Uninformed Search

	Time	Space
BFS	exp. (depth)	exp. (depth)
UCS	exp. (tier)	exp. (tier)
DFS	exp. (depth)	poly. (depth)

	Complete	Optimal
BFS	if no. of nodes finite	same step cost
UCS	step cost +ve, total cost finite	step cost +ve
DFS	infinite depth	soln in shallower depth
IDS	always	always

Informed Search

Heuristics

Admissability Never overestimates cost

$$h(n) \leqslant h^*(n)$$

Consistency Fulfills triangle inequality

$$h(n) \leqslant c(n, a, n') + h(n'), h(G) = 0$$

Dominance A more dominant admissible heuristic function is better for search.

$$h_1(n) \geqslant h_2(n) \implies h_1 \text{ dominant}$$

A*Star search

Admissible $h(n)$: Optimal **w/o visited memory**
Consistent $h(n)$: Optimal **w. visited memory**

Adversarial Search

Minimax

Alpha-beta pruning: Perfect ordering $O(b^{\frac{m}{2}})$

Decision Tree

Entropy

Entropy $B(q)$ of a boolean random var. with prob. q:

$$B(q) = -(qlog_2 q + (1-q)log_2(1-q))$$

Entropy of entire output:

$$H(output) = B(\frac{p}{p+n})$$

To calculate Information Gain:

$$remainder(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(\frac{p_k}{p_k+n_k})$$

$$IG(A) = H(output) - remainder(A)$$

Entropy Table:

b/a	1	2	3	4	5	6	7
2	1				I(a/b, 1 - a/b)		
3	0.9183						
4	0.8113	1					
5	0.7219	0.971					
6	0.65	0.9183	1				
7	0.5917	0.8631	0.9852				
8	0.5436	0.8113	0.9544	1			
9	0.5033	0.7642	0.9183	0.9911			
10	0.469	0.7219	0.8813	0.971	1		
11	0.4395	0.684	0.8454	0.9457	0.994		
12	0.4138	0.65	0.8113	0.9183	0.9799	1	
13	0.3912	0.6194	0.7793	0.8905	0.9612	0.9957	
14	0.3712	0.5917	0.7496	0.8631	0.9403	0.9852	1
15	0.3534	0.5665	0.7219	0.8366	0.9183	0.971	0.9968

Linear Regression

Loss function

$$J_{MSE}(w) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^i) - y^i)^2$$

In matrix form:

$$\frac{1}{2m}(Xw - Y) \cdot (Xw - Y)$$

Minimising loss function

With regard to the weight w_j, the partial derivative is:

$$\frac{\delta}{\delta w_j} J_{MSE}(w) = \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Normal Equation

$$w = (X^T X)^{-1} X^T Y$$

Gradient descent

With learning rate α (hyperparameter),

$$w_j \leftarrow w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Logistic Regression

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

where the derivative is:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Cross-entropy

$$CE(y, \hat{y}) = \sum_{i=1}^{C} -ylog(\hat{y}_i)$$

$$BCE(y, \hat{y}) = -ylog(\hat{y}) - (1-y)log(1-\hat{y})$$

with the partial derivative

$$\frac{\delta}{\delta w_i} J_{BCE}(w) = \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\frac{\delta}{\delta w} J_{BCE}(w) = \frac{1}{m} X^T (\sigma(XW) - y)$$

Regularisation

The **L2 penalty**, when added to the loss function of a Linear Regression model:

$$J_{reg}(w) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2 \right]$$

where λ is the hyperparameter.

The **L1 penalty**:

$$\sum_{j=0}^{d} |w_j|$$

Update rule

$$w_j \leftarrow \left(1 - \frac{\alpha\lambda}{m}\right) w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Normal equation

In matrix form:

$$w = (X^T X + \lambda \mathbb{I})^{-1} X^T y$$

Always invertible as $X^T X$ is **symmetric and positive semi-definite**, $\lambda\mathbb{I}$ is **symmetric and positive definite**, and adding a positive definite matrix to a semi-definite matrix results in a positive definite matirx, which is **always invertible**.

SVM

Maximum margin

$$w \cdot x + b \geqslant 0 \text{ then } +$$

$$margin = \frac{2}{||w||}$$

Note that the norm $||w||$ is calculated as $\sqrt{\sum_i x_i^2}$.

To find the margin, the optimisation problem is as such:

$$\max_w \frac{2}{||w||} s.t. y^{(i)}(w \cdot x^{(i)} + b) - 1 \geqslant 0$$

Distance between point x and decision boundary:

$$\frac{|w^T x + b|}{||w||}$$

or find a point z, satisfying $w^T x + b = 0$, then

$$\frac{w^T(x-z)}{||w||}$$

Unsupervised Learning

Clustering

Centroid:

$$\mu = \frac{1}{N_j} \sum_{i=1}^{N_j} x^{(i)}$$

The distortion is the average distance of the sample to its centroid:

$$J(c^{(1)}, ..., c^{(N)}, \mu_i, ...\mu_K) = \frac{1}{N} \sum_{i=1}^{N} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

K-means

Can get stuck in local optima, but each step never increases distortion

Picking number of clusters
Elbow method, business needs (number of groupings)

Variants **Points as centroids**
K-Medoids Data points closest to centroids used

Dimensionality reduction

Given the transposed data matrix X^T:

$$X^T = U \sum V^T$$

where U is the new orthonormal basis, the $\sum$ is the importance, and the V^T is the linear combination coefficients.

With $\sum$, the singular values can be seen in the diagonals, and are ordered in size. To reduce the dimensions, we can set all σ_j where $j > n$, and n is the number of wanted dimensions to 0.

We can then get the new basis $\hat{U}$ which is U with all the column vectors after n removed, returning a $d \times r$ vector.

The new vectors can then be retrieved:

$$Z := \hat{U}^T X^T$$

where Z is a $r \times n$ matrix.

Reconstruction

$$X^T := \tilde{U}\tilde{U}^T X^T = \tilde{U} Z \approx X^T$$

Neural Networks

Neuron

$$h_w(x) = g(w_x^T) = g(\sum_{j=0}^{d} w_j x_j)$$

with $g(z)$ being an activation function.

Activation functions

Sign function

$$g(z) = \begin{cases} +1 & z \geqslant 0 \\ -1 & z < 0 \end{cases}$$

with derivative 0.

Sigmoid function

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

with derivative

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

ReLU function

$$max(0, x)$$

with derivative

$$\begin{cases} 0 & x \leqslant 0 \\ 1 & x > 0 \end{cases}$$

Leaky ReLU function

$$max(cx, x), 0 < c < 1$$

with derivative

$$\begin{cases} c & x \leqslant 0 \\ 1 & x > 0 \end{cases}$$

Update rule

$$w_j \leftarrow w_j + (y^{(k)} - \hat{y}^{(k)}) x_j^{(k)}$$

Forward propagation

Used for predictions.

$$\hat{y} = g^{[j]}(w^{[j]^T}(.....g^{[1]}(w^{[1]^T}x)))$$

Backward propagation

Used to compute loss function.

Using the chain rule,

$$l = h(g(f(x)))$$

$$\frac{dl}{dx} = \frac{dl}{dy}\frac{dy}{dz}\frac{dz}{dx}$$

Similarly, for derivative of multiple input:

$$l = h(f(x), g(x))$$

$$\frac{dl}{dx} = \frac{\delta l}{\delta z_1}\frac{\delta z_1}{\delta x} + \frac{\delta l}{\delta z_2}\frac{\delta z_2}{\delta x}$$

CNN

For an input feature map of size $x \times x$, with stride s, padding p, and kernel of size $w \times w$.

The output feature map size will be:

$$\lfloor \frac{x+2p-w}{s} + 1 \rfloor \times \lfloor \frac{x+2p-w}{s} + 1 \rfloor$$

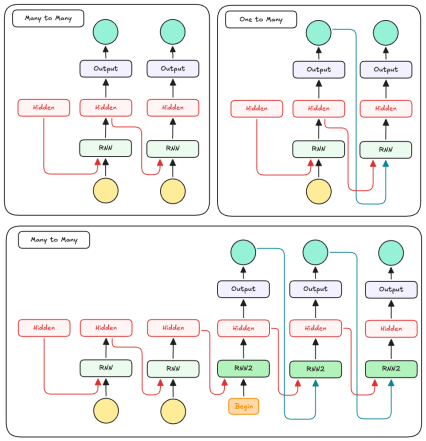
Pooling summarising the feature map
Max-Pool, Average-Pool, Sum-Pool

Receptive fields

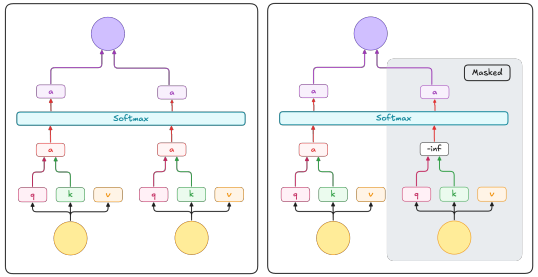
$$r_i = r_{i-1} + (k_i - 1) \cdot j_{i-1}$$

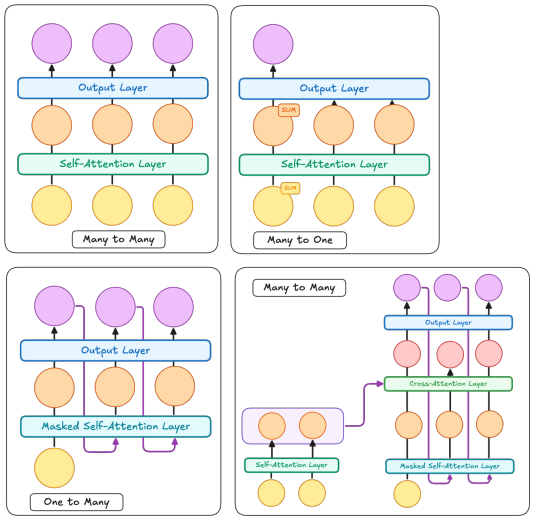
$$j_i = j_{i-1} \cdot s_i$$

RNN



ANN





NN solutions to problems

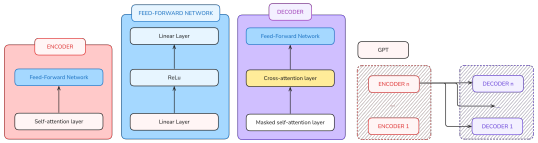
Dropout Randomly set neuron outputs 0 to prevent overfitting.

Early stopping Stop when validation and training loss is at a minimum

Vanishing gradient problem Small gradients multipled repeatedly until zero - Change activation functions

Exploding gradient problem Gradient multiplied until overflowing - Clip gradient within range

Transformers



Metrics

Confusion Matrix

TP	FP (Type 1)
FN (Type 2)	TN

Correctness (Classification)

Accuracy:

$$accuracy = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}^i = y^i)$$

Precision $\mathcal{P}$. Maximise if false positives (Type I) error costly.

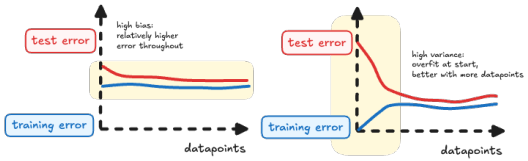
$$\mathcal{P} = \frac{TP}{TP+FP}$$

Recall $\mathcal{R}$. Maximise if false negatives (Type II) error costly.

$$\mathcal{R} = \frac{TP}{TP+FN}$$

$$F_1 = \frac{2}{\frac{1}{\mathcal{P}} + \frac{1}{\mathcal{R}}}$$

Bias & Variance



High bias can cause algorithms to miss relevant relations, resulting in **underfitting**. (Too less features)

High variance can cause algorithms to model random noise, causing **overfitting**. (Too much features)

Preprocessing

Mean normalisation Normalise everything to mean

Min-max scaling Scale everything to $[-1, 1]$