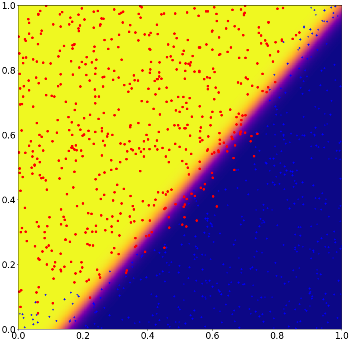
Machine Learning 1.09: Regularisation & Model Types

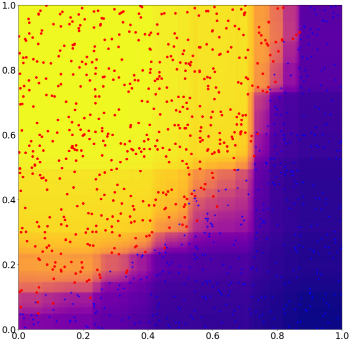
Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

UNIVERSITY OF BATH

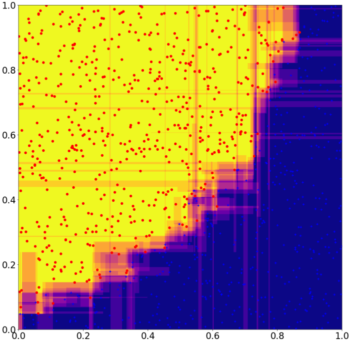
Underfitting & Overfitting



- Underfitting
- Logistic regression



- Balanced
- Tuned random forest
- (scikit learn, `min_impurity_decrease=0.008`, `n_estimators=512`)



- Overfitting
- Badly tuned random forest
- (scikit learn, default parameters)

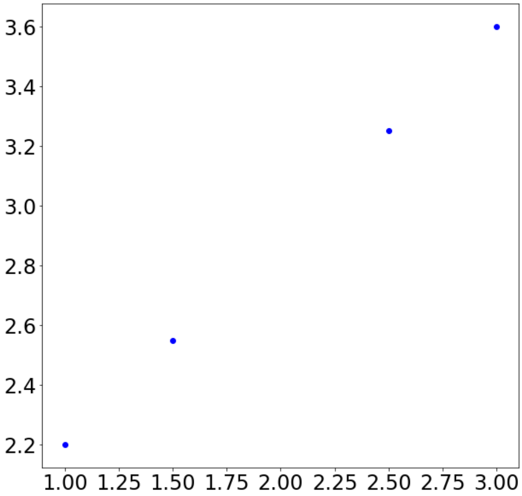
- Fixes overfitting. How?

Regularisation

- Fixes overfitting. How?
- Introduces "extra information"

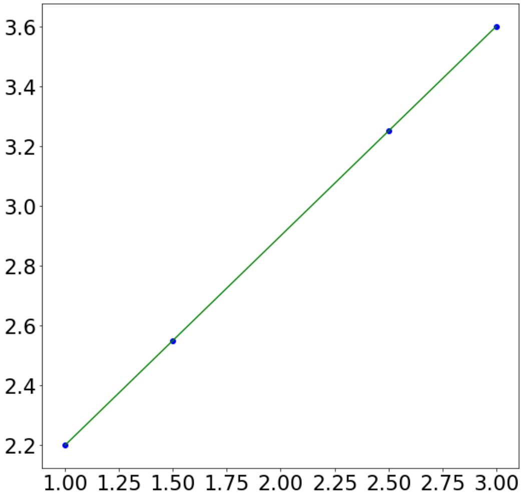
Extra Information

- Simple regression



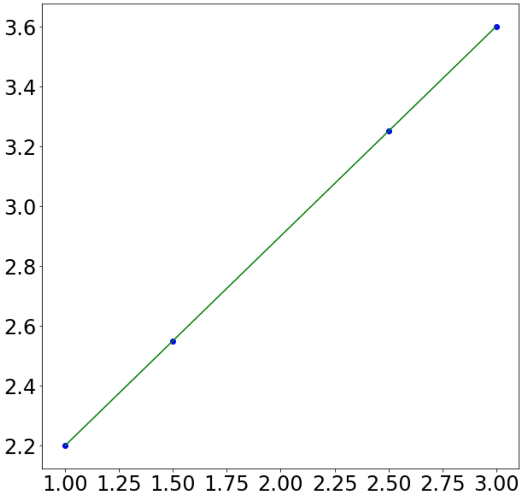
Extra Information

- Simple regression
- Linear solution is obvious – to us!



Extra Information

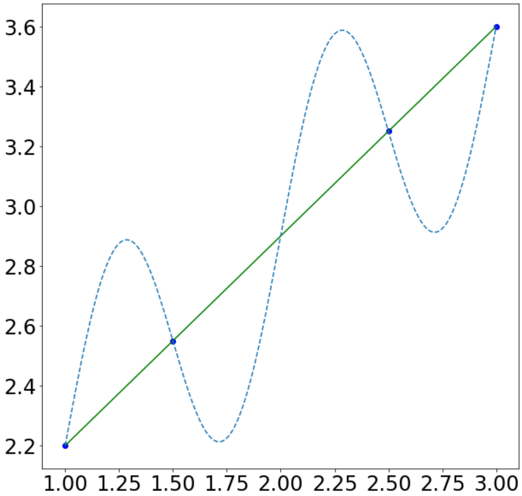
- Simple regression
- Linear solution is obvious – to us!
- Imagine model is general – any function is allowed



Extra Information

- Simple regression
- Linear solution is obvious – to us!

- Imagine model is general – any function is allowed
- Could match a sine curve

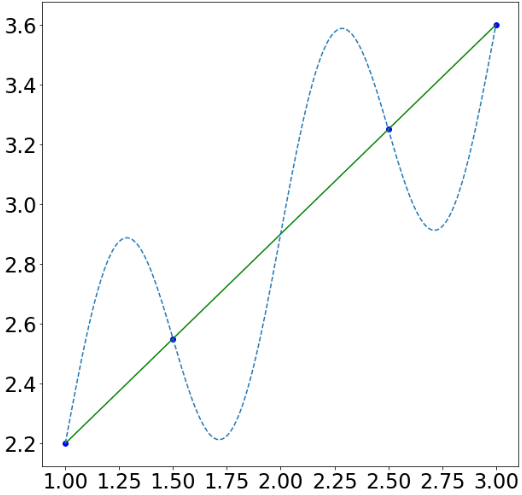


Extra Information

- Simple regression
- Linear solution is obvious – to us!

- Imagine model is general – any function is allowed
- Could match a sine curve

- Still a perfect match at known points
- Model sees this as **identical to a straight line**!

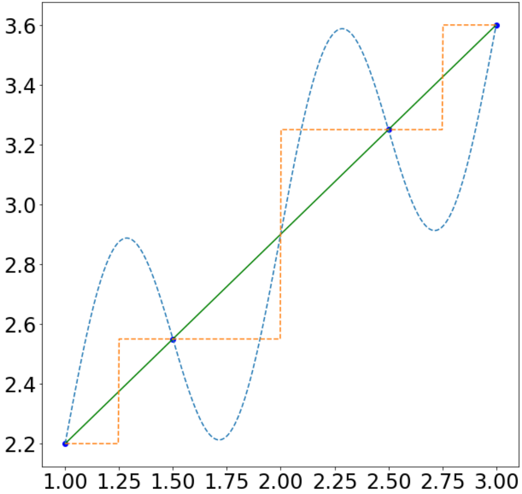


Extra Information

- Simple regression
- Linear solution is obvious – to us!

- Imagine model is general – any function is allowed
- Could match a sine curve

- Still a perfect match at known points
- Model sees this as **identical to a straight line**!

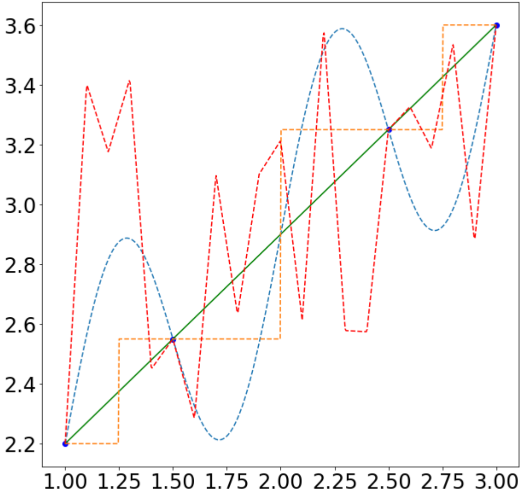


Extra Information

- Simple regression
- Linear solution is obvious – to us!

- Imagine model is general – any function is allowed
- Could match a sine curve

- Still a perfect match at known points
- Model sees this as **identical to a straight line**!



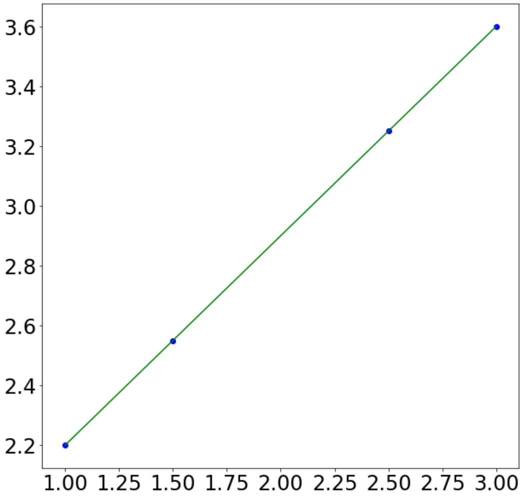
Extra Information

- Simple regression
- Linear solution is obvious – to us!

- Imagine model is general – any function is allowed
- Could match a sine curve

- Still a perfect match at known points
- Model sees this as **identical to a straight line**!

- Regularisation makes it choose the straight line – it encodes **common sense** (for a mathematician)



Occam's Razor

The simplest explanation is usually the correct one

- Idea can be traced back to Aristotle (384–322 BC)
- Ockham's version: "Plurality must never be posited without necessity" (translated from Latin – William of Ockham was a 13th century priest)

- Overfitting is proposing an unjustifiably complex explanation.

Reasons for Regularisation

Avoiding overfitting is one. There are others (first is a specialisation):

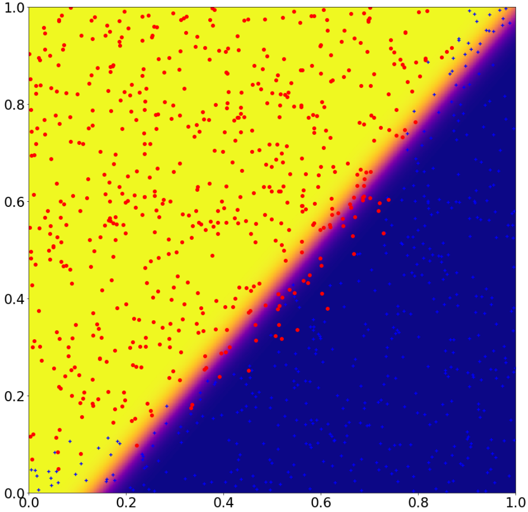
- Solving ill posed problems

- Extra knowledge
- Human understanding
- Easier optimisation

Often several of these at once

Limits

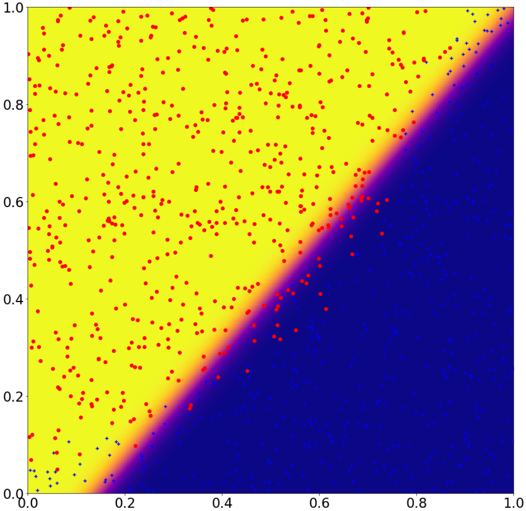
- Limits of model can be seen as regularisation, e.g. Logistic regression can only fit a straight line



Limits

- Limits of model can be seen as regularisation, e.g. Logistic regression can only fit a straight line

- Often too restrictive
- A practical trade off
- Mismatch between data and regularisation common



Quantity of Data

- More data $\implies$ less regularisation.
- Infinite data $\implies$ no regularisation! (simple lookup)

Quantity of Data

- More data $\implies$ less regularisation.
- Infinite data $\implies$ no regularisation! (simple lookup)

- Hyperparameters usually control regularisation strength
- Significant part of hyperparameter optimisation is tuning model to data quantity

Quantity of Data

- More data $\implies$ less regularisation.
- Infinite data $\implies$ no regularisation! (simple lookup)

- Hyperparameters usually control regularisation strength
- Significant part of hyperparameter optimisation is tuning model to data quantity

- Models can have
  - Lower limit, below which they fail
  - Upper limit, above which they stop improving (underfitting)

Non-probabilistic Regularisation

- Model fitting is minimises cost function $C(.)$, by adjusting parameters p, for a function $F(.)$, e.g.

$$C(p) = \sum_{i=1}^{n} (y_i - F(x_i, p))^2$$

Non-probabilistic Regularisation

- Model fitting is minimises cost function $C(.)$, by adjusting parameters p, for a function $F(.)$, e.g.

$$C(p) = \sum_{i=1}^{n} (y_i - F(x_i, p))^2$$

- Regularise by including a term to (for example) encourage small parameters:

$$C(p) = \sum_{i=1}^{n} |y_i - F(x_i, p)| + \lambda |p|$$

- Often equivalent to applying a prior – see later.

Probabilistic Regularisation

Same justifications, but more formal. Three choices:

1. Maximum likelihood (ML)
2. Maximum a posteriori (MAP)
3. Bayesian

1. Maximum Likelihood

- Find model parameters that maximise data probability

- No regularisation!
- Requires enough data to work

Linear Regression: ML I

For each exemplar:

$$y_i = ax_i + b + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

$N(\texttt{mean}, \texttt{standard deviation}^2)$ is the Normal distribution.

Linear Regression: ML I

For each exemplar:

$$y_i = ax_i + b + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

$N(\texttt{mean}, \texttt{standard deviation}^2)$ is the Normal distribution.
Exemplar probability:

$$P(y_i | x_i, a, b, \sigma) \propto \frac{1}{\sigma} \exp\left(\frac{-(ax_i + b - y_i)^2}{2\sigma^2}\right)$$

UNIVERSITY OF BATH

Linear Regression: ML I

For each exemplar:

$$y_i = ax_i + b + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

$N(\texttt{mean}, \texttt{standard deviation}^2)$ is the Normal distribution.
Exemplar probability:

$$P(y_i|x_i, a, b, \sigma) \propto \frac{1}{\sigma} \exp\left(\frac{-(ax_i + b - y_i)^2}{2\sigma^2}\right)$$

Maximum likelihood solution:

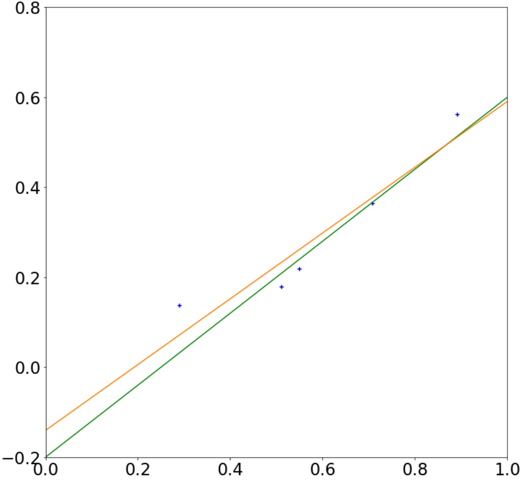
$$[a, b]^T = (X^T X)^{-1} X^T y$$

where

$$X = [[x_1, 1], [x_2, 1], \ldots, [x_n, 1]] \qquad y = [y_1, y_2, \ldots, y_n]^T$$

Given above know ϵ_i $\therefore$ σ is mean of $|\epsilon_i|$

Linear Regression: ML II



green = ground truth, orange = estimate

2. Maximum a posteriori

- Introduce a **prior** over every model parameter
- prior = probability distribution

- Find maximum likelihood solution (again), including prior

- Model now complete – can generate answers **without** data!
- Works however much data you give it.

Linear Regression: MAP I

For each exemplar:

$$y_i = ax_i + b + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

but add priors (one choice among many):

$$a, b \sim N(\mu_0, \Sigma_0), \qquad \sigma^2 \sim \texttt{Inv-Gamma}(\alpha_0, \beta_0)$$

where μ_0, Σ_0, α_0 and β_0 are hyper-parameters.

Linear Regression: MAP I

For each exemplar:

$$y_i = ax_i + b + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

but add priors (one choice among many):

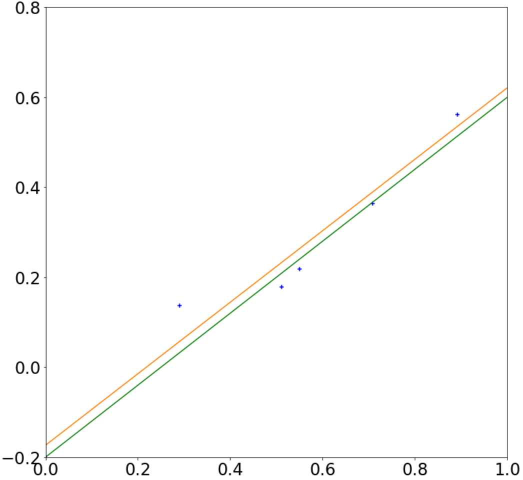
$$a, b \sim N(\mu_0, \Sigma_0), \qquad \sigma^2 \sim \texttt{Inv-Gamma}(\alpha_0, \beta_0)$$

where μ_0, Σ_0, α_0 and β_0 are hyper-parameters.
Answer:

$$[a, b]^T = (X^T X + \Sigma_0^{-1})^{-1}(\Sigma_0^{-1}\mu_0 + X^T y)$$

with same definitions of X and y as before.
Ignoring σ as complicated.

Linear Regression: MAP II



green = ground truth, orange = estimate

3. Bayesian

- Same as MAP, with priors.
- Instead of maximum likelihood solution find **posterior distribution**.

$$P(\texttt{model parameters}|\texttt{data}, \texttt{labels})$$

3. Bayesian

- Same as MAP, with priors.
- Instead of maximum likelihood solution find **posterior distribution**.

$$P(\texttt{model parameters}|\texttt{data}, \texttt{labels})$$

- Has benefits of MAP.
- Plus a **distribution** over models – it knows how certain it is!
- Occam's razor is built in.

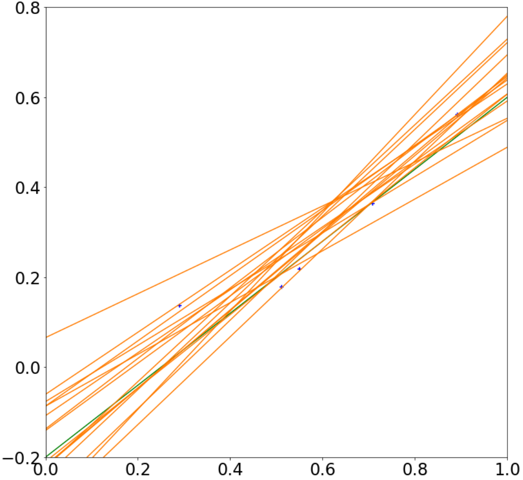
Linear Regression: Bayesian I

Same formulation as MAP.
Answer:

$$[a, b]^T \sim N(\mu_n, \Sigma_n)$$
$$\mu_n = (X^T X + \Sigma_0^{-1})^{-1}(\Sigma_0^{-1}\mu_0 + X^T y)$$
$$\Sigma_n = \sigma^2(X^T X + \Sigma_0^{-1})^{-1}$$

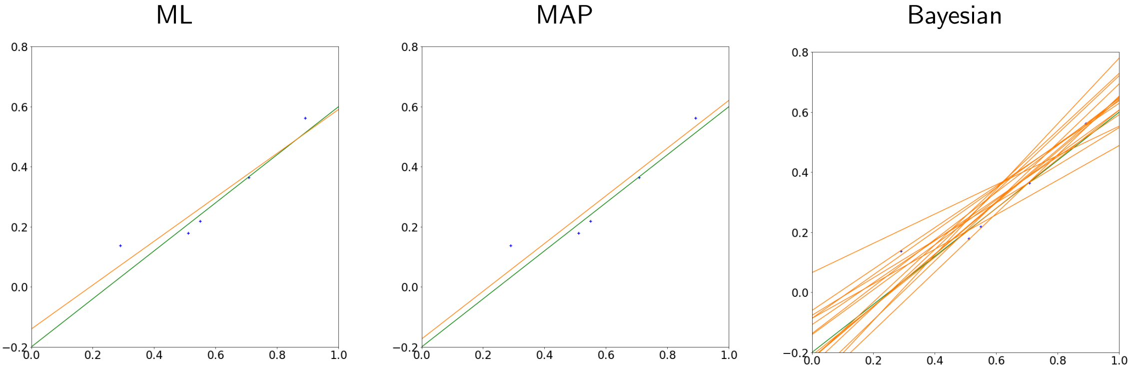
Note: Dependent on σ, which has not been given.

Linear Regression: Bayesian II



green = ground truth, orange = draws from estimate

Comparison



- Given enough data they will all give the same answer
- Not enough:
  - Maximum likelihood fails
  - Maximum a posteriori gives a solution
  - Bayesian gives a solution and tells you how confident it is

Should all models be Bayesian?

- In an ideal world, yes!

Should all models be Bayesian?

- In an ideal world, yes!
- But...
  - Often harder to code and optimise
  - Often slower
  - Good prior problem...

Prior: How do they work?

- Purpose is to regularise – bias towards simple solutions.

Prior: How do they work?

- Purpose is to regularise – bias towards simple solutions.

- Do so by indicating which model parameters are likely, which unlikely
- Assumption that the model is sensible – that you can reason about its parameters
- e.g. a chaotic simulation would be almost impossible to set a prior for

Prior: Types

- Uninformative
- Improper
- Minimum description length

- Extra knowledge
- Data driven (dodgy)
- Human belief

Prior: Conjugate

- A prior that allows an analytic solution
- Choice of Gaussian and inverse Gamma for linear regression made answer analytic

Prior: Conjugate

- A prior that allows an analytic solution
- Choice of Gaussian and inverse Gamma for linear regression made answer analytic

- Problem: Conjugate priors are simple, bad match to data
- Bayesian methods often under perform due to using simple priors

Stupid Approach: Early Stopping

- Model starts simple, gets more convoluted as optimisation runs
- So stop early!

Stupid Approach: Early Stopping

- Model starts simple, gets more convoluted as optimisation runs
- So stop early!

- What does that even mean?
- Your regularisation is too weak - make it stronger!

Model Types

x – Data $\qquad$ y – Label

Model Types

x – Data $\qquad$ y – Label

Discriminative $\qquad$ Generative

Model Types

x – Data $\qquad$ y – Label

Discriminative

- Learns $P(y|x)$

Generative

- Learns $P(y, x)$

Model Types

x – Data $\qquad$ y – Label

Discriminative

- Learns $P(y|x)$
- Used directly

Generative

- Learns $P(y,x)$
- Apply Bayes rules: $P(y|x) = \frac{P(y,x)}{P(x)}$
- Often actually $P(x|y)$ and $P(y)$

Model Types

x – Data $\qquad$ y – Label

Discriminative

- Learns $P(y|x)$
- Used directly

- Learns boundary between data
  (no requirement to be probabilistic)

Generative

- Learns $P(y, x)$
- Apply Bayes rules: $P(y|x) = \frac{P(y,x)}{P(x)}$
- Often actually $P(x|y)$ and $P(y)$

- Learns distribution of data
  (must be probabilistic)

UNIVERSITY OF BATH

Model Types

x – Data $\quad$ y – Label

Discriminative

- Learns $P(y|x)$
- Used directly

- Learns boundary between data
  (no requirement to be probabilistic)
- Can only discriminate between classes

Generative

- Learns $P(y, x)$
- Apply Bayes rules: $P(y|x) = \frac{P(y,x)}{P(x)}$
- Often actually $P(x|y)$ and $P(y)$

- Learns distribution of data
  (must be probabilistic)
- Can also generate data

Model Types

x – Data $\quad$ y – Label

Discriminative

- Learns $P(y|x)$
- Used directly

- Learns boundary between data
  (no requirement to be probabilistic)
- Can only discriminate between classes

Generative

- Learns $P(y,x)$
- Apply Bayes rules: $P(y|x) = \frac{P(y,x)}{P(x)}$
- Often actually $P(x|y)$ and $P(y)$

- Learns distribution of data
  (must be probabilistic)
- Can also generate data
- Handle missing data
- Less vulnerable to overfitting
- Know when they are unreliable

Should all models be Generative?

- In an ideal world, yes!

Should all models be Generative?

- In an ideal world, yes!
- But. . .
    - Often harder to code and optimise
    - Often slower
    - Discriminative approaches often “win”. . .

Discriminative vs Generative

- If winning means highest accuracy then they keep switching places

Discriminative vs Generative

- If winning means highest accuracy then they keep switching places
- Currently, discriminative is winning...
  ...but can already see generative successors
  (GANs, Auto-encoders)

Summary

- Regularisation embodies common sense

- Models can be probabilistic or not
- Probabilistic models have three main approaches (there are others!)
- Models can be discriminative or generative

- Generative Bayesian models are the gold standard

Further Reading

- Chapter 28, of “Information Theory, Inference, and Learning Algorithms” by MacKay.
- Maths for linear regression variants:
  `https://en.wikipedia.org/wiki/Bayesian_linear_regression`