Invariances in Gaussian processes

And how to learn them

ST John
PROWLER.io

Outline

1. What are invariances?
2. Why do we want to make use of them?
3. How can we construct invariant GPs?
4. Where invariant GPs are actually crucial
5. How can we figure out what invariances to employ?

What are invariances?

Function $f(\cdot)$ does not change under some transformation $\mathbf{x} \to t(\mathbf{x})$
i.e. $f(\mathbf{x}) = f(t(\mathbf{x}))$ for $\forall \mathbf{x} \in \mathcal{X} \quad \forall t \in \mathcal{T}$

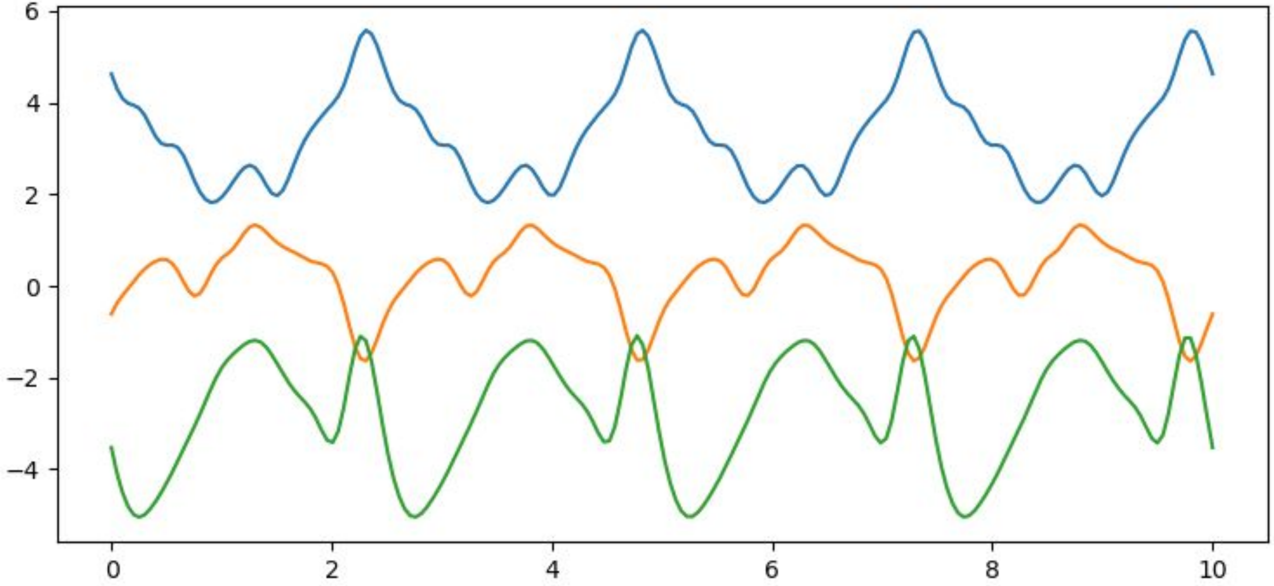
Can be **discrete** or **continuous**

- Translation
- Rotation
- Reflection
- Permutation

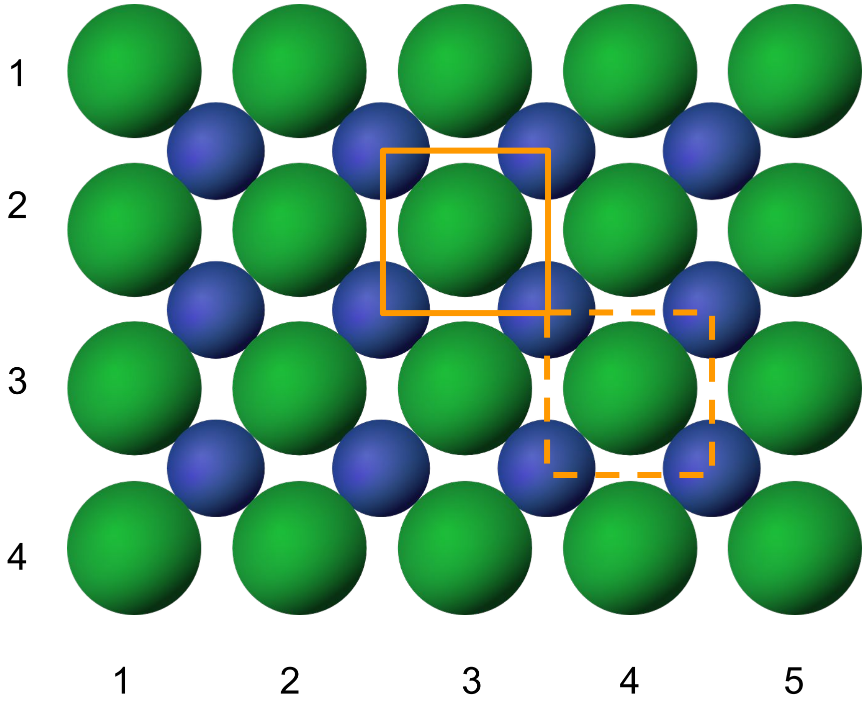
Invariance under discrete translation

$$\mathbf{x} \to \mathbf{x} + n\mathbf{d} \qquad n \in \mathbb{Z} \qquad f(\mathbf{x}) = f(\mathbf{x} + n\mathbf{d}) \qquad \forall \mathbf{x} \in \mathcal{X}$$

Periodic functions



Invariance under discrete translation



Density($\boxed{(2, 3)}$) = Density($\boxed{(3, 4)}$)

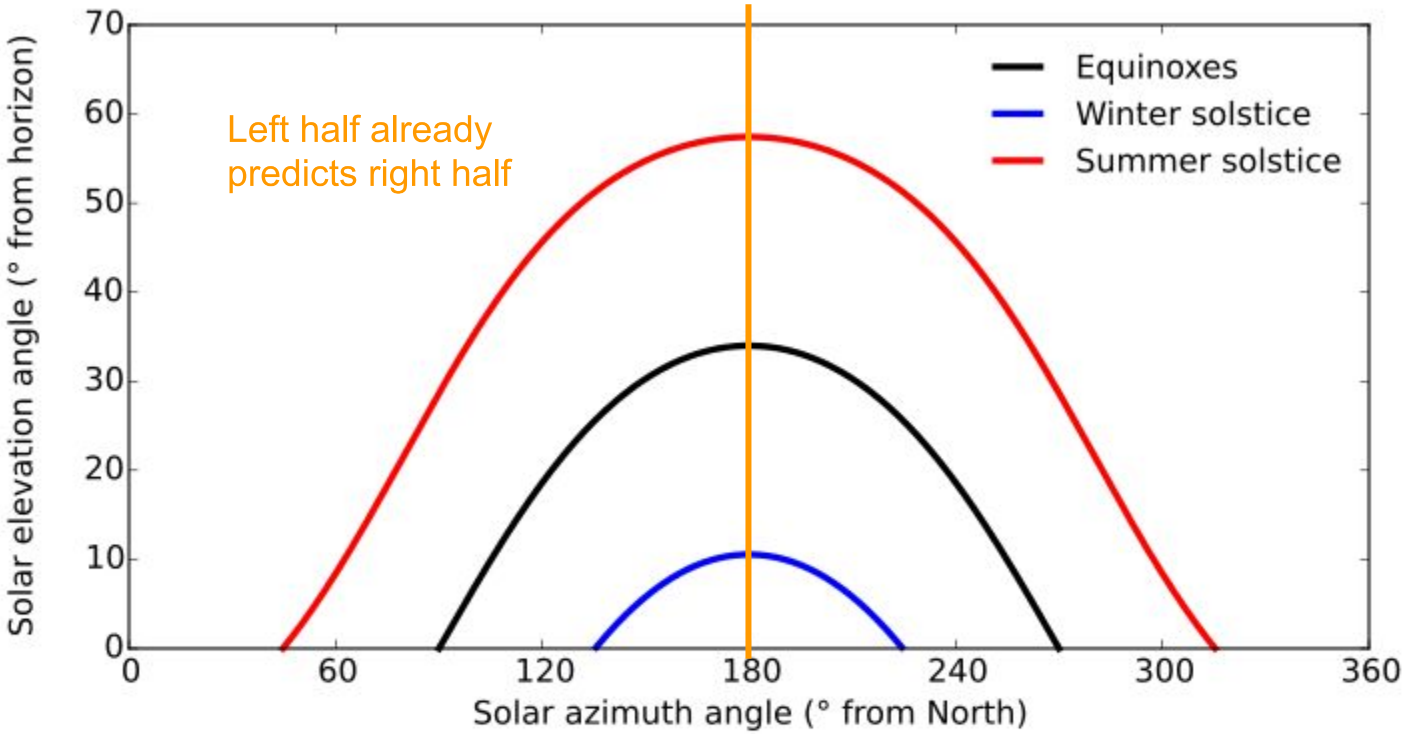
Invariance under discrete rotation

Density of water molecules
as a function of (x, y) point in plane

1/6th of the plane already predicts
the function value everywhere

Invariance under reflection

Solar elevation measured as function of azimuth (for different days)



Invariance under permutation

[100, 200, 1, 1, 1]

Invariance under permutation

[1, 200, 1, 100, 1]

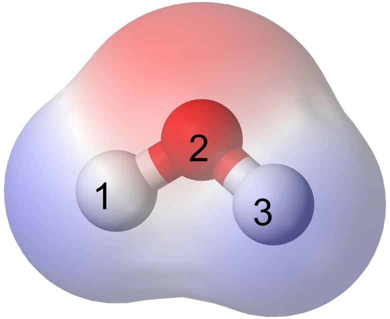
Invariance under permutation

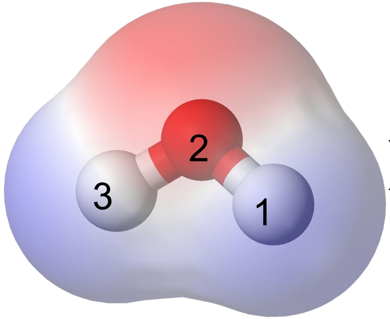
f() = *f*()

$$f(100, 200, 1, 1, 1) = f(1, 200, 1, 100, 1)$$

Different inputs but same function value

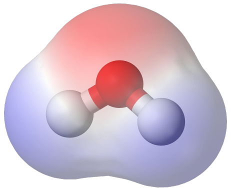
Invariance under permutation

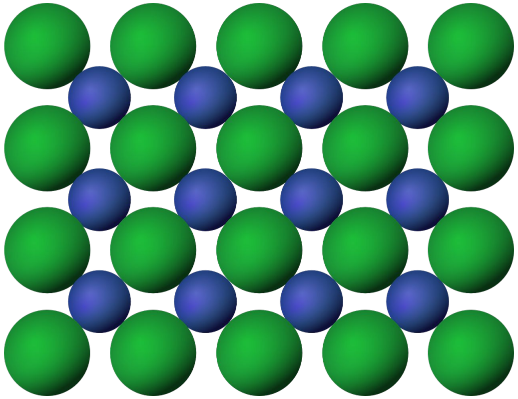
E(

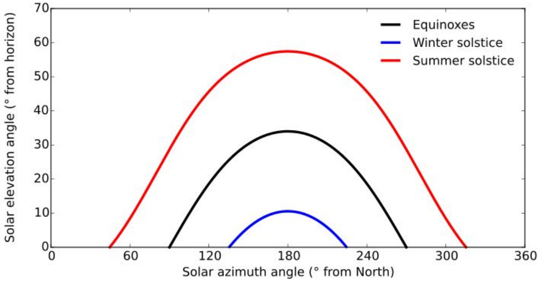
) = *E*(

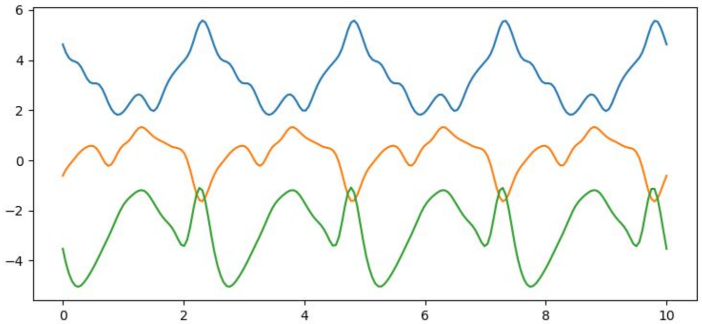
)

Discrete symmetries







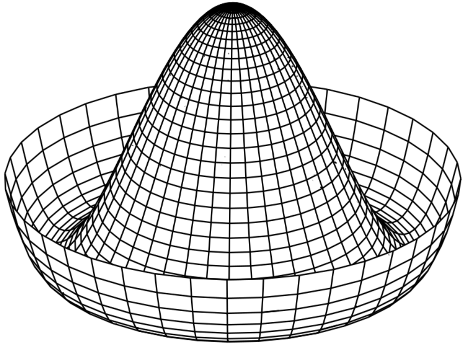


Invariance under continuous transformations

Translation

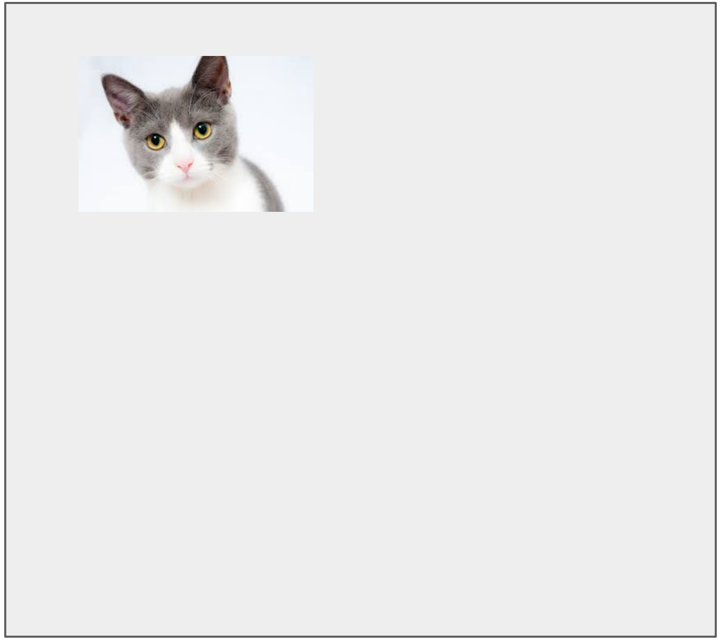
$\mathbf{x} \to \mathbf{x} + n\mathbf{d} \qquad n \in \mathbb{R}$

Rotation



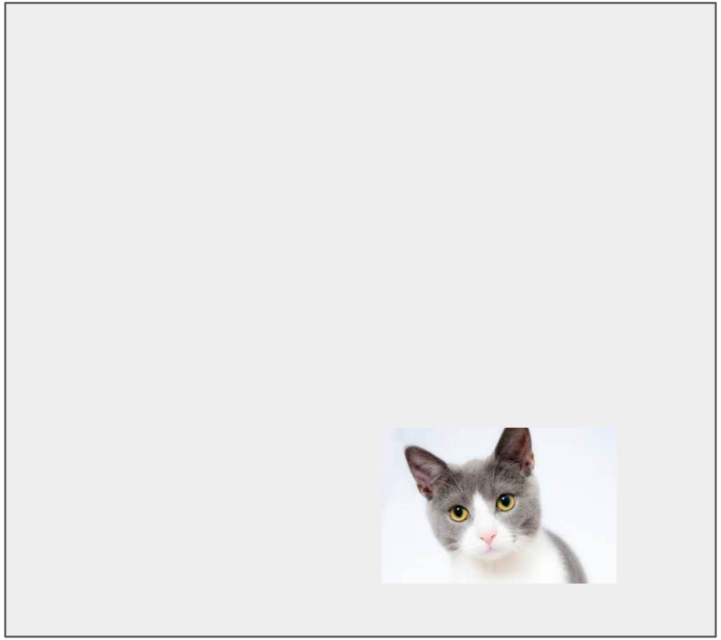
Example: image classification

Class label as a function of image pixel matrix

Label () = “cat”

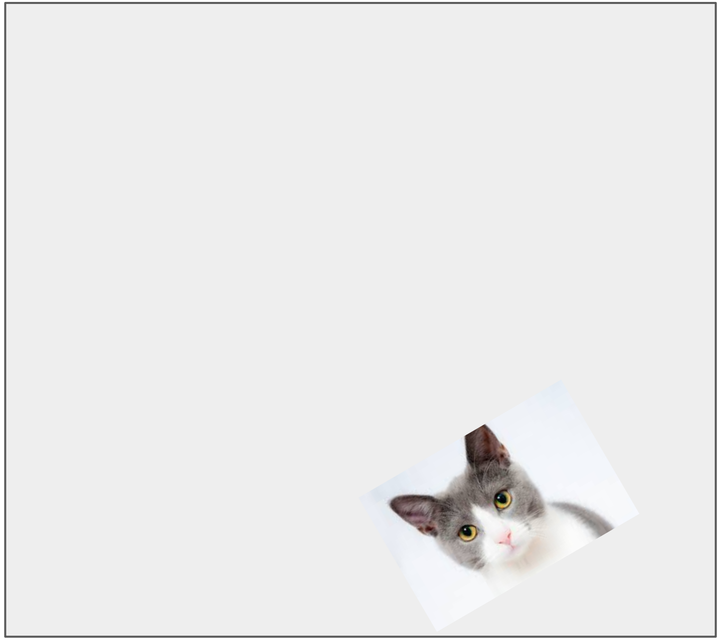
Example: image classification

Class label as a function of image pixel matrix

Label () = “cat”

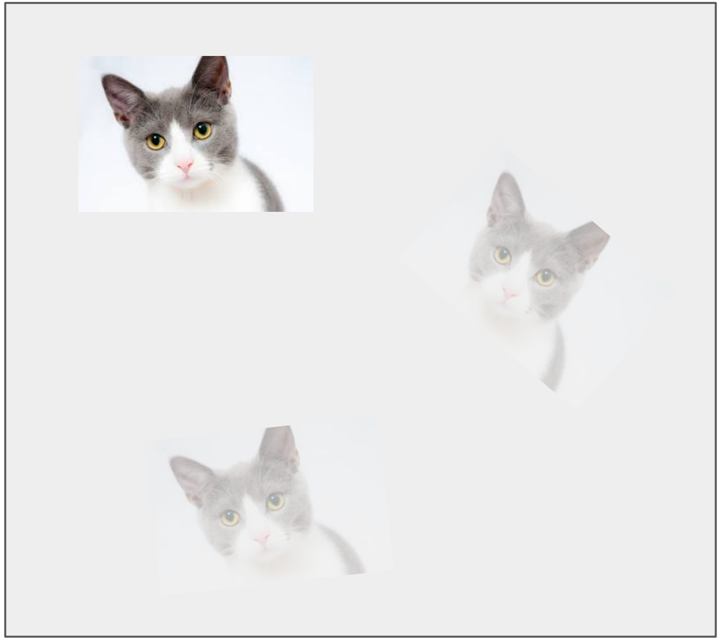
Example: image classification

Class label as a function of image pixel matrix

Label () = “cat”

Example: image classification

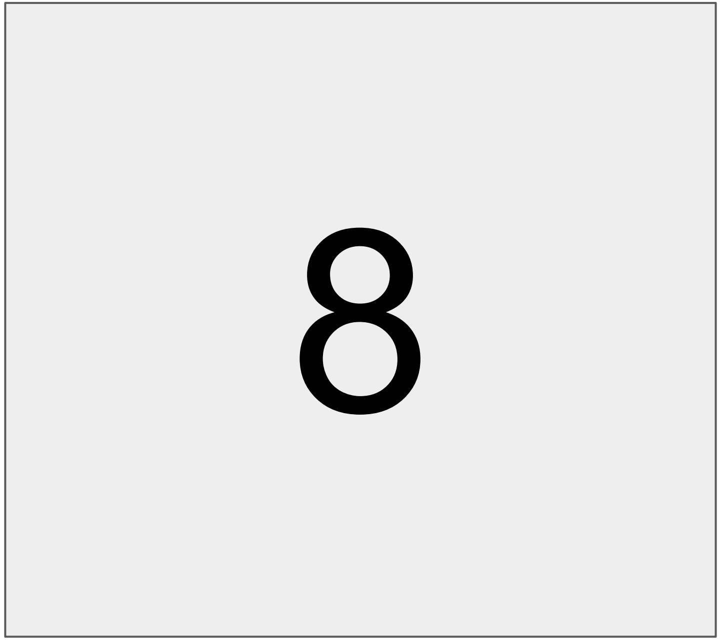
Class label as a function of image pixel matrix

Label () = “cat”

Example: image classification

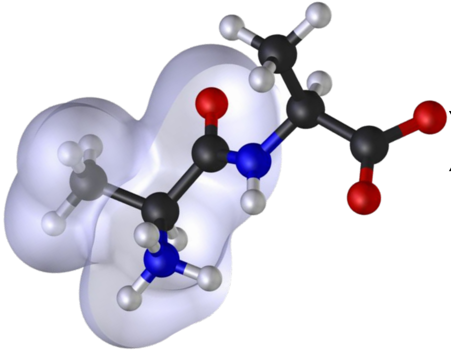
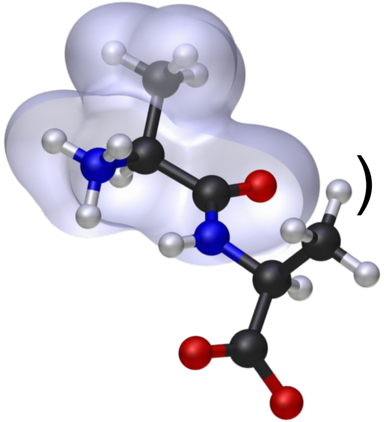
Class label as a function of image pixel matrix

Label (

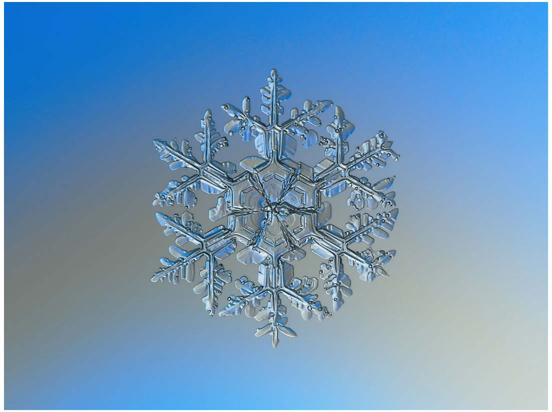


) = “8”

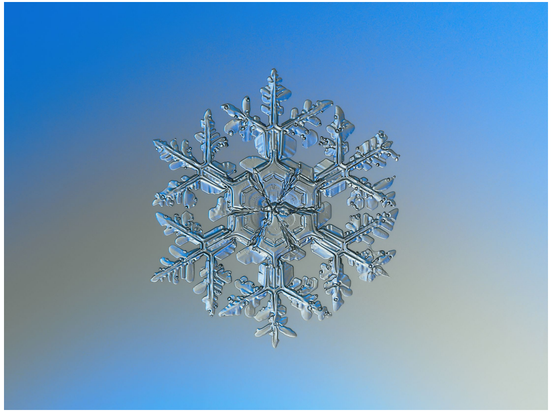
Example: molecular energy

E() = *E*()

Approximately invariant…



Approximately invariant…



6 9

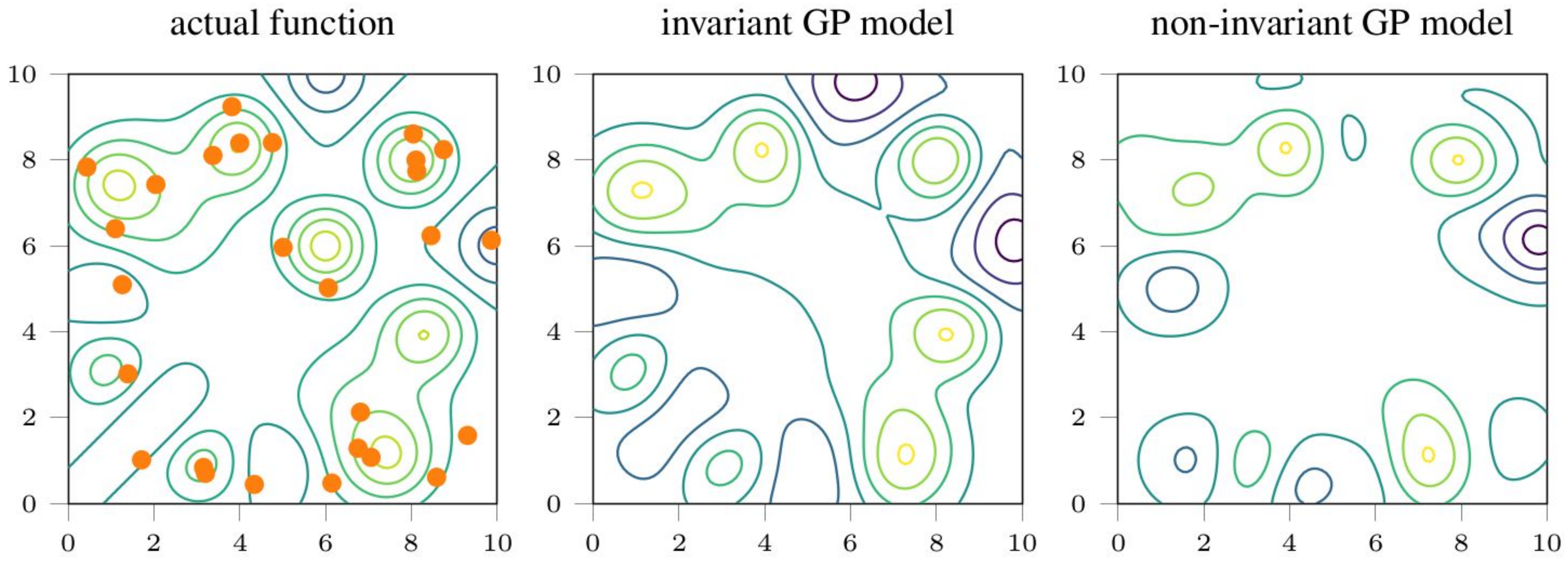
2. Why do we want to use invariances?

- Incorporate prior knowledge about the behaviour of a system
    - *Physical* symmetries, e.g. modelling total energy (and gradients, i.e. forces) of a set of atoms

- Helps generalisation

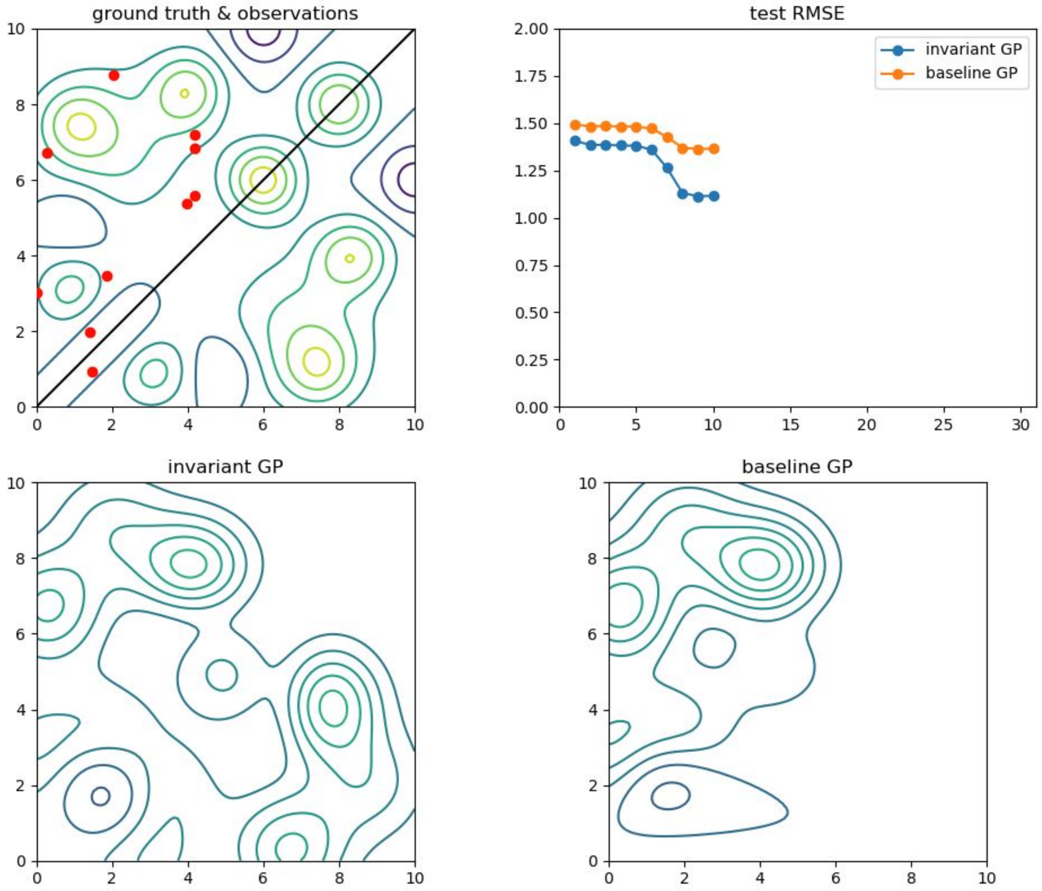
- Improved accuracy vs number of training points

Toy example



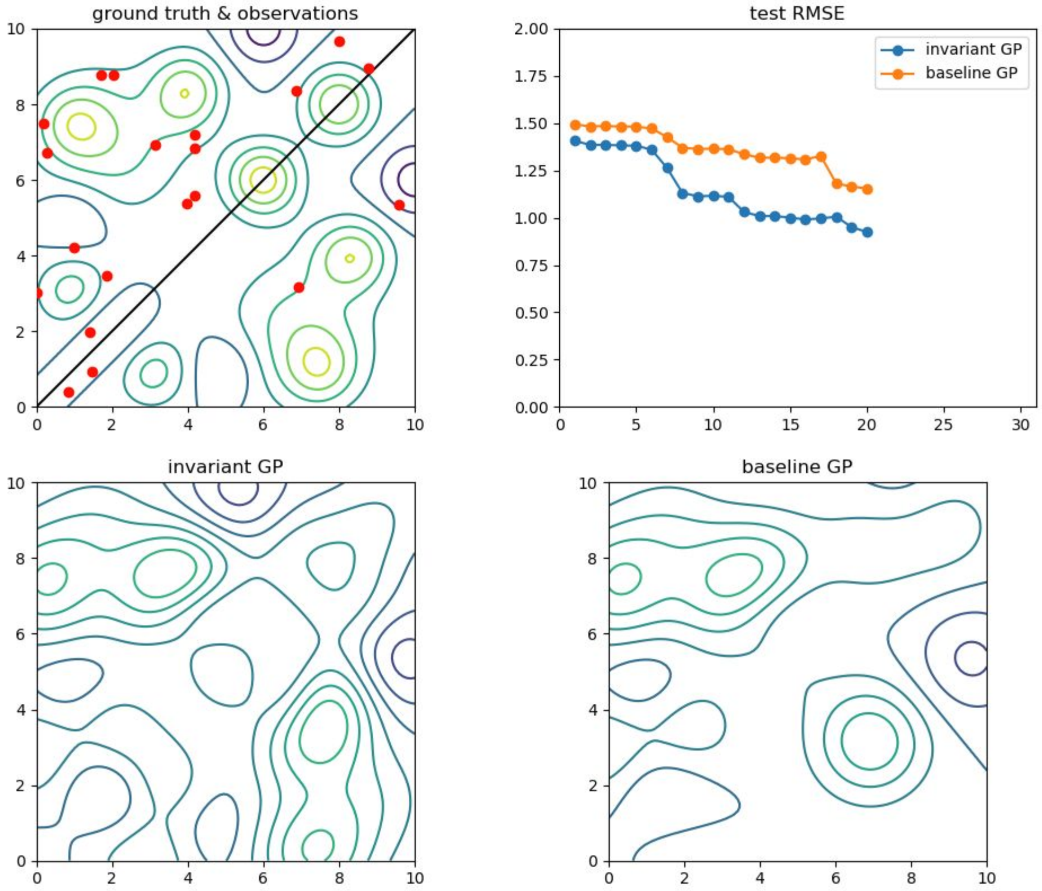
actual function
invariant GP model
non-invariant GP model
0
2
4
6
8
10

Toy example



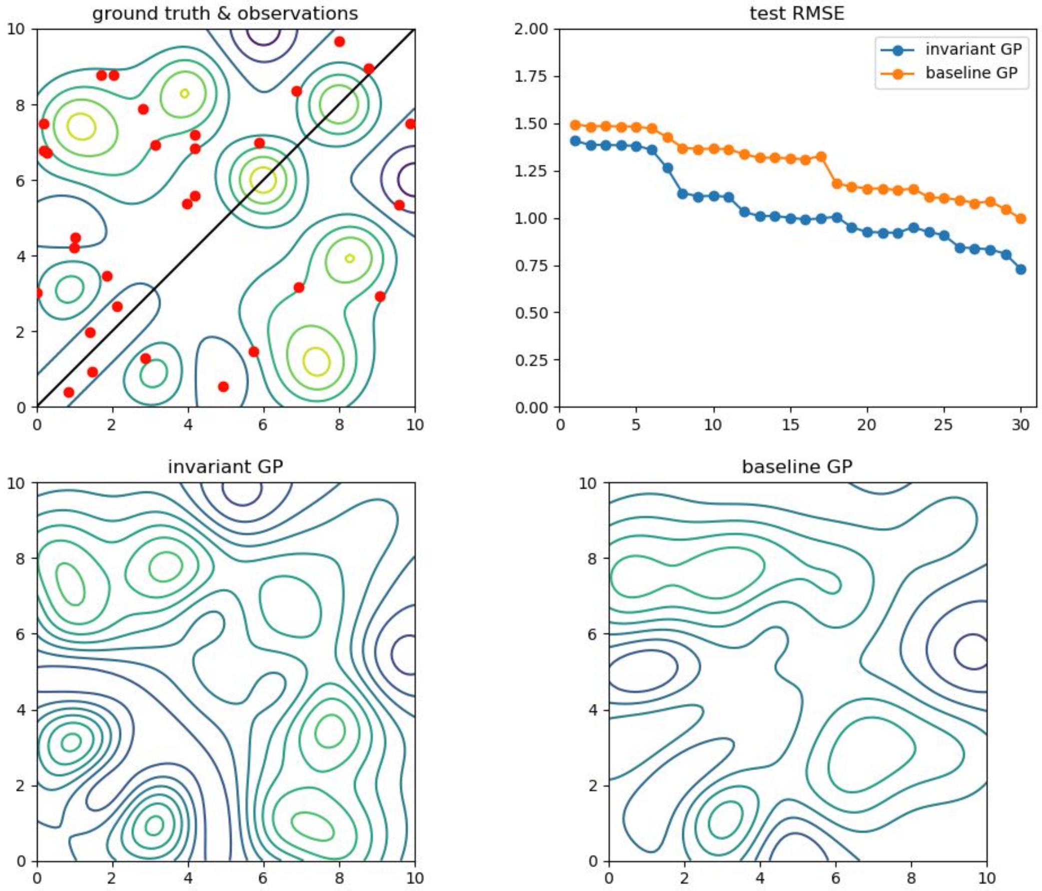
ground truth & observations
test RMSE
invariant GP
baseline GP
invariant GP
baseline GP

Toy example



ground truth & observations
test RMSE
invariant GP
baseline GP
invariant GP
baseline GP

Toy example


ground truth & observations
test RMSE
invariant GP
baseline GP
invariant GP
baseline GP

Constructing invariant GPs

We want a prior over functions that obey the chosen symmetry.

Symmetrise the function: Can do this by

a) appropriate **mapping** to invariant space

b) **sum** over transformations

Permutation-invariant GPs: mapping construction

$$\mathbf{x} = (x_1, x_2) \quad \rightarrow \quad \varphi(\mathbf{x}) = \begin{pmatrix} (x_1 + x_2)/2 \\ |x_1 - x_2|/2 \end{pmatrix}$$

$$f(x_1, x_2) = g((x_1 + x_2)/2, |x_1 - x_2|/2)$$

$$f(\mathbf{x}) = g(\varphi(\mathbf{x})) \quad \rightarrow \quad k_f(\mathbf{x}, \mathbf{x}') = k_g\big(\varphi(\mathbf{x}), \varphi(\mathbf{x}')\big)$$

Permutation-invariant GPs: sum construction

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

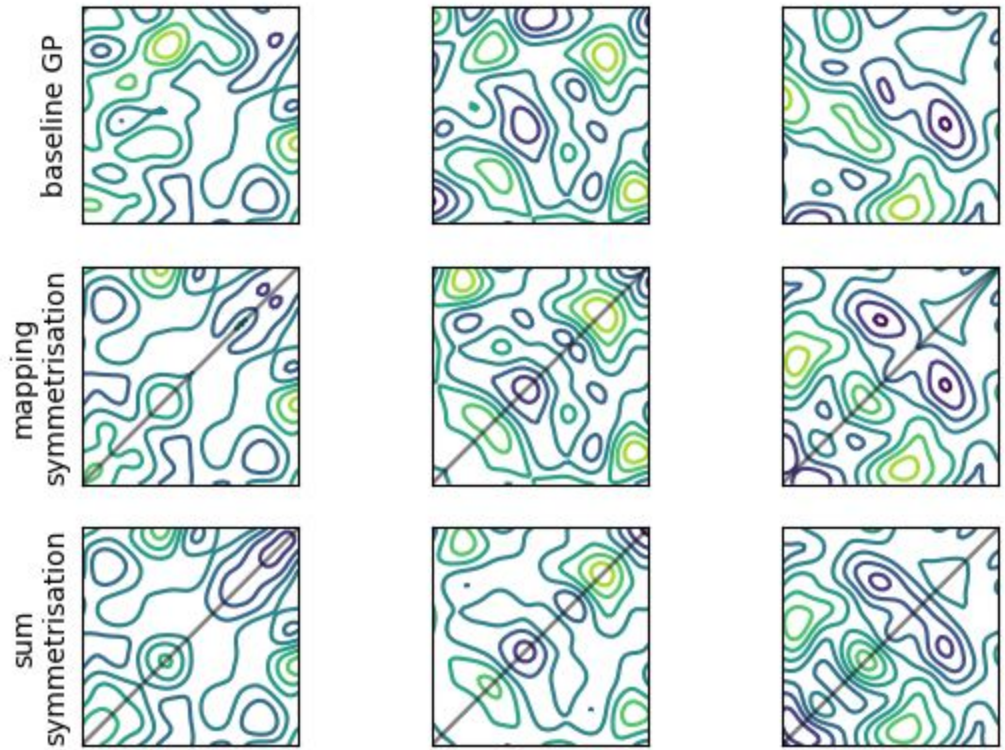
$(x_1, x_2) \to (x_2, x_1)$:

$$f(x_2, x_1) = g(x_2, x_1) + g(x_1, x_2)$$

Invariant sum kernel

$$
\begin{aligned}
\boxed{k_f\big((x_1, x_2), (x_1', x_2')\big)} &= \operatorname{Cov}\big(f(x_1, x_2), f(x_1', x_2')\big) = \mathbb{E}\big[f(x_1, x_2) f(x_1', x_2')\big] \\
&= \mathbb{E}\big[(g(x_1, x_2) + g(x_2, x_1))(g(x_1', x_2') + g(x_2', x_1'))\big] \\
&= \mathbb{E}\big[g(x_1, x_2) g(x_1', x_2')\big] + \mathbb{E}\big[g(x_1, x_2) g(x_2', x_1')\big] \\
&\quad + \mathbb{E}\big[g(x_2, x_1) g(x_1', x_2')\big] + \mathbb{E}\big[g(x_2, x_1) g(x_2', x_1')\big] \\
&\boxed{\begin{aligned} &= k_g\big((x_1, x_2), (x_1', x_2')\big) + k_g\big((x_1, x_2), (x_2', x_1')\big) \\ &\quad + k_g\big((x_2, x_1), (x_1', x_2')\big) + k_g\big((x_2, x_1), (x_2', x_1')\big) \end{aligned}}
\end{aligned}
$$

Samples from the prior



How can we generalise this?

Symmetry group

Transformations can be **composed**:

$$f(x) = f(t(x)) \quad \forall \mathbf{x} \in \mathcal{X} \qquad \forall t \in \mathcal{T}$$

$$f(t(x)) = f(t'(t(x))$$

$$\implies f(x) = f(t'(t(x)) = f((t' \circ t)(x))$$

Set of all compositions of transformations is a **group**; corresponds to symmetries

Orbit of *x*: all points reachable by transformations

$$\mathcal{O}(\mathbf{x}) = \{t(\mathbf{x})|t \in \mathcal{T}\}$$

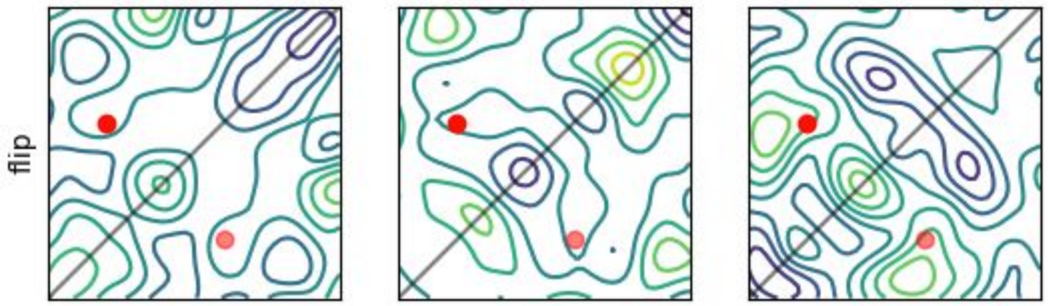
Example: Permutation in 2D

$$\pi(\mathbf{x}) : (x_1, x_2) \mapsto (x_2, x_1)$$
$$\mathcal{T} = \{I(\mathbf{x}), \pi(\mathbf{x})\}$$

$$\mathcal{O}((x_1, x_2)) = \mathcal{O}((x_2, x_1)) = \{(x_1, x_2), (x_2, x_1)\}$$

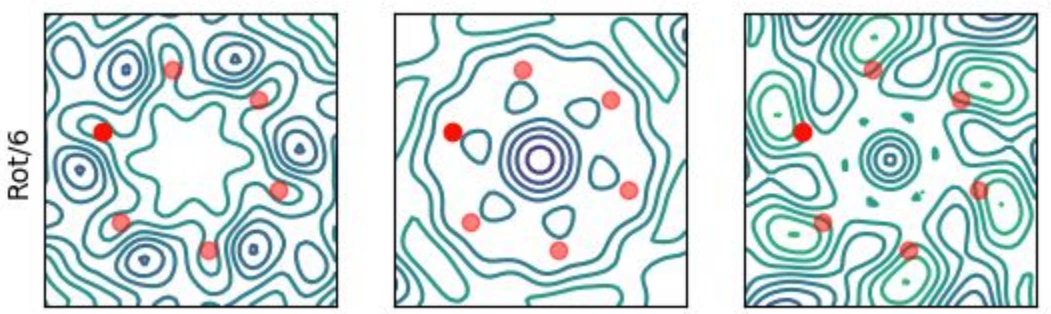
Examples of orbits: permutation invariance

Orbit size = 2



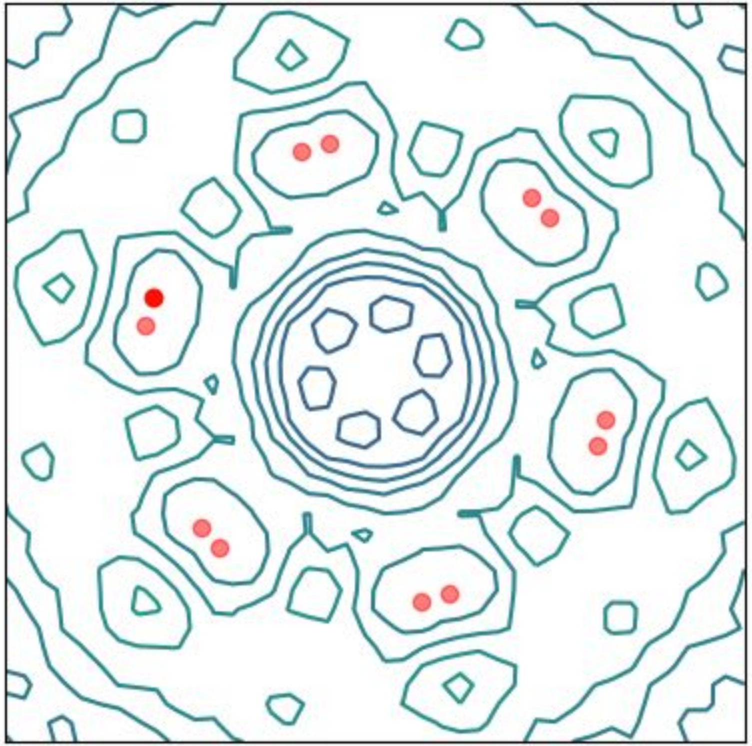
Examples of orbits: six-fold rotation invariance

Orbit size = 6



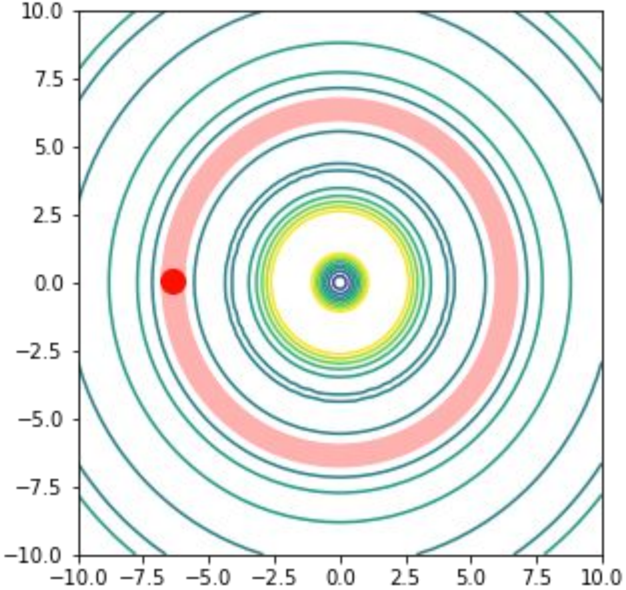
Examples of orbits: permutation **and** six-fold rotation

Orbit size = 12



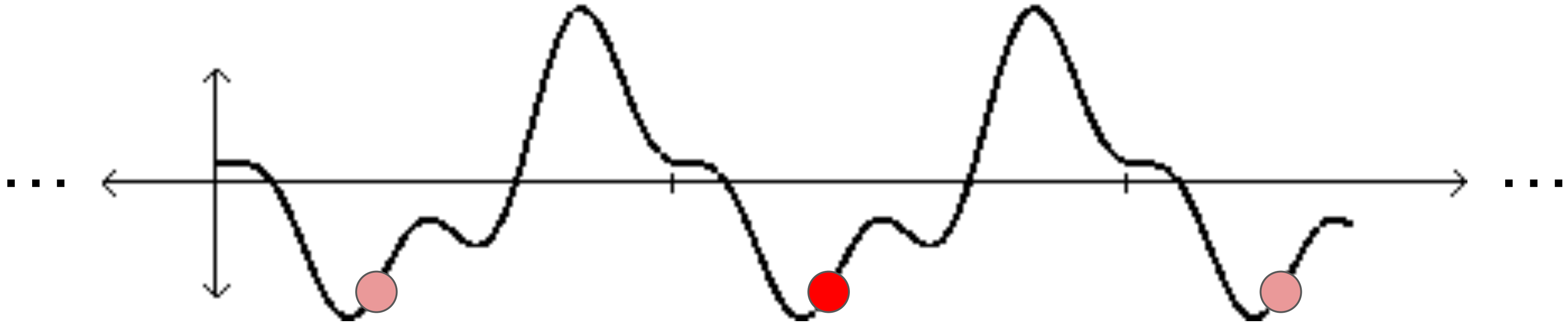
Examples of orbit: continuous rotation symmetry

Uncountably infinite



Orbit of a periodic function in 1D

Countably infinite



Constructing invariant GPs: sum revisited

$$\pi(\mathbf{x}) : (x_1, x_2) \mapsto (x_2, x_1)$$

$$\mathcal{T} = \{I(\mathbf{x}), \pi(\mathbf{x})\}$$

$$\mathcal{O}(\mathbf{x}) = \{t(\mathbf{x}) | t \in \mathcal{T}\}$$

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1) \implies f(\mathbf{x}) = \sum_{\mathbf{x}_o \in \mathcal{O}(\mathbf{x})} g(\mathbf{x}_o)$$

$$k_f(\mathbf{x}, \mathbf{x}') = \mathbb{E}\Big[\sum_{\mathbf{x}_o \in \mathcal{O}(\mathbf{x})} g(\mathbf{x}_o) \sum_{\mathbf{x}_{o'} \in \mathcal{O}(\mathbf{x}')} g(\mathbf{x}_{o'}) \Big] = \sum_{\mathbf{x}_o \in \mathcal{O}(\mathbf{x})} \sum_{\mathbf{x}_{o'} \in \mathcal{O}(\mathbf{x}')} k_g(\mathbf{x}_o, \mathbf{x}_{o'})$$

Applications

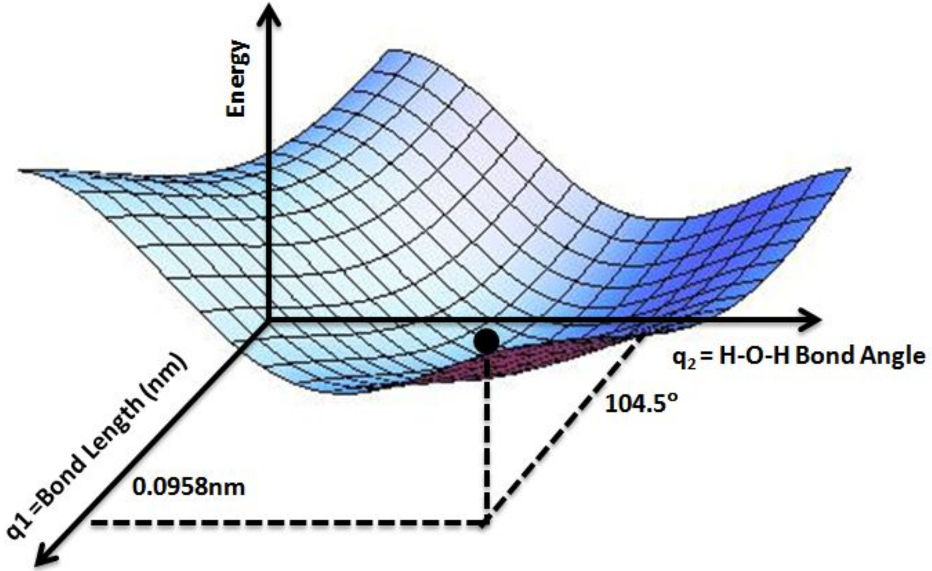
Molecular modelling

Time-evolution of the configuration (position of all atoms) of a system of atoms/molecules

Need Potential Energy Surface (PES)! Gradients = forces (easy with GPs)

$$E(\mathbf{r}_1, \ldots, \mathbf{r}_N)$$

Potential Energy Surface



Modelling Potential Energy Surface

Approximate as sum over k-mers (many-body expansion)

Invariance to rotation/translation of local environment/k-mer

Invariance under permutation of equivalent atoms

Modelling Potential Energy Surface

Many-body expansion, sum over k-mers:

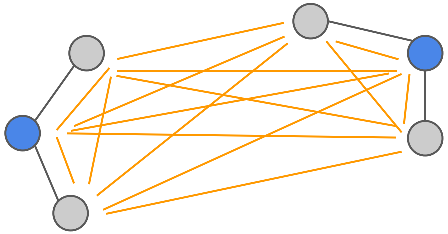
$$E(\mathbf{r}_1, \ldots, \mathbf{r}_N) = \sum_i E^{(1)}(\mathbf{r}_i) + \sum_{i \neq j} E^{(2)}(\mathbf{r}_i, \mathbf{r}_j) + \sum_{i \neq j \neq k} E^{(3)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \cdots$$

$$E(\mathbf{r}_1, \ldots, \mathbf{r}_N) = \sum_i E^{(1)}(\{\mathbf{r}_m : m \in \mathcal{M}_i\}) + \sum_{i \neq j} E^{(2)}(\{\mathbf{r}_m, \mathbf{r}_n : m \in \mathcal{M}_i, n \in \mathcal{M}_j\})$$
$$+ \sum_{i \neq j \neq k} E^{(3)}(\{\mathbf{r}_m, \mathbf{r}_n, \mathbf{r}_p : m \in \mathcal{M}_i, n \in \mathcal{M}_j, p \in \mathcal{M}_k\}) + \cdots$$

Modelling Potential Energy Surface

Invariance to rotation/translation of local environment/k-mer:

Map to interatomic distances



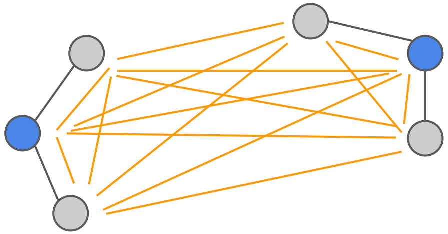
$$E^{(1)}(\{\mathbf{r}_m : m \in \mathcal{M}\}) = E^{(1)}(\{|\mathbf{r}_m - \mathbf{r}_n| : m, n \in \mathcal{M}\})$$

$$E^{(2)}(\{\mathbf{r}_m, \mathbf{r}_n : m \in \mathcal{M}_1, n \in \mathcal{M}_2\}) = E^{(2)}(\{|\mathbf{r}_m - \mathbf{r}_n| : m, n \in (\mathcal{M}_1 \cup \mathcal{M}_2)\})$$

Modelling Potential Energy Surface

Invariance under permutation of equivalent atoms:

sum over them!



How can we find out if an invariance is helpful?

- As usual (like another kernel hyperparameter): *marginal* likelihood

- Unlike “regular” likelihood (equivalent to training-set RMSE):
  - Less overfitting
  - Related to generalisation

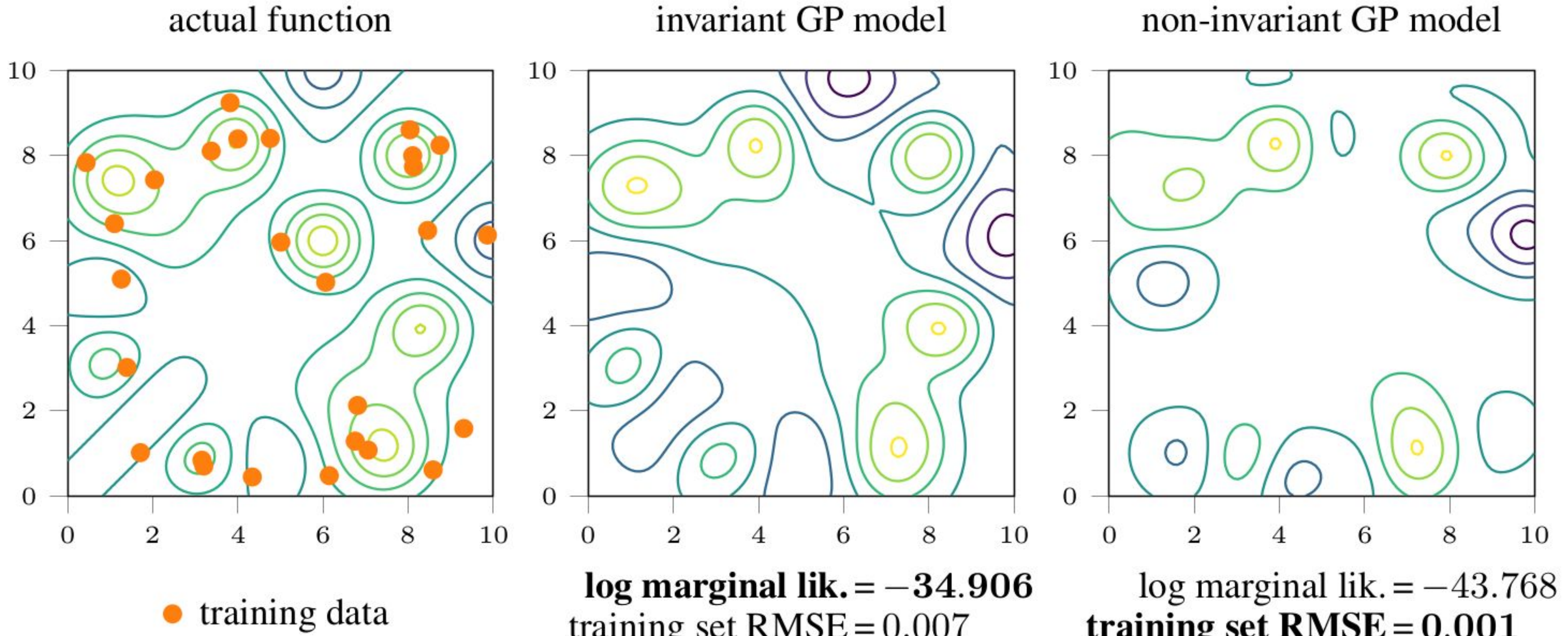
Marginal likelihood and generalisation

Measures how well part of the training set predicts the other training points:

$$p(\mathbf{y}|\theta) = p(\mathbf{y}_1|\theta)p(\mathbf{y}_2|\mathbf{y}_1, \theta)p(\mathbf{y}_3|\mathbf{y}_{1:2}, \theta) \prod_{c=4}^{C} p(\mathbf{y}_c|\mathbf{y}_{1:c-1}, \theta)$$

= how accurately the model generalises during inference,
similar to cross-validation (but differentiable)

Marginal likelihood



log marginal lik. = −34.906
training set RMSE = 0.007
test set RMSE = 0.43

log marginal lik. = −43.768
training set RMSE = 0.001
test set RMSE = 0.87

Summary: we have seen…

How to constrain GPs to give **invariant functions**

When invariance **improves** a model's **generalisation**

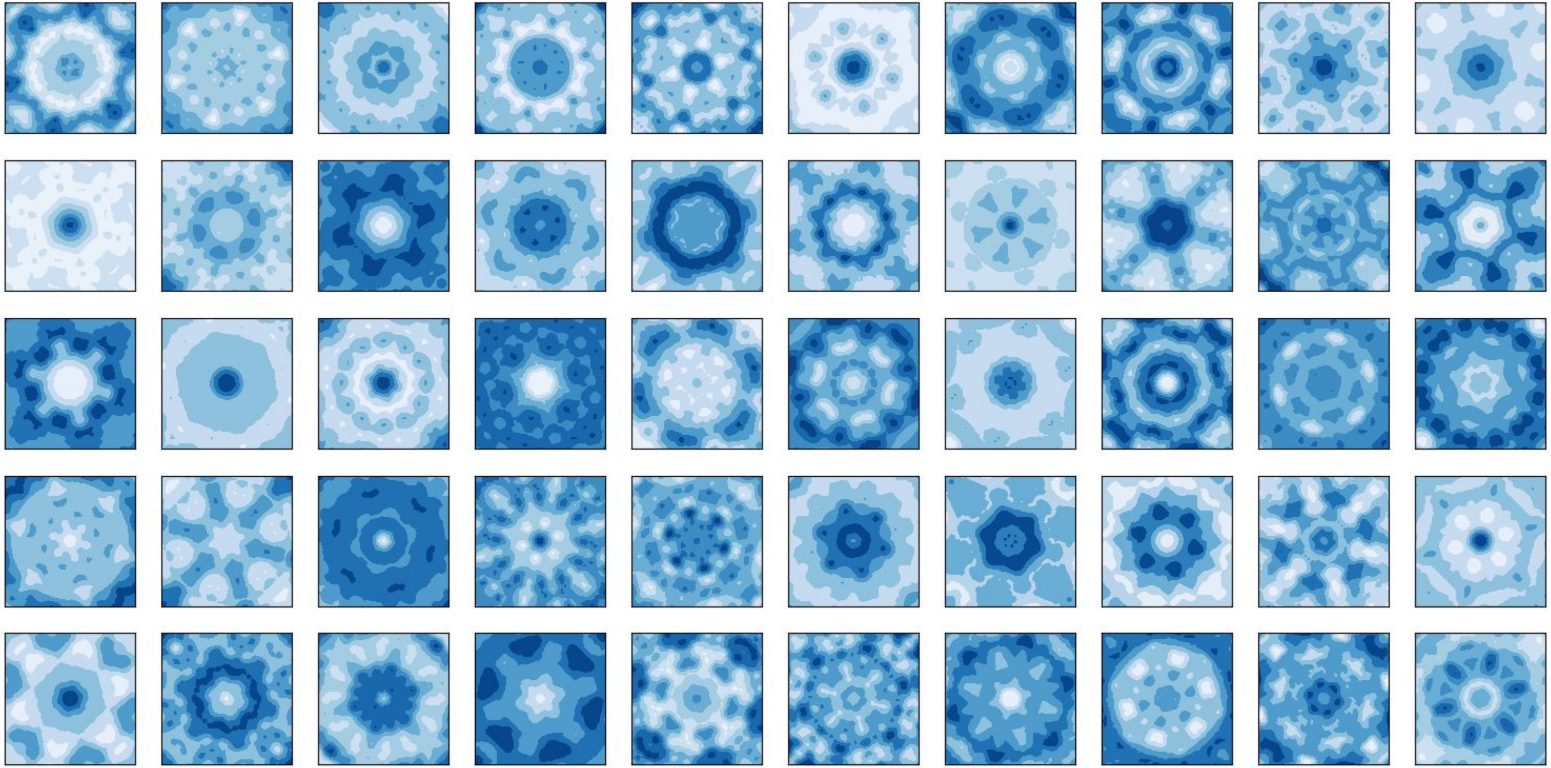
When invariance **increases the marginal likelihood**

That invariances exist in **real-world problems**

Questions?

Next up: how to **learn** invariances…

Snowflake prior



Why not just data augmentation?

Used in deep learning…

Invariances are better:

1. Cubic scaling with number of data points
   vs **linear scaling** with invariances in prior
2. Data augmentation results in same predictive mean, but not **variance**
3. Invariances in the GP prior give us **invariant samples**