Bayesian Deep Learning (MLSS 2019)

Yarin Gal

University of Oxford
yarin@cs.ox.ac.uk

Unless specified otherwise, photos are either original work or taken from Wikimedia, under Creative Commons license

Contents

Today:

- Introduction
- The Language of Uncertainty
- Bayesian Probabilistic Modelling
- Bayesian Probabilistic Modelling of Functions

Introduction

With great power...

- Many engineering advances in ML
- Systems applied to toy data
  → deployed in real-life settings
- Control **handed-over** to automated systems; w many scenarios which can become **life-threatening** to humans
  - Medical: automated decision making or recommendation systems
  - Automotive: autonomous control of drones and self driving cars
  - High frequency trading: ability to affect economic markets on global scale
  - But all of these can be quite dangerous...

With great power...

- Many engineering advances in ML
- Systems applied to toy data
  → deployed in real-life settings
- Control **handed-over** to automated systems; w many scenarios which can become **life-threatening** to humans
  - Medical: automated decision making or recommendation systems
  - Automotive: autonomous control of drones and self driving cars
  - High frequency trading: ability to affect economic markets on global scale
  - But all of these can be quite dangerous...

With great power...

- Many engineering advances in ML
- Systems applied to toy data
  → deployed in real-life settings
- Control **handed-over** to automated systems; w many scenarios which can become **life-threatening** to humans
  - Medical: automated decision making or recommendation systems
  - Automotive: autonomous control of drones and self driving cars
  - High frequency trading: ability to affect economic markets on global scale
  - But all of these can be quite dangerous...

With great power...

- Many engineering advances in ML
- Systems applied to toy data
  → deployed in real-life settings
- Control **handed-over** to automated systems; w many scenarios which can become **life-threatening** to humans
    - Medical: automated decision making or recommendation systems
    - Automotive: autonomous control of drones and self driving cars
    - High frequency trading: ability to affect economic markets on global scale
    - But all of these can be quite dangerous...

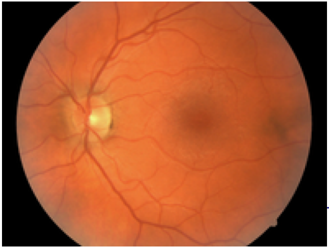
With great power...

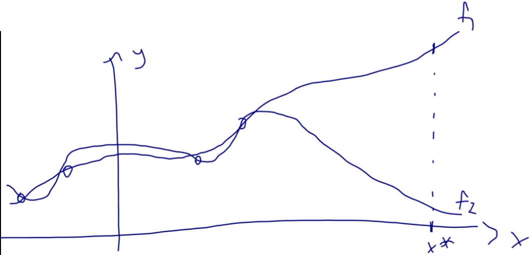
- Many engineering advances in ML
- Systems applied to toy data
  → deployed in real-life settings
- Control **handed-over** to automated systems; w many scenarios which can become **life-threatening** to humans
  - Medical: automated decision making or recommendation systems
  - Automotive: autonomous control of drones and self driving cars
  - High frequency trading: ability to affect economic markets on global scale
  - But all of these can be quite dangerous...

Example: Medical Diagnostics

- *dAIbetes*: an exciting new startup (not really)
  - claims to automatically diagnose diabetic retinopathy
  - accuracy 99% on their 4 train/test patients
  - engineer trained **two** deep learning systems to predict probability y given input fondus image x.

The engineer runs their system on your fondus image x^* (RHS):



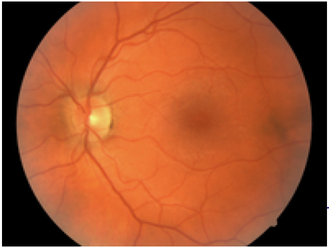


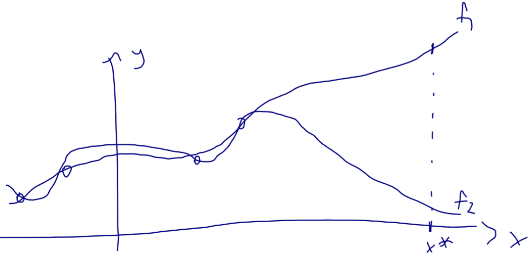
- Which model f_1, f_2 would you want the engineer to use for your diagnosis?

Example: Medical Diagnostics

- *dAIbetes*: an exciting new startup (not really)
  - claims to automatically diagnose diabetic retinopathy
  - accuracy 99% on their 4 train/test patients
  - engineer trained **two** deep learning systems to predict probability y given input fondus image x.

The engineer runs their system on your fondus image x^* (RHS):



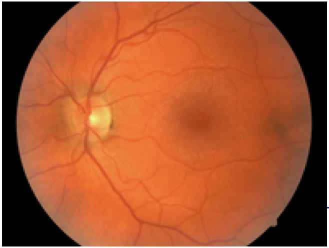


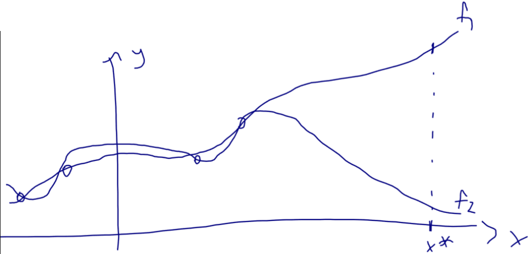
- Which model f_1, f_2 would you want the engineer to use for your diagnosis?

Example: Medical Diagnostics

- *dAIbetes*: an exciting new startup (not really)
  - claims to automatically diagnose diabetic retinopathy
  - accuracy 99% on their 4 train/test patients
  - engineer trained **two** deep learning systems to predict probability y given input fondus image x.

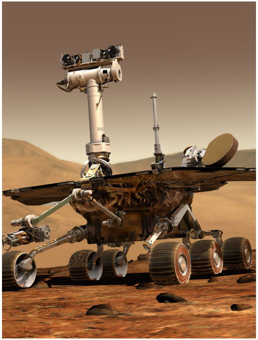
The engineer runs their system on your fondus image x^* (RHS):





- Which model f_1, f_2 would you want the engineer to use for your diagnosis? **None of these!** ('I don't know')

Example: Autonomous Driving



Autonomous systems

- Range from simple robotic **vacuums** to **self-driving cars**
- Largely divided into systems which
  - control behaviour w **rule-based** systems
  - learn and **adapt to environment**

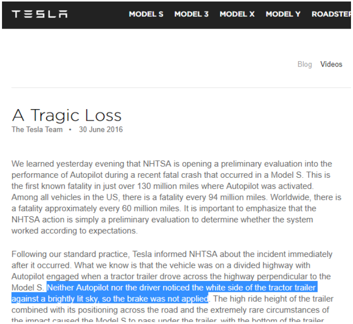
Both can use of ML tools

- ML for low-level feature extraction (**perception**)
- reinforcement learning

Example: Autonomous Driving (cnt)

Real-world example: assisted driving

- first fatality of assisted driving (June 2016)
- low-level system failed to distinguish white side of trailer from bright sky

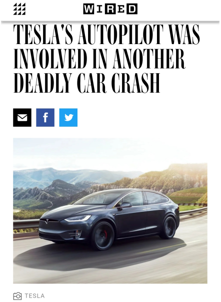


A Tragic Loss

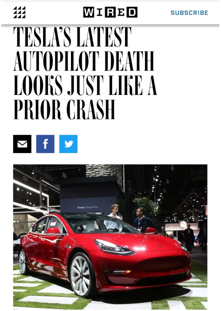
The Tesla Team • 30 June 2016

We learned yesterday evening that NHTSA is opening a preliminary evaluation into the performance of Autopilot during a recent fatal crash that occurred in a Model S. This is the first known fatality in just over 130 million miles where Autopilot was activated. Among all vehicles in the US, there is a fatality every 94 million miles. Worldwide, there is a fatality approximately every 60 million miles. It is important to emphasize that the NHTSA action is simply a preliminary evaluation to determine whether the system worked according to expectations.

Following our standard practice, Tesla informed NHTSA about the incident immediately after it occurred. What we know is that the vehicle was on a divided highway with Autopilot engaged when a tractor trailer drove across the highway perpendicular to the Model S. Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of the impact caused the Model S to pass under the trailer, with the bottom of the trailer



TESLA'S AUTOPILOT WAS INVOLVED IN ANOTHER DEADLY CAR CRASH

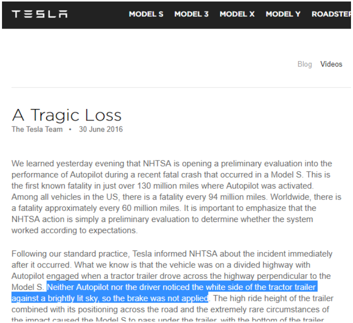


TESLA'S LATEST AUTOPILOT DEATH LOOKS JUST LIKE A PRIOR CRASH

Example: Autonomous Driving (cnt)

Real-world example: assisted driving

- first fatality of assisted driving (June 2016)
- low-level system failed to distinguish white side of trailer from bright sky

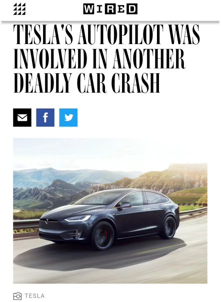


A Tragic Loss

The Tesla Team • 30 June 2016

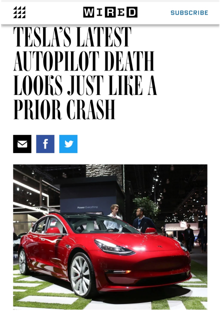
We learned yesterday evening that NHTSA is opening a preliminary evaluation into the performance of Autopilot during a recent fatal crash that occurred in a Model S. This is the first known fatality in just over 130 million miles where Autopilot was activated. Among all vehicles in the US, there is a fatality every 94 million miles. Worldwide, there is a fatality approximately every 60 million miles. It is important to emphasize that the NHTSA action is simply a preliminary evaluation to determine whether the system worked according to expectations.

Following our standard practice, Tesla informed NHTSA about the incident immediately after it occurred. What we know is that the vehicle was on a divided highway with Autopilot engaged when a tractor trailer drove across the highway perpendicular to the Model S. Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of the impact caused the Model S to pass under the trailer, with the bottom of the trailer



TESLA'S AUTOPILOT WAS INVOLVED IN ANOTHER DEADLY CAR CRASH

TESLA



TESLA'S LATEST AUTOPILOT DEATH LOOKS JUST LIKE A PRIOR CRASH

If system had identified its own uncertainty:

- alert user to take control over steering
- propagate uncertainty to decision making

Point estimates

In medical / robotics / science...

X **can't** use ML models giving a single point estimate (single value) in prediction

V **must** use ML models giving an answer that says '10 but I'm uncertain'; or '10 ± 5'

- Give me a **distribution** over possible outcomes!

Point estimates

In medical / robotics / science...

© Corbis

- X **can't** use ML models giving a single point estimate (single value) in prediction
- V **must** use ML models giving an answer that says '10 but I'm uncertain'; or '10 ± 5'
  - Give me a **distribution** over possible outcomes!

Point estimates

In medical / robotics / science...

© Corbis

X **can't** use ML models giving a single point estimate (single value) in prediction

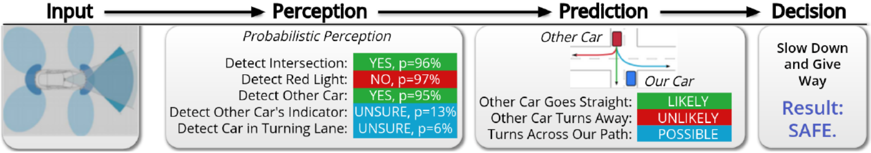
V **must** use ML models giving an answer that says '10 but I'm uncertain'; or '10 ± 5'

- Give me a **distribution** over possible outcomes!

Example: Autonomous Driving (cnt)

ML pipeline in **self-driving cars**

- **process raw sensory input** w perception models
  - eg image segmentation to find where other cars and pedestrians **are**
- output fed into prediction model
  - eg where other car will **go**
- output fed into 'higher-level' decision making procedures
  - eg **rule based system** ("cyclist to your left → do not steer left")
- industry's starting to use uncertainty for lots of components in the pipeline
  - eg **pedestrian prediction** models predict a distribution of pedestrian locations in X timesteps
  - or uncertainty in perception components



Input → Perception → Prediction → Decision

Probabilistic Perception

Detect Intersection:	YES, p=96%
Detect Red Light:	NO, p=97%
Detect Other Car:	YES, p=95%
Detect Other Car's Indicator:	UNSURE, p=13%
Detect Car in Turning Lane:	UNSURE, p=6%

Other Car Goes Straight:	LIKELY
Other Car Turns Away:	UNLIKELY
Turns Across Our Path:	POSSIBLE

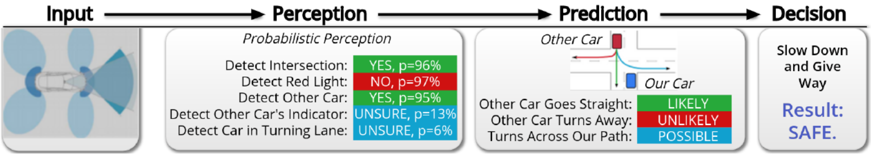
Slow Down and Give Way

Result: SAFE.

Example: Autonomous Driving (cnt)

ML pipeline in **self-driving cars**

- **process raw sensory input** w perception models
  - eg image segmentation to find where other cars and pedestrians **are**
- output fed into prediction model
  - eg where other car will **go**
- output fed into ‘higher-level’ decision making procedures
  - eg **rule based system** (“cyclist to your left → do not steer left”)
- industry’s starting to use uncertainty for lots of components in the pipeline
  - eg **pedestrian prediction** models predict a distribution of pedestrian locations in X timesteps
  - or uncertainty in perception components



Input → Perception → Prediction → Decision

Probabilistic Perception

Detect Intersection:	YES, p=96%
Detect Red Light:	NO, p=97%
Detect Other Car:	YES, p=95%
Detect Other Car's Indicator:	UNSURE, p=13%
Detect Car in Turning Lane:	UNSURE, p=6%

Other Car Goes Straight:	LIKELY
Other Car Turns Away:	UNLIKELY
Turns Across Our Path:	POSSIBLE

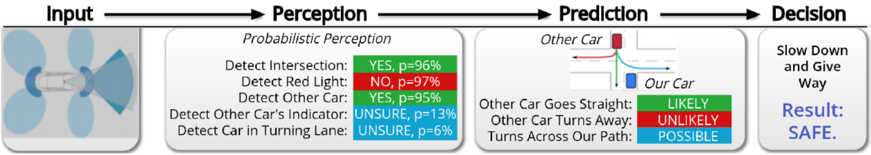
Slow Down and Give Way

Result: SAFE.

Example: Autonomous Driving (cnt)

ML pipeline in **self-driving cars**

- **process raw sensory input** w perception models
  - eg image segmentation to find where other cars and pedestrians **are**
- output fed into prediction model
  - eg where other car will **go**
- output fed into 'higher-level' decision making procedures
  - eg **rule based system** ("cyclist to your left → do not steer left")
- industry's starting to use uncertainty for lots of components in the pipeline
  - eg **pedestrian prediction** models predict a distribution of pedestrian locations in X timesteps
  - or uncertainty in perception components



Probabilistic Perception

Detect Intersection: YES, p=96%
Detect Red Light: NO, p=97%
Detect Other Car: YES, p=95%
Detect Other Car's Indicator: UNSURE, p=13%
Detect Car in Turning Lane: UNSURE, p=6%

Other Car Goes Straight: LIKELY
Other Car Turns Away: UNLIKELY
Turns Across Our Path: POSSIBLE

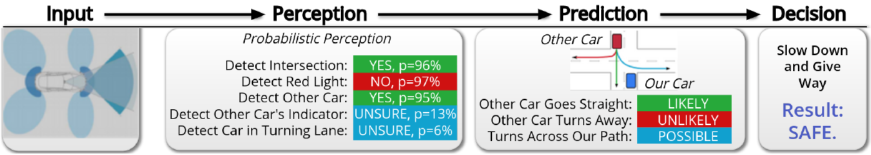
Slow Down and Give Way

Result: SAFE.

Example: Autonomous Driving (cnt)

ML pipeline in **self-driving cars**

- **process raw sensory input** w perception models
  - eg image segmentation to find where other cars and pedestrians **are**
- output fed into prediction model
  - eg where other car will **go**
- output fed into 'higher-level' decision making procedures
  - eg **rule based system** ("cyclist to your left → do not steer left")
- industry's starting to use uncertainty for lots of components in the pipeline
  - eg **pedestrian prediction** models predict a distribution of pedestrian locations in X timesteps
  - or uncertainty in perception components



Input → Perception → Prediction → Decision

Probabilistic Perception

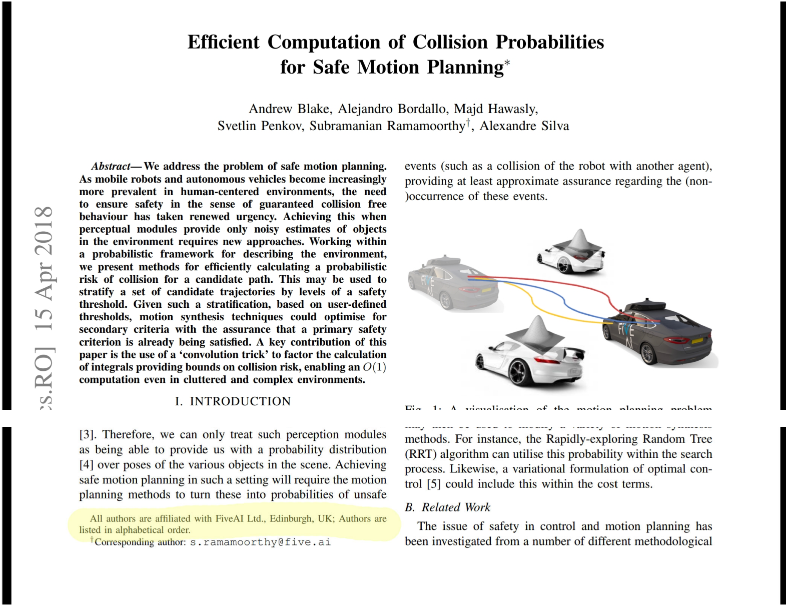
Detect Intersection:	YES, p=96%
Detect Red Light:	NO, p=97%
Detect Other Car:	YES, p=95%
Detect Other Car's Indicator:	UNSURE, p=13%
Detect Car in Turning Lane:	UNSURE, p=6%

Other Car Goes Straight:	LIKELY
Other Car Turns Away:	UNLIKELY
Turns Across Our Path:	POSSIBLE

Slow Down and Give Way

Result: SAFE.

Example: Autonomous Driving (cnt)



[cs.RO] 15 Apr 2018

Efficient Computation of Collision Probabilities for Safe Motion Planning*

Andrew Blake, Alejandro Bordallo, Majd Hawasly,
Svetlin Penkov, Subramanian Ramamoorthy†, Alexandre Silva

Abstract— **We address the problem of safe motion planning. As mobile robots and autonomous vehicles become increasingly more prevalent in human-centered environments, the need to ensure safety in the sense of guaranteed collision free behaviour has taken renewed urgency. Achieving this when perceptual modules provide only noisy estimates of objects in the environment requires new approaches. Working within a probabilistic framework for describing the environment, we present methods for efficiently calculating a probabilistic risk of collision for a candidate path. This may be used to stratify a set of candidate trajectories by levels of a safety threshold. Given such a stratification, based on user-defined thresholds, motion synthesis techniques could optimise for secondary criteria with the assurance that a primary safety criterion is already being satisfied. A key contribution of this paper is the use of a 'convolution trick' to factor the calculation of integrals providing bounds on collision risk, enabling an $O(1)$ computation even in cluttered and complex environments.**

I. INTRODUCTION

[3]. Therefore, we can only treat such perception modules as being able to provide us with a probability distribution [4] over poses of the various objects in the scene. Achieving safe motion planning in such a setting will require the motion planning methods to turn these into probabilities of unsafe events (such as a collision of the robot with another agent), providing at least approximate assurance regarding the (non-)occurrence of these events.

Fig. 1: A visualisation of the motion planning problem

may then be used to modify a variety of motion synthesis methods. For instance, the Rapidly-exploring Random Tree (RRT) algorithm can utilise this probability within the search process. Likewise, a variational formulation of optimal control [5] could include this within the cost terms.

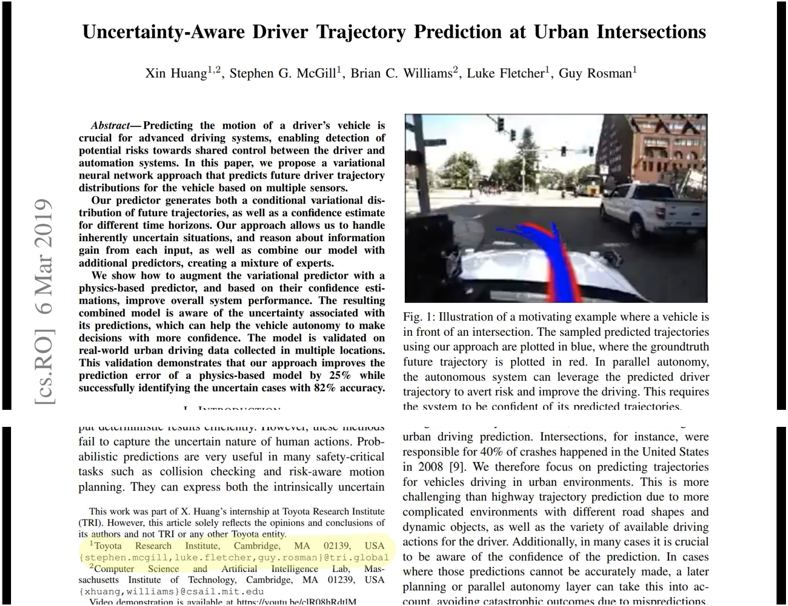
B. Related Work

The issue of safety in control and motion planning has been investigated from a number of different methodological

All authors are affiliated with FiveAI Ltd., Edinburgh, UK; Authors are listed in alphabetical order.

†Corresponding author: s.ramamoorthy@five.ai

Example: Autonomous Driving (cnt)



Uncertainty-Aware Driver Trajectory Prediction at Urban Intersections

Xin Huang[1,2], Stephen G. McGill[1], Brian C. Williams[2], Luke Fletcher[1], Guy Rosman[1]

[cs.RO] 6 Mar 2019

Abstract— Predicting the motion of a driver's vehicle is crucial for advanced driving systems, enabling detection of potential risks towards shared control between the driver and automation systems. In this paper, we propose a variational neural network approach that predicts future driver trajectory distributions for the vehicle based on multiple sensors.

Our predictor generates both a conditional variational distribution of future trajectories, as well as a confidence estimate for different time horizons. Our approach allows us to handle inherently uncertain situations, and reason about information gain from each input, as well as combine our model with additional predictors, creating a mixture of experts.

We show how to augment the variational predictor with a physics-based predictor, and based on their confidence estimations, improve overall system performance. The resulting combined model is aware of the uncertainty associated with its predictions, which can help the vehicle autonomy to make decisions with more confidence. The model is validated on real-world urban driving data collected in multiple locations. This validation demonstrates that our approach improves the prediction error of a physics-based model by 25% while successfully identifying the uncertain cases with 82% accuracy.

Fig. 1: Illustration of a motivating example where a vehicle is in front of an intersection. The sampled predicted trajectories using our approach are plotted in blue, where the groundtruth future trajectory is plotted in red. In parallel autonomy, the autonomous system can leverage the predicted driver trajectory to avert risk and improve the driving. This requires the system to be confident of its predicted trajectories.

I. Introduction

put deterministic results efficiently. However, these methods fail to capture the uncertain nature of human actions. Probabilistic predictions are very useful in many safety-critical tasks such as collision checking and risk-aware motion planning. They can express both the intrinsically uncertain

urban driving prediction. Intersections, for instance, were responsible for 40% of crashes happened in the United States in 2008 [9]. We therefore focus on predicting trajectories for vehicles driving in urban environments. This is more challenging than highway trajectory prediction due to more complicated environments with different road shapes and dynamic objects, as well as the variety of available driving actions for the driver. Additionally, in many cases it is crucial to be aware of the confidence of the prediction. In cases where those predictions cannot be accurately made, a later planning or parallel autonomy layer can take this into account, avoiding catastrophic outcomes due to mispredictions.

This work was part of X. Huang's internship at Toyota Research Institute (TRI). However, this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

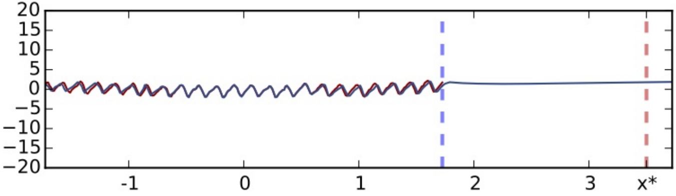
[1] Toyota Research Institute, Cambridge, MA 02139, USA {stephen.mcgill,luke.fletcher,guy.rosman}@tri.global

[2] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA 01239, USA {xhuang,williams}@csail.mit.edu

Video demonstration is available at https://youtu.be/clR08hRdtlM

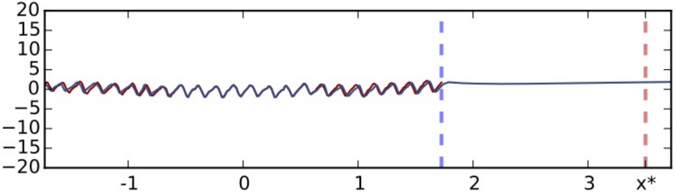
Sources of uncertainty

- Above are some *examples* of uncertainty
- Many other sources of uncertainty
- Test data is very dissimilar to training data
  - model trained on diabetes fondus photos of subpopulation A
  - never saw subpopulation B
    - "images are outside data distribution model was trained on"
  - desired behaviour
    - return a prediction (attempting to extrapolate)
    - +information that image lies outside data distribution
  - (model retrained w subpop. B labels → low uncertainty on these)



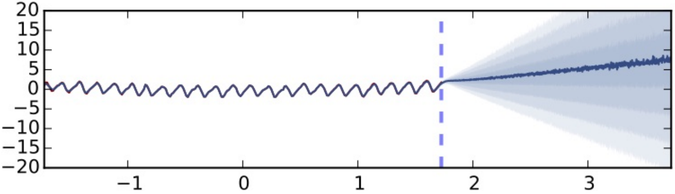
Sources of uncertainty

- Above are some *examples* of uncertainty
- Many other sources of uncertainty
- Test data is very dissimilar to training data
  - model trained on diabetes fondus photos of subpopulation A
  - never saw subpopulation B
    - “images are outside data distribution model was trained on”
  - desired behaviour
    - return a prediction (attempting to extrapolate)
    - +information that image lies outside data distribution
  - (model retrained w subpop. B labels → low uncertainty on these)



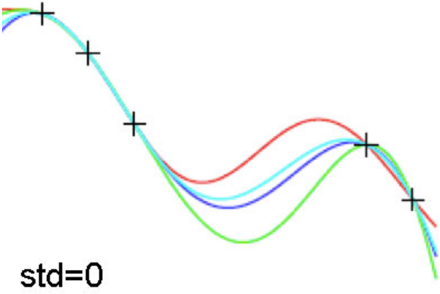
Sources of uncertainty

- Above are some *examples* of uncertainty
- Many other sources of uncertainty
- Test data is very dissimilar to training data
  - model trained on diabetes fondus photos of subpopulation A
  - never saw subpopulation B
    - "images are outside data distribution model was trained on"
  - desired behaviour
    - return a prediction (attempting to extrapolate)
    - +information that image lies outside data distribution
  - (model retrained w subpop. B labels → low uncertainty on these)



Sources of uncertainty (cnt)

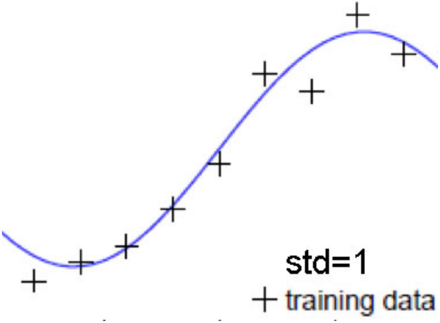
- Uncertainty in model parameters that best explain data
  - large number of possible models can explain a dataset
  - uncertain which model parameters to choose to predict with
  - affects how we predict with new test points



Sources of uncertainty (cnt)

- Training labels are noisy
  - measurement imprecision
  - expert mistakes
  - crowd sourced labels

even infinity data → ambiguity inherent in data itself

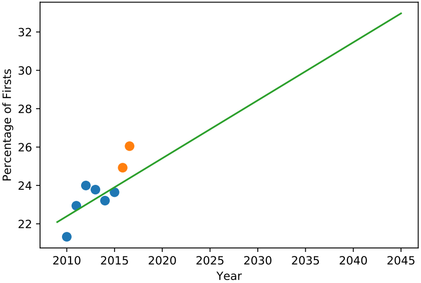


Deep learning models are deterministic

Deep learning does not capture uncertainty:

- regression models output a single scalar/vector

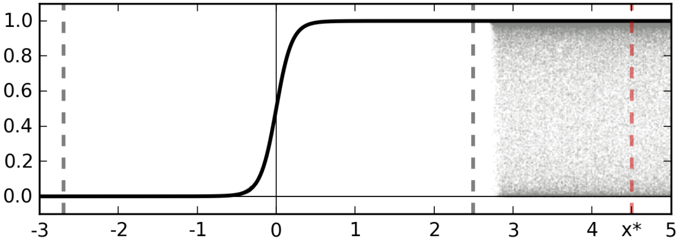
- classification models output a probability vector (erroneously interpreted as model uncertainty)



Deep learning models are deterministic

Deep learning does not capture uncertainty:

- regression models output a single scalar/vector
- classification models output a probability vector (erroneously interpreted as model uncertainty)



Deep learning models are deterministic

Deep learning does not capture uncertainty:

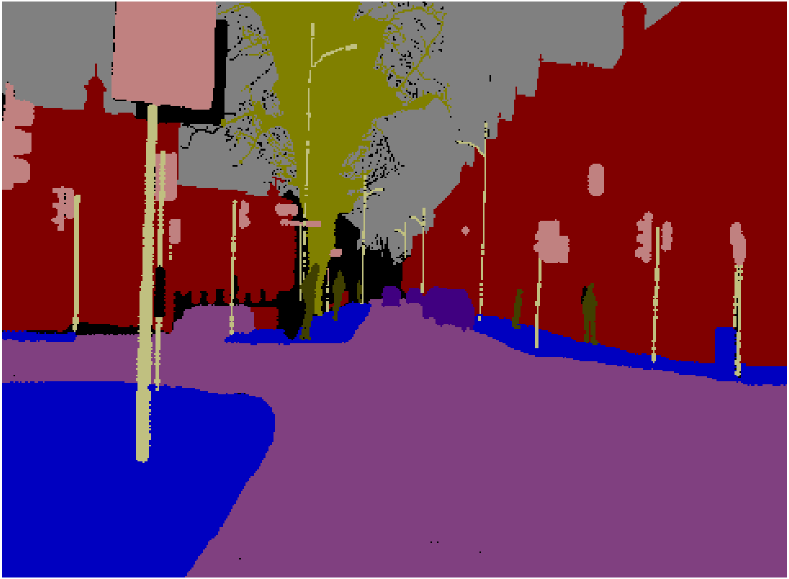
- regression models output a single scalar/vector
- classification models output a probability vector (erroneously interpreted as model uncertainty)

But when combined with *probability theory* **can** capture uncertainty in a principled way

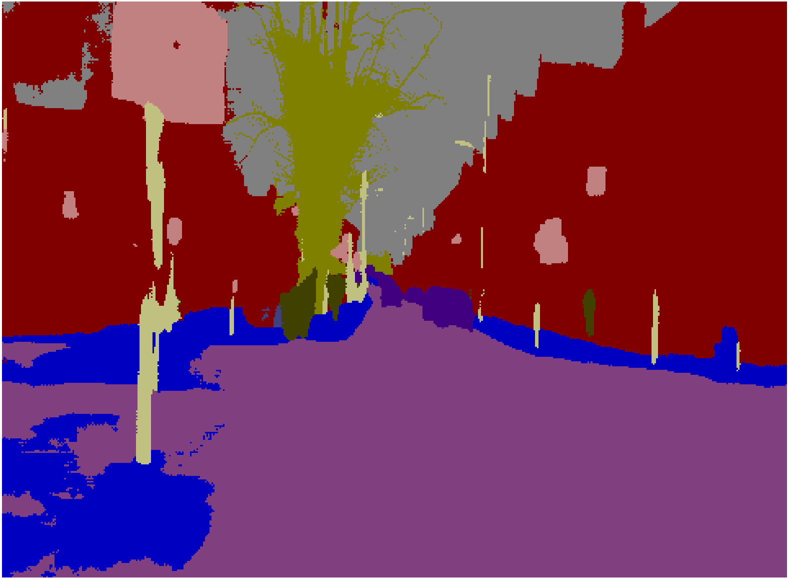
→ known as **Bayesian Deep Learning**

Teaser: Uncertainty in Autonomous Driving

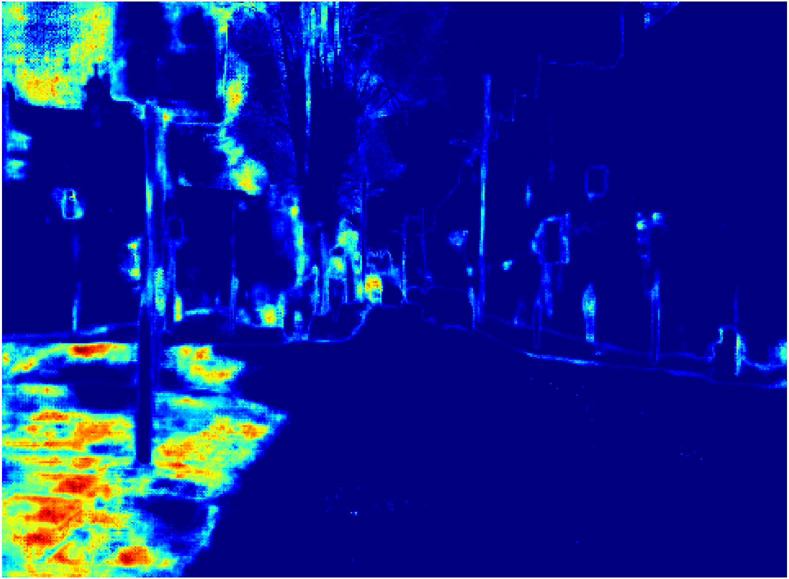
Teaser: Uncertainty in Autonomous Driving



Teaser: Uncertainty in Autonomous Driving



Teaser: Uncertainty in Autonomous Driving



Teaser

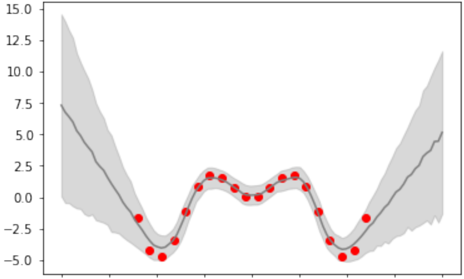
Define model and train on data `x_train`, `y_train`:

```
from tensorflow.keras.layers import Input, Dense, Dropou

inputs = Input(shape=(1,))
x = Dense(512, activation="relu")(inputs)
x = Dropout(0.5)(x, training=True)
x = Dense(512, activation="relu")(x)
x = Dropout(0.5)(x, training=True)
outputs = Dense(1)(x)

model = tf.keras.Model(inputs, outputs)
model.compile(loss="mean_squared_error",
                optimizer="adam")
model.fit(x_train, y_train)
```

Teaser

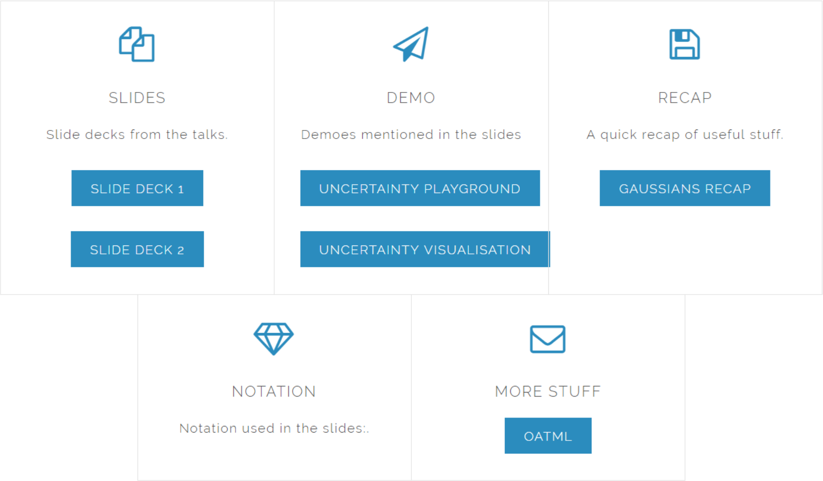
```python
# do stochastic forward passes on x_test:
samples = [model.predict(x_test) for _ in range(100)]
m = np.mean(samples, axis=0) # predictive mean
v = np.var(samples, axis=0) # predictive variance

# plot mean and uncertainty
plt.plot(x_test, m)
plt.fill_between(x_test, m - 2*v**0.5, m + 2*v**0.5,
                 alpha=0.1) # plot two std (95% confiden
```

Playgroud (working code): bdl101.ml/play



Bayesian deep learning

All resources (including these slides): bdl101.ml



SLIDES

Slide decks from the talks.

DEMO

Demoes mentioned in the slides

RECAP

A quick recap of useful stuff.

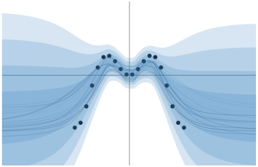
NOTATION

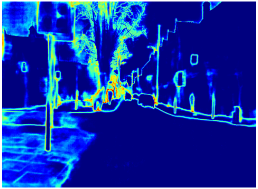
Notation used in the slides:.

MORE STUFF

Bayesian deep learning

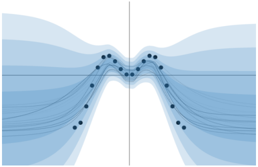
Today and tomorrow we'll understand why this code *makes sense*, and get a taste of

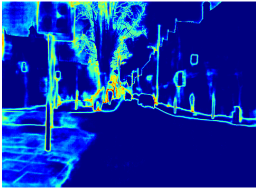




- the formal language of **uncertainty** (Bayesian probability theory)
- tools to use this language in **ML** (Bayesian prob. modelling)
- techniques to scale to real-world **deep learning** systems (modern variational inference)
- developing **big deep learning systems** which convey uncertainty
  - w **real-world** examples

Bayesian deep learning



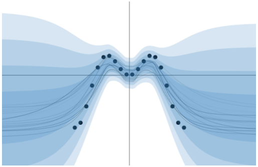


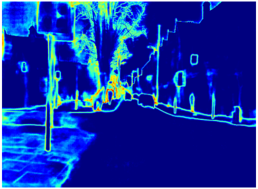
Today and tomorrow we'll understand why this code *makes sense*, and get a taste of

- the formal language of **uncertainty** (Bayesian probability theory)
- tools to use this language in **ML** (Bayesian prob. modelling)
- techniques to scale to real-world **deep learning** systems (modern variational inference)
- developing **big deep learning systems** which convey uncertainty
  - w **real-world** examples

Bayesian deep learning

Today and tomorrow we'll understand why this code *makes sense*, and get a taste of

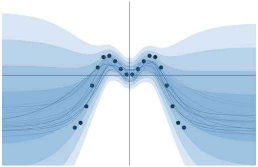


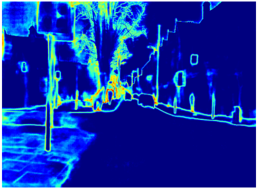


- the formal language of **uncertainty** (Bayesian probability theory)
- tools to use this language in **ML** (Bayesian prob. modelling)
- techniques to scale to real-world **deep learning** systems (modern variational inference)
- developing **big deep learning systems** which convey uncertainty
  - w **real-world** examples

Bayesian deep learning

Today and tomorrow we'll understand why this code *makes sense*, and get a taste of

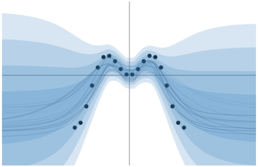


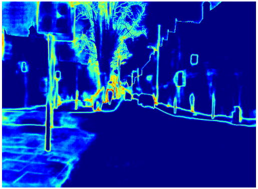


- the formal language of **uncertainty** (Bayesian probability theory)
- tools to use this language in **ML** (Bayesian prob. modelling)
- techniques to scale to real-world **deep learning** systems (modern variational inference)
- developing **big deep learning systems** which convey uncertainty
  - w **real-world** examples

Bayesian deep learning

Today and tomorrow we'll understand why this code *makes sense*, and get a taste of





- the formal language of **uncertainty** (Bayesian probability theory)
- tools to use this language in **ML** (Bayesian prob. modelling)
- techniques to scale to real-world **deep learning** systems (modern variational inference)
- developing **big deep learning systems** which convey uncertainty
    - w **real-world** examples

Basic concepts marked green (if you want to use as tools); Advanced topics marked amber (if you want to develop new stuff in BDL)

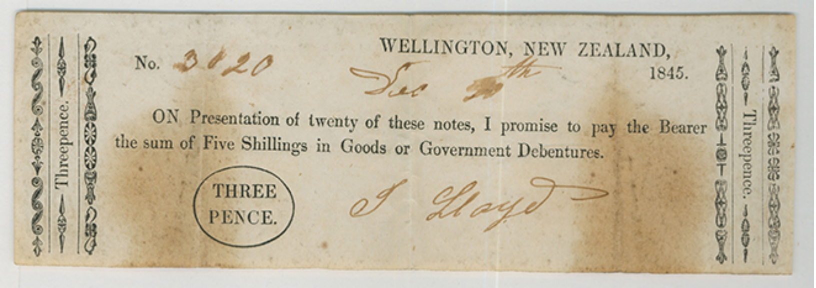
Bayesian Probability Theory:
the Language of Uncertainty

Deriving the laws of *probability theory* from *rational degrees of belief*

Betting game 1 (some philosophy for the soul)

```
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print('Heads')
  else:
    print('Tails')
```

- *unit wager*: a 'promise note' where seller commits to pay note owner £1 if outcome of toss='heads'; a tradeable note; eg..



Threepence.

No. 3420

WELLINGTON, NEW ZEALAND,
Dec [illegible]th 1845.

ON Presentation of twenty of these notes, I promise to pay the Bearer the sum of Five Shillings in Goods or Government Debentures.

THREE PENCE.

J. Lloyd

Threepence.

Betting game 1 (some philosophy for the soul)

```python
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print('Heads')
  else:
    print('Tails')
```

- *unit wager*: a 'promise note' where seller commits to pay note owner £1 if outcome of toss='heads'; a tradeable note; eg..

- would you pay p=£0.01 for a unit wager on 'heads'?
  - pay a penny to buy a note where I commit to paying £1 if 'heads'
- p=£0.99?
  - pay 99 pence for a note where I commit to paying £1 if 'heads'
- up to £0.05?, £0.95 or above?, ...

Betting game 1 (some philosophy for the soul)

```python
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print(’Heads’)
  else:
    print(’Tails’)
```

- *unit wager*: a ‘promise note’ where seller commits to pay note owner £1 if outcome of toss=‘heads’; a tradeable note; eg..

- would you pay p=£0.01 for a unit wager on ‘heads’?
  - pay a penny to buy a note where I commit to paying £1 if ‘heads’
- p=£0.99?
  - pay 99 pence for a note where I commit to paying £1 if ‘heads’
- up to £0.05?, £0.95 or above?, ...

Betting game 1 (some philosophy for the soul)

```python
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print(’Heads’)
  else:
    print(’Tails’)
```

- *unit wager*: a ‘promise note’ where seller commits to pay note owner £1 if outcome of toss=‘heads’; a tradeable note; eg..

- would you pay p=£0.01 for a unit wager on ‘heads’?
  - pay a penny to buy a note where I commit to paying £1 if ‘heads’
- p=£0.99?
  - pay 99 pence for a note where I commit to paying £1 if ‘heads’
- up to £0.05?, £0.95 or above?, ...

Betting game 1b (some philosophy for the soul)

```python
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print('Heads')
  else:
    print('Tails')
```

- *Unit wager*: note seller commits to paying £1 if outcome='heads'

- would you **sell** a unit wager at £p for 'heads'?
  - you get £p for the note, and have to pay £1 if heads

- up to £0.05?, £0.95 or above?, ...

Betting game 1c (some philosophy for the soul)

```python
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print(’Heads’)
  else:
    print(’Tails’)
```

- *Unit wager*: note seller commits to paying £1 if outcome=‘heads’

- what if you have to set price £p, and commit to either **sell** unit wager at £p for ‘heads’, or **buy** one?
  - **I decide** whether to sell to you, or buy from you
  - I sell: you pay £p to buy note where I commit to paying £1 if heads
  - I buy: you get £p for note, and have to pay me £1 if heads

- up to £0.05?, £0.95 or above?, ...

Betting game 1c (some philosophy for the soul)

```python
import numpy as np
def toss():
  if np.random.rand() < 0.5:
    print('Heads')
  else:
    print('Tails')
```

- *Unit wager*: note seller commits to paying £1 if outcome='heads'

- what if you have to set price £p, and commit to either **sell** unit wager at £p for 'heads', or **buy** one?
  - **I decide** whether to sell to you, or buy from you
  - I sell: you pay £p to buy note where I commit to paying £1 if heads
  - I buy: you get £p for note, and have to pay me £1 if heads

- up to £0.05?, £0.95 or above?, ...

Beliefs as willingness to wager

- A person with degree of belief p in event A is assumed to be willing to pay $\leq$ £p for a unit wager on A
- and is willing to sell such a wager for any price $\geq$ £p
- This p captures our degree of **belief about the event** A **taking place** (aka **uncertainty**, confidence)

Beliefs as willingness to wager

- A person with degree of belief p in event A is assumed to be willing to pay $\leq$ £p for a unit wager on A
- and is willing to sell such a wager for any price $\geq$ £p
- This p captures our degree of **belief about the event** A **taking place** (aka **uncertainty**, confidence)

Beliefs as willingness to wager

- A person with degree of belief p in event A is assumed to be willing to pay $\leq$ £p for a unit wager on A
- and is willing to sell such a wager for any price $\geq$ £p
- This p captures our degree of **belief about the event A taking place** (aka **uncertainty**, confidence)

Rational beliefs

- Two notes (unit wagers):
  - Note 1: 'outcome=heads'
  - Note 2: 'outcome=tails'

  you decide p for note 1 and q for note 2; I decide whether to buy from you or sell you each note at the price you determined

- if $p + q < 1$ then I will buy from you note 1 for £p and also note 2 for £q

- whatever outcome you give me £1; but because I gave you $p + q < 1$, you lost £$1 - p - q$

- **Dutch book**: a set of unit wager notes where **you decide the odds** (wager price) and **I decide whether to buy or sell** each note ... and you are guaranteed to always lose money.

Rational beliefs

- Two notes (unit wagers):
  - Note 1: ‘outcome=heads’
  - Note 2: ‘outcome=tails’

  you decide p for note 1 and q for note 2; I decide whether to buy from you or sell you each note at the price you determined

- if $p + q < 1$ then I will buy from you note 1 for £p and also note 2 for £q

- whatever outcome you give me £1; but because I gave you $p + q < 1$, you lost £$1 - p - q$

- **Dutch book**: a set of unit wager notes where **you decide the odds** (wager price) and **I decide whether to buy or sell** each note ... and you are guaranteed to always lose money.

Rational beliefs

- Two notes (unit wagers):
  - Note 1: ‘outcome=heads’
  - Note 2: ‘outcome=tails’

  you decide p for note 1 and q for note 2; I decide whether to buy from you or sell you each note at the price you determined

- if $p + q < 1$ then I will buy from you note 1 for £p and also note 2 for £q

- whatever outcome you give me £1; but because I gave you $p + q < 1$, you lost £$1 - p - q$

- **Dutch book**: a set of unit wager notes where **you decide the odds** (wager price) and **I decide whether to buy or sell** each note ... and you are guaranteed to always lose money.

Rational beliefs

- Two notes (unit wagers):
  - Note 1: ‘outcome=heads’
  - Note 2: ‘outcome=tails’

  you decide p for note 1 and q for note 2; I decide whether to buy from you or sell you each note at the price you determined

- if $p + q < 1$ then I will buy from you note 1 for £p and also note 2 for £q

- whatever outcome you give me £1; but because I gave you $p + q < 1$, you lost £$1 - p - q$

- **Dutch book**: a set of unit wager notes where **you decide the odds** (wager price) and **I decide whether to buy or sell** each note ... and you are guaranteed to always lose money.

Rational beliefs

- Two notes (unit wagers):
  - Note 1: ‘outcome=heads’
  - Note 2: ‘outcome=tails’

  you decide p for note 1 and q for note 2; I decide whether to buy from you or sell you each note at the price you determined

- if $p + q < 1$ then I will buy from you note 1 for £p and also note 2 for £q

- whatever outcome you give me £1; but because I gave you $p + q < 1$, you lost £$1 - p - q$

- **Dutch book**: a set of unit wager notes where **you decide the odds** (wager price) and **I decide whether to buy or sell** each note ... and you are guaranteed to always lose money.

Set of beliefs is called **rational** if no Dutch book exists.

Formalism (rational beliefs = prob theory)

Setup

- Def sample space X of simple events (possible outcomes)
  - e.g. experiment flipping two coins X={HH, HT, TH, TT}
- Let A be an event (a subset of X). A holding true = at least one of the outcomes in A happened
  - e.g. "at least one heads" ↔ A={HH, HT, TH}
- Write p_A for belief of event A (your wager on A happening, assuming all wagers are unit wagers)

Formalism (rational beliefs = prob theory)

Setup

- Def sample space X of simple events (possible outcomes)
  - e.g. experiment flipping two coins X={HH, HT, TH, TT}
- Let A be an event (a subset of X). A holding true = at least one of the outcomes in A happened
  - e.g. "at least one heads" $\leftrightarrow$ A={HH, HT, TH}
- Write p_A for belief of event A (your wager on A happening, assuming all wagers are unit wagers)

Can show that $\{p_A\}_{A\subseteq X}$ are rational beliefs iff $\{p_A\}_{A\subseteq X}$ satisfies laws of probability theory

- Already showed that $p_A + p_{A^c} = 1$
- Try to devise other betting games at home (bdl101.ml/betting)

Formalism (rational beliefs = prob theory)

Setup

- Def sample space X of simple events (possible outcomes)
  - e.g. experiment flipping two coins X={HH, HT, TH, TT}
- Let A be an event (a subset of X). A holding true = at least one of the outcomes in A happened
  - e.g. "at least one heads" ↔ A={HH, HT, TH}
- Write p_A for belief of event A (your wager on A happening, assuming all wagers are unit wagers)

Can show that $\{p_A\}_{A \subseteq X}$ are rational beliefs iff $\{p_A\}_{A \subseteq X}$ satisfies laws of probability theory

- Already showed that $p_A + p_{A^c} = 1$
- Try to devise other betting games at home (bdl101.ml/betting)

Can derive the laws of prob theory from rational beliefs!

- → if you want to be rational, must follow laws of probability (otherwise someone can take advatnge of your model)

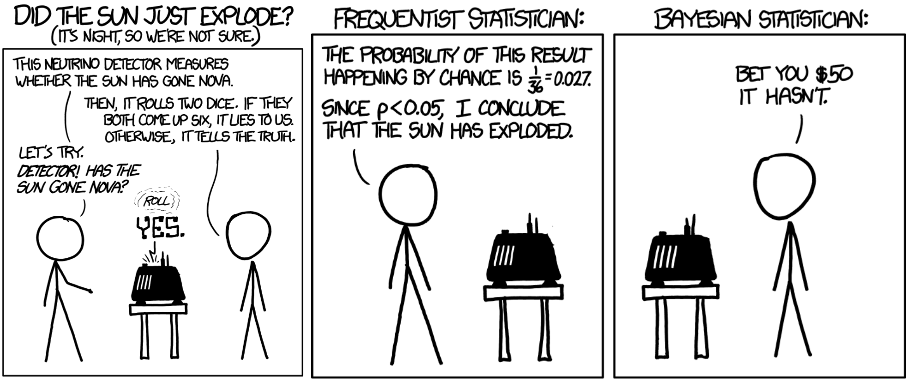
Probability as belief vs frequency

Above known as **Bayesian prob theory**

- forms an **interpretation of the laws of probability**, and formalises **our notion of uncertainty** in events
- vs ‘Frequency as probability’
    - only applicable to repeatable events (eg, try to answer ‘will Trump win 2020’)
    - also other issues; eg p-hacking
    - Psychology journal banning p values

(although there are problems w Bayesian arguments as well)

‘Real-world’ example


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY.
DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.
FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS $\frac{1}{36}$=0.027. SINCE p<0.05, I CONCLUDE THAT THE SUN HAS EXPLODED.
BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

[https://xkcd.com/1132/]

Bayesian Probabilistic Modelling (an Introduction)

Simple idea: *"If you're doing something which doesn't follow from the laws of probability, then you're doing it wrong"*

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)
- eg – astrophysics: gravitational lensing
  - there exists a physics process magnifying far away galaxies
  - Nature chose lensing coeff → gravitational lensing mechanism → transform galaxy
  - We **observe** transformed galaxies, want to **infer** lensing ceoff

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)
- eg – cats vs dogs classification
  - there exist some underlying rules we don't know
  - eg "if has pointy ears then cat"
  - We **observe** pairs (image, "cat" / "no cat"), and want to **infer** underlying mapping from images to labels

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)
- eg – Gaussian density estimation
  - I tell you the process I used to generate data and give 5 data points

$$x_n \sim \mathcal{N}(x_n; \mu, \sigma^2), \qquad \sigma = 1$$

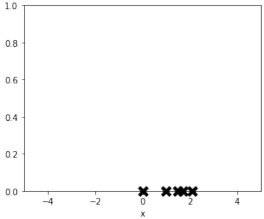
  - you **observe** the points $\{x_1, ..., x_5\}$, and want to **infer** my μ
  - Reminder: Gaussian density with mean μ and variance σ^2

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)

- eg – Gaussian density estimation



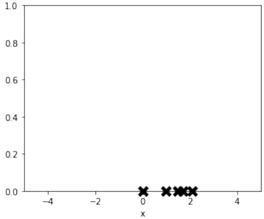
X Which μ generated my data?

V What's the probability that $\mu = 10$ generated my data? (want to infer distribution over μ!)

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)

- eg – Gaussian density estimation



X Which μ generated my data?

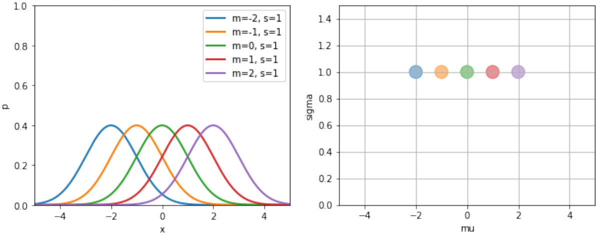
V What's the probability that $\mu = 10$ generated my data? (want to infer distribution over μ!)

Bayesian Probabilistic Modelling

- can't do ML without **assumptions**
  - **must** make some assumptions about how data was generated
  - there always exists some **underlying process** that generated obs
  - in Bayesian probabilistic modelling we make our assumptions about underlying process **explicit**
  - want to **infer** underlying process (find dist that generated data)

- eg – Gaussian density estimation

- These are the hypotheses we'll play with



- I chose a Gaussian (one of those) from which I generated data

Generative story / model

In Bayesian probabilistic modelling

- want to represent our beliefs / assumptions about how data was generated explicitly
- eg via *generative story* ['*My assumptions are...*']:
  - Someone (me / Nature / etc) selected parameters $\mu*, \sigma*$
  - Generated N data points $x_n \sim \mathcal{N}(\mu*, \sigma*^2)$
  - Gave us $\mathcal{D} = \{x_1, ..., x_N\}$
  - $\rightarrow$ how would you formalise this process?
- Bayesian probabilistic model:
  - prior [what I believe params might look like]

$$\mu \sim \mathcal{N}(0, 10), \qquad \sigma = 1$$

  - likelihood [how I believe data was generated given params]

$$x_n \mid \mu, \sigma \sim \mathcal{N}(\mu, \sigma^2)$$

- will update prior belief on μ conditioned on data you give me (infer distribution over μ): $\mu \mid \{x_1, ..., x_N\}$

Generative story / model

In Bayesian probabilistic modelling

- want to represent our beliefs / assumptions about how data was generated explicitly
- eg via *generative story* ['*My assumptions are...*']:
  - Someone (me / Nature / etc) selected parameters $\mu*, \sigma*$
  - Generated N data points $x_n \sim \mathcal{N}(\mu*, \sigma*^2)$
  - Gave us $\mathcal{D} = \{x_1, ..., x_N\}$
  - $\rightarrow$ how would you formalise this process?
- Bayesian probabilistic model:
  - prior [what I believe params might look like]

$$\mu \sim \mathcal{N}(0, 10), \qquad \sigma = 1$$

  - likelihood [how I believe data was generated given params]

$$x_n \mid \mu, \sigma \sim \mathcal{N}(\mu, \sigma^2)$$

- will update prior belief on μ conditioned on data you give me (infer distribution over μ): $\mu \mid \{x_1, ..., x_N\}$

Generative story / model

In Bayesian probabilistic modelling

- want to represent our beliefs / assumptions about how data was generated explicitly
- eg via *generative story* [‘*My assumptions are...*’]:
  - Someone (me / Nature / etc) selected parameters $\mu*, \sigma*$
  - Generated N data points $x_n \sim \mathcal{N}(\mu*, \sigma*^2)$
  - Gave us $\mathcal{D} = \{x_1, ..., x_N\}$
  - $\rightarrow$ how would you formalise this process?
- Bayesian probabilistic model:
  - prior [what I believe params might look like]

$$\mu \sim \mathcal{N}(0, 10), \qquad \sigma = 1$$

  - likelihood [how I believe data was generated given params]

$$x_n \mid \mu, \sigma \sim \mathcal{N}(\mu, \sigma^2)$$

- will update prior belief on μ conditioned on data you give me (infer distribution over μ): $\mu \mid \{x_1, ..., x_N\}$

Generative story / model

In Bayesian probabilistic modelling

- want to represent our beliefs / assumptions about how data was generated explicitly

- eg via *generative story* ['*My assumptions are...*']:
  - Someone (me / Nature / etc) selected parameters $\mu*, \sigma*$
  - Generated N data points $x_n \sim \mathcal{N}(\mu*, \sigma*^2)$
  - Gave us $\mathcal{D} = \{x_1, ..., x_N\}$
  - $\rightarrow$ how would you formalise this process?

- Bayesian probabilistic model:
  - prior [what I believe params might look like]

$$\mu \sim \mathcal{N}(0, 10), \qquad \sigma = 1$$

  - likelihood [how I believe data was generated given params]

$$x_n \mid \mu, \sigma \sim \mathcal{N}(\mu, \sigma^2)$$

- will update prior belief on μ conditioned on data you give me (infer distribution over μ): $\mu \mid \{x_1, ..., x_N\}$

Can you find my Gaussian?

How can you infer μ? (find distribution)

Everything follows the **laws of prob**..

- Sum rule

$$p(X = x) = \sum_{y} p(X = x, Y = y) = \int p(X = x, Y) \mathrm{d}Y$$

- Product rule

$$p(X = x, Y = y) = p(X = x | Y = y) p(Y = y)$$

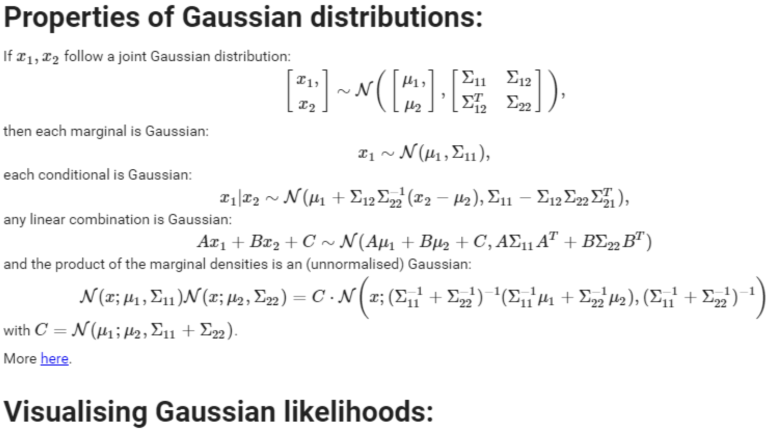
- Bayes rule

$$p(X = x | Y = y, \mathcal{H}) = \frac{p(Y = y | X = x, \mathcal{H}) p(X = x | \mathcal{H})}{p(Y = y | \mathcal{H})}$$

Note: $\mathcal{H}$ is often omitted in conditional for brevity

Can you find my Gaussian?

Remember: products, ratios, marginals, and conditionals of Gaussians are Gaussian!



Properties of Gaussian distributions:

If x_1, x_2 follow a joint Gaussian distribution:

$$\begin{bmatrix} x_1, \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1, \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{bmatrix} \right),$$

then each marginal is Gaussian:

$$x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11}),$$

each conditional is Gaussian:

$$x_1 | x_2 \sim \mathcal{N}(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}\Sigma_{21}^T),$$

any linear combination is Gaussian:

$$Ax_1 + Bx_2 + C \sim \mathcal{N}(A\mu_1 + B\mu_2 + C, A\Sigma_{11}A^T + B\Sigma_{22}B^T)$$

and the product of the marginal densities is an (unnormalised) Gaussian:

$$\mathcal{N}(x; \mu_1, \Sigma_{11})\mathcal{N}(x; \mu_2, \Sigma_{22}) = C \cdot \mathcal{N}\left(x; (\Sigma_{11}^{-1} + \Sigma_{22}^{-1})^{-1}(\Sigma_{11}^{-1}\mu_1 + \Sigma_{22}^{-1}\mu_2), (\Sigma_{11}^{-1} + \Sigma_{22}^{-1})^{-1} \right)$$

with $C = \mathcal{N}(\mu_1; \mu_2, \Sigma_{11} + \Sigma_{22})$.

More here.

Visualising Gaussian likelihoods:

Summary (and playgroud) here: bdl101.ml/gauss

Inference

Bayes rule:

$$p(X = x|Y = y, \mathcal{H}) = \frac{p(Y = y|X = x, \mathcal{H})p(X = x|\mathcal{H})}{p(Y = y|\mathcal{H})},$$

and in probabilistic modelling:

$$\overbrace{p(\mu|\mathcal{D}, \sigma, \mathcal{H})}^{\text{Posterior}} = \frac{\overbrace{p(\mathcal{D}|\mu, \sigma, \mathcal{H})}^{\text{Likelihood}}\overbrace{p(\mu|\sigma, \mathcal{H})}^{\text{Prior}}}{\underbrace{p(\mathcal{D}|\sigma, \mathcal{H})}_{\text{Model evidence}}}$$

with model evidence $p(\mathcal{D}|\sigma, \mathcal{H}) = \int p(\mathcal{D}|\mu, \sigma, \mathcal{H})p(\mu|\sigma, \mathcal{H})\mathrm{d}\mu$ (sum rule).

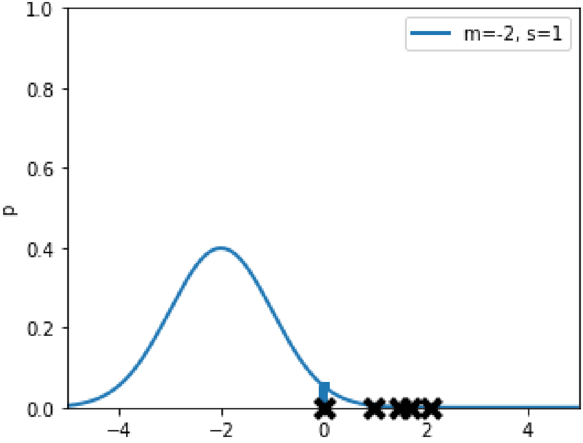
Likelihood

- we explicitly assumed data comes iid from a Gaussian
- compute $p(\mathcal{D}|\mu, \sigma)$ = multiply all $p(x_n|\mu, \sigma)$ (product rule)
- prob of observing data points for given params

The Likelihood in more detail

Likelihood

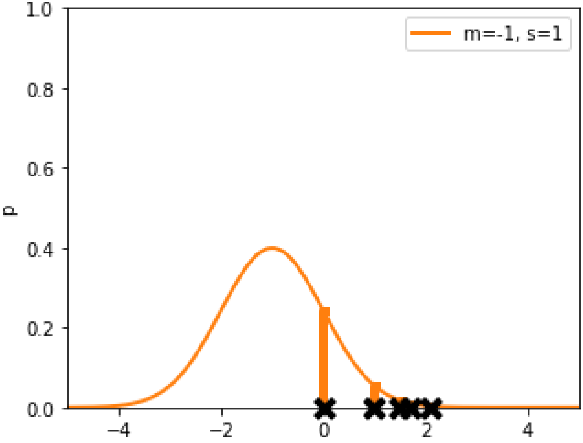
- we explicitly assumed data comes iid from a Gaussian
- compute $p(\mathcal{D}|\mu,\sigma)$ = multiply all $p(x_n|\mu,\sigma)$ (product rule)
- prob of observing data points for given params



The Likelihood in more detail

Likelihood

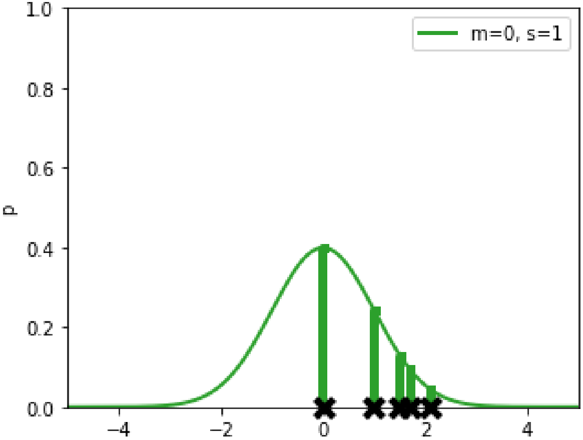
- we explicitly assumed data comes iid from a Gaussian
- compute $p(\mathcal{D}|\mu,\sigma)$ = multiply all $p(x_n|\mu,\sigma)$ (product rule)
- prob of observing data points for given params



The Likelihood in more detail

Likelihood

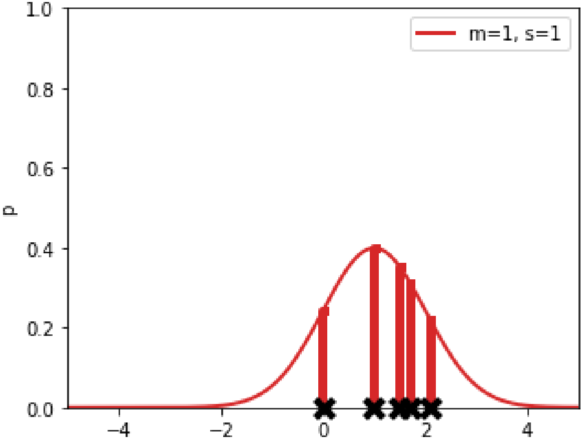
- we explicitly assumed data comes iid from a Gaussian
- compute $p(\mathcal{D}|\mu, \sigma)$ = multiply all $p(x_n|\mu, \sigma)$ (product rule)
- prob of observing data points for given params



The Likelihood in more detail

Likelihood

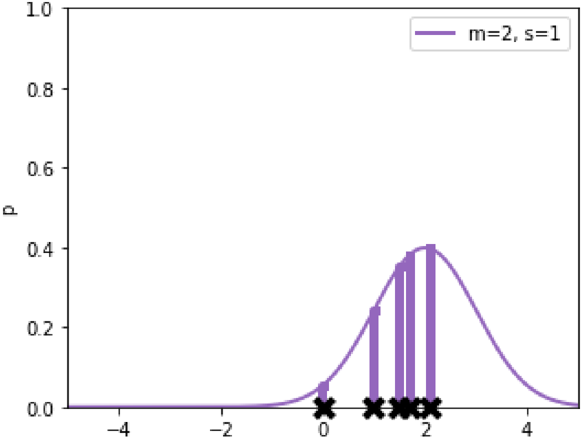
- we explicitly assumed data comes iid from a Gaussian
- compute $p(\mathcal{D}|\mu,\sigma)$ = multiply all $p(x_n|\mu,\sigma)$ (product rule)
- prob of observing data points for given params



The Likelihood in more detail

Likelihood

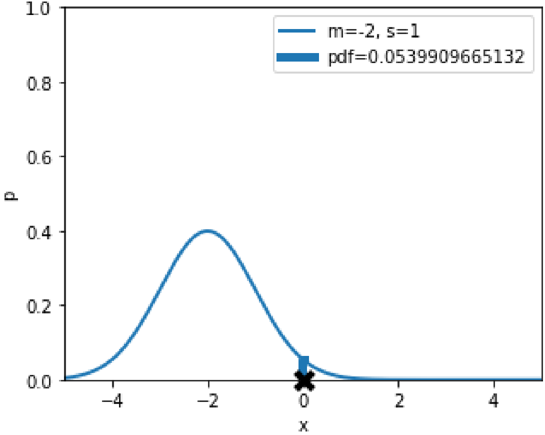
- we explicitly assumed data comes iid from a Gaussian
- compute $p(\mathcal{D}|\mu,\sigma)$ = multiply all $p(x_n|\mu,\sigma)$ (product rule)
- prob of observing data points for given params



Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

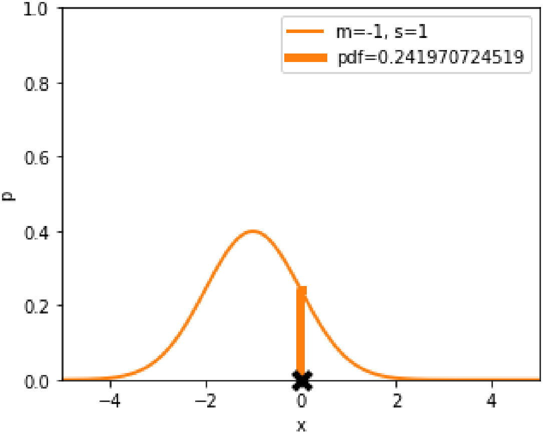
- What does the likelihood look like?



Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

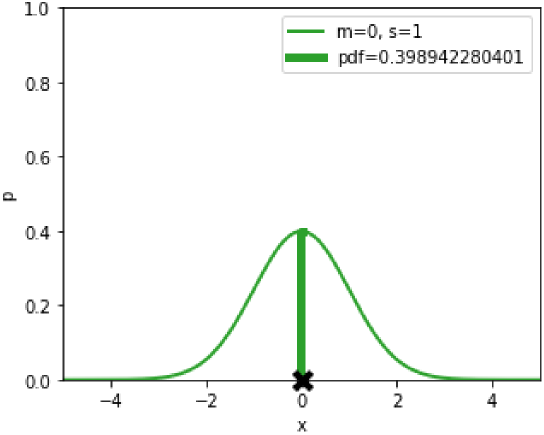
- What does the likelihood look like?



Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

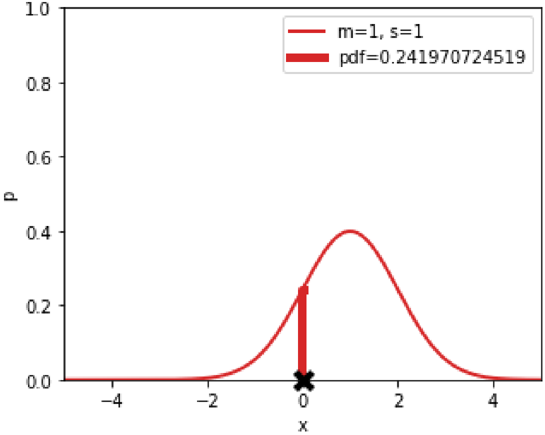
- What does the likelihood look like?



Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

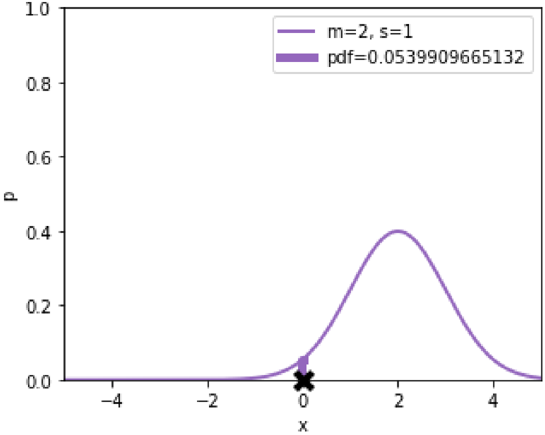
- What does the likelihood look like?



Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

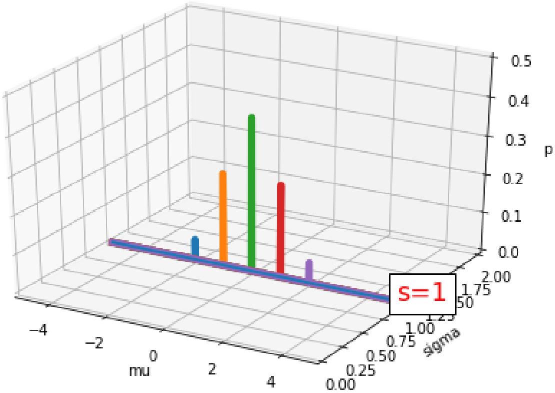
- What does the likelihood look like?



Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

- What does the likelihood look like?

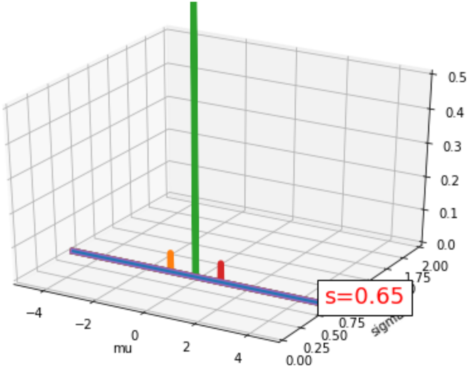


Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

- What does the likelihood look like?

and with smaller σ..



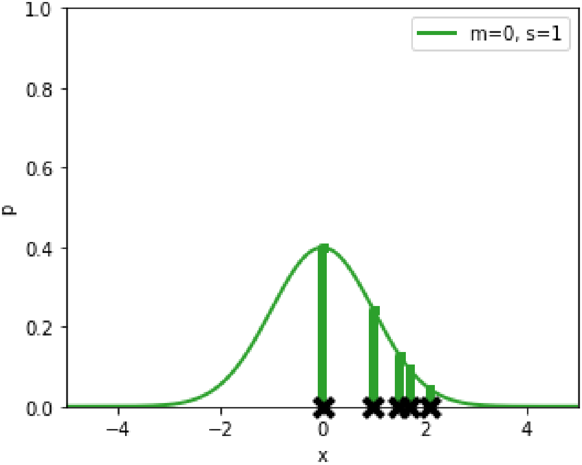
- Trying to max lik will get "absolutely certain that $\sigma = 0$ & $\mu = 0$"
- Does this make sense? (I **told** you $x_n \sim \mathcal{N}$!)
- MLE failure

Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

- What does the likelihood look like?

And with all data:

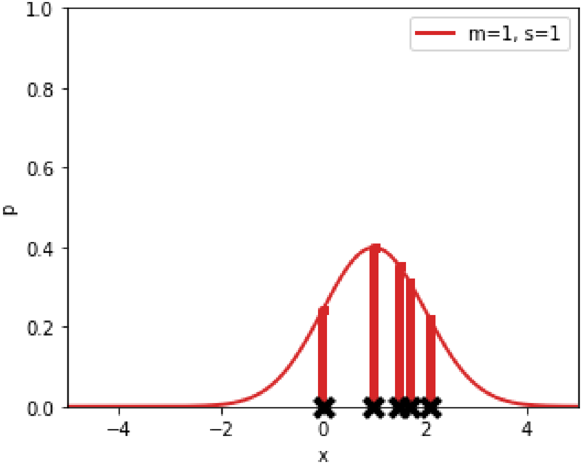


Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

- What does the likelihood look like?

And with all data:

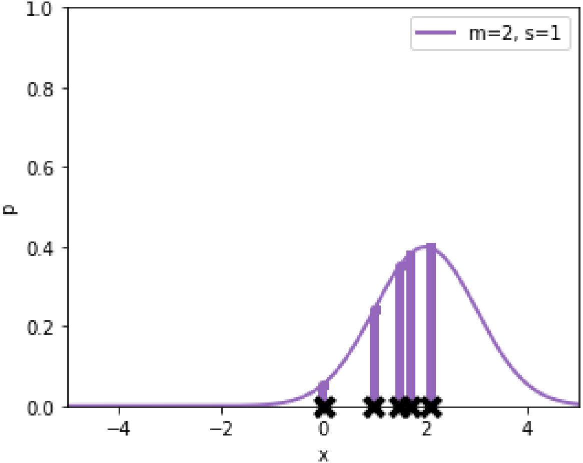


Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

- What does the likelihood look like?

And with all data:

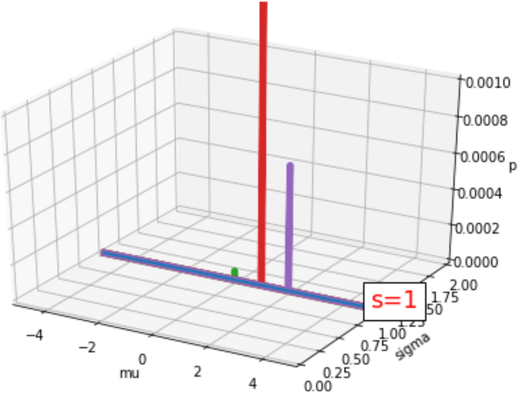


Likelihood as a function of parameters

Reducing dataset from 5 points to 1:

- What does the likelihood look like?

And with all data:



Likelihood function shows how well every value of μ, σ predicted what would happen.

The Posterior in more detail

$$\overbrace{p(\mu|\mathcal{D},\sigma,\mathcal{H})}^{\text{Posterior}} = \frac{\overbrace{p(\mathcal{D}|\mu,\sigma,\mathcal{H})}^{\text{Likelihood}}\overbrace{p(\mu|\sigma,\mathcal{H})}^{\text{Prior}}}{\underbrace{p(\mathcal{D}|\sigma,\mathcal{H})}_{\text{Model evidence}}}$$

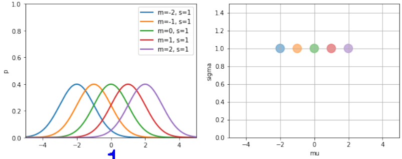
with model evidence $p(\mathcal{D}|\sigma,\mathcal{H}) = \int p(\mathcal{D}|\mu,\sigma,\mathcal{H})p(\mu|\sigma,\mathcal{H})\mathrm{d}\mu$ (sum rule). In contrast to the likelihood, posterior would say

> ‘*with the data you gave me, this is what I currently think μ could be, and I might become more certain if you give me more data*’

- normaliser = marginal likelihood = evidence = sum of likelihood * prior
  - (but often difficult to calculate... more in the next lecture)

The Posterior in more detail

- Eg, inference w prior = ‘we believe data is equally likely to have come from one of the 5 Gaussians w $\sigma = 1$’



$$p(\mu = \mu_i|\sigma, \mathcal{H}) = \frac{1}{5} \text{ and } p(\mu \neq \mu_i \text{ for all } i|\sigma, \mathcal{H}) = 0$$

then marginal likelihood is

$$p(\mathcal{D}|\sigma, \mathcal{H}) = \sum_i p(\mathcal{D}|\mu = \mu_i, \sigma, \mathcal{H}) p(\mu = \mu_i|\sigma, \mathcal{H})$$

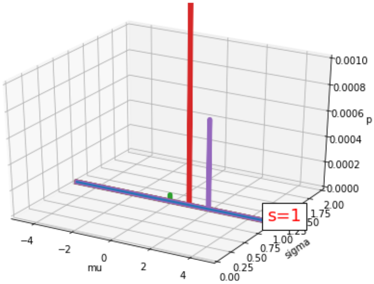
$$= \sum_i p(\mathcal{D}|\mu = \mu_i, \sigma, \mathcal{H}) \frac{1}{5}$$

and posterior is

$$p(\mu = \mu_i|\sigma, \mathcal{D}, \mathcal{H}) = \frac{1/5 p(\mathcal{D}|\mu = \mu_i, \sigma, \mathcal{H})}{\sum_i 1/5 p(\mathcal{D}|\mu = \mu_i, \sigma, \mathcal{H})}$$

The Posterior in more detail

where $p(\mathcal{D}|\mu = \mu_i, \sigma = 1, \mathcal{H})$ is given by

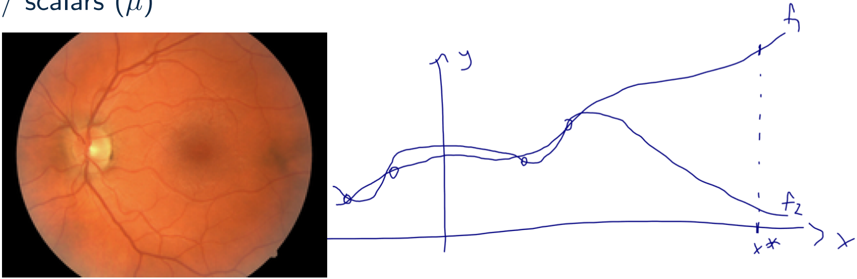


- marginal likelihood of sigma=1 = $p(\mathcal{D}|\sigma = 1, \mathcal{H})$ = 'prob that data came from single Gaussian with param $\sigma = 1$'
- similarly, marginal likelihood of hypothesis = $p(\mathcal{D}|\mathcal{H})$ = 'prob that data came from single Gaussian (with some μ, σ)'

Bayesian Probabilistic Modelling of Functions

Why uncertainty over functions

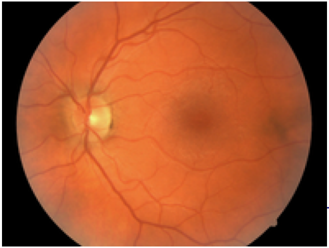
- Example going beyond beliefs over statements (‘heads happened’) / scalars (μ)

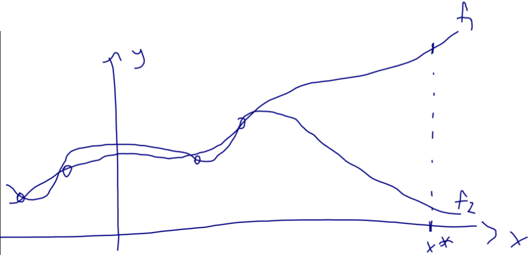


- Would want to know uncertainty (ie belief) of system in prediction
- Want to know distribution over outputs for each input x = dist over *functions*
- First, some preliminaries.. (history, and notation)

Why uncertainty over functions

- Example going beyond beliefs over statements (‘heads happened’) / scalars (μ)

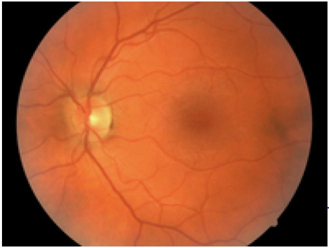


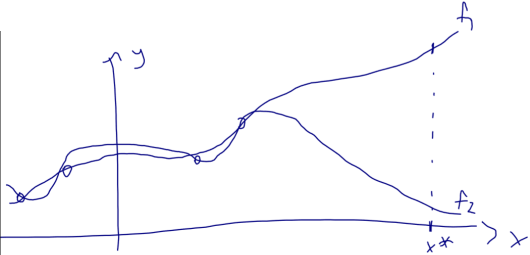


- Would want to know uncertainty (ie belief) of system in prediction
- Want to know distribution over outputs for each input x = dist over *functions*
- First, some preliminaries.. (history, and notation)

Why uncertainty over functions

- Example going beyond beliefs over statements (‘heads happened’) / scalars (μ)





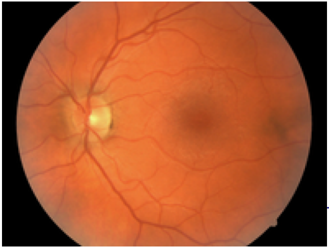
- Would want to know uncertainty (ie belief) of system in prediction

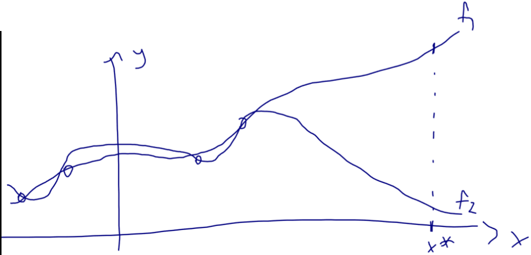
- Want to know distribution over outputs for each input x = dist over *functions*

- First, some preliminaries.. (history, and notation)

Why uncertainty over functions

- Example going beyond beliefs over statements ('heads happened') / scalars (μ)





- Would want to know uncertainty (ie belief) of system in prediction
- Want to know distribution over outputs for each input x = dist over *functions*
- First, some preliminaries.. (history, and notation)

Linear regression

Linear regression [Gauss, 1809]

- Given a set of N input-output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - eg average number of accidents for different driving speeds
- assumes exists linear func mapping **vectors** $\mathbf{x}_i \in \mathbb{R}^Q$ to $\mathbf{y}_i \in \mathbb{R}^D$ (with $\mathbf{y}_i$ potentially corrupted with observation noise)
- model is linear trans. of inputs: $f(x) = W\mathbf{x} + b$, w W some D by Q matrix over reals, b real vector with D elements
- Different params W, b define different linear trans
  - aim: find params that (eg) minimise $1/N \sum_i ||\mathbf{y}_i - (W\mathbf{x}_i + b)||^2$
- but relation between $\mathbf{x}$ and $\mathbf{y}$ need not be linear

Linear regression

Linear regression [Gauss, 1809]

- Given a set of N input-output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - eg average number of accidents for different driving speeds
- assumes exists linear func mapping **vectors** $\mathbf{x}_i \in \mathbb{R}^Q$ to $\mathbf{y}_i \in \mathbb{R}^D$ (with $\mathbf{y}_i$ potentially corrupted with observation noise)
- model is linear trans. of inputs: $f(x) = W\mathbf{x} + b$, w W some D by Q matrix over reals, b real vector with D elements
- Different params W, b define different linear trans
  - aim: find params that (eg) minimise $1/N \sum_i ||\mathbf{y}_i - (W\mathbf{x}_i + b)||^2$
- but relation between $\mathbf{x}$ and $\mathbf{y}$ need not be linear

Linear regression

Linear regression [Gauss, 1809]

- Given a set of N input-output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - eg average number of accidents for different driving speeds
- assumes exists linear func mapping **vectors** $\mathbf{x}_i \in \mathbb{R}^Q$ to $\mathbf{y}_i \in \mathbb{R}^D$ (with $\mathbf{y}_i$ potentially corrupted with observation noise)
- model is linear trans. of inputs: $f(x) = W\mathbf{x} + b$, w W some D by Q matrix over reals, b real vector with D elements
- Different params W, b define different linear trans
  - aim: find params that (eg) minimise $1/N \sum_i ||\mathbf{y}_i - (W\mathbf{x}_i + b)||^2$
- but relation between $\mathbf{x}$ and $\mathbf{y}$ need not be linear

Linear regression

Linear regression [Gauss, 1809]

- Given a set of N input-output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - eg average number of accidents for different driving speeds
- assumes exists linear func mapping **vectors** $\mathbf{x}_i \in \mathbb{R}^Q$ to $\mathbf{y}_i \in \mathbb{R}^D$ (with $\mathbf{y}_i$ potentially corrupted with observation noise)
- model is linear trans. of inputs: $f(x) = W\mathbf{x} + b$, w W some D by Q matrix over reals, b real vector with D elements
- Different params W, b define different linear trans
  - aim: find params that (eg) minimise $1/N \sum_i ||\mathbf{y}_i - (W\mathbf{x}_i + b)||^2$
- but relation between $\mathbf{x}$ and $\mathbf{y}$ need not be linear

Linear regression

Linear regression [Gauss, 1809]

- Given a set of N input-output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - eg average number of accidents for different driving speeds
- assumes exists linear func mapping **vectors** $\mathbf{x}_i \in \mathbb{R}^Q$ to $\mathbf{y}_i \in \mathbb{R}^D$ (with $\mathbf{y}_i$ potentially corrupted with observation noise)
- model is linear trans. of inputs: $f(x) = W\mathbf{x} + b$, w W some D by Q matrix over reals, b real vector with D elements
- Different params W, b define different linear trans
  - aim: find params that (eg) minimise $1/N \sum_i ||\mathbf{y}_i - (W\mathbf{x}_i + b)||^2$
- but relation between $\mathbf{x}$ and $\mathbf{y}$ need not be linear

Linear basis function regression

Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input $\mathbf{x}$ fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of $\mathbf{x}$ itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. = linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. → param. basis functions

Linear basis function regression

Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input $\mathbf{x}$ fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of $\mathbf{x}$ itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. $=$ linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. $\rightarrow$ param. basis functions

Linear basis function regression

Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input $\mathbf{x}$ fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of $\mathbf{x}$ itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. = linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. $\rightarrow$ param. basis functions

Linear basis function regression

Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input $\mathbf{x}$ fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of $\mathbf{x}$ itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. = linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. → param. basis functions

Linear basis function regression

Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input $\mathbf{x}$ fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of $\mathbf{x}$ itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. $=$ linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. $\rightarrow$ param. basis functions

Linear basis function regression

Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input **x** fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of x itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. = linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. $\rightarrow$ param. basis functions

Linear basis function regression

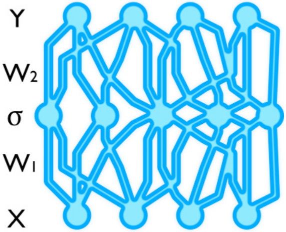
Linear **basis function** regression [Gergonne, 1815; Smith, 1918]

- input $\mathbf{x}$ fed through K fixed scalar-valued non-linear trans. $\phi_k(\mathbf{x})$
- collect into a **feature vector** $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$
- **do linear regression with $\phi(\mathbf{x})$ vector instead of $\mathbf{x}$ itself**
- with scalar input x, trans. can be
  - wavelets parametrised by k: $\cos(k\pi x)e^{-x^2/2}$
  - polynomials of degrees k: x^k
  - sinusoidals with various frequencies: $\sin(kx)$
- When $\phi_k(\mathbf{x}) := \mathbf{x}_k$ and $K = Q$, basis function regr. = linear regr.
- basis functions often assumed fixed and orthogonal to each other (optimal combination is sought)
- but need not be fixed and mutually orth. $\rightarrow$ param. basis functions

Parametrised basis functions

Parametrised basis functions [Bishop, 2006; many others]

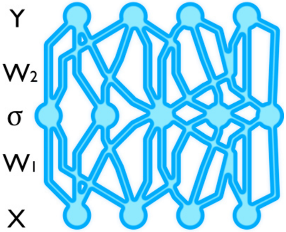
- eg basis functions $\phi_k^{w_k,b_k}$ where scalar-valued function ϕ_k is applied to inner-product $w_k^T\mathbf{x} + b_k$
  - ϕ_k often def'd to be identical for all k (only params change)
  - eg $\phi_k(\cdot) = \tanh(\cdot)$, giving $\phi_k^{w_k,b_k}(\mathbf{x}) = \tanh(w_k^T\mathbf{x} + b_k)$
- feature vector = basis functions' outputs = input to linear trans.
- in vector form:
  - W_1 a matrix of dimensions Q by K
  - b_1 a vector with K elements
  - $\phi^{W_1,b_1}(\mathbf{x}) = \phi(W_1\mathbf{x} + b_1)$
  - W_2 a matrix of dimensions K by D
  - b_2 a vector with D elements
  - model output: $f^{W_1,b_1,W_2,b_2}(\mathbf{x}) = \phi^{W_1,b_1}(\mathbf{x})W_2 + b_2$
- want to find W_1, b_1, W_2, b_2 that minimise $1/N\sum_i ||\mathbf{y}_i - f^{W_1,b_1,W_2,b_2}(\mathbf{x}_i)||^2$



Parametrised basis functions

Parametrised basis functions [Bishop, 2006; many others]

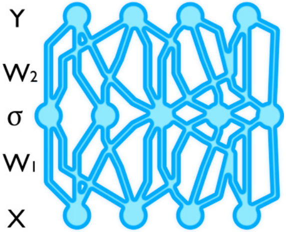
- eg basis functions $\phi_k^{w_k,b_k}$ where scalar-valued function ϕ_k is applied to inner-product $w_k^T\mathbf{x} + b_k$
  - ϕ_k often def'd to be identical for all k (only params change)
  - eg $\phi_k(\cdot) = \tanh(\cdot)$, giving $\phi_k^{w_k,b_k}(\mathbf{x}) = \tanh(w_k^T\mathbf{x} + b_k)$
- feature vector = basis functions' outputs = input to linear trans.
- in vector form:
  - W_1 a matrix of dimensions Q by K
  - b_1 a vector with K elements
  - $\phi^{W_1,b_1}(\mathbf{x}) = \phi(W_1\mathbf{x} + b_1)$
  - W_2 a matrix of dimensions K by D
  - b_2 a vector with D elements
  - model output: $f^{W_1,b_1,W_2,b_2}(\mathbf{x}) = \phi^{W_1,b_1}(\mathbf{x})W_2 + b_2$
- want to find W_1, b_1, W_2, b_2 that minimise $1/N \sum_i ||\mathbf{y}_i - f^{W_1,b_1,W_2,b_2}(\mathbf{x}_i)||^2$



Parametrised basis functions

Parametrised basis functions [Bishop, 2006; many others]

- eg basis functions $\phi_k^{w_k, b_k}$ where scalar-valued function ϕ_k is applied to inner-product $w_k^T \mathbf{x} + b_k$
  - ϕ_k often def'd to be identical for all k (only params change)
  - eg $\phi_k(\cdot) = \tanh(\cdot)$, giving $\phi_k^{w_k, b_k}(\mathbf{x}) = \tanh(w_k^T \mathbf{x} + b_k)$
- feature vector = basis functions' outputs = input to linear trans.
- in vector form:
  - W_1 a matrix of dimensions Q by K
  - b_1 a vector with K elements
  - $\phi^{W_1, b_1}(\mathbf{x}) = \phi(W_1 \mathbf{x} + b_1)$
  - W_2 a matrix of dimensions K by D
  - b_2 a vector with D elements
  - model output: $f^{W_1, b_1, W_2, b_2}(\mathbf{x}) = \phi^{W_1, b_1}(\mathbf{x}) W_2 + b_2$

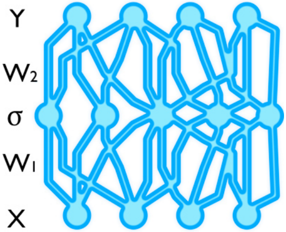


- want to find W_1, b_1, W_2, b_2 that minimise $1/N \sum_i ||\mathbf{y}_i - f^{W_1, b_1, W_2, b_2}(\mathbf{x}_i)||^2$

Parametrised basis functions

Parametrised basis functions [Bishop, 2006; many others]

- eg basis functions $\phi_k^{w_k,b_k}$ where scalar-valued function ϕ_k is applied to inner-product $w_k^T \mathbf{x} + b_k$
  - ϕ_k often def'd to be identical for all k (only params change)
  - eg $\phi_k(\cdot) = \tanh(\cdot)$, giving $\phi_k^{w_k,b_k}(\mathbf{x}) = \tanh(w_k^T \mathbf{x} + b_k)$
- feature vector = basis functions' outputs = input to linear trans.
- in vector form:
  - W_1 a matrix of dimensions Q by K
  - b_1 a vector with K elements
  - $\phi^{W_1,b_1}(\mathbf{x}) = \phi(W_1 \mathbf{x} + b_1)$
  - W_2 a matrix of dimensions K by D
  - b_2 a vector with D elements
  - model output: $f^{W_1,b_1,W_2,b_2}(\mathbf{x}) = \phi^{W_1,b_1}(\mathbf{x}) W_2 + b_2$
- want to find W_1, b_1, W_2, b_2 that minimise $1/N \sum_i ||\mathbf{y}_i - f^{W_1,b_1,W_2,b_2}(\mathbf{x}_i)||^2$



Hierarchy of parametrised basis functions

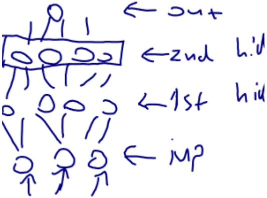
Hierarchy of parametrised basis functions [Rumelhart et al., 1985]

- called “NNs” for historical reasons

- **layers**
  - =‘feature vectors’ in hierarchy
  - linear trans. = ‘inner product’ layer = ‘fully connected’ layer
  - ‘input layer’, ‘output layer’, ‘hidden layers’
  - trans. matrix = weight matrix = W; intercept = bias = b

- **units**
  - elements in a layer

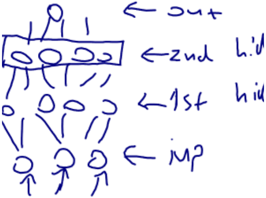
- **feature vector** (overloaded term)
  - often refers to the penultimate layer (at top of model just before softmax / last linear trans.)
  - denote feature vector



Hierarchy of parametrised basis functions

Hierarchy of parametrised basis functions [Rumelhart et al., 1985]

- called “NNs” for historical reasons
- **layers**
  - ='feature vectors' in hierarchy
  - linear trans. = 'inner product' layer = 'fully connected' layer
  - 'input layer', 'output layer', 'hidden layers'
  - trans. matrix = weight matrix = W; intercept = bias = b

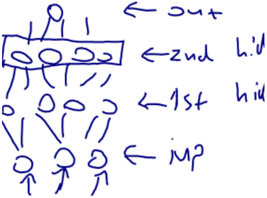


- **units**
  - elements in a layer
- **feature vector** (overloaded term)
  - often refers to the penultimate layer (at top of model just before softmax / last linear trans.)
  - denote feature vector

Hierarchy of parametrised basis functions

Hierarchy of parametrised basis functions [Rumelhart et al., 1985]

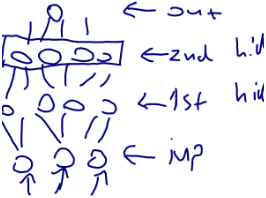
- called “NNs” for historical reasons
- **layers**
  - =‘feature vectors’ in hierarchy
  - linear trans. = ‘inner product’ layer = ‘fully connected’ layer
  - ‘input layer’, ‘output layer’, ‘hidden layers’
  - trans. matrix = weight matrix = W; intercept = bias = b
- **units**
  - elements in a layer
- **feature vector** (overloaded term)
  - often refers to the penultimate layer (at top of model just before softmax / last linear trans.)
  - denote feature vector



Hierarchy of parametrised basis functions

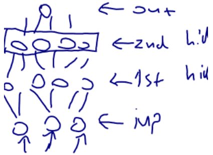
Hierarchy of parametrised basis functions [Rumelhart et al., 1985]

- called “NNs” for historical reasons
- **layers**
- **units**
- **feature vector** (overloaded term)
  - often refers to the penultimate layer (at top of model just before softmax / last linear trans.)
  - denote feature vector $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), .., \phi_K(\mathbf{x})]$ with K units (a K by 1 vector)
  - denote feature matrix $\Phi(\mathbf{X}) = [\phi(\mathbf{x}_1)^T, ..., \phi(\mathbf{x}_N)^T]$, N by K matrix



Hierarchy of parametrised basis functions

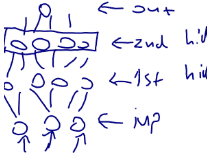
- **regression**
  - compose multiple basis function layers into a regression model
  - result of last trans. also called "model output"; often no non-linearity here



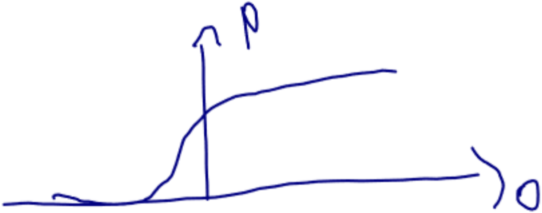
- **classification**
  - further compose a **softmax** function at the end; also called "logistic" for 2 classes
  - "squashes" its input → probability vector; prob vector also called model output / softmax vector / softmax layer
- "building blocks"
  - layers are simple
  - modularity in layer composition → versatility of deep models
  - many engineers work in field → lots of tools that scale well

Hierarchy of parametrised basis functions

- **regression**
  - compose multiple basis function layers into a regression model
  - result of last trans. also called "model output"; often no non-linearity here

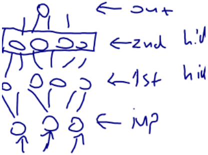


- **classification**
  - further compose a **softmax** function at the end; also called "logistic" for 2 classes
  - "squashes" its input → probability vector; prob vector also called model output / softmax vector / softmax layer



Hierarchy of parametrised basis functions

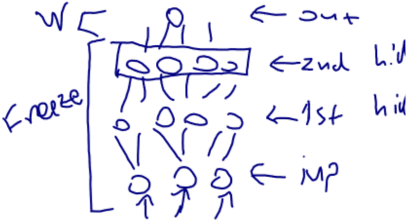
- **regression**
  - compose multiple basis function layers into a regression model
  - result of last trans. also called “model output”; often no non-linearity here



- **classification**
  - further compose a **softmax** function at the end; also called “logistic” for 2 classes
  - “squashes” its input → probability vector; prob vector also called model output / softmax vector / softmax layer
- “building blocks”
  - layers are simple
  - modularity in layer composition → versatility of deep models
  - many engineers work in field → lots of tools that scale well

Assumptions **for the moment**

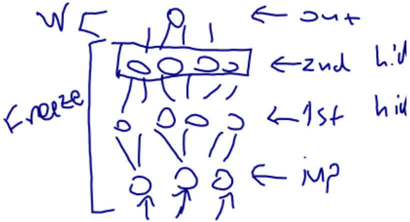
- we'll use deep nets, and denote W to be the weight matrix of the last layer and b the bias of last layer
- (for the moment) look only at last layer W, everything else fixed – ie weights other than W do not change
  - later we'll worry about other layers



- assume that y is scalar
  - so W is K by 1
  - write w_k for the k'th elem

Assumptions **for the moment**

- we'll use deep nets, and denote W to be the weight matrix of the last layer and b the bias of last layer
- (for the moment) look only at last layer W, everything else fixed – ie weights other than W do not change
  - later we'll worry about other layers



- assume that y is scalar
  - so W is K by 1
  - write w_k for the k'th elem

Assumptions **for the moment**

- we'll use deep nets, and denote W to be the weight matrix of the last layer and b the bias of last layer
- (for the moment) look only at last layer W, everything else fixed – ie weights other than W do not change
  - later we'll worry about other layers
- assume that y is scalar
  - so W is K by 1
  - write w_k for the k'th elem
- assume that output layer's b is zero (or, obs y's are normalised)
  - both will simplify derivations here (but pose no difficulty otherwise)
- then $f^W(\mathbf{x}) = \sum w_k \phi_k(\mathbf{x}) = W^T \phi(x)$ with $\phi(x)$ a 'frozen' feature vec for some NN
- some notation you'll need to remember...
  $\mathbf{X}, x, N, \mathbf{x}_n, Q, D, K, \mathcal{D} = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_N, y_N)\} = \mathbf{X}, \mathbf{Y}$

Assumptions **for the moment**

- we'll use deep nets, and denote W to be the weight matrix of the last layer and b the bias of last layer

- (for the moment) look only at last layer W, everything else fixed – ie weights other than W do not change
  - later we'll worry about other layers

- assume that y is scalar
  - so W is K by 1
  - write w_k for the k'th elem

- assume that output layer's b is zero (or, obs y's are normalised)
  - both will simplify derivations here (but pose no difficulty otherwise)

- then $f^W(\mathbf{x}) = \sum w_k \phi_k(\mathbf{x}) = W^T \phi(x)$ with $\phi(x)$ a 'frozen' feature vec for some NN

- some notation you'll need to remember...
  $\mathbf{X}, x, N, \mathbf{x}_n, Q, D, K, \mathcal{D} = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_N, y_N)\} = \mathbf{X}, \mathbf{Y}$

Assumptions **for the moment**

- we'll use deep nets, and denote W to be the weight matrix of the last layer and b the bias of last layer
- (for the moment) look only at last layer W, everything else fixed – ie weights other than W do not change
  - later we'll worry about other layers
- assume that y is scalar
  - so W is K by 1
  - write w_k for the k'th elem
- assume that output layer's b is zero (or, obs y's are normalised)
  - both will simplify derivations here (but pose no difficulty otherwise)
- then $f^W(\mathbf{x}) = \sum w_k \phi_k(\mathbf{x}) = W^T \phi(x)$ with $\phi(x)$ a 'frozen' feature vec for some NN
- some notation you'll need to remember...
  $\mathbf{X}, x, N, \mathbf{x}_n, Q, D, K, \mathcal{D} = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_N, y_N)\} = \mathbf{X}, \mathbf{Y}$

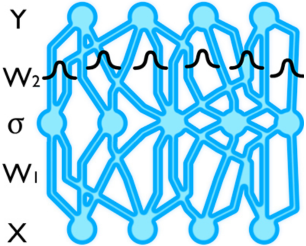
Assumptions **for the moment**

- we'll use deep nets, and denote W to be the weight matrix of the last layer and b the bias of last layer
- (for the moment) look only at last layer W, everything else fixed – ie weights other than W do not change
  - later we'll worry about other layers
- assume that y is scalar
  - so W is K by 1
  - write w_k for the k'th elem
- assume that output layer's b is zero (or, obs y's are normalised)
  - both will simplify derivations here (but pose no difficulty otherwise)
- then $f^W(\mathbf{x}) = \sum w_k \phi_k(\mathbf{x}) = W^T \phi(x)$ with $\phi(x)$ a 'frozen' feature vec for some NN
- some notation you'll need to remember...
  $\mathbf{X}, x, N, \mathbf{x}_n, Q, D, K, \mathcal{D} = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_N, y_N)\} = \mathbf{X}, \mathbf{Y}$

Generative story

Want to put dist over functions..

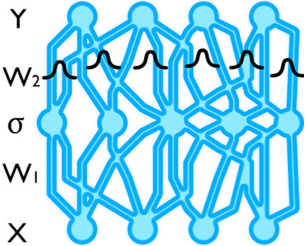
- difficult to put belief over funcs., but easy to put over NN params
- assumptions for the moment: our data was generated from the fixed ϕ (NN) using *some* W (which we want to infer)



Generative story

Want to put dist over functions..

- difficult to put belief over funcs., but easy to put over NN params
- assumptions for the moment: our data was generated from the fixed ϕ (NN) using *some* W (which we want to infer)



Generative story [what we assume about the data]

- Nature chose W which def's a func: $f^W(x) := W^T\phi(x)$
- generated **func. values** with inputs $x_1, .., x_N$: $f_n := f^W(x_n)$
- corrupted func. values with noise [also called "obs noise"] $y_n := f_n + \epsilon_n,\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$ [additive Gaussian noise w param σ]
- we're given **observations** $\{(x_1, y_1), ..., (x_N, y_N)\}$ and $\sigma = 1$

Model

- qs
  - how can we find function value f^* for a new x^*?
  - how can we find our confidence in this prediction?
  - $\rightarrow$ 'everything follows from the laws of probability theory'

- we build a **model**:
  - put prior dist over params W

$$p(W) = \mathcal{N}(W; 0_K, s^2 I_K)$$

  - likelihood [conditioned on W generate obs by adding gaussian noise]

$$p(y|W, x) = \mathcal{N}(y; W^T \phi(x), \sigma^2)$$

- prior belief "w_k is more likely to be in interval $[-1, 1]$ than in $[100, 200]$" means that the **func. values** are likely to be more smooth than erratic (we'll see later why)

- we want to infer W (find dist over W given $\mathcal{D}$)

Model

- qs
  - how can we find function value f^* for a new x^*?
  - how can we find our confidence in this prediction?
  - $\rightarrow$ ‘everything follows from the laws of probability theory’

- we build a **model**:
  - put prior dist over params W

$$p(W) = \mathcal{N}(W; 0_K, s^2 I_K)$$

  - likelihood [conditioned on W generate obs by adding gaussian noise]

$$p(y|W, x) = \mathcal{N}(y; W^T \phi(x), \sigma^2)$$

- prior belief “w_K is more likely to be in interval $[-1, 1]$ than in $[100, 200]$” means that the **func. values** are likely to be more smooth than erratic (we’ll see later why)

- we want to infer W (find dist over W given $\mathcal{D}$)

Model

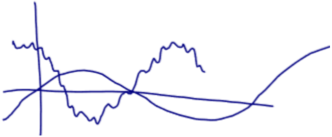
- qs
  - how can we find function value f^* for a new x^*?
  - how can we find our confidence in this prediction?
  - $\rightarrow$ 'everything follows from the laws of probability theory'
- we build a **model**:
  - put prior dist over params W

$$p(W) = \mathcal{N}(W; 0_K, s^2 I_K)$$

  - likelihood [conditioned on W generate obs by adding gaussian noise]

$$p(y|W, x) = \mathcal{N}(y; W^T \phi(x), \sigma^2)$$

- prior belief "w_k is more likely to be in interval $[-1, 1]$ than in $[100, 200]$" means that the **func. values** are likely to be more smooth than erratic (we'll see later why)



Model

- qs
  - how can we find function value f^* for a new x^*?
  - how can we find our confidence in this prediction?
  - $\rightarrow$ 'everything follows from the laws of probability theory'
- we build a **model**:
  - put prior dist over params W

$$p(W) = \mathcal{N}(W; 0_K, s^2 I_K)$$

  - likelihood [conditioned on W generate obs by adding gaussian noise]

$$p(y|W,x) = \mathcal{N}(y; W^T\phi(x), \sigma^2)$$

- prior belief "w_k is more likely to be in interval $[-1,1]$ than in $[100,200]$" means that the **func. values** are likely to be more smooth than erratic (we'll see later why)
- we want to infer W (find dist over W given $\mathcal{D}$)

Analytic inference w functions [new technique!]

$$p(w \mid x, y)$$

$$= p(y \mid x, w)\, \overset{p(w)}{p(w \mid \underline{x})} \,/\, \underline{p(y \mid x)}$$

Analytic inference w functions [new technique!]

$$p(w \mid x, y)$$

$$= p(y \mid x, w)\, \overset{p(w)}{p(w \mid \underline{x})} \,/\, \underline{p(y \mid x)}$$

only terms dep w

$$\propto p(y \mid x, w)\, p(w)$$

Analytic inference w functions [new technique!]

$$p(w|x,y)$$

$$= p(y|x,w)\,\overset{p(w)}{p(w|\underline{\underline{x}})}\,/\,\underline{\underline{p(y|x)}}$$

$$\underset{\substack{\text{dep}\\ w}}{\overset{\text{only terms}}{\propto}} p(y|x,w)\,p(w)$$

$$= \left[\prod_n \mathcal{N}(y_n\,;\,w^T\phi(x_n), \sigma^2)\right]\mathcal{N}(w; 0, s^2 I)$$

Analytic inference w functions [new technique!]

$$p(w \mid x, y)$$

$$= p(y \mid x, w)\, \overset{p(w)}{p(w \mid \underline{\underline{x}})} \;/\; \underline{\underline{p(y \mid x)}}$$

only terms dep w

$$\propto p(y \mid x, w)\, p(w)$$

$$= \left[\prod_n \mathcal{N}(y_n ; w^T \phi(x_n), \sigma^2)\right] \mathcal{N}(w; 0, s^2 I)$$

$$= \prod_n \underline{c_n}\, e^{-\frac{1}{2}\sigma^{-2}(y_n - w^T\phi(x))^2} \cdot \underline{c}\; e^{-\frac{1}{2}s^{-2} w^T w}$$

Analytic inference w functions [new technique!]

$$p(w|x,y)$$

$$= p(y|x,w)\, \overset{p(w)}{p(w|\underline{\underline{x}})} \,/\, \underline{\underline{p(y|x)}}$$

only terms dep w

$$\propto p(y|x,w)\,p(w)$$

$$= \left[\prod_n \mathcal{N}(y_n ; w^T\phi(x_n), \sigma^2)\right] \mathcal{N}(w; 0, s^2 I)$$

$$= \prod_n \underline{c_n}\, e^{-\frac{1}{2}\sigma^{-2}(y_n - w^T\phi(x))^2} \cdot \underline{c}\; e^{-\frac{1}{2}s^{-2}w^Tw}$$

$$\propto e\left\{-\frac{1}{2}\overbrace{\sum_n \sigma^{-2}(y_n - w^T\phi(x_n))^2 + s^{-2}w^Tw}\right\}$$

Analytic inference w functions [new technique!]

$$p(w|x,y)$$

$$= p(y|x,w)\,\overset{p(w)}{p(w|\underline{x})}\,/\,\underline{p(y|x)}$$

only terms dep w

$$\propto p(y|x,w)\,p(w)$$

$$= \left[\prod_n \mathcal{N}(y_n\,;\,w^T\phi(x_n),\,\sigma^2)\right]\mathcal{N}(w;0,\,s^2 I)$$

$$= \prod_n \underline{c_n}\, e^{-\frac{1}{2}\sigma^{-2}(y_n - w^T\phi(x))^2} \cdot \underline{c}\; e^{-\frac{1}{2}s^{-2}w^Tw}$$

$$\propto e\{-\tfrac{1}{2}\overbrace{\sum_n \sigma^{-2}(y_n - w^T\phi(x_n))^2 + s^{-2}w^Tw}\}$$

"w" terms: $\sum_n \sigma^{-2}(-2\,y_n w^T\phi(x_n) + \underbrace{(w^T\phi(x_n))}_{= \phi^T(x_n)w = w^T\phi(x_n)}{}^T(w^T\phi(x_n))$

$+ s^{-2}w^Tw$

Analytic inference w functions [new technique!]

$$= \sigma^{-2} \sum_n \left(-2\, y_n\, \underline{w}^T \phi(x_n) \right)$$

$$+ \sigma^{-2} \sum_n \underline{w}^T \phi(x_n)\, \phi(x_n)^T \underline{w}$$

$$+ s^{-2}\, \underline{w}^T \underline{w}$$

Analytic inference w functions [new technique!]

$$= \sigma^{-2} \sum_n \left(-2 y_n \underline{w}^T \phi(x_n)\right)$$

$$+ \sigma^{-2} \sum_n \underline{w}^T \phi(x_n) \phi(x_n)^T \underline{w}$$

$$+ s^{-2} \underline{w}^T \underline{w}$$

$$= \underline{w}^T \boxed{\left(\sigma^{-2} \sum_n \phi(x_n) \phi(x_n)^T + s^{-2} I_k\right)} \underline{w}$$

$$-2 w^T \underline{\sigma^{-2} \sum_n y_n \phi(x_n)}$$

Analytic inference w functions [new technique!]

$$= \sigma^{-2} \sum_n \left(-2 y_n \underline{w}^T \phi(x_n)\right)$$

$$+ \sigma^{-2} \sum_n \underline{w}^T \phi(x_n) \phi(x_n)^T \underline{w}$$

$$+ s^{-2} \underline{w}^T \underline{w}$$

$$= \underline{w}^T \boxed{\left(\sigma^{-2} \sum_n \phi(x_n) \phi(x_n)^T + s^{-2} I_k\right)} \underline{w}$$

$$\underline{-2 w^T \sigma^{-2} \sum_n y_n \phi(x_n)}$$

Know $\boxed{p(w|D) := \mathcal{N}(\mu', \Sigma')}$

$$(w-\mu')^T \Sigma'^{-1} (w-\mu') = \underline{w}^T \boxed{\Sigma'^{-1}} \underline{w} - 2 w^T \underline{\Sigma'^{-1} \mu'} + \mu'^T \Sigma'^{-1} \mu'$$

Analytic inference w functions [new technique!]

$$= \underline{w}^T \boxed{\left(\sigma^{-2} \sum_n \phi(x_n)\phi(x_n)^T + s^{-2} I_k\right)} \underline{w}$$

$$\underline{-2\, w^T \sigma^{-2} \sum_n y_n \phi(x_n)}$$

Know $\boxed{p(w|D) := \mathcal{N}(\mu', \Sigma')}$

$$(w-\mu')^T \Sigma'^{-1} (w-\mu') = \underline{w}^T \boxed{\Sigma'^{-1}} \underline{w} - 2 w^T \underline{\Sigma'^{-1} \mu'} + \mu'^T \Sigma'^{-1} \mu'$$

- posterior variance

$$\Sigma' = (\sigma^{-2} \sum_n (\phi(x_n)\phi(x_n)^T) + s^{-2} I_K)^{-1}$$

and in vector form: $(\sigma^{-2}\Phi(\mathbf{X})^T\Phi(\mathbf{X}) + s^{-2} I_K)^{-1}$

- posterior mean

$$\mu' = \Sigma'\sigma^{-2} \sum_n (y_n \phi(x_n))$$

and in vector form: $\Sigma'\sigma^{-2}\Phi(\mathbf{X})^T\mathbf{Y}$

Analytic predictions with functions

How do we predict function values y^* for new x^*?

- use prob theory to perform preds!

$$
\begin{aligned}
& p(y^*|x^*, X, Y) \\
&= \int p(y^*, W|x^*, X, Y)\mathrm{d}W \qquad \text{sum rule} \\
&= \int p(y^*|x^*, W, X, Y)p(W|X, Y)\mathrm{d}W \qquad \text{product rule} \\
&= \int p(y^*|x^*, W)p(W|X, Y)\mathrm{d}W \qquad \text{model assumptions}
\end{aligned}
$$

- how to eval? [a new technique!]
  - likelihood $p(y^*|x^*, W)$ is Gaussian
  - posterior $p(W|X, Y)$ is Gaussian (from above)
  - so *predictive* $p(y^*|x^*, X, Y)$ is Gaussian..

Analytic predictions with functions

$$p(y^* | x^*, D) = \mathcal{N}(\mu^*, \Sigma^*)$$

$$\mu^* = E_{p(y^*|x^*, D)}\{y^*\}$$

Analytic predictions with functions

$$p(y^* | x^*, D) = \mathcal{N}(\mu^*, \Sigma^*)$$

$$\mu^* = E_{p(y^* | x^*, D)}[y^*]$$

$$= \int p(y^* | x^*, D) \cdot y^* \, dy^*$$

Analytic predictions with functions

$$p(y^* \mid x^*, D) = \mathcal{N}(\mu^*, \Sigma^*)$$

$$\mu^* = E_{p(y^* \mid x^*, D)}\{y^*\}$$

$$= \int p(y^* \mid x^*, D) \cdot y^* \, dy^*$$

$$= \int y^* \left(\int p(y^* \mid x^*, w) \, p(w \mid D) \, dw \right) dy^*$$

Analytic predictions with functions

$$p(y^*|x^*, D) = \mathcal{N}(\mu^*, \Sigma^*)$$

$$\mu^* = E_{p(y^*|x^*, D)}\{y^*\}$$

$$= \int p(y^*|x^*, D) \cdot y^* \, dy^*$$

$$= \int y^* \left(\int p(y^*|x^*, w)\, p(w|D)\, dw \right) dy^*$$

$$= \int \left(\underbrace{\int y^* p(y^*|x^*, w)\, dy^*}_{E_{lik}(y^*) = w^T \phi(x^*)} \right) p(w|D)\, dw$$

Analytic predictions with functions

$$p(y^* | x^*, D) = \mathcal{N}(\mu^*, \Sigma^*)$$

$$\mu^* = E_{p(y^*|x^*, D)}\{y^*\}$$

$$= \int p(y^*|x^*, D) \cdot y^* \, dy^*$$

$$= \int y^* \left(\int p(y^*|x^*, w)\, p(w|D) \, dw \right) dy^*$$

$$= \int \Big(\underbrace{\int y^* p(y^*|x^*, w) \, dy^*}_{E_{lik}(y^*) = w^T \phi(x^*)} \Big) p(w|D) \, dw$$

$$= \underbrace{\int w^T p(w|D) \, dw}_{E_{post}[w]^T = \mu'^T} \phi(x^*) = \mu'^T \phi(x^*)$$

Analytic predictions with functions

$$\mu^* = E_{p(y^*|x^*,D)}[y^*]$$

$$= \int p(y^*|x^*,D)\cdot y^* \, dy^*$$

$$= \int y^* \left(\int p(y^*|x^*,w)\, p(w|D)\, dw\right) dy^*$$

$$= \int \left(\underbrace{\int y^* p(y^*|x^*,w)\, dy^*}_{E_{lik}(y^*) = w^T\phi(x^*)}\right) p(w|D)\, dw$$

$$= \underbrace{\int w^T p(w|D)\, dw}_{E_{post}[w]^T = \mu'^T}\, \phi(x^*) = \mu'^T \phi(x^*)$$

- Homework: Predictive variance

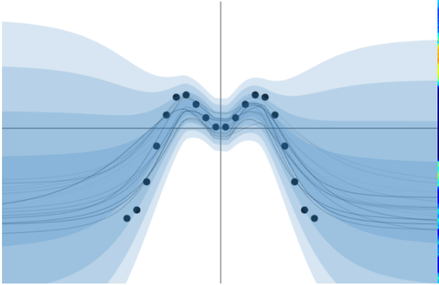
What you should be able to do now

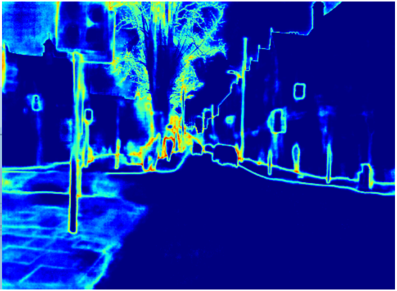
- perform density estimation with scalars
- know when MLE fails (and why)
- use Bayes law to make more informed decisions in your life
- win against your friends in a series of bets
- argue with frequentists about how to interpret the laws of probability
- argue with philosophers about the nature of subjective beliefs
- use Bayesian probability in ML **correctly**
- perform predictions in Bayesian probabilistic modelling **correctly**

What we will cover next

In the next lecture we'll

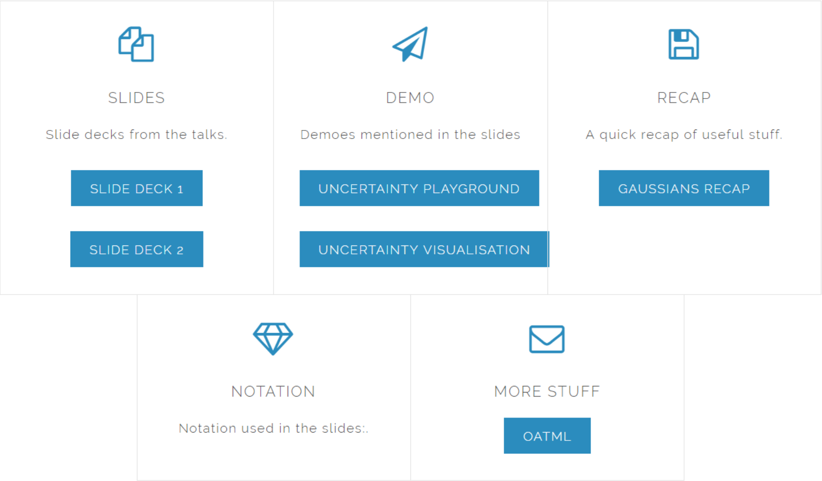
- decompose uncertainty into epistemic and aleatoric components
- use uncertainty in regression **correctly**
- develop tools to scale the ideas above to large deep models
- develop **big deep learning systems** which convey uncertainty
  - w **real-world** examples





Bayesian deep learning

All resources (including these slides): bdl101.ml



SLIDES

Slide decks from the talks.

DEMO

Demoes mentioned in the slides

RECAP

A quick recap of useful stuff.

NOTATION

Notation used in the slides:.

MORE STUFF