Machine Learning

Gaussian Processes

Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

October 10, 2017

http://www.carlhenrik.com

Introduction

Kernels

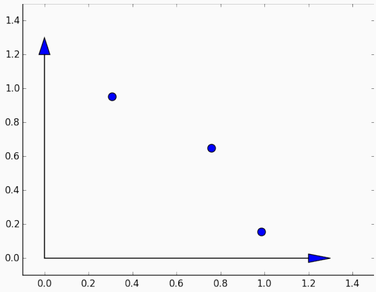
$$k(\mathsf{x}_i, \mathsf{x}_j) = \phi(\mathsf{x}_i)^{\mathrm{T}} \phi(\mathsf{x}_j)$$

- kernel functions describe inner-products in an induced representation
- induced representation lives in what is called a Hilbert Space
- importantly the space is metric

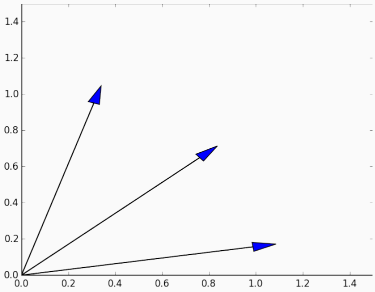
Kernels

$$\begin{aligned}\sigma(\mathsf{X}, \mathsf{Y}) &= \mathbb{E}\left[(\mathsf{X} - \mathbb{E}[\mathsf{X}])^{\mathrm{T}}(\mathsf{Y} - \mathbb{E}[\mathsf{Y}])\right] = \\ &= \mathbb{E}[\mathsf{X}^{\mathrm{T}}\mathsf{Y}] - \mathbb{E}[\mathsf{X}]^{\mathrm{T}}\mathbb{E}[\mathsf{Y}] = \{\mathbb{E}[\mathsf{X}] = \mathbb{E}[\mathsf{Y}] = \mathbf{0}\} = \\ &= \mathbb{E}[\mathsf{X}^{\mathrm{T}}\mathsf{Y}]\end{aligned}$$

Kernels



1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4



1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4

Kernels

$$\sigma(\mathbf{X}, \mathbf{Y}) = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \end{bmatrix} \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \end{bmatrix} =$$

$$= \begin{bmatrix} x_{11}y_{11} + x_{21}y_{21} + x_{31}y_{31} & x_{11}y_{12} + x_{21}y_{22} + x_{31}y_{32} \\ x_{12}y_{11} + x_{22}y_{21} + x_{32}y_{31} & x_{12}y_{12} + x_{22}y_{22} + x_{32}y_{32} \end{bmatrix}$$

Kernels

$$\sigma(\mathbf{X},\mathbf{Y}) = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \end{bmatrix} \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \end{bmatrix} =$$

$$= \begin{bmatrix} x_{11}y_{11} + x_{21}y_{21} + x_{31}y_{31} & x_{11}y_{12} + x_{21}y_{22} + x_{31}y_{32} \\ x_{12}y_{11} + x_{22}y_{21} + x_{32}y_{31} & x_{12}y_{12} + x_{22}y_{22} + x_{32}y_{32} \end{bmatrix}$$

$$\sigma(\mathbf{X}^{\mathrm{T}},\mathbf{Y}^{\mathrm{T}}) = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \begin{bmatrix} y_{11} & y_{21} & y_{31} \\ y_{12} & y_{22} & y_{32} \end{bmatrix} =$$

$$= \begin{bmatrix} x_{11}y_{11} + x_{12}y_{12} & x_{11}y_{21} + x_{12}y_{22} & x_{11}y_{31} + x_{12}y_{32} \\ x_{21}y_{11} + x_{22}y_{12} & x_{21}y_{21} + x_{22}y_{22} & x_{21}y_{31} + x_{22}y_{32} \\ x_{31}y_{11} + x_{32}y_{12} & x_{31}y_{21} + x_{32}y_{22} & x_{31}y_{31} + x_{32}y_{32} \end{bmatrix}$$

Kernels and Covariances

- Covariance between columns: $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ (data-dimensions)
- Covariance between rows: $\mathbf{X}\mathbf{Y}^{\mathrm{T}}$ (data-points)
- Kernels: $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{y})$
- A kernel function describes the co-variance of the data points

Kernel Example

$$k(\mathsf{x}_i, \mathsf{x}_j) = \sigma^2 e^{-\frac{1}{2\ell^2}(\mathsf{x}_i - \mathsf{x}_j)^{\mathrm{T}}(\mathsf{x}_i - \mathsf{x}_j)}$$

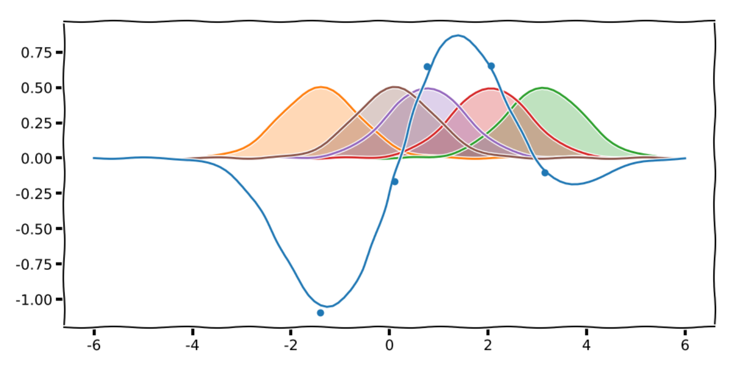
Exponented Quadratic

- How does the data vary along the dimensions spanned by the data
- RBF, Squared Exponential, Exponentiated Quadratic
- Co-variance smoothly decays with distance
- You can build new kernels out of other kernels [1] p. 296

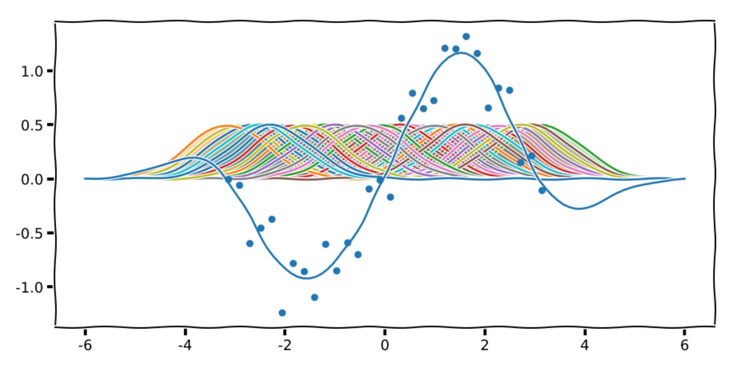
Kernel Regression

$$
\begin{aligned}
\mathbf{y}(\mathbf{x}_*) &= \mathbf{w}^{\mathrm{T}}\mathbf{x}_* = \mathbf{a}^{\mathrm{T}}\mathbf{x}\mathbf{x}_* = \mathbf{a}^{\mathrm{T}}k(\mathbf{x},\mathbf{x}_*) = \\
&= ((\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{t})^{\mathrm{T}}k(\mathbf{x},\mathbf{x}_*) = k(\mathbf{x}_*,\mathbf{x})(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{t} \\
\lambda &= \frac{\sigma^2}{\tau^2} \\
p(\mathbf{t}|\mathbf{w},\mathbf{x}) &= \prod_n^N p(t_n|\mathbf{w},\mathbf{x}) = \prod_n^N \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\mathbf{x}_n,\sigma^2\mathbf{I}) \\
p(\mathbf{w}) &= \mathcal{N}(\mathbf{0},\tau^2\mathbf{I})
\end{aligned}
$$

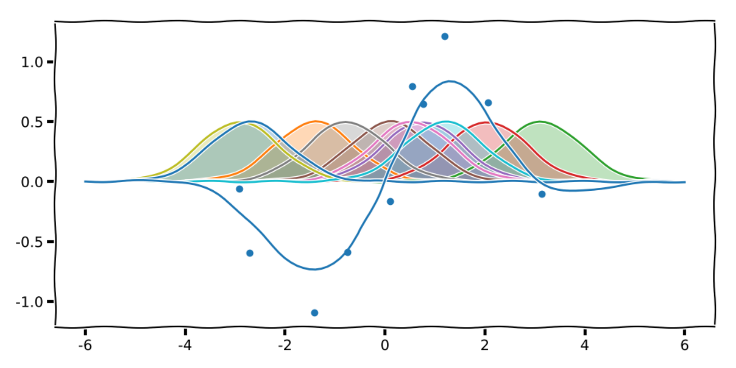
Kernel Regression



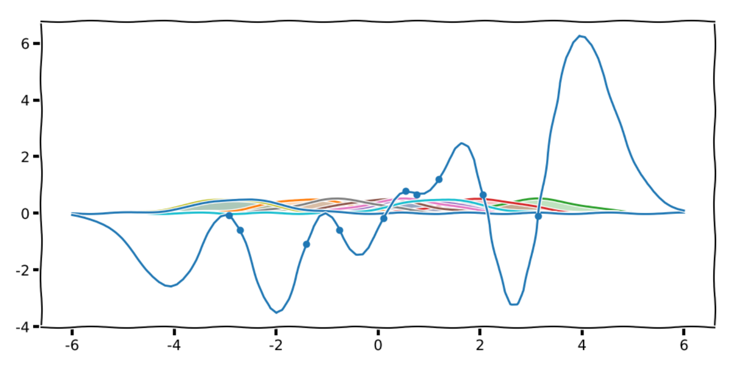
Kernel Regression



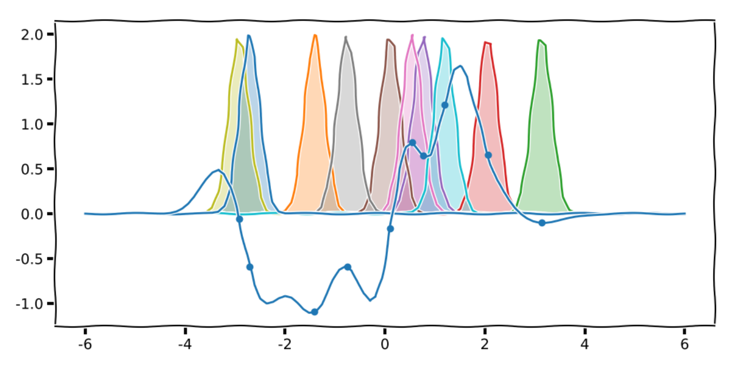
Kernel Regression



Kernel Regression



Kernel Regression

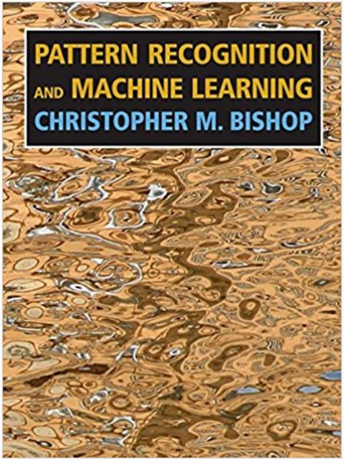


Question

- *How do you know which basis function to choose?*
- *Would you call this overfitting?*

Gaussian Processes

Book


PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

GPSS[1]

IUDICIUM POSTERIUM DISCIPULUS EST PRIORIS

[1] `http://gpss.cc`

Uncertainty

- We have uncertainty in our observed outputs

Uncertainty

- We have uncertainty in our observed outputs
- We have no uncertainty in our mapping

Uncertainty

- We have uncertainty in our observed outputs
- We have no uncertainty in our mapping
    - Linear, it is a line

Uncertainty

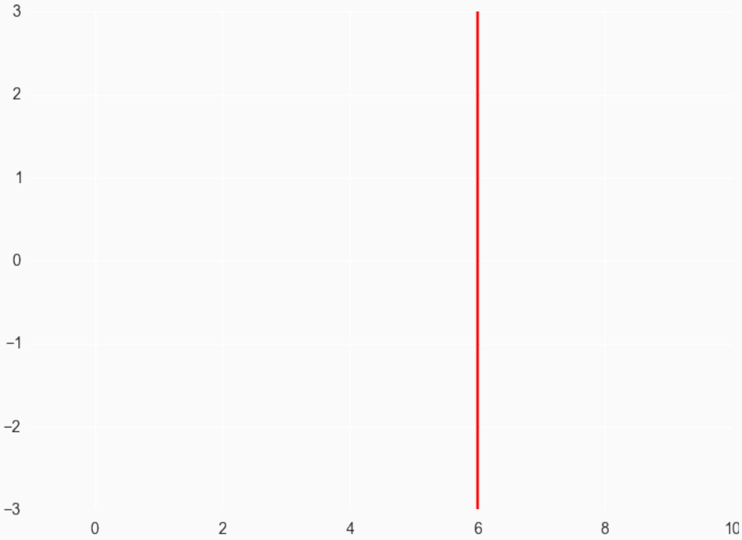
- We have uncertainty in our observed outputs
- We have no uncertainty in our mapping
    - Linear, it is a line
    - Kernels, it is this specific basis function

Uncertainty

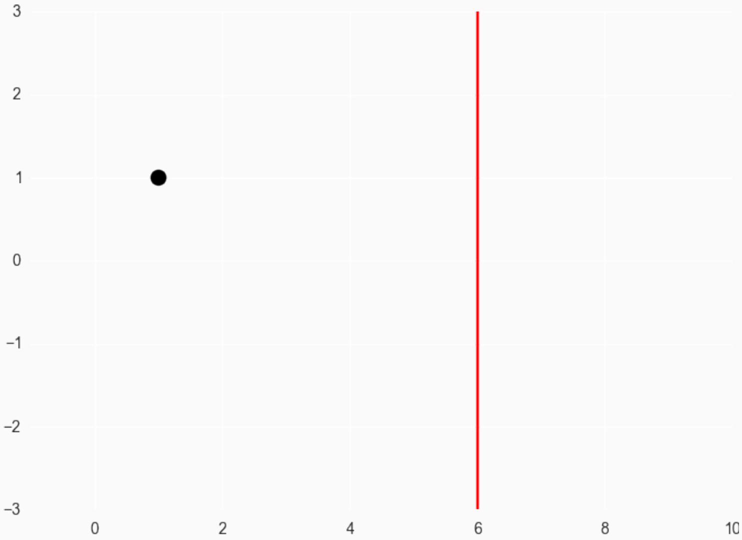
- We have uncertainty in our observed outputs
- We have no uncertainty in our mapping
  - Linear, it is a line
  - Kernels, it is this specific basis function
- *need a prior assumption over the space of functions*

Functions

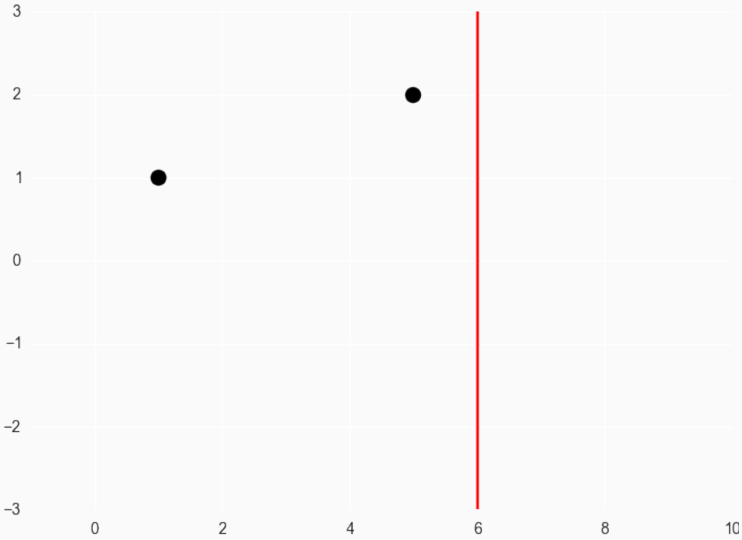
Functions



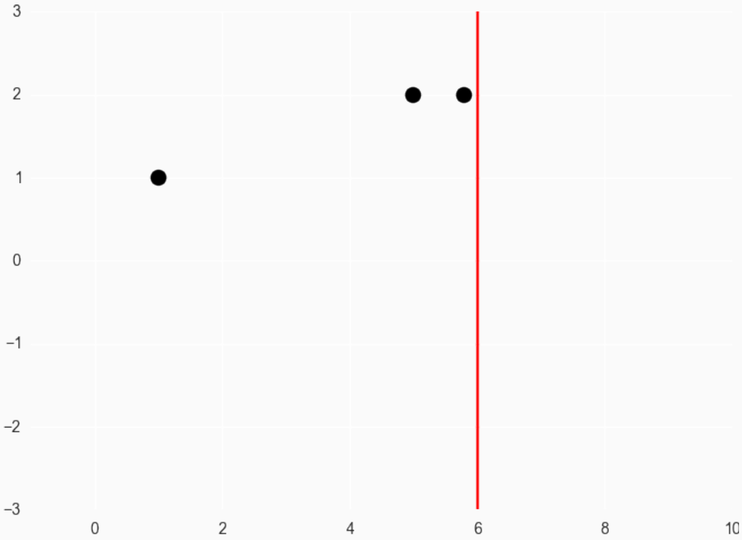
Functions



Functions



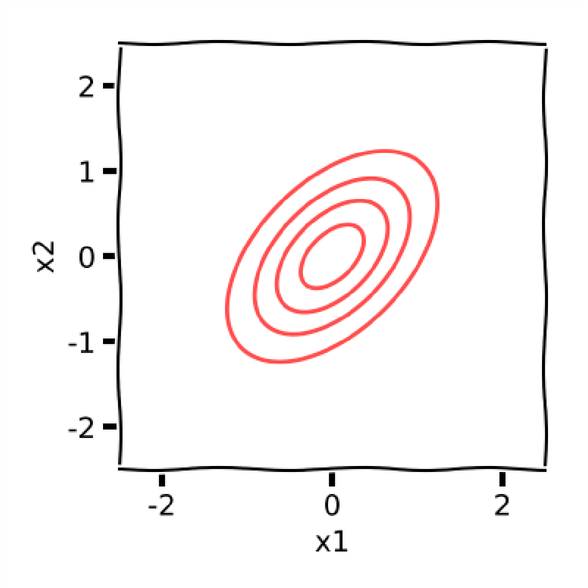
Functions



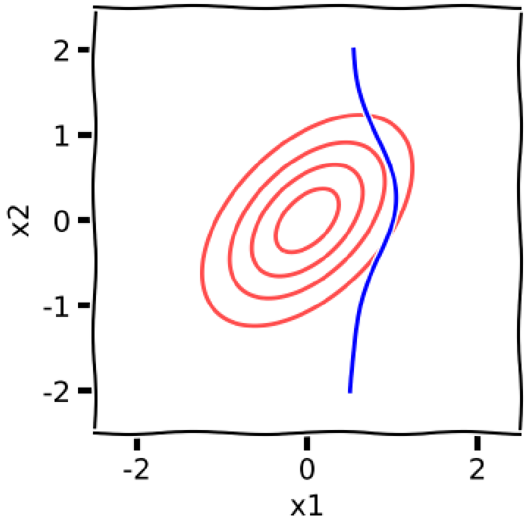
Conditional Gaussians

$$\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}\right)$$

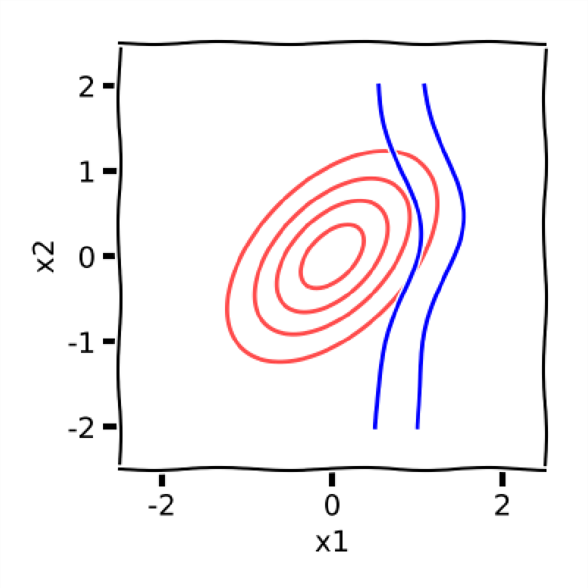
Conditional Gaussians



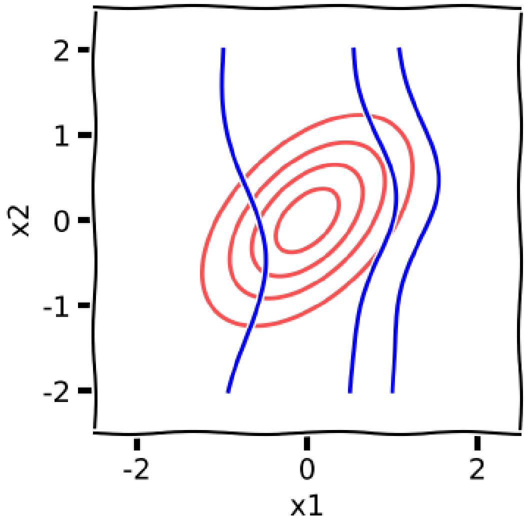
Conditional Gaussians



Conditional Gaussians



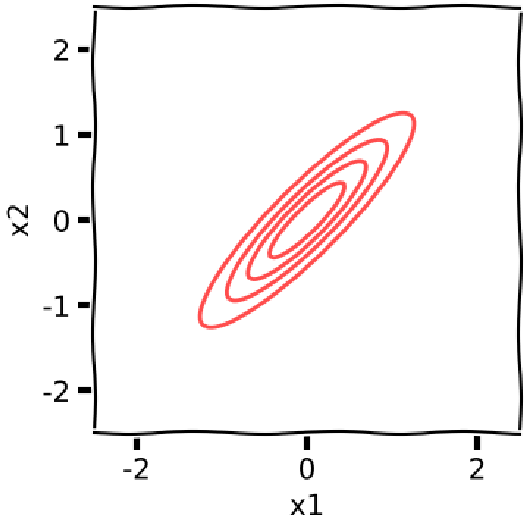
Conditional Gaussians



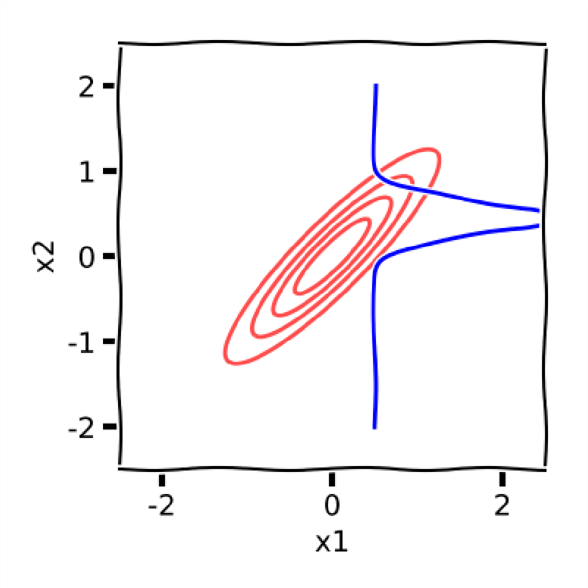
Conditional Gaussians

$$\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}\right)$$

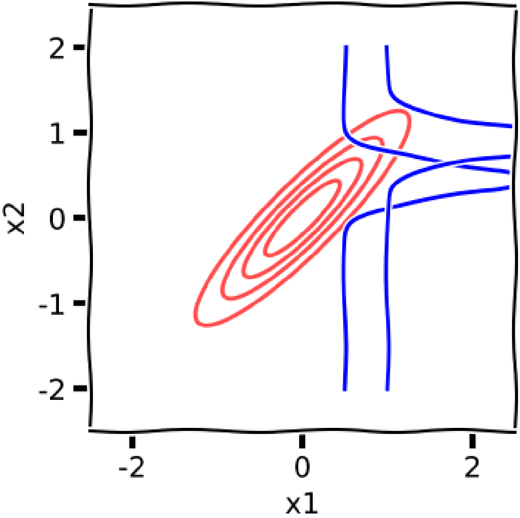
Conditional Gaussians



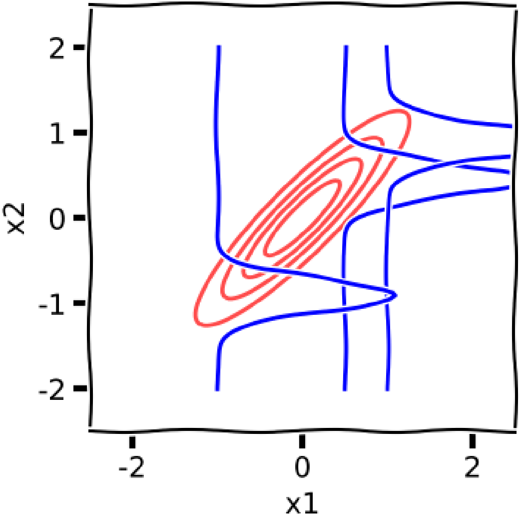
Conditional Gaussians



Conditional Gaussians



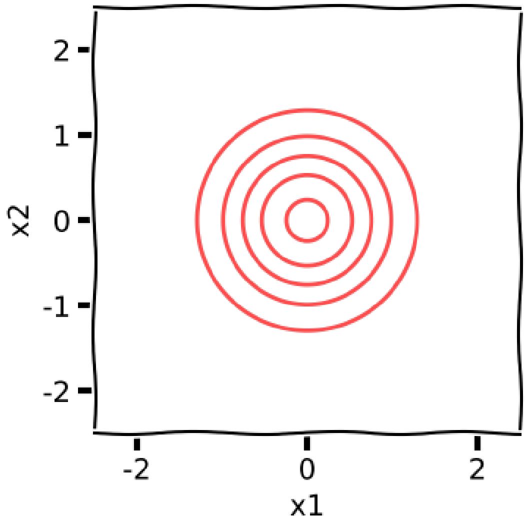
Conditional Gaussians



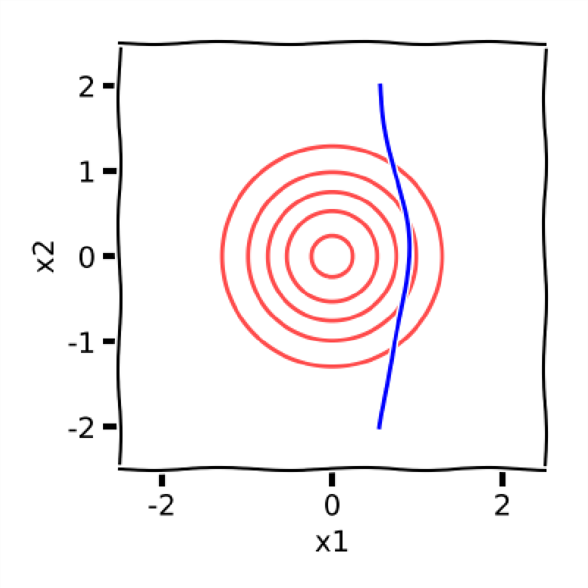
Conditional Gaussians

$$\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

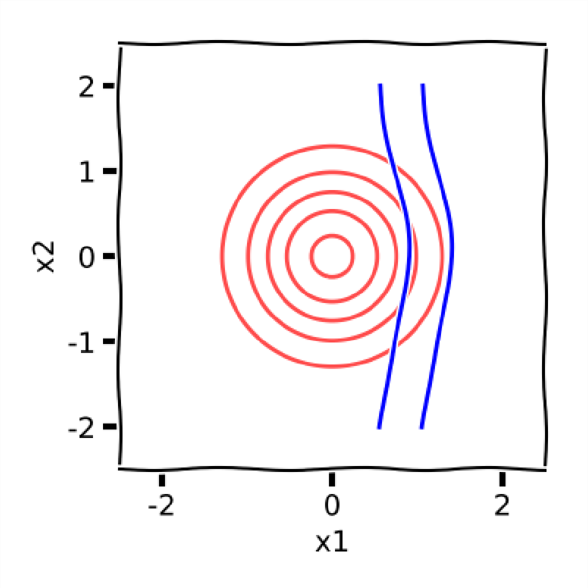
Conditional Gaussians



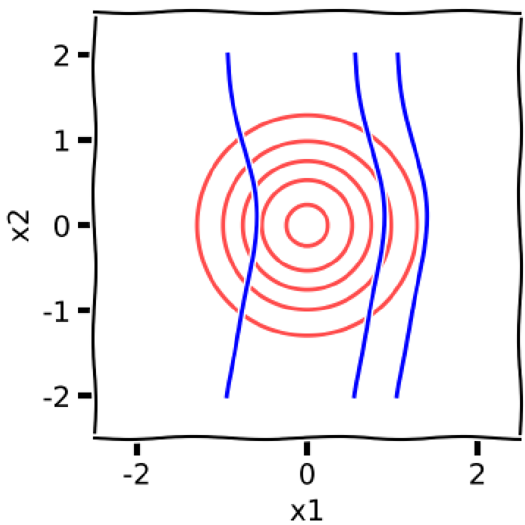
Conditional Gaussians



Conditional Gaussians

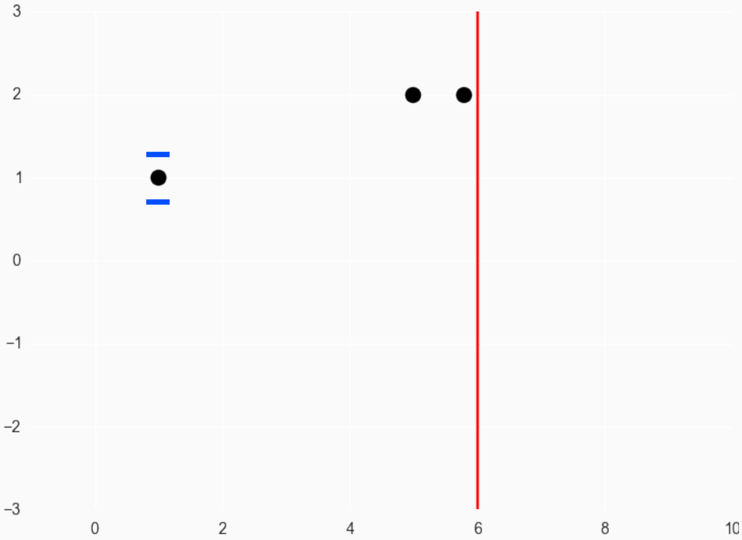


Conditional Gaussians

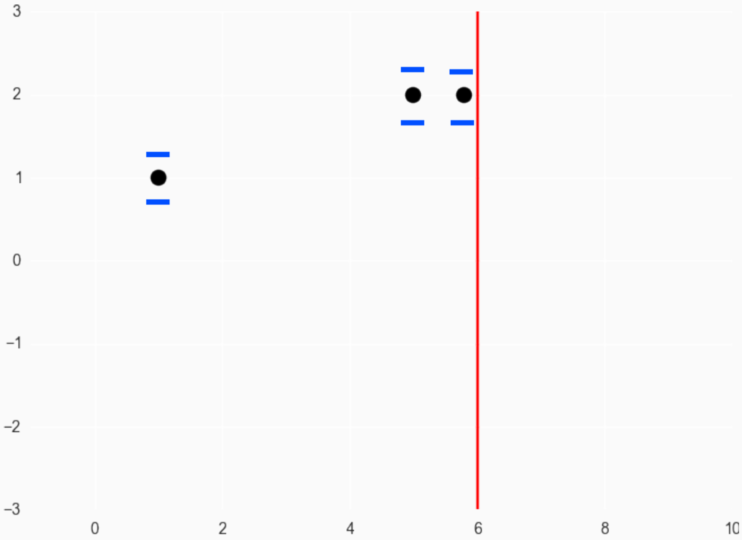


Eureka

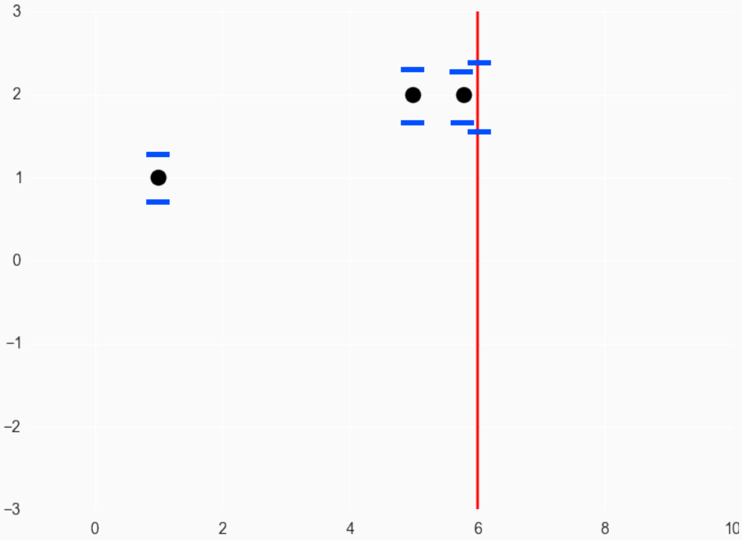
Functions



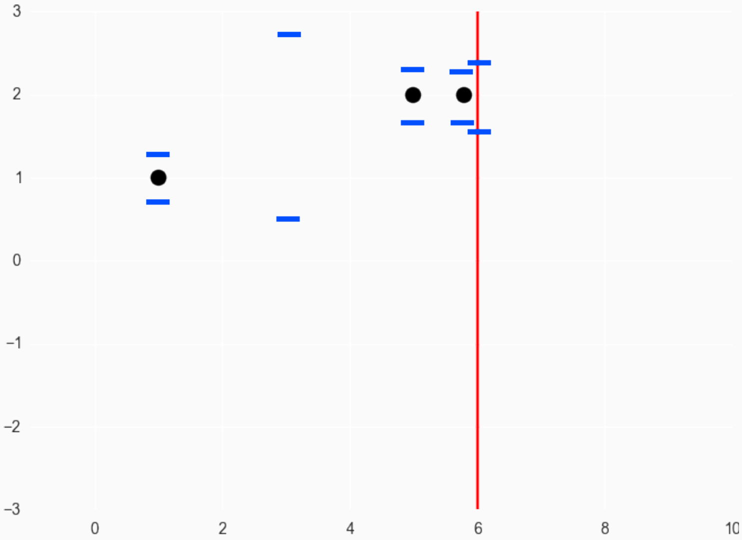
Functions



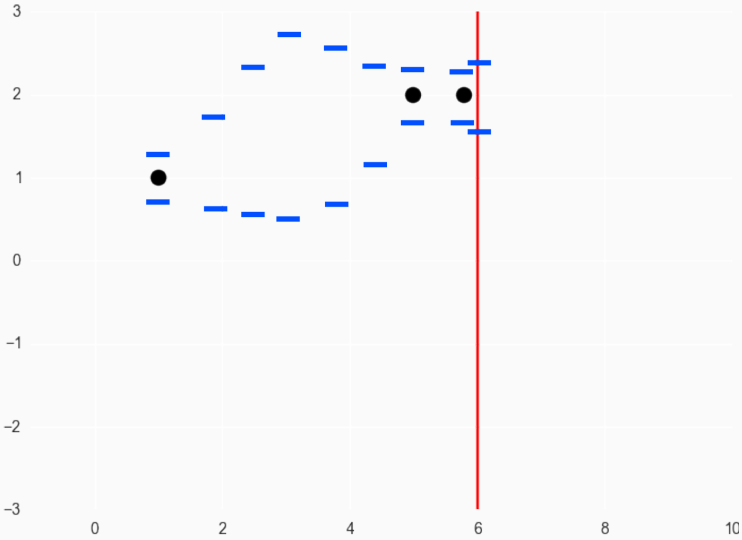
Functions



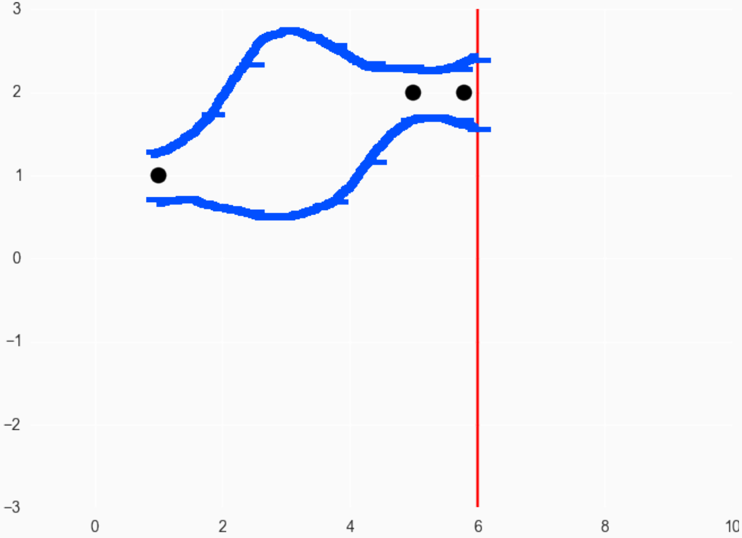
Functions



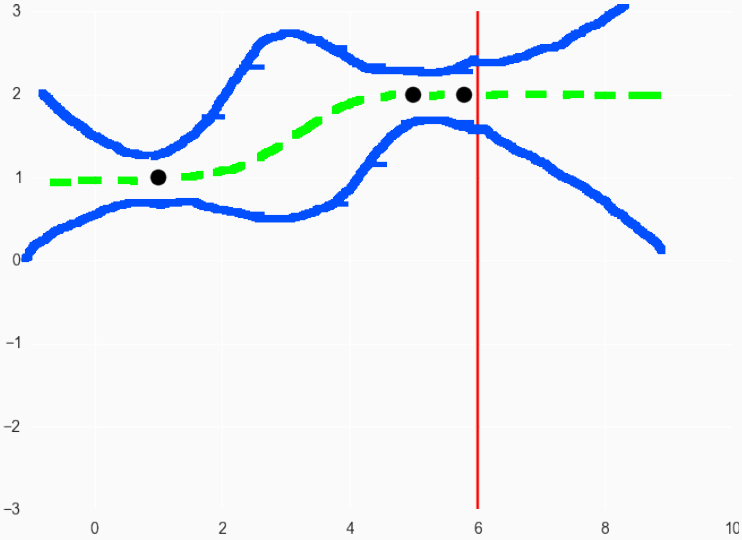
Functions



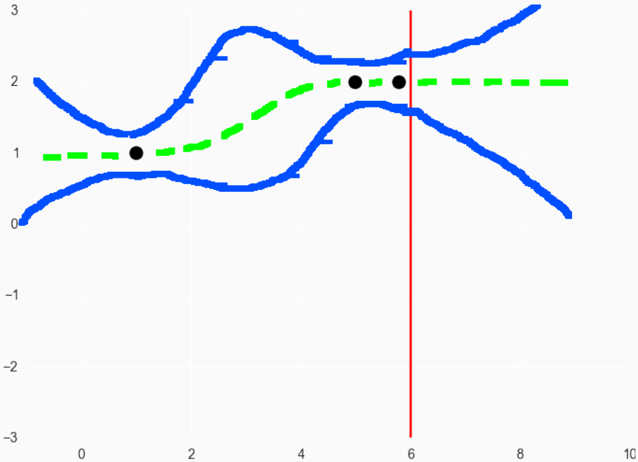
Functions



Functions



Functions



If all instantiations of the function is jointly Gaussian such that the co-variance structure depends on how much information an observation provides for the other we will get the curve above.

Uncertainty over functions

- Regression model,

$$\mathbf{y}_i = f(\mathbf{x}_i) + \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

- Introduce f_i as *instansiation* of function,

$$f_i = f(\mathbf{x}_i),$$

- as a new random variable.

Uncertainty over functions

- Regression model,

$$\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon$$
$$\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

- Introduce f_i as *instansiation* of function,

$$f_i = f(\mathbf{x}_i),$$

- as a new random variable.
- now we have a "handle" to specify our assumptions over

Uncertainty over functions

Model,

$$p(\mathbf{y}, \mathbf{f}, \mathbf{x}, \boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{x}, \boldsymbol{\theta})p(\mathbf{x})p(\boldsymbol{\theta})$$

Want to "push" $\mathbf{x}$ through a mapping f of which we are uncertain,

$$p(\mathbf{f}|\mathbf{x}, \boldsymbol{\theta}),$$

prior over instantiations of function.

Gaussian Processes [1] Ch 6.4.2

$$p(\mathbf{f}|\mathbf{x}, \boldsymbol{\theta}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}))$$

Definition (Gaussian Process)

A Gaussian Process is an infinite collection of random variables who any subset is jointly gaussian. The process is specified by a mean function $\mu(\cdot)$ and a co-variance function $k(\cdot, \cdot)$

Gaussian Processes

$$p(\mathbf{f}|\mathbf{x},\boldsymbol{\theta}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x},\mathbf{x}))$$

$$\mathbf{y}_i = f_i + \boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

$$p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{x},\boldsymbol{\theta})\mathrm{d}f$$

$\mathcal{GP}$ is infinite, but we only observe finite amount of data. This means conditioning on a subset of the data, the $\mathcal{GP}$ is a just a Gaussian distribution, which is self-conjugate and we know how to do everything

The Mean Function

- Function of only the input location
- What do I expect the function value to be only accounting for the input location

The Covariance Function

- Function of two input locations
- How should the information from other locations with known function value observations effect my estimate

The Mean Function

- Function of only the input location
- What do I expect the function value to be only accounting for the input location
- We will assume this to be constant

The Covariance Function

- Function of two input locations
- How should the information from other locations with known function value observations effect my estimate

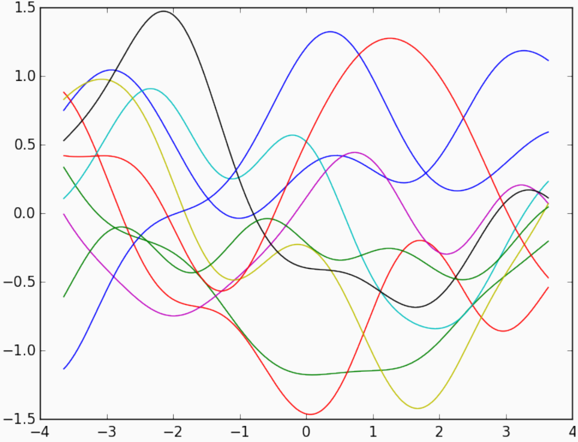
The Mean Function

- Function of only the input location
- What do I expect the function value to be only accounting for the input location
- We will assume this to be constant

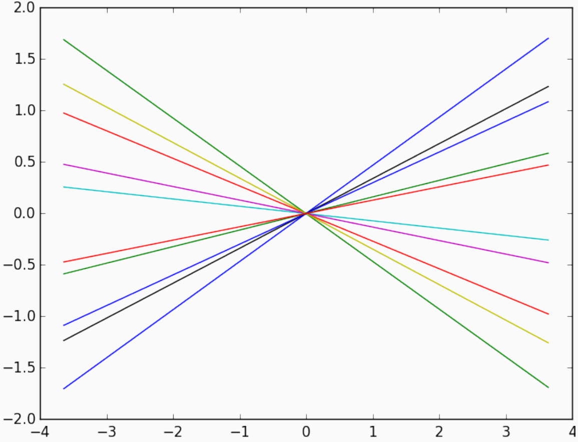
The Covariance Function

- Function of two input locations
- How should the information from other locations with known function value observations effect my estimate
- Encodes the behavior of the function

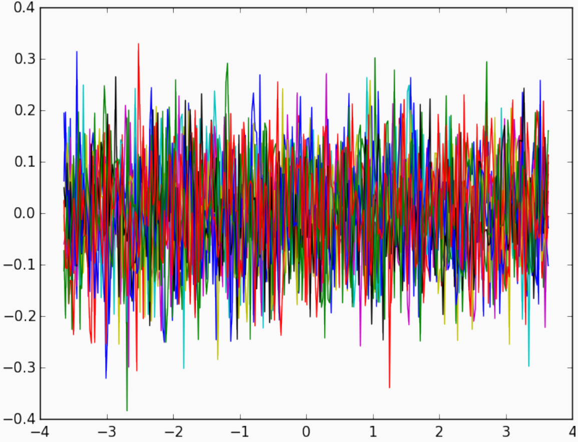
Gaussian Processes: Samples



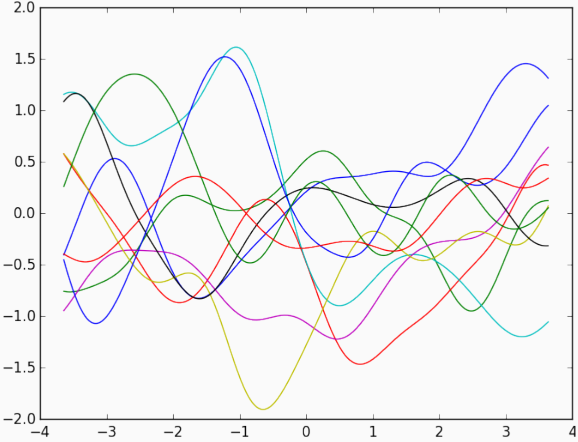
Gaussian Processes: Samples



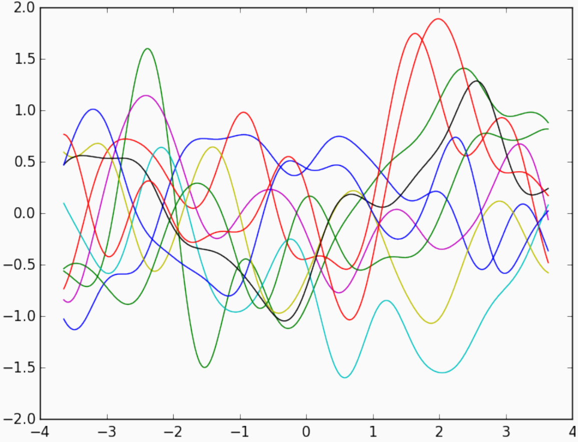
Gaussian Processes: Samples



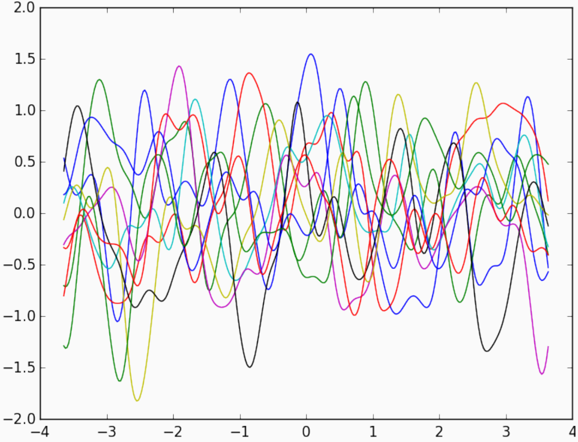
Gaussian Processes: Samples



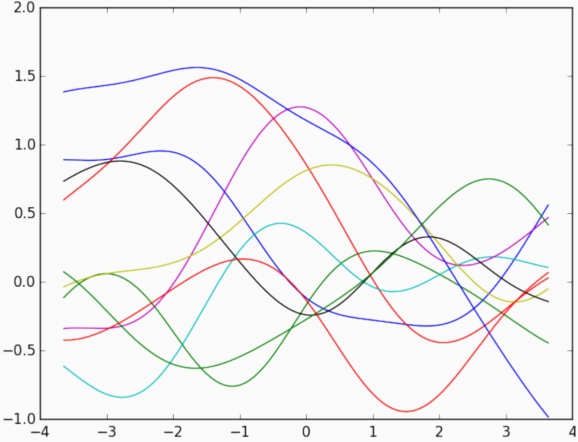
Gaussian Processes: Samples



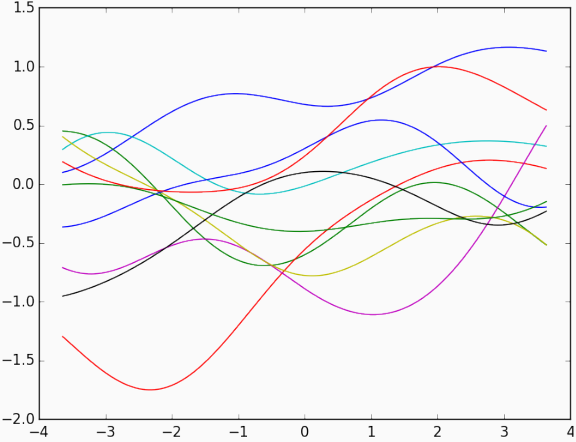
Gaussian Processes: Samples



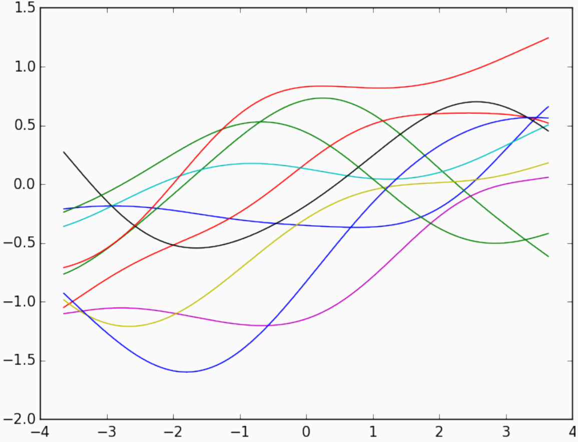
Gaussian Processes: Samples



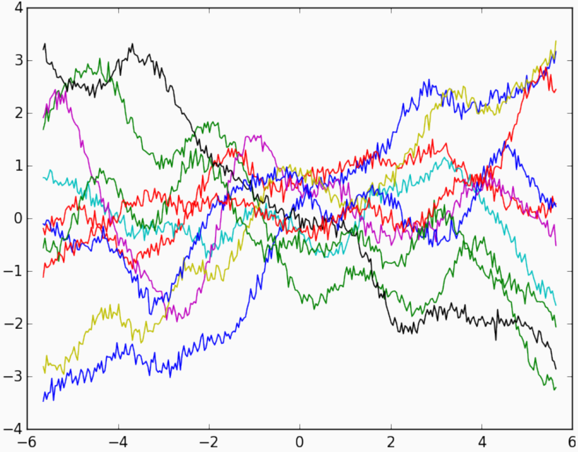
Gaussian Processes: Samples



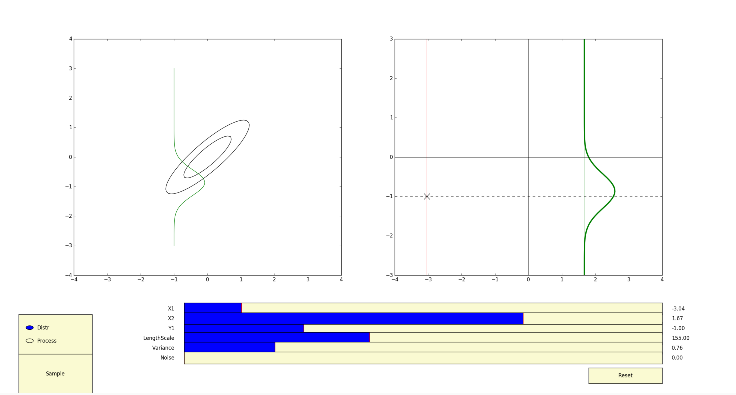
Gaussian Processes: Samples



Gaussian Processes: Samples

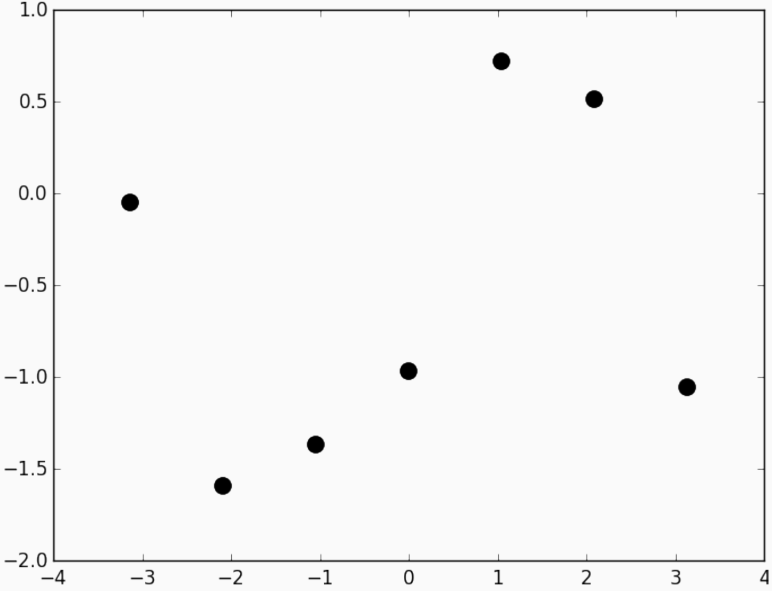


Demo



Distr
Process
Sample
X1
X2
Y1
LengthScale
Variance
Noise
-3.04
1.67
-1.00
155.00
0.76
0.00
Reset

Gaussian Processes: Prediction



Gaussian Processes: Prediction

- All instantiations are jointly Gaussian

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

Gaussian Processes: Prediction

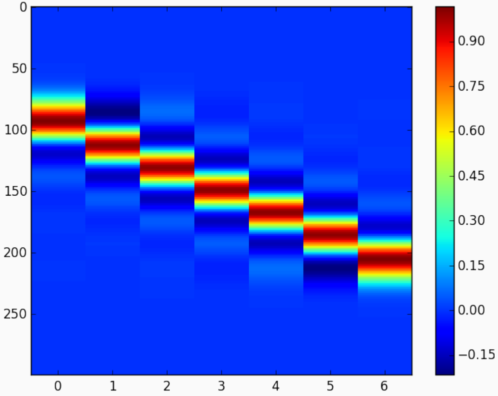
- All instantiations are jointly Gaussian

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

- Conditional (same as always)

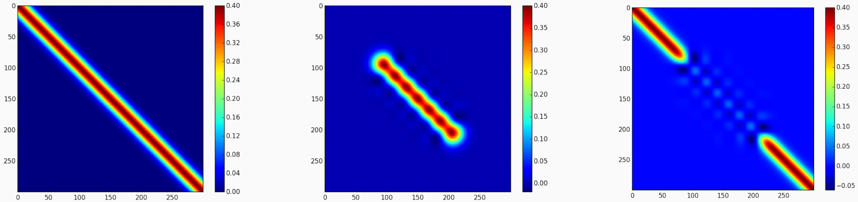
$$\begin{aligned} p(f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(& k(\mathbf{x}_*, \mathbf{X})^{\mathrm{T}} K(\mathbf{X}, \mathbf{X})^{-1} \mathbf{f}, \\ & k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})^{\mathrm{T}} K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{x}_*)) \end{aligned}$$

Gaussian Processes: Prediction



$$k(\mathsf{x}_*, \mathsf{x})^{\mathrm{T}} K(\mathsf{x}, \mathsf{x})^{-1} \mathbf{f}$$

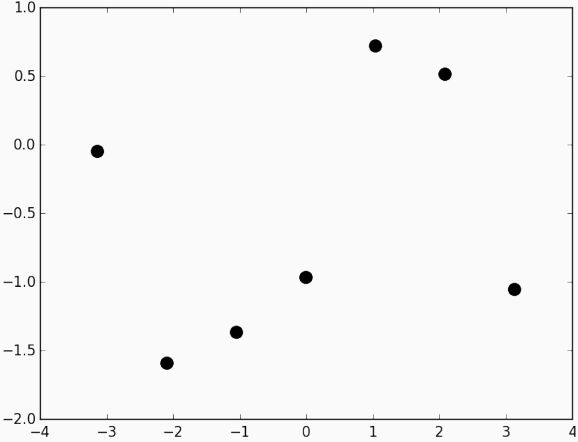
Gaussian Processes: Prediction



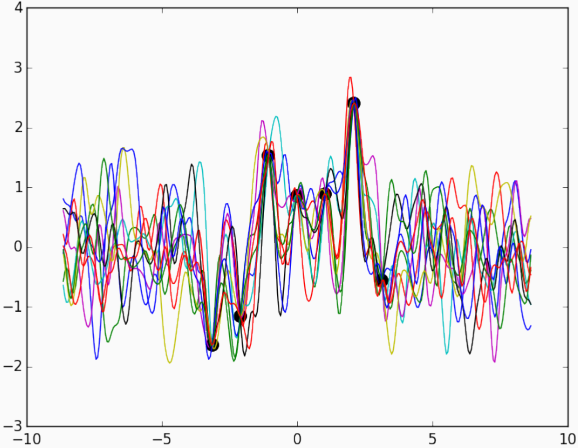
Covariance

$$k(\mathsf{x}_*, \mathsf{x}_*) - k(\mathsf{x}_*, \mathsf{x})^{\mathrm{T}} K(\mathsf{x}, \mathsf{x})^{-1} K(\mathsf{x}, \mathsf{x}_*)$$

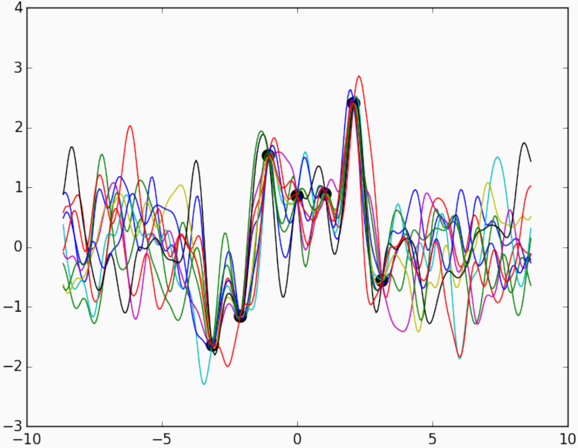
Gaussian Processes: Prediction



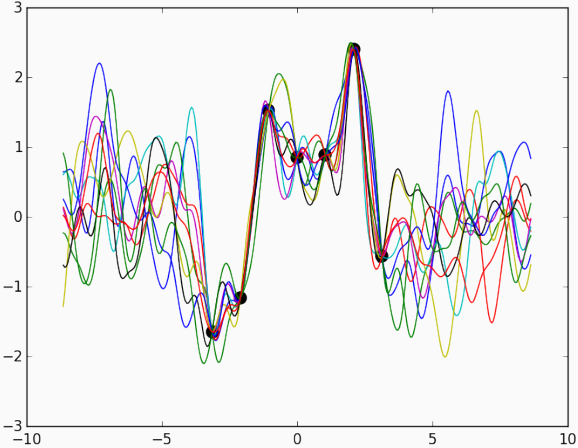
Gaussian Processes: Prediction



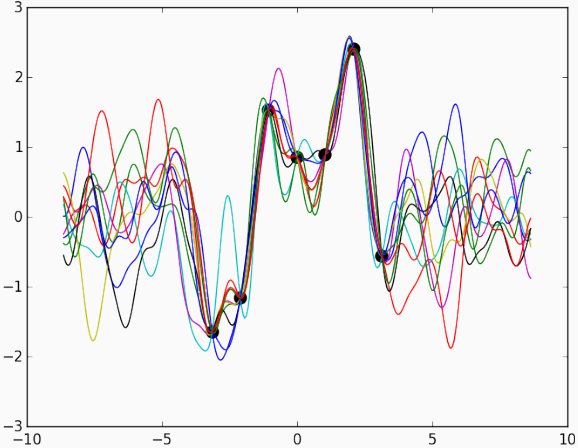
Gaussian Processes: Prediction



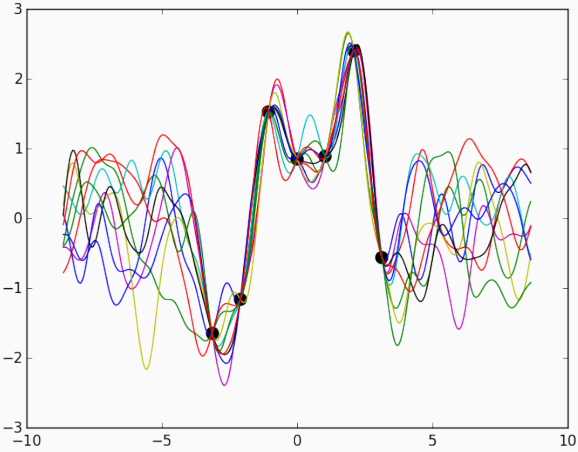
Gaussian Processes: Prediction



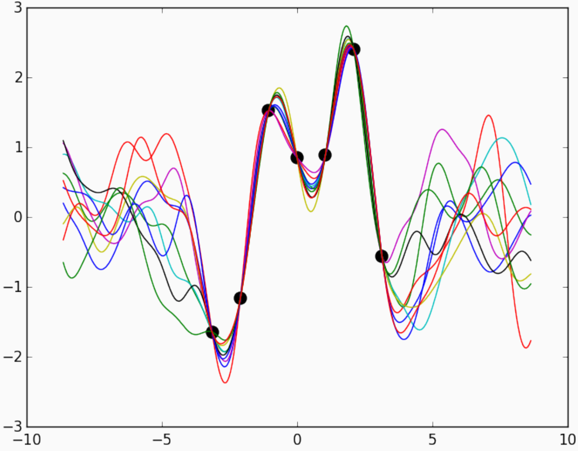
Gaussian Processes: Prediction



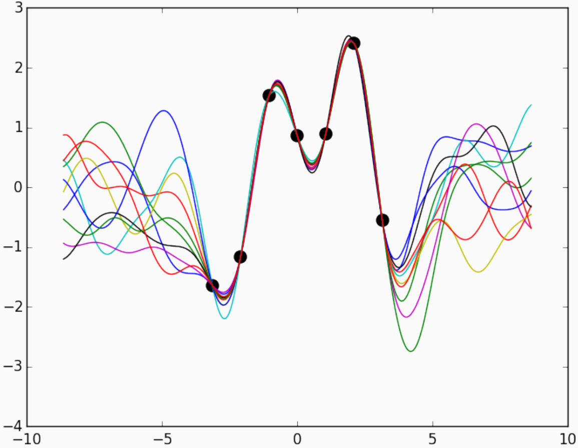
Gaussian Processes: Prediction



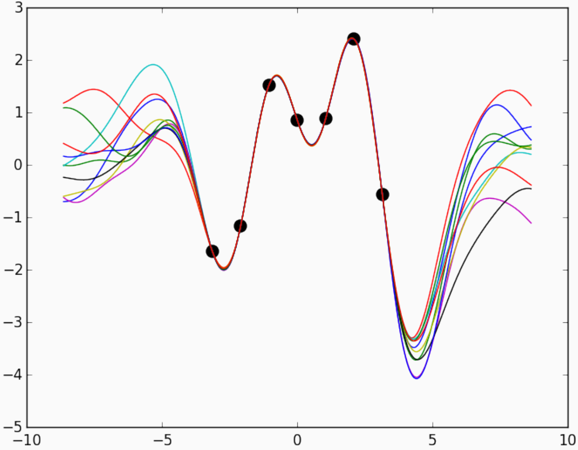
Gaussian Processes: Prediction



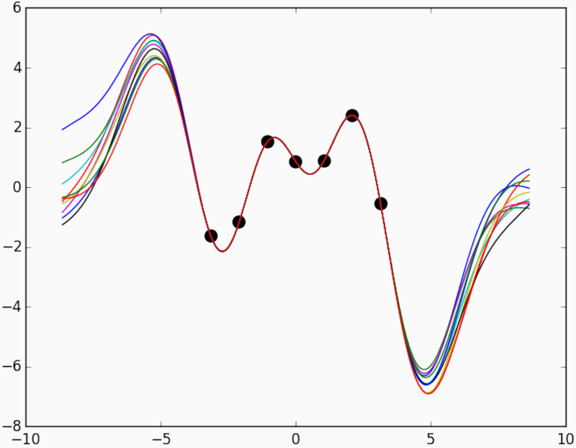
Gaussian Processes: Prediction



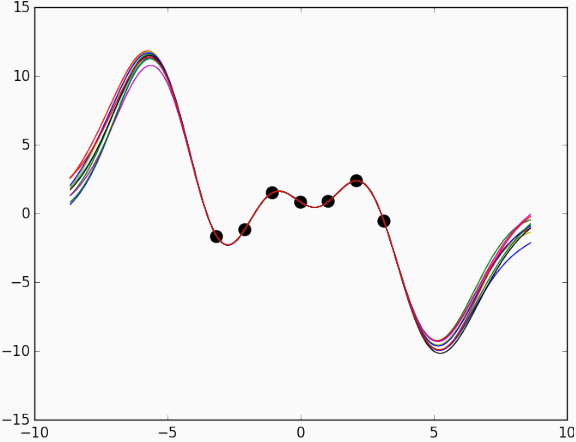
Gaussian Processes: Prediction



Gaussian Processes: Prediction



Gaussian Processes: Prediction



Gaussian Processes: Noisy observations

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

$$\begin{aligned} p(f_*|\mathbf{x}_*,\mathbf{x},\mathbf{f},\boldsymbol{\theta}) = \mathcal{N}(&k(\mathbf{x}_*,\mathbf{x})^{\mathrm{T}}(K(\mathbf{x},\mathbf{x}+\sigma^2\mathbf{I}))^{-1}\mathbf{f}, \\ &k(\mathbf{x}_*,\mathbf{x}_*) - k(\mathbf{x}_*,\mathbf{x})^{\mathrm{T}}(K(\mathbf{x},\mathbf{x})+\sigma^2\mathbf{I})^{-1}K(\mathbf{x},\mathbf{x}_*)) \end{aligned}$$

- Add noise to observations

Gaussian Processes: Noisy observations

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

$$\begin{aligned} p(f_*|\mathbf{x}_*, \mathbf{x}, \mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(&k(\mathbf{x}_*, \mathbf{x})^{\mathrm{T}}(K(\mathbf{x}, \mathbf{x} + \sigma^2\mathbf{I}))^{-1}\mathbf{f}, \\ &k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x})^{\mathrm{T}}(K(\mathbf{x}, \mathbf{x}) + \sigma^2\mathbf{I})^{-1}K(\mathbf{x}, \mathbf{x}_*)) \end{aligned}$$

- Add noise to observations
- *Do you recognise the mean?*

Question 1-14

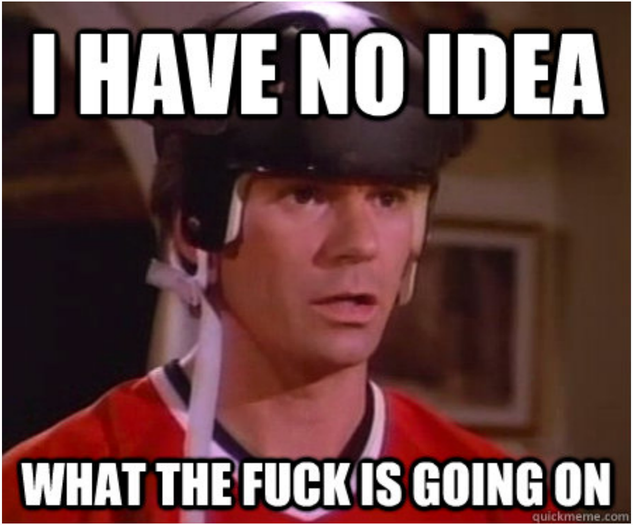
Summary

Summary

- Repeat of the machine learning proceedure
  - assumption + data + compute $\rightarrow$ updated assumption
  - don't worry it will become clear eventually
- Gaussian processes
  - infinite generalisation of Gaussian distribution
  - prior over the space of functions
  - contains all functions

eof

Part II


I HAVE NO IDEA
WHAT THE FUCK IS GOING ON
quickmeme.com

I don't think so

$$p(\text{ML}|\text{COMS30007}) = \frac{p(\text{COMS30007}|\text{ML})p(\text{ML})}{p(\text{COMS30007})}$$

The Unit so far

- Why is Machine Learning Hard
  - we don't learn how to solve a specific task
  - we learn how to learn how to solve every task
  - its meta knowledge

The Unit so far

- Why is Machine Learning Hard
  - we don't learn how to solve a specific task
  - we learn how to learn how to solve every task
  - its meta knowledge
- Why is Machine Learning Easy
  - you have examples of learning from anything
  - its the one thing that humans are good at

How much of the math should I know

$$\beta \left(\alpha \mathbf{I} + \beta \phi(\mathbf{X})^{\mathrm{T}} \phi(\mathbf{X})\right)^{-1} \phi(\mathbf{X})^{\mathrm{T}} \mathbf{t}$$

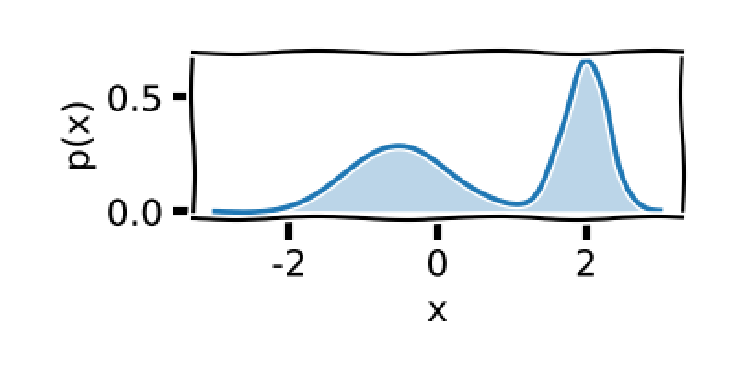
Laplace Demon [2]

Laplace's Demon [2]
An intelligence which at a given instant knew all the forces acting in nature and the position of every object in the universe - if endowed with a brain sufficiently vast to make all necessary calculations - could describe with a single formula the motions of the largest astronomical bodies and those of the smallest atoms. To such an intelligence, nothing would be uncertain; the future, like the past, would be an open book.

Laplace Demon [2]

All these efforts in the search for truth tend to lead the mind continously towards the intelligence we have just mentioned, although it will always remain infinetly distant from this intelligence.

Uncertainty



Variables

```python
def f(x):
    if x == 1:
        return 2
    else:
        return 1

x = 1
print(f(x))
```

Definition (Variable)

In elementary mathematics, a variable is an alphabetic character representing a number, called the value of the variable, which is either arbitrary, not fully specified, or unknown.

Random Variables

```
import numpy as np

def f(x):
    if x > 3.0:
        return 2
    else:
        return 1

x = np.random.normal(10.0,2.0,1)
```

Definition (Random Variable)

In probability and statistics, a random variable, random quantity, aleatory variable, or stochastic variable is a variable whose possible values are numerical outcomes of a random phenomenon.[2]

[2] https://en.wikipedia.org/wiki/Random_variable

Random Variables

```python
import numpy as np

def f(x):
    if x > 3.0:
        return 2
    else:
        return 1

x = np.random.normal(10.0,2.0,1)
```

$$p(x) = \mathcal{N}(x|10.0, 2.0)$$

[2] https://en.wikipedia.org/wiki/Random_variable

Random Variables

```python
import numpy as np

def f(x):
    return np.random.normal(x,1.0,1)

x = np.random.normal(10.0,2.0,1)
```

- x is random

$$p(x) = \mathcal{N}(x|10.0, 2.0)$$

- f is random

$$p(f|x) = \mathcal{N}(f|x, 1.0)$$

Conditional Distributions

$$p(\mathbf{w}|m_0, S_0) = \mathcal{N}(\mathbf{w}|m_0, S_0) = \frac{1}{((2\pi)^D|S_0|)^{\frac{1}{2}}} e^{\frac{1}{2}(\mathbf{w}-\mathbf{m_0})^{\mathrm{T}} S^{-1}(\mathbf{w}-\mathbf{m_0})}$$

- The above is a function of $\mathbf{w}$
- The function has "parameters" m_0 and S_0
- In order to evaluate the function the parameters needs to be set

Interpreting Bayesian Probabilities[3]

"Our prior is our assumption - when we say updated assumption, is this correct to say this is our posterior? Except this updated assumption isn't really an assumption, because it is a function over y and we therefore can't use it as a prior in another equation."

[3] reddit URL

Interpreting Bayesian Probabilities[3]

"When we state m_n and s_n as the parameters of our posterior, does this mean 'after we've multiplied the prior with n amounts of likelihoods from n data points'?"

Interpreting Bayesian Probabilities[3]

"Given the marginal likelihood can be interpreted as the distribution/probability of observing our data given our model, does this mean that the predicted posterior is simply our marginal likelihood? Or is the predicted posterior simply an entirely different model (using our previous results from the model learning w)?"

Dual Formulation[4]

"Do we need to really know the derivation for this? And when it says that in this form the prediction is now made using the training set.. does it use both the x and t values from the training set or just the target value 't'?"

[4] reddit URL

Conditionals[5]

A few things bug me:

- This posterior is valid - $p(w|x, y) \propto p(y|x, w)p(w)$
- Why is it not: $p(w|x, y) \propto p(y|x, w)p(x, w)$
- Or even $p(w|x, y) \propto p(y, x|w)p(w)$

[5] reddit URL

Marginal Distribution[5]

"The marginal distribution is written as: $p(y|x)$ *Why would it not be* $p(x, y)$ *or even* $p(y)$*"*

Prediction[6]

$$p(t_*|\mathbf{t}, \mathbf{x}_*, \mathbf{X}, \alpha, \beta) = \int p(t_*|\mathbf{x}_*, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \alpha, \beta) \mathrm{d}\mathbf{w}$$

- Likelihood

$$p(t_*|\mathbf{x}_*, \mathbf{w}, \beta) = \mathcal{N}\left(t_*|\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_*), \beta^{-1}\right) = f_1(t_*)$$

- Posterior

$$\begin{aligned} p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\beta\left(\alpha\mathbf{I} + \beta\phi(\mathbf{X})^{\mathrm{T}}\phi(\mathbf{X})\right)^{-1}\phi(\mathbf{X})^{\mathrm{T}}\mathbf{t}, \\ = \left(\alpha\mathbf{I} + \beta\phi(\mathbf{X})^{\mathrm{T}}\phi(\mathbf{X})\right)^{-1}\right) = f_2(\mathbf{w}) \end{aligned}$$

[6] reddit URL

Prediction[6]

- Likelihood

$$p(t_*|\mathbf{x}_*, \mathbf{w}, \beta) = \mathcal{N}\left(t_*|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_*), \beta^{-1}\right) = f_1(t_*)$$

- Posterior

$$\begin{aligned} p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\beta\left(\alpha\mathbf{I} + \beta\phi(\mathbf{X})^{\mathrm{T}}\phi(\mathbf{X})\right)^{-1}\phi(\mathbf{X})^{\mathrm{T}}\mathbf{t}, \\ = \left(\alpha\mathbf{I} + \beta\phi(\mathbf{X})^{\mathrm{T}}\phi(\mathbf{X})\right)^{-1}) = f_2(\mathbf{w}) \end{aligned}$$

$$\begin{aligned} p(t_*|\mathbf{t}, \mathbf{x}_*, \mathbf{X}, \alpha, \beta) &= \int p(t_*|\mathbf{x}_*, \mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \alpha, \beta)\mathrm{d}\mathbf{w} \\ &= \int f_1(t_*)f_2(\mathbf{w})\mathrm{d}\mathbf{w} = \int g(t_*, \mathbf{w})\mathrm{d}\mathbf{w} = h(t_*) \end{aligned}$$

[6] reddit URL

I don't think so

$$p(\mathrm{ML}|\mathrm{COMS30007}) = \frac{p(\mathrm{COMS30007}|\mathrm{ML})p(\mathrm{ML})}{p(\mathrm{COMS30007})}$$

References

Christopher M. Bishop.
***Pattern Recognition and Machine Learning (Information Science and Statistics)*.**
Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Pierre Simon Laplace.
A philosophical essay on probabilities, 1814.