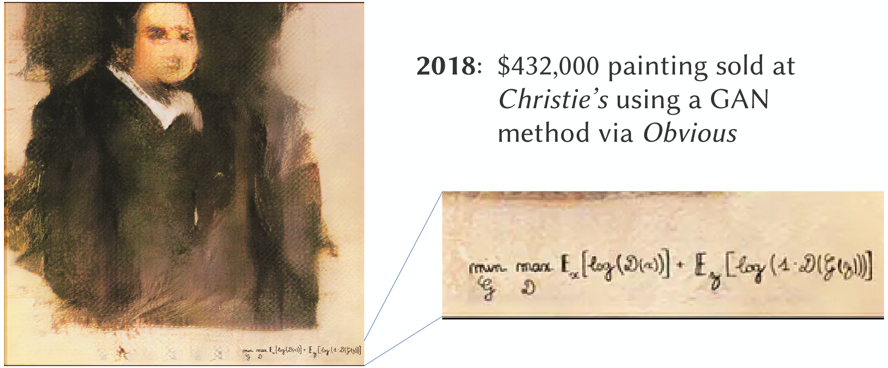
Machine Learning 101

DESY Summer Program 2025 | July 31st 2025

DESY : Stephen Jiggins

Contact Info: stephen.jiggins@desy.de

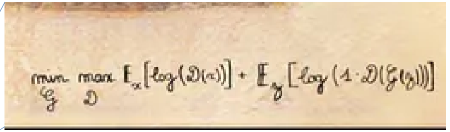
What is Machine Learning?



Portrait of Edmond de Belamy

2018: \$432,000 painting sold at *Christie's* using a GAN method via *Obvious*

What is Machine Learning?



Portrait of Edmond de Belamy

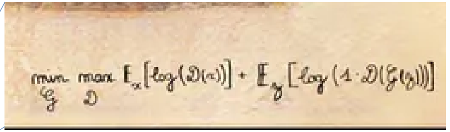
2018: \$432,000 painting sold at *Christie's* using a GAN method via *Obvious*

2022: Colorado state fair winner using *DALL · E2.*

Sparks the question of morality in the age of AI realism

X_1

What is Machine Learning?



Portrait of Edmond de Belamy

2018: $432,000 painting sold at *Christie's* using a GAN method via *Obvious*

2022: Colarado state fair winner using *DALL · E2.*

Sparks the question of morality in the age of AI realism

2023: Tricking the world is not hard with such technology!


Support the Guardian
Fund independent journalism with €5 per month
Support us
The Guardian
News | Opinion | Sport | Culture | Lifestyle | More

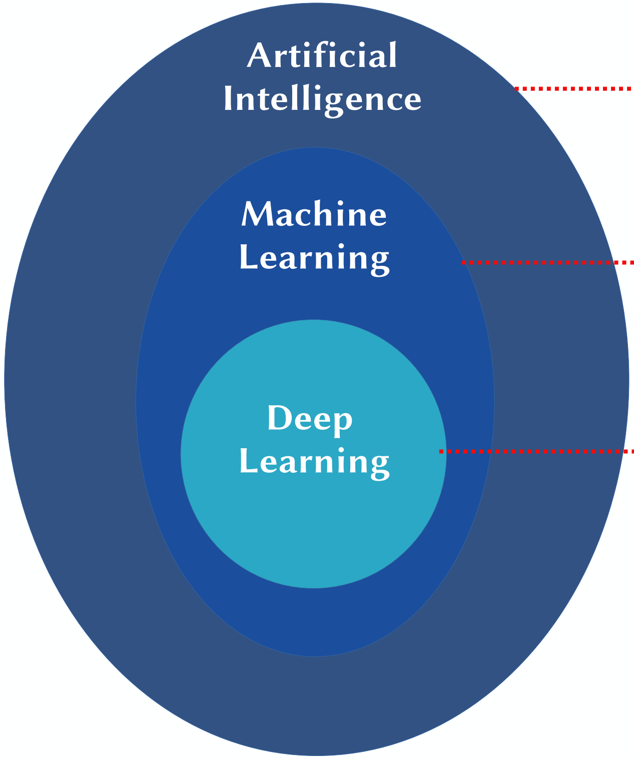
I thought I was immune to being fooled online. Then I saw the pope in a coat

Joel Golby

An encounter with an AI-generated image of his holiness has changed me: I now have sympathy for credulous baby boomers

'I thought wearing a really big coat and looking like a Metal Gear Solid 2 boss battle might have been part of his ongoing cool guy shtick. Lord, forgive me.' Photograph: Reddit

What is Machine Learning?



Artificial Intelligence:
Engineering behind intelligent machines and programs

Machine Learning:
Ability to learn without being explicitly programmed

Deep Learning:
Learning based on deep neural networks

What is Machine Learning?

'Giving computers the ability to learn without explicitly programming them'
- Arthur Samuel, 1959

What is Machine Learning?

‘~~Giving computers the ability to learn without explicitly programming them~~’

- Arthur Samuel, 1959

Paper: link

- *‘A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E.’* - Mitchell 1997

[Mitchell 1997]: ISBN 0070428077

What is Machine Learning?

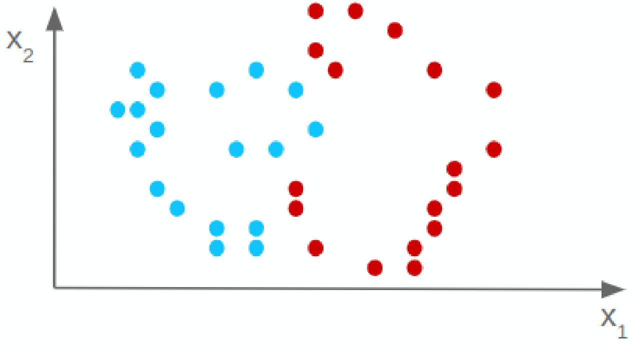
~~'Giving computers the ability to learn without explicitly programming them'~~
- Arthur Samuel, 1959

Paper: link

- *'A computer program is said to learn from* ***experience E*** *with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E.'* - Mitchell 1997

- **Experience (E):** The stimulus that drives learning, is the set of **examples** $\boldsymbol{x}$, or a **dataset** $\mathcal{D}$ of many examples $\mathbf{x} = \{x\}_N$:

  Supervised: The dataset examples have an associated label or target $\mathbf{y} = \{y\}_N$



[Mitchell 1997]: ISBN 0070428077

What is Machine Learning?

~~'Giving computers the ability to learn without explicitly programming them'~~

- Arthur Samuel, 1959

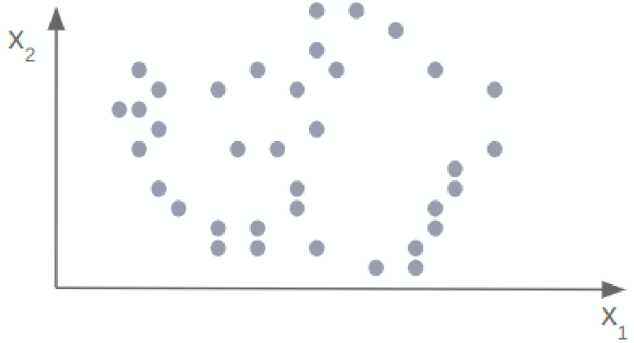
Paper: link

- *'A computer program is said to learn from* ***experience E*** *with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.'* - Mitchell 1997

- **Experience (E):** The stimulus that drives learning, is the set of **examples** $\boldsymbol{x}$, or a **dataset** $\mathcal{D}$ of many examples $\mathbf{x} = \{x\}_N$:

  Supervised: The dataset examples have an associated label or target $\mathbf{y} = \{y\}_N$

  Unsupervised: The dataset examples have **no** labels or targets



[Mitchell 1997]: ISBN 0070428077

What is Machine Learning?

~~*'Giving computers the ability to learn without explicitly programming them'*~~

\- Arthur Samuel, 1959

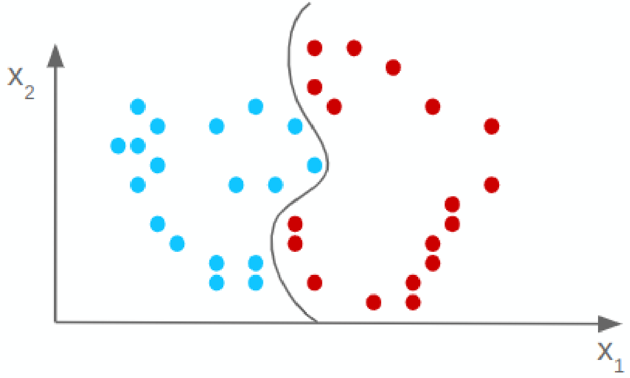
Paper: link

- *'A computer program is said to learn from experience E with respect to some class of **tasks T** and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E.'* - Mitchell 1997

- **Task (T):** Given an **example** $\boldsymbol{x} \in \mathbb{R}^n$ the **model** $\boldsymbol{f}$ should learn a prediction $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. E.g.

  Classification: Assign the example to a category $f : \mathbb{R}^n \to \{1,\ldots,k\}$

  Regression: Predict the value of a target $f : \mathbb{R}^n \to \mathbb{R}$



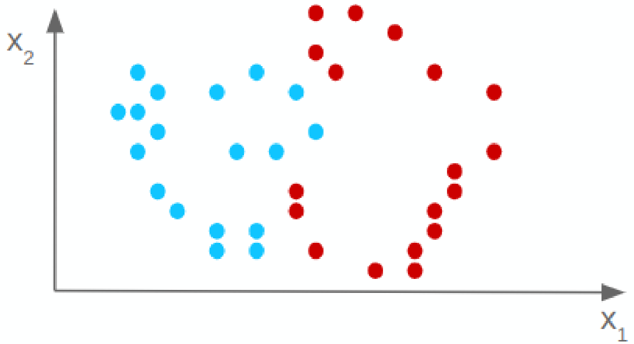
[Mitchell 1997]: ISBN 0070428077

What is Machine Learning?

~~*'Giving computers the ability to learn without explicitly programming them'*~~
- Arthur Samuel, 1959

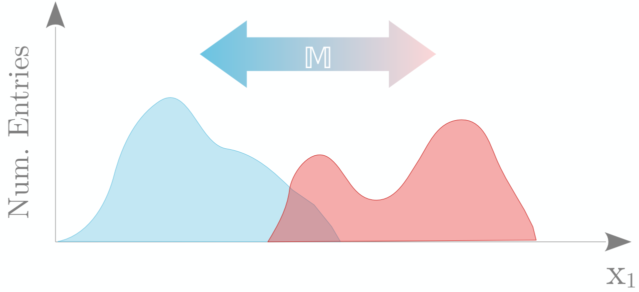
Paper: link

- *'A computer program is said to learn from experience E with respect to some class of tasks T and* ***performance measure P*** *, if its performance at tasks in T , as measured by P , improves with experience E.'* - Mitchell 1997



- **Performance (P):** Evaluate the performance of the **model *f***

  Measure Theory: Generalisation of geometric distances for a measureable space (Ω, F), such that for a measure $\mathbb{M} : F \rightarrow [0, +\infty]$

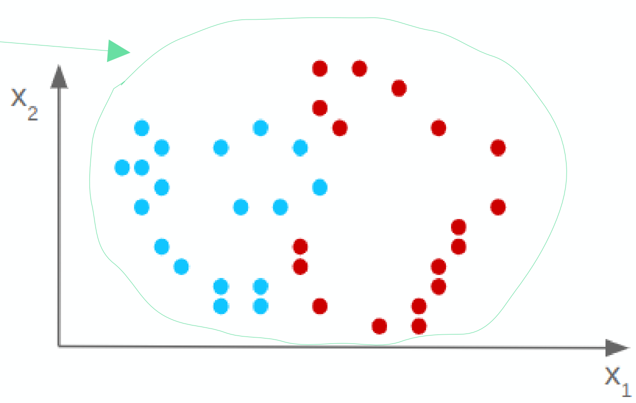


[Mitchell 1997]: ISBN 0070428077

Basic Terminology - Day-to-day

- **Datasets:** The data from which the algorithm will need to learn from:

$$\mathcal{D} = \{ \{x_i , y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$$



Basic Terminology - Day-to-day

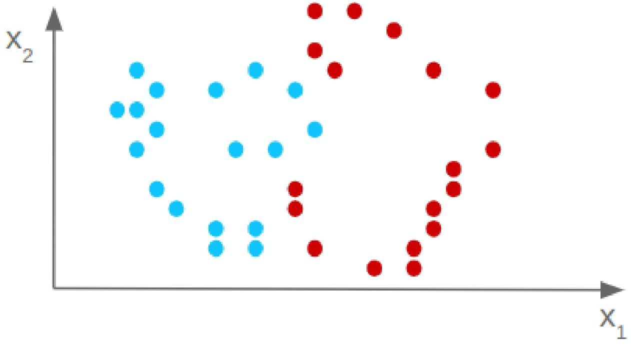
- **Datasets:** The data from which the algorithm will need to learn from:

$$\mathcal{D} = \{ \{x_i , y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$$

- **Features:** Each dimension of the example x, or input space $\mathbb{R}^X$, which represents some *feature* such as pixel color, particle energy, etc...:

$$x \in \mathbb{R}^X$$

$\mathbb{R}^2$



Basic Terminology - Day-to-day

- **Datasets:** The data from which the algorithm will need to learn from:

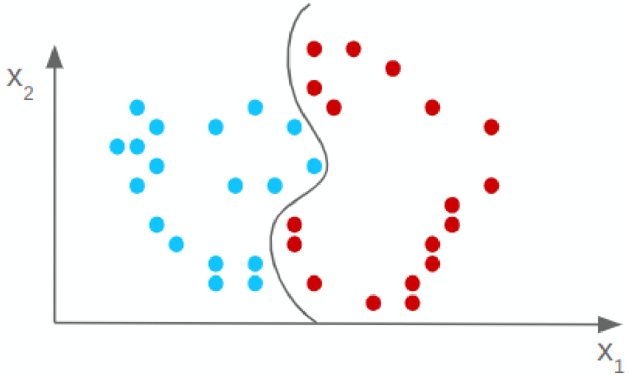
$$\mathcal{D} = \{ \{x_i , y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$$

- **Features:** Each dimension of the example x, or input space $\mathbb{R}^X$, which represents some *feature* such as pixel color, particle energy, etc...:

$$x \in \mathbb{R}^X$$

- **Algorithms:** The process in which the model f defined by a parameter set Φ is optimised according to some **objective function**:

$$\Phi^* = \arg\min_{\Phi} [\ f\ ]$$



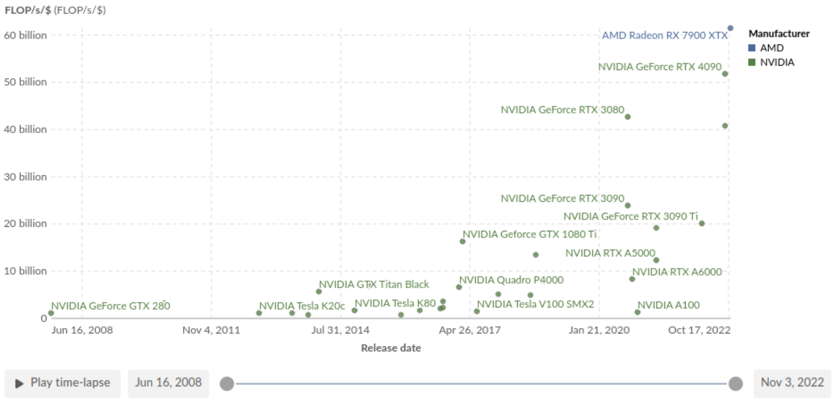
Is ML useful?

AI/ML Summer

- **The boom in ML/AI has been primarily the result of:**

  Datasets: The vast amount of data from images curated by ImageNet to Protein Data Bank (PDB)

  Processing Power: The computational power of CPU/GPUs and now emerging accelerator platforms such as TPUs etc…



- **At a secondary level it is also the result of:**

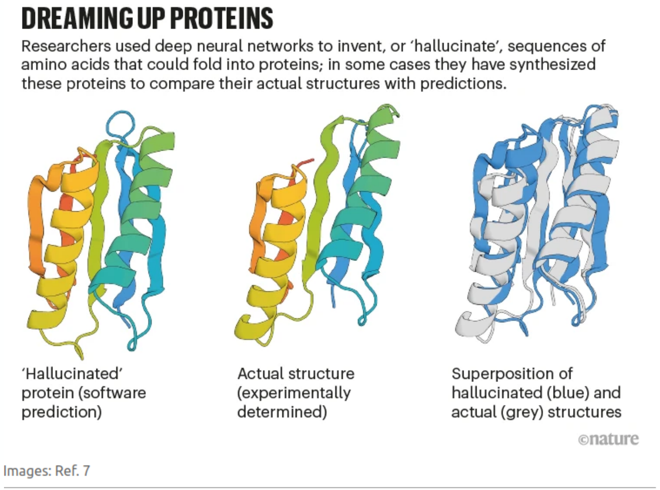
  Open-source Libraries: PyTorch (Meta), Tensorflow + JAX (Google), etc…

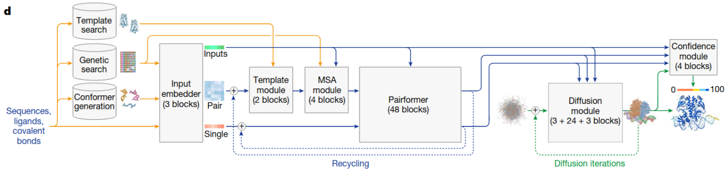
  High—Level Interpretable Languages: Julia, Python, etc…

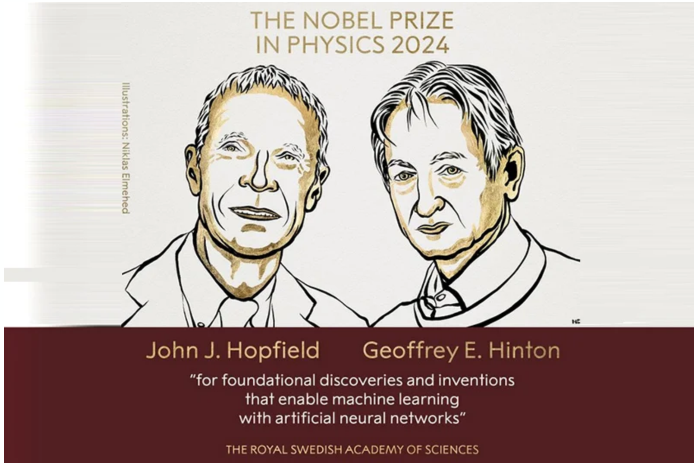
AI/ML Summer

- **Bio-molecular research:**

  AlphaFold1.0 → 3.0 has lead to substantial advancements in predicting protein structures:

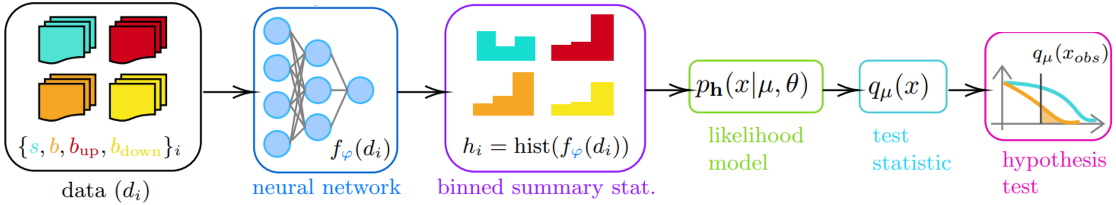






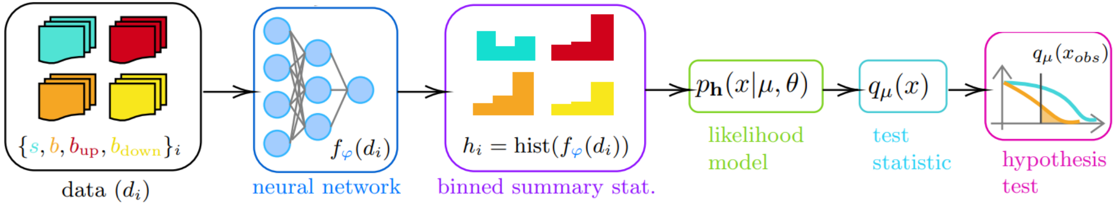
High Energy Physics – ML Developments

→ Checkout the HEP Machine Learning Living Review



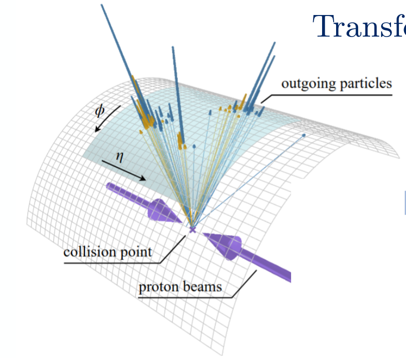
High Energy Physics – ML Developments

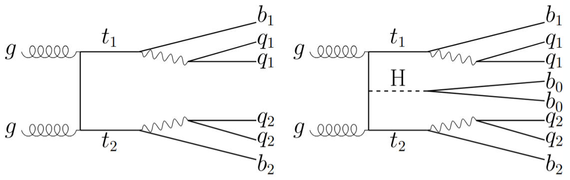
→ Checkout the HEP Machine Learning Living Review



SPANet Transformer - arXiv:2106.03898

Transformer model using scaled dot-product equivariant properties for top-jet reconstruction





tt

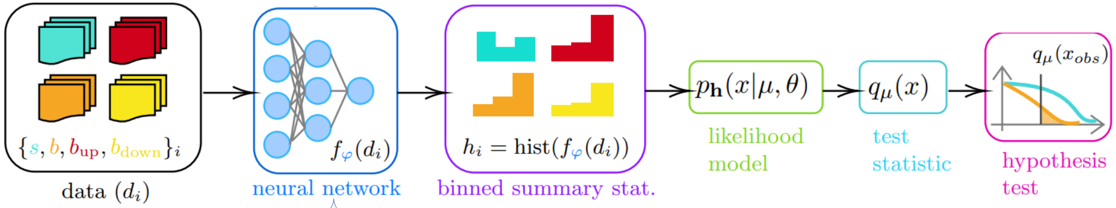
	N_{jets}	Event Fraction	SPA-NET Efficiency Event	SPA-NET Efficiency Top Quark	χ^2 Efficiency Event	χ^2 Efficiency Top Quark
All Events	== 6	0.245	0.643	0.696	0.424	0.484
	== 7	0.282	0.601	0.667	0.389	0.460
	≥ 8	0.320	0.528	0.613	0.309	0.384
	Inclusive	**0.848**	**0.586**	**0.653**	**0.392**	**0.457**
Complete Events	== 6	0.074	0.803	0.837	0.593	0.643
	== 7	0.105	0.667	0.754	0.413	0.530
	≥ 8	0.145	0.521	0.662	0.253	0.410
	Inclusive	**0.325**	**0.633**	**0.732**	**0.456**	**0.552**

ttH

	N_{jets}	Event Fraction	SPA-NET Efficiency Event	SPA-NET Efficiency Higgs	SPA-NET Efficiency Top	χ^2 Efficiency Event	χ^2 Efficiency Higgs	χ^2 Efficiency Top
All Events	== 8	0.261	0.370	0.497	0.540	0.044	0.151	0.053
	== 9	0.313	0.343	0.492	0.514	0.038	0.146	0.066
	≥ 10	0.313	0.294	0.472	0.473	0.030	0.135	0.072
	Inclusive	**0.972**	**0.330**	**0.485**	**0.502**	**0.039**	**0.146**	**0.062**
Complete Events	== 8	0.042	0.532	0.657	0.663	0.016	0.151	0.063
	== 9	0.070	0.422	0.601	0.596	0.013	0.146	0.076
	≥ 10	0.115	0.306	0.545	0.523	0.008	0.134	0.080
	Inclusive	**0.228**	**0.383**	**0.583**	**0.572**	**0.012**	**0.144**	**0.073**

High Energy Physics – ML Developments

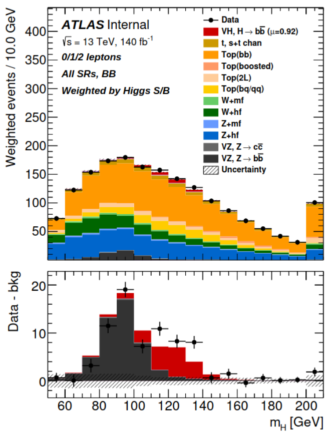
→ Checkout the HEP Machine Learning Living Review

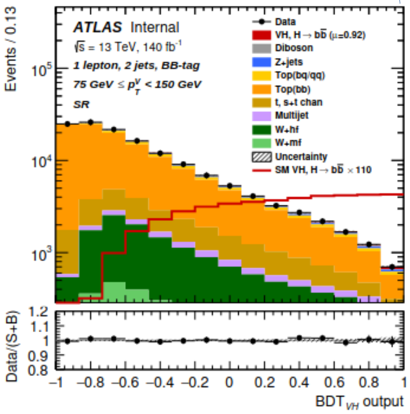


Typical signal classification in data analyses, e.g. Higgs discovery:

Analysis	Years of data collection	Sensitivity without machine learning	Sensitivity with machine learning	Ratio of P values	Additional data required
CMS[24] $H \to \gamma\gamma$	2011–2012	2.2σ, $P = 0.014$	2.7σ, $P = 0.0035$	4.0	51%
ATLAS[43] $H \to \tau^+\tau^-$	2011–2012	2.5σ, $P = 0.0062$	3.4σ, $P = 0.00034$	18	85%
ATLAS[99] $VH \to bb$	2011–2012	1.9σ, $P = 0.029$	2.5σ, $P = 0.0062$	4.7	73%
ATLAS[41] $VH \to bb$	2015–2016	2.8σ, $P = 0.0026$	3.0σ, $P = 0.00135$	1.9	15%
CMS[100] $VH \to bb$	2011–2012	1.4σ, $P = 0.081$	2.1σ, $P = 0.018$	4.5	125%

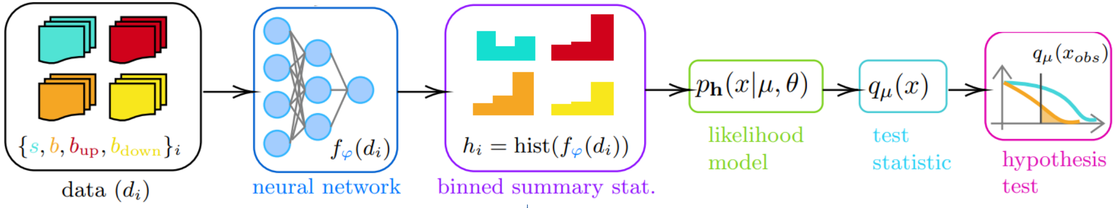
Source: A. Radovic et al., Nature 560(2018) no. 7716,41





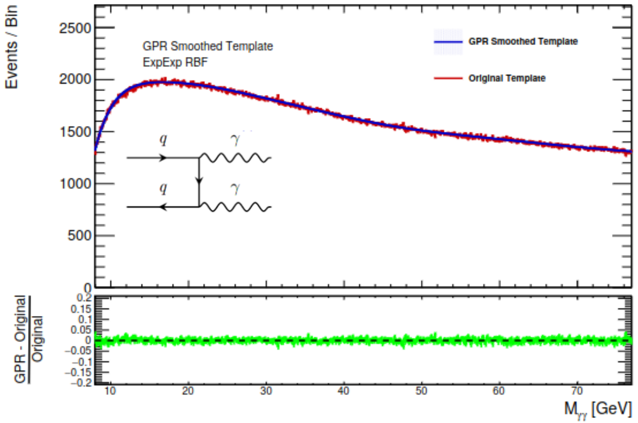
High Energy Physics – ML Developments

→ Checkout the HEP Machine Learning Living Review



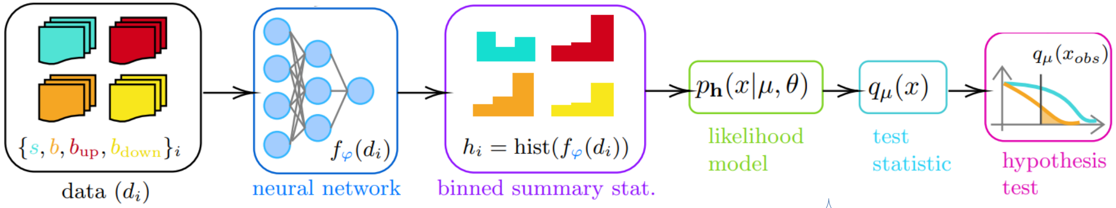
Background Estimation by smoothing data-driven backgrounds using gaussian processes for H→γγ

Source: CDS-2791079



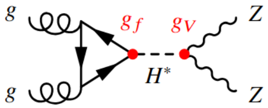
High Energy Physics – ML Developments

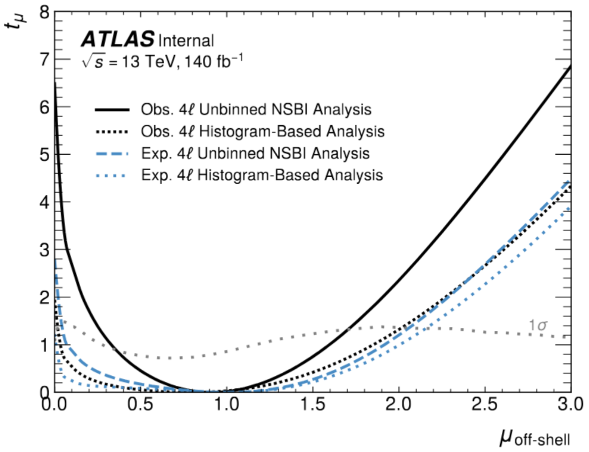
→ Checkout the HEP Machine Learning Living Review



New likelihood inferencing paradigm:

H→ZZ off-shell analysis using unbinned likelihood ratios from neural networks

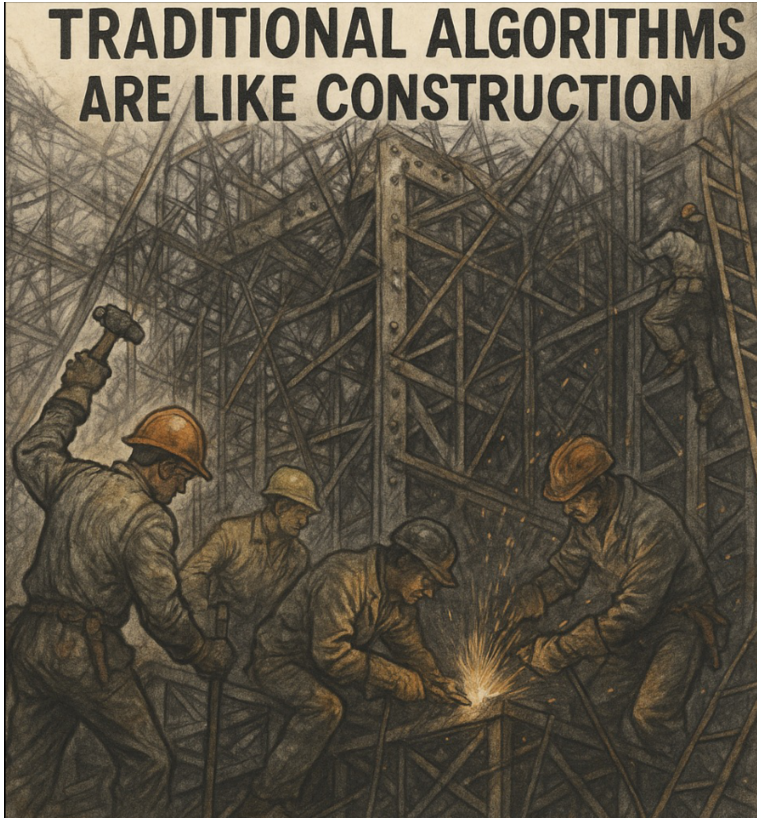




Source: ATL-COM-PHYS-2024-24

Learning Law

Machine Learning versus Algorithm Development



Learning Law

- **Four key parts** to the 'Machine Learning' process:

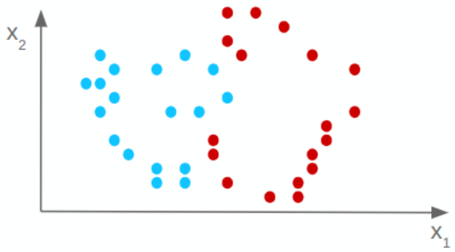
Dataset:

Supervised:

$$\mathcal{D} = \{ \{x_i , y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$$

Unsupervised:

$$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$$



Learning Law

- **Four key parts** to the ‘Machine Learning’ process:

Model f:

E.g. Neural network with parameters Φ

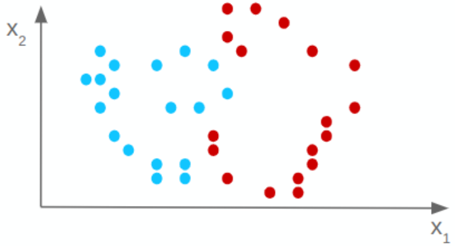
Dataset:

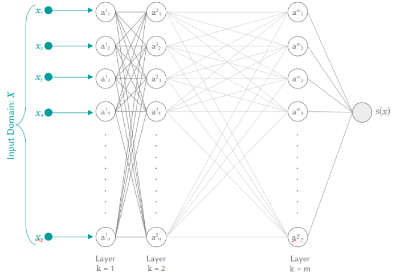
Supervised:

$\mathcal{D} = \{ \{x_i , y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$

Unsupervised:

$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$





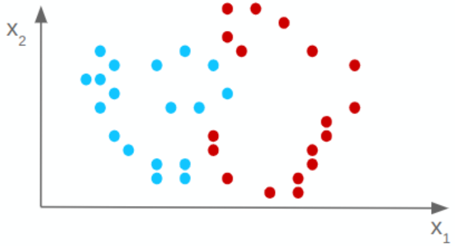
Learning Law

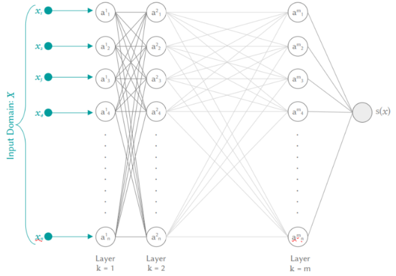
- **Four key parts** to the 'Machine Learning' process:

Model f:
E.g. Neural network with parameters Φ

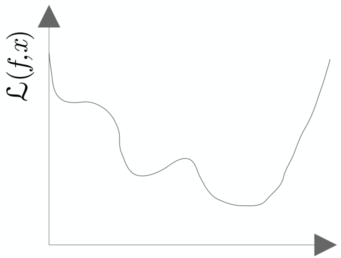
Dataset:
Supervised:
$\mathcal{D} = \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$

Loss/Cost Function
$\mathcal{L}(f,x)$:
Function for calculating agreement between prediction of model and implicit data distribution





$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Learning Law - Recipe

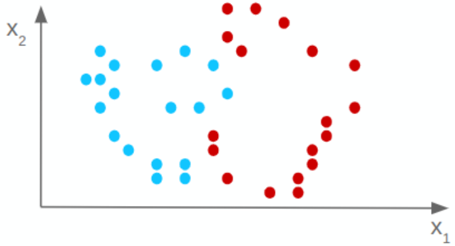
- **Four key parts** to the ‘Machine Learning’ process:

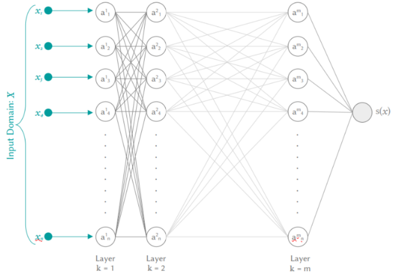
Dataset:
Supervised:
$\mathcal{D} = \{\ \{x_i\ ,\ y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$\mathcal{D} = \{\ \{x_i\} \in \mathbb{R}^X\ \}_N$

Model f:
E.g. Neural network with parameters Φ

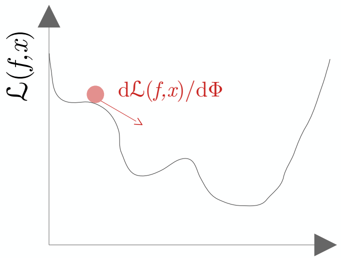
Loss/Cost Function $\mathcal{L}(f,x)$:
Function for calculating agreement between prediction of model and implicit data distribution

Optimise parameters Φ:
Optimisation algorithm





$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Brief Statistics Review

Frequentist Statistics

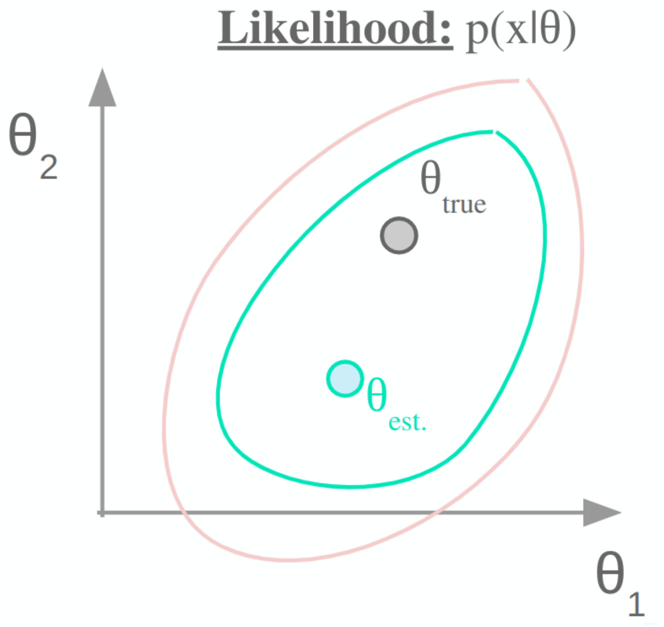
- **Probability** is attributed only to the data **x,** meaning probability of outcomes is obtained by repeatable experiments:

$$P(\mathbf{x} = x) = \lim_{n \to \infty} \frac{n_x}{N}$$

- **Conditional probability**, the probability of an outcome **(x**$=x$**)** conditioned on the occurance of another random process **(y**$=y$**):**

$$P(\mathbf{x} = x | \mathbf{y} = y) = \frac{P(\mathbf{x} = x, \mathbf{y} = y)}{P(\mathbf{x} = x)}$$

- **Important** to note that frequentist statistics assumes that the data is drawn from p(**x**$=x$ | θ), with a set of parameters that characterise the underlying true distribution θ.



Frequentist Statistics

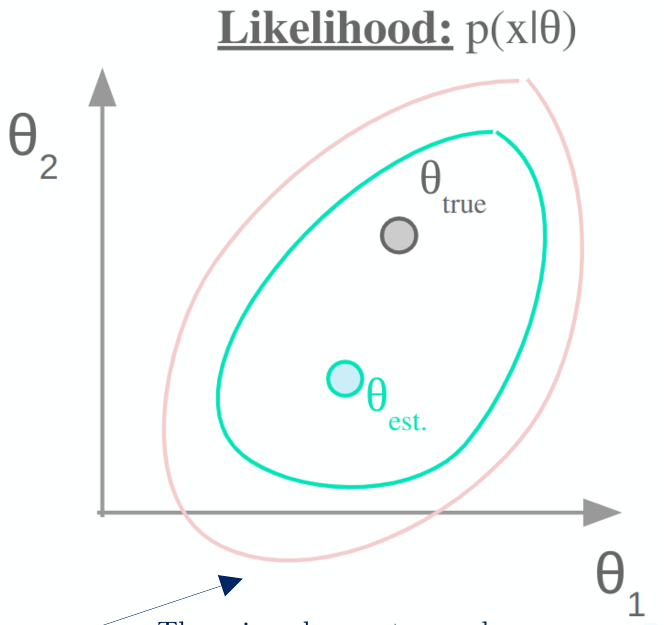
- **Probability** is attributed only to the data **x**, meaning probability of outcomes is obtained by repeatable experiments:

$$P(\mathbf{x} = x) = \lim_{n \to \infty} \frac{n_x}{N}$$

- **Conditional probability**, the probability of an outcome $(\mathbf{x}{=}x)$ conditioned on the occurance of another random process $(\mathbf{y}{=}y)$:

$$P(\mathbf{x} = x | \mathbf{y} = y) = \frac{P(\mathbf{x} = x, \mathbf{y} = y)}{P(\mathbf{x} = x)}$$

- **Important** to note that frequentist statistics assumes that the data is drawn from $p(\mathbf{x}{=}x \mid \theta)$, with a set of parameters that characterise the underlying true distribution θ.



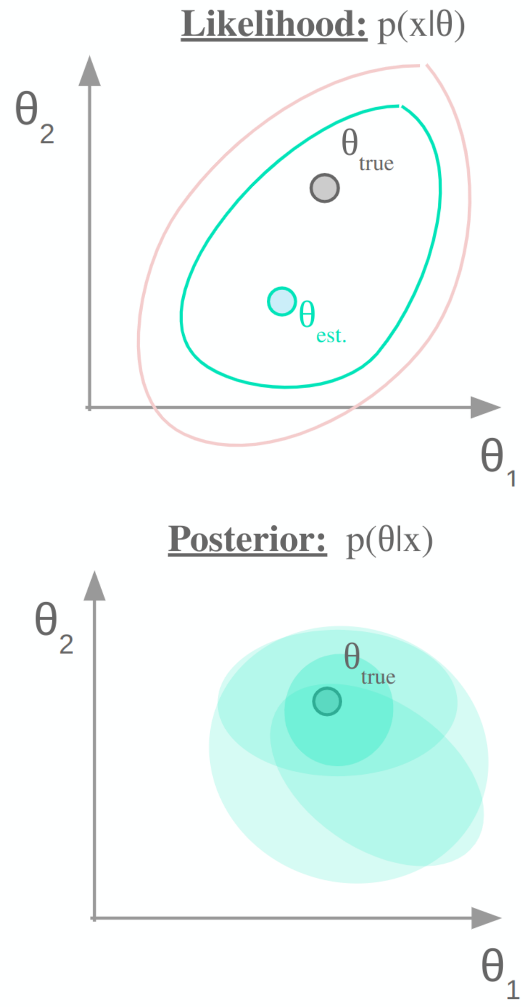
There is only one true value of θ_{true}. The goal is to estimate it ~ $\theta_{est.}$

Bayesian Statistics

- **Probability** is a degree of belief meaning that the observed data **x,** is not necessarily defined by repeatability:

$$P(H|\mathbf{x}=x) = \frac{P(\mathbf{x}=x|H)P(H)}{\underbrace{\int P(\mathbf{x}=x|H)P(H)dH}_{\text{Normalise over all possible hypotheses} \sim \text{marginal probability}}}$$

$$= \frac{P(\mathbf{x}=x|H)P(H)}{P(\mathbf{x}=x)}$$



Estimators

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

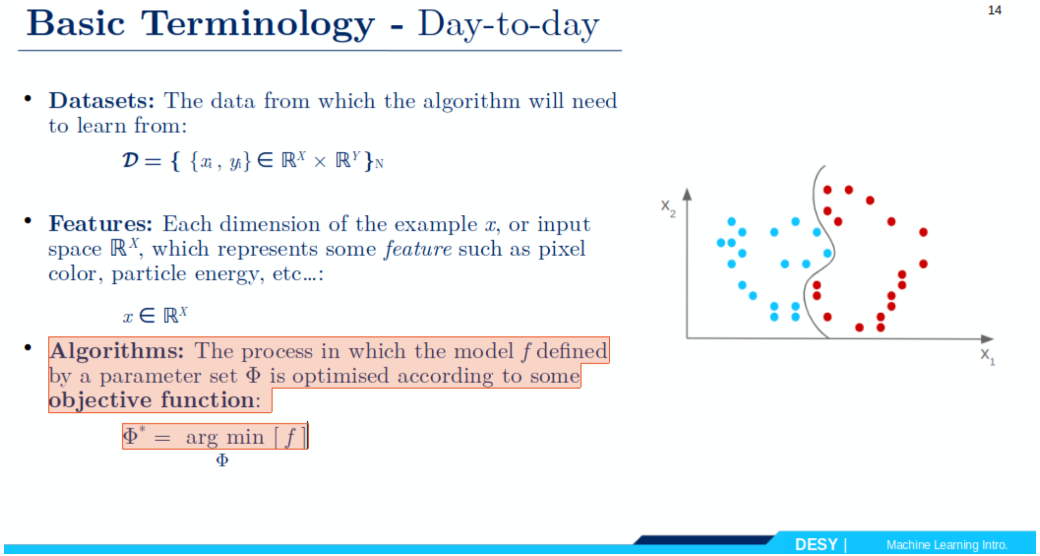
$$\hat{\theta} = g(\{x\}_m)$$

Estimators

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

Recall on slide 14 of this lecture that from a mathematical perspective a ML model is the set of optimal parameter that match **data** *and* **predictions***!*



Estimators

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

- Is there a recipe for extracting from the data the optimal parameters or **best estimator?**

Estimators

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

- Is there a recipe for extracting from the data the optimal parameters or **best estimator?**

→ **Maximum Likelihood Method:**
General approach to estimating the point estimator

$$\hat{\theta} = \arg\max_{\theta} \; p(\{x\}_m|\theta)$$
$$= \arg\max_{\theta} \prod_{i}^{m} p(\; x \;|\theta)$$

Log-likelihood is preferred for numerical stability:

$$-\ln(\mathcal{L}(\theta|\{x\}_m)) = -\sum_{i}^{m} \ln p(x_i|\theta)$$

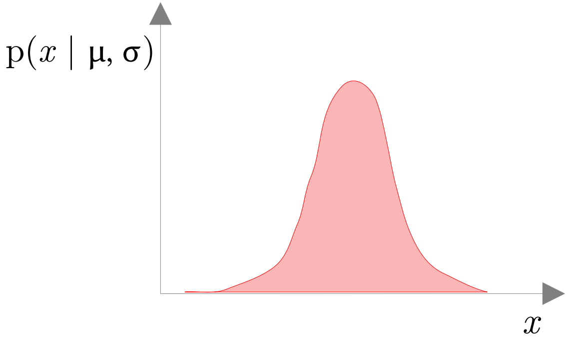
Guassian Estimator Example

Maximum Likelihood Method:

In a counting experiment, *model* data as Gaussian distributed:

$$p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

Model



Guassian Estimator Example

Maximum Likelihood Method:

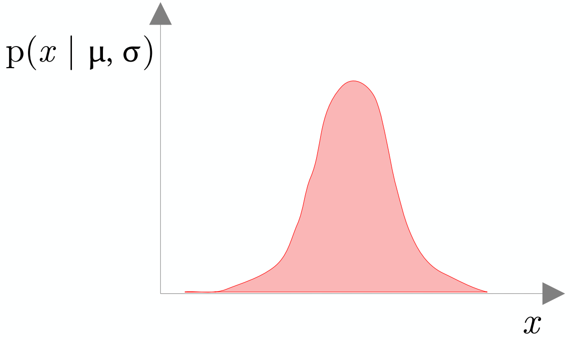
In a counting experiment, *model* data as Gaussian distributed:

$$p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

Fill in the maximum likelihood formula:

$$-\ln(\mathcal{L}(\theta|\{x\}_m)) = -\sum_i^m \ln p(x_i|\theta) = -\sum_i^m \ln(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

Model



Guassian Estimator Example

Maximum Likelihood Method:

In a counting experiment, *model* data as Gaussian distributed:

$$p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

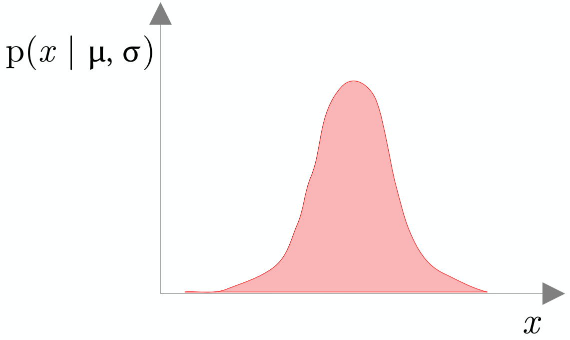
Fill in the maximum likelihood formula:

$$-\ln(\mathcal{L}(\theta|\{x\}_m)) = -\sum_i^m \ln p(x_i|\theta) = -\sum_i^m \ln(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

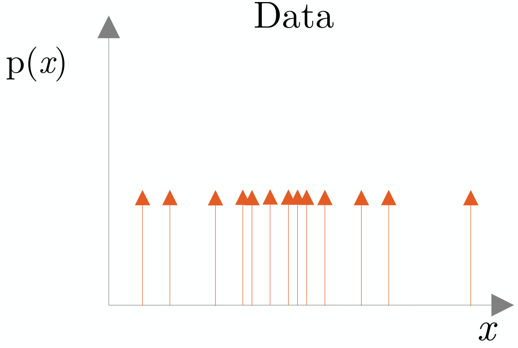
We have some **data** $\mathbf{\{x\}_m}$ which forms an empirical distribution (more on this later), lets **maximise** the likelihood:

$$\frac{\partial}{\partial\theta}(-\ln\mathcal{L}(\theta|\{x\}_m)) \mapsto -\frac{1}{\sigma^2}\sum_i^m(x_i-\mu) = \sum_i^m x_i - m\mu = 0$$

Model



Data



Guassian Estimator Example

Maximum Likelihood Method:

In a counting experiment, *model* data as Gaussian distributed:

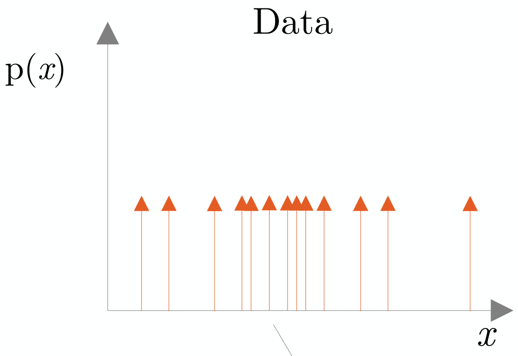
$$p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

Fill in the maximum likelihood formula:

$$-\ln(\mathcal{L}(\theta|\{x\}_m)) = -\sum_i^m \ln p(x_i|\theta) = -\sum_i^m \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

We have some **data** $\mathbf{\{x\}_m}$ which forms an empirical distribution (more on this later), lets **maximise** the likelihood:

$$\frac{\partial}{\partial\theta}(-\ln\mathcal{L}(\theta|\{x\}_m)) \mapsto -\frac{1}{\sigma^2}\sum_i^m (x_i-\mu) = \sum_i^m x_i - m\mu = 0$$



$$\hat{\mu} = \frac{1}{m}\sum_i^m x_i$$

With a bit of algebra we have derived the **sample mean**

Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

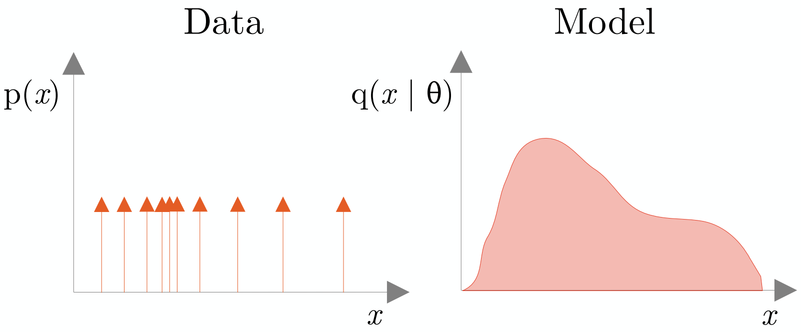
$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$

Key Points:

I) Data with low probability has high information content

II) Independent data samples are additive: $\mathrm{I}(x_1 \oplus x_2) = \mathrm{I}(x_1) + \mathrm{I}(x_2)$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

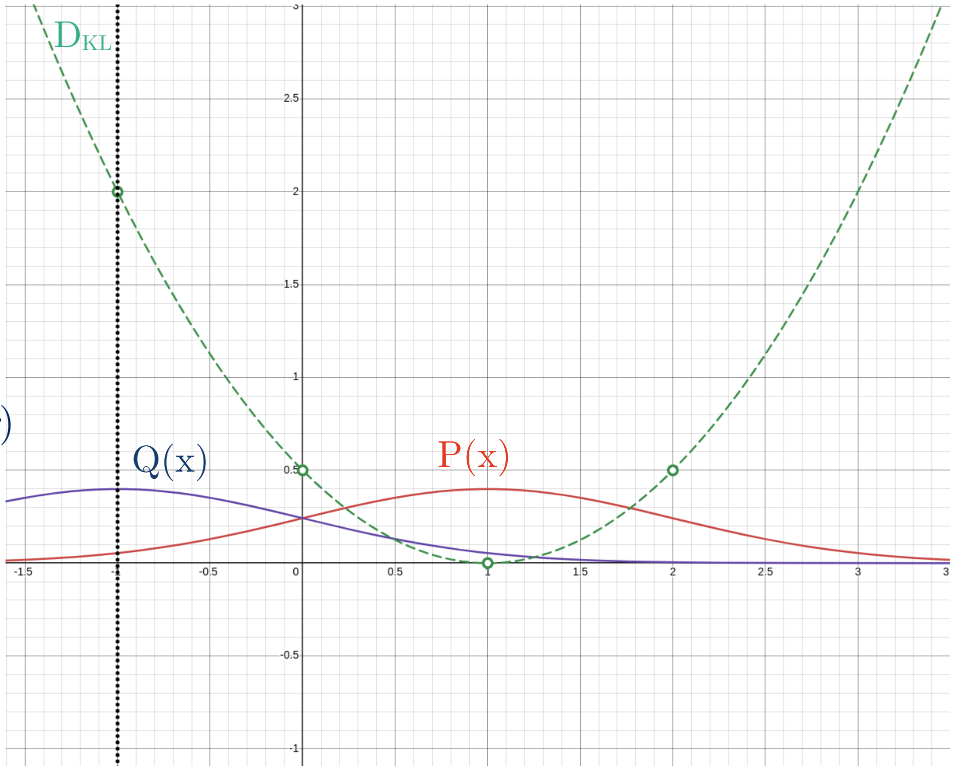
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

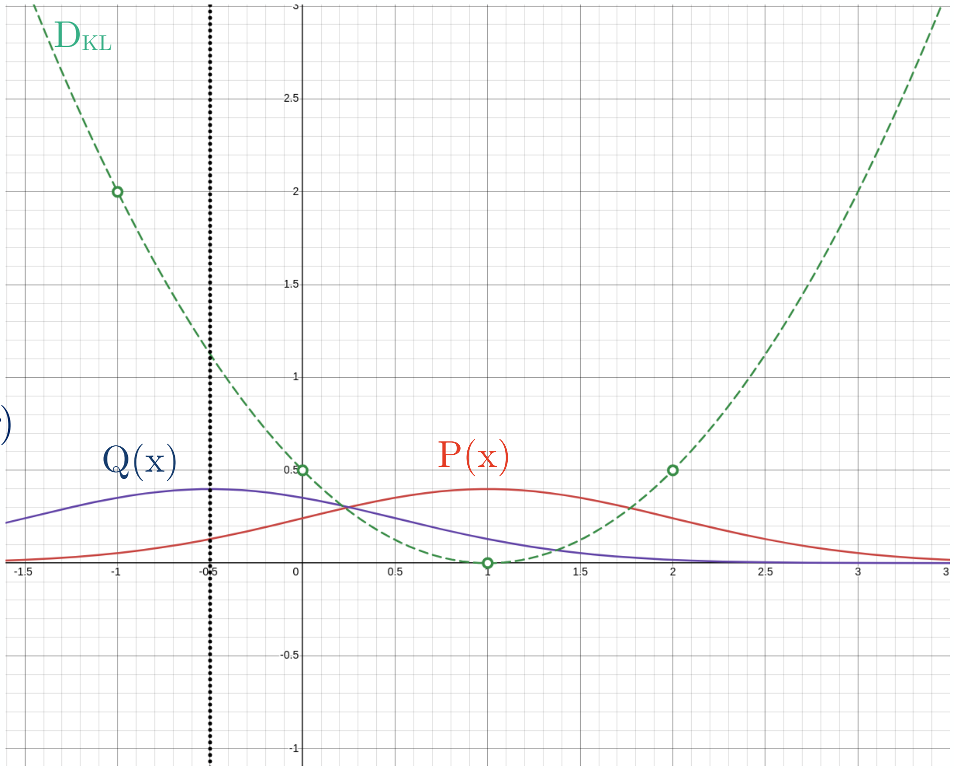
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

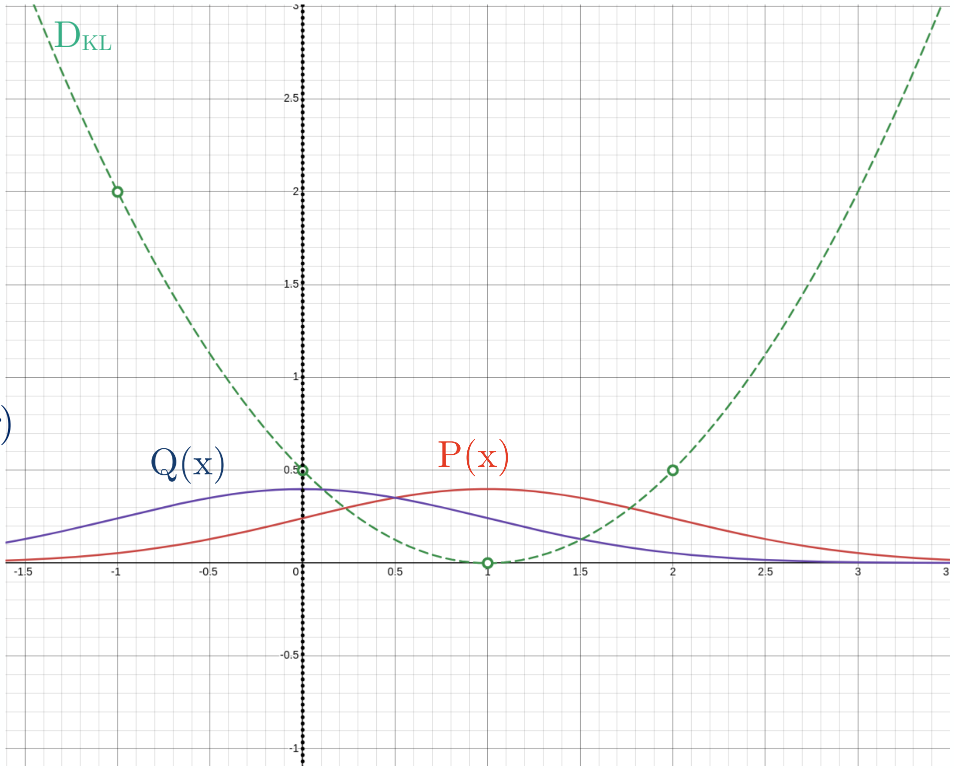
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

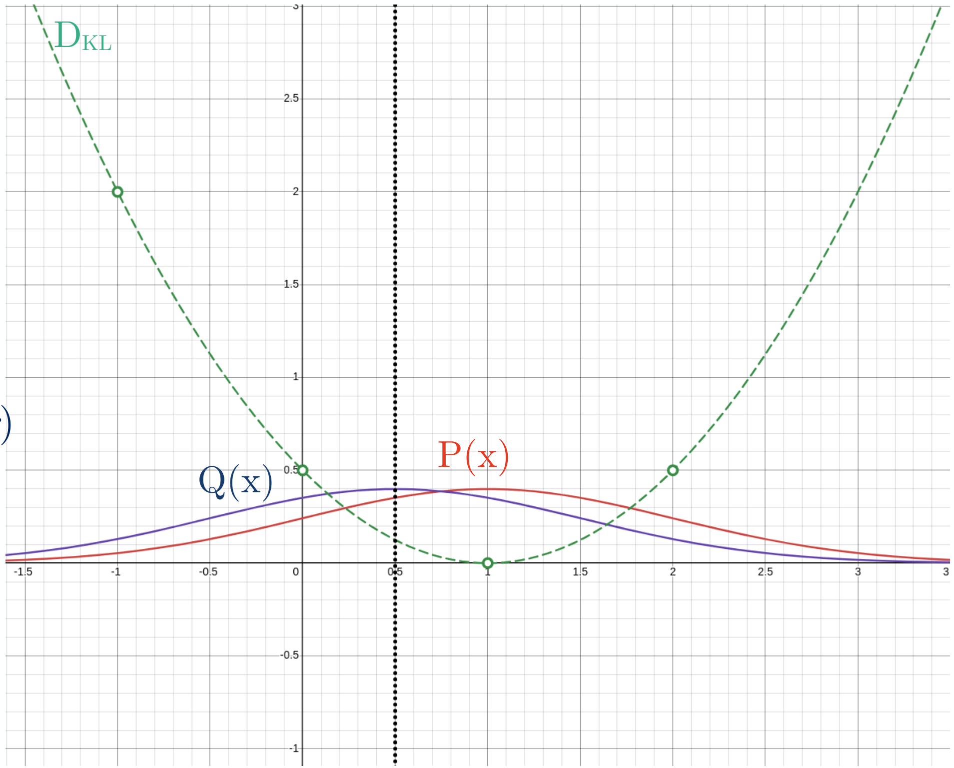
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

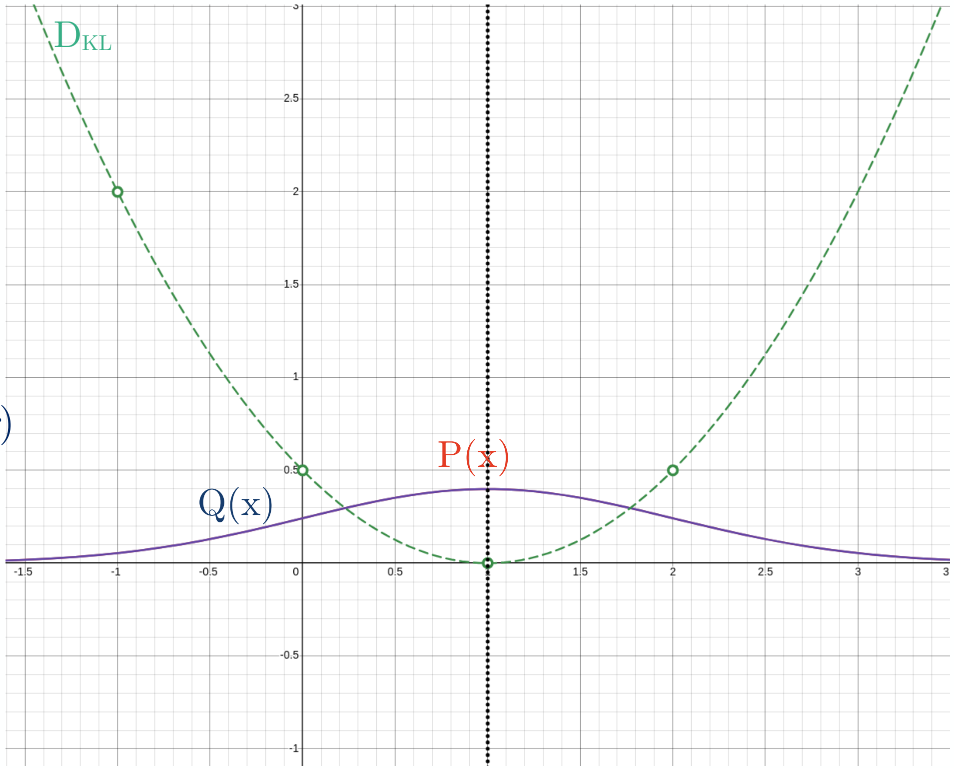
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

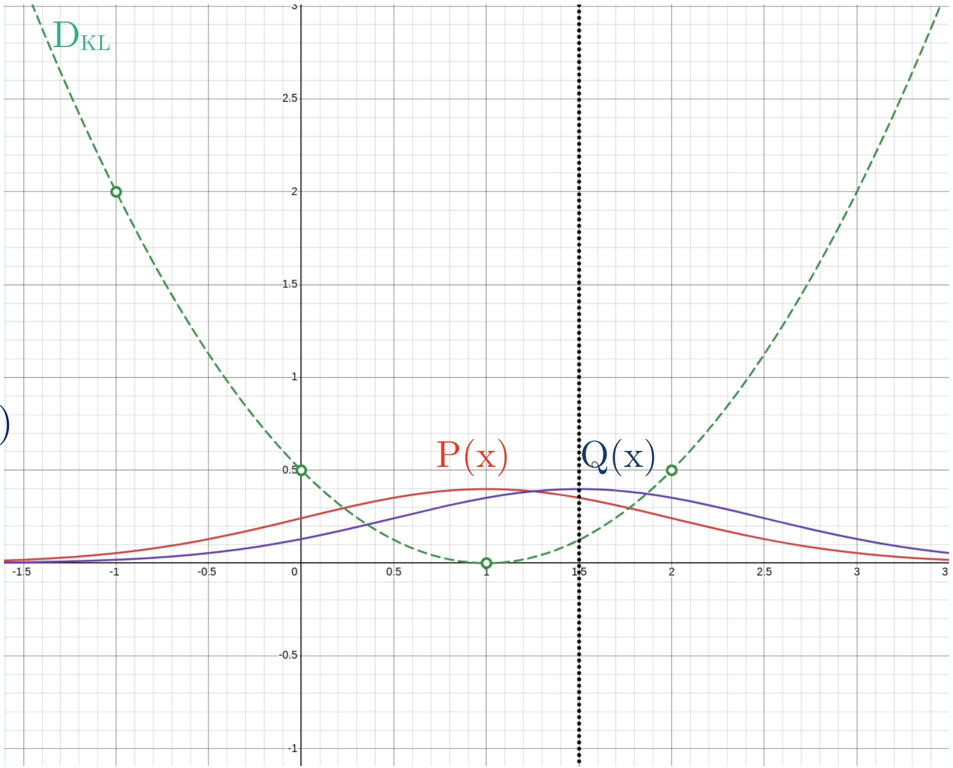
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

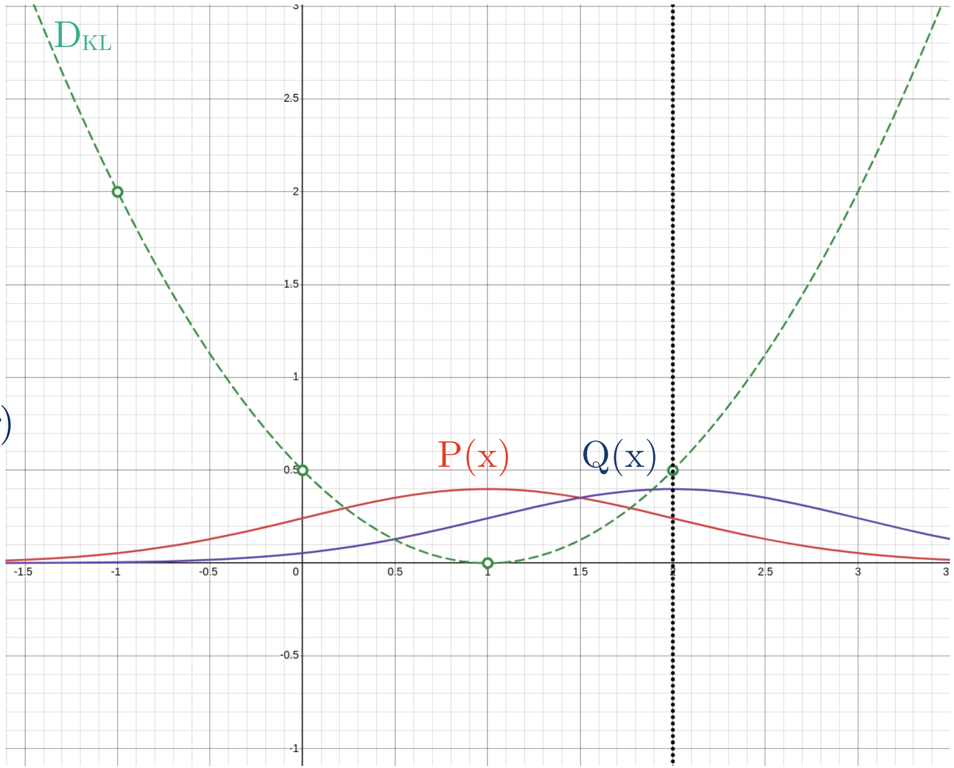
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

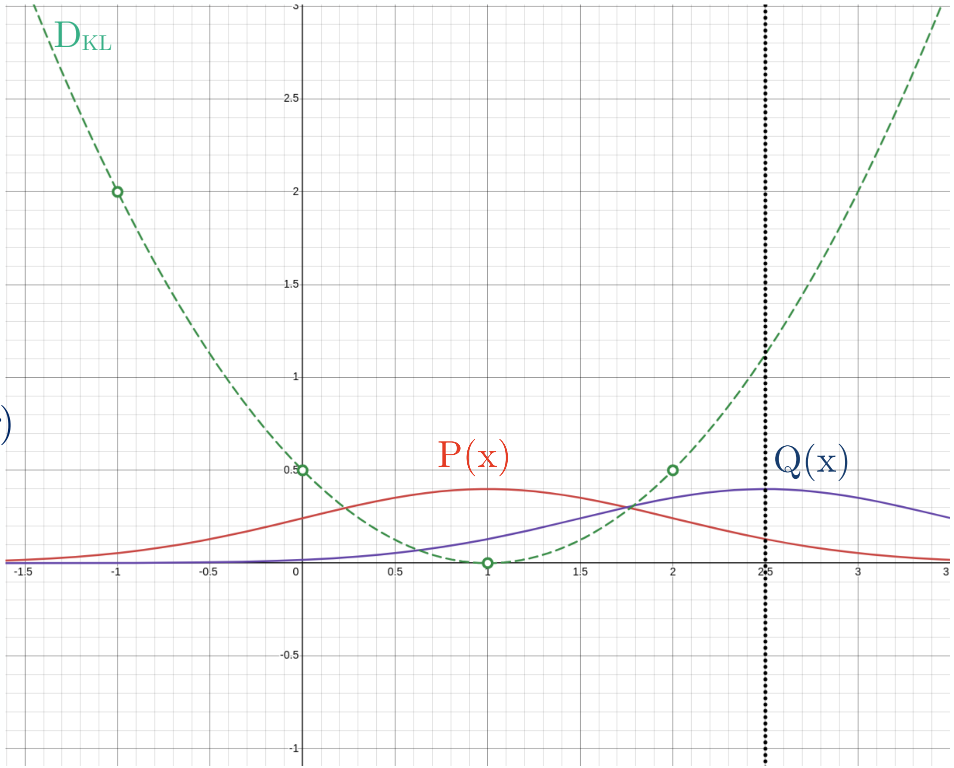
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Information Theory

- **Self-Information**:

$$I(x) = -\ln(P(x))$$

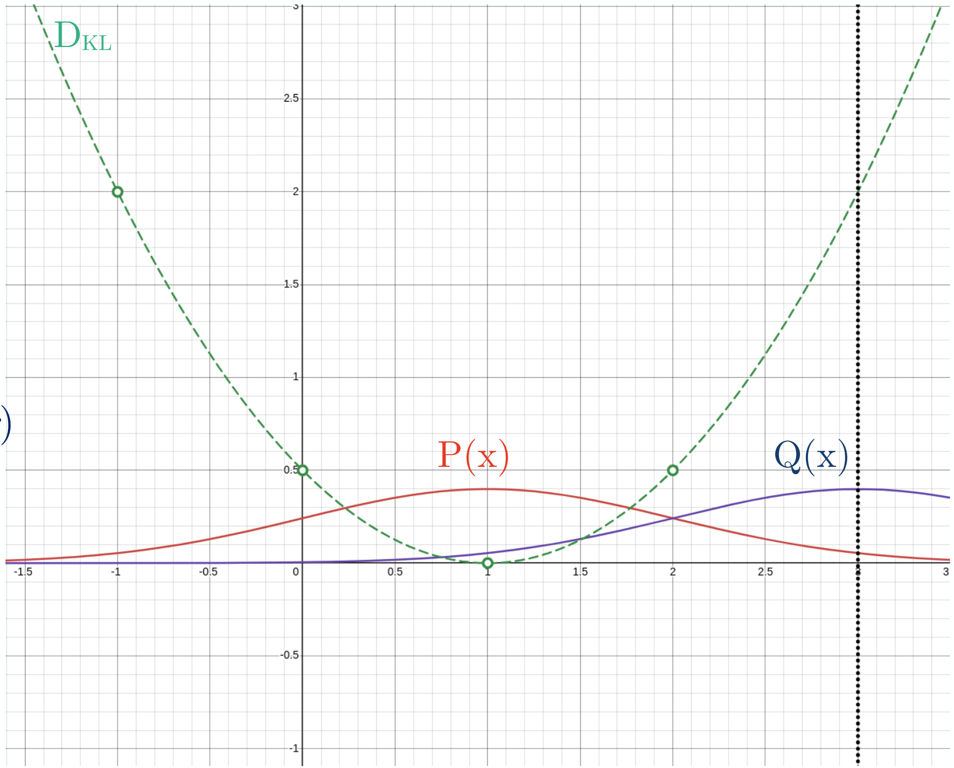
- **Uncertainty** in an entire distribution $P(x)$ – **Shannon Entropy:**

$$H(x) = \mathbb{E}_{x\sim P}\left[I(x)\right] = -\mathbb{E}_{x\sim P}\left[\ln(P(x))\right]$$

- **Relative Entropy** between two distributions $P(x)$ and $Q(x)$:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(\frac{P(x)}{Q(x)}\right)\right]$$

$$= \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \arg\max_{\theta} \; \mathbb{E}_{x \sim p_{\text{data}}} \left[\log(p_{\text{model}}(x|\theta))\right]$$

- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x \sim P} \left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$

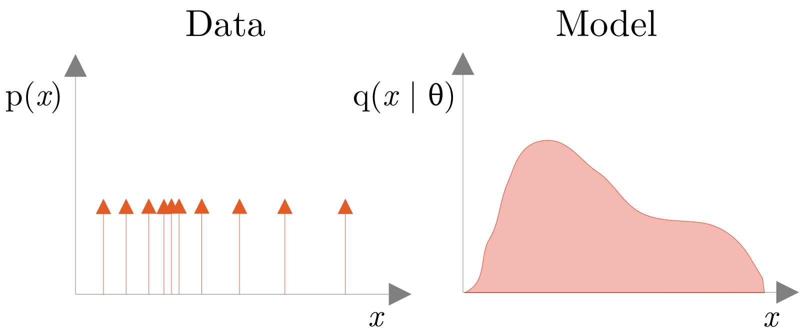
Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \underset{\theta}{\arg\max}\ \mathbb{E}_{x\sim p_{\text{data}}}[\log(p_{\text{model}}(x|\theta))]$$

- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

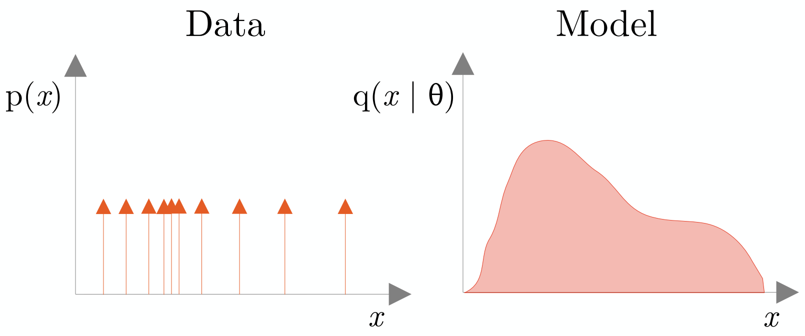
$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{x\sim p_{\text{data}}}[\log(p_{\text{model}}(x|\theta))]$$



- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

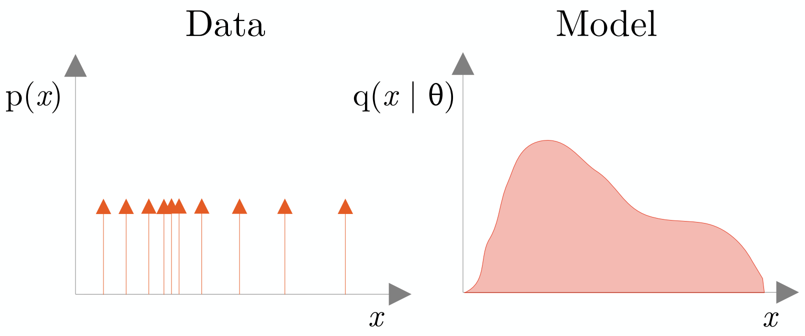
$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}[\ln(P(x)) - \ln(Q(x))]$$

1 $$\hat{\theta} = \arg\min\left[\int p(x)\log(p(x))dx - \int p(x)\log(q(x|\theta))dx\right]$$

Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \underset{\theta}{\arg\max}\ \mathbb{E}_{x\sim p_{\text{data}}}\left[\log(p_{\text{model}}(x|\theta))\right]$$



- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$

1 $$\hat{\theta} = \arg\min\left[\int p(x)\log(p(x))dx - \int p(x)\log(q(x|\theta))dx\right]$$

2 $$\hat{\theta} = \arg\min\left[-\int p(x)\log(q(x|\theta))dx\right]$$

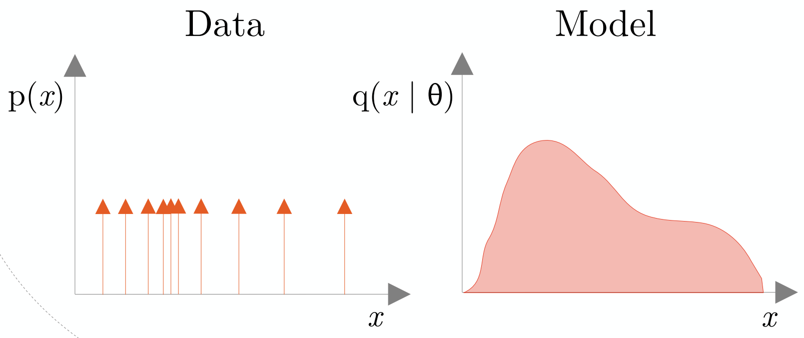
Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \arg\max_{\theta} \; \mathbb{E}_{x\sim p_{\text{data}}}\left[\log(p_{\text{model}}(x|\theta))\right]$$

- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



1 $$\hat{\theta} = \arg\min\left[\int p(x)\log(p(x))dx - \int p(x)\log(q(x|\theta))dx\right]$$

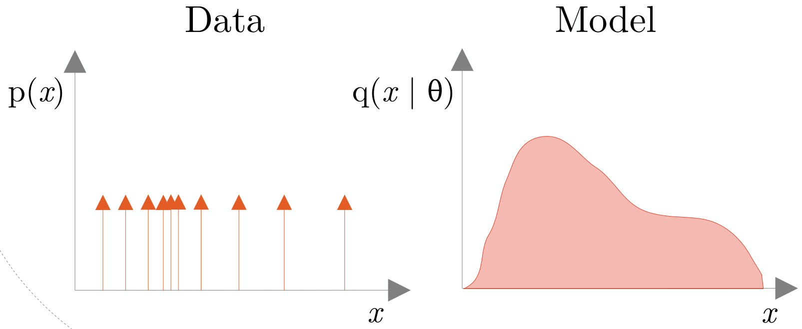
2 $$\hat{\theta} = \arg\min\left[-\int p(x)\log(q(x|\theta))dx\right]$$

$$p(x) = \frac{1}{m}\sum_{i}^{m}\delta(x - x_i)$$

Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \underset{\theta}{\arg\max}\ \mathbb{E}_{x\sim p_{\text{data}}}\left[\log(p_{\text{model}}(x|\theta))\right]$$



- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$

1 $$\hat{\theta} = \arg\min\left[\int p(x)\log(p(x))dx - \int p(x)\log(q(x|\theta))dx\right]$$

3 $$\hat{\theta} = \arg\min\left[-\int\left[\frac{1}{m}\sum_i^m \delta(x - x_i)\right]\log(q(x|\theta))dx\right]$$

$$p(x) = \frac{1}{m}\sum_i^m \delta(x - x_i)$$

Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

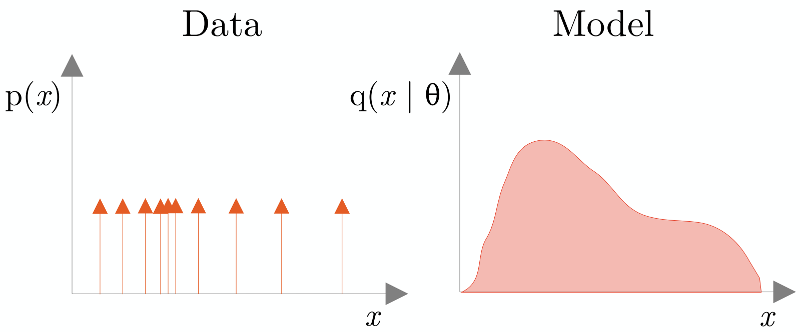
$$\hat{\theta} = \underset{\theta}{\arg\max}\ \mathbb{E}_{x\sim p_{\text{data}}}\left[\log(p_{\text{model}}(x|\theta))\right]$$

- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$

4

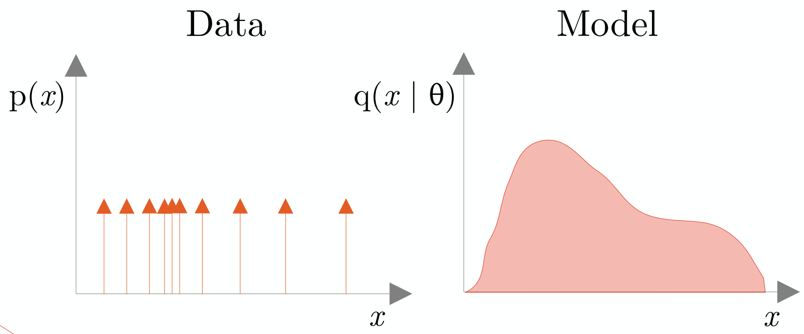
$$\hat{\theta} = \arg\min\left[-\frac{1}{m}\sum_{i}^{m}\log(q(x_i|\theta))\right]$$



Maximum Likelihood Estimator

- **Intuitive understanding** of the maximum likelihood comes from the idea of *distance* between two distribution

$$\hat{\theta} = \arg\max_{\theta} \; \mathbb{E}_{x \sim p_{\text{data}}} [\log(p_{\text{model}}(x|\theta))]$$



- **Kullback-Leibler Divergence** is equivalent to maximum likelihood – lets see how this works:

$$\mathcal{D}_{\text{KL}}(P||Q) = \mathbb{E}_{x \sim P} [\ln (P(x)) - \ln (Q(x))]$$

4 $$\hat{\theta} = \arg\min \left[-\frac{1}{m} \sum_{i}^{m} \log(q(x_i|\theta)) \right]$$

5 $$\hat{\theta} = \arg\min \mathbb{E}_{x \sim p} [-\log(q(x|\theta))]$$

Learning Process & Estimators

Learning Law

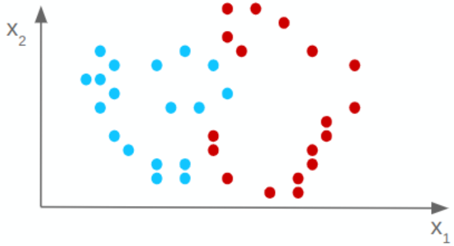
- **Four key parts** to the 'Machine Learning' process:

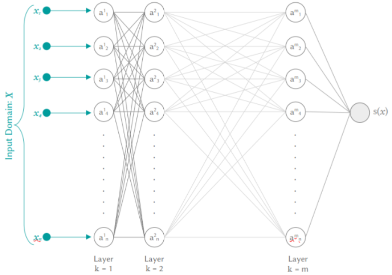
Dataset:
Supervised: $\mathcal{D} = \{ \{x_i , y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised: $\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$

Model f:
E.g. Neural network with parameters Φ

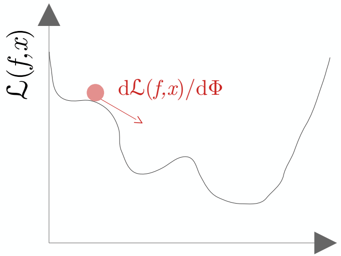
Loss/Cost Function $\mathcal{L}(f,x)$:
Function for calculating agreement between prediction of model and implicit data distribution

Optimise parameters Φ:
Optimisation algorithm





$$\mathcal{D}_{KL}(P||Q) = \mathbb{E}_{x\sim P}\left[\ln\left(P(x)\right) - \ln\left(Q(x)\right)\right]$$



Supervised vs Unsupervised

Dataset:
Supervised:
$\mathcal{D} = \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$

- **Supervised** learning:
  Examples $\mathbf{x} = \{x\}_\mathrm{m}$ paired with targets $\mathrm{y} = \{y\}_\mathrm{m}$ that instruct the algorithm on what to learn

$$p(y|x)$$

- **Unsupervised** learning:
  Examples $\mathbf{x} = \{x\}_\mathrm{m}$ with no targets

$$p(x)$$

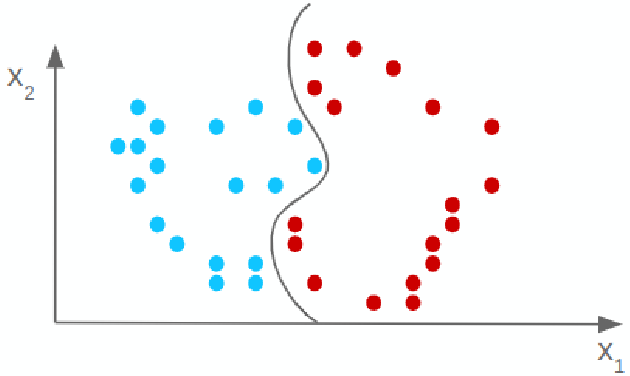
Supervised vs Unsupervised

Dataset:
Supervised:
$\mathcal{D} = \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$

- **Supervised** learning:

  Examples $\mathbf{x} = \{x\}_m$ paired with targets y $= \{y\}_m$ that instruct the algorithm on what to learn

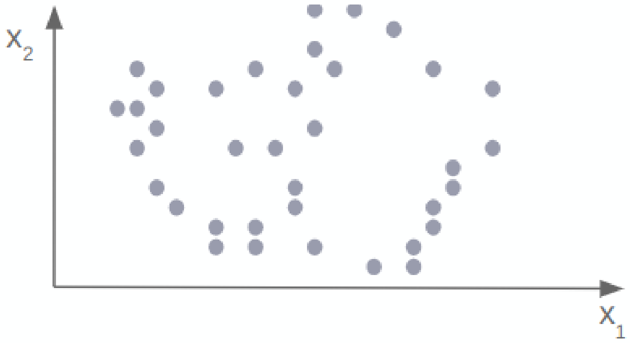
  $p(y|x)$ Conditional Density Estimation



- **Unsupervised** learning:

  Examples $\mathbf{x} = \{x\}_m$ with no targets

  $p(x)$ Density Estimation



Supervised vs Unsupervised

Dataset:
Supervised:
$= \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$= \{ \{x_i\} \in \mathbb{R}^X \}_N$

- **Supervised** learning:
  Examples $\mathbf{x} = \{x\}_m$ paired with targets y $= \{y\}_m$ that instruct the algorithm on what to learn

  $p(y|x)$ Conditional Density Estimation

- **Unsupervised** learning:
  Examples $\mathbf{x} = \{x\}_m$ with no targets

  $p(x)$ Density Estimation

Decomposing the problem into m-1 conditional density estimation problems ~ supervised learning problems

$$p(\{x\}_m) = \prod_i^m p(x_i|\{x\}_{m-i})$$

Supervised vs Unsupervised

Dataset:
Supervised:
$= \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$= \{ \{x_i\} \in \mathbb{R}^X \}_N$

- **Supervised** learning:
  Examples $\mathbf{x} = \{x\}_m$ paired with targets y $= \{y\}_m$
  that instruct the algorithm on what to learn

$p(y|x)$ Conditional Density Estimation

- **Unsupervised** learning:
  Examples $\mathbf{x} = \{x\}_m$ with no targets

$p(x)$ Density Estimation

$$p(\{y\}_m|\{x\}_m) = \frac{p(\{x\}_m, \{y\}_m)}{\sum_{\tilde{y}} p(\{x\}_m, \{\tilde{y}\}_m)}$$

Learning the joint distribution p(x,y) via unsupervised learning techniques, then infer.

Supervised vs Unsupervised

Dataset:
Supervised:
$\mathcal{D} = \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$
Unsupervised:
$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$

- **Supervised** learning:

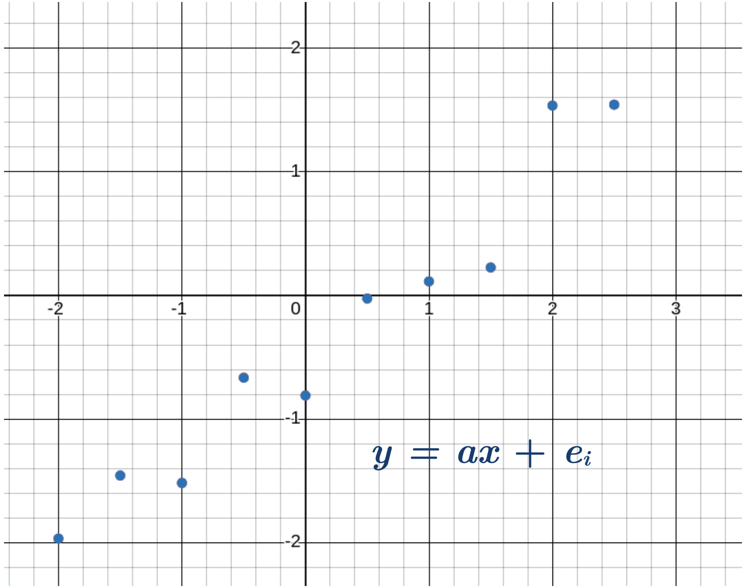
  Examples $\mathbf{x} = \{x\}_m$ paired with targets $y = \{y\}_m$ that instruct the algorithm on what to learn

  $$p(y|x)$$

- **Unsupervised** learning:

  Examples $\mathbf{x} = \{x\}_m$ with no targets

  $$p(x)$$



Linear Regression Example

Model

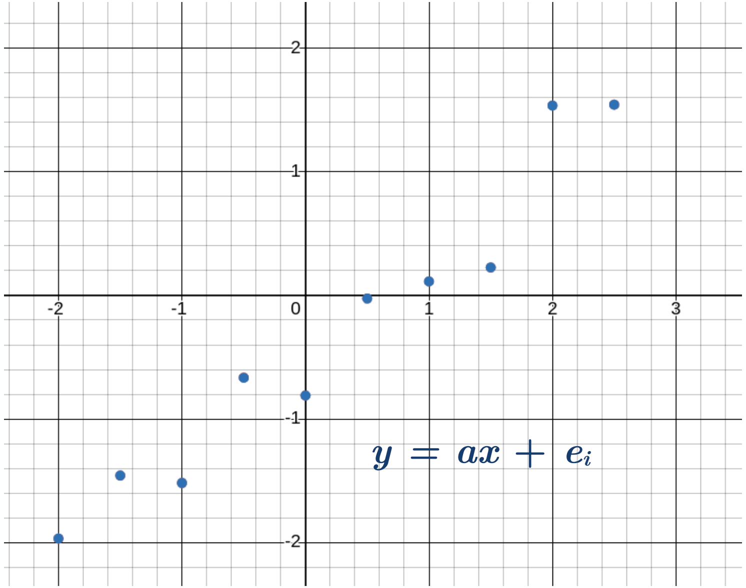
Model f:
E.g. Neural network with parameters Φ

- **Parametric linear model:**

$$\hat{y} = \boldsymbol{w}^T \mathbf{x}$$

- **Configurable parameters of model $\Phi = w$**

$$y = \begin{pmatrix} w_1, & \ldots, & w_n \end{pmatrix} \begin{pmatrix} x_1, \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$



Linear Regression Example

Performance Measure: Loss Function

Loss/Cost Function $\mathcal{L}(f,x)$:
Function for calculating agreement between prediction of model and implicit data distribution

- **Parametric linear model:**

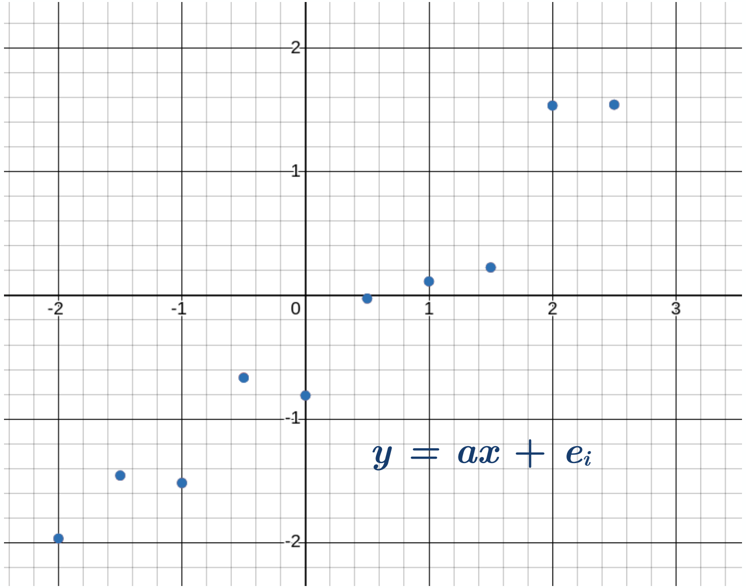
$$\hat{y} = \boldsymbol{w}^T \mathbf{x}$$

- **Configurable parameters of model $\Phi = w$**

$$y = \begin{pmatrix} w_1, & \dots, & w_n \end{pmatrix} \begin{pmatrix} x_1, \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

- **Loss:** Mean squared error

$$\mathcal{L} = \frac{1}{m} \sum_i (\hat{y} - y)^2$$



Linear Regression Example

Performance Measure: Loss Function

Loss/Cost Function $\mathcal{L}(f,x)$:
Function for calculating agreement between prediction of model and implicit data distribution

- **Parametric linear model:**

$$\hat{y} = \boldsymbol{w}^T \mathbf{x}$$

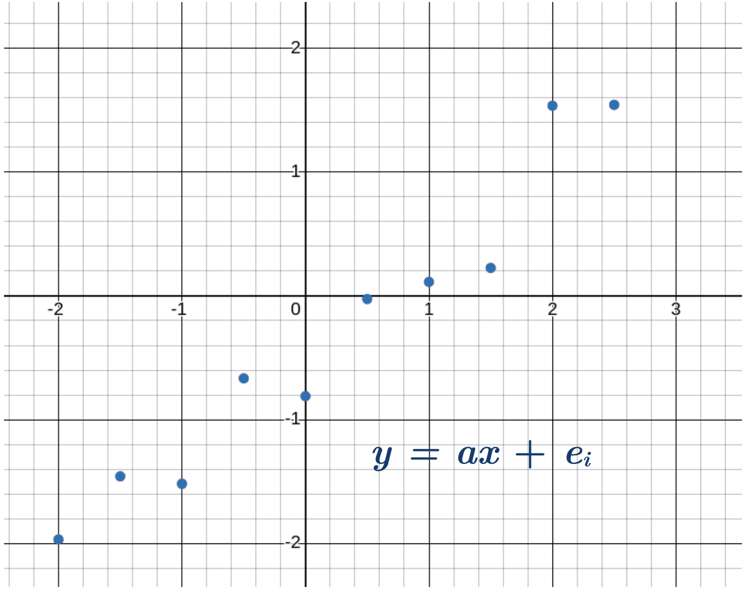
- **Configurable parameters of model** $\Phi = w$

$$y = \begin{pmatrix} w_1, & \ldots, & w_n \end{pmatrix} \begin{pmatrix} x_1, \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

- **Loss:** Mean squared error

$$\mathcal{L} = \frac{1}{m} \sum_i (\hat{y} - y)^2$$

Why the mean squared error?



Linear Regression Example

Maximum Likelihood Approach

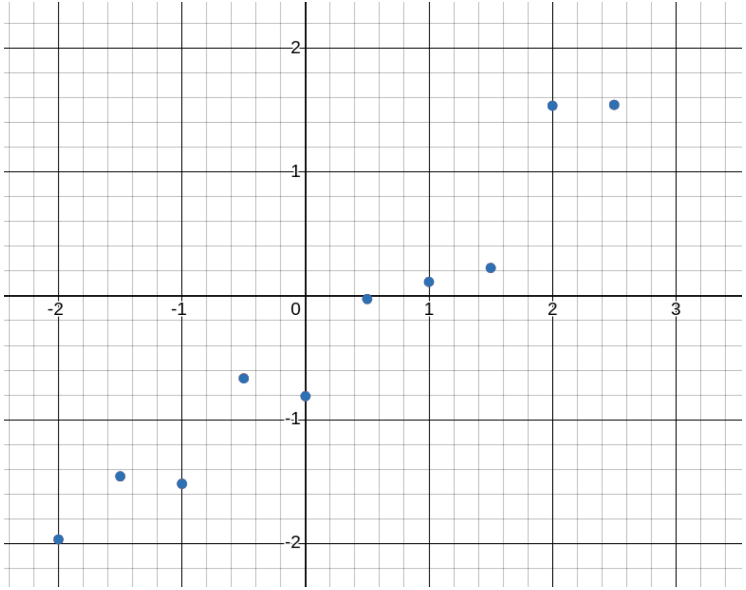
Optimise parameters Φ:
Optimisation algorithm

- I said there was a general likelihood approach to *estimating/learning* parameters:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i}^{m} \ln p(x_i|\theta)$$

- **General MLE** estimate also applies to conditional probability:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i}^{m} \ln \left(p(y_i|x_i, \theta)\right)$$



Linear Regression Example

Optimise parameters Φ:
Optimisation algorithm

Maximum Likelihood Approach

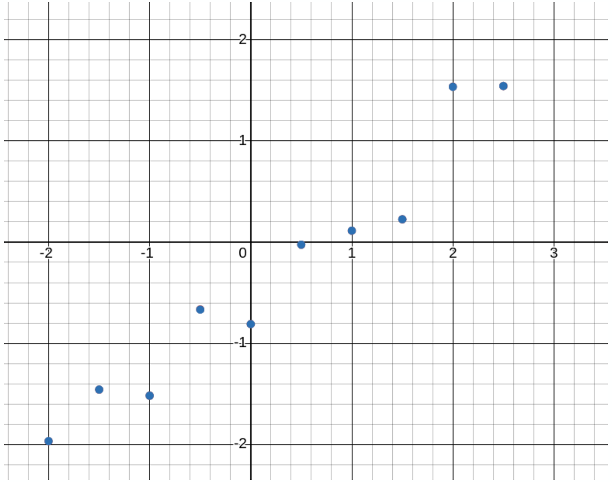
- **Probabilistic view of regression:**

  Data generated with an error term that is gaussian distributed, meaning repeated experiments have some noise:

$$y_i = \alpha x_i + e_i \quad \Rightarrow \quad y_i \sim \mathcal{N}(\alpha x_i, \sigma)$$

$$e_i \sim \mathcal{N}(0, \sigma)$$

Linear Regression Example



Maximum Likelihood Approach

Optimise parameters Φ:
Optimisation algorithm

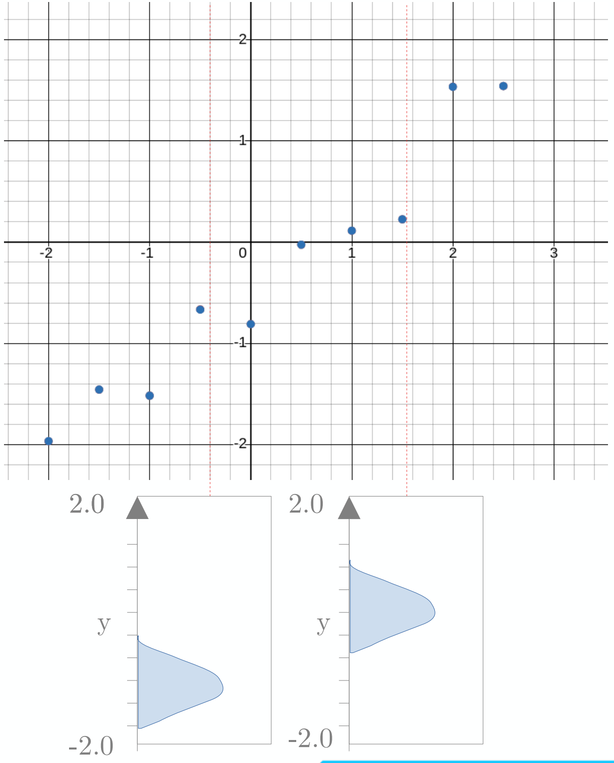
- **Probabilistic view of regression:**

  Data generated with an error term that is gaussian distributed, meaning repeated experiments have some noise:

$$y_i = \alpha x_i + e_i \quad \Rightarrow \quad y_i \sim \mathcal{N}(\alpha x_i, \sigma)$$

$$e_i \sim \mathcal{N}(0, \sigma)$$

Linear Regression Example



Maximum Likelihood Approach

Optimise parameters Φ:

Optimisation algorithm

Probabilistic view of regression:

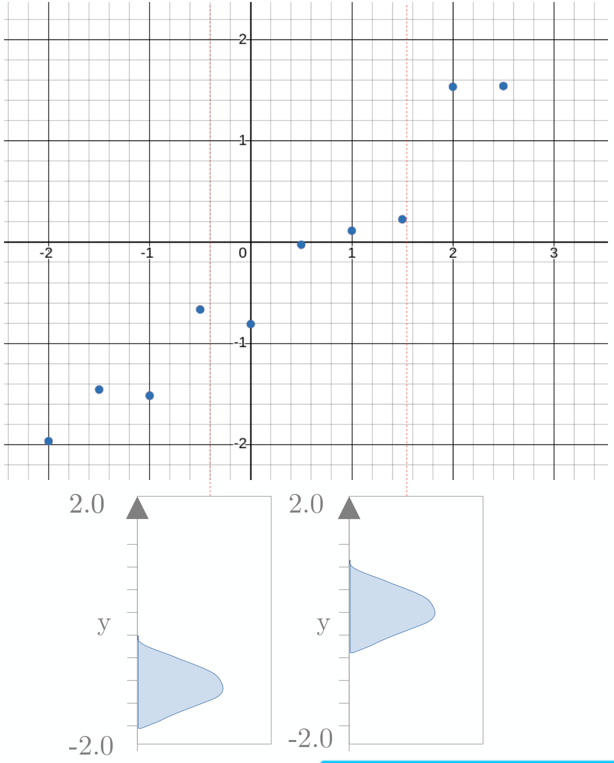
Data generated with an error term that is gaussian distributed, meaning repeated experiments have some noise:

$$y_i = \alpha x_i + e_i \quad \Rightarrow \quad y_i \sim \mathcal{N}(\alpha x_i, \sigma)$$

$$e_i \sim \mathcal{N}(0, \sigma)$$

$$p(y_i|x_i, \alpha) = \mathcal{N}(y_i|\hat{y}_i, \sigma) = \mathcal{N}(y_i|\alpha x_i, \sigma)$$

Linear Regression Example



Maximum Likelihood Approach

Optimise parameters Φ:
Optimisation algorithm

- **Generalise MLE** estimate the conditional probability:

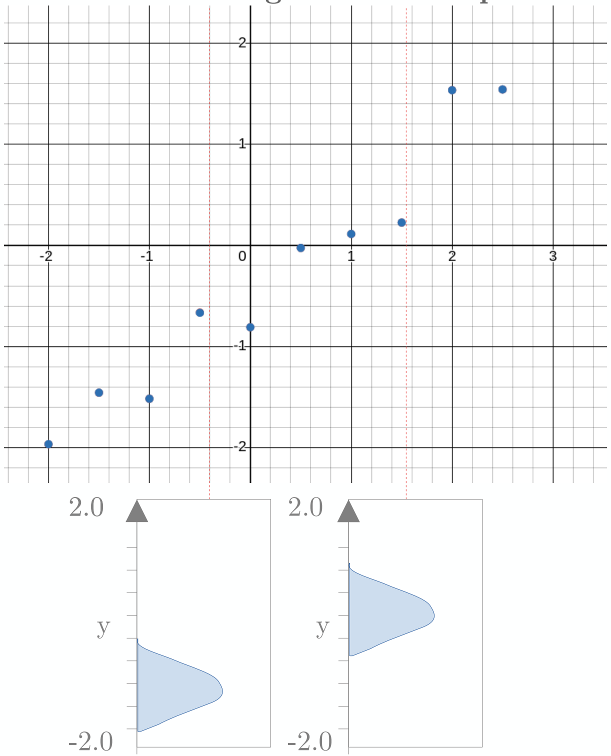
$$\hat{\theta} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(p(y_i|x_i,\theta)\right)$$

$$p(y_i|x_i,\alpha) = \mathcal{N}(y_i|\alpha x_i,\sigma)$$

- **Probabilistic view of regression:**

$$\hat{\alpha} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\alpha x_i-\mu)^2}{2\sigma^2}}\right)$$

Linear Regression Example



Maximum Likelihood Approach

Optimise parameters Φ:
Optimisation algorithm

- **Generalise MLE** estimate the conditional probability:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(p(y_i|x_i,\theta)\right)$$

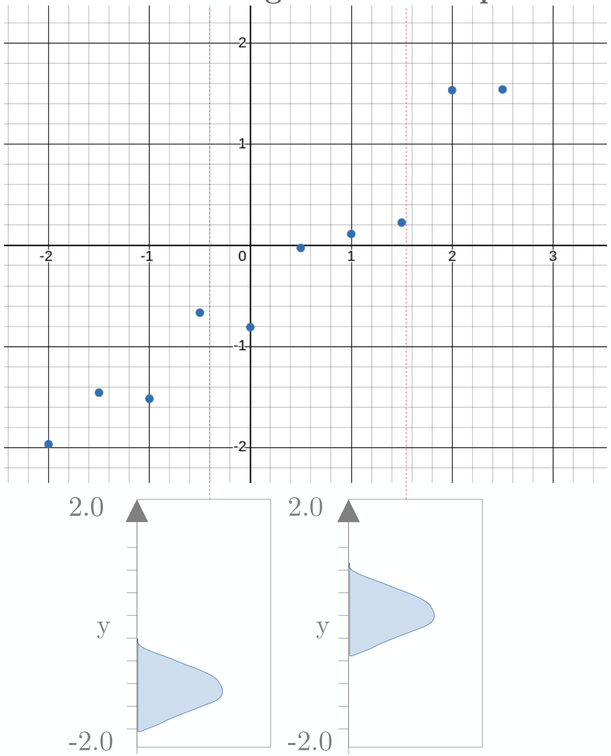
$$p(y_i|x_i,\alpha) = \mathcal{N}(y_i|\alpha x_i,\sigma)$$

- **Probabilistic view of regression:**

$$\hat{\alpha} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\alpha x_i-\mu)^2}{2\sigma^2}}\right)$$

$$= -m\ln(\sigma) - \frac{m}{2}\ln(2\pi) - \sum^{m} \frac{(\alpha x_i-\mu)^2}{2\sigma^2}$$

Linear Regression Example



Maximum Likelihood Approach

Optimise parameters Φ:
Optimisation algorithm

- **Generalise MLE** estimate the conditional probability:

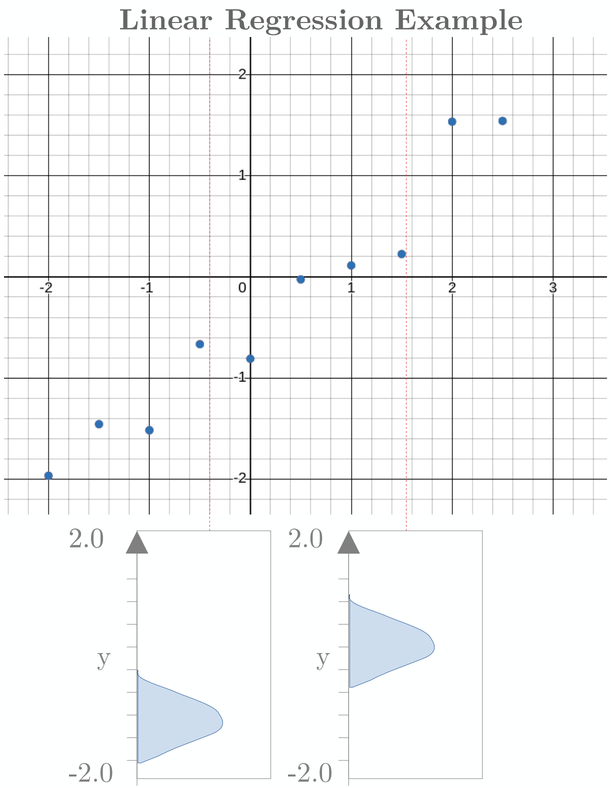
$$\hat{\theta} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(p(y_i|x_i,\theta)\right)$$

- **Probabilistic view of regression:**

$$\hat{\alpha} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\alpha x_i - \mu)^2}{2\sigma^2}}\right)$$

$$= -m\ln(\sigma) - \frac{m}{2}\ln(2\pi) - \sum^{m} \frac{(\alpha x_i - \mu)^2}{2\sigma^2}$$

- **Maximising** the log-likelihood is equivalent to Mean Squared Error minimisation



Gradient Descent

Optimise parameters Φ:

Optimisation algorithm

- **First Order Gradient Descent:**

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\mathrm{train})} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0$$

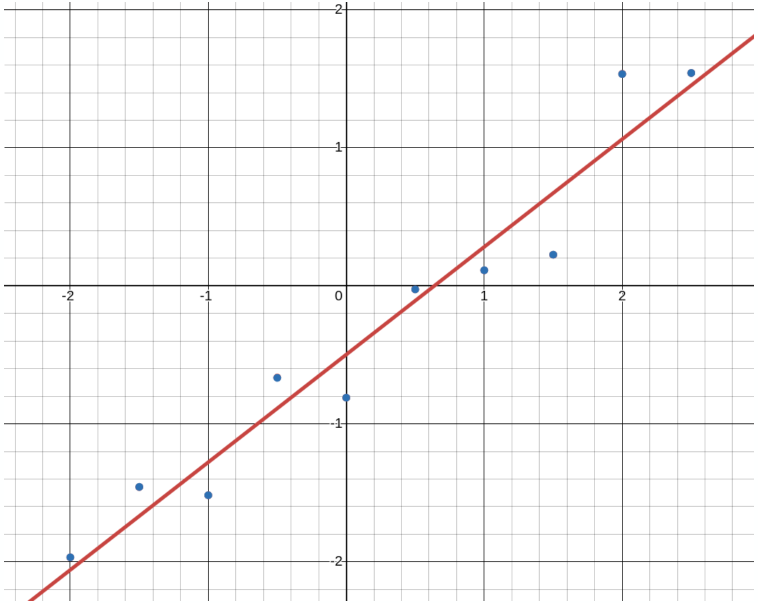
$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} ||\boldsymbol{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}\right)^\top \left(\boldsymbol{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}\right) = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2\boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} + \boldsymbol{y}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})}\right) = 0$$

$$\Rightarrow 2\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} = 0$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})}\right)^{-1} \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})}$$



Linear Regression Example

Capacity & Bias-Variance Trade-Off

- **Capacity** of a model ~ the number of degrees of freedom that can be tuned

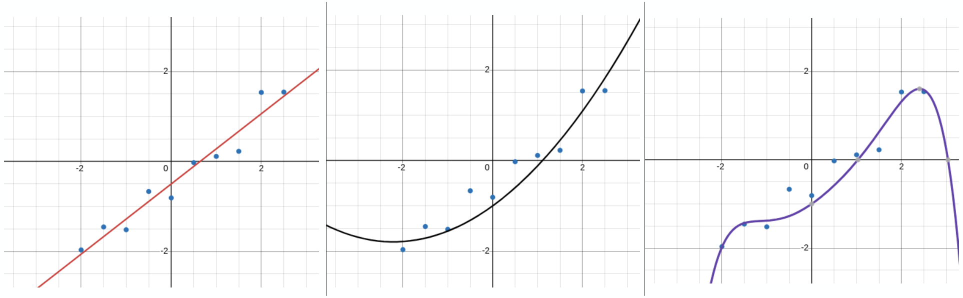
  1st Order Polynomial: $\hat{y} = wx + b$

  2nd Order Polynomial: $\hat{y} = w_1 x + w_2 x^2 + b$

  .

  .

  9th Order Polynomial: $\hat{y} = \sum_{i}^{9} w_i x^i + b$



Capacity & Bias-Variance Trade-Off

- **Capacity** of a model ~ the number of degrees of freedom that can be tuned

    1st Order Polynomial: $\hat{y} = wx + b$

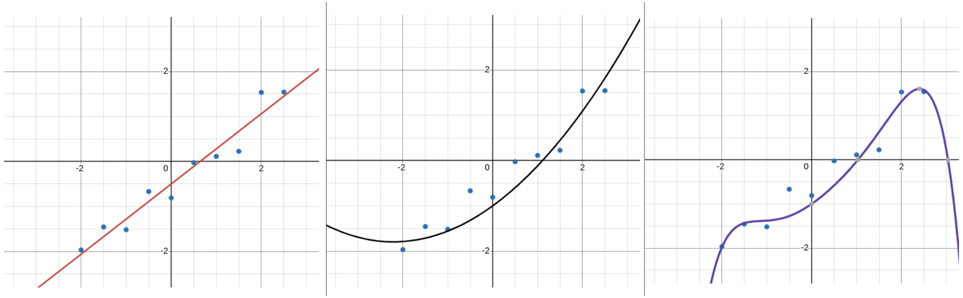
    2nd Order Polynomial: $\hat{y} = w_1 x + w_2 x^2 + b$

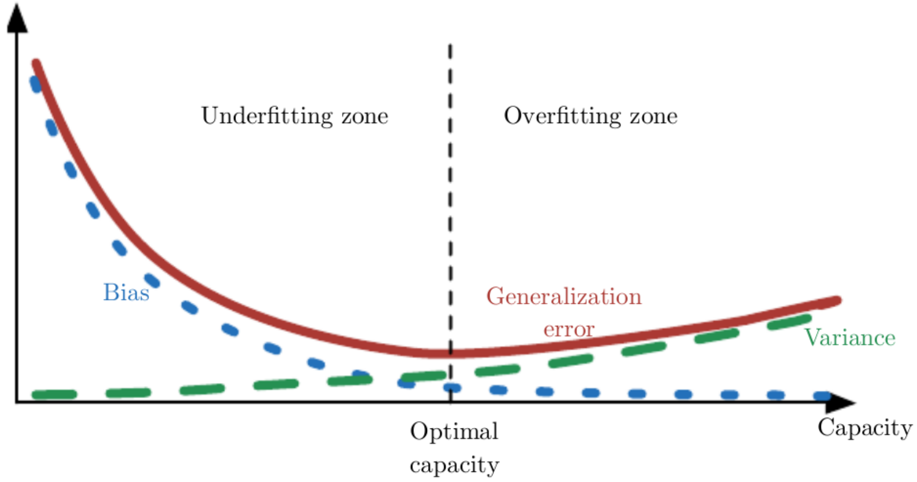
    .

    .

    9th Order Polynomial: $\hat{y} = \sum_{i}^{9} w_i x^i + b$

- **Under/Over-fitting** avoided by matching capacity of model to complexity of the problem





Estimators

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

- **Bias** of an estimator is given by:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

- **Variance** of an estimator:

$$\text{Var}(\hat{\theta})$$

Estimators

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

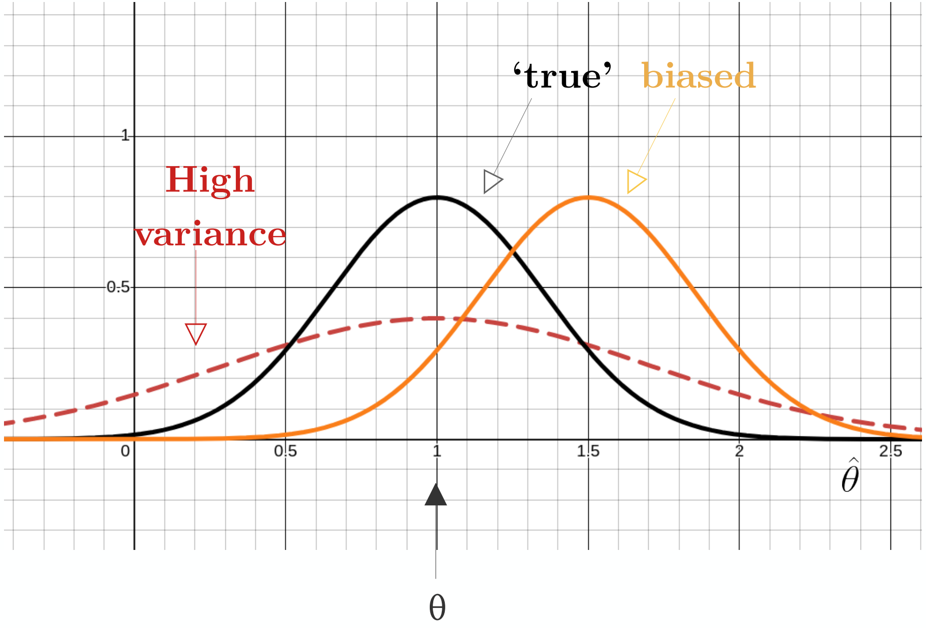
$$\hat{\theta} = g(\{x\}_m)$$

- **Bias** of an estimator is given by:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

- **Variance** of an estimator:

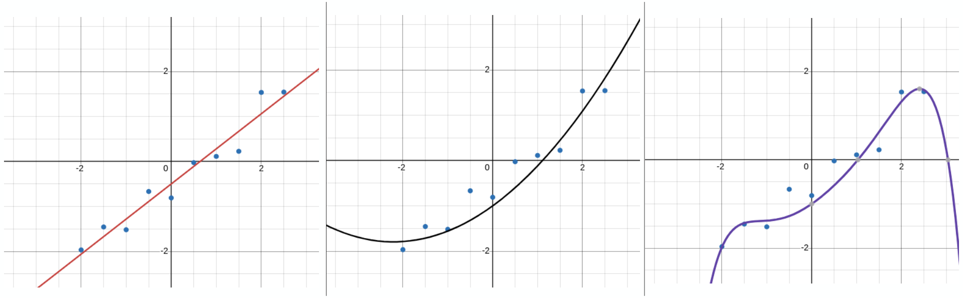
$$\text{Var}(\hat{\theta})$$

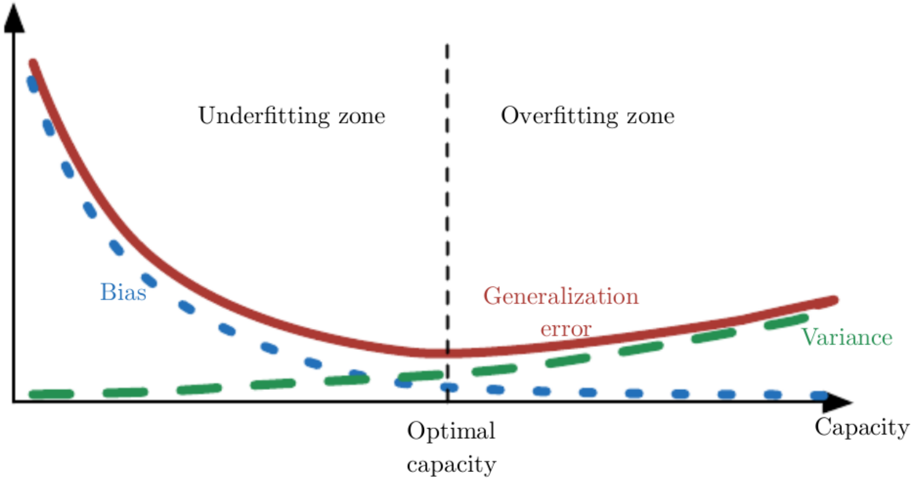


Capacity & Bias-Variance Trade-Off

- **Bias-Variance** trade-off:

$$
\begin{aligned}
\mathrm{MSE}(\hat{\theta},\theta) &= \mathbb{E}\left[(\hat{\theta}-\theta)^2\right] \\
&= \mathbb{E}\left[\hat{\theta}^2 - 2\hat{\theta}\theta + \theta^2\right] \\
&= \mathbb{E}\left[\hat{\theta}^2\right] - 2\mathbb{E}\left[\hat{\theta}\right]\theta + \theta^2 \\
&= \mathrm{Var}(\hat{\theta}) + \mathbb{E}\left[\hat{\theta}\right]^2 - 2\mathbb{E}\left[\hat{\theta}\right]\theta + \theta^2 \\
&= \mathrm{Var}(\hat{\theta}) + \mathrm{bias}(\hat{\theta})^2
\end{aligned}
$$

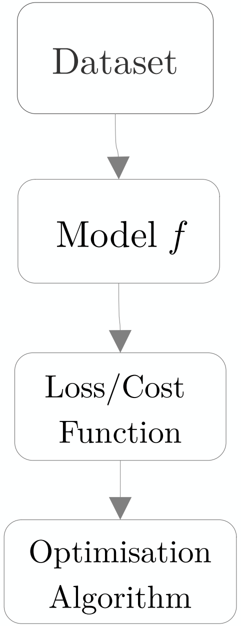




Learning Recipe

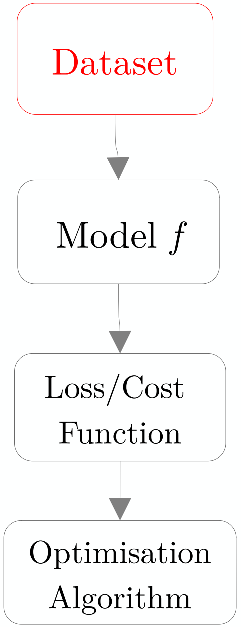
Recap

General recipe for *most* ML algorithms or learning processes:



Recap

General recipe for *most* ML algorithms or learning processes:



Experience

- **Supervised** learning:
  Examples $\mathbf{x} = \{x\}_m$ paired with targets $y = \{y\}_m$ that instruct the algorithm on what to learn

  $p(y|x)$ Conditional Density Estimation

- **Unsupervised** learning:
  Examples $\mathbf{x} = \{x\}_m$ with no targets

  $p(x)$ Density Estimation

Dataset:

Supervised:

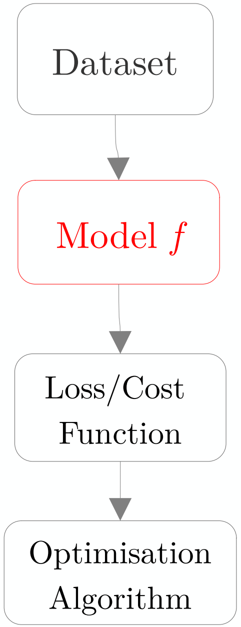
$\mathcal{D} = \{ \{x_i, y_i\} \in \mathbb{R}^X \times \mathbb{R}^Y \}_N$

Unsupervised:

$\mathcal{D} = \{ \{x_i\} \in \mathbb{R}^X \}_N$

Recap

General recipe for *most* ML algorithms or learning processes:



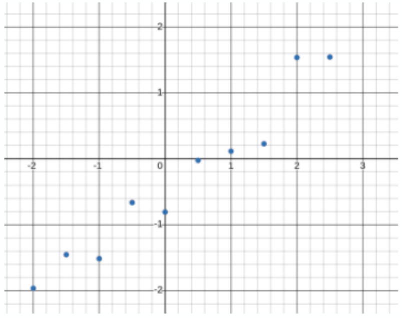
Task

- **Parametric linear model:**

$$\hat{y} = \boldsymbol{w}^T \mathbf{x}$$

- **Configurable parameters of model $\Phi = w$**

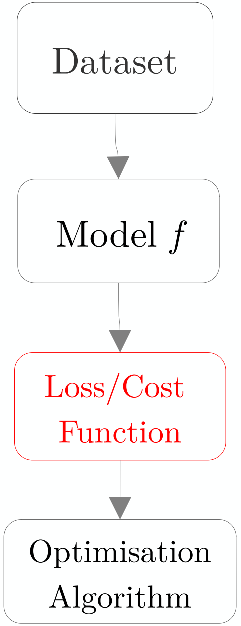
$$y = \begin{pmatrix} w_1, \ \ldots, \ w_n \end{pmatrix} \begin{pmatrix} x_1, \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$



Linear Regression Example

Recap

General recipe for *most* ML algorithms or learning processes:



Performance

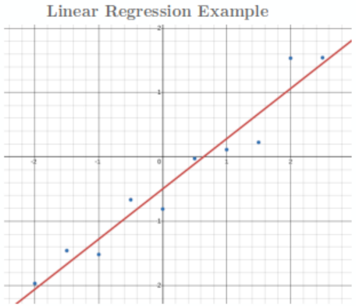
- **Generalise MLE** estimate the conditional probability:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i}^{m} \ln\left(p(y_i | x_i, \theta)\right)$$

- **Probabilistic view of regression:**

$$\hat{m} = \arg\max_{\theta} \sum_{i}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\alpha x_i - y_i)^2}{2\sigma^2}}$$

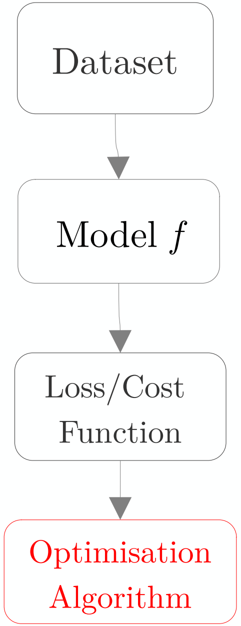
$$= -m\log(\sigma) - \frac{m}{2}\log(2\pi) - \sum_{i}^{m} \frac{(\alpha x_i - \mu)^2}{2\sigma^2}$$



- **Maximising** the log-likelihood is equivalent to MSE minimisation

Recap

General recipe for *most* ML algorithms or learning processes:



- **First Order Gradient Descent:**

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0 \tag{5.6}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} \|\hat{\boldsymbol{y}}^{(\mathrm{train})} - \boldsymbol{y}^{(\mathrm{train})}\|_2^2 = 0 \tag{5.7}$$

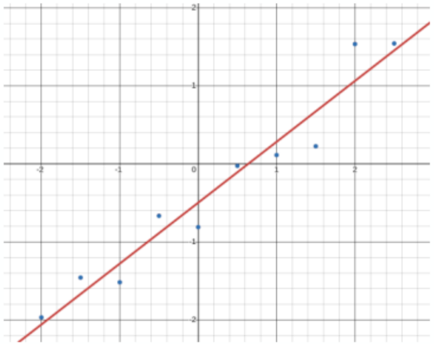
$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} \|\boldsymbol{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}\|_2^2 = 0 \tag{5.8}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}\right)^\top \left(\boldsymbol{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}\right) = 0 \tag{5.9}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2\boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} + \boldsymbol{y}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})}\right) = 0 \tag{5.10}$$

$$\Rightarrow 2\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} = 0 \tag{5.11}$$

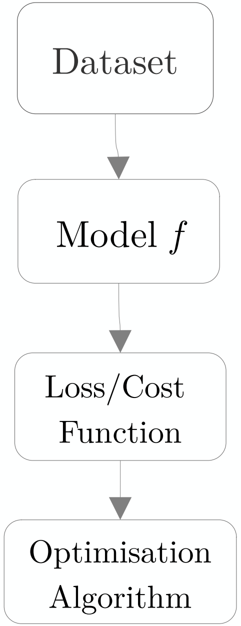
$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})}\right)^{-1} \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} \tag{5.12}$$



Linear Regression Example

Recap

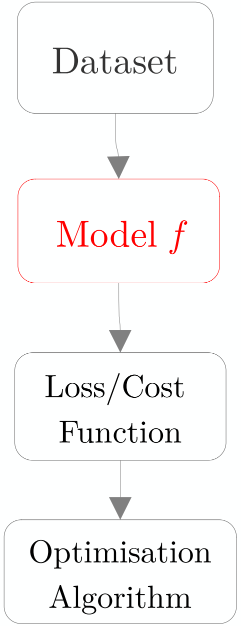
General recipe for *most* ML algorithms or learning processes:



Are we done?

Recap

General recipe for *most* ML algorithms or learning processes:



Are we done?

- How do we change the **family of functions** that a model learns? Scale capacity?

- **Capacity** of a model ~ the number of degrees of freedom that can be tuned

  1^{st} Order Polynomial: $\hat{y} = wx + b$

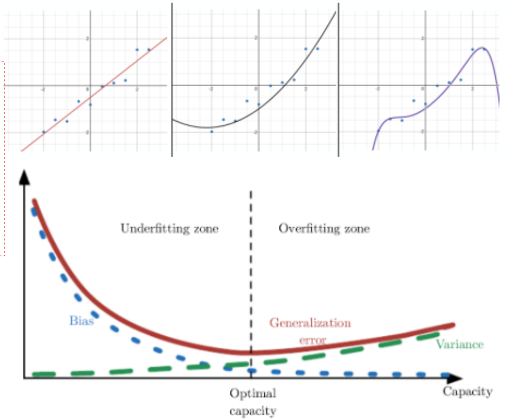
  2^{nd} Order Polynomial: $\hat{y} = w_1 x + w_2 x^2 + b$

  .

  .

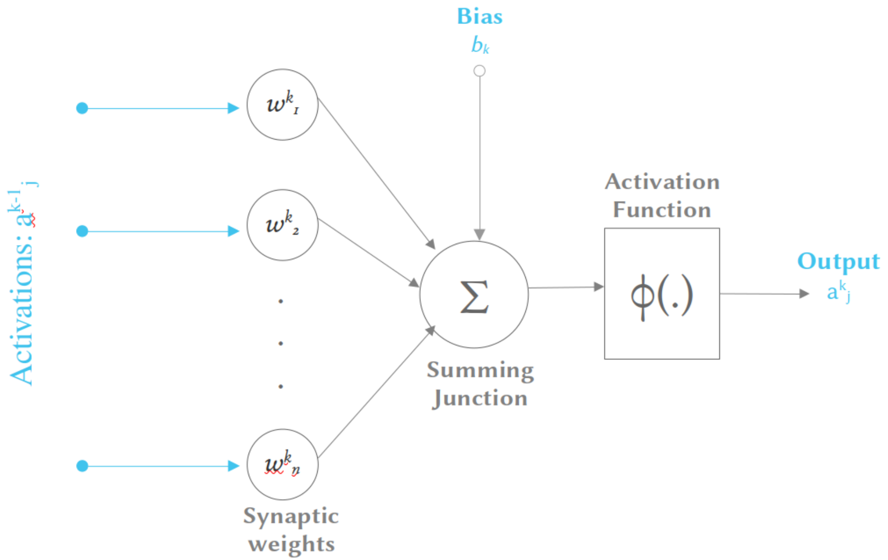
  9^{th} Order Polynomial: $\hat{y} = \sum_i^9 w_i x^i + b$

- **Under/Over-fitting** avoided by matching capacity of model to complexity of the problem



Neural Networks

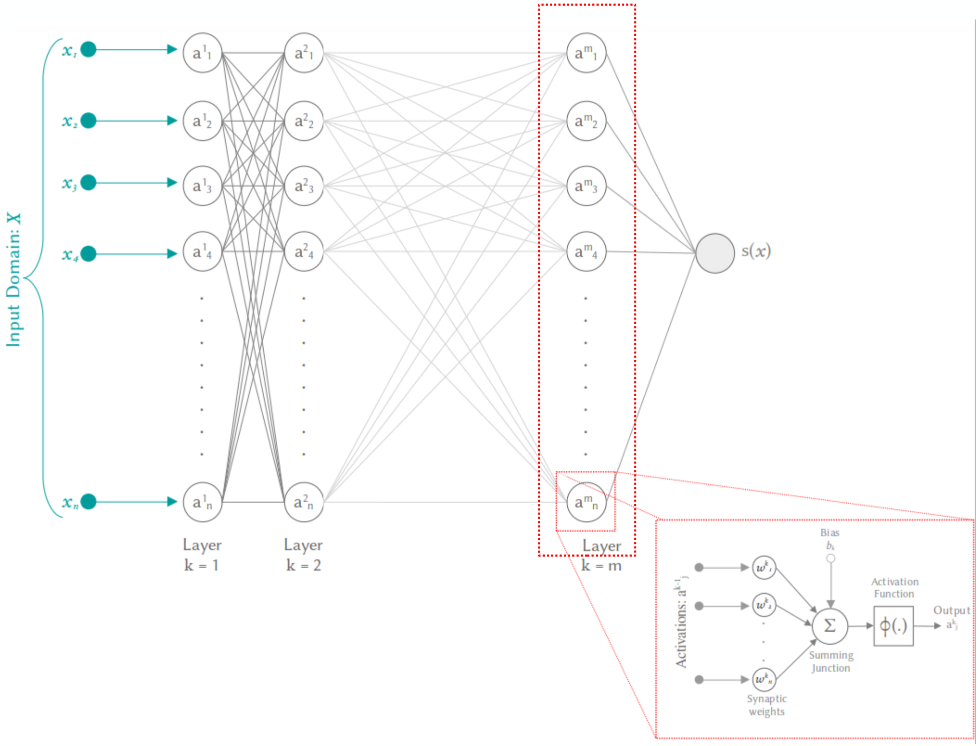
Neuron



- **Building block** of neural networks:

$$a_i^{(k+1)} = \phi\left(\sum_j^n w_{i,j}^{(k)} a_j^{(k)} + b^{(k)}\right)$$

Neuron → Layer



- **Building block** of neural networks:

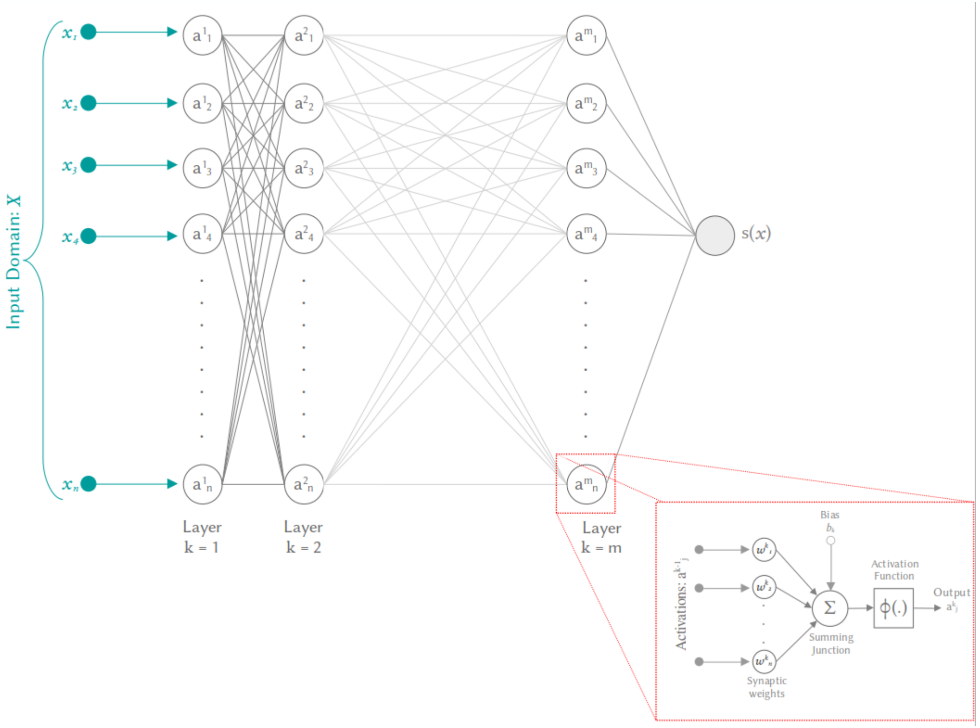
$$a_i^{(k+1)} = \phi\left(\sum_j^n w_{i,j}^{(k)} a_j^{(k)} + b^{(k)}\right)$$

Where, $a^{(k)}_i$ is the activation value, and the $\Phi=\{w^{(k)}_i, b^{(k)}\}$ are the configurable weights & biases

- **Neural Network** layer is the combination of many neurons densely connected (mostly…):

$$\begin{bmatrix} a_0^{(k+1)} \\ \dots \\ a_n^{(k+1)} \end{bmatrix} = \phi\left(\begin{bmatrix} w_0^{(0)} & \dots & w_n^{(0)} \\ \dots & \dots & \dots \\ w_0^{(k)} & & w_n^{(k)} \end{bmatrix} \begin{bmatrix} a_0^{(k)} \\ \dots \\ a_n^{(k)} \end{bmatrix} + \begin{bmatrix} b_0^{(k)} \\ \dots \\ b_n^{(k)} \end{bmatrix}\right)$$

Neural Network

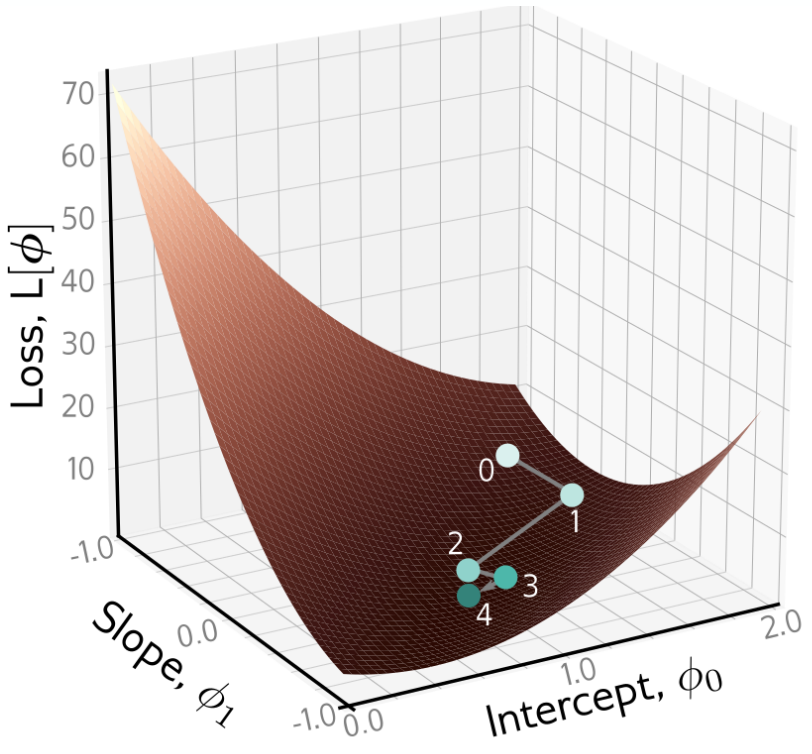


- A **total network** is therefore:

$$\mathbf{a}^{(1)} = \phi\left(\mathbf{W}^{(0)}\mathbf{x} + \beta^{(0)}\right)$$
$$\vdots$$
$$\mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right)$$
$$\vdots$$
$$\mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$

- ϕ is the activation function, which can take many forms
- Optimisation is done via typically gradient based descent algorithms

Stochastic Gradient Descent



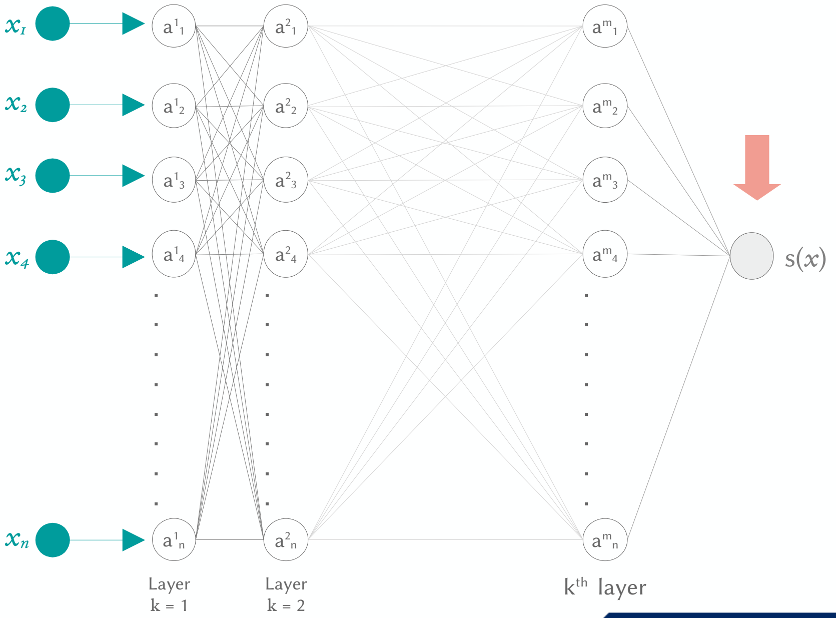
- **Stochastic Gradient Descent** is an adaptation of gradient descent:
  1) Calculate gradients:
  $$\frac{\partial \mathcal{L}}{\partial \Phi} = \begin{bmatrix} \frac{d\mathcal{L}}{\partial \phi_0} \\ \dots \\ \frac{d\mathcal{L}}{\partial \phi_h} \end{bmatrix}$$
  2) Update the parameters in direction that minimises loss:
  $$\phi_i^{(t+1)} \leftarrow \phi_i^{(t)} - \alpha . \sum_{j \in \mathcal{B}_t} \frac{\partial \mathcal{L}_j}{\partial \phi_{i,j}^{(t)}}$$
- **Batch based** training with iterative updates to the gradients based on an **epoch** 't', but the key goal is to learn:

$$\frac{\partial \mathcal{L}_j}{d\beta_j^{(k)}} \qquad \frac{\partial \mathcal{L}_j}{d\mathbf{W}_j^{(k)}}$$

Backwards Propagation – Optimisation Algorithm

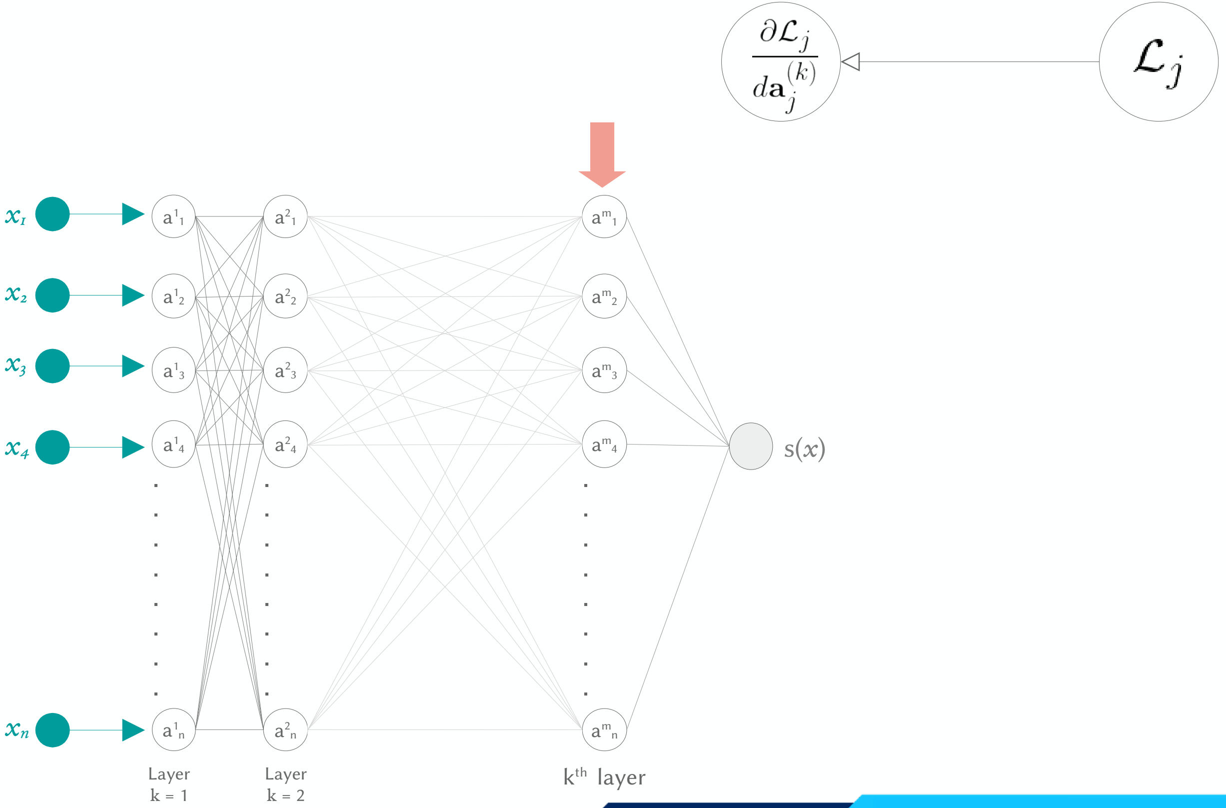
$$\mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$

$\mathcal{L}_j$



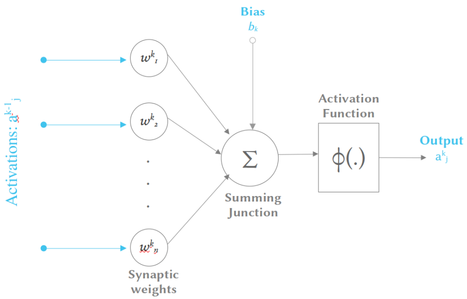
Backwards Propagation – Optimisation Algorithm

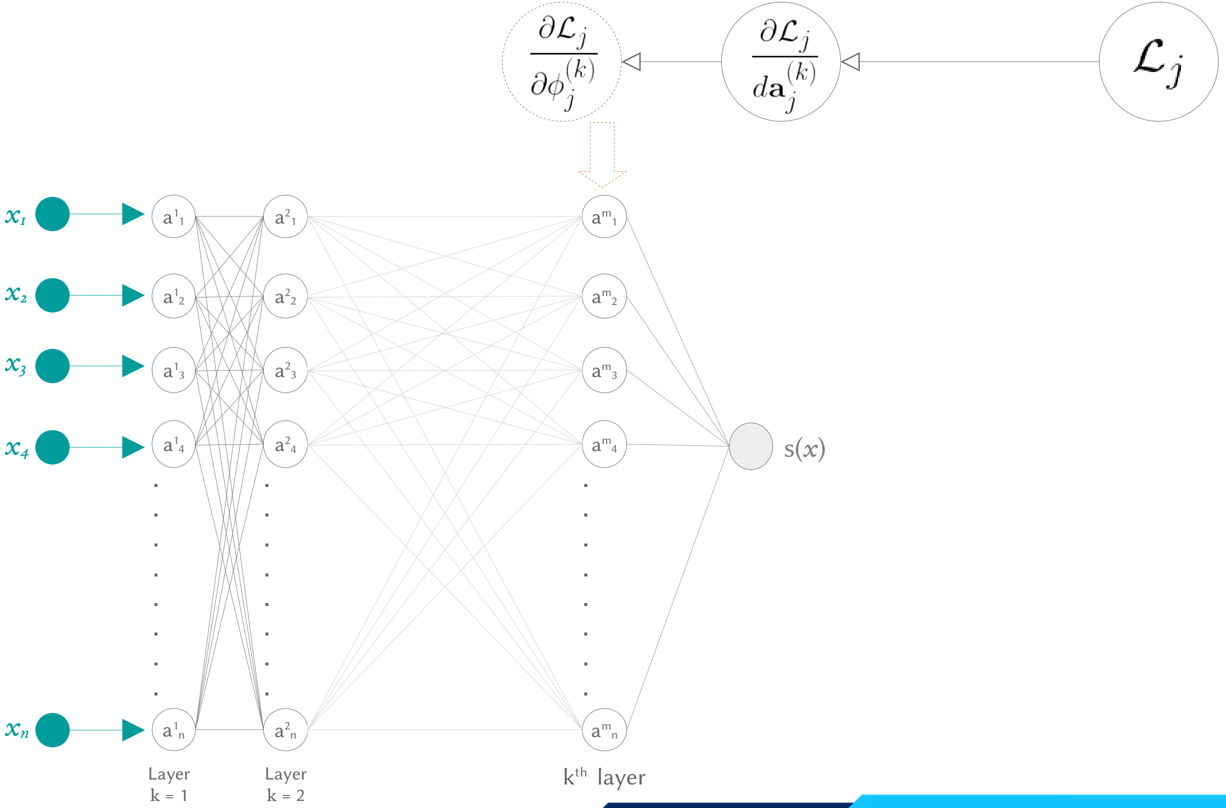
$$\mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$



Backwards Propagation – Optimisation Algorithm

$$\mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$

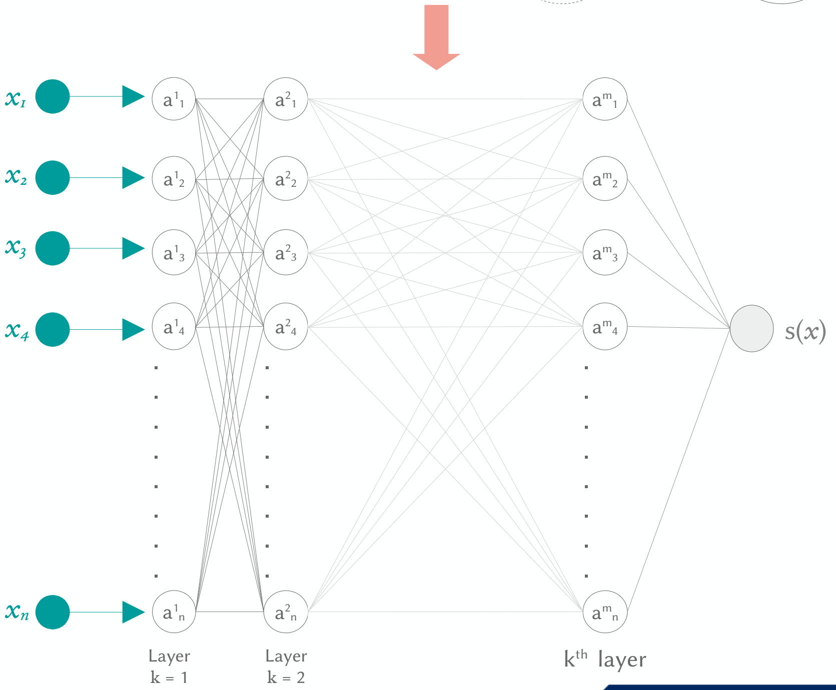




Backwards Propagation – Optimisation Algorithm

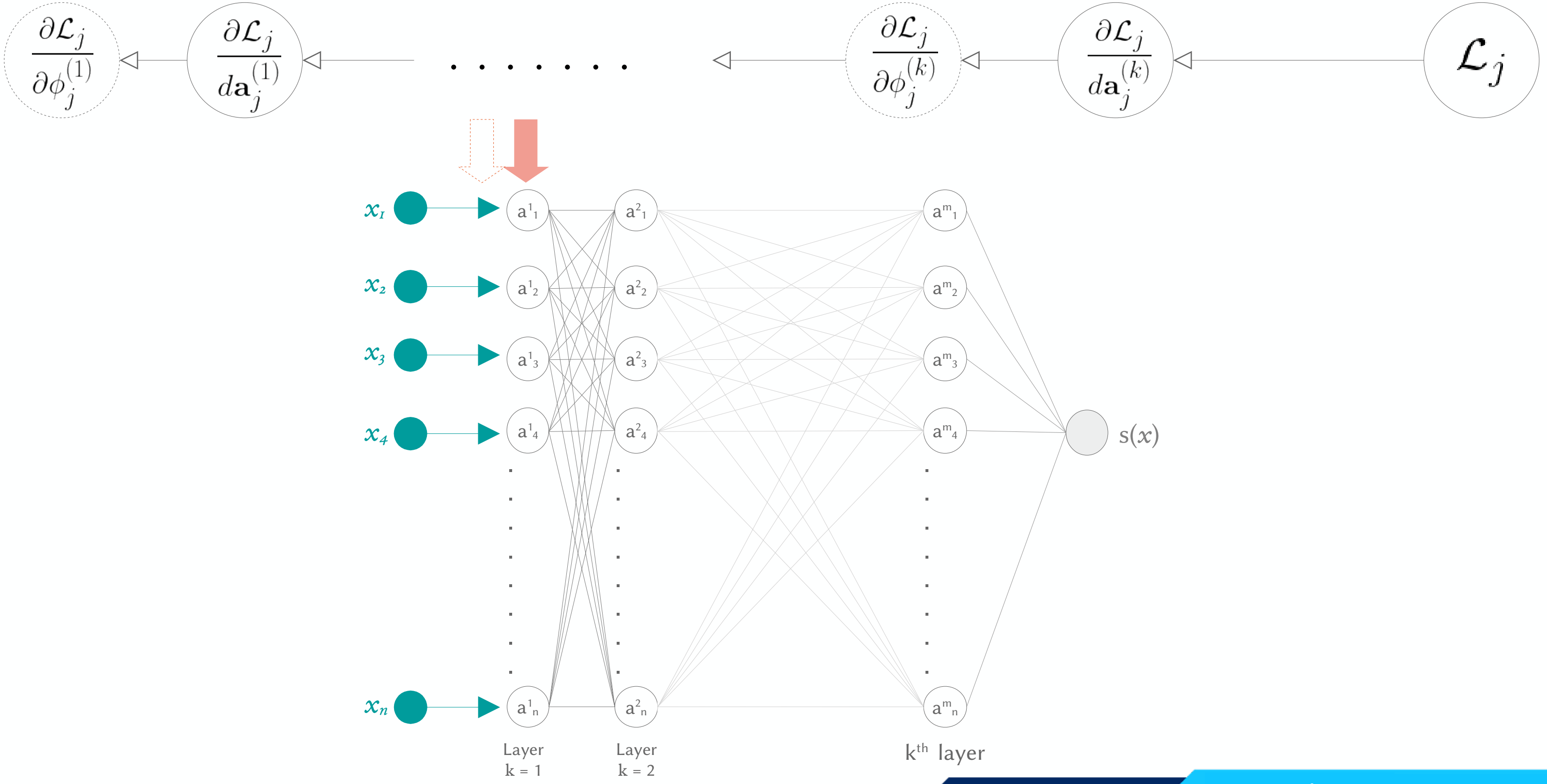
$$\mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$

$$\cdots\cdots \leftarrow \frac{\partial \mathcal{L}_j}{\partial \phi_j^{(k)}} \leftarrow \frac{\partial \mathcal{L}_j}{d\mathbf{a}_j^{(k)}} \leftarrow \mathcal{L}_j$$

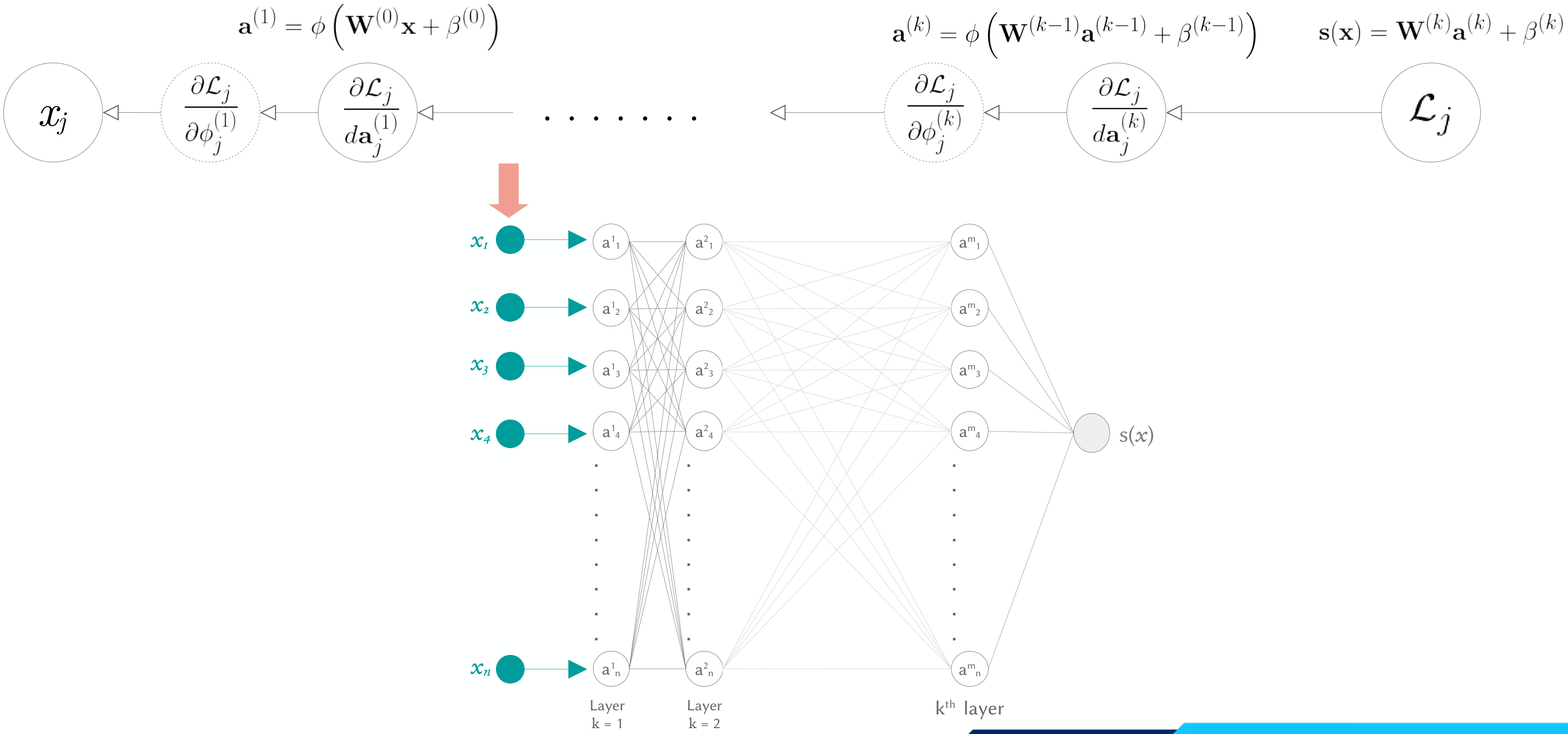


Backwards Propagation – Optimisation Algorithm

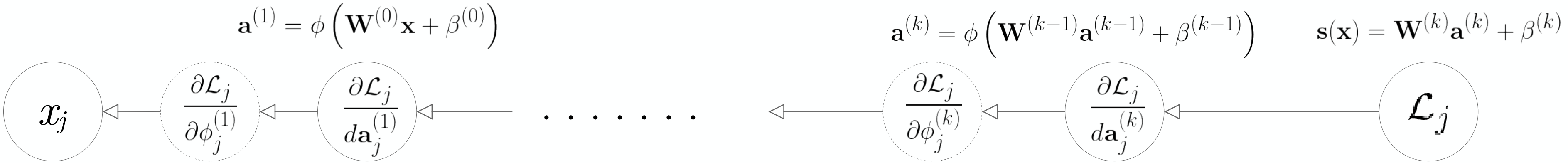
$$\mathbf{a}^{(1)} = \phi\left(\mathbf{W}^{(0)}\mathbf{x} + \beta^{(0)}\right) \qquad \mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$



Backwards Propagation – Optimisation Algorithm



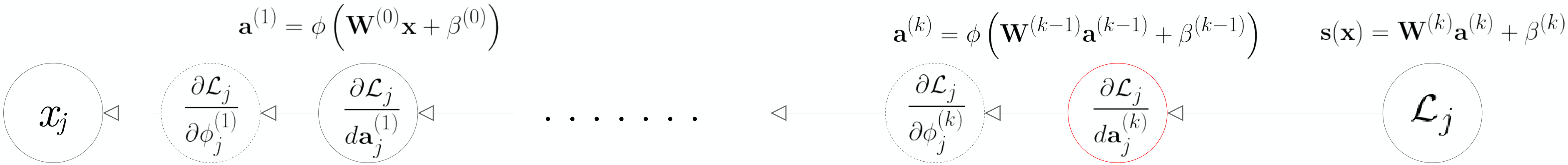
Backwards Propagation – Optimisation Algorithm



- **Chain rule** used to decompose the differential behaviour of the loss $\mathcal{L}$ per-layer (intermediate) variables in reverse order:

$$\frac{\partial y}{\partial x^{(1)}} = \frac{\partial x^{(2)}}{\partial x^{(1)}} \frac{\partial x^{(3)}}{\partial x^{(2)}} \frac{\partial y}{\partial x^{(3)}}$$

Backwards Propagation – Optimisation Algorithm



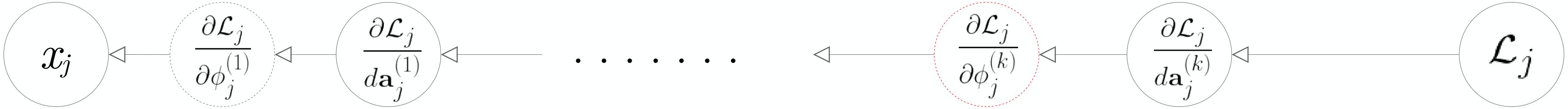
$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

- **Chain rule** used to decompose the differential behaviour of the loss $\mathcal{L}$ per-layer (intermediate) variables in reverse order:

$$\frac{\partial y}{\partial x^{(1)}} = \frac{\partial x^{(2)}}{\partial x^{(1)}} \frac{\partial x^{(3)}}{\partial x^{(2)}} \frac{\partial y}{\partial x^{(3)}}$$

Backwards Propagation – Optimisation Algorithm

$$\mathbf{a}^{(1)} = \phi\left(\mathbf{W}^{(0)}\mathbf{x} + \beta^{(0)}\right) \qquad \mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$



- **Chain rule** used to decompose the differential behaviour of the loss $\mathcal{L}$ per-layer (intermediate) variables in reverse order:

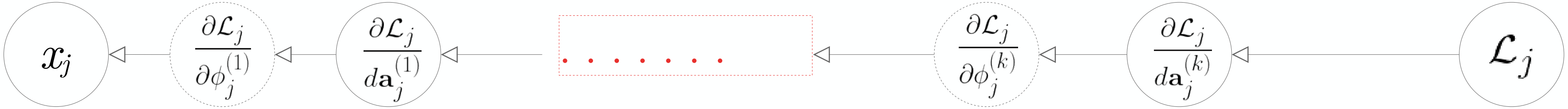
$$\frac{\partial y}{\partial x^{(1)}} = \frac{\partial x^{(2)}}{\partial x^{(1)}} \frac{\partial x^{(3)}}{\partial x^{(2)}} \frac{\partial y}{\partial x^{(3)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \phi_j^{(k)}} = \frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}} \frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

Backwards Propagation – Optimisation Algorithm

$$\mathbf{a}^{(1)} = \phi\left(\mathbf{W}^{(0)}\mathbf{x} + \beta^{(0)}\right) \qquad \mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$



- **Chain rule** used to decompose the differential behaviour of the loss $\mathcal{L}$ per-layer (intermediate) variables in reverse order:

$$\frac{\partial y}{\partial x^{(1)}} = \frac{\partial x^{(2)}}{\partial x^{(1)}}\frac{\partial x^{(3)}}{\partial x^{(2)}}\frac{\partial y}{\partial x^{(3)}}$$

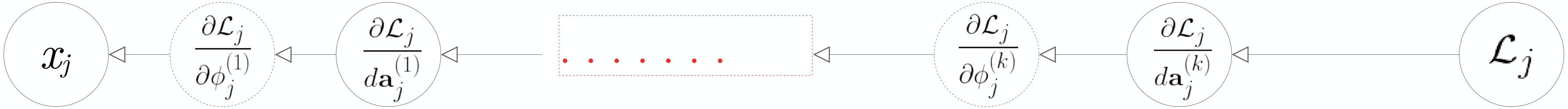
$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \phi_j^{(k)}} = \frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}}\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k-1)}} = \frac{\partial \phi_j^{(k)}}{\partial \mathbf{a}_j^{(k-1)}}\left(\frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}}\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}\right)$$

Backwards Propagation – Optimisation Algorithm

$$\mathbf{a}^{(1)} = \phi\left(\mathbf{W}^{(0)}\mathbf{x} + \beta^{(0)}\right) \qquad \mathbf{a}^{(k)} = \phi\left(\mathbf{W}^{(k-1)}\mathbf{a}^{(k-1)} + \beta^{(k-1)}\right) \qquad \mathbf{s}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{a}^{(k)} + \beta^{(k)}$$



- **Chain rule** used to decompose the differential behaviour of the loss $\mathcal{L}$ per-layer (intermediate) variables in reverse order:

$$\frac{\partial y}{\partial x^{(1)}} = \frac{\partial x^{(2)}}{\partial x^{(1)}}\frac{\partial x^{(3)}}{\partial x^{(2)}}\frac{\partial y}{\partial x^{(3)}}$$

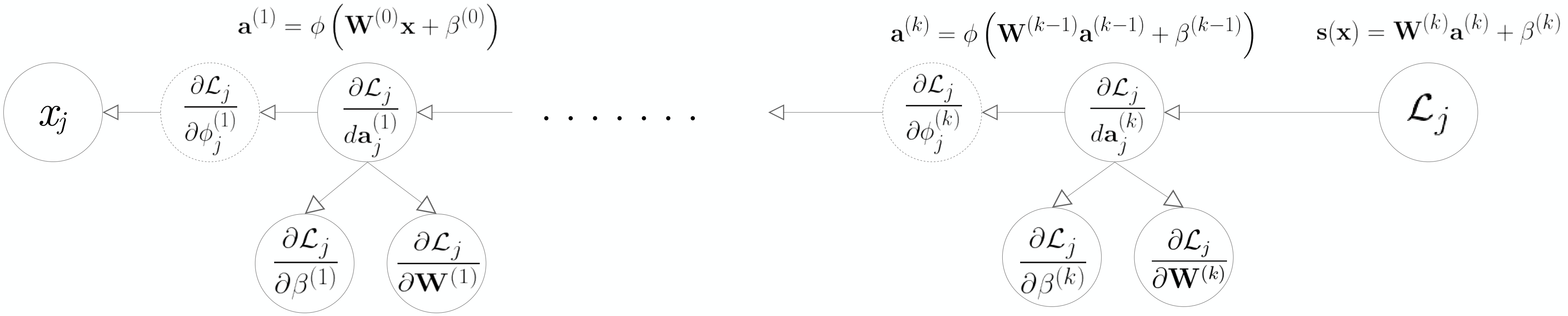
$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \phi_j^{(k)}} = \frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}}\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k-1)}} = \frac{\partial \phi_j^{(k)}}{\partial \mathbf{a}_j^{(k-1)}}\left(\frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}}\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}\right)$$

$$\frac{\partial \mathcal{L}_j}{\partial \phi_j^{(k-1)}} = \frac{\partial \mathbf{a}_j^{(k-1)}}{\partial \phi_j^{(k-1)}}\left(\frac{\partial \phi_j^{(k)}}{\partial \mathbf{a}_j^{(k-1)}}\frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}}\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}\right)$$

Backwards Propagation – Optimisation Algorithm



- **Chain rule** again used to decompose the differential behaviour of the loss $\mathcal{L}$ per-parameter $\Phi = \{W, \beta\}$:

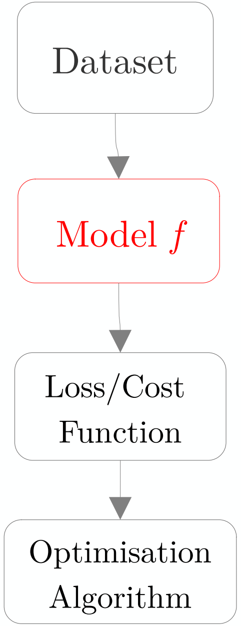
$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{W}_j^{(k)}} = \frac{\partial \mathbf{a}_j^{(k)}}{\partial \mathbf{W}^{(k)}} \frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \beta_j^{(k)}} = \frac{\partial \mathbf{a}_j^{(k)}}{\partial \beta^{(k)}} \frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}}$$

$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k-1)}} = \frac{\partial \phi_j^{(k)}}{\partial \mathbf{a}_j^{(k-1)}} \left(\frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}} \frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}} \right)$$

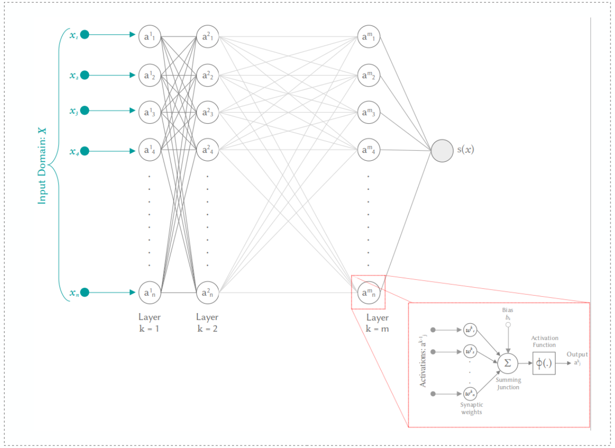
Recap

General recipe for *most* ML algorithms or learning processes:

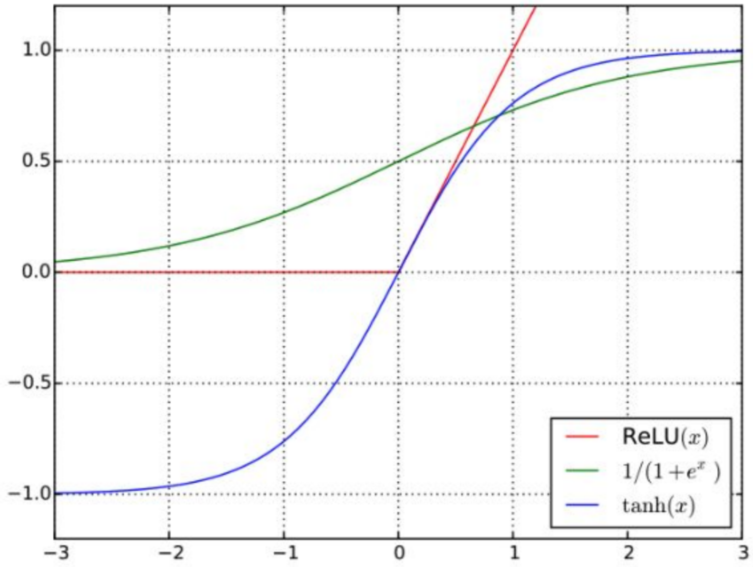


Are we done?

- How do we change the **family of functions** that a model learns? Scale capacity?
- What if we want to predict a discrete value and not a continuous value (classification)?



Activation Functions



- **Choice** of ϕ activation functions is pathological
- Decisions based on an array of factors
- The most common is the concept of **vanishing gradients**:

$$\frac{\partial \mathcal{L}_j}{\partial \mathbf{W}_j^{(k-2)}} = \frac{\partial \mathbf{a}_j^{(k-2)}}{\partial \mathbf{W}_j^{(k-2)}} \left(\underbrace{\frac{\partial \phi_j^{(k-1)}}{\partial \mathbf{a}_j^{(k-2)}}} \frac{\partial \mathbf{a}_j^{(k-1)}}{\partial \phi^{(k-1)}} \underbrace{\frac{\partial \phi_j^{(k)}}{\partial \mathbf{a}_j^{(k-1)}}} \frac{\partial \mathbf{a}_j^{(k)}}{\partial \phi^{(k)}} \frac{\partial \mathcal{L}_j}{\partial \mathbf{a}_j^{(k)}} \right)$$

Sigmoid	ReLU	Sigmoid	ReLU
$\leq 1/4$	≤ 1	$\leq 1/4$	≤ 1

Classification – Discrete Value Predictions

- **Task:** Classify using a dataset drawn from a joint distribution $p(x,y)$:

  Features $\boldsymbol{x} \in \mathbb{R}^n$

  Labels $\boldsymbol{y} \in \mathbb{R}$

- **Goal** is to predict y given an instance of x: $p(y=1|\mathbf{x}) = \dfrac{p(\mathbf{x}|y)p(y=1)}{p(\mathbf{x})}$

Classification – Discrete Value Predictions

- **Task:** Classify using a dataset drawn from a joint distribution $p(x,y)$:

  Features $\boldsymbol{x} \in \mathbb{R}^n$

  Labels $\boldsymbol{y} \in \mathbb{R}$

- **Goal** is to predict y given an instance of x:

$$p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y=1)}{p(\mathbf{x})}$$

$$= \frac{1}{\frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x}|y)p(y=1)} + 1}$$

$$= \frac{p(\mathbf{x}|y)p(y=1)}{p(\mathbf{x}|y=0)p(y=0) + p(\mathbf{x}|y=1)p(y=1)}$$

$$= \frac{1}{1 + \exp\left(\log\left(\frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x}|y=1)p(y=1)}\right)\right)}$$

Classification – Discrete Value Predictions

- **Task:** Classify using a dataset drawn from a joint distribution $p(x,y)$:

  Features $\boldsymbol{x} \in \mathbb{R}^n$

  Labels $\boldsymbol{y} \in \mathbb{R}$

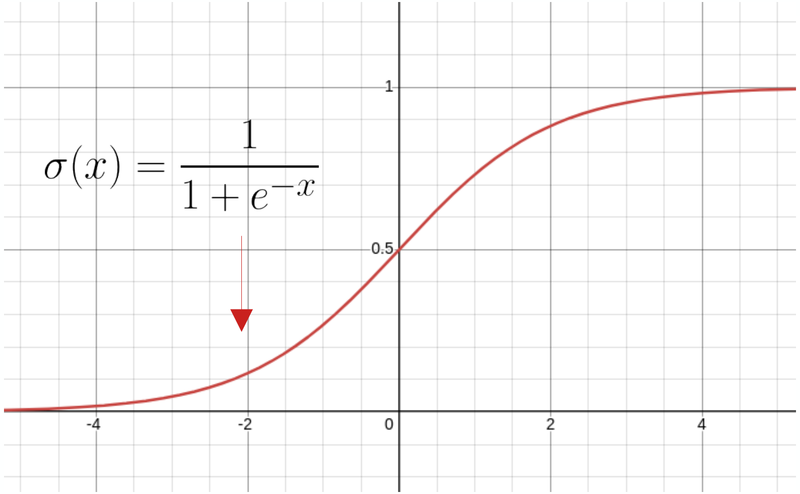
- **Goal** is to predict y given an instance of x:

$$p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y=1)}{p(\mathbf{x})}$$

$$= \frac{1}{\frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x}|y)p(y=1)} + 1}$$

$$= \frac{p(\mathbf{x}|y)p(y=1)}{p(\mathbf{x}|y=0)p(y=0) + p(\mathbf{x}|y=1)p(y=1)}$$

$$= \frac{1}{1 + \exp\left(\log\left(\frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x}|y=1)p(y=1)}\right)\right)}$$



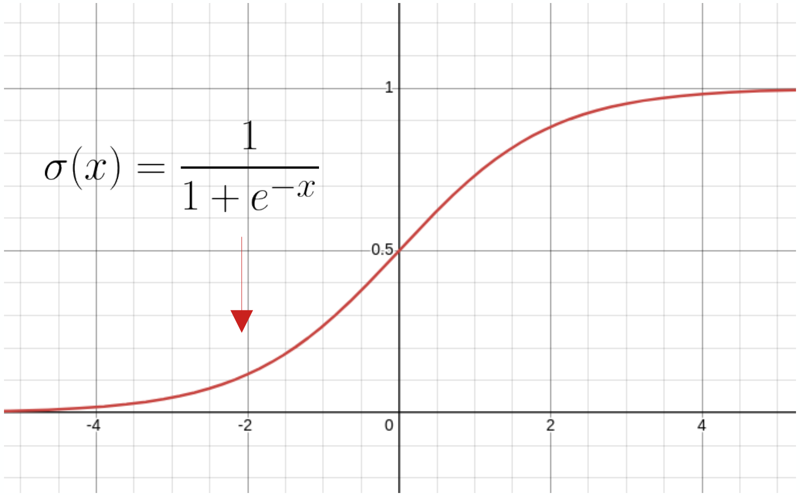
Classification – Discrete Value Predictions

- **Task:** Classify using a dataset drawn from a joint distribution $p(x,y)$:

  Features $\boldsymbol{x} \in \mathbb{R}^n$

  Labels $\boldsymbol{y} \in \mathbb{R}$

- **Goal** is to predict y given an instance of x:



Log-Likelihood Ratio

$$p(y=1|\mathbf{x}) = \sigma\left(\log\left(\frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)}\right) + \log\left(\frac{p(y=1)}{p(y=0)}\right)\right)$$

Class Marginal
(does not depend on x)

Classification – Discrete Value Predictions

- **Task:** Classify using a dataset drawn from a joint distribution $p(x,y)$:

  Features $\boldsymbol{x} \in \mathbb{R}^n$

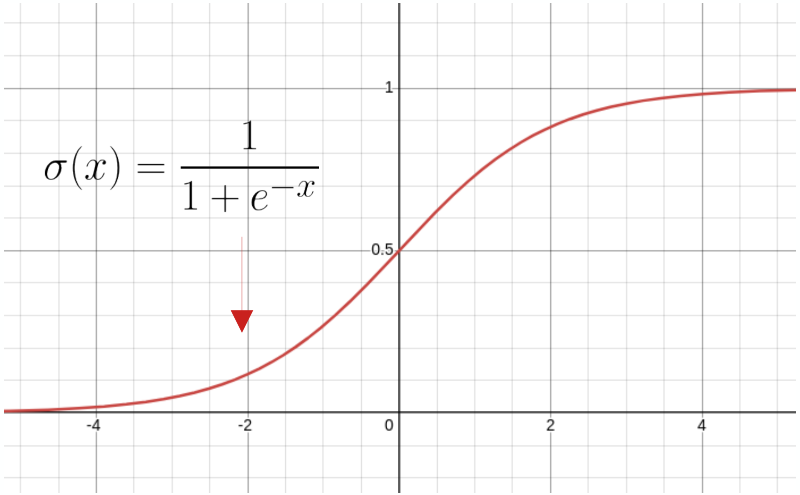
  Labels $\boldsymbol{y} \in \mathbb{R}$

- **Goal** is to predict y given an instance of x:

Log-Likelihood Ratio

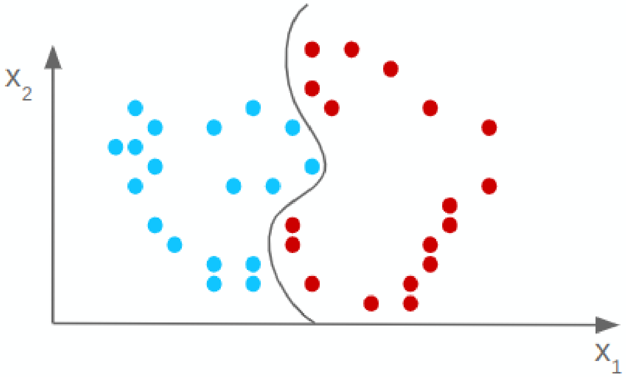
$$p(y=1|\mathbf{x}) = \sigma\left(\log\left(\frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)}\right) + \log\left(\frac{p(y=1)}{p(y=0)}\right)\right)$$

Class Marginal
(does not depend on x)



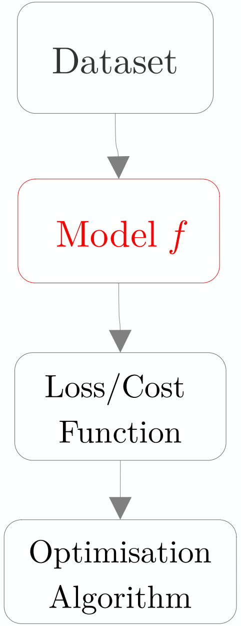
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

The sigmoid converts the distance from a normal regression problem to a probabilistic decision boundary



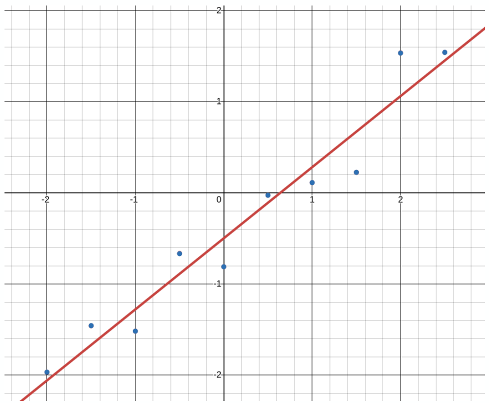
Recap

General recipe for *most* ML algorithms or learning processes:



Are we done?

- How do we change the **family of functions** that a model learns? Scale capacity? ✓
- What if we want to predict a discrete value and not a continuous value (classification)? ✓



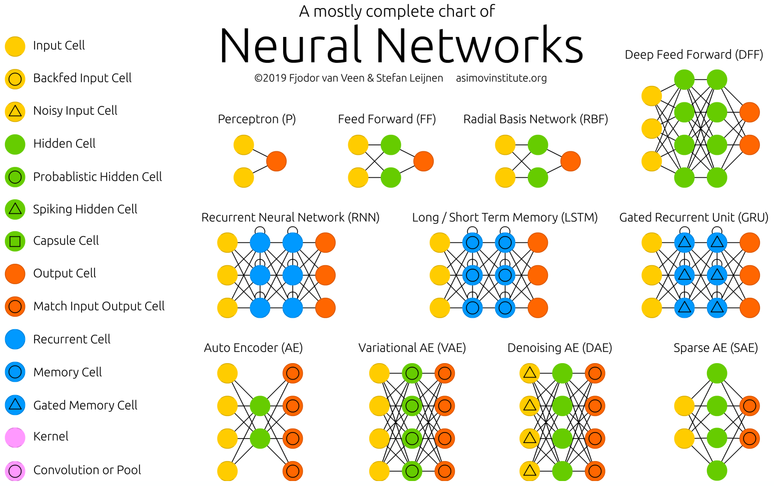
Linear Regression Example

Neural Zoo

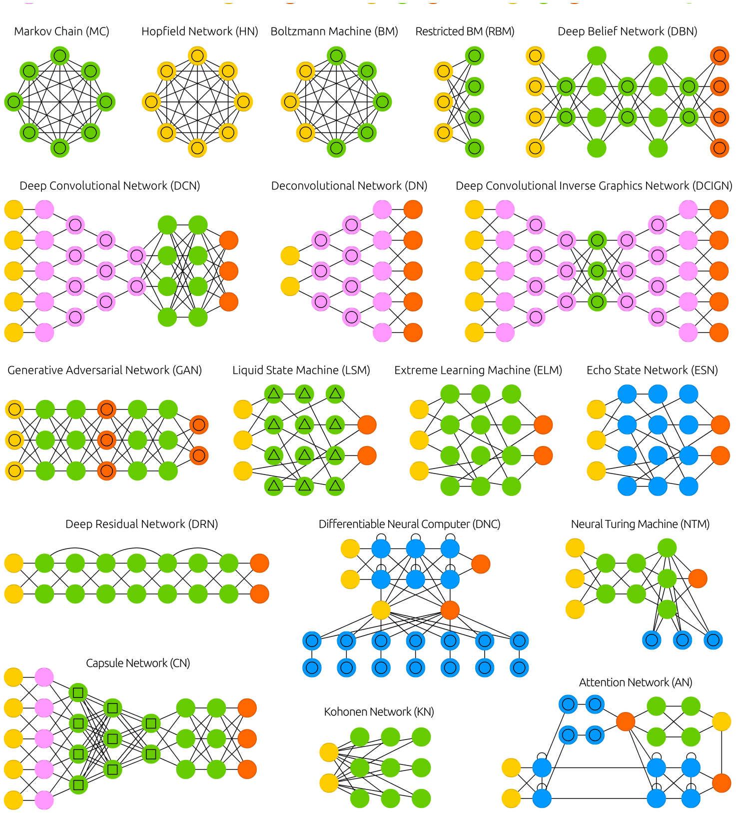
Diversity of Neural Networks

- **Ever growing** number of fundamental building blocks to neural based systems

- **No universally best block** for all uses cases, the various building blocks have different strengths and weaknesses



Source: link



Neural Zoo
Convolutional Neural Networks

Convolutional Neural Network

- **Extract** relevant **features** from a high dimensional input domain space given by the pixel space of an image of height and width **H** × **W** → $x \in \mathbb{R}^{H \times W}$

- **Salient features** ~ e.g. the lines of contrast between the foreground and the background

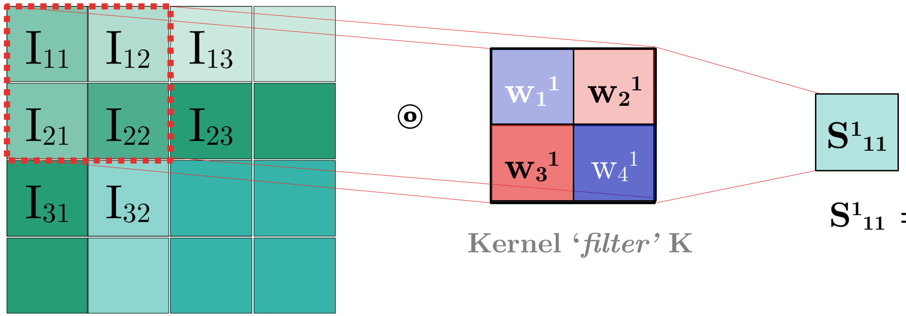
Convolutional Neural Network

- **Extract** relevant **features** from a high dimensional input domain space given by the pixel space of an image of height and width $\mathbf{H} \times \mathbf{W} \rightarrow x \in \mathbb{R}^{\mathrm{H \times W}}$

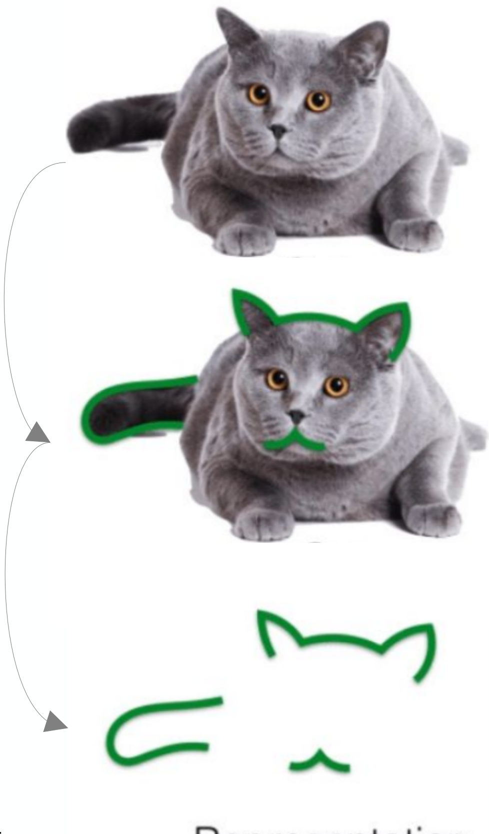
- **Salient features** ~ e.g. the lines of contrast between the foreground and the background

- The key building block is the **kernel filter:**

Image



$$S^1_{11} = w_1^1 I_{11} + w_2^1 I_{12} + w_3^1 I_{21} + w_4^1 I_{22}$$

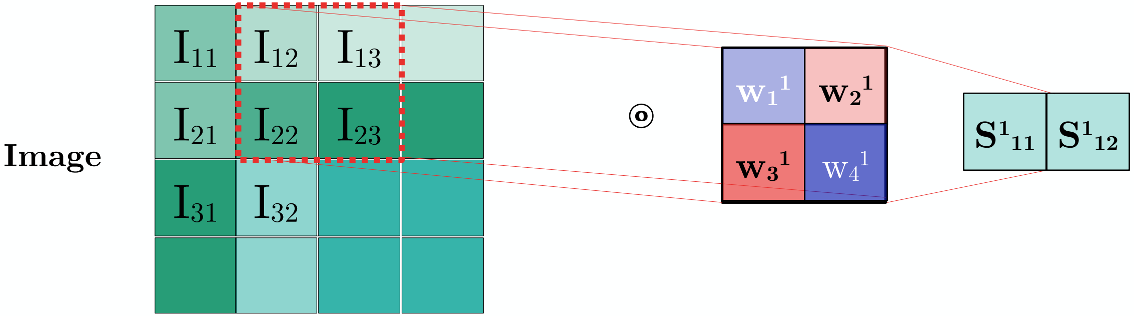


Convolutional Neural Network

- **Extract** relevant **features** from a high dimensional input domain space given by the pixel space of an image of height and width **H** × **W** → $x \in \mathbb{R}^{H\times W}$

- **Salient features** ~ e.g. the lines of contrast between the foreground and the background

- The key building block is the **kernel filter:**

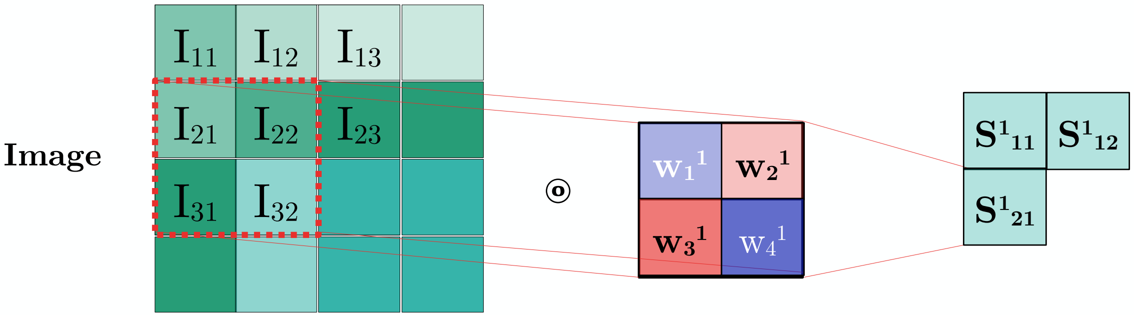


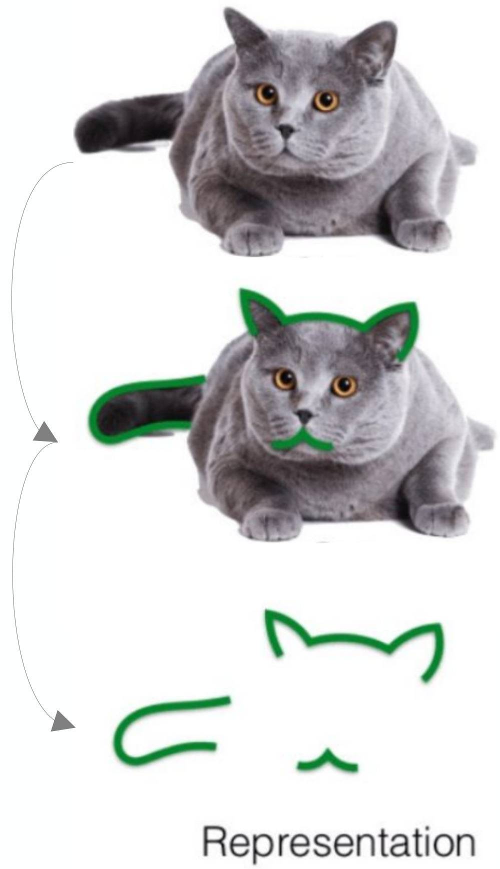
Convolutional Neural Network

- **Extract** relevant **features** from a high dimensional input domain space given by the pixel space of an image of height and width **H** × **W** → $x \in \mathbb{R}^{H \times W}$

- **Salient features** ~ e.g. the lines of contrast between the foreground and the background

- The key building block is the **kernel filter:**





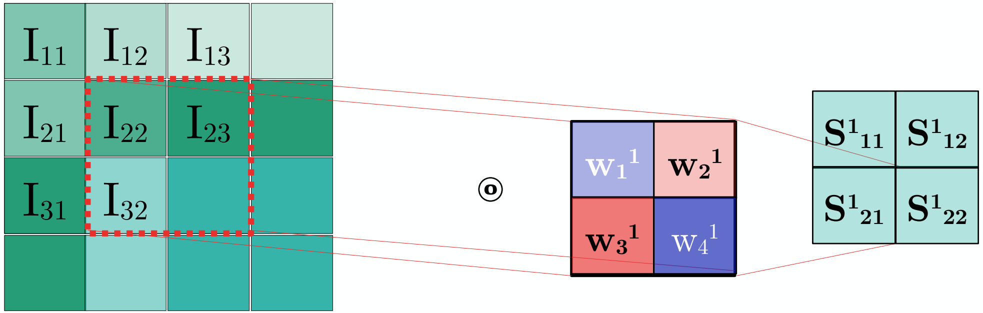
Convolutional Neural Network

- **Extract** relevant **features** from a high dimensional input domain space given by the pixel space of an image of height and width **H** × **W** → $x \in \mathbb{R}^{H \times W}$

- **Salient features** ~ e.g. the lines of contrast between the foreground and the background

- The key building block is the **kernel filter:**

Image



Convolutional Neural Network

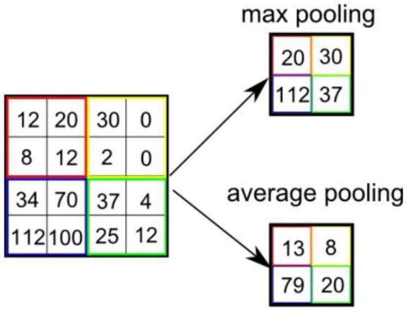
- **Formally** the kernel convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

- **Salient feature map 'S' is by default smaller:**

$$\mathbf{S} \in \mathbb{R}^{(H-K+2P/S+1)\times(W—K+2P/S+1)}$$

- **Compressing** image size further to reduce input dimension by **max/average pooling**



Convolutional Neural Network

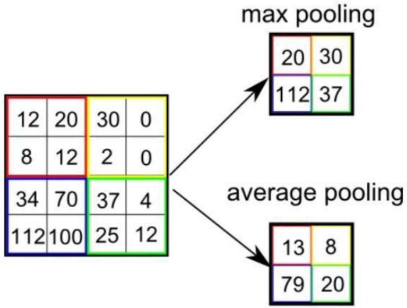
- **Formally** the kernel convolution operation:

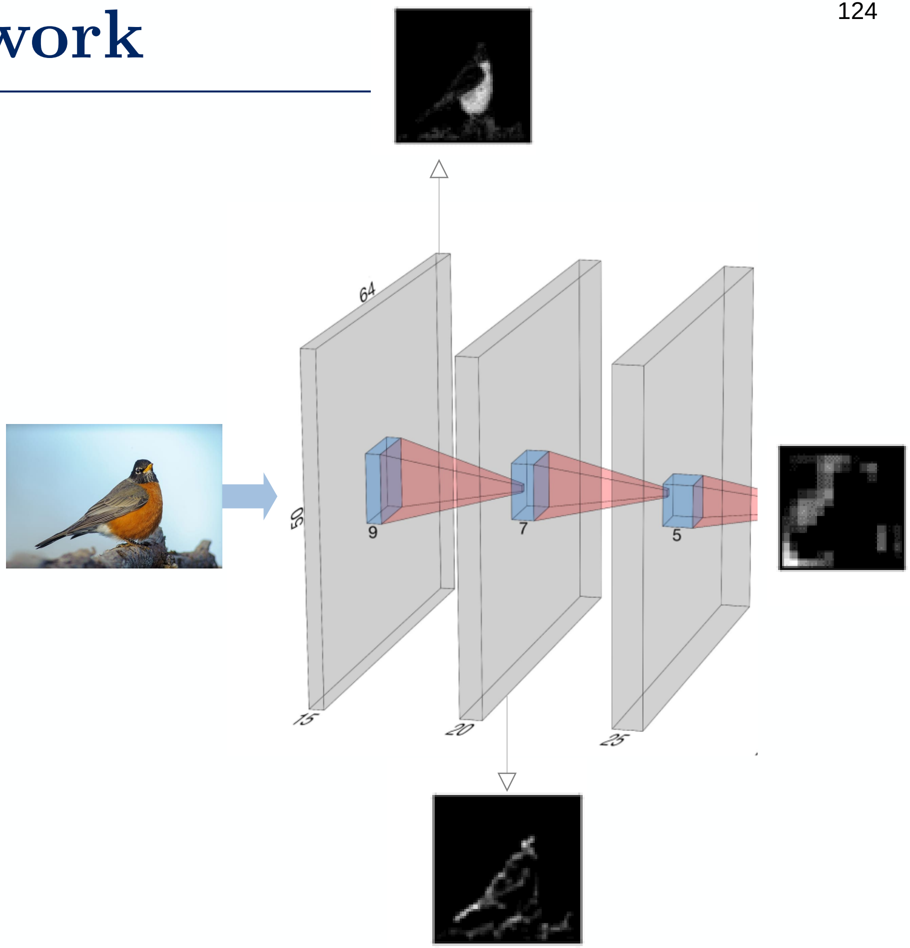
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

- **Salient feature map 'S' is by default smaller:**

$$\mathbf{S} \in \mathbb{R}^{(\text{H-K+2P/S +1})\times(\text{W—K + 2P/S + 1})}$$

- **Compressing** image size further to reduce input dimension by **max/average pooling**





Convolutional Neural Network

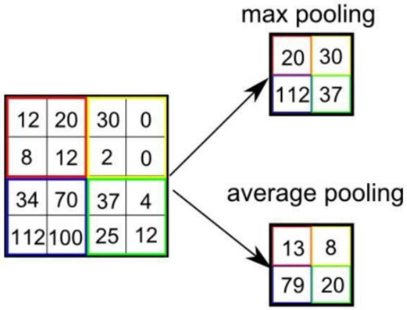
- **Formally** the kernel convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

- **Salient feature map ‘S’ is by default smaller:**

$$\mathbf{S} \in \mathbb{R}^{(H\text{-}K+2P/S+1)\times(W - K + 2P/S + 1)}$$

- **Compressing** image size further to reduce input dimension by **max/average pooling**

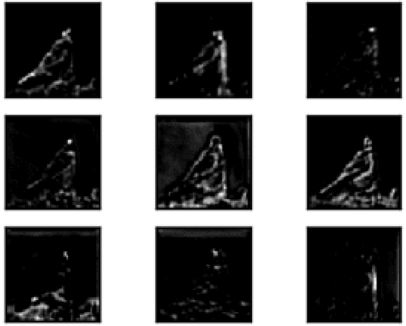


CNN Layer 1

x6 Kernels

CNN Layer 2

x6 Kernels



Convolutional Neural Network

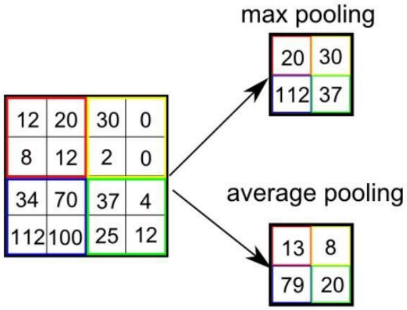
- **Formally** the kernel convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

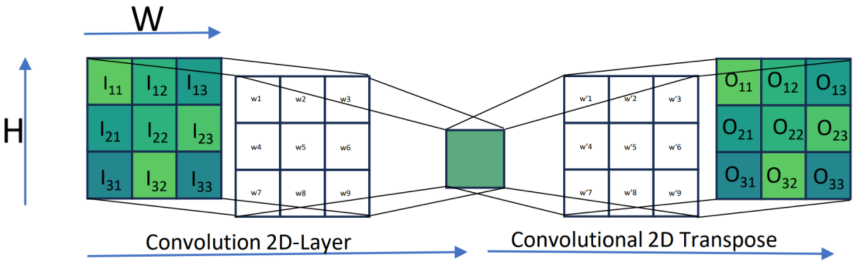
- **Salient feature map 'S' is by default smaller:**

$$\mathbf{S} \in \mathbb{R}^{(H\text{-}K+2P/S\ +1)\times(W—K\ +\ 2P/S\ +\ 1)}$$

- **Compressing** image size further to reduce input dimension by **max/average pooling**



- The **inverse transpose operation** expands the image:



Neural Zoo
Transformers

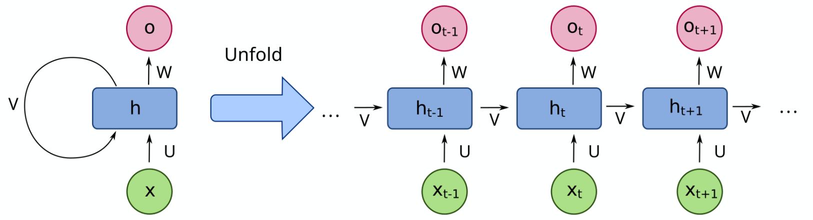
Transformers

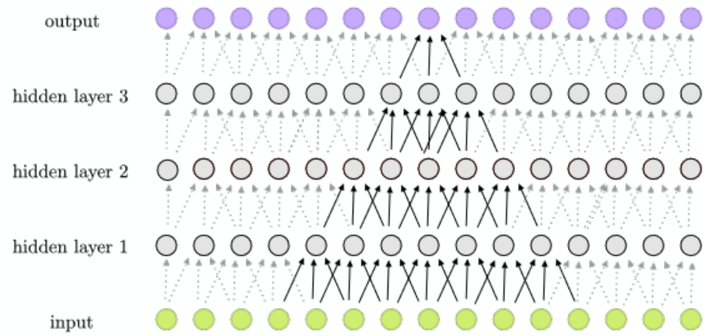
- **Before Transformers:**
  - **Recurrent Neural Networks:**

    Sequential models utilise recurrent connections, in which the output of the network is passed from time step t to t+1 via a recurrent unit – great sequences

  - **Convolution Neural Networks:**

    Model that utilises kernel filters to learn 'salient' features by convoluting nearby activity into increasingly abstract features – great for local information extraction in sequences





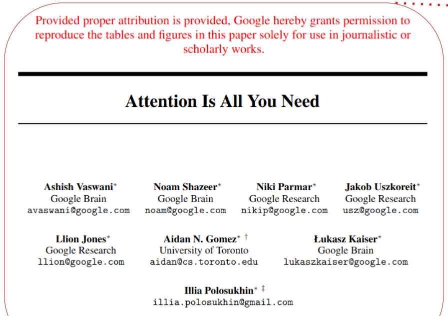
Transformers

- Before Transformers:
  - Recurrent Neural Networks:

    Sequential models utilise recurrent connections, in which the output of the network is passed from time step t to t+1 via a recurrent unit – great sequences
  - Convolution Neural Networks:

    Model that utilises kernel filters to learn 'salient' features by convoluting nearby activity into increasingly abstract features – great for local information extraction in sequences

- **Transformers:**
  - Introduced **attention mechanisms**
  - Provides information about all positions **simultaneously**
  - Great for **sequences,** and geometrical data such as **images**



Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Source: arXiv:1706.03762

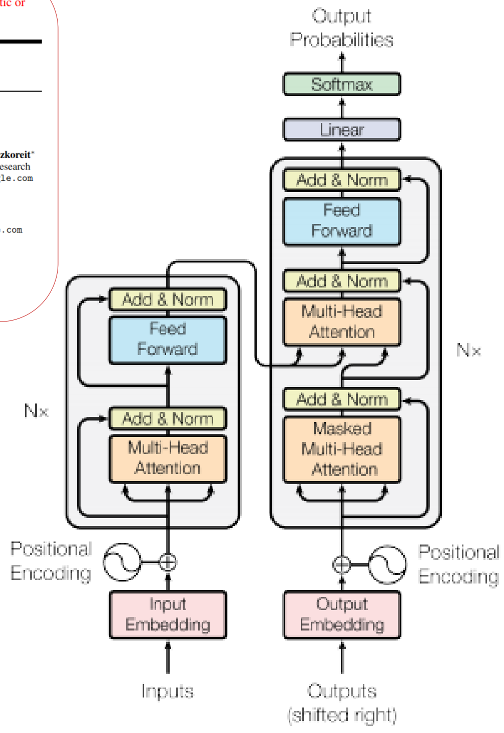


Figure 1: The Transformer - model architecture.

Context Matters – Global Attention

The distribution of pixel values can be generated to reflect a conditional probability:

$p(x_i|\mathcal{C} = \text{'Small cute animal with wide eyes'})$

Context Matters – Global Attention

The distribution of pixel values can be generated to reflect a conditional probability:

$$p(x_i|\mathcal{C} = \text{'Small cute animal with wide eyes'})$$

But what we want is each pixel to be dependent on the context and the rest of the pixel values:

$$p(x_i|\mathcal{C} = \text{'Small cute...'}, \sum_{j \neq i}^{H \times W} x_j)$$

Transformers – Scaled dot-product attention

- **Advantages of Transformers (physics focus):**
  - Handle variable length data
  - Capture **long-range dependencies** in data
  - Permutation/order invariant
  - Highliy parallelisable

- **Attention Mechanism:**
  - The success of transformers resides at *first order* in the introduction of the attention mechanism
  - Understanding attention needs you to understand the idea of:
    - **vector embeddings**
    - **attention**

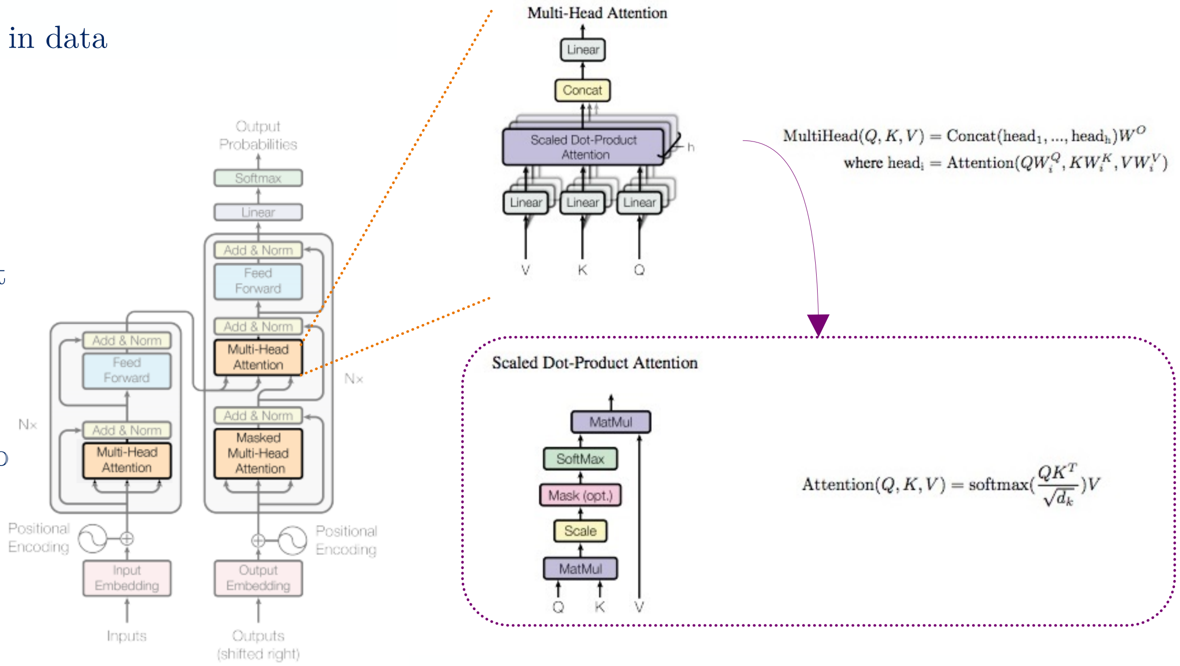


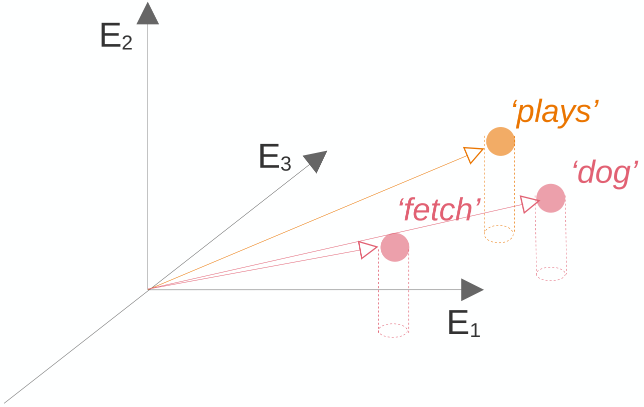
Figure 1: The Transformer - model architecture.

Transformers – Vector Embeddings

- **Transformers act on vectors:**
  - **Tokenisation+embedding** turns each element of the sequence into a vector that resides in a vector space:

$x_i =$ dog | plays | fetch → Embedding →

$$\begin{pmatrix} +.0.36 & +0.63 & -0.23 \\ +.0.21 & +0.47 & +0.18 \\ +.0.25 & +0.92 & -0.47 \end{pmatrix}^T$$



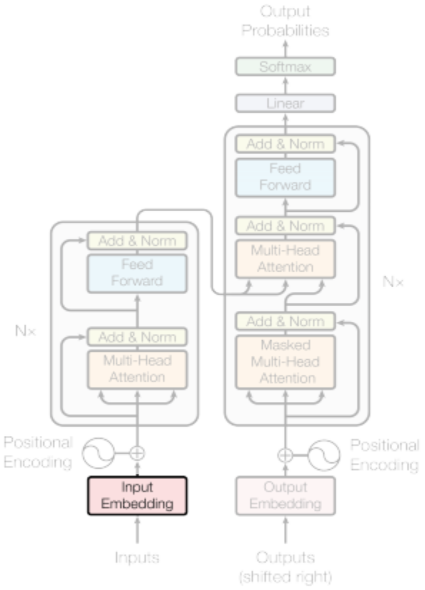
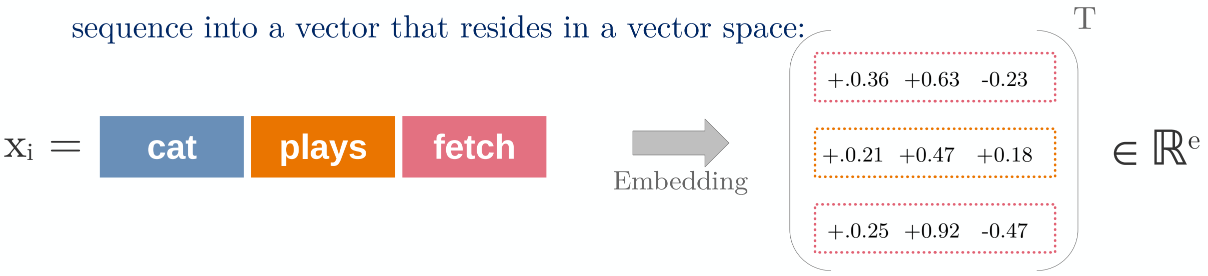
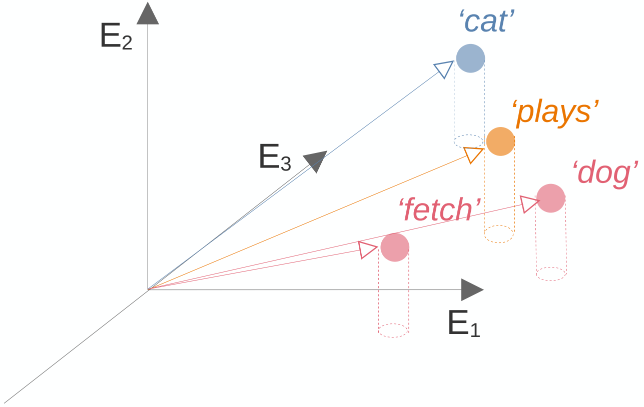


Figure 1: The Transformer - model architecture.

Transformers – Vector Embeddings

- **Transformers act on vectors:**
  - **Tokenisation+embedding** turns each element of the sequence into a vector that resides in a vector space:





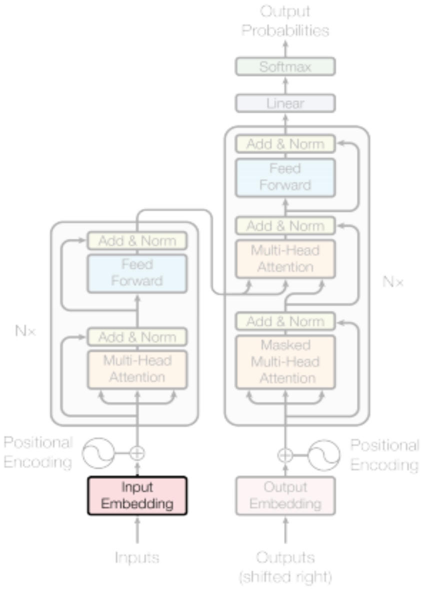
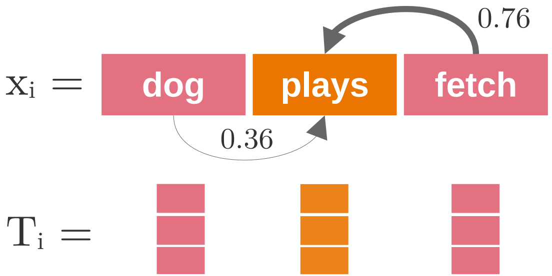
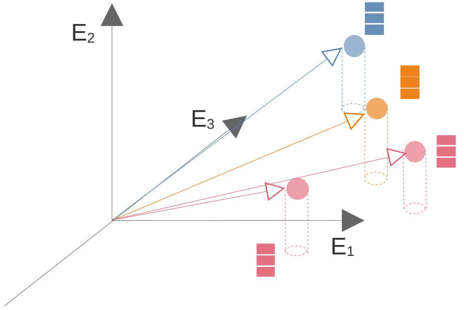


Figure 1: The Transformer - model architecture.

Transformers – Attention

- **Attention = Dynamic flow of information**
  - Each **embedded token** stores which token matters and by how much





  - **Scaled-dot product** attention is one type of attention that achieves this goal:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  - Three key ingredients:
    - **Query (Q)** : What am I looking for?
    - **Key (K)** : What do I have to offer?
    - **Value (V)** : What do I share if picked?

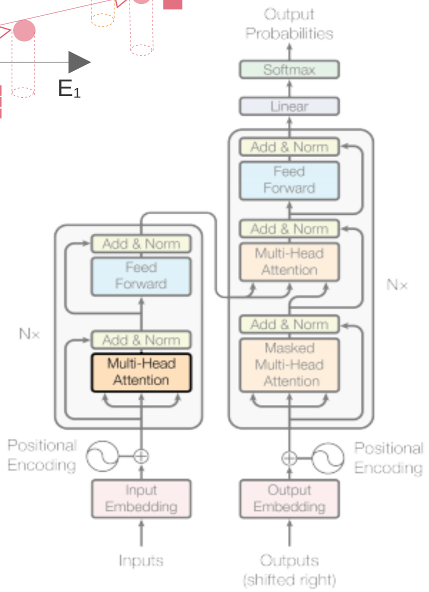
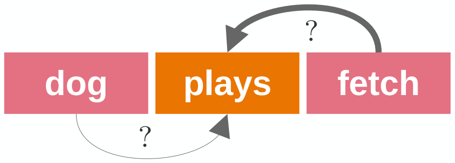
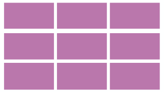
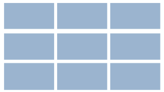


Figure 1: The Transformer - model architecture.

Transformers – Visualising Attention

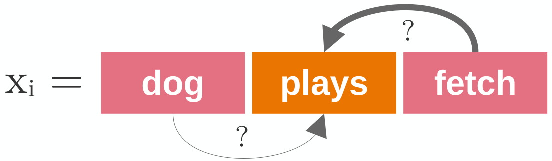
- **Lets see how this works for a simple case:**

$x_i =$ 

- **1) Define scaled-dot product** attention weights for optimisation:
  - $\mathbf{W_Q}$ **:** Matrix of weights for queries, shape = [n, e] 
  - $\mathbf{W_K}$ **:** Matrix of weights for keys, shape = [n, e] 
  - $\mathbf{W_V}$ **:** Matrix of weights for values, shape = [n, e] 

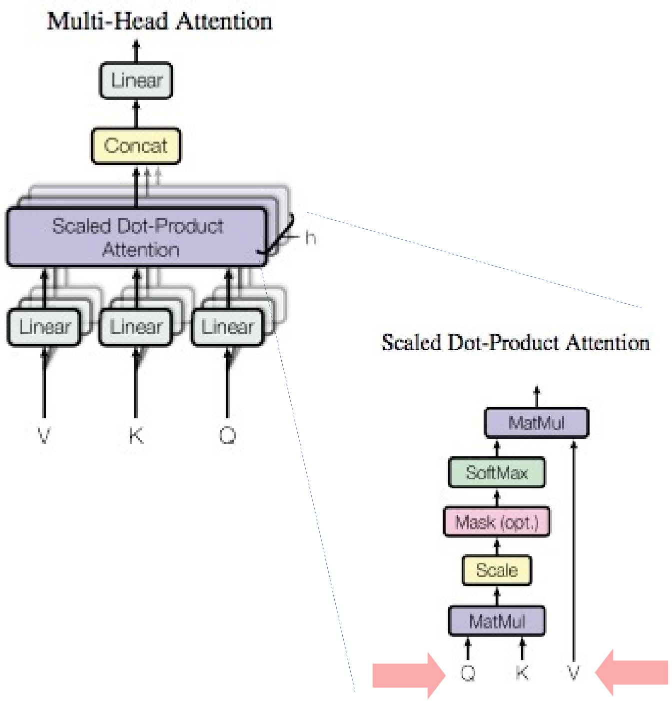
Transformers – Visualising Attention

- **Lets see how this works for a simple case:**



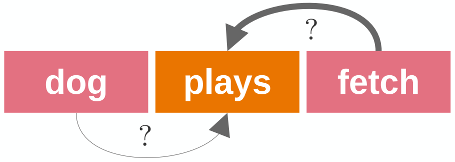
- **2)** Compute the query, key and value triplet **(Q,K,V)** for **'plays'**
  - $\mathbf{Q_{plays}}$: $W_Q \cdot T_p$ = · = [1,e]
  - $\mathbf{K_{d(f)}}$: $W_K \cdot T_{d(f)}$ = · = [1,e]
  - $\mathbf{V_{d(f)}}$: $W_V \cdot T_{d(f)}$ = · = [1,e]

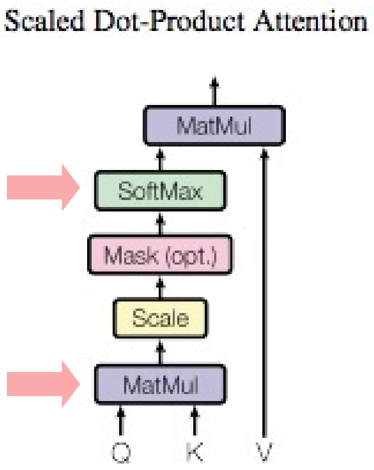
$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

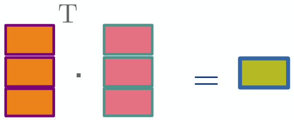
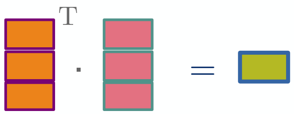


Transformers – Visualising Attention

- **Lets see how this works for a simple case:**

$x_i =$ 

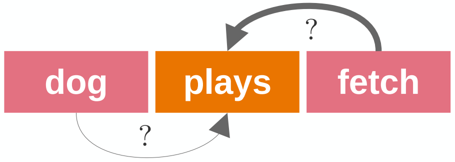


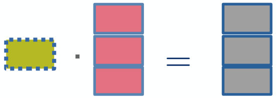
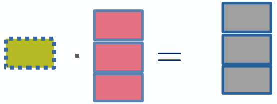
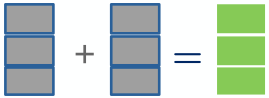
- **2)** Compute the **similarity** between **'dog(fetch)'** to ''**play**'
  - $\mathbf{S}_{\textbf{plays-dog}}$ **:** $Q_{plays} \,.\, K_d =$  $=$ [1,1]
  - $\mathbf{S}_{\textbf{plays-fetch}}$ **:** $Q_{plays} \,.\, K_f =$  $=$ [1,1]

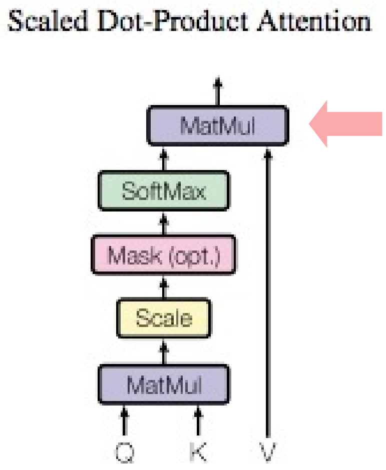
- **3)** Convert to **Similarity** to **'scaled weights'**:
  - $\mathbf{a}^{\mathbf{p}}_{\mathbf{d(f)}}$ **:** $\sigma(\, S_{plays\text{-}dog(fetch)} \,) = \sigma($  $) =$  Softmax forces values to be in range [0,1]

Transformers – Visualising Attention

- **Lets see how this works for a simple case:**

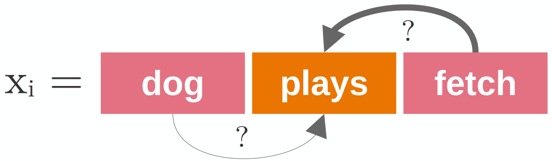
$x_i =$ 

- **4)** Form the context of ‘**plays**’ relative to ‘**dog**’ & ‘**fetch**’:
  - $\mathbf{C_{plays\text{-}dog}}$ **:** $a^{p}_{d} \,.\, V_d \;=$  [1,e]
  - $\mathbf{C_{plays\text{-}fetch}}$ **:** $a^{p}_{f} \,.\, V_f \;=$  [1,e]
- 5) What is the total context of ‘**plays**’?
  - $\mathbf{C_{plays}}$: $\mathbf{C_{plays\text{-}dog}} + \mathbf{C_{plays\text{-}fetch}} =$  [1,e]

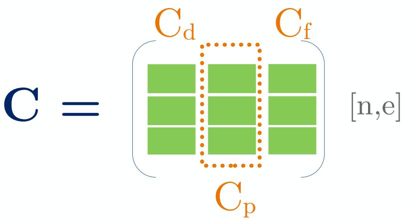


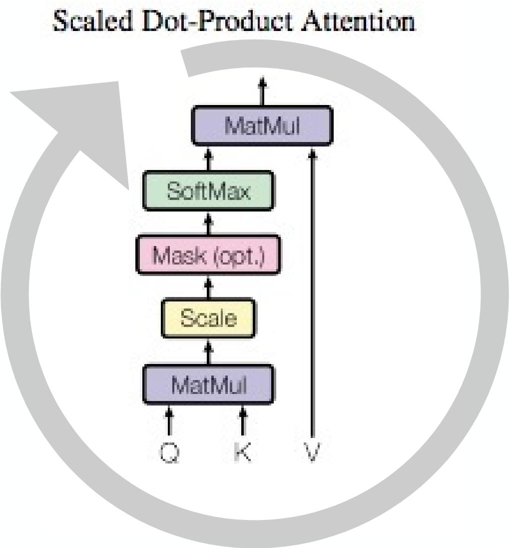
Transformers – Visualising Attention

- **Lets see how this works for a simple case:**



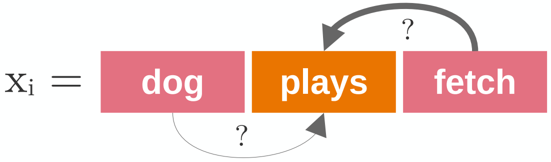
- **6)** Repeat now for **'dog'** and **'fetch'**:



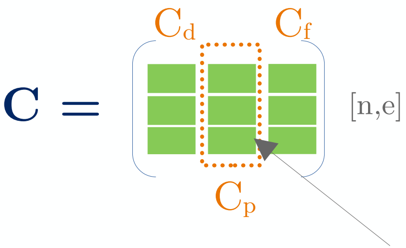


Transformers – Visualising Attention

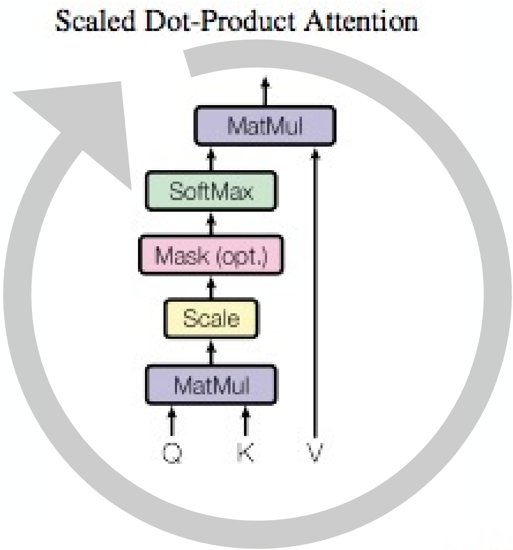
- **Lets see how this works for a simple case:**



- **6)** Repeat now for **'dog'** and **'fetch'**:



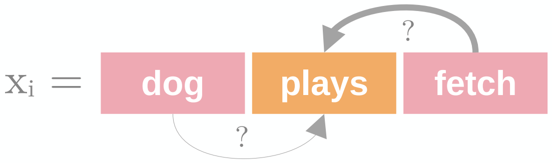
What is the context of all embedding vectors to the word 'play'?



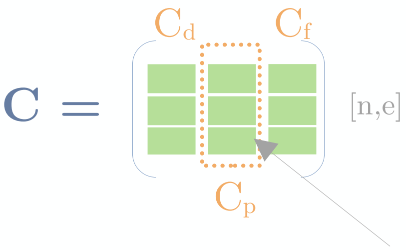
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformers – Visualising Attention

- Lets see how this works for a simple case



- **6)** Repeat now for **'dog'** and **'fetch'**:



What is the context of all embedding vectors to the word 'play'?

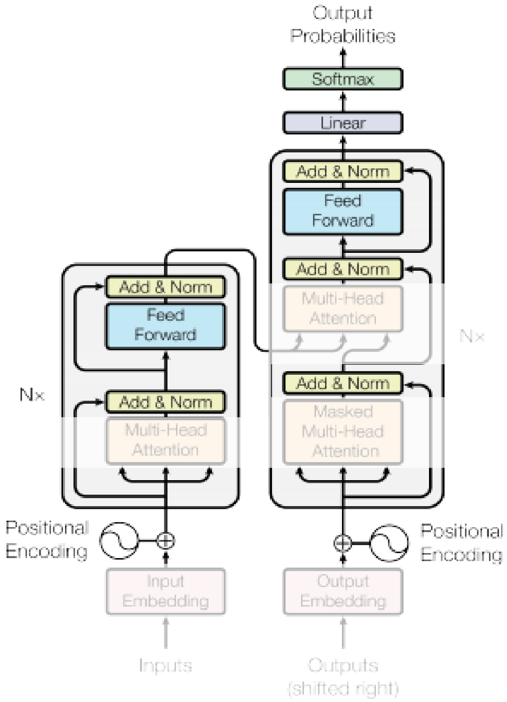


Figure 1: The Transformer - model architecture.

The rest of the transformer is actually quite simple
– just MLPs and adding/normalisation

The End

Thank you!

Resources

- **Books:**

  [1] I.Goodfellow, Y.Bengio, and A.Courville, *'Deep Learning'*, MIT Press, 2016, http://www.deeplearningbook.org

  [2] Simon J.D. Prince, '*Understanding Deep Learning'*, MIT Press, 2023, http://udlbook.com

  [3] Kevin P. Murphy, "Probabilistic Machine Learning: An introduction", MIT Press, 2022, probml.ai

- **Lectures:**

  [1] ATLAS-D 2023, F.Meloni, https://indico.cern.ch/event/1263122/

  [2] G.Louppe, Info8010 Deep Learning, 2024, https://github.com/glouppe/info8010-deep-learning

- **Misc.** :

  [1] Kaare Petersen, Michael Pedersen, '*Matrix Cookbook*', 2012, https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

Backup

Point Estimators: Technicality of continuous variables

- **Covariance:** Linear correlation between two variables:

$$Cov(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])\right]$$

- **Deterministically Correlated Random Variables:**
  → Two random variables X & Y, such that $y = g(x)$:

$$p_x(x) = p_y(g(x)) \left| \frac{\partial g(x)}{\partial x} \right|$$

  → In higher dimensions $\mathbf{x}$ & $\mathbf{y}$, for $\mathbf{x} = g(\mathbf{y})$:

$$p_x(\mathbf{x}) = p_y(g(\mathbf{x})) \left| \det \left(\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

Point Estimators: Technicality of continuous variables

- **Covariance:** Linear correlation between two variables:

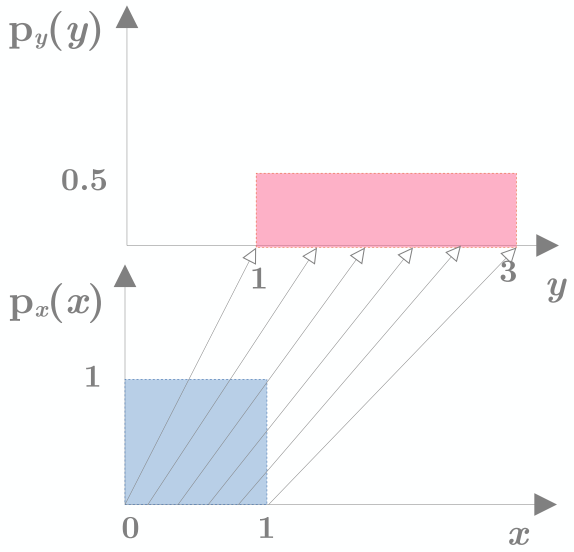
$$Cov(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])\right]$$

- **Deterministically Correlated Random Variables:**
  → Two random variables X & Y, such that $y = g(x)$:

$$p_x(x) = p_y(g(x)) \left| \frac{\partial g(x)}{\partial x} \right|$$

  → In higher dimensions $\mathbf{x}$ & $\mathbf{y}$, for $\mathbf{x} = g(\mathbf{y})$:

$$p_x(\mathbf{x}) = p_y(g(\mathbf{x})) \left| \det \left(\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$



Estimator Properties

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

- **Bias** of an estimator is given by:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

- **Variance** of an estimator:

$$\text{Var}(\hat{\theta})$$

- **Example – Gaussian:**
Sample $\{x\}_m$ generated by a Gaussian *pdf*:

$$\hat{\sigma}^2 = \frac{1}{m}\sum\left[(x_i - \hat{\mu})^2\right] \longleftarrow p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

→ Bias calculation:

$$\text{bias}(\hat{\sigma}^2) = \underbrace{\mathbb{E}(\hat{\sigma}^2)} - \sigma^2 = -\frac{\sigma^2}{m}$$

$$\mathbb{E}(\hat{\sigma}^2) = \mathbb{E}\left[\frac{1}{m}\sum(x_i - \hat{\mu})^2\right] = \frac{m-1}{m}\sigma^2$$

Estimator Properties

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

- **Bias** of an estimator is given by:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

- **Variance** of an estimator:

$$\text{Var}(\hat{\theta})$$

- **Example – Gaussian:**
Sample $\{x\}_m$ generated by a Gaussian *pdf*:

$$\hat{\sigma}^2 = \frac{1}{m-1}\sum\left[(x_i - \hat{\mu})^2\right] \triangleleft\!\!-\!\!-\!\!-\; p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

$\rightarrow$ Bias calculation:

$$\text{bias}(\hat{\sigma}^2) = \mathbb{E}(\hat{\sigma^2}) - \sigma^2 = 0$$

Estimator Properties

- **Point Estimator, or statistic**, is any function of the data that infers from the data some parameter of interest θ:

$$\hat{\theta} = g(\{x\}_m)$$

- **Bias** of an estimator is given by:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

- **Variance** of an estimator:

$$\text{Var}(\hat{\theta})$$

- **Example – Gaussian:**
Sample $\{x\}_m$ generated by a Gaussian *pdf*:

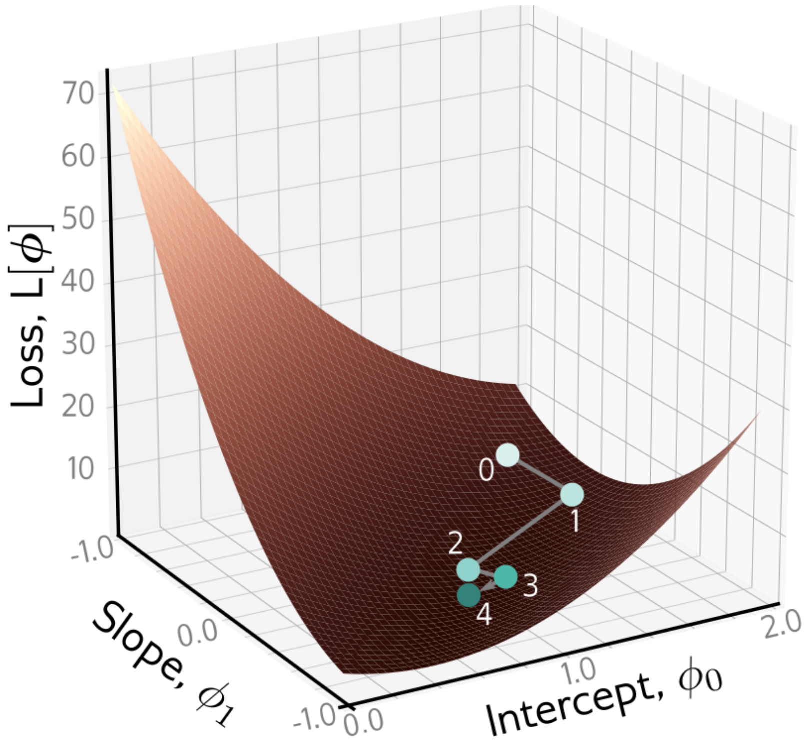
$$\hat{\mu} = \frac{1}{m}\sum_i^m x_i \longleftarrow p(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

→ Variance calculation:

$$\text{Var}(\hat{\mu}) = \text{Var}(\frac{1}{m}\sum_i^m x_i) = \mathbb{E}\left[\left(\frac{1}{m}(\sum_i^m x_i - \mathbb{E}(\sum_i^m x_i))\right)^2\right]$$

$$= \frac{1}{m^2}\sum_i^m \text{Var}(x_i) = \frac{\sigma^2}{m^2}$$

Gradient Descent



- Optimising the model f is achieved by minimising the **loss/cost function**:

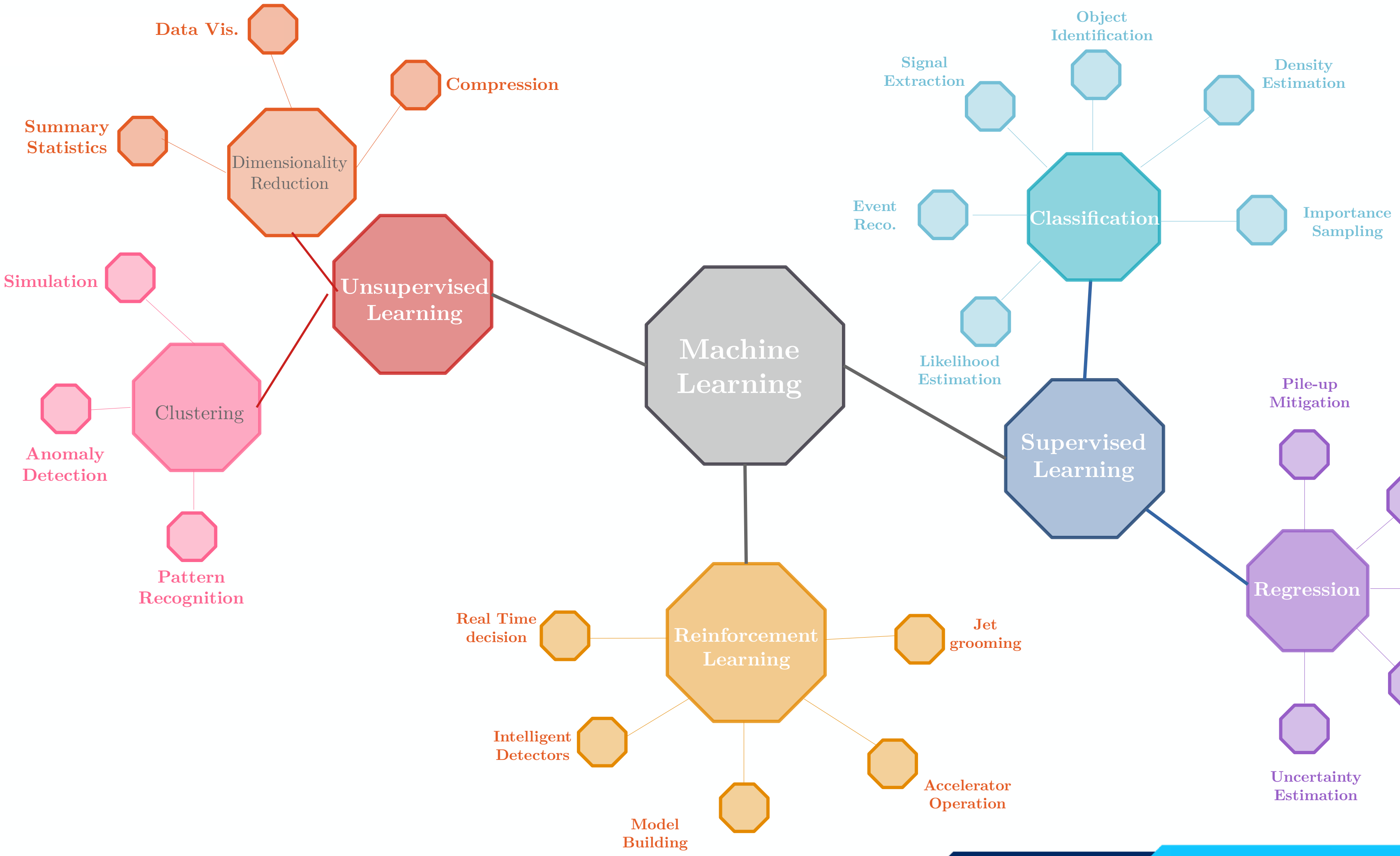
$$\hat{\Phi} = \arg\min \left[\mathcal{L}(f(\mathbf{x}|\Phi), \mathbf{x}) \right]$$

- **Gradient descent** has primarily two steps:
  1) Calculate gradients:

$$\frac{\partial \mathcal{L}}{\partial \Phi} = \begin{bmatrix} \frac{d\mathcal{L}}{\partial \phi_0} \\ \cdots \\ \frac{d\mathcal{L}}{\partial \phi_h} \end{bmatrix}$$

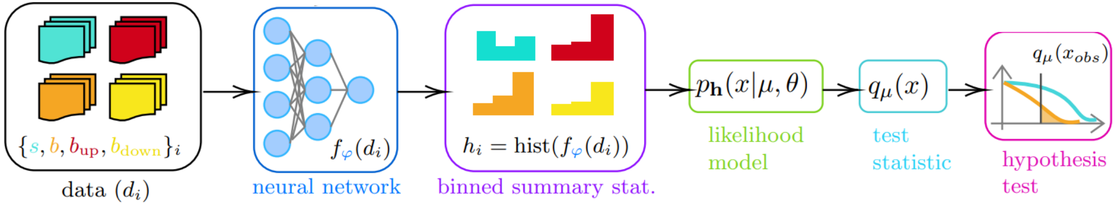
  2) Update the parameters in direction that minimises loss:

$$\phi_i \leftarrow \phi_i - \alpha \frac{\partial \mathcal{L}}{\partial \phi_i}$$


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Data Vis.
Compression
Summary Statistics
Clustering
Simulation
Anomaly Detection
Pattern Recognition
Supervised Learning
Classification
Object Identification
Signal Extraction
Density Estimation
Event Reco.
Importance Sampling
Likelihood Estimation
Regression
Pile-up Mitigation
Calibration
Function Approx.
Parameter Estimation
Uncertainty Estimation
Reinforcement Learning
Real Time decision
Jet grooming
Intelligent Detectors
Model Building
Accelerator Operation

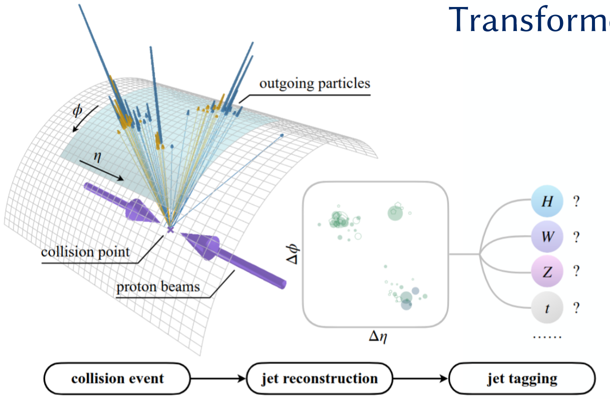
High Energy Physics – ML Developments

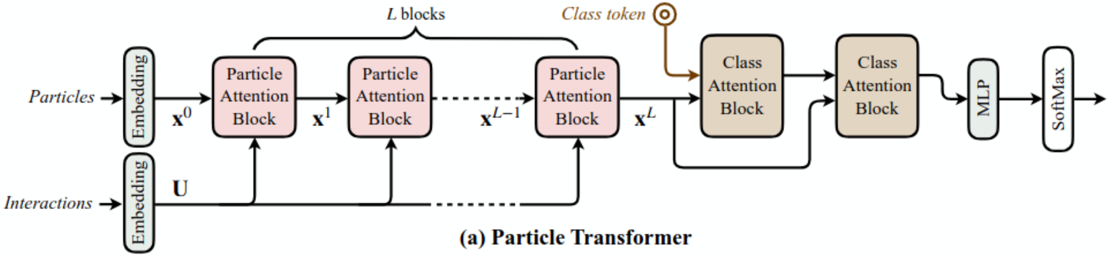
→ Checkout the HEP Machine Learning Living Review



Particle Transformer - arXiv:2202.03772

Transformer model for tagging jets at the LHC





(a) Particle Transformer

Probability Triplet

Can likely skip

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \to [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \rightarrow [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

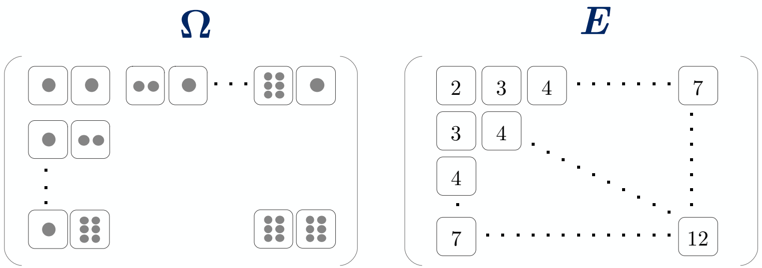
- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Random Variable:** Measurable function $X : \Omega \rightarrow E$ from sample to measurable space:



- **Realisation:** An instance sampled from the random variable distribution:

$$x \sim X$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \to [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

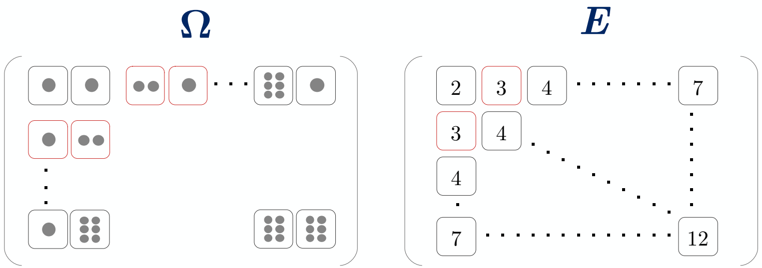
- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Random Variable:** Measurable function $X : \Omega \to E$ from sample to measurable space:



- **Realisation:** An instance sampled from the random variable distribution:

$$x \sim X$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \to [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

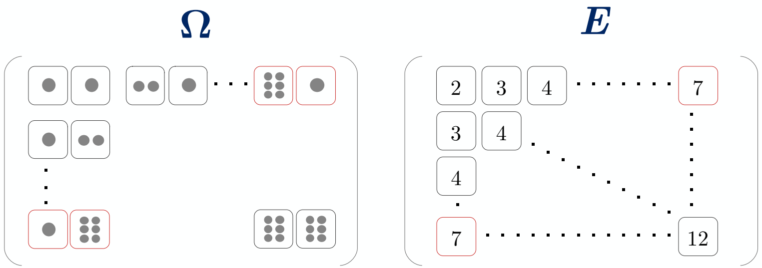
- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Random Variable:** Measurable function $X : \Omega \to E$ from sample to measurable space:



- **Realisation:** An instance sampled from the random variable distribution:

$$x \sim X$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \rightarrow [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

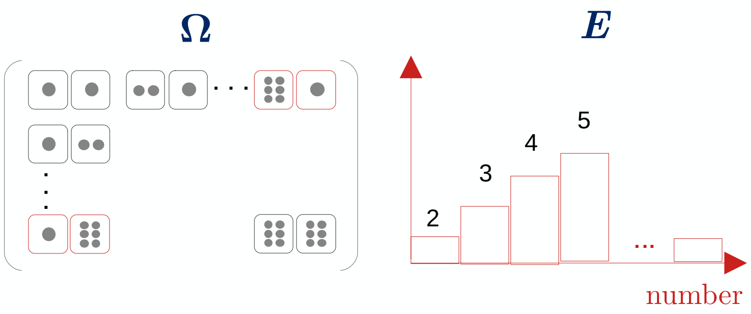
- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Random Variable:** Measurable function $X : \Omega \rightarrow E$ from sample to measurable space:



- **Realisation:** An instance sampled from the random variable distribution:

$$x \sim X$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \rightarrow [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

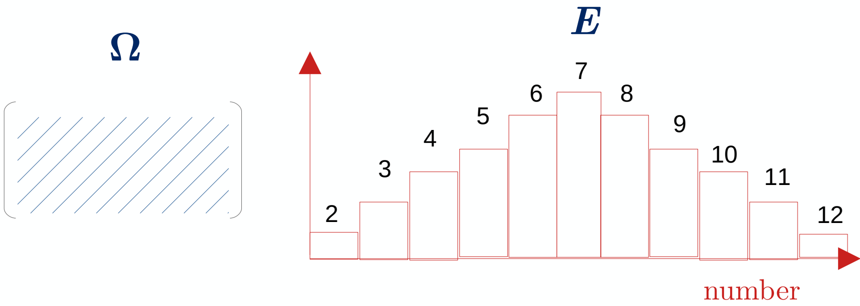
- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Random Variable:** Measurable function $X : \Omega \rightarrow E$ from sample to measurable space:



- **Realisation:** An instance sampled from the random variable distribution:

$$x \sim X$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \rightarrow [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

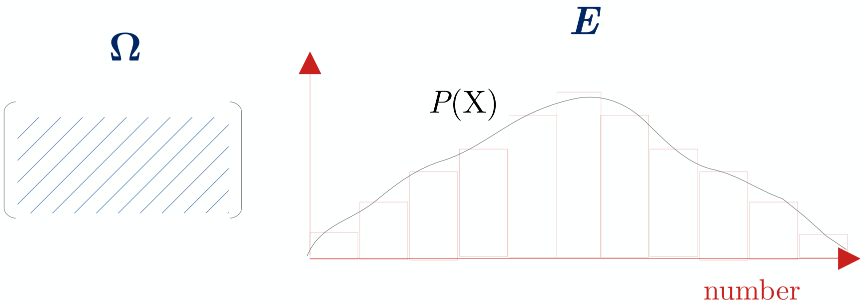
- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Random Variable:** Measurable function $X : \Omega \rightarrow E$ from sample to measurable space:



- **Realisation:** An instance sampled from the random variable distribution:

$$x \sim X$$

[1] *kind of…see uncountable sets*

Continuous Probability

- **Absolute continuous probability** is the infinitesimal sum[1] over the **probability density function** $p : \mathbb{R} \to [0, \infty]$:

$$P(X \in A) = \int_A p(x)dx$$

- **Marginal probability** density:

$$p(x) = \int p(x, y)dy$$

- **Conditional probability** density:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **(Non-)Conditional Independence:**

  Two random variables are independent if:

$$p(\mathbf{x} = x, \mathbf{y} = y) = p(\mathbf{x} = x)p(\mathbf{y} = y)$$

$$\forall x \in \mathbf{x},\ \forall y \in \mathbf{y}$$

[1] *kind of…see uncountable sets*

Point Estimators: Expectation, bias, variance

- **Expectation:** Expected value of a function of random variables

$$\mathbb{E}_{x\sim p}[f(x)] = \int p(x)f(x)dx$$

- **Variance:** Variation of samples from a random variable:

$$Var(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$$

- **Covariance:** Linear correlation between two variables:

$$Cov(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])\right]$$