

## Symmetry



1. **Inductive Bias**
2. Invariance
3. Group Theory



Inductive Bias, Symmetry, Invariance, Group theory

## Inductive Bias

- Learning machines differ in how they split up the space of inputs
- Perceptrons split the space using a hyperplane, RBFs split the space depending on some set of centres, MLPs successive split the world at different layers
- These differences are known as the **inductive bias** of the machine
- The different inductive bias lead to different performance
- How can we exploit the inductive bias to get good performance?

## The Real World

- The problems that we care about come from a highly structured world where the solutions inherit some of the properties of that world
- Usually similar input have similar outputs (we don't expect to change one pixel in an image and that it should be classified differently)
- The classes are often quite well separated

## Smoothness

- A natural assumption is smoothness (the classification surface or regression surface changes slowly as we change the input)
- Often there is a hyperparameter that controls this
  - ★ The kernel parameter  $\gamma$  in SVMs (or  $\ell$  in Gaussian Processes)
  - ★  $k$  in  $k$ -nearest neighbours
  - ★ The size of the  $L_2$  regularisation term in neural networks
- We often have to choose these using trial and error (using a validation set)

## Outline

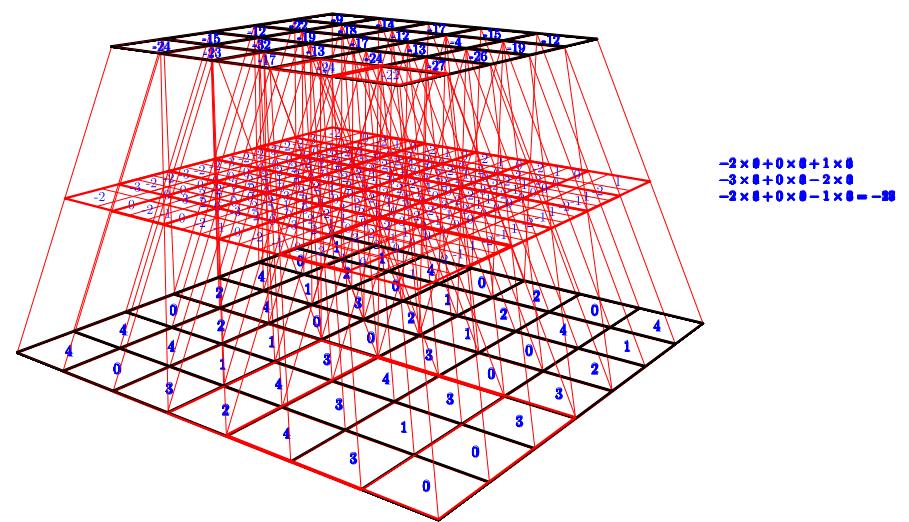
1. Inductive Bias
2. Invariance
3. Group Theory



## Symmetry

- There are sometimes symmetries in the world we can exploit
- In image classification we consider an object should be classified the same wherever it is in the image
- The classification should be invariant to translations (scaling and perhaps rotations)
- The classification function possesses a translational symmetry
- Learning machines that respect translational invariance have far less propensity to overfit—they don't need to learn that the same object in a different part of the image is the same

## Convolutional Neural Networks



## Equivariance

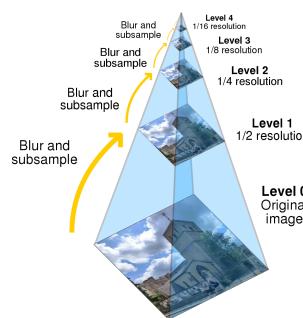
- CNNs are not translationally invariant
- The output will be different if we shift the input
- They are (up to edge effects) **equivariant** in that we would get the same result if we shifted both the input and the output in the same way
- They are not perfectly equivariant (they break at the pixel level and at the full image level—due to edges)
- But they are sufficient good that CNNs have dominated computer vision for years (ignoring visual transformers)

## Other Symmetries

- The identity of an object should not change if we move the camera further away or rotate the camera
- CNNs are not naturally scale or rotationally invariant (although there are attempts to remedy this)
- Rotation invariance doesn't always reflect "the world"—people usually stand on their feet not on their heads
- Although you want rotational invariance when interpreting aerial images

## Scale Symmetry

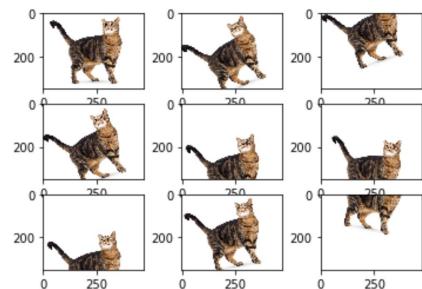
- CNNs are not scale invariant
- There are various pyramid image processing techniques to try and build in approximate scale invariance
- However, in many image classification tasks we concentrate on the object in the foreground
- Many datasets rescale objects in the image so they are roughly the same size



## Learning Invariance

- Currently we are not smart enough to come up with networks that are scale invariant or rotational invariant or invariant to small distortions
- We therefore hope that our networks learns this by training on objects of different sizes, orientations, lighting conditions, etc.
- Knowing this is an invariance we want, it is now common practice to do data augmentation where we perform transformations that replicate physical plausible changes in the data
- We are removing spurious rules by (artificially) increasing our dataset size

## Augmentations



- Data augmentation is standard practice in deep learning for images and makes a very considerable difference

## Normalisation

- Sometimes clever normalisation can make a lot of difference to the performance of an algorithm
- We might do this by feature engineering (divide the length on the nose by the width between the persons eyes)
- We might do this by normalising the output—use cosine similarity rather than distance

$$\cos(\theta) = \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

- These make the results invariant to the distance of the camera or the length of  $\mathbf{a}$  and  $\mathbf{b}$

## Other Symmetries Exist

- In NLP it is now common to translate sentences to a foreign language and back to the original language to learn invariance to minor changes in the words used
- There are cases where we care about sets of objects, but not their ordering
- An example might be sets of objects in an image or vertices in a graph
- Here constructing operators that are equivariant (the same if we permute the inputs and outputs) can again reduce overfitting

## Outline

1. Inductive Bias
2. Invariance
3. Group Theory



- Symmetries are very well studied in mathematics
- The language of symmetries are **groups**
- These capture the structure of transformations that represent some symmetry
- Group theory is highly abstract and it is easy to lose sight of its connection to symmetry, but it has proved itself extraordinarily powerful in revealing structures

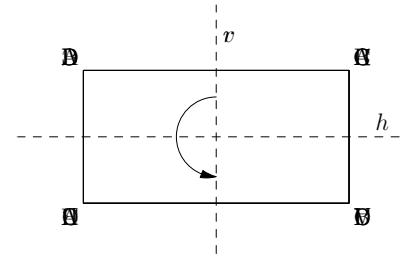
## Tossing a coin

- We consider a coin with two actions
  - ★ flip the coin:  $f$
  - ★ leave the coin alone:  $e$
- These form a group (the cyclic group  $C_2$ )
  - ★ Closure is obvious  $f \cdot e = f$ ,  $f \cdot f = e$ ,  $e \cdot e = e$  and  $e \cdot f = f$
  - ★ Associativity: e.g.  $(e \cdot f) \cdot f = e \cdot (f \cdot f)$
  - ★  $e$  is the identity
  - ★  $e^{-1} = e$  and  $f^{-1} = f$

- Groups consists of a set (we can think of the elements as transformations),  $\mathcal{G} = \{a, b, \dots\}$
- And an operator “.” (which we can often think of as composition)
- That is,  $a \cdot b$  we can interpret as apply transformation  $b$  and then apply transformation  $a$
- The set and transform form a group if they satisfy four axioms
  - ★ closure: for all  $a, b \in \mathcal{G}$  then  $a \cdot b \in \mathcal{G}$
  - ★ associativity:  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
  - ★ existence of an identity  $e$  such that  $a \cdot e = a$  for all  $a$
  - ★ Every element,  $a$ , has an inverse  $a^{-1}$  such that  $a^{-1} \cdot a = e$

## The Four Group

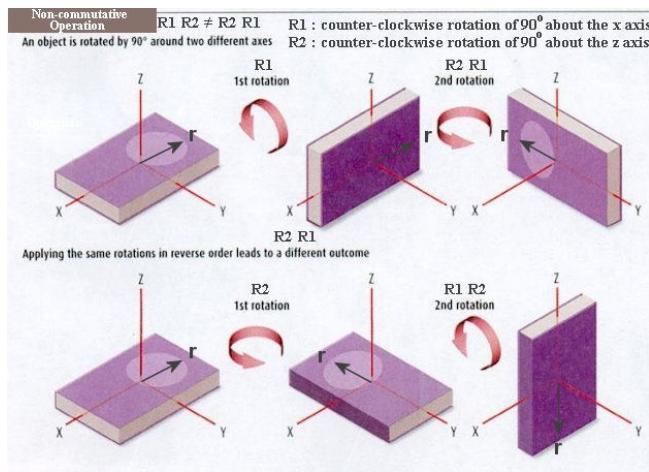
- A slightly less trivial example represents the set of transformations that leaves a rectangle invariant



	e	v	h	r
e	e	v	h	r
v	v	e	r	h
h	h	r	e	v
r	r	h	v	e

## Non-commutativity

- Not all groups commute
- For 3-d rotations :  $R_x \cdot R_z \neq R_z \cdot R_x$



## Permutations

- The set of permutations of  $n$  objects form a group (known as the symmetric group  $S_n$ )
- The composition of a permutation is a permutation
- There is an identity (do nothing)
- For every permutation there is an inverse permutation
- Obviously  $S_n$  has  $n!$  elements (you can reorder elements in  $n!$  ways)
- Permutations have a lot of structure (in terms of elements cycling, subgroups, etc.)

## Associativity

- We can think of the elements of a group as transformations acting on some object (e.g. rotations of a book)
- We consider  $\cdot$  to denote composition so  $a \cdot b$  denotes apply action  $b$  then action  $a$
- When we say  $c = a \cdot b$  we mean that action  $c$  is equivalent to apply action  $b$  followed by action  $a$
- Suppose  $g = a \cdot b$  and  $h = b \cdot c$  then

$$g \cdot c = a \cdot h$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

- These correspond to an equivalent set of actions

## Infinite Groups

- The set of integers under addition form a group
- Its closed, has an identity, 0, and each element  $n$  has an inverse  $-n$
- The group describes the set of discrete shifts
- The set of rational number under addition also forms a group
- Nothing new here, you've known about these since you were a toddler

- The set of reals under addition form a group and describes translations in one dimension
- The set of reals excluding 0 form a group under multiplication with an identity 1 and inverse of  $x$  being  $1/x$
- These describe scale transformation
- These are not terribly interesting groups (or at least groups you are so familiar with that they are now boring)
- Things get more interesting in high dimensional space

## Rotations $SO(n)$

- The set of  $n \times n$  orthogonal matrices (matrices,  $\mathbf{M}$ , such that  $\mathbf{M}^T \mathbf{M} = \mathbf{M} \mathbf{M}^T = \mathbf{I}$ ) form a group  $O(n)$
- Note that  $\det(\mathbf{M}) = \pm 1$
- These correspond to rotations and reflections
- The group of matrices that don't correspond to reflections (where  $\det(\mathbf{M}) = 1$ ) form a subgroup  $SO(n)$
- $SO(2)$  consist of the 2-d rotation matrices

$$\begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

- Continuous groups are known as Lie Groups
- They describe things like general translational invariance, rotational invariance, relativistic invariance
- The transformations can be represented by a set of matrices
- To study these one looks at infinitesimal transformations and the algebra between these infinitesimal transformations (Lie algebras)
- This is much studied in physics and occasionally studied in machine learning, although more on the sidelines

## Summary

- Invariances are important as if a machine properly captures a true invariance it can massively limit the space of functions being approximated—reducing the chances of finding spurious rules
- Often it is hard to build-in invariances into the machine learning model and practitioners will often resort to data augmentation in the hope that these invariances can be learned
- There is a mathematical language around describing the structure of symmetries: **group theory**
- You should know *group theory* exists, it is occasionally used in machine learning, but not that much
- I'm not going to expect you to know any details about group theory