Statistical modeling: the two cultures

Leo Breiman

Statistical Science, 2001

There are two cultures in the use of statistical modeling to reach conclusions from data.

- One assumes that the data are generated by a given [probabilistic] data model.
- The other uses algorithmic models and treats the data mechanism as unknown.

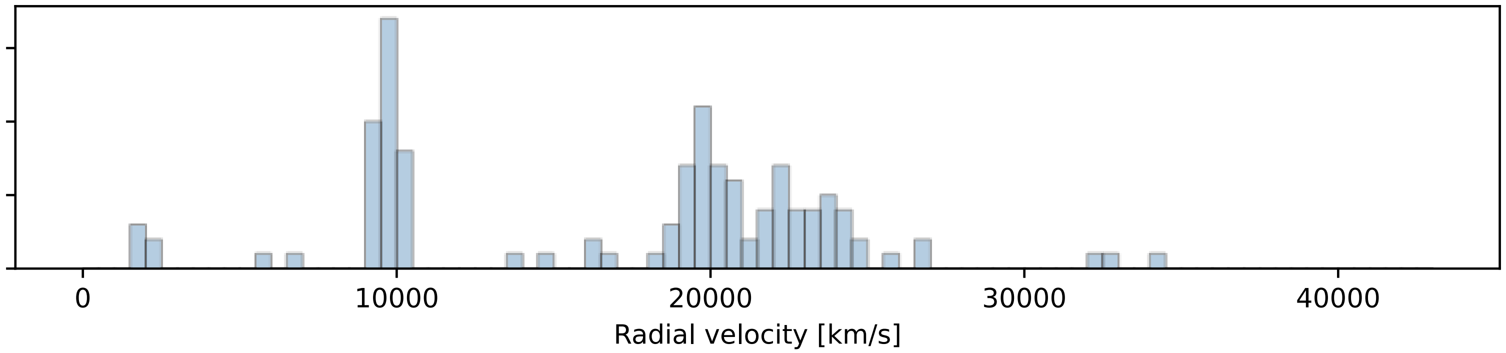
The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems.

In the mid-1980s two powerful new algorithms for fitting data became available: neural nets and decision trees. A new research community using these tools sprang up. Their goal was predictive accuracy. The community consisted of young computer scientists, physicists and engineers plus a few aging statisticians. They began using the new tools in working on complex prediction problems where it was obvious that data models were not applicable: speech recognition, image recognition, nonlinear time series prediction, handwriting recognition, prediction in financial markets.

Speeds of galaxies in the Corona Borealis region

Postman, Huchra, Geller (1986)

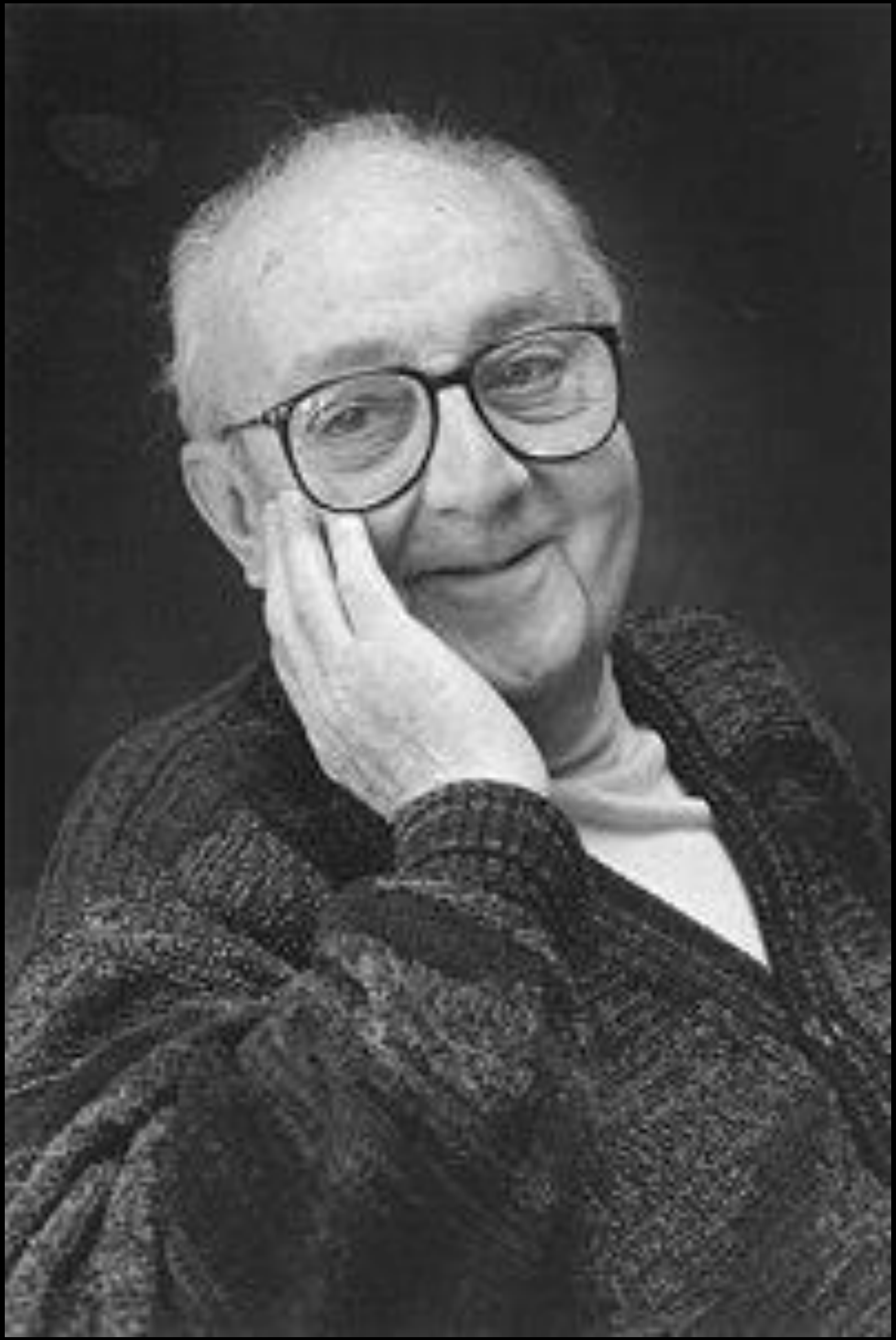
A histogram of radial velocities of 120 galaxies



How might you complete this code?

```
def rgalaxy(...):
    # TODO: return a single random galaxy speed

def rgalaxies(size):
    return [rgalaxy(…) for _ in range(size)]
```



George Box

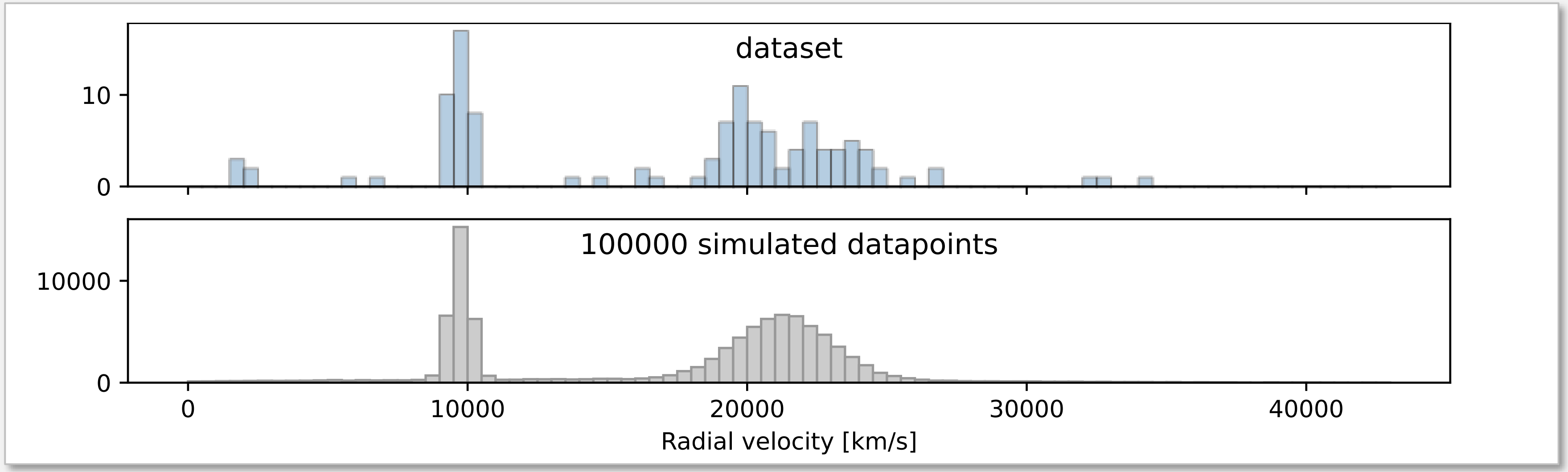
1919–2013

"All models are wrong,
but some are useful"

so, don't get hung up about coming up with the "right" model – just charge ahead and invent something!

Speeds of galaxies in the Corona Borealis region

Postman, Huchra, Geller (1986)




```
def rgalaxy(p,μ,σ):
    k = np.random.choice([0,1,2], p=p)
    x = np.random.normal(loc=μ[k], scale=σ[k])
    return x

def rgalaxies(size, p,μ,σ):
    return [rgalaxy(p,μ,σ) for _ in range(size)]
```

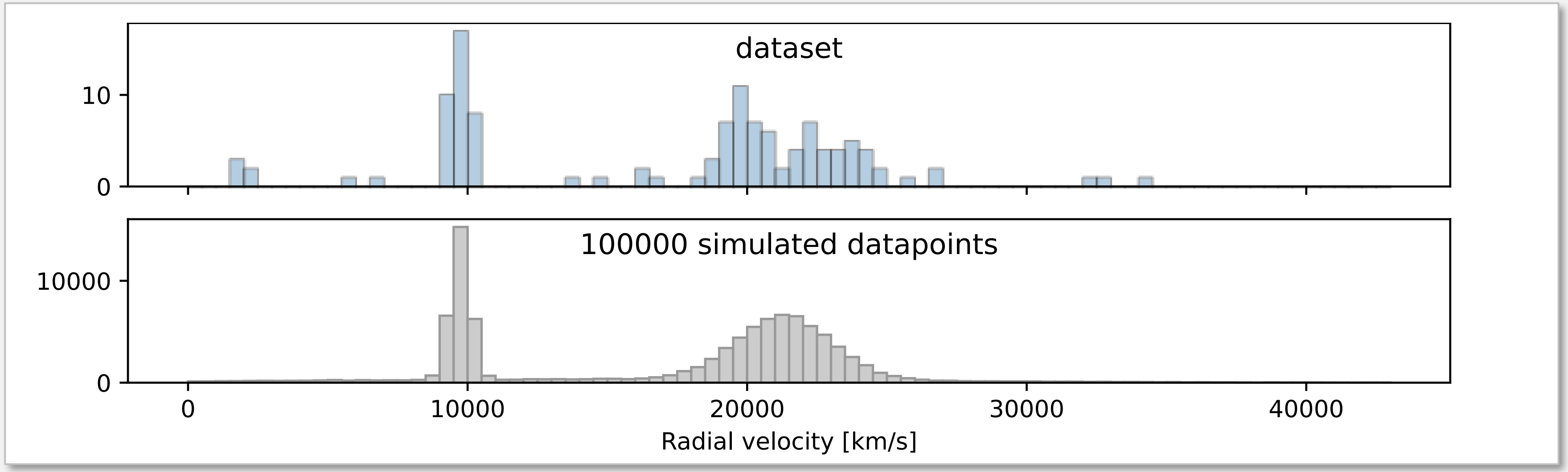
How would you write this in random variable notation?

$$X_i \sim \cdots$$

```
p = [0.28, 0.54, 0.18]
μ = [9740, 21300, 15000]
σ = [340, 1700, 10600]
```

Speeds of galaxies in the Corona Borealis region

Postman, Huchra, Geller (1986)




```
def rgalaxy(p,μ,σ):
    k = np.random.choice([0,1,2], p=p)
    x = np.random.normal(loc=μ[k], scale=σ[k])
    return x

def rgalaxies(size, p,μ,σ):
    return [rgalaxy(p,μ,σ) for _ in range(size)]
```

$$k = \begin{cases} 0 & \text{with prob. } p_0 \\ 1 & \text{with prob. } p_1 \\ 2 & \text{with prob. } p_2 \end{cases} \qquad k \sim \text{Cat}(p)$$

$$X \sim N(\mu_k, \sigma_k^2)$$

There are standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

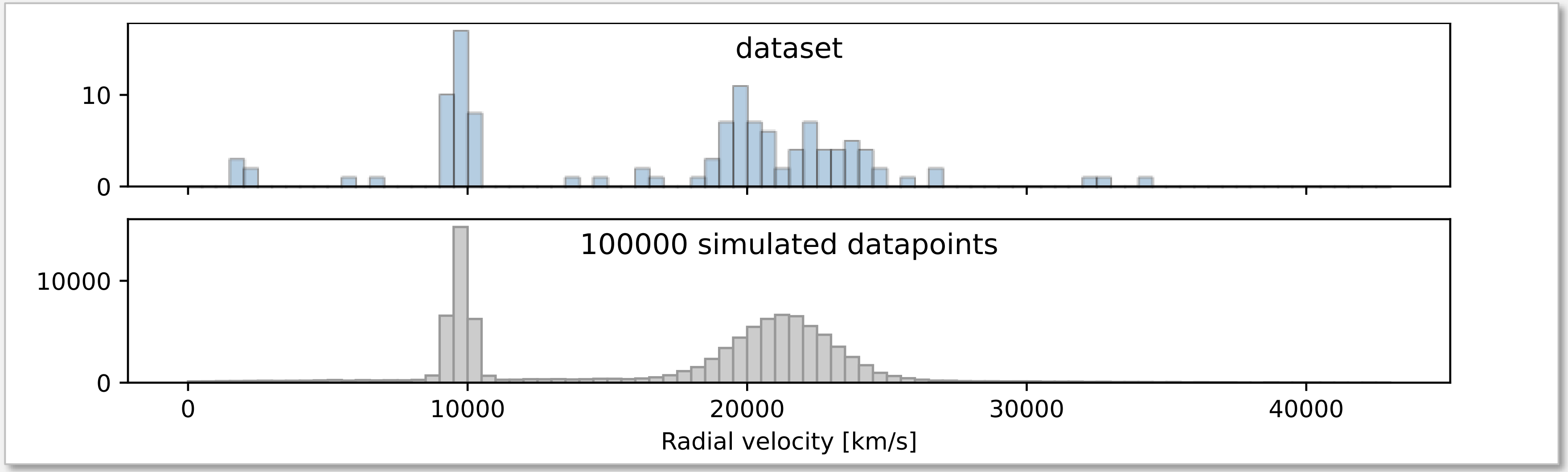
Binomial $X \sim \text{Bin}(n,p)$	$\mathbb{P}(X=x) = \binom{n}{x} p^x (1-p)^{n-x}$ $x \in \{0,1,\ldots,n\}$	For count data, e.g. number of heads in n coin tosses
Poisson $X \sim \text{Pois}(\lambda)$	$\mathbb{P}(X=x) = \frac{\lambda^x e^{-\lambda x}}{x!}$ $x \in \{0,1,\ldots\}$	For count data, e.g. number of buses passing a spot
Categorical $X \sim \text{Cat}([p_1,\ldots,p_k])$	$\mathbb{P}(X=x) = p_x$ $x \in \{1,\ldots,k\}$	For picking one of a fixed number of choices

CONTINUOUS RANDOM VARIABLES

Uniform $X \sim U[a,b]$	$\text{pdf}(x) = \frac{1}{b-a}$ $x \in [a,b]$	A uniformly-distributed floating point value
Normal / Gaussian $X \sim N(\mu,\sigma^2)$	$\text{pdf}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$ $x \in \mathbb{R}$	For data about magnitudes, e.g. temperature or height
Pareto $X \sim \text{Pareto}(\alpha)$	$\text{pdf}(x) = \alpha\, x^{-(\alpha+1)}$ $x \geq 1$	For data about “cascade” magnitudes, e.g. forest fires
Exponential $X \sim \text{Exp}(\lambda)$	$\text{pdf}(x) = \lambda\, e^{-\lambda x}$ $x > 0$	For waiting times, e.g. time until next bus
Beta $X \sim \text{Beta}(a,b)$	$\text{pdf}(x) \propto x^{a-1}(1-x)^{b-1}$ $x \in (0,1)$	Arises in Bayesian inference

Speeds of galaxies in the Corona Borealis region

Postman, Huchra, Geller (1986)



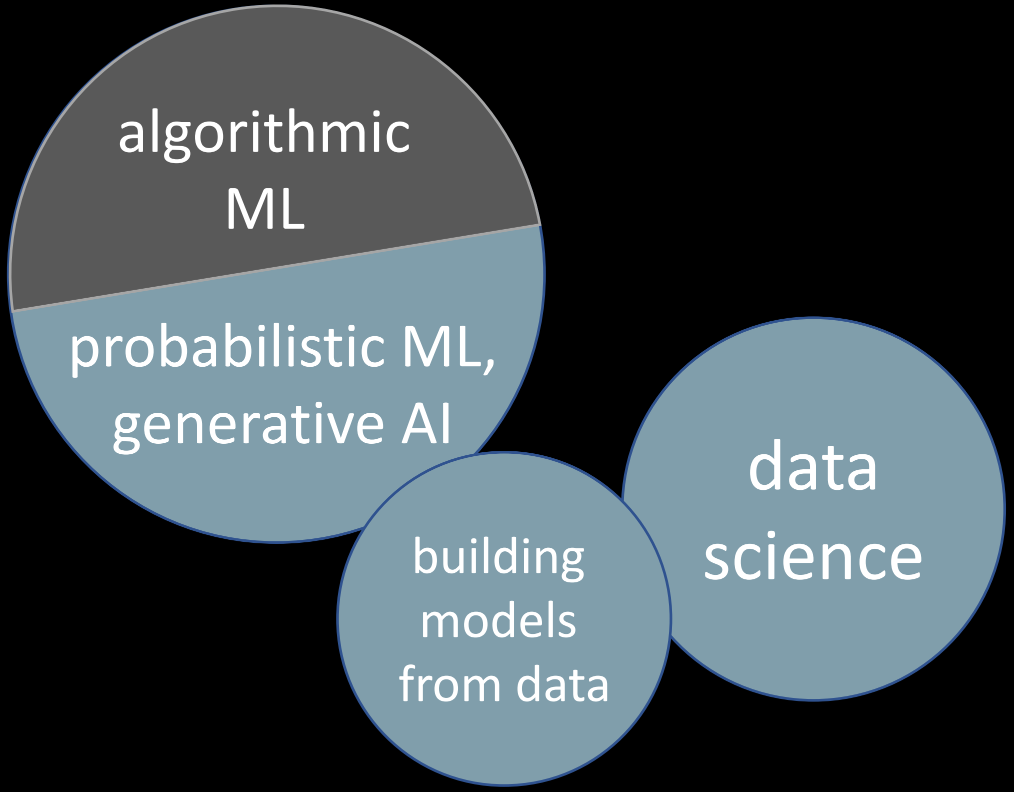

```
def rgalaxy(p,μ,σ):
    k = np.random.choice([0,1,2], p=p)
    x = np.random.normal(loc=μ[k], scale=σ[k])
    return x
```

When we fit this model (i.e. learn the parameters), it tells us the location and shape of the clusters.

What is data science? What's the difference between data science and machine learning?

What looks like "design an ML algorithm to find clusters" …

can be restated as "formulate a suitable probability model and fit it"



§1.5 Better notation for likelihood

All of machine learning is based on a single idea:

1. Write out a probability model

1.5. Find an expression for the likelihood

2. Fit the model from data
   by maximizing the likelihood

This is behind
- A-level statistics formulae
- our climate model
- ChatGPT training

"The likelihood for X of x"

X is a random variable, i.e. a function
x is a value, e.g. a floating point number

The *likelihood function* for a random variable X is written $\Pr_X(x)$ and defined as

$$\Pr_X(x) = \mathbb{P}(X = x) \quad \text{in the case where } X \text{ is discrete}$$

and as

$$\Pr_X(x) = \mathrm{pdf}(x) \quad \text{in the case where } X \text{ is continuous with prob. density function } \mathrm{pdf}(x)$$

For parameterized random variables, write

$\Pr_X(x\,;\theta)$ or $\Pr_X(x \mid \theta)$ or $\Pr_X(x)$

Transforms of random variables:
$\Pr_{X+Y}(0.2)$ or $\Pr_{X^2}(z)$

I call the RNG for X, and I call the RNG for Y, and I add the two outputs together. What's the chance I got 0.2?

The $\Pr_X(x)$ notation keeps track of

- the random variable X
- an observation x

Pairs of random variables:
$\Pr_{X,Y}(x, y)$

$\Pr_{X,Y}(x, y)$ is called the *joint likelihood* of X and Y

$\Pr_{X,Y}(x, y) = \mathbb{P}(X = x \text{ and } Y = y)$
for discrete random variables

$\Pr_{X,Y}(x, y) =$ <something similar/>
for continuous random variables

Independent random variables:

$$\Pr_{X,Y}(x,y) = \Pr_X(x)\,\Pr_Y(y)$$

Independent identically-distributed (IID) sample from X:

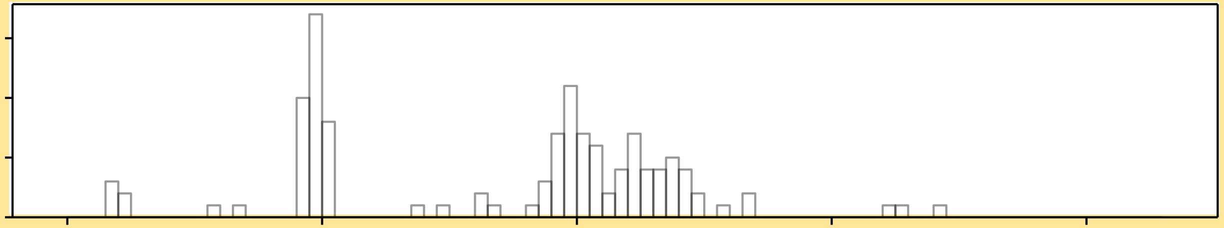
$$\Pr(x_1,\dots,x_n) = \Pr_X(x_1) \times \cdots \times \Pr_X(x_n)$$

Sequential generation of X then Y:

$$\Pr_{X,Y}(x,y) = \Pr_X(x)\,\Pr_Y(y\,;x)$$

Exercise. Write down the joint likelihood $\Pr_{K,X}(k,x)$ for

```
def rgalaxy(p,μ,σ):
    k = np.random.choice([0,1,2], p=p)
    return np.random.normal(loc=μ[k], scale=σ[k])
```



$$\Pr_{K,X}(k,x)$$

$$= \Pr_K(k)\,\Pr_X(x\,;k)$$

$$= p_k \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-(x-\mu_k)^2/2\sigma_k^2}$$

Maximum Likelihood Estimation, again

If we've seen an outcome x, and we've proposed a probability model X, and if its distribution involves some unknown parameters θ,

the *maximum likelihood estimator* for θ is

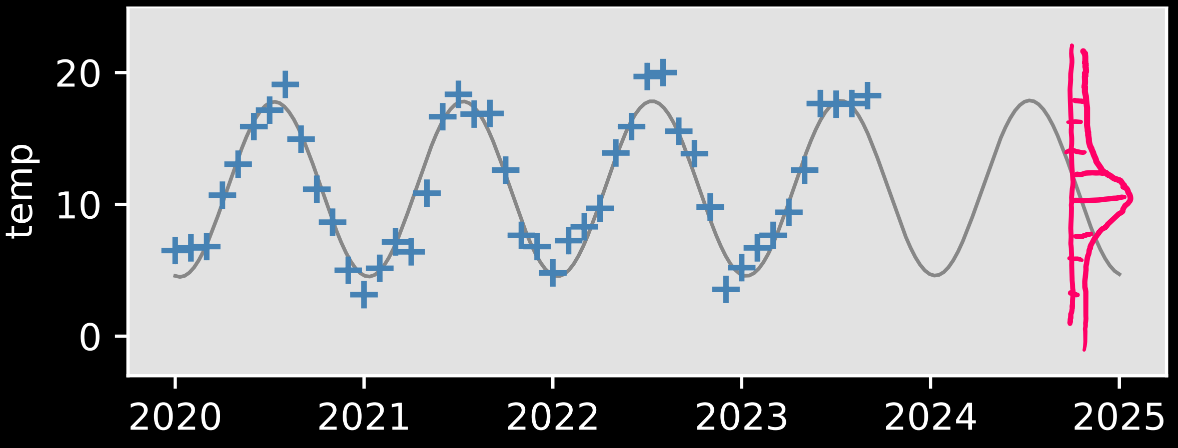
$$\hat{\theta} = \arg\max_{\theta} \Pr_X(x\,;\theta)$$

- x could be discrete or continuous
- x could be a single observation or a dataset with many observations

The point of the likelihood notation is so that we can write down a single equation and have it cover all these cases.

We've looked at two types of model:

supervised

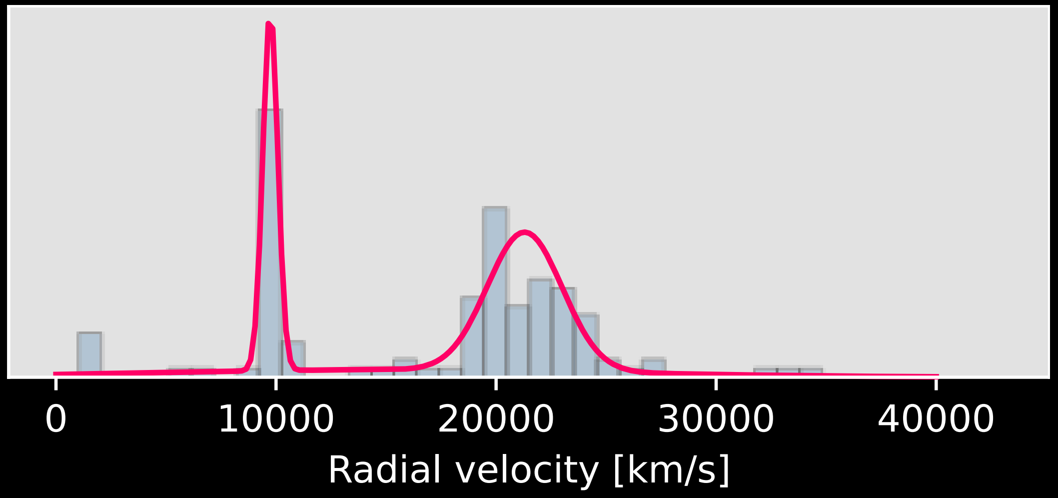


Given a dataset of (t_i, temp_i) pairs, $i \in \{1, \dots, n\}$,
I'd like to learn how temperatures have been changing.

~~i.e. I'd like to predict temp as a function of t.~~

I'd like to fit a probability model for Temp, where the parameters of the distribution depend on t

generative



Given a dataset $[x_1, \dots, x_n]$ of galaxy speeds,
I'd like to fit a probability model.

(This lets me generate new values, similar but not identical to the dataset.)

Terminology for supervised learning

How can I PREDICT temp GIVEN t?

station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16	10	45.8	97.8	1.1	8.7	4.9
⋮									

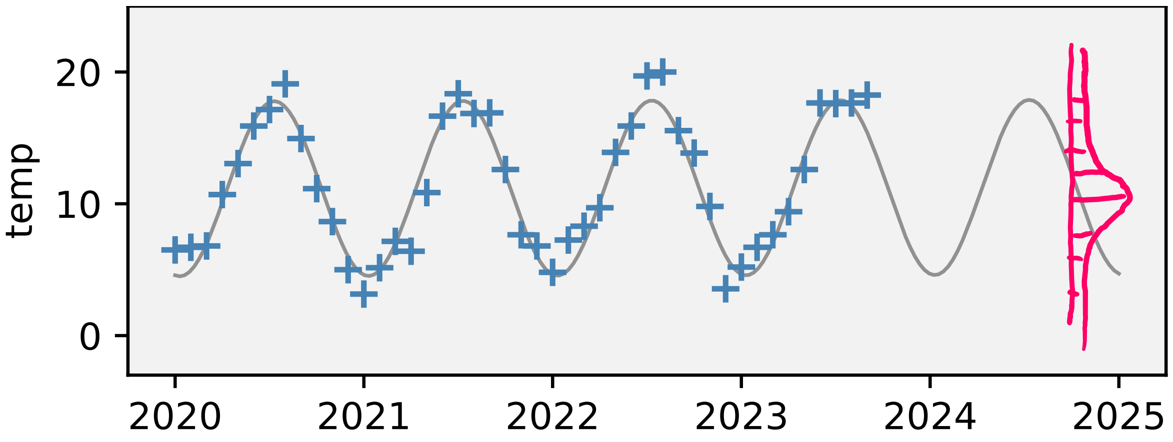
called the PREDICTOR variable,
or the FEATURE,
or the COVARIATE

called the RESPONSE,
or the LABEL variable

- Here the response is real-valued, so we call it REGRESSION.
- If the response were categorical, we'd call it CLASSIFICATION.

https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data

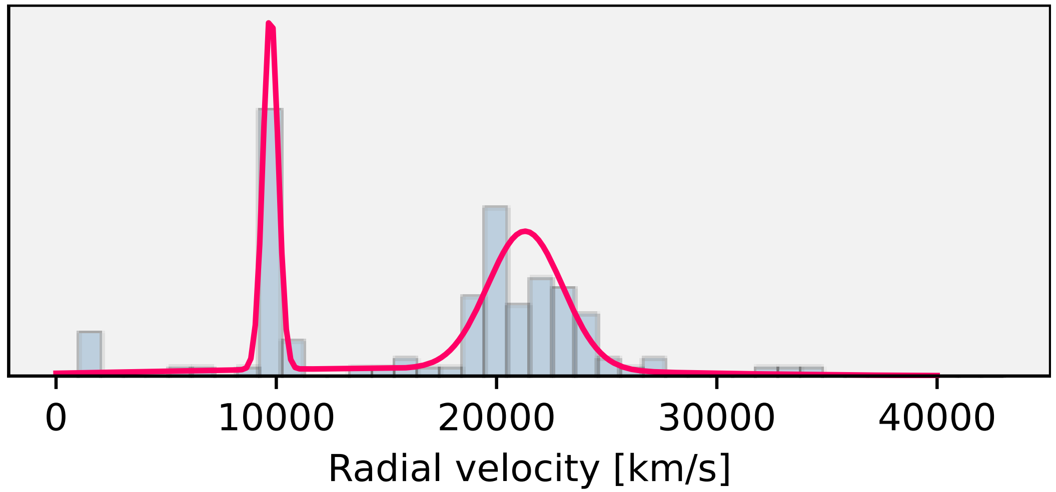
supervised learning



Given a dataset $(x_1, y_1), \dots, (x_n, y_n)$ where y_i is the label in record i and x_i is the predictor variable or variables …

1. Propose a probability model for the response Y, with likelihood
$$\Pr_Y(y; x, \theta)$$
2. Model the dataset as independent observations; thus the likelihood of the dataset is
$$\Pr(\text{dataset}) = \prod_{i=1}^{n} \Pr_Y(y_i; x_i, \theta)$$
3. Learn θ using maximum likelihood estimation

generative modelling



Given a dataset $x_1, \dots, x_n$ …

1. Propose a probability model i.e. a random variable X, with likelihood
$$\Pr_X(x; \theta)$$
2. Model the dataset as independent observations; thus the likelihood of the dataset is
$$\Pr(\text{dataset}) = \prod_{i=1}^{n} \Pr_X(x_i; \theta)$$
3. Learn θ using maximum likelihood estimation

Exercise 1.6.1 (Fitting a Normal distribution)
Given a numerical dataset $x_1, \dots, x_n$, fit a $\text{Normal}(\mu, \sigma^2)$ distribution, where μ and σ are unknown.

This is a GENERATIVE MODELLING task.

μ , σ are parameters we want to estimate.

Model for a single observation

$$X \sim N(\mu, \sigma^2)$$

Likelihood for a single observation

$$\Pr_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Log likelihood of the dataset

$$\log \Pr(x_1, \dots, x_n) = \log\left[\Pr(x_1) \times \dots \times \Pr(x_n)\right]$$ assuming the observations are independent

$$= \sum_{i=1}^{n} \log \Pr(x_i)$$

$$= \sum_{i=1}^{n} \log\left\{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i-\mu)^2/2\sigma^2}\right\} = \frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

Maximize over unknown parameters

$$\left.\begin{aligned} \frac{\partial}{\partial\mu}\log \Pr(x_1 \dots x_n) &= 0 \\ \frac{\partial}{\partial\sigma}\log \Pr(x_1 \dots x_n) &= 0 \end{aligned}\right\}$$

$$\hat{\mu} = \frac{1}{n}\sum x_i$$

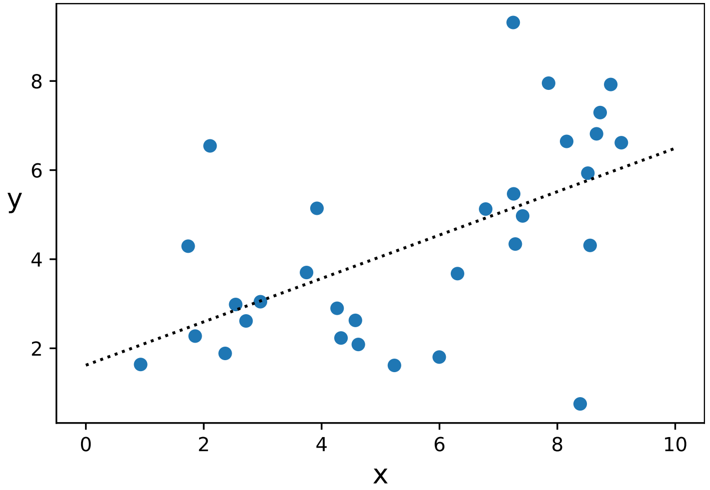
$$\hat{\sigma} = \sqrt{\frac{1}{n}\sum(x_i - \hat{\mu})^2}$$

Exercise 1.7.1 (Straight-line fit)

Given a labelled dataset $(x_1, y_1), \ldots, (x_n, y_n)$ consisting of pairs of numbers, fit the model

$$Y_i \sim a + b\, x_i + \text{Normal}(0, \sigma^2)$$

where σ is given and a and b are parameters to be estimated.



This is a supervised task.
label = y
predictor = x.

Model for a single observation:

$$Y \sim a + bx + N(0, \sigma^2) \sim N(a+bx, \sigma^2)$$

by linearity of the Gaussian dist.

Likelihood of a single observation:

$$\Pr_Y(y; a, b, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(y-a-bx)^2/\sigma^2}$$

Log likelihood of the dataset:

$$\log \Pr(y_1, \ldots, y_n; a, b, \sigma) = \frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - a - bx_i)^2$$

Optimize over the unknown parameters:

```
x = np.array([...])
y = np.array([...])
σ = ...

def logPr(y, x, θ):
    a,b = θ
    loglik = scipy.stats.norm.logpdf(y, loc=a+b*x, scale=σ)
    return np.sum(loglik)

initial_guess = ...
â, b̂ = scipy.optimize.fmin(lambda θ: -logPr(y,x,θ),
                                   initial_guess)
```

Data from Randall Munroe, https://xkcd.com/2048/, via https://gitlab.com/b-rowlingson

Algorithmic *versus* probabilistic machine learning*

* non-examinable

ALGORITHMIC VIEW OF ML

We're given a labelled dataset.
We want to learn to predict the label.
We do this by minimizing a loss function.

Statistical modeling: the two cultures

Leo Breiman

Statistical Science, 2001

There are two cultures in the use of statistical modeling to reach conclusions from data.

- One assumes that the data are generated by a given stochastic data model.
- The other uses algorithmic models and treats the data mechanism as unknown.

The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems.

In the mid-1980s two powerful new algorithms for fitting data became available: neural nets and decision trees. A new research community using these tools sprang up. Their goal was predictive accuracy. The community consisted of young computer scientists, physicists and engineers plus a few aging statisticians. They began using the new tools in working on complex prediction problems where it was obvious that data models were not applicable: speech recognition, image recognition, nonlinear time series prediction, handwriting recognition, prediction in financial markets.

ground truth:
Let y_i be the actual observed temperature at time t_i

timepoint t

predicted temperature $\hat{y}$ at timepoint t

it's our job as modellers to find θ so as to minimize prediction error, e.g.

pick θ to minimize

$$\sum_i L(y_i, \hat{y}(t_i))$$

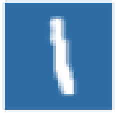
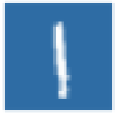
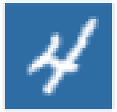
where

$$L(y, \hat{y}) = (y - \hat{y})^2$$

Neural network classification

The MNIST database of handwritten images consists of records (x_i, y_i) where $x_i \in \mathbb{R}^{28\times 28}$ is a greyscale image with 28×28 pixels, and $y_i \in \{0, \dots, 9\}$ is the digit.

We'd like to predict the digit, given an image. How might we learn to do this?

image	digit
	2
	1
	3
	1
	4

Data from http://yann.lecun.com/exdb/mnist/

ground truth:
Let y_i be the actual observed label in the dataset

it's our job as modellers to find θ so as to maximize prediction accuracy, i.e.

pick θ to minimize

$$\sum_i L(y_i, \hat{y}(t_i))$$

where

$$L(y, \hat{y}) = 1_{y \neq \hat{y}}$$

Supervised Learning

Data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

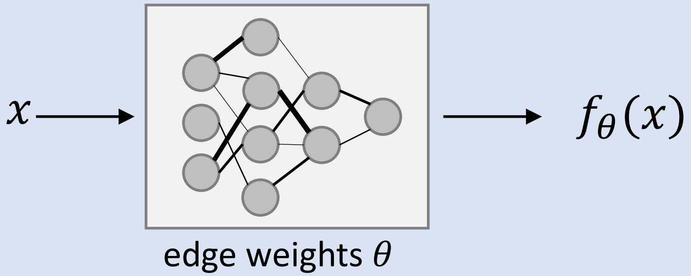
Labels: $y_1, y_2, \ldots, y_n$

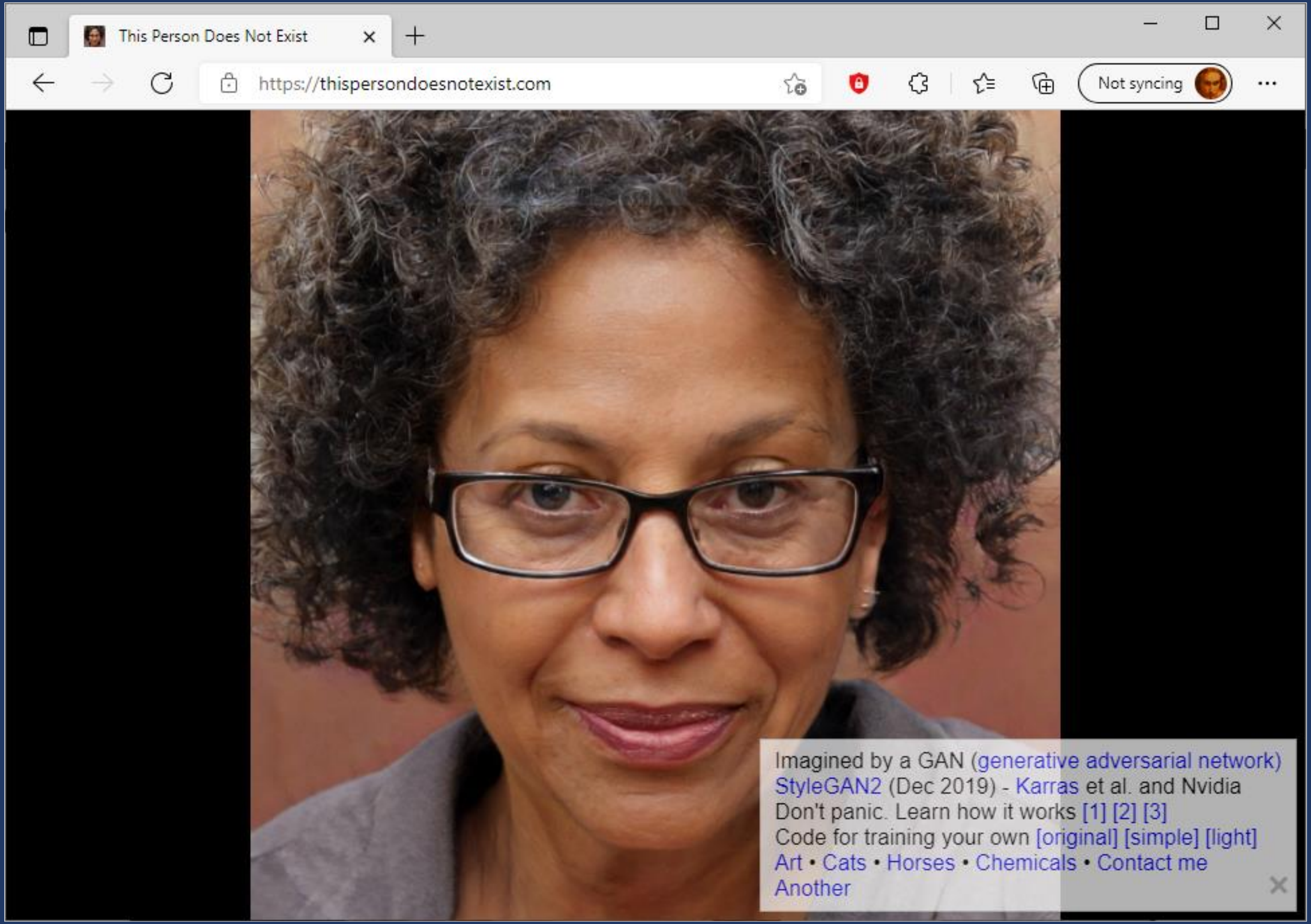
Task: Predict the label

$$y_i \approx f_\theta(x_i)$$

Training goal: Invent a loss function and learn θ to minimize the prediction loss

$$\sum_i L(y_i, f_\theta(x_i))$$





This is machine learning, too! But what are the labels, and what's the loss function?

ALGORITHMIC VIEW OF MACHINE LEARNING

Supervised Learning

Data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

Labels: $y_1, y_2, \ldots, y_n$

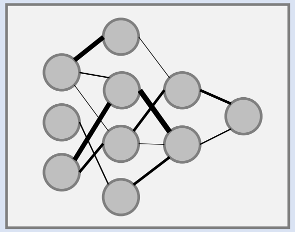
Task: Predict the label

$$y_i \approx f_\theta(x_i)$$

Training goal: Invent a loss function and learn θ to minimize the prediction loss

$$\sum_i L(y_i, f_\theta(x_i))$$

x →



→ $f_\theta(x)$

edge weights θ

Generative Modelling

Data: $\{x_1, x_2, \ldots, x_n\}$

Labels: n/a

Task: learn to synthesize new values similar (but not identical) to those in the dataset, ...

Training goal: ???

This course teaches a different way to think of modelling …

ALGORITHMIC VIEW OF MODELLING

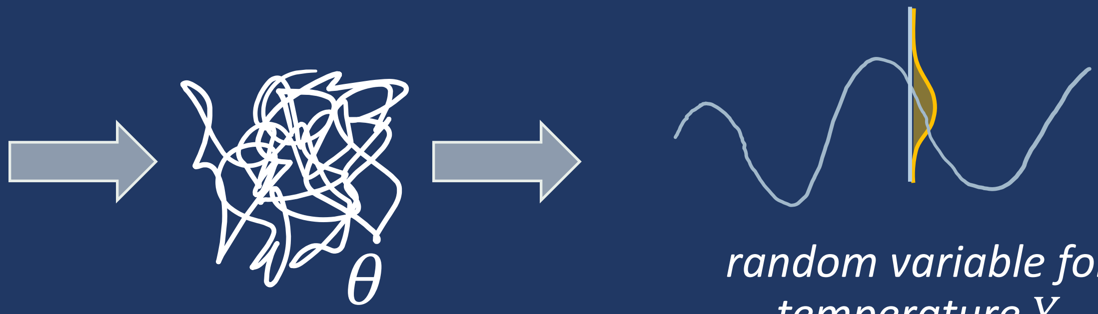
timepoint t

predicted temperature $\hat{y}$ at timepoint t

Goal:
minimize prediction error

PROBABILITY MODELLER'S VIEW

timepoint t

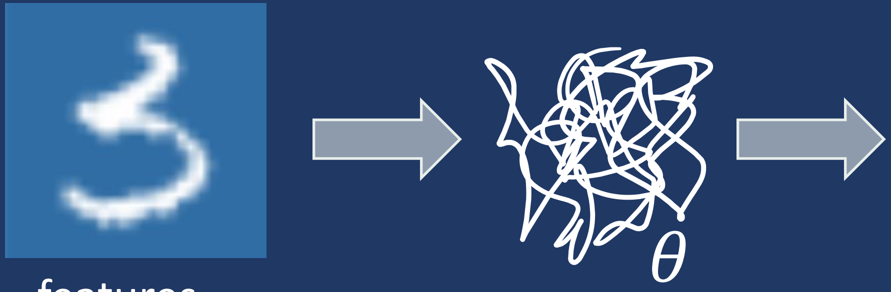


random variable for temperature Y at timepoint t

Goal:
find a good-fitting distribution

This course teaches a different way to think of modelling …

ALGORITHMIC VIEW OF MODELLING

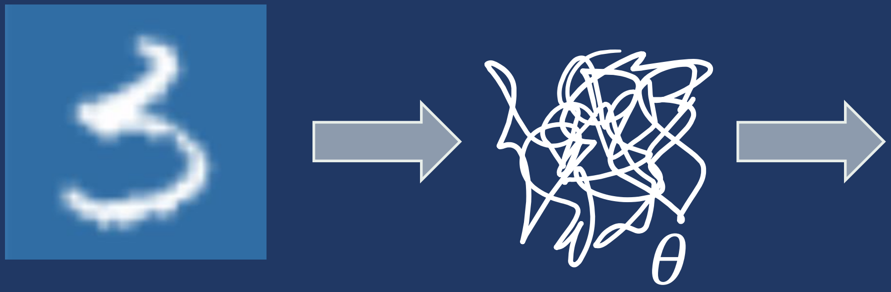


features

5
predicted label $\hat{y}$

Goal:
maximize prediction accuracy

PROBABILITY MODELLER'S VIEW



$$\begin{cases} \text{5 with prob 45\%} \\ \text{3 with prob 41\%} \\ \text{6 with prob \ \ 5\%} \\ \quad\quad \dots \end{cases}$$

random predicted label Y

Goal:
find a good-fitting distribution

PROBABILISTIC MACHINE LEARNING

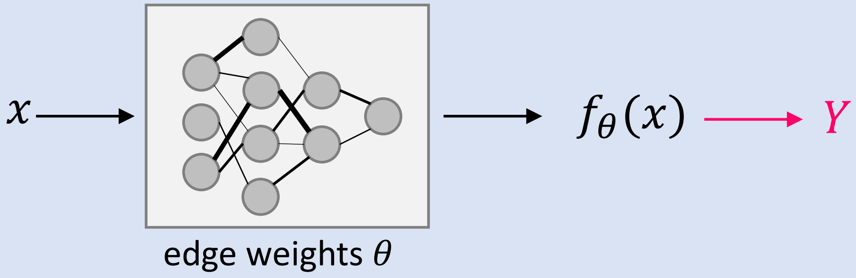
Supervised Learning

Data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

Labels: $y_1, y_2, \ldots, y_n$

Task: fit the probability model $\Pr_Y(y\,; f_\theta(x))$

Training goal: MLE



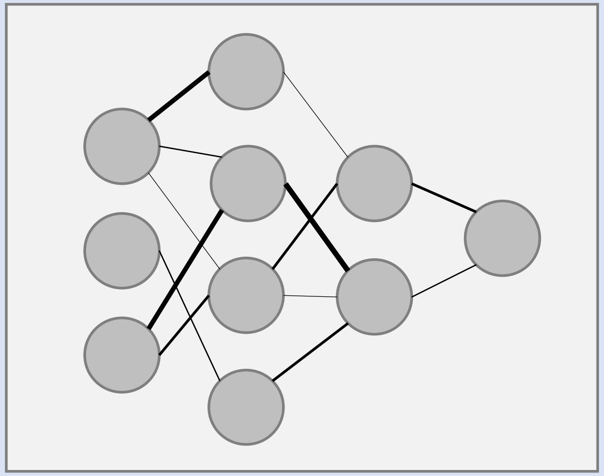
Generative Modelling

Data: $\{x_1, x_2, \ldots, x_n\}$

Labels: n/a

Task: learn to synthesize new values similar (but not identical) to those in the dataset, ...

Training goal: ???

random noise Z →  → $X = f_\theta(Z)$

edge weights θ

QUESTION. How could we even use neural networks to generate novel images? What should the input be?

The output X is a random variable.

I.e. If I ran this network lots of times, each time with different noise, I get different X. I could plot a histogram of these outputs.

Write $\Pr_X(x)$ for its likelihood function, as usual.

PROBABILISTIC MACHINE LEARNING

Supervised Learning

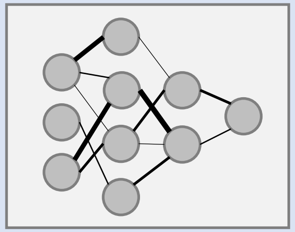
Data: $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

Labels: $y_1, y_2, \dots, y_n$

Task: fit the probability model $\Pr_Y(y\,; f_\theta(x))$

Training goal: MLE

$x \longrightarrow$



edge weights θ

$\longrightarrow f_\theta(x) \longrightarrow Y$

Generative Modelling

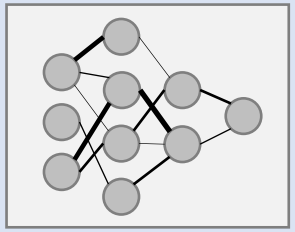
Data: $\{x_1, x_2, \dots, x_n\}$

Labels: n/a

Task: fit the probability model $\Pr_X(x\,; \theta)$

Training goal: MLE

random noise Z $\longrightarrow$



edge weights θ

$\longrightarrow X = f_\theta(Z)$