Analytic Geometry

DEF (*Dot Product*)The dot product between two vectors u and v is defined as

$$u^\top v = u_1v_1 + u_2v_2 + \ldots + u_nv_n = \sum_{i=1}^{n} u_iv_i$$

DEF (*Bilinearity*) The dot product is bilinear, i.e. for any vectors u, v, w and scalar a,

$$au^\top v = au^\top v = u^\top av$$
$$u + v^\top w = u^\top w + v^\top w$$
$$w^\top u + v = w^\top u + w^\top v$$

DEF (*Commutativity*)The dot product is commutative, i.e. $u^\top v = v^\top u$

DEF (*Inner Product*) The inner product between two vectors u and v is defined as

$$\langle u, v \rangle$$

Dot product is a special case of inner product.

DEF (ℓ_2 *Norm*) The ℓ_2 norm of a vector v is defined as

$$\|v\|_2 = \sqrt{v^\top v} = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

Also called the Euclidean norm.

DEF (ℓ_2 *properties*) For all vectors u, v and scalar a,

- The ℓ_2 norm is non-negative, i.e. $\|v\|_2 \geq 0$.
- $\|au\|_2 = |a| \|u\|_2$ for any scalar a.
- $\|u\|_2$ is zero if and only if u is the zero vector.
- The triangle inequality holds, i.e. $\|u + v\|_2 \leq \|u\|_2 + \|v\|_2$.
- $\|x - y\|_2 = \|y - x\|_2$, also called symmetry.
- $\|u + v\|_2^2 = \|u\|_2^2 + 2u^\top v + \|v\|_2^2$
- $\cos\theta = \frac{u^\top v}{\|u\|_2 \|v\|_2}$ (can be proved using the law of cosines)

THM (*Cauchy-Schwarz Inequality*) For any vectors u, v, the following inequality holds:

$$|\langle u, v \rangle| \leq \|u\|_2 \|v\|_2$$

DEF (*Line*) A line is a set of points

$$\{x : x = u + tv \text{ for some } t \in \mathbb{R}\}$$

where u is a point on the line and $v \neq 0$ is the direction vector.

DEF (*Plane*) A plane is a set of points

$$\{x : v^\top x - u = 0\}$$

where v is the normal vector to the plane and u is the shift from the origin.

DEF (*Angle between two vectors*) The angle θ between two vectors u and v is given by

$$\cos\theta = \frac{u^\top v}{\|u\|_2 \|v\|_2}$$

DEF (*Projection*) The vector $\|u\|_2 \cos\theta \frac{v}{\|v\|_2}$ is a projection of u onto v.

DEF (*Distance between a point and a plane*) The distance between a point z and a plane $v^\top x - u = 0$ is given by

$$\frac{|v^\top z - u|}{\|v\|_2}$$

DEF (*Distance between a point and a line*) The distance between a point z and a line $x = u + tv$ is given by

$$\left\| z - u - \frac{z - u^\top v}{\|v\|_2^2} v \right\|_2$$

DEF (*Singular Value Decomposition (SVD)*) The SVD of a matrix X is $U\Sigma V^T$, where $U^\top U = I, V^\top V = I$ and Σ is a diagonal matrix:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_n \end{bmatrix}$$

and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$.

THM (*Eckart-Young Theorem*) Let $\Sigma_k = \text{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$ where $k \leq d$. The matrix $U\Sigma_k V^T$ is the optimal solution to the following problem:

$$\min_{\hat{X}} \left\| X - \hat{X} \right\|_F \text{ s.t. } \text{rank}(\hat{X}) \leq k$$

The matrices $Z = U\Sigma_k$ and $W = V^T$ are the optimal solution to

$$\min_{Z,W} \|X - ZW\|_F \text{ s.t. } Z \in \mathbb{R}^{n \times k}, W \in \mathbb{R}^{k \times d}$$

THM (*Useful Identities*)

- $A(A^\top A)^{-1} = (AA^T)^{-1}A^T$

Calculus

DEF (*Derivative*) The derivative of a function $f : \mathbb{R} \to \mathbb{R}$ at x_0 is defined as:

$$(D_x f)(x_0) = \left(\frac{d}{dx} f\right)(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

DEF (*Directional Derivative*) The directional derivative of a function $f : \mathbb{R}^d \to \mathbb{R}$ along the direction v at $x_0 \in \mathbb{R}^d$ is defined as:

$$(D_v f)(x_0) = \lim_{h \to 0} \frac{f(x_0 + hv) - f(x_0)}{h}$$

DEF (*Partial Derivative*) The partial derivative is a directional derivative along the direction of coordinate axes. For a function $f : \mathbb{R}^d \to \mathbb{R}$, the partial derivative with respect to the i-th coordinate is denoted as:

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_d) - f(x_1, \ldots, x_i, \ldots, x_d)}{h}$$

DEF (*Gradient*) The gradient of a function is the vector consisting of all partial derivatives denoted by ∇f or $\frac{\partial f}{\partial x}$. For a function $f : \mathbb{R}^d \to \mathbb{R}$,

EX (*Gradient Identities*)

- Given a function $f(a) = b^\top a$, $(\nabla f)(a) = b$
- Given a function $f(a) = \|a\|_2^2$, $(\nabla f)(a) = 2a$
- Given a function $f(a) = a^\top Aa$, $(\nabla f)(a) = (A + A^T)a$

DEF (*Hessian*) The Hessian of a function $f : \mathbb{R}^d \to \mathbb{R}$ is the matrix of second partial derivatives:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \ldots & \frac{\partial^2 f}{\partial x_1 x_d} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \ldots & \frac{\partial^2 f}{\partial x_2 x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d x_1} & \frac{\partial^2 f}{\partial x_d x_2} & \ldots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}$$

EX (*Hessian Example*) Given a function $f(x,y) = x^2 - y^2$, the Hessian is:x:

$$\nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

The 2 indicates that it looks like a cup along the x-axis and the -2 indicates that it looks like an upside-down cup along the y-axis.

Optimisation

DEF (*Minimum*) The minimum of a function $f : \mathbb{R}^d \to \mathbb{R}$ is written as $\min_x f(x)$, and has the property that $\min_x f(x) \leq f(y)$ for all $y \in \mathbb{R}^d$. The value x^* such that $f(x^*) = \min_x f(x)$ is called the minimizer.

DEF (*Convexity*) A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for any $0 \leq \alpha \leq 1$, we have

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

In general,

$$f(\sum_{i=1}^{k} \alpha_i x_i) \leq \sum_{i=1}^{k} \alpha_i f(x_i)$$

DEF (*Concavity*) A function f is concave if $-f$ is convex.

THM (*First Order Condition (Convexity)*) If f is convex then,

- $f(x) \geq f(y) + \nabla f(y)^\top (x - y)$

DEF (*Positive Semidefinite*) A matrix A is positive semidefinite if for all vectors $v \neq 0$ we have $v^\top A v \geq 0$. Also written as $A \succeq 0$.

THM (*Convex Function Implies Positive Semidefinite Hessian*) If f is convex, then $\nabla^2 f(x) \succeq 0$.

DEF (*Positive Definite*) A matrix A is positive definite if for all vectors $v \neq 0$ we have $v^\top A v > 0$. Also written as $A \succ 0$.

DEF (*Affine*) A function f is affine if $f(x) = Ax + b$ for some matrix A and vector b.

THM (*Affine Transform Preservation*) If f is convex, then $g(x) = f(Ax + b)$ is also convex.

THM (*Non-negative Weighted Sum*) If $f_1, f_2, \ldots, f_k$ are convex functions, then $f(x) = \sum_{i=1}^{k} \beta_i f_i(x)$ is convex for all $\beta_i \geq 0$.

EX (*Gradient of Quadratic Form*) $\nabla_x (x^\top A x) = (A + A^\top)x$

DEF (*Strictly Convex*) A function $f : \mathbb{R}^d \to \mathbb{R}$ is called strictly if for $0 \leq \alpha \leq y$ we have

$$f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y)$$

for any $x \neq y$.

DEF (*First Order Condition (Strict Convexity)*) If f is strictly convex then,

- $f(x) > f(y) + \nabla f(y)^\top (x - y)$

THM (*Unique Minimizer*) If f is strictly convex, then f has a unique minimizer.

THM (*Jensen's Inequality*) If a function f is convex,

$$f(\mathbb{E}_{x\sim p}[x]) \leq \mathbb{E}_{x\sim p}[f(x)]$$

DEF (*Subgradient*) A subgradient at x is a vector g that satisfies

$$f(y) \geq f(x) + g^\top (y - x)$$

for any y, and the set of subgradients at x is denoted as $\partial f(x)$. $\nabla f(x) \in \partial f(x)$ if f is differentiable at x. In other words subgradients are tangents that are below the function.

EX (*Constrained Optimisation Problem*) An example of a constrained optimisation problem is

$$\min_x x^2 \text{ s.t. } -2.5 \leq x \leq -0.5$$

DEF (*Feasible Solution*) A feasible solution is a point that satisfies all the constraints.

DEF (*Lagrangian*) If you have an optimisation problem of the form

$$\min_x f(x) \text{ s.t. } h(x) \leq 0$$

the **Lagrangian** is defined as

$$f(x) + \lambda h(x)$$

for some $\lambda \geq 0$ (Lagrange multiplier).

ALG (*Solving the Lagrangian*)

- Solve $g(\lambda) = \min_x [f(x) + \lambda h(x)]$
- Find $\hat{\lambda}$ such that $\min_x [f(x) + \hat{\lambda} h(x)]$ gives a feasible solution
- Suppose $\hat{x}$ is the solution to the above, and $x^* = \arg\min_{x:h(x)\leq 0} f(x)$ (the optimal solution), then

$$f(\hat{x}) = f(\hat{x}) + \hat{\lambda} h(\hat{x}) \leq f(x^*) + \hat{\lambda} h(x^*) \leq f(x^*)$$

Probability

DEF (*Gaussian Distribution*) We write $x \sim \mathcal{N}(\mu, \Sigma)$ to denote that x is a random variable with mean μ and covariance Σ. It means that the probability density function of x is given by

$$p(x) = \frac{1}{2\pi^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

DEF (*Statistical Independence*) Two variables x and y are independent if

$$p(x,y) = p(x)p(y)$$

Equivalently,

$$p(x|y) = p(x)$$

. The independence of x and y is denoted by $x \perp y$.

DEF (*Statistical Independence [general]*) If $\{x_1, \ldots, x_n\} \perp \{y_1, \ldots, y_m\}$ then

$$p(x_1, \cdots, x_n, y_1, \cdots, y_m) = p(x_1, \cdots, x_n) p(y_1, \cdots, y_m)$$

EX (*Factorisation of a joint distribution*) Suppose $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$. If $\{x, y\} \perp z$ then

$$p(x,y,z) = p(x,y)p(z)$$

The original joint distribution (of size $|\mathcal{X}| \times |\mathcal{Y}| \times |\mathcal{Z}|$) can be factorised into two distributions of size $|\mathcal{X}| \times |\mathcal{Y}|$ and $|\mathcal{Z}|$.

DEF (*Mutual Independence*) A set of variables $\{x_1, \ldots, x_n\}$ are mutually independent if

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i)$$

DEF (*Pairwise Independence*) A set of variables $\{x_1, \ldots, x_n\}$ are pairwise independent if

$$p(x_i, x_j) = p(x_i)p(x_j)$$

for all $i \neq j$.

THM (*Mutual Independence implies Pairwise Independence*) If a set of variables $\{x_1, \ldots, x_n\}$ are mutually independent, then they are pairwise independent. Converse is not true!

DEF (*Conditional Independence*) The variables x and y are conditionally independent given z if

$$p(x, y|z) = p(x|z)p(y|z)$$

This is denoted by $x \perp y|z$.
DEF (*Marginilisation*) The marginal distribution of x is obtained by summing out all other variables.

$$p(x) = \sum_y p(x, y)$$
$$p(x|z) = \sum_y p(x, y|z)$$
$$p(x, y|z) = p(x|z)p(y|z)$$

DEF (*Bayes Rule*) Bayes rule is a way to invert conditional probabilities.

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

DEF (*Chain Rule*) Any joint distribution can be factorised into a product of conditional distributions.

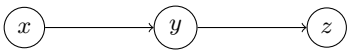
$$p(x_1, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_n|x_1, \ldots, x_{n-1})$$

DEF (*(directed) Graph Representation*) A directed graph is a set of nodes connected by edges. Each vertex is a random variable and each edge represents a direct dependency. It is directed and acyclic (DAG). A distribution factorises according to the graph if

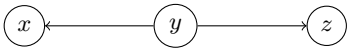
$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i|\text{parents}(x_i))$$

DEF (*Graph structures*)

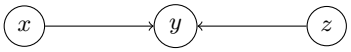
- Chain $x \perp z|y$



- Common cause $x \perp z|y$



- v-structure $x \perp z$ but $x \not\perp z|y$



Training

DEF (*Loss Function*) Given a predicted output $\hat{y}$ and observed output y, the loss function measures how close the model's prediction is
DEF (*Zero-One Loss*) $L(y, \hat{y}) = \mathbb{I}_{\{y \neq \hat{y}\}}$
DEF (*Mean Squared Error (MSE)*) $L(y, \hat{y}) = (y - \hat{y})^2$
DEF (*Hinge Loss*) $L(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y})$
DEF (*Gradient Descent*) The gradient descent algorithm is given by

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

where η is the learning rate/step size.
DEF (*Convergence*) Given $\epsilon > 0$, we say that an algorithm converges to a point x^* if

$$f(x_t) - f(x^*) \leq \epsilon$$

DEF (*Convergence Rate*) The convergence rate of an algorithm is the rate at which the algorithm converges to the optimal point. There are three types of convergence rates:

- Sublinear: $f(x_t) - f(x^*) \leq \frac{c}{t^2}$ $(\epsilon = O(\frac{1}{t^2}), t = O(\frac{1}{\sqrt{\epsilon}}))$
- Linear: $f(x_t) - f(x^*) \leq cr^t$ $(\epsilon = O(r^t), t = O(\log \frac{1}{\epsilon}))$
- Quadratic: $f(x_t) - f(x^*) \leq cr^{2^t}$

ALG (*Stochastic Gradient Descent (SGD)*) Sample a random point x_t, y_t from the dataset and compute the gradient at that point. Repeat until solution is satisfactory.
ALG (*Mini-batch Gradient Descent*) Sample a mini-batch of points x_t, y_t from the dataset and compute the gradient at that point. Repeat until solution is satisfactory.
DEF (*Training*) The act of minimising the loss function by adjusting the model's parameters is known as training.

Regression

DEF (*Regression*) The learning of relationships between input variables x and a numerical output y
DEF (*Feature Transformation*) We call ϕ a feature transformation. It transforms the input space X into a new space Z.
EX (*Feature Transformation Examples*)

- Degree 2: $\phi(x) : [1, x_1, x_2] \rightarrow [1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$ (quadratic)

DEF (*Linear Regression*) Given a dataset $S = \{(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n)\}$, minimise the MSE loss function

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = w^T \phi(x_i)$
DEF (*Closed-form Solution*) The closed-form solution to linear regression is given by

$$w = (\Phi\Phi^T)^{-1}\Phi y$$

DEF (*Probalistic Interpretation*) The probabilistic interpretation of linear regression is that

$$y = w^T \phi(x) + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, 1)$ which implies $y_i \sim \mathcal{N}(w^T \phi(x_i), 1)$. So the log likelihood (L) of the data is

$$L = \sum_{i=1}^{N} \left[-\frac{1}{2} \log(2\pi) - \frac{1}{2} (y_i - w^T \phi(x_i))^2 \right]$$

Classification

DEF (*Classification*) Given a dataset S, the goal is to learn a function h that maps input variables x to y, where y is a class label.
DEF (*Linear Seperability*) A dataset is linearly seperable if there exists a hyperplane that can seperate the data points
DEF (*Training of Classification*) Find parameters θ such that the zero-one loss is minimised
DEF (*Logistic Regression (binary)*) A linear classifier that models the probability of a binary class label. The model is given by

$$p(y|x, \theta) = \sigma(-y\theta^T x) = \frac{1}{1 + \exp(-y(w^\top x + b))}$$

where σ is the sigmoid function. The classification is done by

$$h(x) = \text{sgn}(w^\top x + b)$$

DEF (*Maximum Likelihood Estimation (MLE)*) The MLE of the parameters θ is given by

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \prod_{i=1}^{n} p(y_i|x_i, \theta)$$

DEF (*Log Likelihood*) The log likelihood is applying log to the MLE to obtain a more numerically stable solution:

$$L = \sum_{i=1}^{n} \log p(y_i|x_i, \theta)$$

ALG (*Training of Logistic Regression*) The training of logistic regression is done by maximising the log likelihood L of w and b.

$$L = \sum_{i=1}^{n} \log p(y_i|x_i, \theta)$$

There are no closed-form solutions to this problem, so iterative methods like gradient ascent are used which is equivalent to minimising the negative log likelihood.

DEF (*Multiclass Classification*) Using the softmax function, we can extend logistic regression to multiclass classification. Softmax is defined as

$$\text{softmax}\left(\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}\right) = \begin{bmatrix} \frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}} \\ \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}} \\ \vdots \\ \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \end{bmatrix}$$

DEF (*Logistic Regression (multiclass)*) The model is given by

$$p(y|x, \theta) = \frac{\exp(\theta_y^T x)}{\sum_{i=1}^{k} \exp(\theta_i^T x)}$$

The classification is done by

$$h(x) = \underset{y}{\text{argmax}}\, \theta_y^T x$$

DEF (*Support Vector Machine (SVM) (hard margin)*) A linear classifier that finds the hyperplane that maximises the margin between the classes. It does so by solving the following optimisation problem:

$$\min_w \frac{1}{2} \|w\|_2^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1$$

So the lagrangian is

$$L(\alpha, w) = \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^{n} \alpha_i (y_i(w^T x_i + b) - 1)$$

Solving the lagrangian yields

$$w = \sum_{i=1}^{n} \alpha_i y_i \cdot \overbrace{x_i}^{\text{support vectors}}$$

The classification is done by

$$h(x) = \text{sgn}(w^T x + b) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i x_i^T x + b\right)$$

DEF (*SVM (soft margin)*) Add tolerance to the margin to allow for misclassification. The optimisation problem becomes

$$\min_{w,b}(w^\top w + C \sum_{i=1}^{n} \xi_i) \quad \text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i$$

The constraint is also known as the hinge loss so when solving the lagrangian

DEF (*Kernel*) A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is defined as

$$k(x, x') = \phi(x)^T \phi(x')$$

for some feature function ϕ. It is a measure of similarity between two points in the input space.

EX (*Common Kernels*)

- Linear: $k(x, x') = x^T x'$
- Polynomial: $k(x, x') = (x^T x' + 1)^d$
- Gaussian: $k(x, x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\sigma^2}\right)$
- Hyperbolic Tangent: $k(x, x') = \tanh(kx^T x' + c)$

DEF (*Making Kernels*) A kernel should satisfy

- $k(x, z) = \langle \phi(x), \phi(z) \rangle = \langle \phi(z), \phi(x) \rangle = k(z, x)$ (symmetric)
- $k(z, x)^2 = k(x, x)k(z, z)$
- $$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

  is positive semi-definite

THM (*Mercer's Theorem*) Supoose k is a continuous symmetric non-negative definite kernel, then k can be expressed as

$$k(x, z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(z)$$

where $\{\phi\}$ are eigen-functions, $\|\phi_i\|_2 = 1$ and $\{\lambda_i\}$ are positive eigen values $\lambda_i \geq 0$.

DEF (*Kernel Trick*) The kernel trick is a method to transform the input space into a higher dimensional space without explicitly computing the transformation. This is done by using a kernel function

$$k(x, x') = \phi(x)^T \phi(x')$$

where ϕ is the transformation function.

Neural Networks

DEF (*Perceptron (Single-layer Neural Network)*) A simple linear classifier that learns a weight vector w to classify data points. The perceptron training algorithm can be used to learn the weights. Can model logical functions like AND, OR and NOT.

THM (*Universal Approximation Theorem*) A single-output node NN with a single hidden layer with finite neurons can app finite neurons can approximate continuous (and discontinuous) functions

DEF (*Multi-layer Perceptron (MLP) / Feedforward NN*) A neural network with multiple layers of perceptrons. The network is feedforward as there are no cycles in the network. It can model complex decision boundaries (piecewise linear) [XOR, etc] The perceptron training algorithm can not be used to learn the weights.

The notation $w_{ij}^{(l)}$ denotes the weight from the ith neuron in layer $l-1$ to the jth neuron in layer l.

DEF (*Activation Function*) Allows for non-linear decision boundaries. They should be differentiable. Also controls the output range to a specific range. Common activation functions are:

- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic Tangent (tanh): $\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$
- ReLU: $f(x) = \max(0, x)$ (faster than tanh)
- Leaky ReLu solves the 'dying ReLU' problem

DEF (*Computation Graph*) Represents computation as a directed graph comprising of simple operations on vectors and matrices which allows for automatic differentiation.

DEF (*Training of Single-Layer Network*) Given an error function $E_n = \frac{1}{2}(y_n - \hat{y}_n)^2$ (MSE), where $\hat{y}_n = g(a_{nk})$ (some activation function g) [usually sigmoid], and $a_{nk} = w_k^T x_n$ (weighted sum of inputs), the weights are updated using the gradient descent algorithm

$$w_{t+1} = w_t - \eta \frac{\partial E_n}{\partial w_k}$$

By chain rule,

$$\frac{\partial E_n}{\partial w_k} = \frac{\partial E_n}{\partial \hat{y}_n} \frac{\partial \hat{y}_n}{\partial a_{nk}} \frac{\partial a_{nk}}{\partial w_k}$$

0.1. Generalisation

DEF (*Independtly and Identically Distributed (i.i.d.)*) A dataset is said to be i.i.d. if the data points come from the same distribution and are statistically independent of each other.

DEF (*Training Error*) For a training set S, the training error for a loss ℓ and a program h is defined as

$$L_S(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(x_i))$$

DEF (*Generalisation Error*) For a program h, the generalisation error is defined as

$$Ł_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(y, h(x))\right]$$

The goal of learning is to minimise the generalisation error.

DEF (*Hypothesis Class*) A hypothesis class $\mathcal{H}$ is a set of possible programs of a particular form.

EX (*Hypothesis Class Example*) The set of all linear functions is a hypothesis class i.e.

$$\mathcal{H}_{lin} = \{h(x) = w^T x \mid w \in \mathbb{R}^d\}$$

DEF (*Learning Algorithm*) A learning algorithm is a function that takes a data set of size m and returns a function from the hypothesis class $\mathcal{H}$

THM (*Probably Approximately Correct (PAC)*) A hypothesis class $\mathcal{H}$ is PAC-learnable with a learning algorithm A if for any distribution $\mathcal{D}$, and any $\epsilon > 0$ and $0 \leq \delta \leq 1$, there exists $N > 0$ such that for any $n \geq N$:

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[L_{\mathcal{D}}(A(S)) - \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') > \epsilon\right] < \delta$$

In other words, with high probability, the program learned by A achieves a similar error to the best program in $\mathcal{H}$.

DEF (*Empirical Risk Minimisation (ERM)*) Minimising the loss on a training set is also known as empirical risk minimisation

$$A_{ERM,\mathcal{H}(s)} = h_{ERM} = \operatorname*{argmin}_{h\in\mathcal{H}} L_S(h)$$

THM (*No free lunch theorem*) Suppose $|\mathcal{X}| = 2m$. For any learning algorithm A, there is a distribution $\mathcal{D}$ and $f : \mathcal{X} \to \{0, 1\}$ such that $L_{\mathcal{D}}(f) = 0$, but

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[L_{\mathcal{D}}(A(S)) \geq \frac{1}{10}\right] \geq \frac{1}{10}$$

THM (*Error Decomposition*)

$$Ł_{\mathcal{D}}(h) = \underbrace{L_{\mathcal{D}}(h) - \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h')}_{\text{Estimation Error}} + \underbrace{\min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') - L_S(h)}_{\text{Approximation Error}}$$

DEF (*Uniform Convergence*) A hypothesis class $\mathcal{H}$ has uniform convergence property if for any distribution $\mathcal{D}$, and any $\epsilon > 0$ and $0 \leq \delta \leq 1$, there exists $N > 0$ such that for any $n \geq N$ and every $h \in \mathcal{H}$:

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[|L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\right] < \delta$$

THM (*Uniform Convergence implies PAC learnability*) If a hypothesis class $\mathcal{H}$ has the uniform convergence property, then it is PAC learnable with ERM.

DEF (*Shattering*) A set of n points is shattered by $\mathcal{H}$ if there is an arrangement of n points such that classifiers in $\mathcal{H}$ can produce all 2^n ways of labelling the points.

DEF (*Vapnik-Chervonenkis (VC) Dimension*) VC Dimension is the largest number of points that $\mathcal{H}$ can shatter

DEF (*VC Generalisation Bounds*) With probability $1 - \delta$, for all $h \in \mathcal{H}$:

$$L_{\mathcal{D}}(h) \leq L_S(h) + 2\sqrt{\frac{8d\log(en/d) + 2\log(4/\delta)}{n}}$$

d is the VC dimension of $\mathcal{H}$.

EX (*(*)VC Dimension)

- For linear classifers, VC-dim$(\mathcal{H}_{lin}) = p + 1$
- For MLP with p edges, VC-dim$(\mathcal{H}_{mlp}) = O(p \log p)$

DEF (*Surrogate Loss*) A surrogate loss is a loss function that is easier to optimise than the original loss function. Cross entropy or log likelihood are common surrogate losses for 0-1 loss.

DEF (*Underfitting*) A model h is underfitting if there is another model f that has a lower training i.e. $L_S(f) < L_S(h)$.

DEF (*Overfitting*) A model h is overfitting if there is another model f that has a higher training error (S) but a lower test error (S') i.e. $L_S(f) > L_S(h)$ and $L_{S'}(f) < L_{S'}(h)$.

DEF (*Development Set*) A development set is a subset of the training set that is used to tune hyperparameters.

DEF (*Stability*) A learning algorithm is stable if the learned program does not change much in performance when we change the data set slightly.

THM (*Stability prevents Overfitting*) Stable learning algorithms don't overfit

DEF (λ-*strongly convex*) A function f is λ-strongly convex if for all x, y and $\lambda > 0$:

$$f(y) \geq f(x) + \nabla f(x)^\top y - x + \frac{\lambda}{2}\|y - x\|_2^2$$

THM (L_2 *Regularisation*) Regularisation is a technique to prevent overfitting by adding a penalty term to the loss function. For L_2 regularisation, the loss function is modified to

$$L = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(x_i)) + \frac{\lambda}{2}\|w\|_2^2$$

Note that if the loss function is convex, then the regularised loss function is λ-strongly convex.

DEF (*Hardness of Optimising Neural Networks*) Training a 2-layer 3 node neural network is NP-complete. If we could minimise the loss function in polynomial time, then we would be able to solve the P=NP problem.

DEF (*Overparameterisation*) Overparameterisation is the practice of using more parameters than necessary to fit the training data. It helps with optimisation.

DEF (*Interpolation*) A model interpolates the data if it achieves zero training error.

0.2. PCA

DEF (*Dimensionality Reduction*) Dimensionality reduction is the process of reducing the number of features in a dataset. This is used for visualisation, exploration and compression.

DEF (*Principal Component Analysis (PCA)*) PCA is a method for dimensionality reduction. It finds the directions that maximise the variance in the data. These directions are called principal components. Each principle component is orthogonal to the others.

DEF (*Maximum Variance Formulation*) An optimisation problem that PCA solves is to find the directions that maximise the variance in the data. This can be solved with lagrange multipliers.

Expectation Maximisation (EM)

DEF (*Expectation Maximisation*) If can't solve an optimisation problem directly, EM is used. It yields a lower bound of the original loss. It utilises Jensen's inequality.

DEF (*K-means Distortion*)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

DEF (*K-means Steps*)

- Randomly initialise $\mu_{k=1,\dots,K}$
- Minimise J with respect to r_{nk} (Expectation)
- Minimise J with respect to μ_k (Maximisation)
- Repeat until convergence

DEF (*K-means Solutions*)

$$\mu_k = \underbrace{\frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}}_{\text{Mean of cluster}}, (k = \arg\min_j \|x_n - \mu_j\|^2)$$

DEF (*GMM Probabilities*)

$$p(z_k = 1) = \pi_k, p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}, p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}$$

DEF (*GMM Responsibility Function*)

$$r_{nk} = p(z_k = 1|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

DEF (*Kullback-Leibler Divergence*) Measure how well one probability distribution approximates another

$$KL(p\|q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] = \sum_{x\in X} p(x) \log \frac{p(x)}{q(x)}$$

Not a metric as it is not symmetric $KL(p\|q) \neq KL(q\|p)$