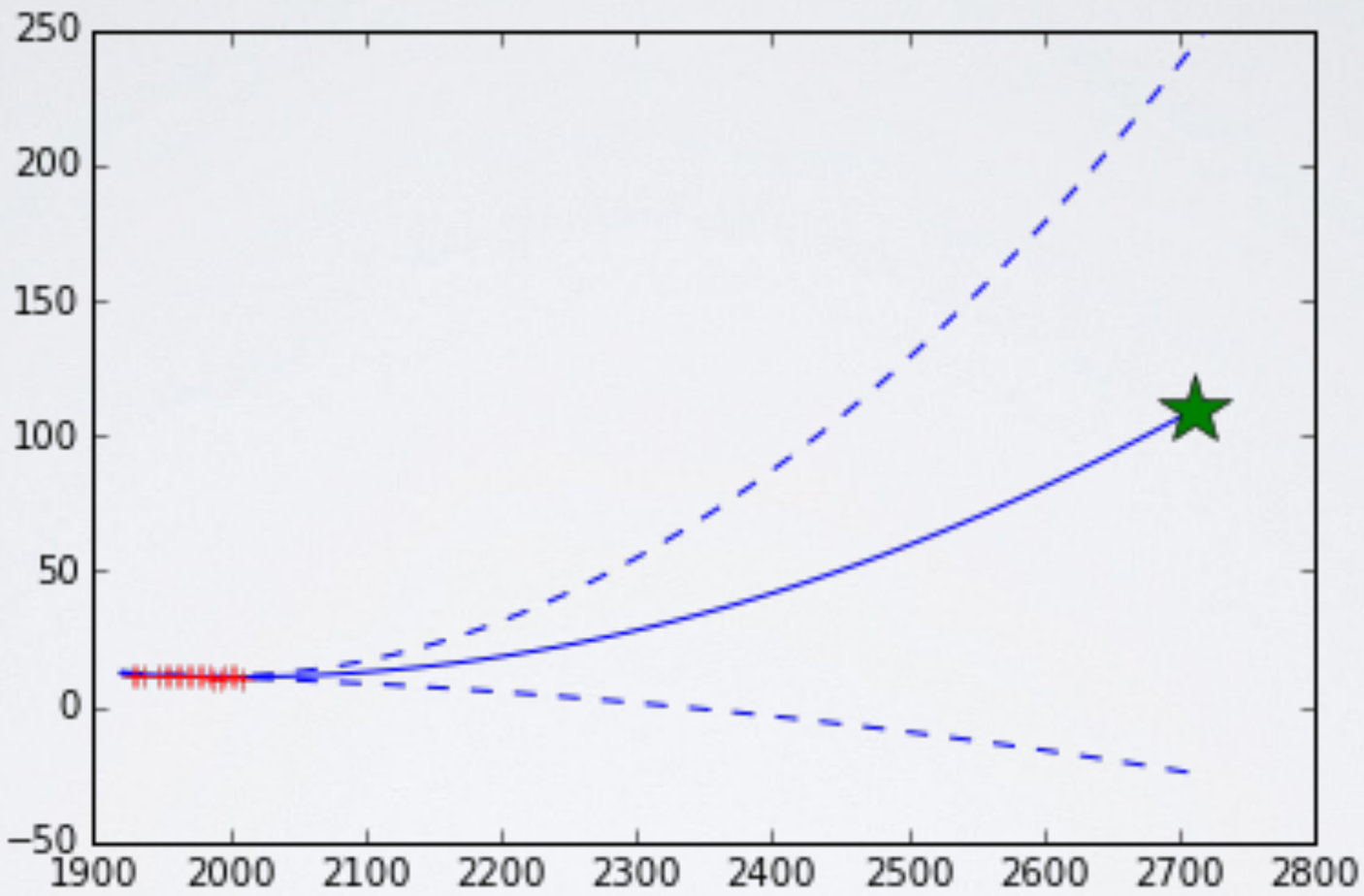
$$= \mathcal{N}(\boldsymbol{\phi}_*^{\boldsymbol{T}} \boldsymbol{\mu}, \boldsymbol{\phi}_*^T \mathbf{A} \boldsymbol{\phi}_*)$$

OLYMPICS, WITH ADDED CONFIDENCE



Quadratic model, two standard deviation errors

AND IN THE FUTURE…



Not a good prediction, but at least we're not confident!

PROBLEM 2 - CHOOSING BASIS FUNCTIONS

We still had to assume our data could be fitted by a quadratic.

Can we be less strict about our assumptions and still do inference?

Yes! Using Gaussian process regression, we just have to assume the function is '**smooth**'

GP - THE INTUITION

Don't do inference in **weight space**

Instead, do it in **function space**

Learn a **distribution over functions**

WHY DO THIS?

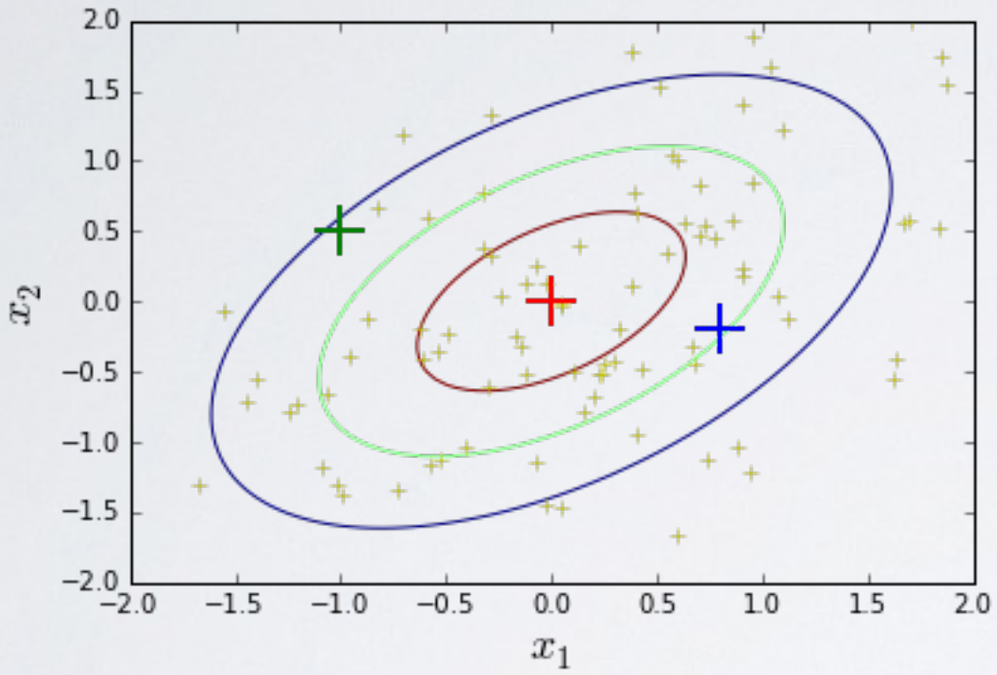
We can answer some interesting questions using this approach:

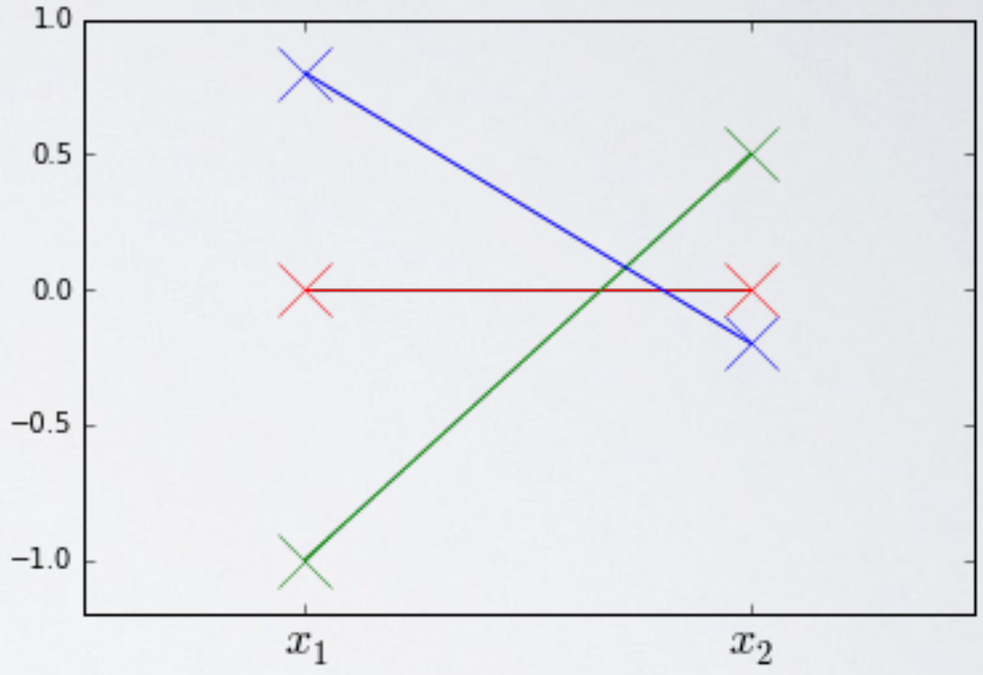
- How can we rank the likelihoods of different functions for a given dataset?
- What's the most likely function for this data (mean of the function distribution)?
- What are some other possible functions? (sampling from the posterior)

DEFINITION

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

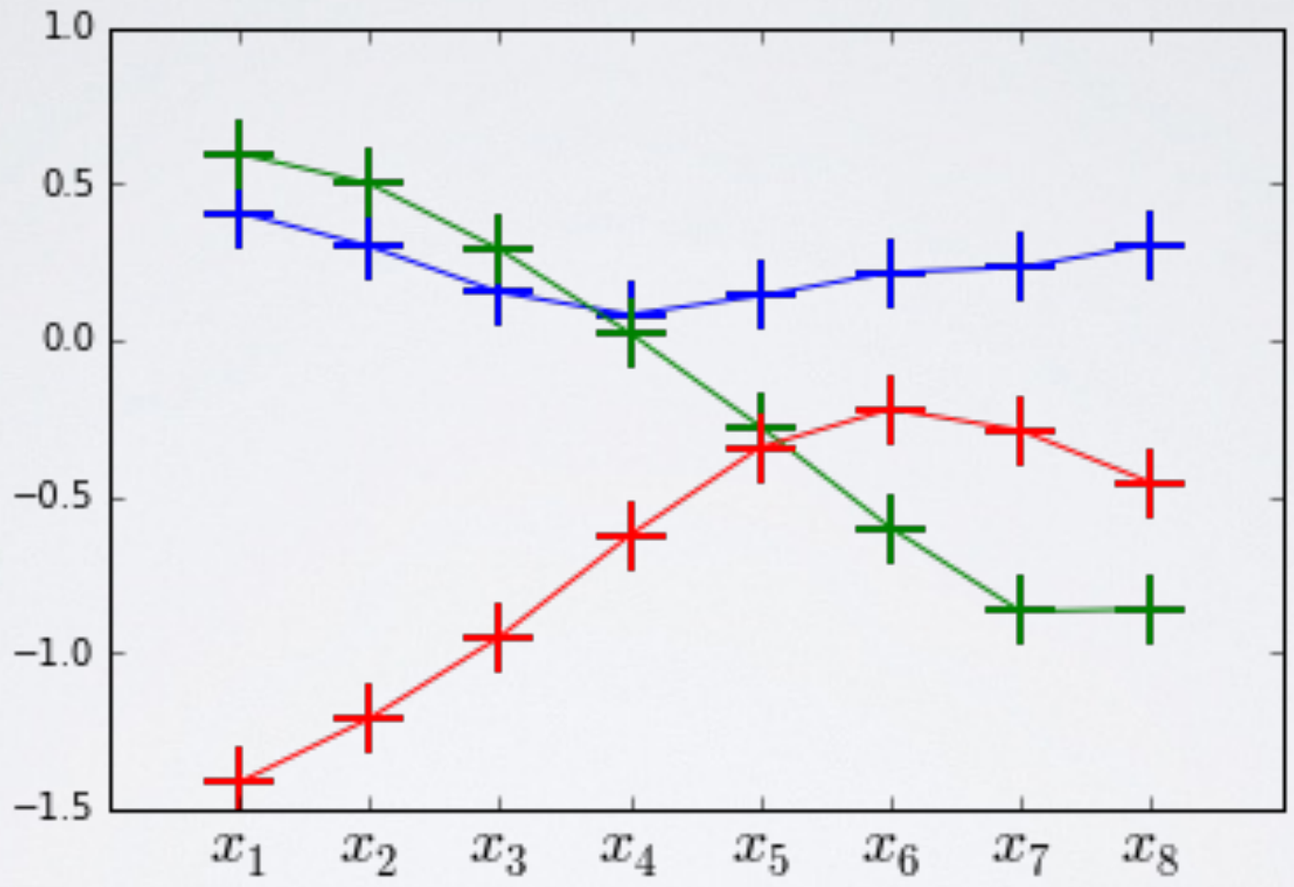
WHAT DOES THAT MEAN?





Visualising draws from a 2D Gaussian.

8D?



What's the limit?

LARGE DIMENSIONS

- If we index the components of the Gaussian with the reals instead of integers, our draws become uncountably infinite vectors.
- Sounds bad, but no problem - these draws are **functions!**
- No reason to stop there - index with $\mathbb{R}^2$ to get functions of two variables, or with strings, or with graphs…
- For each element in the index set, we get a Gaussian distribution.
- Need to know how these distributions are correlated.

GP CHARACTERISTICS

For each point $\mathbf{x}$ in the index set, we need a function which gives the mean of the corresponding Gaussian - the **mean function** $\mu(\mathbf{x})$

Also need to know the correlation/covariance between the Gaussians indexed by $\mathbf{x}$ and $\mathbf{z}$ - this is given by the **covariance function** $k(\mathbf{x}, \mathbf{z})$ (also known as the kernel)

These two functions completely specify the Gaussian Process.

NB: the mean function is often assumed to be zero, so in many examples the GP is defined only by the kernel.

COVARIANCE FUNCTIONS

Gaussian kernel (extremely common)

$$k(\mathbf{x}, \mathbf{z}) = \sigma^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}(x_d - z_d)^2/l_d^2\right)$$

Periodic kernel (only 1D inputs)

$$k(x, z) = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x - z|/p)}{l^2}\right)$$

Bias and noise kernels

$$k(\mathbf{x}, \mathbf{z}) = 1 \qquad k(\mathbf{x}, \mathbf{z}) = \delta_{\mathbf{x},\mathbf{z}}$$

Kernels can be added and/or multiplied to create new kernels.

GP PREDICTIONS

Data

$$\{\mathbf{x}_i, y_i\}_{i=1}^N = (\mathbf{X}, \mathbf{y})$$

Model

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

$$f \sim GP(\mathbf{0}, \mathbf{K}) \qquad (K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j))$$

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

Goal

Predict posterior $P(\mathbf{f}_*|\mathbf{y})$ at test points $\mathbf{X}_*$

GP PREDICTIONS

By GP definition, joint prior on observations and predictions is Gaussian:

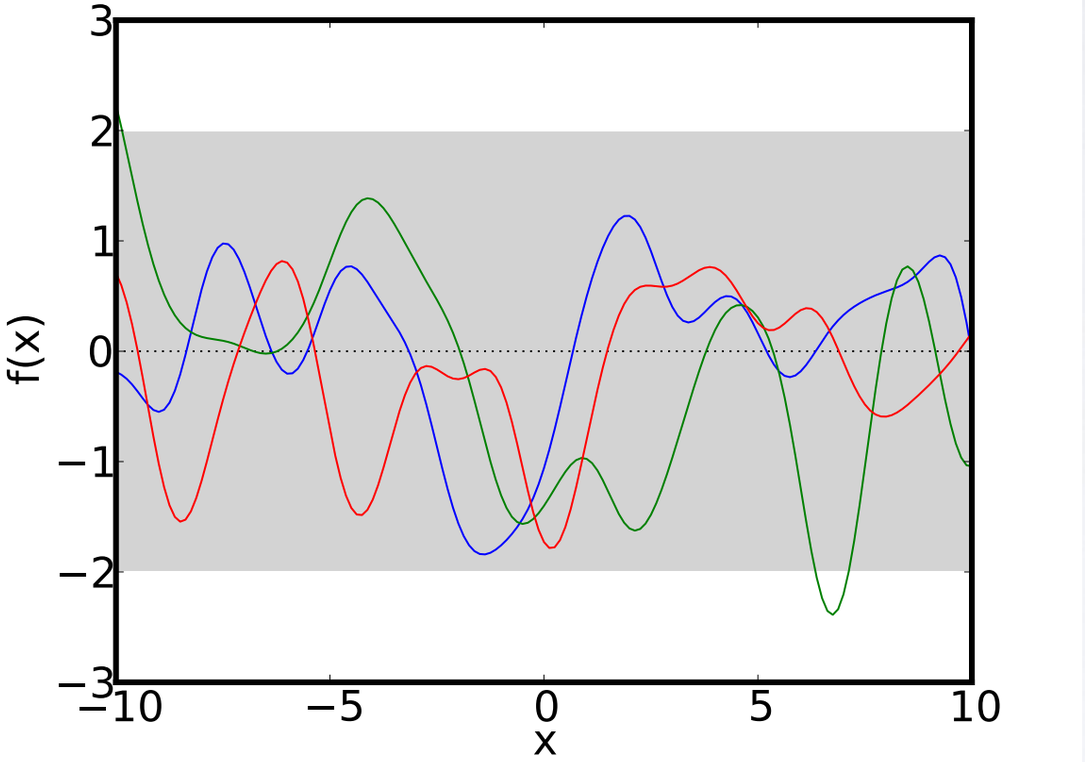
$$P\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix}\right) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I} & \mathbf{K}(\mathbf{X},\mathbf{X}_*)\\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}) & \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*)\end{bmatrix}\right)$$

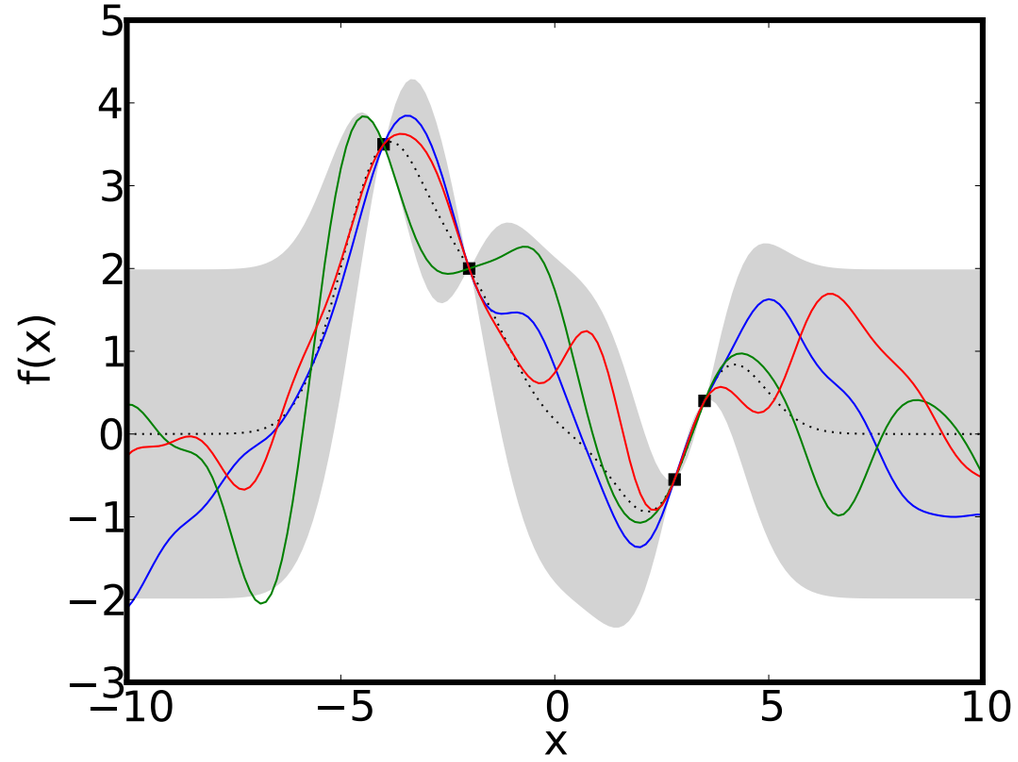
Using some standard results about Gaussian conditionals, you can show that the posterior is $P(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\boldsymbol{\mu} = \mathbf{K}(\mathbf{X}_*,\mathbf{X})[\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}]^{-1}\mathbf{y}$$

$$\boldsymbol{\Sigma} = \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) - \mathbf{K}(\mathbf{X}_*,\mathbf{X})[\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}]^{-1}\mathbf{K}(\mathbf{X},\mathbf{X}_*)$$

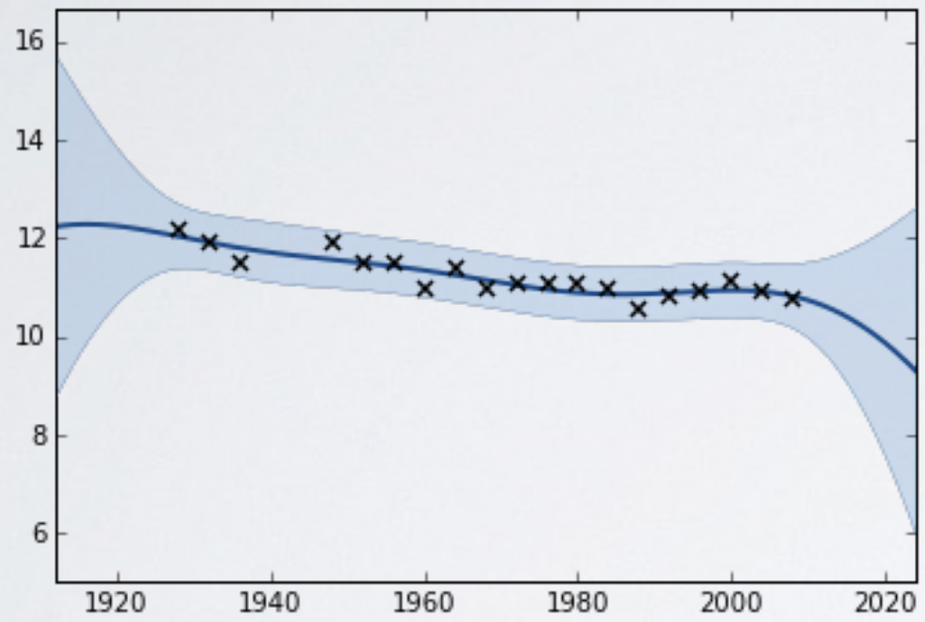
FROM PRIOR TO POSTERIOR

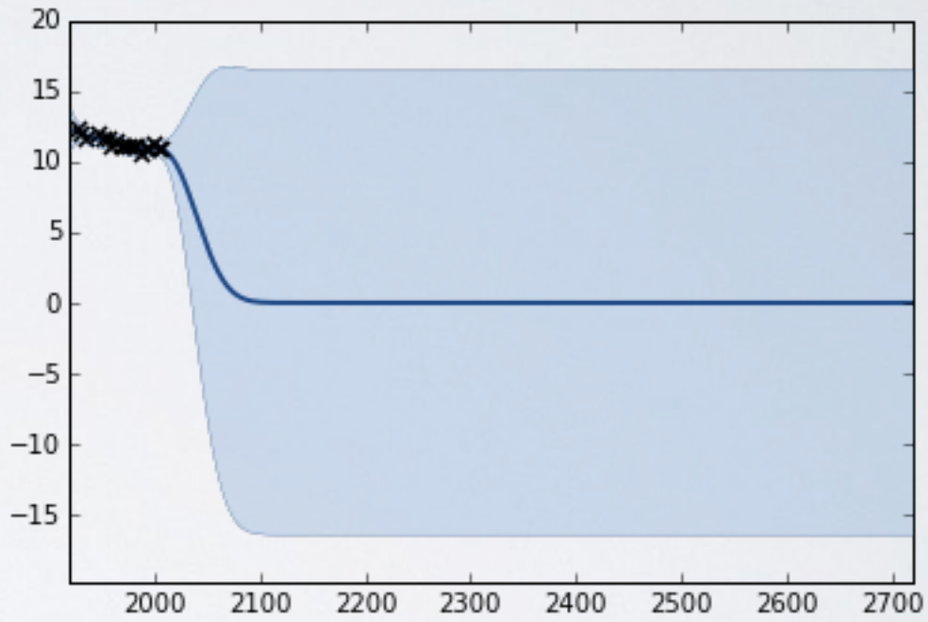




NB: noise-free observations

VISUALISING PREDICTIONS





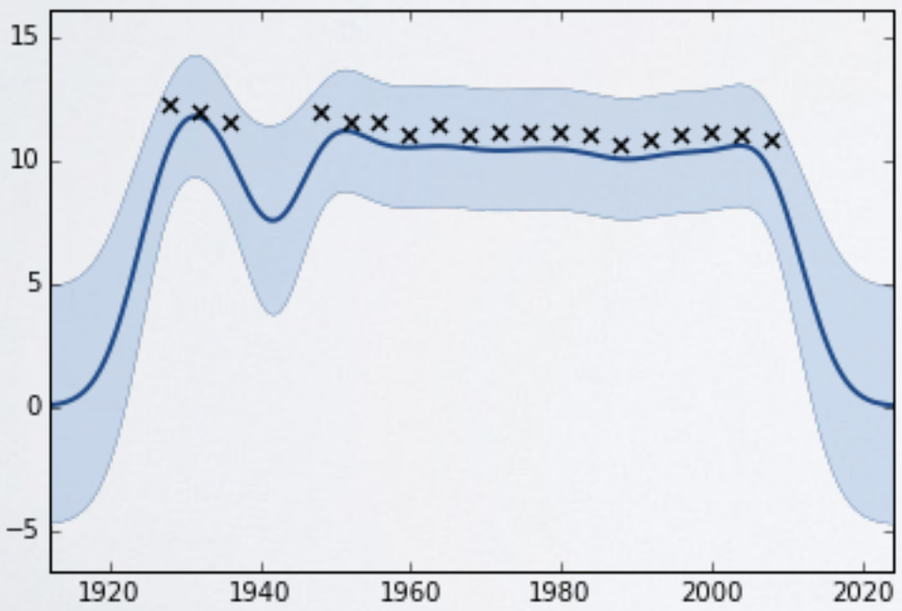
Gaussian kernel - two standard deviation error.
Confidence interval includes all the data.
Huge uncertainty when projecting away from data.

LEARNING IN THE GP

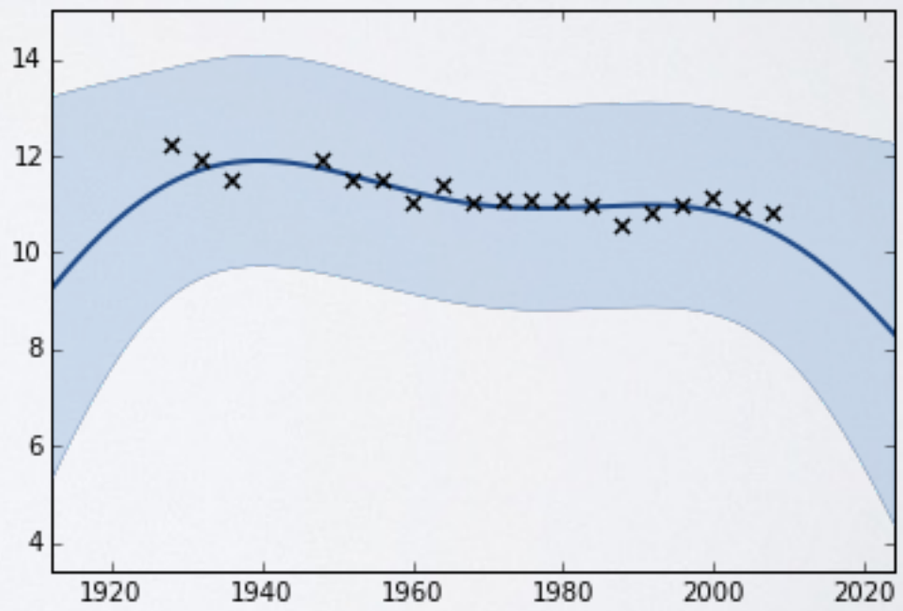
Covariance functions have parameters - how should we choose these?

$$k(\mathbf{x}, \mathbf{z}) = \sigma^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}(x_d - z_d)^2/l_d^2\right)$$

In 1D, 2 **hyperparameters**: a noise variance and a length scale.



$\sigma^2 = l = 5$



$\sigma^2 = 15, l = 30$

MARGINAL LIKELIHOOD

The marginal likelihood provides the probability of the observations given the hyperparameters $\boldsymbol{\theta}$. It's another Gaussian, with log probability:

$$\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\ln\det(\mathbf{K}_{\boldsymbol{\theta}} + \sigma^{\mathbf{2}}\mathbf{I}) - \frac{1}{2}\mathbf{y}^T(\mathbf{K}_{\boldsymbol{\theta}} + \sigma^{\mathbf{2}}\mathbf{I})^{-1}\mathbf{y} + \text{const}$$

This can be optimised as a function of the hyperparameters using standard gradient descent techniques - this is typically the way GPs are trained.

(**NB:** a fully Bayesian approach would be to infer a distribution over the hyperparameters and marginalise some more. This is costly computationally.

COMPARING KERNEL CHOICES

The marginal likelihood also lets us compare different kernels - if the data are more likely under one model, that should be preferred.

In theory we can compare models using only training data, but the marginal likelihood tends to be sensitive to aspects of the prior GP. Always best to keep a test set!

GPY: A PYTHON GP LIBRARY

Thankfully, a number of nice people have implemented the math already. You can get started with GPs easily:

```
import GPy
import pods

data = pods.datasets.olympic_100m_women()
x, y = data['X'], data['Y']

k = GPy.kern.RBF(1)
m = GPy.models.GPRegression(x, y, k)

m.constrain_positive('.*')
m.optimize()
m.plot()
```

EXAMPLE USE: TOUCH OFFSETS

You don't touch where you think you do!

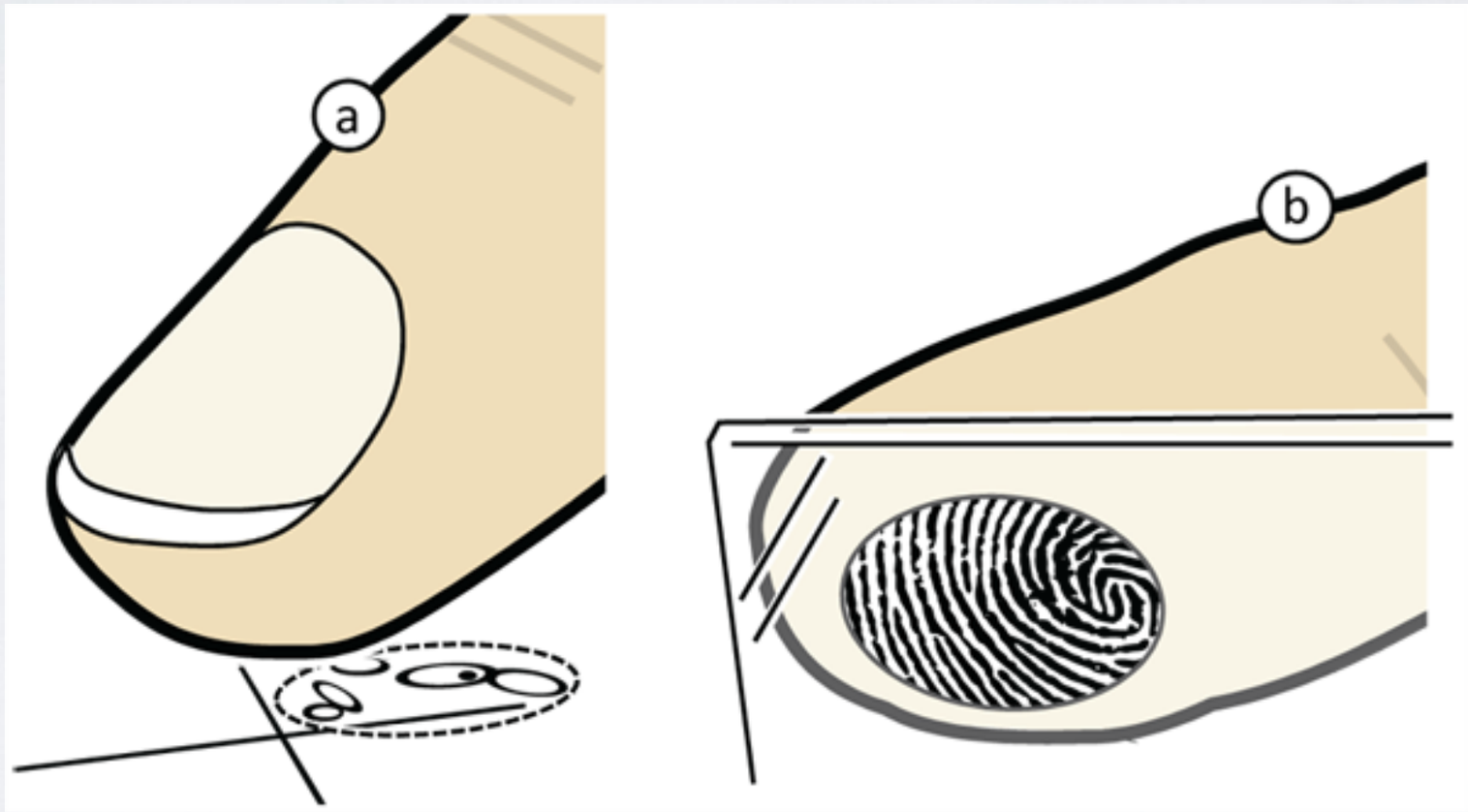


Image: Holz and Baudisch, CHI 2010

MODELLING OFFSETS

This is a regression problem!

Inputs:

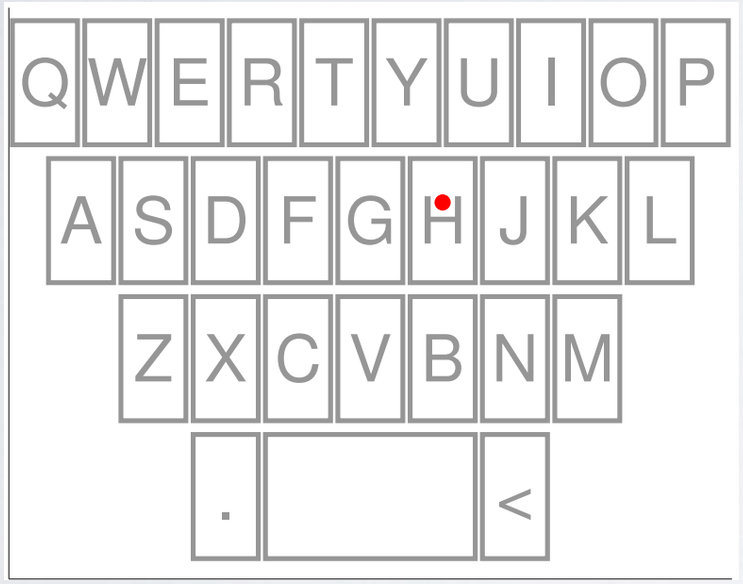
Touch locations recorded by device
Intended targets (e.g. from a crosshair hitting exercise)

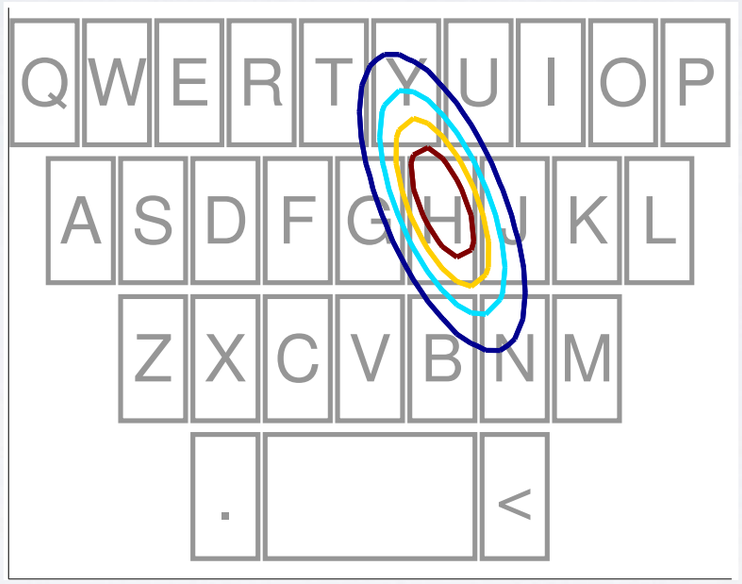
Outputs:

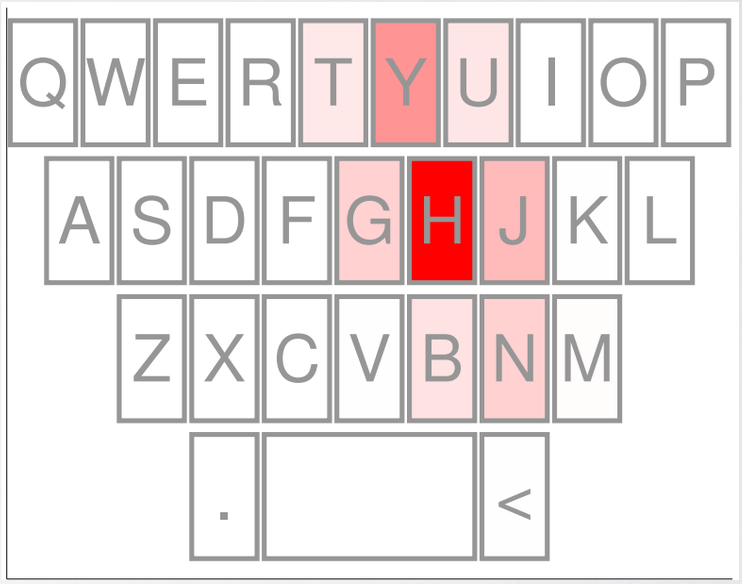
Offsets in x- and y- directions for new recorded touches

NB: need two GPs for this, one to predict each dimension. Multiple output GPs are possible, but outwith scope of this course.

ADVANTAGES OF PREDICTING A DISTRIBUTION







FURTHER READING

- Gaussian Processes for Machine Learning, Rasmussen and Williams, MIT Press, 2006 (www.gaussianprocess.org/gpml)

- Kernel cookbook, http://people.seas.harvard.edu/~dduvenaud/cookbook/

- GP Summer School website, www.gpss.cc

- GPy Python module, gpy.readthedocs.org

- Code for figures in these slides, darylweir.com/gaussianprocceses.ipynb