1 Linear Algebra

Vector Properties

Linear independence — Linear combination $A\boldsymbol{u} = u_1\boldsymbol{a_1} + ... + u_n\boldsymbol{a_n} = \sum_{i=1}^{n} u_i\boldsymbol{a_i}$, where A is a vector matrix and $\boldsymbol{u}$ provides scaling values, is linearly independent if any of the following holds:

- If equation system $A\boldsymbol{u} = 0$ then $\boldsymbol{u} = 0$
- A is full rank

Unique representation theorem: Any vector v that can be represented by a set of linearly independent vectors $\boldsymbol{a_1}, ..., \boldsymbol{a_n}$ has a unique representation $v = \sum_{i=1}^{n} u_i\boldsymbol{a_i}$ in terms of these vectors

Unit vector — $\boldsymbol{u} = \frac{\vec{\boldsymbol{u}}}{\|\vec{\boldsymbol{u}}\|}$, therefore $\|\boldsymbol{u}\|^2 = 1$

Inner product — $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}^\top \boldsymbol{v} = \sum_{i=1}^{n} u_i v_i = cos(\varphi)\ \|\boldsymbol{u}\|\ \|\boldsymbol{v}\|$ resp. $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_W = \boldsymbol{u}^\top W \boldsymbol{v} = \sum_i u_i v_i w_i$ where W is either a diagonal matrix with $w_i > 0$ or a pd matrix — Properties:

- $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{u}$
- $(\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{w} = \boldsymbol{u} \cdot \boldsymbol{w} + \boldsymbol{v} \cdot \boldsymbol{w}$
- $(\alpha \boldsymbol{u}) \cdot \boldsymbol{v} = \alpha(\boldsymbol{u} \cdot \boldsymbol{v})$
- Positive definite: $\boldsymbol{u} \cdot \boldsymbol{u} \geq 0$
- $\boldsymbol{u} \cdot \boldsymbol{u} = 0 \Leftrightarrow \boldsymbol{u} = 0_v$

ℓ_2 *norm* — $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u} \cdot \boldsymbol{u}}$ resp. $\|\boldsymbol{u}\|_W^2 = \boldsymbol{u}^\top W \boldsymbol{u} = \sum_i u_i^2 w_i$ where W is a diagonal matrix with $w_i > 0$ — Properties:

- $\|\alpha \boldsymbol{u}\| = |\alpha| \|\boldsymbol{u}\|$
- Positive definite (see inner product)
- Equals 0 for zero vector (see inner product)

ℓ_1 *norm* — $|\boldsymbol{u}| = \sum_i |u_i|$

Distance and angle between two vectors —

- Distance: $d = \|\boldsymbol{u} - \boldsymbol{v}\| = \sqrt{(u_1 - v_1)^2 + ... + (u_n - v_n)^2}$
- Angle: $cos(\varphi) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|}\|$

Cauchy Schwarz inequality — $|\boldsymbol{u} \cdot \boldsymbol{v}| \leq \|\boldsymbol{u}\|\ \|\boldsymbol{v}\|$ with equality iff $\varphi = 0$ i.e. $\boldsymbol{u} = \alpha \boldsymbol{v}$ or if $\boldsymbol{u}$ or $\boldsymbol{v} = 0_v$

Proof:

- First direction of proof: If $\boldsymbol{u} = \alpha \boldsymbol{v}$ or $\boldsymbol{u}$ or $\boldsymbol{v} = 0_v$, we can show that the equality holds
- Second direction of proof: If $\boldsymbol{u} \neq \alpha \boldsymbol{v}$ or $\boldsymbol{u}$ and $\boldsymbol{v} \neq 0_v$, we can show that the inequality cannot hold:
  - $\boldsymbol{u}$ can be decomposed into $\boldsymbol{u_v} + \boldsymbol{u_{v^\perp}}$
  - Then, we have $\|\boldsymbol{u} \cdot \boldsymbol{v}\| = \|(\boldsymbol{u_v} + \boldsymbol{u_{v^\perp}}) \cdot \boldsymbol{v}\| = \|\boldsymbol{u_v}\| \cdot \|\boldsymbol{v}\|$
  - Based on Pythagorean theorem, we know that $\|\boldsymbol{u}\|^2 > \|\boldsymbol{u_v}\|^2$
  - Then, we have $\|\boldsymbol{u} \cdot \boldsymbol{v}\| = \|\boldsymbol{u_v}\| \cdot \|\boldsymbol{v}\| < \|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|$

Triangle inequality — $\|\boldsymbol{u} + \boldsymbol{v}\| \leq \|\boldsymbol{u}\| + \|\boldsymbol{v}\|$ resp. $\|\boldsymbol{u} - \boldsymbol{v}\| \leq \|\boldsymbol{u}\| + \|\boldsymbol{v}\|$ with equality iff $\varphi = 0$ i.e. $\boldsymbol{u} = \alpha \boldsymbol{v}$ or if $\boldsymbol{u}$ or $\boldsymbol{v} = 0_v$

Other inequalities —

- $\|n^k\| \leq \|n\|^k$
- $|\sum_i n_i| \leq \sum_i |n_i|$

Orthogonal vectors — Properties:

- $\boldsymbol{u} \cdot \boldsymbol{v} = 0$
- $\|\boldsymbol{u} + \boldsymbol{v}\|^2 = \|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2$
- *Pythagorean theorem*: $\|\boldsymbol{u} - \boldsymbol{v}\|^2 = \|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2$
- Non-zero pairwise orthogonal vectors $\boldsymbol{u_n}$ and $\boldsymbol{u_m}$ are linearly independent
  Proof:
  - Let $\sum_n \alpha_n \boldsymbol{u_n} = 0_v$
  - Then, $0_v \cdot \boldsymbol{u_m} = (\sum_n \alpha_n \boldsymbol{u_n}) \cdot \boldsymbol{u_m} = \sum_n \alpha_n (\boldsymbol{u_n} \cdot \boldsymbol{u_m}) = \alpha_m \|\boldsymbol{u_m}\|^2$ for the case $m = n$, since in all other cases $m \neq n$, the inner product $\boldsymbol{u_n} \cdot \boldsymbol{u_m} = 0$ due to orthogonality
  - Then, $\alpha_m = 0$ for all m, meaning that all $\boldsymbol{u_m}$ are linearly independent

Orthonormal vectors — Vectors are orthonormal iff $\|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1$ and $\boldsymbol{u} \cdot \boldsymbol{v} = 0$

Projection — Projection of $v \in V$ onto $s \in S$ given by: $v_S = \frac{\boldsymbol{v} \cdot \boldsymbol{s}}{\|\boldsymbol{s}\|^2}\boldsymbol{s} = (\boldsymbol{v} \cdot \boldsymbol{s})\boldsymbol{s}$ if $\boldsymbol{s}$ is a unit vector

Vector Spaces

Vector space V — Properties:

- Additive closure: If $\boldsymbol{u}, \boldsymbol{v} \in V$ then $\boldsymbol{u} + \boldsymbol{v} \in V$
- Scalar closure: If $\boldsymbol{u} \in V$ then $\alpha \boldsymbol{u} \in V$
- $\exists 0_v$ such that $\boldsymbol{u} + 0_v = \boldsymbol{u}$
- $\exists -\boldsymbol{u}$ such that $\boldsymbol{u} + -\boldsymbol{u} = 0_v$
- $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$
- $(\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w} = \boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w})$
- $\alpha(\beta \boldsymbol{u}) = (\alpha\beta)\boldsymbol{u}$
- $\alpha(\boldsymbol{u} + \boldsymbol{v}) = \alpha \boldsymbol{u} + \alpha \boldsymbol{v}$
- $\boldsymbol{u}(\alpha + \beta) = \alpha \boldsymbol{u} + \beta \boldsymbol{u}$

Subspace S — Properties: S is a subspace of V iff (*subspace test*):

- $0_v \in S$
- Additive closure
- Scalar closure

Proof:

- If S is a subspace of V subspace properties immediately follow
- If subspace properties are satisfied for S, S must be a subspace of V because operations are inherited (for addition, multiplication) resp. can be derived from subspace properties (for $0_V, -v$)

Extensions of subspaces:

- The intersection of multiple subspaces is a subspace, since we can derive zero vector, additive closure and scalar closure (subspace test)
- *Direct sum* of multiple subspaces:
  - Is given by the Cartesian product $U \oplus V = \{(u, v)\}$, i.e. each element in $U \oplus V$ is an ordered pair of vectors
  - Is equipped with componentwise operations: $(u_1, v_1) + (u_2, v_2) = (u_1 + u_2, v_1 + v_2)$ and $a(u, v) = (au, av)$
  - Is a subspace, since we can derive zero vector, additive closure and scalar closure (subspace test)

Invariant subspace H — H is an invariant subspace of S spanned by S if $S\boldsymbol{h} \in H$ for all $\boldsymbol{h} \in H$ — Properties:

- S has an eigenvector in H
- If S is symmetric, $H^\perp$ is also an invariant subspace of S

Orthogonal complement $S^\perp$ — Subspace, composed of set of vectors that are orthogonal to S — Properties:

- The intersection of S and $S^\perp$ is $\{0_v\}$
- $\dim(S) + \dim(S^\perp) = \dim(V)$

Span — Span of $\{\boldsymbol{s_i}\}_{i=1}^n$ is the set of all vectors that can be expressed as a linear combination of $\{\boldsymbol{s_i}\}_{i=1}^n$: $\sum_{i=1}^n u_i \boldsymbol{s_i}$

Span of matrix A is the span of its column vectors: $A\boldsymbol{u} = u_1\boldsymbol{a_1} + ... + u_n\boldsymbol{a_n} = \sum_{i=1}^n u_i\boldsymbol{a_i}$

A span is a subspace, since for a linear combination, we can derive zero vector, additive closure and scalar closure (subspace test)

(Orthonormal) basis — Unique set of all (orthonormal) vectors $\{\boldsymbol{s_i}\}_{i=1}^n$ that are linearly independent and span the whole of a subspace.

- *Orthonormal representation theorem*: Any vector $\boldsymbol{x} \in S$ can be expressed as linear combination of resp. projection to orthonormal basis: $\boldsymbol{x} = \sum_i (\boldsymbol{x} \cdot \boldsymbol{s_i})\boldsymbol{s_i}$
- *Parseval's theorem*: Extension of orthonormal representation theorem: $\boldsymbol{x} \cdot \boldsymbol{y} = \sum_i (\boldsymbol{x} \cdot \boldsymbol{s_i})(\boldsymbol{y} \cdot \boldsymbol{s_i})$ resp. $\|\boldsymbol{x}\|^2 = \sum_i |(\boldsymbol{x} \cdot \boldsymbol{s_i})|^2$
- *Gram Schmidt orthonormalization*: Procedure to generate orthonormal basis $\{\boldsymbol{s_i}\}_{i=1}^n$ from linearly independent vectors $\{\boldsymbol{x^{(i)}}\}_{i=1}^n$:
  - $\tilde{\boldsymbol{s_1}} = \boldsymbol{x_1}$
  - $\tilde{\boldsymbol{s_k}} = \boldsymbol{x_k} - \sum_{i=1}^{k-1} (\boldsymbol{x_k} \cdot \boldsymbol{s_1})\boldsymbol{s_1}$ for $k > 1$
  - $\boldsymbol{s_i} = \frac{\tilde{\boldsymbol{s_i}}}{\|\tilde{\boldsymbol{s_i}}\|}$

Dimension d —

- A vector space is *finite-dimensional* if $V = span(S)$
- Dimension is given by number of vectors in basis of S
- If $V = \mathbb{R}^n$, each vector has d elements

Convexity —

- A subspace is convex, if $\alpha \boldsymbol{u} + (1 - \alpha)\boldsymbol{v}$ is also in the subspace

Orthogonal vectors in spaces —

- Let S be spanned by orthonormal $\boldsymbol{s_1}, \boldsymbol{s_2}, ...$ and $v \in V$
- *Orthogonal decomposition theorem*: $\boldsymbol{v} = \boldsymbol{v_S} + \boldsymbol{v_{S^\perp}}$ where $\boldsymbol{v} \in V$, $\boldsymbol{v_S} \in S$ and $\boldsymbol{v_{S^\perp}} \in S^\perp$
- *Orthogonality principle*
  - $\boldsymbol{v_S}$ is the projection of $\boldsymbol{v} \in V$ to S iff $(\boldsymbol{v} - \boldsymbol{v_S}) \cdot \boldsymbol{s_i} = 0$
  - This can be rewritten to linear equation system $\boldsymbol{v} \cdot \boldsymbol{s_i} = \boldsymbol{v_S} \cdot \boldsymbol{s_i} = \sum_k \alpha_k (\boldsymbol{s_k} \cdot \boldsymbol{s_i})$ since $\boldsymbol{v_S}$ can be expressed as linear combination of resp. projection to orthonormal basis $\boldsymbol{v_S} = \sum_k \alpha_k \boldsymbol{s_k}$
- $\boldsymbol{v_{S^\perp}} = \boldsymbol{v} - \boldsymbol{v_S} = \boldsymbol{v} - \sum_k (\boldsymbol{v} \cdot \boldsymbol{s_k})\boldsymbol{s_k}$
- *Approximation in a subspace theorem*:
  - Unique best representation of $\boldsymbol{v}$ in S is given by projection of $\boldsymbol{v}$ to S: $\|\boldsymbol{v} - \boldsymbol{s'}\| \geq \|\boldsymbol{v} - \boldsymbol{v_S}\|$ for some arbitrary $\boldsymbol{s'} \in S$
  - Any subset U of S is closest to $\boldsymbol{v}$ iff it is closest to $\boldsymbol{v_S}$
    Proof:
    - $\arg\min_u \|\boldsymbol{v} - \boldsymbol{u}\| = \arg\min_u \|\boldsymbol{v} - \boldsymbol{u}\|^2 = \arg\min_u \|\boldsymbol{v_S} + \boldsymbol{v_{S^\perp}} - \boldsymbol{u}\|^2 = \arg\min_u \|\boldsymbol{v_S} - \boldsymbol{u}\|^2 + \|\boldsymbol{v_{S^\perp}}\|^2$ given Pythagorean theorem $= \arg\min_u \|\boldsymbol{v_S} - \boldsymbol{u}\|^2$
- We have:
  - $\|\boldsymbol{v_S} + \boldsymbol{v}\|^2 = \|\boldsymbol{v_S}\|^2 + \|\boldsymbol{v}\|^2 + 2\boldsymbol{v_S} \cdot \boldsymbol{v} = \|\boldsymbol{v_S}\|^2 + \|\boldsymbol{v}\|^2 + 2\boldsymbol{v_S} \cdot (\boldsymbol{v_S} + \boldsymbol{v_{S^\perp}}) = 3\|\boldsymbol{v_S}\|^2 + \|\boldsymbol{v}\|^2$
  - $\|\boldsymbol{v_S} - \boldsymbol{v}\|^2 = \|\boldsymbol{v_S}\|^2 + \|\boldsymbol{v}\|^2 - 2\boldsymbol{v_S} \cdot \boldsymbol{v} = \|\boldsymbol{v_S}\|^2 + \|\boldsymbol{v}\|^2 - 2\boldsymbol{v_S} \cdot (\boldsymbol{v_S} + \boldsymbol{v_{S^\perp}}) = \|\boldsymbol{v}\|^2 - \|\boldsymbol{v_S}\|^2$

Linear Equations

Let $X\boldsymbol{b} = \boldsymbol{y}$ where $X \in \mathbb{R}^{n \times m}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^n$ and $\boldsymbol{b}$ is unknown

- Number of distinct equations = Number of linearly independent rows in $[X|\boldsymbol{b}] = \text{rank}([X|\boldsymbol{b}]) \leq \min(n, m+1)$
- Number of LHS solutions should = Number of RHS solutions = $\text{rank}(X) \leq \min(n, m)$

Solutions:

- If $\text{rank}(X) < \text{rank}([X|\boldsymbol{b}])$, system is inconsistent (no solution)
- If $\text{rank}(X) = \text{rank}([X|\boldsymbol{b}]) < m$, system is singular (infinitely many solutions) and underdetermined because we have fewer distinct equations than unknowns
- If $\text{rank}(X) = \text{rank}([X|\boldsymbol{b}]) = m = n$, system is non-singular (exactly one solution) and exactly determined
- If $\text{rank}(X) = \text{rank}([X|\boldsymbol{b}]) = m < n$, system is non-singular and overdetermined

General Matrix Properties

Matrices —

- $A \in \mathbb{R}^{n \times m}$ with elements A_{ij}, rows $i = 1, ..., n$, columns $j = 1, ..., m$
- Transpose $A^\top$
- Identity matrix I with 1 on diagonal, 0 elsewhere
- Scalar matrix K with k on diagonal, 0 elsewhere

Operations —

- Element-wise addition: Returns matrix of same size
- Element-wise scalar multiplication: Returns matrix of same size
- Matrix multiplication:
  - $\boldsymbol{A^{n \times p} B^{p \times m}} = \boldsymbol{C^{n \times m}}$
    - $r_v \times c_v = s$
    - $c_v \times r_v = M$
    - $M \times c_v = c_v$
    - $r_v \times M = r_v$
    - $M \times M = M$
  - Element in C is sum-product of row in A and column in B: $C_{ij} = \boldsymbol{A^{(i)}} \cdot \boldsymbol{B^{(j)}}$
  - Column vector in C is a linear combination of the columns in A: $\boldsymbol{C^{(j)}} = \boldsymbol{AB^{(j)}} = \sum_p \boldsymbol{A^{(j=p)}} b_p^{(j)}$

- Row vector in C is a linear combination of the rows in B: $C^{(i)} = A^{(i)}B = \sum_p a_p^{(i)} B^{(i=p)}$
- $C = A[B^{(j=1)}|...|B^{(j=m)}]$
- $C = [A^{(i=1)}|...|A^{(i=n)}]^\top B = [A^{(i=1)}B|...|A^{(i=n)}B]^\top$

Implications —

- $Ae_k = A^{(j=k)}$ and $e_k^\top A = A^{(i=k)}$ where $e_k = 1$ on k^{th} element and 0 everywhere else
- Matrix form:
  - In following $^{(j)}$ refers to column vector and $^{(i)}$ to row vector, however written as column vector
  - $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}^\top \boldsymbol{v} = \sum_i u_i v_i = c$
  - $\boldsymbol{u}\boldsymbol{v}^\top = \boldsymbol{C}$ with $u_i v_j = C_{ij}$
  - $\boldsymbol{A}\boldsymbol{u} = \sum_{j=i} \boldsymbol{A}^{(j)} u_i = \boldsymbol{c}$ with $\boldsymbol{A}^{(i)} \cdot \boldsymbol{u} = \boldsymbol{A}^{(i)\top} \boldsymbol{u} = c_i$
  - $\boldsymbol{u}^\top \boldsymbol{A} = \sum_{j=i} \boldsymbol{A}^{(i)\top} u_j = \boldsymbol{c}^\top$ with $\boldsymbol{u} \cdot \boldsymbol{A}^{(j)} = \boldsymbol{u}^\top \boldsymbol{A}^{(j)} = c_j$
  - $\boldsymbol{A}\boldsymbol{B} = \sum_{j=i} \boldsymbol{A}^{(j)} \boldsymbol{B}^{(i)\top} = \boldsymbol{C}$ with $\boldsymbol{A}^{(i)} \cdot \boldsymbol{B}^{(j)} = \boldsymbol{A}^{(i)\top} \boldsymbol{B}^{(j)} = C_{ij}$
  - $\boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u} = \sum_i \sum_j x_i A_{ij} x_j$, which we can specify:
    - If A is symmetric: $= \sum_{i<j} x_i (A_{ij} + A_{ji}) x_j + \sum_i A_{ii} x_i^2$, since $A_{ij} = A_{ji}$ and so we can only sum over $i \le j$
    - If A is diagonal: $= \sum_i x_i A_{ii} x_i$ since all off-diagonal terms are 0
  - Indexing a vector $\boldsymbol{u}$ on i returns the i^{th} element: $\boldsymbol{u}_i = u_i$
  - Indexing $(\boldsymbol{A}\boldsymbol{u})$ on i returns the i^{th} element for the vector, generated via the i^{th} row for the matrix: $(\boldsymbol{A}\boldsymbol{u})_i = \boldsymbol{A}^{(i)}\boldsymbol{u} = \sum_j A_{ij} u_j$ (! here the row is not written as a column vector, but as a true row vector)
- Moving between instance-level $\rightarrow$ data-level:
  - $\boldsymbol{x}^{(i)}\boldsymbol{y} = \boldsymbol{a} \rightarrow \boldsymbol{X}^\top \boldsymbol{y} = \boldsymbol{a}$ where $\boldsymbol{X}$ consists of rows $\boldsymbol{x}^{(i)}$
  - $\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\top} = \boldsymbol{A} \rightarrow \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{A}$ where $\boldsymbol{X}$ consists of rows $\boldsymbol{x}^{(i)}$
  - $\boldsymbol{x}^{(i)} \cdot \boldsymbol{\beta} = y_i \rightarrow \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ where $\boldsymbol{X}$ consists of rows $\boldsymbol{x}^{(i)}$

Properties —

- $(\boldsymbol{A}+\boldsymbol{B})^\top = \boldsymbol{A}^\top + \boldsymbol{B}^\top$
- $(\alpha \boldsymbol{A})^\top = \alpha \boldsymbol{A}^\top$
- $(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top$
- $(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C} = \boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})$
- $\boldsymbol{A}+\boldsymbol{B} = \boldsymbol{B}+\boldsymbol{A}$
- $\alpha(\boldsymbol{A}+\boldsymbol{B}) = \alpha\boldsymbol{A} + \alpha\boldsymbol{B}$
- $(\alpha+\beta)\boldsymbol{A} = \alpha\boldsymbol{A} + \beta\boldsymbol{A}$
- $(\alpha\beta)\boldsymbol{A} = \alpha(\beta\boldsymbol{A})$
- $(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{x} = \boldsymbol{C}\boldsymbol{x}$
- $(\boldsymbol{A}\boldsymbol{B})\boldsymbol{x} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{x}) = \boldsymbol{C}\boldsymbol{x}$
- $\boldsymbol{A} = 0.5(\boldsymbol{A}+\boldsymbol{A}^\top) + 0.5(\boldsymbol{A}-\boldsymbol{A}^\top) = \boldsymbol{B}+\boldsymbol{C}$ where $\boldsymbol{B}$ is symmetric, but not $\boldsymbol{C}$
- $\boldsymbol{A} = \boldsymbol{A}\boldsymbol{I} = \boldsymbol{I}\boldsymbol{A}$
- $\boldsymbol{A}\boldsymbol{k} = \boldsymbol{A}\boldsymbol{K} = \boldsymbol{K}\boldsymbol{A}$
- $\text{rank}(\boldsymbol{A}\boldsymbol{B}) = \min(\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B}))$
- $\boldsymbol{A}^\top\boldsymbol{A}$ satisfies:
  - Symmetric
  - Psd
  - Has rank m iff it is pd
  - Invertible iff it has rank m and it is pd
  - $\text{rank}(\boldsymbol{A}^\top\boldsymbol{A}) = \text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^\top)$
  - $\text{rank}(\boldsymbol{A}^\top\boldsymbol{A}) = \text{rank}([\boldsymbol{A}^\top\boldsymbol{A}|\boldsymbol{A}^\top\boldsymbol{x}])$

Matrix terminology —

- *Kernel* $\text{null}(\boldsymbol{X})$ contains set of vectors $\boldsymbol{b}$ such that linear map $\boldsymbol{X}\boldsymbol{b} = 0$
- *Nullity* = $\dim(\text{null}(\boldsymbol{X}))$
- *Image* $\text{range}(\boldsymbol{X})$ contains set of vectors $\boldsymbol{b}$ generated by linear map $\boldsymbol{X}\boldsymbol{b}$ resp. is space spanned by columns of $\boldsymbol{X}$
- *Row space* is space spanned by rows of $\boldsymbol{X}$
- *Column rank* = $\dim(\text{colspace}(\boldsymbol{X}))$ = number of linearly independent columns
- *Row rank* = $\dim(\text{rowspace}(\boldsymbol{X}))$ = number of linearly independent rows
- *Rank* = column rank = row rank = $\dim(\text{range}(\boldsymbol{X}))$ = $\dim(\text{range}(\boldsymbol{X}^\top)) \le min(n, m)$
- For invertible $\boldsymbol{B}$, $\text{colspace}(\boldsymbol{X}\boldsymbol{B}) = \text{colspace}(\boldsymbol{X})$ and $\text{rowspace}(\boldsymbol{X}\boldsymbol{B}) = \text{rowspace}(\boldsymbol{X})$
- *Rank nullity theorem*: $\text{Rank}(\boldsymbol{X}) + \text{nullity}(\boldsymbol{X}) = m$

Matrices as linear maps — $\boldsymbol{X}$ maps $\boldsymbol{b}$ from $\mathbb{R}^m$ to $\mathbb{R}^n$: $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{y}$ with $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^n$

- *Injective*: $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{y}$ has at most one solution, happens iff columns of X are linearly independent ($\text{rank}(\boldsymbol{X}) = m \le n$)
- *Surjective*: $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{y}$ has at least one solution, happens iff rows of X are linearly independent ($\text{rank}(\boldsymbol{X}) = n \le m$)
- *Bijective*: Mapping is both injective and surjective, i.e. $m = n$

Projection matrices —
Generally:

- Projection matrix satisfies $\boldsymbol{P} = \boldsymbol{P}^2$
- Proof:
  - Let S be spanned by $\{y_i\}_{i=1}^n$, which are column vectors of the matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
  - Then, $\boldsymbol{A}\boldsymbol{c}$ are linear combinations of $\{y_i\}_{i=1}^n$
  - A vector $(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{c})$ is orthogonal to the columnspace of $\boldsymbol{A}$, if: $\text{columnspace}(\boldsymbol{A}) \cdot (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{c}) = \boldsymbol{A}^\top(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{c}) = \boldsymbol{A}^\top\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{c} = 0$
  - Then, $\boldsymbol{c} = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{x}$
  - With this definition of $\boldsymbol{c}$, we have $\boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top$ as the projection matrix $\boldsymbol{P}$
  - $\boldsymbol{P}^2 = (\boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top)(\boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top) = \boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top = \boldsymbol{P}$

Via orthonormal basis: Let S be spanned by orthonormal $\{\boldsymbol{b}_i\}_{i=1}^n$, which are column vectors of the matrix $\boldsymbol{B} \in \mathbb{R}^{m \times n}$

- Projection of $\boldsymbol{x}$ onto S is given by: $\boldsymbol{u} = \sum_i (\boldsymbol{x} \cdot \boldsymbol{b}_i)\boldsymbol{b}_i = \sum_i \boldsymbol{b}_i\boldsymbol{b}_i^\top\boldsymbol{x} = \boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{x} = \boldsymbol{C}\boldsymbol{x}$
- Projection of $\boldsymbol{x}$ onto $S^\perp$ is given by: $\boldsymbol{x} - \boldsymbol{u} = \boldsymbol{I}\boldsymbol{x} - \boldsymbol{C}\boldsymbol{x}$

Via SVD: Let S be spanned by $\{y_i\}_{i=1}^n$, which are column vectors of the matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

- Projection of $\boldsymbol{x}$ onto S is given by: $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{A}^\#\boldsymbol{x}$ since $\boldsymbol{A}\boldsymbol{A}^\#$ is a projection matrix due to $\boldsymbol{A}\boldsymbol{A}^\# = (\boldsymbol{A}\boldsymbol{A}^\#)^2$
- $\boldsymbol{s} = \boldsymbol{U}_+\boldsymbol{U}_+^\top\boldsymbol{x}$
- *SVD Projection Energy*: $\sum_{l=1}^m |\boldsymbol{u}_k \cdot \boldsymbol{y}_l|^2 = \sigma_k^2$
  Proof: $\sum_{l=1}^m |\boldsymbol{u}_k \cdot \boldsymbol{y}_l| = \|\boldsymbol{u}_k^\top\boldsymbol{A}\| = \boldsymbol{u}_k^\top\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{u}_k = \boldsymbol{u}_k^\top\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{S}^\top\boldsymbol{U}^\top\boldsymbol{u}_k = \boldsymbol{e}_k^\top\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{e}_k = \sigma_k^2$

Square Matrix Properties

Square matrix terminology —

- *Diagonal matrix*:
  - Def: Has $\{d_i\}_{i=1}^n$ on diagonal and 0 everywhere else
  - For diagonal matrices: $\boldsymbol{D}\boldsymbol{D}^\top = \boldsymbol{D}^\top\boldsymbol{D}$
- *Inverse matrix*:
  - Def: $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$
  - Is unique
  - For diagonal matrices: $\boldsymbol{A}^{-1}$ can be calculated by inverting all diagonal elements
- *Symmetric (Hermitian) matrix*:
  - $\boldsymbol{A}^\top = \boldsymbol{A}$
  - Properties:
    - $(\boldsymbol{x} + \boldsymbol{A}^{-1}\boldsymbol{b})^\top\boldsymbol{A}(\boldsymbol{x} + \boldsymbol{A}^{-1}\boldsymbol{b}) - \boldsymbol{b}^\top\boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} + 2\boldsymbol{x}^\top\boldsymbol{b}$
    - If $\boldsymbol{A}$ and $\boldsymbol{B}$ are symmetric, $\boldsymbol{A}+\boldsymbol{B}$ is also symmetric
- *Orthogonal (unitary) matrix*:
  - Def: $\boldsymbol{A}^\top = \boldsymbol{A}^{-1}$
  - $\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{I}$
  - Rows and columns are orthonormal
  - $\|\boldsymbol{A}\boldsymbol{x}\| = \|\boldsymbol{x}\|$
  - $(\boldsymbol{A}\boldsymbol{x}) \cdot (\boldsymbol{A}\boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$
- *Involution matrix*: $\boldsymbol{A}^{-1} = \boldsymbol{A}$
- *Determinant*:
  - Function which maps $\boldsymbol{A}$ to a scalar
  - Properties:
    - $\det(\boldsymbol{I}) = 1$
    - $\det(\boldsymbol{A}\boldsymbol{B}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$
    - $\det(\boldsymbol{A}^\top) = \det(\boldsymbol{A})$
    - $\det(\boldsymbol{A}^{-1}) = (\det(\boldsymbol{A}))^{-1}$
    - $\det(\alpha\boldsymbol{A}) = \alpha^2\det(\boldsymbol{A})$
    - Determinant of diagonal matrix is product of diagonal elements

Invertible matrix theorem — Following statements are equivalent for square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$:

- $\boldsymbol{A}$ is invertible
- Only solution to $\boldsymbol{A}\boldsymbol{x} = 0$ is $\boldsymbol{x} = 0_v$
  Proof:
  - $\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = 0 \Rightarrow \boldsymbol{I}\boldsymbol{x} = 0 \Rightarrow \boldsymbol{x} = 0_v$
- $\boldsymbol{A}$ is non-singular
- Columns (and rows) of $\boldsymbol{A}$ are linearly independent
- $\text{rank}(\boldsymbol{A}) = n$
- $\det(\boldsymbol{A}) = 0$
- Singular values of $\boldsymbol{A}$ are strictly positive

Inversely, if $\boldsymbol{A}$ is not invertible, the columns and rows are not linearly independent, etc.

Matrix inversion lemma —

- Let $\boldsymbol{B} \in \mathbb{R}^{n \times n}$, $\boldsymbol{D} \in \mathbb{R}^{m \times m}$, $\boldsymbol{C} \in \mathbb{R}^{n \times m}$.
  Then, $\boldsymbol{A} = \boldsymbol{B}^{-1} + \boldsymbol{C}\boldsymbol{D}^{-1}\boldsymbol{C}^\top$ is invertible:
  $\boldsymbol{A}^{-1} = \boldsymbol{B} - \boldsymbol{B}\boldsymbol{C}(\boldsymbol{D} + \boldsymbol{C}^\top\boldsymbol{B}\boldsymbol{C})^{-1}\boldsymbol{C}^\top\boldsymbol{B}$
- Let $\boldsymbol{v} \in \mathbb{R}^n$.
  Then,
  $(\alpha\boldsymbol{I} + \boldsymbol{v}\boldsymbol{v}^\top)^{-1}\boldsymbol{v} = (\alpha + \|\boldsymbol{v}\|^2)^{-1}\boldsymbol{v} = \boldsymbol{v}^\top(\alpha\boldsymbol{I} + \boldsymbol{v}\boldsymbol{v}^\top)^{-1} = \boldsymbol{v}^\top(\alpha + \|\boldsymbol{v}\|^2)^{-1}$

Quadratic form — Quadratic form of square matrix $\boldsymbol{M}$: $\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}$. Can be expressed as quadratic form of a symmetric matrix $\boldsymbol{A}$: $\boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{A} = 0.5 \times (\boldsymbol{M} + \boldsymbol{M}^\top) + 0.5 \times (\boldsymbol{M} - \boldsymbol{M}^\top)$.

Eigenvectors and eigenvalues —

- $\boldsymbol{q}$ is an eigenvector of $\boldsymbol{A}$ associated with an eigenvalue λ if it remains on the same line after transformation by a linear map: $\boldsymbol{A}\boldsymbol{q} = \lambda\boldsymbol{q}$
- Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. $\boldsymbol{A}$ can have between $1 - n$ eigenvalues, each with multiple eigenvectors. Eigenvectors for distinct eigenvalues are linearly independent
- *Spectral radius*: $\rho(\boldsymbol{A})$ is the largest eigenvalue of $\boldsymbol{A}$
- If there exists a non-trivial solution for $\boldsymbol{q}$, $(\boldsymbol{A} - \lambda\boldsymbol{I})$ is not invertible and characteristic polynomial $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0$
- *Eigendecomposition resp. diagonalization*: $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$ where $\boldsymbol{Q}$ is a matrix with the eigenvectors as columns and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues on the diagonal
- $\det(\boldsymbol{A}) = \det(\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}) = \prod_{i=1}^n \lambda_i$
- *Symmetric eigendecomposition resp. unitary diagonalization*: For symmetric $\boldsymbol{A}$: $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ where $\boldsymbol{Q}$ is an orthogonal matrix with the eigenvectors as columns and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues on the diagonal
- *Spectral theorem*: Square matrix $\boldsymbol{A}$ is symmetrically diagonizable, iff $\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{A}^\top\boldsymbol{A}$
- *Spectral theorem for symmetric matrices*: Every symmetric matrix $\boldsymbol{A}$ is symmetrically diagonizable (due to Spectral theorem) and all its eigenvalues are real

Positive definite (pd) and positive semi-definite matrices (psd) —

- $\boldsymbol{A} > 0$ iff $\boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} > 0$
- $\boldsymbol{A} \ge 0$ iff $\boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} \ge 0$

Properties:

- If A is p(s)d, αA is also p(s)d
- If A and B are p(s)d, $A+B$ is also p(s)d
- If $\det(A) = \prod_{i=1}^{n} \lambda_i > (\geq)\ 0$ resp. $\{\lambda_i\}_{i=1}^{n} > (\geq)\ 0$ for pd (psd)

Pd properties:

- I is pd
- If A is pd, A^{-1} is pd
- *Cholesky decomposition*: If A is pd, $A = BB^\top$
- If A and B are pd, $(AB)^{-1} = B^{-1}A^{-1}$

Psd properties:

- If A is psd, $BAB^\top$ is psd

Singular Value Decomposition (SVD)

SVD —

- For $A \in \mathbb{R}^{n\times m}$, orthogonal rotation matrix $U \in \mathbb{R}^{n\times n}$, diagonal scaling and projection matrix $S \in \mathbb{R}^{n\times m}$, and orthogonal rotation matrix $V \in \mathbb{R}^{m\times m}$: $A = USV^\top$
- For symmetric $A \in \mathbb{R}^{n\times n}$: $A = USU^\top$
- In S:
  - Diagonal elements $\sigma_1, \ldots$ are the *singular values* of A
  - If $\sigma_1 \geq \sigma_2 \ldots \geq 0$, S is unique
  - *Spectral norm* = $\sigma_{max} = \|A\|_{\text{operator}} = sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$

    Proof:
    - $A = U\Sigma V^T$
    - For any vector $x \in \mathbb{R}^N$, we have: $\|Ax\|_2 = \|U\Sigma V^T x\|_2$
    - Since U is orthogonal, we can write: $\|Ax\|_2 = \|\Sigma V^T x\|_2$
    - Let $y = V^T x$. Substituting, we get: $\|Ax\|_2 = \|\Sigma y\|_2$
    - The diagonal matrix Σ scales the components of y by the singular values σ_i: $\|\Sigma y\|_2 = \sqrt{\sum_{i=1}^{r}(\sigma_i y_i)^2}$
    - The supremum of $\|\Sigma y\|_2$ occurs when all the weight is on the largest singular value σ_1
    - Then, we see that: $\|A\|_2 = \sup_{y\neq 0} \frac{\|\Sigma y\|_2}{\|y\|_2}$ is achieved when y is aligned with the singular vector corresponding to σ_1, giving: $\|A\|_2 = \sigma_1 = \sigma_{\max}(A)$
  - Largest singular value σ_{max} is always greater than largest eigenvalue $\rho(A)$
  - *Condition number* = $\sigma_{max}/\sigma_{min}$
  - For square A: Iff $\sigma_1, \sigma_2, \ldots > 0$, A is invertible
- SVD is closely related to spectral theorem:
  - According to spectral theorem, every matrix A is symmetrically diagonizable (i.e. $A = Q\Lambda Q^\top$), iff $AA^\top = A^\top A$
  - If we apply SVD to $AA^\top$ resp. $A^\top A$:
    - $AA^\top = USV^\top VS^\top U^\top = U(SS^\top)U^\top$ since V is orthogonal and $V^\top V = I$
    - $A^\top A = VS^\top U^\top USV^\top = V(S^\top S)V^\top$ since U is orthogonal and $U^\top U = I$
  - $SS^\top$ and $S^\top S$ are diagonal matrices with elements $\sigma_1^2, \sigma_2^2, \ldots$
  - Given symmetric diagonalization for any matrix, we see that
    - S with σ_i contains square root of eigenvalues of $AA^\top$ resp. $A^\top A$
    - U contains eigenvectors of $AA^\top$ as columns resp. V contains eigenvectors of $A^\top A$ as columns
  - According to spectral theorem, symmetric matrix A is symmetrically diagonizable (i.e. $A = Q\Lambda Q^\top$)
  - If we apply SVD to symmetric matrix A, we see that
    - S contains absolute value of eigenvalues of A
    - U contains eigenvectors of A as columns
- Note for exam: Find orthonormal decomposition resp. SVD decomposition of matrix $A \in \mathbb{R}^{2\times 2}$:

  1) Orthonormal decomposition

  If rows of A are orthogonal:
  1. If required: Take non-zero submatrix of A A
  2. Calculate diagonal scaling matrix S with $S_{ii} = f_i$, where f_i represents how much bigger the norm for the row vector in row i is vs. the norm for the row vector in row 1
  3. Divide elements in row i of A by f_i to obtain A'
  4. Then, we have: $A = SA'$
  5. This ensures that A' has orthogonal columns
  6. Calculate the norm of the rows in A' with the norm for the i^{th} row being n_i
  7. Divide elements in row i of A' by n_i to obtain A''
  8. Multiply elements S_{ii} in diagonal scaling matrix S by n_i to obtain S'
  9. Then, we have: $A = S'A''$
  10. This ensures that A'' is orthonormal

  If columns of A are orthogonal: Similar to above

  2) SVD decomposition:
  1. $A = IS'A''$
  2. Then, $I = U$ in SVD
  3. If submatrix was taken previously, fill up remaining diagonal elements of S' with 0 to match size of original matrix. Then, $S' = S$ in SVD
  4. If submatrix was taken previously, fill up remaining diagonal elements of A'' with 1 to guarantee size of original matrix. Then, $A''^\top = V$ in SVD

Pseudo Inverse —

- Pseudo Inverse satisfies certain conditions that make it behave like an inverse for matrices that might not be invertible in the usual sense
- $A^\# = VS^\# U^\top$
- $S^\# = \begin{bmatrix} S_+^{-1} & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 0 & \ldots & 0 \end{bmatrix}$ is obtained from S by first transposing it and then taking the inverse of non-zero singular values
- $A^\#$ is unique
- If $rank(A)$ = number of rows in A then:
  - $S^\top$ and $S^\#$ have full column rank and $S^\# = S^\top(SS^\top)^{-1}$
  - $AA^\# = I$
  - $A^\# = A^\top(AA^\top)^{-1}$
  - Pseudo inverse provides mininum norm solution, when system $y = Ax$ is underdetermined: $x = A^\top(AA^\top)^{-1}y$
- If $rank(A)$ = number of columns in A then:
  - $S^\top$ and $S^\#$ have full row rank and $S^\# = (S^\top S)^{-1}S^\top$
  - $A^\# A = I$
  - $A^\# = (A^\top A)^{-1}A^\top$
  - Pseudo inverse provides least squares solution, when system $y = Ax$ is overdetermined: $x = (A^\top A)^{-1}A^\top y$

Properties —

- $AA^\# A = A$
- $A^\# AA^\# = A^\#$
- $(A^\top)^\# = (A^\#)^\top$
- $(AA^\top)^\# = (A^\#)^\top A^\#$
- $A^\# x = 0 \Leftrightarrow x^\top A = 0 \Leftrightarrow A^\top x = 0$
- Properties can be proven by replacing A by its SVD and $A^\#$ by its definition
- Column space of $A^\#$ equals column space of $A^\top$
- $AA^\# = USS^\# U^\top = U_+ U_+^\top$
- Property can be proven by replacing A and $A^\#$ by their SVD
- $SS^\# = I_+$

Hilbert Space S

Equivalence modulo norm zero:

- Challenge: In some cases, $v \cdot v = \Leftrightarrow v = 0_v$ does not hold
- Issue is resolved by defining equivalence classes of vectors: $[v] = \{v' \in V : \|v - v'\| = 0\}$

  Proof:
  - $v\,R\,v' \iff \|v - v'\| = 0$
  - We want to show that the relation R is an equivalence relation
  - For this, we show reflexivity, symmetry, and transitivity
  - Reflexivity: since $\|v - v\| = 0$, then $v\,R\,v$
  - Symmetry: since $\|v - v'\| = \|v' - v\| = 0$, then, if $v\,R\,v'$, then $v'\,R\,v$
  - Transitivity: if $v\,R\,v'$ and $v'\,R\,v''$, then $\|v - v'\| = \|v' - v''\| = 0$. Since from the triangle inequality we obtain $\|v - v''\| = \|(v - v') + (v' - v'')\| \leq \|v - v'\| + \|v' - v''\| = 0$, then $v\,R\,v''$
- Implications: Modified meaning of equality:
  - For functions: $f = g$ means $\int_T |f(t) - g(t)|^2 dt = 0$
  - For random variables: $X = Y$ means $\mathbb{E}[|X - Y|^2] = 0$

Existence (convergence) of the inner product:

- Challenge: In some cases, inner product does not exist for all $v, w \in V$
- Issue is resolved by restricting attention to subsets of V where the norm is finite: $V_{fn} = \{v \in |\|v\| < \infty\}$

Hilbert spaces:

- Vector space with an inner product that satisfies the additional condition of *completeness*:
  - Every Cauchy sequence in V converges to an element in V resp.
    limit vectors, that Cauchy sequences tend towards, are also elements of V
  - *Cauchy sequence*: Sequence of points that get closer and closer
- If we make modifications for above challenges (equivalence modulo norm zero, existence of inner product), vector spaces can be transformed to Hilbert spaces

Other

Common exp and log rules —

- $a^m \cdot a^n = a^{m+n}$
- $\frac{a^m}{a^n} = a^{m-n}$
- $(ab)^n = a^n b^n$
- $\left(\frac{a}{b}\right)^n = \frac{a^n}{b^n}$
- $a^{-n} = \frac{1}{a^n}$
- $a^0 = 1$
- $a^1 = a$
- $\log(xy) = \log x + \log y$
- $\log\left(\frac{x}{y}\right) = \log x - \log y$
- $\log(x^n) = n\log x$
- $\log 1 = 0$
- $\log(x < 1) < 0$
- $\log(x > 1) > 0$
- $e^{log(x)} = log(e^x) = x$

Geometric series —

- Finite: $S_n = \sum_{i=1}^{n} a_i r^{i-1} = a_1\left(\frac{1-r^n}{1-r}\right)$
- Infinite: $S = \sum_{i=0}^{\infty} a_i r^i = \frac{a_1}{1-r}$ for $r < 1$

2 Calculus

Derivatives

Rules —

- Sum rule: $\frac{\partial f+g}{\partial x} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial x}$
- Product rule: $\frac{\partial f\times g}{\partial x} = f \times \frac{\partial g}{\partial x} + g \times \frac{\partial f}{\partial x}$
- Chain rule: $\frac{\partial f(g)}{\partial x} = \frac{\partial f}{\partial g} \times \frac{g}{\partial x}$
- Scalar multipliers of the whole gradient can be ignored, even if the variable we are deriving wrt is included in this scalar

Common derivatives —

- $\frac{\partial x^n}{\partial x} = nx^{n-1}$
- $\frac{\partial e^{kx}}{\partial x} = k \times e^{kx}$
- $\frac{\partial log(x)}{\partial x} = \frac{1}{x}$
- $\frac{\partial \sqrt{x}}{\partial x} = \frac{1}{2\sqrt{x}}$
- $\frac{\partial \sin(x)}{\partial x} = \cos(x)$
- $\frac{\partial \cos(x)}{\partial x} = -\sin(x)$

Partial and directional derivative —

- For a function that depends on n variables $\{x_i\}_{i=2}^n$, partial derivative is slope of tangent line along direction of one specific variable x_i
- Directional derivative is slope of tangent line along direction of selected unit vector $\boldsymbol{u}$

Gradient —
- Given scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$, returns vector containing first-order partial derivatives:

$$\nabla_{\boldsymbol{x}} f : [\frac{\partial f}{\partial x_1} ... \frac{\partial f}{\partial x_n}]^\top$$

- Gradient points in direction of greatest upward slope of f
- Magnitude of gradient equals rate of change when moving into direction of greatest upward slope

Hessian —
- Given scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$, returns matrix containing second-order partial derivatives:

$$\mathcal{H} = \nabla_{\boldsymbol{x}}^2 f : \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & ... & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ ... & ... & ... \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & ... & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- $\mathcal{H}$ is symmetric

Jacobian —
- Given vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ with $\boldsymbol{f} = [f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]^\top$, returns matrix containing first-order partial derivatives:

$$\nabla_{\boldsymbol{x}} \boldsymbol{f} : \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & ... & \frac{\partial f_1}{\partial x_n} \\ ... & ... & ... \\ \frac{\partial f_m}{\partial x_1} & ... & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Matrix calculus rules —
- $\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}$
- $\frac{\partial \boldsymbol{x}^\top A \boldsymbol{x}}{\partial \boldsymbol{x}} = (A + A^\top)\boldsymbol{x}$
- $\frac{\partial \boldsymbol{a}^\top A \boldsymbol{b}}{\partial A} = \boldsymbol{a}\boldsymbol{b}^\top$
- For symmetric A: $\frac{\partial \boldsymbol{x}^\top A \boldsymbol{x}}{\partial \boldsymbol{x}} = 2A\boldsymbol{x}$
- For square A:
  - $\frac{\partial \boldsymbol{a}^\top A^{-1} \boldsymbol{b}}{\partial A} = -(A^\top)^{-1}\boldsymbol{a}\boldsymbol{c}^\top(A^\top)^{-1}$
  - $\frac{\partial log(|A|)}{\partial A} = (A^\top)^{-1}$

Extrema

Conditions for local minima and maxima —
- Point is a stationary point, i.e. first-order derivative = 0
- If Hessian is pd, it's a local minimum, if Hessian is nd, it's a local maximum, if Hessian is indefinite, it's a saddle point
- Local minima and maxima are the unique global minima and maxima in strictly convex functions resp. one of possibly infinitely many global minima and maxima in convex functions

Convexity —
- For a convex function:
  - $f(\lambda x + (1-\lambda)y \leq \lambda f(x) + (1-\lambda)f(y)$ with $\lambda \in [0,1]$
  - Hessian of stationary point(s) is psd
  - Global minimum exists, but may not be unique
- For a strictly convex function:
  - $f(\lambda x + (1-\lambda)y < \lambda f(x) + (1-\lambda)f(y)$ with $\lambda \in [0,1]$
  - Hessian of stationary point is pd
  - Unique global minimum exists
- Sum of convex functions $f_2(x) + f_1(x)$ is also convex, sum of convex and strictly convex function is strictly convex
- Chain of convex functions $f_2(f_1(x))$, where outer function $f_2(x)$ is increasing, is also convex
- Scalar multiple of convex function $\lambda f(x)$, where $\lambda \geq 0$, is also convex
- Any norm is convex

Nature of optimum — What does Hessian and function look like?
- If Hessian is pd and loss function is strictly convex, stationary point is a global minimum, and there is a unique solution
- If Hessian is psd and loss function is convex, stationary point is a global minimum, and there may be a geometrically unique or infinitely many solutions
- If Hessian is p(s)d but loss function is not convex, stationary point may be a local minimum and there may be a geometrically unique or infinitely many solutions

Optimization approach — Is function differentiable, continuous, and are relevant terms invertible?
- If yes, analytically solvable
- If no, numerically solvable (e.g. via gradient descent)

Constrained optimization —
- *Lagrangian function*: $\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$, where $g(x)$ is an $(m-1)$ dimensional constraint surface and λ is the Lagrange multiplier
- $\nabla_{\boldsymbol{x}} \mathcal{L} = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) + \lambda \nabla_{\boldsymbol{x}} g(\boldsymbol{x})$
- $\nabla_\lambda \mathcal{L} = g(\boldsymbol{x})$
- Solution is feasible if it fulfills constraints and optimal, if no other feasible solution produces a lower error
- Minimizing over Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$ corresponds to minimizing log-loss:
  - $\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{D}|\boldsymbol{x})\rho(\boldsymbol{x})$
  - $= \arg\min_{\boldsymbol{x}} (-log p(\boldsymbol{D}|\boldsymbol{x}) + k(\boldsymbol{x}))$
  - where $k(\boldsymbol{x}) = -log\rho(\boldsymbol{x})$

  $\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$

For equality constraints: Minimize $f(x)$ subject to $g(x) = 0$
- Gradient of $f(x)$ must be orthogonal to constraint surface, otherwise (if it points into any direction along the constraint surface) $f(x)$ could still decrease for movements along the constraint surface
- On the constraint surface, $g(x)$ is a constant, so moving along any direction on the constraint surface has a directional derivative of 0. Since the gradient of $g(x)$ points into the direction of steepest ascent, it must be orthogonal to the constraint surface, otherwise (if it points into any direction along the constraint surface) $g(x)$ would not be constant on the constraint surface
- Then, gradients are parallel at optimum: $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) = \lambda \times \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^*)$
- To find x^* and λ^*:
  - $\nabla_{\boldsymbol{x}} \mathcal{L} = 0$, expresses parallelity condition at minimum x^*
  - $\nabla_\lambda \mathcal{L} = 0$, expresses constraint
  - This is an unconstrained optimization problem
- Optimum x^* and λ^* represents a saddle point of $\mathcal{L}$

For inequality constraints: Minimize $f(x)$ subject to $g(x) \leq 0$
- If x^* lies in $g(x) < 0$, constraint is inactive
- Otherwise, if x^* lies in $g(x) = 0$, constraint is active:
  - Gradient of $f(x)$ must point towards $g(x) < 0$ region, otherwise (if it would point away from $g(x) < 0$ region) the optimum would lie in this region
  - Then, gradients are anti-parallel at optimum: $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) = -\lambda \times \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^*)$
- To find x^* and λ^*:
  - $\nabla_{\boldsymbol{x}} \mathcal{L} = 0$ subject to *Karush Kuhn Tucker conditions*:
    - $g(\boldsymbol{x}) \leq 0$
    - $\lambda \geq 0$
    - *Complementary slackness condition*: $\lambda g(\boldsymbol{x}) = 0$, with $\lambda = 0, g(\boldsymbol{x}) < 0$ for inactive constraints and $\lambda > 0, g(\boldsymbol{x}) = 0$ for active constraints
  - $\nabla_\lambda \mathcal{L} = 0$ given complementary slackness condition
  - This is not an unconstrained optimization problem, but can be solved via duality
- Optimum x^* and λ^* represents a saddle point of $\mathcal{L}$

For multiple constraints: Minimize $f(x)$ subject to m inequality constraints $\{g^{(i)}(\boldsymbol{x}) \leq 0\}_{i=1}^m$ and p equality constraints $\{h^{(j)}(\boldsymbol{x}) = 0\}_{j=1}^p$
- Then, Lagrangian is given by: $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^m \mu^{(i)} g^{(i)}(\boldsymbol{x}) + \sum_{j=1}^p \lambda^{(j)} h^{(j)}(\boldsymbol{x})$
- Then, general solution x^*, λ^*, μ^* is given by: $\nabla_{\boldsymbol{x}} \mathcal{L} = 0$ subject to:
  - $\{g^{(i)}(\boldsymbol{x}) \leq 0\}_{i=1}^m$ and $\{h^{(j)}(\boldsymbol{x}) = 0\}_{j=1}^p$
  - $\{\mu^{(i)} \geq 0\}_{i=1}^m$
  - $\{\mu^{(i)} g^{(i)}(\boldsymbol{x}) = 0\}_{i=1}^m$

Primal problem:
- $\min_{\boldsymbol{x}}[\max_{\boldsymbol{\lambda},\boldsymbol{\mu}} \mathcal{L}]$
- $\max_{\boldsymbol{\lambda},\boldsymbol{\mu}} \mathcal{L} = f(\boldsymbol{x}) + \max_{\boldsymbol{\lambda},\boldsymbol{\mu}} [\sum_{i=1}^m \mu^{(i)} g^{(i)}(\boldsymbol{x}) + \sum_{j=1}^p \lambda^{(j)} h^{(j)}(\boldsymbol{x})]$
- Second term gives rise to barrier function:
  - $= 0$ subject to constraints being met, given complementary slackness condition for inequality constraints and $h^{(j)}(\boldsymbol{x}) = 0$ for equality constraints, which implies that dual problem becomes $\min_{\boldsymbol{x}}(f(\boldsymbol{x}))$
  - $= \infty$ otherwise, which implies that primal problem cannot be solved
- Let $f(x^*)$ be the solution to the primal problem

Solving inequality constraints via duality:
- *Dual problem*: $\max_{\boldsymbol{\lambda},\boldsymbol{\mu}}[\min_{\boldsymbol{x}} \mathcal{L}]$
- Let $\min_{\boldsymbol{x}} \mathcal{L} = \theta(\boldsymbol{\lambda}, \boldsymbol{\mu})$
- Let $\theta(\lambda^*, \mu^*)$ be the solution to the dual problem

Weak duality:
- Weak duality always holds and gives a lower bound of minimum of primal problem
- Given minimax theorem, $f(x^*) = \min_{\boldsymbol{x}}[\max_{\boldsymbol{\lambda},\boldsymbol{\mu}} \mathcal{L}] = \min_{\boldsymbol{x}}(f(\boldsymbol{x}))$ (provided barrier function) $\geq \max_{\boldsymbol{\lambda},\boldsymbol{\mu}}[\min_{\boldsymbol{x}} \mathcal{L}] = \theta(\lambda^*, \mu^*)$
- $\min_{\boldsymbol{x}} \mathcal{L}$ is an unconstrained optimization problem
- $\max_{\boldsymbol{\lambda},\boldsymbol{\mu}}[\min_{\boldsymbol{x}} \mathcal{L}]$ is a concave maximization problem

Strong duality:
- Strong duality holds under certain conditions, for example *Slater's condition* if there exists a solution that strictly fulfills all constraints $\{g^{(i)}(\boldsymbol{x}) < 0\}_{i=1}^m$ and $\{h^{(j)}(\boldsymbol{x}) < 0\}_{j=1}^p$
- Then, $f(x^*) = \min_{\boldsymbol{x}}[\max_{\boldsymbol{\lambda},\boldsymbol{\mu}} \mathcal{L}] = \max_{\boldsymbol{\lambda},\boldsymbol{\mu}}[\min_{\boldsymbol{x}} \mathcal{L}] = \theta(\lambda^*, \mu^*)$
- $\min_{\boldsymbol{x}} \mathcal{L}$ can be solved for general solution x^* in terms of λ, μ
- Plug x^* back into $\mathcal{L}$ and maximize to find solutions λ^*, μ^*
- Specify x^* based on λ^*, μ^*

Integrals

Indefinite integral —
- $F(x) = \int f(x)dx$
- $F'(x) = f(x)$

Definite integral —
- $F(b) - F(a) = \int_a^b f(x)dx$
- $F'(x) = f(x)$

Common integrals —
- $f(x) = x^n \to F(x) = \frac{x^{n+1}}{n+1}$ for $n \neq 1$
- $f(x) = \frac{1}{x} \to F(x) = log(x)$
- $f(x) = e^x \to F(x) = e^x$

3 Probability and Statistics

Terminology

Notation —
- $A \cap B$ is the intersection of A and B, i.e. A and B
- $A \cup B$ is the union of A and B, i.e. A or (inclusive) B

Kolmogorov axioms — Probability space defined by:

- *Sample space*: All possible outcomes $\Omega = \{\omega_1, ..., \omega_n\}$, e.g. for a dice toss $\{1,2,3,4,5,6\}$
- *Event space*:
  - All possible results
  - Corresponds to the *powerset* of the sample space:
    - Powerset includes the empty set, single-item sets, ..., full-item sets
    - E.g. powerset of $\{1,2,3,4,5,6\}$ is $\{\{\},\{1\},\{2\},...,\{1,2\},\{2,3\}...,\{1,2,3,4,5,6\}\}$ where e.g. event $\{1,2\}$ refers to the event of rolling a 1 or 2
    - Powerset has size $2^{|\Omega|}$
  - An *event* is a subset of the sample space
  - E.g. tossing an even number $\{2,4,6\}$ or $P(X \leq r)$
- *Probability measure*: Function that assigns a probability to an event, e.g. $p(\text{tossing an even number}) = \frac{3}{6} = \frac{1}{2}$

Axioms:

- Event space must be a *sigma algebra*:
  - $\Omega \in$ event space
  - If A is in event space with $P = a$, its complement is also in event space with $P = 1 - a$
  - If $A_1, ...A_n$ are in event space with $P = a_1, ..., a_n$, their union is also in event space with $P = a_1 + ... + a_n$
- Probability measure must satisfy:
  - $0 \leq P(A) \leq 1$
  - $P(\Omega) = 1$
  - If $A_1, A_2, ...$ are in event space and do not intersect, then $P(A_1 \cup A_2 \cup ...) = \int_{n=1}^{\infty} P(A_n)$

Variables —

- *Target space*: Numeric values that the random variable can take, e.g. for a dice toss $\{1,2,3,4,5,6\}$
- *Random variable*:
  - Function that takes an element in sample space and returns a numeric value, e.g. $\mathcal{X}(3) = 3$
  - *Discrete random variable*: Characterized by pmf, event given by $\{\omega \in \Omega | X(\omega) = s\}$
  - *Continuous random variable*: Characterized by pdf, event given by $\{\omega \in \Omega | X(\omega) \leq r\}$ (or $>, =, \geq, <$, any unions and intersections)
  - In general, if case is mixed (e.g. X is discrete, Y is continuous), then the joint probability is defined by the continuous terminology, the marginal probability defined by the terminology of the respective variable, and the conditional probability defined by the terminology of the respective dependent variable
- *Independent random variables*:
  - $P(A|B) = P(A)$ and $P(B|A) = P(B)$
  - $P(A \cap B) = P(A)P(B)$ resp. $F_{X_1,...,X_n}(r_1, ..., r_n) = F_{X_1}(r_1), ..., F_{X_n}(r_n)$ and $f_{X_1,...,X_n}(x_1, ..., x_n) = f_{X_1}(x_1), ..., f_{X_n}(x_n)$

  Proof:
  - Direction 1: $f_{X_1,X_2,...,X_n}(x_1, x_2, \ldots, x_n) = \frac{\partial^n}{\partial x_1 \partial x_2 ... \partial x_n} F_{X_1,X_2,...,X_n}(x_1, x_2, \ldots, x_n) = \frac{\partial^n}{\partial x_1 \partial x_2 ... \partial x_n} F_{X_1}(x_1) F_{X_2}(x_2) \ldots F_{X_n}(x_n) = \frac{\partial}{\partial x_1} F_{X_1}(x_1) \frac{\partial}{\partial x_2} F_{X_2}(x_2) \ldots \frac{\partial}{\partial x_n} F_{X_n}(x_n) = f_{X_1}(x_1) f_{X_2}(x_2) \ldots f_{X_n}(x_n)$
  - Direction 2: $F_{X_1,X_2,...,X_n}(x_1, x_2, \ldots, x_n) = \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} \ldots \int_{-\infty}^{x_n} f_{X_1,X_2,...,X_n}(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n) d\tilde{x}_1 \, d\tilde{x}_2 \ldots d\tilde{x}_n = \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} \ldots \int_{-\infty}^{x_n} f_{X_1}(\tilde{x}_1) f_{X_2}(\tilde{x}_2) \ldots f_{X_n}(\tilde{x}_n) d\tilde{x}_1 \, d\tilde{x}_2 \ldots d\tilde{x}_n = \int_{-\infty}^{x_1} f_{X_1}(\tilde{x}_1) d\tilde{x}_1 \int_{-\infty}^{x_2} f_{X_2}(\tilde{x}_2) d\tilde{x}_2 \ldots \int_{-\infty}^{x_n} f_{X_n}(\tilde{x}_n) d\tilde{x}_n = F_{X_1}(x_1) F_{X_2}(x_2) \ldots F_{X_n}(x_n)$
  - Unnormalized correlation: $\mathbb{E}(\mathcal{X}\mathcal{Y}) = \mathbb{E}(\mathcal{X})\mathbb{E}(\mathcal{Y})$
  - Covariance: $\text{Cov}(\mathcal{X}, \mathcal{Y}) = 0$
  - Functions of independent random variables are also independent
  - A subset of a larger set of independent random variables is also independent

  Proof for discrete case:
  - Assume $X_1, ..., X_n$ are independent
  - We aim to show that $X_1, ..., X_{n-1}$ are also independent
  - $P(X_1, ..., X_{n-1}) = \sum_n P(X_1, ..., X_n) = \sum_n P(X_1)...P(X_n) = P(X_1)...P(X_{n-1}) \sum_n P(X_n) = P(X_1)...P(X_{n-1}) \times 1$

  Proof for continuous case:
  - Assume $X_1, ..., X_n$ are independent
  - We aim to show that $X_1, ..., X_{n-1}$ are also independent
  - $F_{X_1,...,X_{n-1}}(X_1, ..., X_{n-1}) = \lim_{X_n \to \infty} F_{X_1,...,X_{n-1}}(X_1, ..., X_n) = \lim_{X_n \to \infty} F_{X_1}(X_1)...F_{X_n}(X_n) = F_{X_1}(X_1)...F_{X_{n-1}}(X_{n-1}) \lim_{X_n \to \infty} F_{X_n}(X_n) = F_{X_1}(X_1)...F_{X_{n-1}}(X_{n-1}) \times 1$
- *Conditionally independent random variables*: 2 random variables $\mathcal{X}$ and $\mathcal{Y}$ are conditionally independent, if there is a confounder $\mathcal{L}$ that causally affects both variables, but if we control for this confounder, the variables are not causally connected
- *I.I.D. random variables*: Independent and from identical distribution
- *Orthogonal random variables*:
  - Unnormalized correlation: $\mathbb{E}(\mathcal{X}\mathcal{Y}) = 0$
  - Covariance not particularly defined
  - Not necessarily independent
- *Uncorrelated random variables*:
  - Unnormalized correlation: $\mathbb{E}(\mathcal{X}\mathcal{Y}) = \mathbb{E}(\mathcal{X})\mathbb{E}(\mathcal{Y})$
  - Covariance: $\text{Cov}(\mathcal{X}, \mathcal{Y}) = 0$
  - Not necessarily independent

Events —

- *Complement*: $P(A^C) = 1 - P(A)$ and $P(A \cup A^C) = P(A) + P(A^C)$
- *Disjoint / mutually exclusive vs. joint / mutually inclusive*
- Subset $A \subset B$ with $P(A) < P(B)$
- Valid events for continuous random variables: All sets than can be formed from left and right inclusive interval $[0, a]$ are events:
  - $(a, 1] = [0, a]^c \in$ event space, with $P = 1 - a$
  - $(a, b] = ([0, a] \cup (b, 1])^c = ([0, a] \cup [0, b]^c)^c \in$ event space, with $P = 1 - (a + 1 - b) = 1 - a - 1 + b = b - a$
  - $\{0\} \in$ event space, with $P = 0$
  - $\{a\} \in$ event space, with $P = 0$
  - $[a, b] = \{a\} \cup (a, b] = \{a\} \cup ([0, a] \cup [0, b]^c)^c \in$ event space, with $P = 0 + b - a$
  - $[a, b) = [a, b] \backslash \{b\} = ([0, a] \cup [0, b]^c)^c \backslash \{b\} \in$ event space, with $P = b - a - 0$

PMF, CDF, PDF —

- *Cumulative density function (CDF)*: $F(r) = p(X \leq r)$
- *Probability mass function (PMF)* for discrete random variables: $p(x) = p(X = x)$
- *Probability density function (PDF)* for continuous random variables: $f(x)$
- Properties of CDF and PDF:
  - Derivative of CDF by x returns PDF: $f(x) = \frac{\partial F(x)}{\partial x}$
  - Integral of PDF by x returns CDF: $\int_{-\infty}^{r} f(x)dx = F(r) = p(X \leq r)$
  - CDF is monotonically non-decreasing: If $s < r, F(s) < F(r)$
  - CDF is between 0 and 1: $lim_{r \to -\infty} F(r) = 0$ and $lim_{r \to \infty} F(r) = 1$
  - CDF is right-continuous: $lim_{s \to -r^+} F(s) = F(r)$
  - For CDF: $lim_{s \to -r^-} F(s) = F(x < r) = F(s) - F(x = r)$
  - $\int_a^b f(x)dx = F(b) - F(a) = p(a < X \leq b)$
  - $\int_{-\infty}^{\infty} f(x)dx = 1$

Probabilities —

- Probability for single variable:
  Marginal and total probability:
  - If X, Y are discrete: $p(x) = \sum_{\mathcal{Y}} p(x, y) = \sum_{\mathcal{Y}} p(x|y)p(y)$
  - If X, Y are continuous: $f(x) = \int_{-\infty}^{\infty} f(x, y)dy = \int_{-\infty}^{\infty} f(y)f(x|y)dy$ and $F(r) = \int_{-\infty}^{r} \int_{-\infty}^{\infty} f(x, y)dydx$
  - If X is discrete, Y is continuous: $p(x) = \int_{-\infty}^{\infty} f(x, y)dy = \int_{-\infty}^{\infty} p(x|y)f(y)dy$
  - If X is continuous, Y is discrete: $f(x) = \sum_{\mathcal{Y}} f(x, y) = \sum_{\mathcal{Y}} f(x|y)p(y)$ and $F(r) = \sum_{\mathcal{Y}} p(y)p(X \leq r|y) = \sum_{\mathcal{Y}} p(y)F(r|y)$
- *Joint probability* $P(A, B)$: Probability for combination of variables
  - If X is discrete: $p(x_1, ..., x_n)$
  - If X is continuous: $f(x_1, ..., x_n) = \frac{\partial^n F_{X_1,...,X_n}(x_1,...,x_n)}{\partial x_1, ..., \partial x_n}$ and $F(r_1, ..., r_n) = p(x_1 \leq r_1, ..., x_n \leq r_n) = \int_{-\infty}^{r_1} ... \int_{-\infty}^{r_n} f_{X_1,...,X_n}(x_1, ..., x_n)dx_n...dx_1$
- *Conditional probability* $P(A|B) = \frac{P(A \cap B)}{P(B)}$: Probability for variable, given other variable
  - If X, Y are discrete: $p(x|y) = \frac{p(x,y)}{p(y)}$
  - If X, Y are continuous: $f(x|y) = \frac{f(x,y)}{f(y)}$
  - If X is discrete, Y is continuous: $p(x|y) = \frac{f(x,y)}{f(y)}$
  - If X is continuous, Y is discrete: $f(x|y) = \frac{f(x,y)}{p(y)}$
  - Properties:
    - $P(A|B) = 1 - P(A^C|B)$
    - $P(A_1|B) + P(A_2|B) + ... = 1$
    - If conditioning on subset S: $p(x|S) = \begin{cases} p(x)/p(x \in S) & x \in S \\ 0 & x \notin S \end{cases}$
- Bayesian terminology:
  - *Prior* $P(\text{parameter})$
  - *Posterior* $P(\text{parameter}|\text{data})$
  - *Likelihood* $P(\text{data}|\text{parameter})$
  - *Evidence* $P(\text{data})$
- *Bayes theorem*: Posterior $P(A|B) = \frac{\text{Likelihood } P(B|A) \times \text{Prior } P(A)}{\text{Evidence } P(B)}$ where $P(B)$ can be rewritten in marginalized form over A
- Attention! In $p(\cdot|\theta)$ the $|$ can either refer to parametrizing on θ (parameter is part of the distribution's form but isn't observed or fixed) or conditioning on θ (parameter takes a observed and fixed value, and we evaluate the distribution on this condition)

Measures

n^{th} *moment* — $\mathbb{E}(\mathcal{X}^n) = \int_{-\infty}^{\infty} x^n f(x)dx$

Expected value — Generally:

- If X is discrete: $\mathbb{E}(\mathcal{X}) = \sum_{\mathcal{X}} x \times p(x)$
- If X is continuous: $\mathbb{E}(\mathcal{X}) = \int_{-\infty}^{\infty} x \times f(x)dx$
- If Y is discrete: $\mathbb{E}[X] = \sum_{\mathcal{Y}} \mathbb{E}[X|Y = y]p(y)$
- If Y is continuous: $\mathbb{E}[X] = \int_{-\infty}^{\infty} \mathbb{E}[X|Y = y]f(y)dy$

For functions:
- $g(X)$ is a function
- If X is discrete: $\mathbb{E}(g(\mathcal{X})) = \sum_{\mathcal{X}} g(x) \times p(x)$
- If X is continuous: $\mathbb{E}(g(\mathcal{X})) = \int_{-\infty}^{\infty} g(x) \times f(x) dx$

For probabilities:
- Count as functions
- A is an event, X is a random variable
- If X is discrete: $\mathbb{E}[p(X|A)] = \sum_x p(x|A)p(x)$
- If X is continuous: $\mathbb{E}[p(X|A)] = \int_{-\infty}^{\infty} f(x|A)f(x)dx$
- If X is discrete: $\mathbb{E}[p(A|X)] = \sum_x p(A|x)p(x) = p(A)$
- If X is continuous: $\mathbb{E}[p(A|X)] = \int_{-\infty}^{\infty} p(A|x)f(x)dx = p(A)$

For conditions:
- A is an event, X is a random variable
- If X is discrete: $\mathbb{E}(X|A) = \sum_x x \times p(x|A)$ resp.
- If X is continuous: $\mathbb{E}(X|A) = \int_{-\infty}^{\infty} x \times f(x|A)dx$
- $\mathbb{E}(A|X) = P(A|X)$

For vectors:
- Expectation of a vector is the expectation of each of its elements
- If X is discrete: $\mathbb{E}(\boldsymbol{x}) = \sum_{x_1} ... \sum_{x_n} \boldsymbol{x}^\intercal p(x_1, ..., x_n) = \boldsymbol{\mu}$
- If X is continuous: $\mathbb{E}(\boldsymbol{x}) = \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} \boldsymbol{x}^\intercal f_{X_1,...,X_n}(x_1, ..., x_n) dx_1 ... dx_n = \boldsymbol{\mu}$

Properties:
- $\mathbb{E}(\alpha) = \alpha$
- $\mathbb{E}(\alpha\mathcal{X} + \beta) = \alpha\mathbb{E}(\mathcal{X}) + \beta$
- $\mathbb{E}(\alpha\mathcal{X} + \beta\mathcal{Y}) = \alpha\mathbb{E}(\mathcal{X}) + \beta\mathbb{E}(\mathcal{Y})$
- For orthogonal variables:
  - $\mathbb{E}(\mathcal{X}\mathcal{Y}) = 0$
  - $\mathbb{E}((\mathcal{X} + \mathcal{Y})^2) = \mathbb{E}(\mathcal{X}^2) + \mathbb{E}(\mathcal{Y}^2)$
- For independent variables:
  - $\mathbb{E}(\mathcal{X}|\mathcal{Y}) = \mathbb{E}(\mathcal{X})$
  - $\mathbb{E}(\mathcal{X}\mathcal{Y}) = \mathbb{E}(\mathcal{X})\mathbb{E}(\mathcal{Y})$

    Proof:
    - $\mathbb{E}(\mathcal{X}\mathcal{Y}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xyf(x,y)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xyf(x)f(y)dxdy = \int_{-\infty}^{\infty} xf(x)dy \int_{-\infty}^{\infty} yf(y)dy = \mathbb{E}(\mathcal{X})\mathbb{E}(\mathcal{Y})$
- For vectors: If $\boldsymbol{y} = \boldsymbol{Ax}$: $\mathbb{E}(\boldsymbol{Ax}) = \boldsymbol{A}\mathbb{E}(\boldsymbol{x})$

  Proof:
  - $\mathbb{E}(\boldsymbol{Ax}) = \mathbb{E}[(\boldsymbol{A}^{(1)}\boldsymbol{x}, ..., \boldsymbol{A}^{(m)}\boldsymbol{x})^\intercal]$ where $\boldsymbol{A}^{(i)}$ is the i^{th} row of $\boldsymbol{A}$
  - $= (\mathbb{E}[\boldsymbol{A}^{(1)}\boldsymbol{x}], ..., \mathbb{E}[\boldsymbol{A}^{(m)}\boldsymbol{x}])^\intercal = (\boldsymbol{A}^{(1)}\mathbb{E}[\boldsymbol{x}], ..., \boldsymbol{A}^{(m)}\mathbb{E}[\boldsymbol{x}])^\intercal = \boldsymbol{A}\mathbb{E}(\boldsymbol{x})$
- $\mathbb{E}[\mathbb{E}(X|A)] = \mathbb{E}(X)$

  Proof:
  - $\mathbb{E}[\mathbb{E}(X|A)] = \int_{-\infty}^{\infty} f_A(a)\mathbb{E}(X|A)da = \int_{-\infty}^{\infty} f_A(a)\int_{-\infty}^{\infty} xf_{X|A}(x|a)dxda = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xf_{X,A}(x,a)dxda = \int_{-\infty}^{\infty} x\int_{-\infty}^{\infty} f_{X,A}(x,a)dadx = \int_{-\infty}^{\infty} xf_X(x)dx = \mathbb{E}(X)$
- *Cauchy Schwarz inequality*: $\mathbb{E}(\mathcal{X},\mathcal{Y})^2 \leq \mathbb{E}(\mathcal{X}^2)\mathbb{E}(\mathcal{Y}^2)$

Median — Real number M defined by $P(X < M) = P(X > M)$

Standard deviation — $\sqrt{\mathbb{V}(\mathcal{X})}$

Covariance —
- Univariate variance of a random variable: $\mathbb{V}(\mathcal{X}) = \mathbb{E}((\mathcal{X} - \mathbb{E}(\mathcal{X}))^2) = \mathbb{E}(\mathcal{X}^2) - \mathbb{E}(\mathcal{X})^2$ where $\mathbb{E}(\mathcal{X}^2)$ is the unnormalized correlation resp. inner product
- Univariate covariance of two random variables: $\text{Cov}(\mathcal{X},\mathcal{Y}) = \mathbb{E}((\mathcal{X} - \mathbb{E}(\mathcal{X}))(\mathcal{Y} - \mathbb{E}(\mathcal{Y}))) = \mathbb{E}(\mathcal{X}\mathcal{Y}) - \mu_{\mathcal{X}}\mu_{\mathcal{Y}}$ where $\mathbb{E}(\mathcal{X}\mathcal{Y})$ is the unnormalized correlation resp. inner product
- Proof (schematically for variance): $\mathbb{V}(\mathcal{X}) = \mathbb{E}((\mathcal{X} - \mathbb{E}(\mathcal{X}))^2) = \mathbb{E}[\mathcal{X}^2 - \mathcal{X}\mathbb{E}(\mathcal{X}) - \mathcal{X}\mathbb{E}(\mathcal{X}) + \mathbb{E}(\mathcal{X})^2] = \mathbb{E}[\mathcal{X}^2] - \mathbb{E}[\mathcal{X}]\mathbb{E}(\mathcal{X}) - \mathbb{E}[\mathcal{X}]\mathbb{E}(\mathcal{X}) + \mathbb{E}(\mathcal{X})^2 = \mathbb{E}(\mathcal{X}^2) - \mathbb{E}(\mathcal{X})^2$ where $\mathbb{E}(\mathcal{X}^2)$ is the second moment
- Multivariate covariance matrix of a vector:
  - $\Sigma = \text{Cov}(\boldsymbol{x}) = \mathbb{E}((\boldsymbol{x} - \mathbb{E}(\boldsymbol{x}))(\boldsymbol{x} - \mathbb{E}(\boldsymbol{x}))^\intercal) = \mathbb{E}(\boldsymbol{x}\boldsymbol{x}^\intercal) - \mathbb{E}(\boldsymbol{x})\mathbb{E}(\boldsymbol{x})^\intercal = \boldsymbol{R} - \boldsymbol{\mu}_X\boldsymbol{\mu}_X^\intercal = \begin{bmatrix} \mathbb{E}(x_1^2) & ... & \mathbb{E}(x_1x_m) \\ ... & ... & ... \\ \mathbb{E}(x_mx_1) & ... & \mathbb{E}(x_m^2) \end{bmatrix} - \begin{bmatrix} \mathbb{E}(x_1)^2 & ... & \mathbb{E}(x_1)\mathbb{E}(x_m) \\ ... & ... & ... \\ \mathbb{E}(x_m)\mathbb{E}(x_1) & ... & \mathbb{E}(x_m)^2 \end{bmatrix} = \begin{bmatrix} \mathbb{V}(x_1) & ... & \text{Cov}(x_1,x_m) \\ ... & ... & ... \\ \text{Cov}(x_m,x_1) & ... & \mathbb{V}(x_m) \end{bmatrix}$ where $\boldsymbol{R} = \mathbb{E}(\boldsymbol{x}\boldsymbol{x}^\intercal)$ is the unnormalized correlation matrix
  - Σ and $\boldsymbol{R}$ are symmetric and psd

Properties - variance:
- $\mathbb{V}(\alpha) = 0$
- $\mathbb{V}(\alpha\mathcal{X} + \beta) = \alpha^2\mathbb{V}(\mathcal{X})$
- $\mathbb{V}(\mathcal{X} + \mathcal{Y}) = \mathbb{V}(\mathcal{X}) + 2\text{Cov}(\mathcal{X},\mathcal{Y}) + \mathbb{V}(\mathcal{Y})$
- For uncorrelated (and independent) variables: $\mathbb{V}(\mathcal{X} + \mathcal{Y}) = \mathbb{V}(\mathcal{X}) + \mathbb{V}(\mathcal{Y})$
- For independent variables: $\mathbb{V}(\mathcal{X}\mathcal{Y}) = \mathbb{E}((\mathcal{X}\mathcal{Y})^2)\mathbb{E}(\mathcal{X}\mathcal{Y})^2$
- For vector $\boldsymbol{y} = \boldsymbol{Ax}$: $\mathbb{V}_y = \boldsymbol{A}\mathbb{V}_X\boldsymbol{A}^\intercal$
- For zero-mean variable: $\mathbb{V}(\mathcal{X}) = \mathbb{E}(\mathcal{X}^2) - \mathbb{E}(\mathcal{X})^2 = \mathbb{E}(\mathcal{X}^2)$ since $\mathbb{E}(\mathcal{X}) = 0$

Properties - covariance:
- $\text{Cov}(\mathcal{X},\mathcal{X}) = \mathbb{V}(\mathcal{X})$
- $\text{Cov}((\alpha\mathcal{X} + \beta\mathcal{Y}),\mathcal{Z}) = \alpha\text{Cov}(\mathcal{X},\mathcal{Z}) + \beta\text{Cov}(\mathcal{Y},\mathcal{Z})$
- If covariance of 2 random variables is 0 resp. $\mathbb{E}(\mathcal{X}\mathcal{Y}) = \mathbb{E}(\mathcal{X})\mathbb{E}(\mathcal{Y})$, they are uncorrelated, but not necessarily independent
- If unnormalized correlation of 2 random variables is 0 resp. $\mathbb{E}(\mathcal{X}\mathcal{Y}) = 0$, they are orthogonal, but not necessarily independent
- For vector $\boldsymbol{y} = \boldsymbol{Ax}$:
  - $\Sigma_y = \boldsymbol{A}\Sigma_X\boldsymbol{A}^\intercal$
  - $\boldsymbol{R}_y = \boldsymbol{A}\boldsymbol{R}_X\boldsymbol{A}^\intercal$
- For zero-mean variables: $\text{Cov}(\mathcal{X},\mathcal{Y}) = \mathbb{E}(\mathcal{X}\mathcal{Y}) - \mu_{\mathcal{X}}\mu_{\mathcal{Y}} = \mathbb{E}(\mathcal{X},\mathcal{Y})$ since $\mu_{\mathcal{X}} = \mu_{\mathcal{Y}} = 0$
- *Cauchy Schwarz inequality*: $\text{Cov}(\mathcal{X},\mathcal{Y})^2 \leq \mathbb{V}(\mathcal{X})\mathbb{V}(\mathcal{Y})$

Correlation — Normalized covariance
- Univariate correlation of a random variable: $\text{Cor}(\mathcal{X},\mathcal{Y}) = \frac{\text{Cov}(\mathcal{X},\mathcal{Y})}{\sqrt{\mathbb{V}(\mathcal{X})}\sqrt{\mathbb{V}(\mathcal{Y})}}$
- Multivariate correlation matrix of a vector:
  - $\boldsymbol{P} = \text{Cor}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} 1 & ... & \text{Cor}(\mathcal{X}_1,\mathcal{X}_m) \\ ... & ... & ... \\ \text{Cor}(\mathcal{X}_m,\mathcal{X}_1) & ... & 1 \end{bmatrix}$
  - $\boldsymbol{P}$ is symmetric and psd
  - Correlation is bounded between 0 and 1, given Cauchy Schwarz Inequality
  - If correlation of two random variables is 0, they are not necessarily independent

Probability Distributions

Normal distribution — $\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2)$

For univariate, PDF: $\frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-(x-\mu)^2}{2\sigma^2}) = \frac{1}{\sigma\sqrt{2\pi}} exp(-x^2\frac{1}{2\sigma^2} + 2x\frac{\mu}{2\sigma^2} - \frac{\mu^2}{2\sigma^2})$

For multivariate, PDF: $\frac{1}{2\pi^{n/2}}\frac{1}{|\Sigma|^{1/2}} exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\intercal\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu}))$ where the term in the exponent is a quadratic form

Convolution: $\int \mathcal{N}(a; Bc, D) \times \mathcal{N}(c; e, F)dc = \int \mathcal{N}(a; Be, D + BFB^\intercal)$

Standard normal distribution — Normal distribution, standardized via z-score $z = \frac{x-\mu}{\sigma}$, which results in $\mu = 0$ and $\sigma = 1$

Bernoulli distribution — trial with success (probability p) or failure (probability $1 - p$)
- $\mathcal{X} \sim \text{Bernoulli}(p)$
- PDF: $p(x)p^x(1-p)^x$
- Mean: $\mathbb{E}(x) = p$
- Variance: $\mathbb{V}(x) = p(1-p)$

Binomial distribution — n independent Bernoulli trials with k successes
- $\mathcal{X} \sim \text{Bin}(n,p)$
- PDF: $\binom{n}{k}p^k(1-p)^{n-k}$
- Mean: $\mathbb{E}(x) = np$
- Variance: $\mathbb{V}(x) = np(1-p)$

Poisson distribution —
- $\mathcal{X} \sim \text{Pois}(\lambda)$
- PDF: $e^{-\lambda}\frac{\lambda^x}{x!}$
- Mean: $\mathbb{E}(x) = \lambda$
- Variance: $\mathbb{V}(x) = \lambda$

Beta distribution —
- X takes values $\in [0,1]$
- Represents the probability of a Bernoulli process after observing $\alpha - 1$ successes and $\beta - 1$ failures
- $\mathcal{X} \sim \text{Beta}(\alpha,\beta)$ where $\alpha,\beta > 0$
- PDF: $\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}$ where $\Gamma(\alpha) = \int_0^\infty u^{\alpha-1}e^{-u}du$
- Mean: $\mathbb{E}(x) = \frac{\alpha}{\alpha+\beta}$
- Variance: $\mathbb{V}(x) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

Dirichlet distribution —
- X takes values $\in [0,1]$
- Multivariate extension of Beta distribution
- $Dir(\boldsymbol{x}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{k=1}^n u_k^{\alpha_k-1}$, where $B(\alpha)$ is the multivariate generalization of the Beta function: $B(\alpha) = \frac{\prod_{k=1}^n \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^n \alpha_k)}$

Uniform distribution —
- Assume x is uniformly distributed between $[a,b]$
- PDF: $f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$
- CDF: $F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & x > b \end{cases}$

Other Concepts

Law of large numbers — Sample mean of iid variables converges to population mean as $n \to \infty$
- *Weak law of large numbers*: $\lim_{n\to\infty} P\left(\left|m_X - \frac{1}{n}\sum_{k=1}^n X_k\right| < \varepsilon\right) = 1$
- *Strong law of large numbers*: $\lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^n X_k = m_X$ with probability 1

Union bound — $P(\bigcup_i A_i) \leq \sum_i P(A_i)$

Jensen's inequality — Relates expected value of a convex function of a random variable to the convex function of the expected value of that random variable

$\mathbb{E}(f(\mathcal{X})) \geq f(\mathbb{E}(\mathcal{X}))$

Markov's inequality — $p(x \geq t) \leq \frac{\mathbb{E}(x)}{t}$

Interesting only for $t \geq \mathbb{E}(x)$ because $p(x \geq t)$ must then be less than or equal to 1

Generalizations:
- $p(|x| \geq t) \leq \frac{\mathbb{E}(g(|x|))}{g(t)}$
- $p(|x| \geq t) \leq \frac{\mathbb{E}(|x|^n)}{t^n}$

Proof:
- $\mathbb{E}[g(|X|)] = \int_{-\infty}^{\infty} g(|X|)f_X(x)dx$
- $\mathbb{E}[g(|X|)] = \int_{|X|<t} g(|X|)f_X(x)dx + \int_{|X|\geq t} g(|X|)f_X(x)dx$
- $\mathbb{E}[g(|X|)] \geq \int_{|X|\geq t} g(|X|)f_X(x)dx$
- Since g is monotonically increasing, $g(|X|) \geq g(t)$ for $|X| \geq t$. Then: $\int_{|X|\geq t} g(|X|)f_X(x)dx \geq \int_{|X|\geq t} g(t)f_X(x)dx$
- $\int_{|X|\geq t} g(t)f_X(x)dx = g(t)\int_{|X|\geq t} f_X(x)dx = g(t)P(|X| \geq t)$
- Then: $\mathbb{E}[g(|X|)] \geq g(t)P(|X| \geq t)$

Hoeffding's Lemma — For random variable with $\mathbb{E}[x] = 0$, and $a \leq x \leq b$, and $s > 0$: $\mathbb{E}[\exp(sx)] = \exp(s^2(b-a)^2/8)$

Hoeffding's Inequality — For random variables x_i that fall in the interval $[a_i, b_i]$ with probability 1, and $S_n = \sum_{i=1}^n x_i$, and $t > 0$:

- $P(S_n - \mathbb{E}_X S_n \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$
- $P(S_n - \mathbb{E}_X S_n \leq -t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$

Proof:

- Consider the probability $P(S_n - \mathbb{E}[S_n] \geq t)$
- Using Markov's inequality:
  $P(S_n - \mathbb{E}[S_n] \geq t) = P\left(e^{s(S_n - \mathbb{E}[S_n])} \geq e^{st}\right) \leq \frac{\mathbb{E}[e^{s(S_n - \mathbb{E}[S_n])}]}{e^{st}}$
- Using independence of $X_1, \ldots, X_n$:
  $\mathbb{E}[e^{s(S_n - \mathbb{E}[S_n])}] = \prod_{i=1}^n \mathbb{E}[e^{s(X_i - \mathbb{E}[X_i])}]$
- For each term, we use the fact that $X_i \in [a_i, b_i]$, and apply the lemma inequality: $\prod_{i=1}^n \mathbb{E}[e^{s(X_i - \mathbb{E}[X_i])}] \leq \prod_{i=1}^n \exp\left(\frac{s^2(b_i - a_i)^2}{8}\right)$
- Plugging this back in: $e^{-st} \times \mathbb{E}[e^{s(S_n - \mathbb{E}[S_n])}] \leq$
  $e^{-st} \times \prod_{i=1}^n \exp\left(\frac{s^2(b_i - a_i)^2}{8}\right) = e^{-st} \times \exp\left(\frac{s^2}{8} \sum_{i=1}^n (b_i - a_i)^2\right) =$
  $P(S_n - \mathbb{E}[S_n] \geq t) \leq \exp\left(-st + \frac{s^2}{8} \sum_{i=1}^n (b_i - a_i)^2\right)$
- If we set $s = \frac{4t}{\sum_{i=1}^n (b_i - a_i)^2}$ to minimize the bound, we get:
  $P(S_n - \mathbb{E}[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$
- Similarly for $P(S_n - \mathbb{E}[S_n] \leq -t)$,

In the special case of normalized sums of iid variables, where $\tilde{S} = S_n/n$ and $t = n\epsilon$:

- Delta given by:
  - $P(\tilde{S}_n - \mathbb{E}_X \tilde{S}_n \geq \epsilon) \leq \exp\left(-\frac{2n\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2/n}\right)$
  - As $n \to \infty$, this $P(\tilde{S}_n - \mathbb{E}_X \tilde{S}_n \geq \epsilon) \to 0$
- Absolute deviation given by:
  - $P(|\tilde{S}_n - \mathbb{E}_X \tilde{S}_n| \geq \epsilon) = P(\tilde{S}_n - \mathbb{E}_X \tilde{S}_n \geq \epsilon \vee \tilde{S}_n - \mathbb{E}_X \tilde{S}_n \leq -\epsilon)$
  - By the union bound: $= P(\tilde{S}_n - \mathbb{E}_X \tilde{S}_n \geq \epsilon) + P(\tilde{S}_n - \mathbb{E}_X \tilde{S}_n \leq -\epsilon)$
  - By Hoeffding's inequality: $\leq 2\exp\left(-\frac{2n\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2/n}\right)$

Chebychev's inequality — $p(|x - \mu_x| \geq \alpha|\sigma_x|) \leq \frac{1}{\alpha^2}$ resp.

$p(|x - \mu_x| \geq \alpha) \leq \frac{|\sigma_x|}{\alpha^2}$

Interesting only for $\alpha > 1$

Implications:

- For n variables: $p(|S_n - \mu_x| \geq \epsilon) \leq \frac{\sigma_x^2}{n\epsilon^2}$ where $S_n = \frac{1}{n}\sum_{k=1}^n X_k$ is the sample mean

Proof:

- $P(|S_n - m_X| \geq \varepsilon) = P\left(\left|\frac{1}{n}\sum_{k=1}^n X_k - m_X\right| \geq \varepsilon\right)$
- Using Chebyshev's inequality, using the fact that S_n has mean m_X and variance $\mathrm{Var}(S_n)$: $P(|S_n - m_X| \geq \varepsilon) \leq \frac{\mathrm{Var}(S_n)}{\varepsilon^2}$
- $\mathrm{Var}(S_n) = \mathrm{Var}\left(\frac{1}{n}\sum_{k=1}^n X_k\right) = \frac{1}{n^2} n \mathrm{Var}(X_k) = \frac{\sigma_X^2}{n}$
- Then, we get: $P(|S_n - m_X| \geq \varepsilon) \leq \frac{\sigma_X^2}{n\varepsilon^2}$

Sufficient statistics —

- $Z = g(Y)$ is a sufficient statistic for estimating X if X can be estimated as well from Z as from Y, i.e. condensing Y to Z does not entail any loss of information about X
- Conditioned on Z, Y is independent of X: $p(Y|Z, X) = p(Y|Z)$
- For sufficient statistics, the MLE of X from Y is the same as the MLE of X from Z: $argmag_x p(Y|X)\rho(y) = argmag_x p(Z|X)\rho(y)$
- $p(X|Z) = p(X|Y)$

Hypothesis Testing

Terminology —

- *Hypothesis*:
  - H_0: Accepted null hypothesis, e.g. $p = p_0$, $p_1 - p_2 = p_{0,1} - p_{0,2} = 0$
  - H_A: Alternative hypothesis, e.g. $p \neq p_0$, $p_1 - p_2 \neq p_{0,1} - p_{0,2} \neq 0$
- Errors:
  - *True positive*: Chose H_0, and H_0 obtains
  - *False negative, type I error*: Chose H_A, but H_0 obtains
  - *True negative*: Chose H_A, and H_A obtains
  - *False positive, type II error*: Chose H_0, but H_A obtains
- *Significance level* α:
  - $\alpha \geq p(\text{type I error}) = p(\bar{x} \geq c \mid H_0)$ with equality for continuous variables
  - If α is small, the probability that we are erroneously rejecting H_0 is very small
  - Set by us, typically at 5%
  - If $\mathcal{X} \sim \mathcal{N}(\theta, 1)$ and $H_0 : \theta = 0$: $\alpha = p(\bar{x} \geq c \mid H_0) = p(\sqrt{n}\bar{x} \geq \sqrt{n}c \mid H_0) = p(z_n \geq \sqrt{n}c \mid H_0) = 1 - \Phi(\sqrt{n}c)$ where
    - Φ is the CDF of the normal distribution
    - $z_n \mid H_0 = \frac{\bar{x} - 0}{1/\sqrt{n}} = \sqrt{n}\bar{x}$
- *Critical value* z:
  - For two-sided: $z_{\alpha/2}$, $z_{1-\alpha/2}$
  - For one-sided upper tail: $z_{1-\alpha}$
  - For one-sided lower tail: z_α
  - Associated z-score with α
  - Corresponds to critical value c prior to z-score transformation
  - If $\mathcal{X} \sim \mathcal{N}(\theta, 1)$ and $H_0 : \theta = 0$:
    $\alpha = 1 - \Phi(\sqrt{n}c) \Rightarrow c = \frac{1}{\sqrt{n}}\Phi^{-1}(1 - \alpha)$ where Φ is the CDF of the normal distribution
- *P-value* p:
  - For two-sided: $p = P(|z| \geq z_n)$
  - For one-sided upper tail: $p = P(z \geq z_n)$
  - For one-sided lower tail: $p = P(z \leq z_n)$
  - Probability, given H_0 that we observe a value as or more extreme as the observed value z_n
  - Smallest significance level resp. largest confidence level, at which we can reject H_0 given the sample observed
  - If p-value is less than significance level resp. if observed value is more extreme than critical value, reject H_0, because the probability that we are erroneously doing so is very small
- *Confidence level*: $1 - \alpha$, probability, given H_0, that we retain H_0
- *Beta*: $\beta = p(\text{type II error})$
- *Power*:
  - $1 - \beta = p(\bar{x} \geq c \mid H_1)$
  - Probability, given H_A, that we reject H_0
  - If $\mathcal{X} \sim \mathcal{N}(\theta, 1)$ and $H_0 : \theta = 0$:
    $1 - \beta = p(\bar{x} \geq c \mid H_1) = p(\sqrt{n}(\bar{x} - 1) \geq \sqrt{n}(c - 1) \mid H_1) = p(z_n \geq \sqrt{n}(c - 1) \mid H_1) = p(z_n \geq \sqrt{n}(c - 1) \mid H_0) = 1 - \Phi(\sqrt{n}(c - 1))$ where
    - Φ is the CDF of the normal distribution
    - $z_n \mid H_1 = \frac{\bar{x} - 1}{1/\sqrt{n}} = \sqrt{n}(\bar{x} - 1)$
    - We can switch from $\mid H_1$ to $\mid H_2$ because the two distributions follow the same form, just shifted
- Test types:
  - *Two-sided*: $H_0 : p = p_0, H_A : p \neq p_0$
  - *One-sided upper tail*: $H_0 : p \leq p_0, H_A : p > p_0$
  - *One-sided lower tail*: $H_0 : p \geq p_0, H_A : p < p_0$
- Calculating *test statistic*:
  - $z_n \mid H_0 = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$

Multiple comparisons problem — Accumulation of false positive rate (α) for K tests, due to independence of tests:
$P(|\text{false rejections of } H_0| > 0) = 1 - P(|\text{false rejections of } H_0| = 0) = 1 - (1 - \alpha)^K$

Corrections for multiple comparisons problem — Bonferroni correction:
New significance level set to $\alpha* = \alpha/K$

Neyman Pearson test —

- Maximizes power while controlling type I errors
- Sets α such that $\alpha \geq p(\text{type I error})$
- Then minimizes $p(\text{type II error})$
- This is achieved by a likelihood-ratio test with threshold θ, such that α equals or is as close as possible to $p(\text{type I error})$:
  - If $\Lambda(x) = \frac{p(x|p_0)}{p(x|p_A)} > \theta$, we reject H_0
  - Then, we have
    $P(\Lambda(x) > \theta|H_0) = P(\frac{p(x|p_0)}{p(x|p_A)} > \theta|H_0) = P(\text{type I error}) = \alpha$
  - The smaller α, the larger θ

Bayesian Hypothesis Testing —

- If $\Lambda(x) = \frac{p(x|p_0)}{p(x|p_A)} > \theta$, we reject H_0
- $\theta = \frac{k(p_A, p_0)P(p_0)}{k(p_0, p_A)P(p_A)}$
- In this case, θ subsumes both the prior $p(x)$ and the costs $k(\hat{x}, x)$

Confidence Sets

Frequentist confidence sets:

- If we have $p(y|x)$ and set a threshold θ
- Go through each discrete x
- Construct a set J of values of y, such that $\sum_y p(y|X = x) \geq \theta$
- Return the smallest possible set of y values (starting with y value contributing the most) such that $p(y \in J|X = x) \geq \theta$

4 Information Theory

Description

Entropy —

- $H(x) = -\sum_x p(x)log(p(x)) = -\sum_{x,y} p(x, y)log(p(x))$ resp. $H(x) = -\int p(x)log(p(x))dx$
- Measure of randomness in a variable resp. quantifies uncertainty of a distribution

Properties:

- $H(x) \geq 0$
- $H(x)$ is maximized, when x is a uniform random variable
- For independent variables: $H(x, y) = H(x) + H(y)$

Conditional entropy —

- $H(x|y) = -\sum_{x,y} p(y)p(x|y)log(p(x|y)) = -\sum_{x,y} p(x, y)log(\frac{p(x,y)}{p(y)})$
- Measure of how much information of x is revealed by y

Properties:

- $0 \leq H(x|y) \leq H(x)$ with equality if when x is independent with y resp. if y completely determines x

---

Mutual information —

- $I(x; y) = H(x) - H(x|y) = -\sum_{x,y} p(x, y)log(\frac{p(x)p(y)}{p(x,y)})$
- Measure of how much information of x is left after y is revealed

Properties:

- $0 \leq I(x; y) \leq H(x)$ with equality if y completely determines x resp. if x is independent with y

---

KL divergence —

- $KL(p;q) = \sum_x p(x) log(\frac{p(x)}{q(x)})$
- Measures the extra information or inefficiency when approximating a true distribution over x, p, with a predicted one, q

Properties:
- $KL(p;q) \geq 0$

Cross entropy —
- $CE(p|q) = KL(p;q) + H(p) = -\sum_x p(x) log(q(x))$
- Measures the total uncertainty when using the predicted distribution q to represent the true distribution p, combining both the model's error and the intrinsic uncertainty of the true distribution

5 ML Set-Up

Formalization

Data —
- Features $\mathcal{X} \in \mathbb{R}^m$
- Response $\mathcal{Y}$
- Training data $\mathcal{D}$ vs. test data

Representation —
- Model resp. function f:
  - Models relationship between features and response based on noisy training data
  - $f_\theta : \mathbb{R}^m \to \mathcal{Y}$
  - θ are parameters characterizing f which are optimized during training
  - f is constrained by hyperparameters which are tuned during validation
- Model resp. function class $\mathcal{F}$: Determines model resp. function structure

Loss resp. objective function —
- $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, \hat{y}_i)$
- Distinguishes good from bad model resp. function
- Training vs. test error: Test error empirically estimates the true error

Optimization —
- During training, algorithm selects function with parameters θ^* that yields optimal results, based on loss resp. objective function
- Search for optimal parameters is governed by hyperparameters
- Models can be selected for:
  - Prediction performance: Model with best performance based on loss resp. objective function
  - Inference: Model which best explains the underlying process generating the data

Evaluation metric —
- Can coincide with loss resp. objective function, but not always the case

6 ML Paradigms

Frequentism

Description —
- Parametric approach
- θ as fixed, unknown quantity, X as random, and known quantity
- Makes point estimate
- Focuses on maximizing likelihood $p(X|\theta)$ to infer posterior $p(\theta|X)$
- Only requires differentiation methods
- High variance, but low bias

MLE estimator
- Maximizes log-likelihood:
  $\hat{\theta} = \arg\max_\theta(L) = \arg\max_\theta(p(y_1,...,y_n|\boldsymbol{x_i},\theta)) = \arg\max_\theta(\prod_{i=1}^n p(y_i|\boldsymbol{x_i},\theta)) = \arg\max_\theta(\sum_{i=1}^n log(p(y_i|\boldsymbol{x_i},\theta)))$
- In discrete case:
  - $\hat{\theta} = \arg\max_\theta(L) = \arg\max_\theta(\prod_{i=1}^n p(y_i|\boldsymbol{x_i},\theta)) = \arg\max_\theta \prod_{j=1}^k p_j^{N_j} = \arg\max_\theta \sum_{i=1}^n N_j log(p_j)$ where
    - $j = 1,...,k$ is the number of classes
    - N_j county how often the outcome class j appears in y
    - $p_j = p(y_i = j|\boldsymbol{x_i},\theta)$
  - We can further expand to
    $\hat{\theta} = \arg\max_\theta(L) = \arg\max_\theta \sum_{i=1}^n N_j log(p_j) = \arg\max_\theta \sum_{i=1}^n \frac{N_j}{n} log(p_j) = \arg\max_\theta \sum_{i=1}^n \frac{N_j}{n}(log(\frac{p_j}{N_j/n}) + log(N_j/n)) = \arg\max_\theta \sum_{i=1}^n \frac{N_j}{n} log(\frac{p_j}{N_j/n}) = \arg\min_\theta \sum_{i=1}^n \frac{N_j}{n} log(\frac{N_j/n}{p_j}) = \arg\min_\theta \sum_{i=1}^n \tilde{p}_j log(\frac{\tilde{p}_j}{p_j})$
  - This is the KL divergence between the empirical distribution and the model distribution
  - This can be solved using constrained optimization with strong duality subject to $\sum_j p_j = 1$
  - We then get $\theta_{MLE} = N_j/n$ which minimizes the KL divergence when $\tilde{p}_j = p_j$
- *Score*:
  - The score is the derivative of the log-likelihood:
    $\Lambda = \frac{\partial}{\partial\theta} log(p(y|\boldsymbol{x},\theta)) = \frac{\frac{\partial}{\partial\theta} p(y|\boldsymbol{x},\theta)}{p(y|\boldsymbol{x},\theta)}$
  - The expected score is given by:
    $\mathbb{E}(\Lambda) = \int p(y|\boldsymbol{x},\theta) \frac{\frac{\partial}{\partial\theta} p(y|\boldsymbol{x},\theta)}{p(y|\boldsymbol{x},\theta)} dx = \frac{\partial}{\partial\theta} \int p(y|\boldsymbol{x},\theta) dx = \frac{\partial}{\partial\theta} \times 1 = 0$
- Advantages:
  - *Consistent*: $\hat{\theta} \to \theta$ as $n \to \infty$
  - *Asymptotically normal*: $\frac{1}{\sqrt{n}}(\hat{\theta} - \theta)$ converges to $\mathcal{N}(0, \boldsymbol{J}^{-1}(\theta)\boldsymbol{I}(\theta)\boldsymbol{J}^{-1}(\theta))$ where $\boldsymbol{J} = -\mathbb{E}[\frac{\partial^2 log(p(y|\boldsymbol{x},\theta))}{\partial\theta\partial\theta^\intercal}]$ and where $\boldsymbol{I}$ is the Fisher information
  - *Asymptotically efficient*: $\hat{\theta}$ minimizes $\mathbb{E}[(\hat{\theta} - \theta)^2] \to \frac{1}{I_n(\theta)}$ as $n \to \infty$ where I is the Fisher information
    - Nonetheless, n necessarily the best estimator, especially for small samples in a multivariate context, where the *Stein estimator* outperforms
    - Cf. Rao-Cramer bound
  - *Equivariant*: If $\hat{\theta}$ is MLE of θ, then $g(\hat{\theta})$ is MLE of $g(\theta)$
- Proofs of advantages:
  - Asymptotically normal:
    - We start with the score and set it to 0 for optimization with regard to θ: $\Lambda = \frac{\partial}{\partial\theta} log(p(y|\boldsymbol{x},\theta)) = 0$
    - With a Taylor expansion, we can show that $(\hat{\theta} - \theta)\sqrt{n} = \frac{1}{\sqrt{n}}\Lambda[-\frac{1}{n}\frac{\partial^2}{\partial\theta\partial\theta^\intercal}\sum_{i=1}^n p(y_i|\boldsymbol{x_i},\theta))]^{-1}$ where Λ is the score
    - We set $\boldsymbol{J} = [-\frac{1}{n}\frac{\partial^2}{\partial\theta\partial\theta^\intercal}\sum_{i=1}^n p(y_i|\boldsymbol{x_i},\theta))]$
    - $\frac{1}{\sqrt{n}}\Lambda$ is a random vector with covariance matrix $\boldsymbol{I}$ and converges to the normal distribution $\sim \mathcal{N}(0,\boldsymbol{I})$
    - Then, $(\hat{\theta} - \theta)\sqrt{n} = \boldsymbol{J}^{-1}\frac{1}{\sqrt{n}}\Lambda \sim \boldsymbol{J}^{-1}\mathcal{N}(0,\boldsymbol{I})$
    - $\mathbb{V}(\boldsymbol{J}^{-1}\frac{1}{\sqrt{n}}\Lambda) = \mathbb{E}[\boldsymbol{J}^{-1}\boldsymbol{I}\boldsymbol{J}^{-1}]$
    - This equality is given because $\mathbb{V}(x) = \mathbb{E}[x - \mathbb{E}(x)] = \mathbb{E}[x]$ if $\mathbb{E}(x) = 0$, which is the case here, given that the expected score is 0
    - So we have shown that $(\hat{\theta} - \theta)\sqrt{n} = \boldsymbol{J}^{-1}\frac{1}{\sqrt{n}}\Lambda \sim \mathcal{N}(0, \boldsymbol{J}^{-1}\boldsymbol{I}\boldsymbol{J}^{-1})$
  - Equivariant:
    - Let $t = g(\theta)$ and $h = g^{-1}$
    - Then, $\theta = h(t) = h(g(\theta))$
    - For all t we have: $L(t) = \prod_i p(y_i|\boldsymbol{x_i},h(t))) = p(y_i|\boldsymbol{x_i},\theta) = L(\theta)$
    - Hence, for all t we can say: $L(t) = L(\theta)$ and $L(\hat{t}) = L(\hat{\theta})$
- Equivalent to minimizing KL divergence between observed distribution of the data $\hat{p}(x)$ and the family of distributions over the parameter space $q(x|\theta)$:
  - MLE estimator given by $\hat{\theta}_{MLE} = \arg\min_\theta \prod_{i=1}^n q(x_i|\theta) = \arg\min_\theta \sum_{i=1}^n \log q(x_i|\theta)$
  - KL estimator given by $\hat{\theta}_{KL} = \arg\min_\theta D_{KL}(\hat{p}(x)\|q(x|\theta))$ where $D_{KL}(\hat{p}(x)\|q(x|\theta)) = \mathbb{E}\left[\log \frac{\hat{p}(x)}{q(x|\theta)}\right]$
  - $D_{KL}(\hat{p}(x)\|q(x|\theta)) = \frac{1}{n}\sum_{i=1}^n \log \frac{\hat{p}(x_i)}{q(x_i|\theta)} = \frac{1}{n}\sum_{i=1}^n \log \hat{p}(x_i) - \frac{1}{n}\sum_{i=1}^n \log q(x_i|\theta)$
  - The term $\frac{1}{n}\sum_{i=1}^n \log \hat{p}(x_i)$ does not depend on θ, so minimizing D_{KL} is equivalent to maximizing: $\frac{1}{n}\sum_{i=1}^n \log q(x_i|\theta)$
  - This is equivalent to the log-likelihood maximization criterion for MLE
  - Therefore, $\hat{\theta}_{KL} = \hat{\theta}_{MLE}$ as $n \to \infty$ due to the law of large numbers
- In the case of classification for 2 classes, log loss is equivalent to binary cross entropy:
  - Given two classes $y \in \{0,1\}$:
    - Predicted probability for class 1: $p_1 = \sigma(z) = \frac{1}{1+e^{-z}}$
    - Predicted probability for class 0: $p_0 = 1 - p_1$
  - Binary cross entropy: $\text{BCE}(y,p_1) = -[y\log(p_1) + (1-y)\log(1-p_1)]$
    - When $y = 1$: $\text{BCE}(1,p_1) = -\log(p_1)$
    - When $y = 0$: $\text{BCE}(0,p_1) = -\log(1-p_1)$
  - Log loss: $\text{LL}(y,p) = -\log(p_y)$
    - When $y = 1$: $\text{LL}(y,p) = -\log(p_1)$
    - When $y = 0$: $\text{LL}(y,p) = -\log(p_0) = -\log(1-p_1)$

Probably Approximately Correct (PAC) estimator Framework provides guarantees about the generalization ability of a learning algorithm

Setting:
- *Hypothesis class* $\mathcal{H}$: Set of functions that can be expressed by the algorithm
- *Concept class* $\mathcal{C}$: Set of possible target functions that represent true mappings from input to output
- *Specific concept* c:
  - True function c that maps inputs to outputs
  - Estimated function $\hat{c}$
- *Learning algorithm* $\mathcal{A}$:
  - Receives samples $\mathcal{Z} = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ as inputs
  - $c(x_i) = y_i$ for all i
  - $\mathcal{A}$ outputs $\hat{c} \in \mathcal{C}$

PAC learning model:
- $\mathcal{A}$ can learn c from $\mathcal{C}$ if, given a sufficiently large sample, it outputs $\hat{c}$ that generalizes well with high probability resp.

  if given $\mathcal{Z}$ of size $n > \text{poly}\big(1/\epsilon, 1/\delta, \text{size}(c)\big)$, it outputs $\hat{c}$ such that: $P(\mathcal{R}(\hat{c}) \leq \epsilon) \geq 1 - \delta$ where:
  - ϵ: Error tolerance (how much $\hat{c}$ deviates from c), is between 0 and 0.5
  - δ: Confidence (how likely $\hat{c}$ generalizes well), is between 0 and 0.5
  - $\text{size}(c)$: Complexity of the concept

- A concept class $\mathcal{C}$ is *PAC-learnable* from a hypothesis class $\mathcal{H}$ if there is an algorithm $\mathcal{A}$ that can learn any concept in $\mathcal{C}$
- If algorithm $\mathcal{A}$ runs in time polynomial in $1/\epsilon$ and $1/\delta$, then $\mathcal{C}$ is *efficiently PAC-learnable*
- *Strong PAC learning*: Demand arbitrarily small error ϵ with high probability $1-\delta$
- *Weak PAC learning*: Demand that risk is bounded for large (not trivial) error ϵ, used frequently in ensemble learning

Bayesianism

Description —

- Parametric approach
- θ as random, unknown quantity, X as random, and known quantity
- Makes estimate in form of distribution
- Leverages prior and likelihood to infer posterior: $p(\theta|X,y) = \frac{p(\theta)p(y|X,\theta)}{p(y|X)} = \frac{p(\theta)p(y|X,\theta)}{\int p(\theta)p(y|X,\theta)d\theta} \propto p(\theta)p(y|X,\theta) = p(\theta,y|X)$
- Focuses on minimizing cost function
  $\mathbb{E}[k(\theta',\Theta)|X,y] = \int_\theta p(\theta|X,y)\times k(\theta',\theta)d\theta \propto \int_\theta p(\theta,y|X)\times k(\theta',\theta)d\theta$
  resp. $\sum p(\theta|X,y)\times k(\theta',\theta)$
- Requires integration methods for normalizing constant in denominator, which can be intractable, in which case mean / MAP estimator can provide an alternative
- Low variance, but high bias

MMSE estimator

- Minimizes mean squared error as cost function $k(\hat\theta,\theta) = |\hat\theta-\theta|^2$
- The resulting estimate is the mean of the posterior: $\hat\theta = \mathbb{E}[\theta|X,y]$
  Proof:
  - $\mathbb{E}[|\hat\theta-\theta|^2|y] = \int(\hat\theta-\theta)^2 p(\theta\mid y)\,d\theta = \hat\theta^2 - 2\hat\theta\int\theta p(\theta\mid y)\,d\theta + \int\theta^2 p(\theta\mid y)\,d\theta$
  - Taking the derivative with respect to $\hat\theta$: $\frac{\partial\mathbb{E}[|\hat\theta-\theta|^2|y]}{\partial\hat\theta} = 2\hat\theta - 2\int\theta p(\theta\mid y)\,d\theta$
  - Setting the derivative to zero: $\hat\theta = \int\theta p(\theta\mid y)\,d\theta = \mathbb{E}[\theta\mid y]$
- Returns single point estimate

Median estimator

- Minimizes mean absolute error as cost function $k(\hat\theta,\theta) = |\hat\theta-\theta|$
- The resulting estimate is the median of the posterior
  Proof:
  - Bayesian cost function given by: $\mathbb{E}[|\hat\theta-\theta||y] = \int|\hat\theta-\theta|p(\theta\mid y)\,d\theta$
  - The integral splits into two parts:
    $= \int_{-\infty}^{\hat\theta}(\hat\theta-\theta)p(\theta\mid y)\,d\theta + \int_{\hat\theta}^{\infty}(\theta-\hat\theta)p(\theta\mid y)\,d\theta$
  - Taking the derivative with respect to $\hat\theta$: $\frac{\partial\mathbb{E}[|\hat\theta-\theta||y]}{\partial\hat\theta}$ for each term separately
  - $\frac{\partial}{\partial\hat\theta}\int_{-\infty}^{\hat\theta}(\hat\theta-\theta)p(\theta\mid y)\,d\theta = \int_{-\infty}^{\hat\theta}p(\theta\mid y)\,d\theta - \hat\theta p(\hat\theta\mid y)$
  - $\frac{\partial}{\partial\hat\theta}\int_{\hat\theta}^{\infty}(\theta-\hat\theta)p(\theta\mid y)\,d\theta = -\int_{\hat\theta}^{\infty}p(\theta\mid y)\,d\theta - \hat\theta p(\hat\theta\mid y)$
  - Combining the two derivatives, we get:
    $\int_{-\infty}^{\hat\theta}p(\theta\mid y)\,d\theta - \int_{\hat\theta}^{\infty}p(\theta\mid y)\,d\theta$
  - Setting the derivative to zero: $\int_{-\infty}^{\hat\theta}p(\theta\mid y)\,d\theta = \int_{\hat\theta}^{\infty}p(\theta\mid y)\,d\theta$
  - Since the total probability is 1, this implies:
    $\int_{-\infty}^{\hat\theta}p(\theta\mid y)\,d\theta = 0.5$
- Returns single point estimate

MAP estimator

- Maximizes posterior:
  $\hat\theta = \arg\max_\theta(p(\theta|X)) \propto \arg\max_\theta p(\theta|X)p(X)$
- In discrete case:
  - MAP minimizes zero-one loss as cost function:
    $k(\hat\theta,\theta) = \begin{cases}1 & \hat\theta\neq\theta\\ 0 & \hat\theta=\theta\end{cases}$
    Proof:
    * Bayesian cost function given by:
      $R(\hat x) = \sum_x \kappa(\hat x,x)P(X=x\mid Y=y)$
    * If we substitute $\kappa(\hat x,x)$ we get: $R(\hat x) = \sum_{x\neq\hat x}P(X=x\mid Y=y)$ since when $\kappa(\hat x,x)=0$ (i.e., $x=\hat x$), the term contributes nothing
    * We need to minimize
      $\sum_{x\neq\hat x}P(X=x\mid Y=y) = 1-P(X=\hat x\mid Y=y)$
    * This is equivalent to maximizing $P(X=\hat x\mid Y=y)$, which is the MAP estimate
  - We can make $\theta_{MLE} = N_j/n$ more robust by setting a prior $p(\theta)\propto\prod_{i=1}^n p_j^v$ with parameter $0<v\leq 1$
  - $\hat\theta = \arg\max_\theta(p(\theta|y)) = \arg\max_\theta(p(y|\theta)p(\theta)) =$
    $\arg\max_\theta\prod_{j=1}^k p_j^{N_j}\prod_{j=1}^k p_j^v = \arg\max_\theta\prod_{j=1}^k p_j^{N_j+v} =$
    $\arg\max_\theta\sum_{j=1}^k(N_j+v)\log(p_j) =$
    $\arg\max_\theta\sum_{j=1}^k\frac{N_j+v}{n+kv}\log(\frac{p_j}{(N_j+v)/(n+kv)}) =$
    $\arg\min_\theta\sum_{j=1}^k\frac{N_j+v}{n+kv}\log(\frac{(N_j+v)/(n+kv)}{p_j}) =$
    $\arg\min_\theta\sum_{j=1}^k\tilde p_j\log(\frac{\tilde p_j}{p_j})$
  - This is the KL divergence
  - This can be solved using constrained optimization with strong duality subject to $\sum_j p_j = 1$
  - We then get $\theta_{MAP} = (N_j+v)/(n+kv)$ which minimizes the KL divergence when $\tilde p_j = p_j$
- The resulting estimate is the mode of the posterior
- Returns single point estimate

Statistical Learning

Description —

- We want to minimize expected risk $\mathcal{R}(f) = \mathbb{E}_{X,Y}[1f(X)\neq Y]$, but this is difficult because
  - We don't have access to the joint distribution of X,Y
  - We cannot find f, without any assumptions on its structure
  - It's unclear how to minimize the expected value
- Therefore, we make following choices:
  - We collect sample Z
  - We restrict space of possible choices of f to a set $\mathcal{H}$
  - We use a loss function to approximate the expected value
- With these choices, we approximate the expected risk via the empirical risk $\hat{\mathcal{R}}(f) = \hat L(Z,f) = \frac{1}{n}\sum_i L(y_i,f(x_i))$

7 Model Optimization

Gradient Descent

Numeric optimization procedure

Gradient descent —

- Uses entire training set to evaluate whether new parameter is more optimal than previous one
- Slow and less likely to escape local minima due to randomness, but accurate
- Algorithm:
  1. Set $\eta>0$
  2. Randomly initialize $\beta_{(t=0)}$
  3. $\beta_{(t+1)}\leftarrow\beta_{(t)}-\eta\nabla_\beta LO|_{\beta=\beta_{(t)}}$
  4. $t\leftarrow t+1$
  5. Repeat 3 and 4 until $\nabla_\beta LO = 0$

Stochastic gradient descent —

- Uses only one training sample or mini-batch to evaluate whether new parameter is more optimal than previous one
- Fast and more likely to escape local minima due to randomness, but represents an approximation
- Algorithm:
  1. Set $\eta>0$
  2. Randomly initialize $\beta_{(t=0)}$
  3. Shuffle training data and initialize $i\leftarrow 1$
  4. $\beta_{(t+1)}\leftarrow\beta_{(t)}-\eta\nabla_\beta LO$ for observation i $|_{\beta=\beta_{(t)}}$
  5. $t\leftarrow t+1$
  6. $i\leftarrow i+1$
  7. Repeat 4 to 6 until $i=n+1$
  8. Repeat 2 to 6 until $\nabla_\beta LO = 0$
- Justification for SGD is given by *Robbins-Monro algorithm* which iteratively find the root (or zero) of an unknown function when only noisy observations of the function are available:
  - Algorithm:
    1. Choose learning rates $\eta_1,\eta_2,\ldots$, typically decreasing over time
    2. Randomly initialize $\beta_{(t=0)}$
    3. $\beta_{(t+1)}\leftarrow\beta_{(t)}-\eta\nabla_\beta LO|_{\beta=\beta_{(t)}}$ where LO is noisy
  - For convergence:
    * $\sum_{t=1}^\infty\eta_t=\infty$ to ensure sufficient exploration
    * $\sum_{t=1}^\infty\eta_t^2\leq\infty$ to avoid overly large updates
    * Then, $lim_{t\to\infty}p(|y_t-y|>\epsilon)=0$ for any $\epsilon>0$

Hyperparameters —

- Learning rate η: Determines step size, if too small algorithm is slow to converge, if too large algorithm may diverge
- Batch size b: Number of samples from training set used to evaluate optimality of β at each step
- Epoch: Number of times model works through entire training set. Every epoch, β is updated n/b times

Modifications —

- Data should be standardized resp. scaled, otherwise the gradient of the largest predictor dominates the gradient of the loss function, leading to uneven updating of β and slow convergence
- A momentum term can be added to the updating function to ensure smooth updating of β:
  $\beta_{(t+1)}\leftarrow\beta_{(t)}-\eta\nabla_\beta LO|_{\beta=\beta_{(t)}}+\alpha(\beta_{(t)}-\beta_{(t-1)})$
- For stochastic gradient descent, a smoothing step can be added because stochastic gradient descent hovers around desired solution: $\hat\beta_{(t+1)}\leftarrow\frac{1}{L+1}\sum_{j=t-L}^t\beta_{(t)}$

8 Model Evaluation

Estimator Evaluation Criteria

Criteria —

- Consistency: $\hat\theta\to\theta$ as $n\to\infty$
- Bias: $\mathbb{E}(\hat\theta)-\theta$
  - Unbiased: $\mathbb{E}(\hat\theta)=\theta$
  - Asymptotically unbiased: $\mathbb{E}[(\hat\theta-\theta)^2]=0$ as $n\to\infty$
  - Asymptotically efficient: $\mathbb{E}[(\hat\theta-\theta)^2]=I$ as $n\to\infty$ where I is Fisher information

Bias Variance Tradeoff

- Mean squared error $\mathbb{E}[(\hat f(X)-y)^2]$ can be decomposed into:
  $(\mathbb{E}[\hat f(X)]-f(X))^2 + \mathbb{V}(\hat f(X)) + \mathbb{E}[\epsilon^2] =$
  $\text{bias}^2 + \text{variance} + \text{irreducible error}$
  Proof:

- $y = f(X) + \epsilon$
- $\mathbb{E}[(\hat{f}(X) - y)^2] = \mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)] + \mathbb{E}[\hat{f}(X)] - f(X) + \epsilon)^2] = \mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)])^2] + \mathbb{E}[(\mathbb{E}[\hat{f}(X)] - f(X))^2] + \mathbb{E}[\epsilon^2] - 2\mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)])(\mathbb{E}[\hat{f}(X)] - f(X) + \epsilon)]$
- Third term $\mathbb{E}[\epsilon^2]$ is the variance of y: $= \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2 = \mathbb{V}(y) = \sigma^2$
- Fourth term $2\mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)])(\mathbb{E}[\hat{f}(X)] - f(X) + \epsilon)]$ equals 0:
  - $2\mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)])(\mathbb{E}[\hat{f}(X)] - f(X) + \epsilon)] = 2(\mathbb{E}[\hat{f}(X)] - f(X) + \epsilon)\mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)])]$ because $(\mathbb{E}[\hat{f}(X)] - f(X) + \epsilon)$ is deterministic
  - In last equation, second term equals 0, so whole equation is 0
- Then, we are left with: variance + bias^2 + irreducible error

- *Bias*: Error generated by the fact that we approximate a complex relationship via a simpler model (small function class) with a certain presupposed parametric form
- *Variance*: Error generated by the fact that we estimate the model parameters with a noisy training sample (small sample), rather than the population
- *Irreducible error*: Error generated by measurement error and the fact that we estimate y as a function of X, when it is a function of many other factors
- *Bias variance tradeoff*: Bias and variance cannot be reduced simultaneously
  - High variance associated with overfitting: Model corresponds too closely to particular training set resp. performs poorly on unseen data, but well on training set
  - High bias associated with underfitting: Model fails to capture underlying relationships resp. performs poorly on both training set and unseen data

Approximating Generalisation Loss via Empirical Loss

Via resampling methods —
Cross-validation:

- Partition data $\mathcal{Z}$ into K equally sized disjoint subsets: $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2 \cup ... \cup \mathcal{Z}_K$
- Produce estimator $\hat{f}^{-v}$ from $\mathcal{Z} \backslash \mathcal{Z}_v$ for $v \leq K$
- Empirical loss given by: $\hat{\mathcal{R}}^{cv} = \frac{1}{n} \sum_{i \leq n} LO(y_i - \hat{f}^{-k(i)}(x_i))$ where $k(i)$ maps i to partition $\mathcal{Z}_{k(i)}$ where (x_i, y_i) belongs

Bootstrapping:

- Draw B samples with replacement of size n from data $\mathcal{Z}$: $\mathcal{Z}^{*b}$
- Compute estimate $S(\mathcal{Z}^{*b})$ for each bootstrap sample
- For each estimate $S(\mathcal{Z}^{*b})$, we can give a mean and variance:
  - $\overline{S} = \frac{1}{B} \sum_b S(\mathcal{Z}^{*b})$
  - $\sigma^2(S) = \frac{1}{B-1} \sum_b (S(\mathcal{Z}^{*b}) - \overline{S})^2$
- Out-of-bag loss given by: $\hat{\mathcal{R}}^{bs} = \frac{1}{n} \sum_{i=1}^n \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} LO(y_i - \hat{f}^{*b}(x_i))$ where C^{-i} contains all bootstrap indices b so that $\mathcal{Z}^{*b}$ does not contain (x_i, y_i)
- Empirical loss given by: $\hat{\mathcal{R}}(\mathcal{A}) = \frac{1}{B} \sum_{b=1}^B \frac{1}{n} \sum_{i=1}^n LO(y_i - \hat{f}^{*b}(x_i))$
- Empirical loss of bootstrap uses training data to estimate $\hat{\mathcal{R}}$, i.e. it is generally too optimistic. We can correct this by combining the empirical and out-of-bag loss:
  - Probability that (x_i, y_i) is not in sample $\mathcal{Z}^{*b}$ of size n is given by $(1 - \frac{1}{n})^n = \frac{1}{e}$ as $n \to \infty \approx \frac{1}{3}$
  - Probability that (x_i, y_i) is in sample $\mathcal{Z}^{*b}$ of size n is given by $1 - \frac{1}{e}$ as $n \to \infty \approx \frac{2}{3}$
  - We then define: $\hat{\mathcal{R}}^{(0.632)} = 0.368\hat{\mathcal{R}}(\mathcal{A}) + 0.632\hat{\mathcal{R}}^{bs}$

9 Estimating Common Distributions

Gaussian

Frequentism (MLE) —

- Likelihood (excl. constants): $L = (\frac{1}{\sigma})^n \prod_{i=1}^n exp(-\frac{1}{2\sigma^2}(x^{(i)} - \mu)^2) = \frac{1}{\sigma^n} exp(-\frac{1}{2\sigma^2} \sum_{i=1}^n (x^{(i)} - \mu)^2) = \frac{1}{\sigma^n} exp(-\frac{1}{2\sigma^2} \sum_{i=1}^n (x^{(i)} - \overline{x} + \overline{x} - \mu)^2) = \frac{1}{\sigma^n} exp(-\frac{\sum_{i=1}^n (x^{(i)} - \overline{x})^2}{2\sigma^2}) exp(-\frac{n(\overline{x} - \mu)^2}{2\sigma^2}) = \frac{1}{\sigma^n} exp(-\frac{nS^2}{2\sigma^2}) exp(-\frac{n(\overline{x} - \mu)^2}{2\sigma^2})$ where $S = \frac{1}{n}(x^{(i)} - \overline{x})^2$ is the covariance matrix
- Log-likelihood: $LL = -nlog(\sigma) - \sum_{i=1}^n (\frac{1}{2\sigma^2}(x^{(i)} - \mu)^2) = -nlog(\sigma) - \frac{nS^2}{2\sigma^2} - \frac{n(\overline{x} - \mu)^2}{2\sigma^2}$
- μ_{MLE} is sample mean: $\frac{1}{n} \sum_{i=1}^n x^{(i)}$:
  - Derivative of log-likelihood wrt μ: $\nabla_\mu LL = \nabla_\mu (-\sum_{i=1}^n (\frac{x^{(i)2} - 2x^{(i)}\mu + \mu^2}{2\sigma^2})) = \nabla_\mu (-\sum_{i=1}^n (-\frac{x^{(i)}\mu}{\sigma^2} + \frac{\mu^2}{2\sigma^2})) = -\sum_{i=1}^n (-\frac{x^{(i)}}{\sigma^2} + \frac{2\mu}{2\sigma^2}) = \sum_{i=1}^n (\frac{x^{(i)} - \mu}{\sigma^2}) = \sum_{i=1}^n x^{(i)} - n\mu = 0$
- μ_{MLE} is an unbiased estimator:
  - $\mathbb{E}[\mu_{MLE}] = \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^n x_i\right] = \frac{1}{n} \sum_{i=1}^n \mathbb{E}[x_i] = \mu$
- ${\sigma^2}_{MLE}$ is sample variance: $\frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu)^2$:
  - Derivative of log-likelihood wrt σ: $\nabla_\sigma LL = -n\nabla_\sigma log(\sigma) - \nabla_\sigma (\sum_{i=1}^n (\frac{(x^{(i)} - \mu)^2}{2\sigma^2})) = \frac{-n}{\sigma} - \nabla_\sigma (\sum_{i=1}^n \frac{1}{2}\sigma^{-2}(x^{(i)} - \mu)^2) = \frac{-n}{\sigma} - (\sum_{i=1}^n -1\sigma^{-3}(x^{(i)} - \mu)^2) = -n + \sum_{i=1}^n (\frac{(x^{(i)} - \mu)^2}{\sigma^2}) = 0$
- ${\sigma^2}_{MLE}$ is a biased estimator:
  - $\mathbb{E}[\Sigma_{MLE}] = \frac{1}{n} \sum_{i=1}^n \mathbb{E}\left[(x_i - \mu_{MLE})(x_i - \mu_{MLE})^\top\right]$
  - $= \frac{1}{n} \sum_{i=1}^n \mathbb{E}\left[x_i x_i^\top\right] - \frac{1}{n} \sum_{i=1}^n \mathbb{E}\left[x_i \mu_{MLE}^\top\right] - \frac{1}{n} \sum_{i=1}^n \mathbb{E}\left[\mu_{MLE} x_i^\top\right] + \mathbb{E}\left[\mu_{MLE} \mu_{MLE}^\top\right]$
  - $= \frac{1}{n} \sum_{i=1}^n \mathbb{E}\left[x_i x_i^\top\right] - \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \mathbb{E}\left[x_i x_j^\top\right] - \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \mathbb{E}\left[x_i x_j^\top\right] + \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \mathbb{E}\left[x_i x_j^\top\right]$
  - $= \frac{1}{n} \sum_{i=1}^n \mathbb{E}\left[x_i x_i^\top\right] - \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \mathbb{E}\left[x_i x_j^\top\right]$
  - $= \frac{1}{n} n(\Sigma + \mu\mu^\top) - \frac{1}{n^2}\left(n^2 \mu\mu^\top + n\Sigma\right)$

  Proof:
  - $\mathbb{E}\left[x_i x_j^\top\right] = \delta_{ij}\Sigma + \mu\mu^\top$ where $\delta = 1$ if $i = j$

    Proof:
    - $\Sigma = \mathbb{E}[(x_i - \mu)(x_i - \mu)^\top] = \mathbb{E}\left[x_i x_i^\top - 2x_i\mu^\top + \mu\mu^\top\right] = \mathbb{E}\left[x_i x_i^\top\right] - \mu\mu^\top$
    - For $i \neq j$, covariance is 0: $\mathbb{E}\left[x_i x_j^\top\right] = \mathbb{E}[x_i]\mathbb{E}[x_j] = \mu\mu^\top$
  - $... + n\Sigma$ since in n cases $\delta = 1$
  - $= \Sigma - \frac{1}{n}\Sigma$
  - $\mathbb{E}[\Sigma_{MLE}] = \Sigma - \frac{1}{n}\Sigma$
- ${\sigma^2}_{MLE}$ estimate satisfies: $\mathbb{E}[\sigma^2_{MLE}] = \mathbb{E}[xx^\top] - \mu\mu^\top = \frac{n-1}{n}\Sigma$

  Proof:
  - $\mathbb{E}[\mu\mu^\top] = \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^n x^{(i)} \frac{1}{n} \sum_{j=1}^n x^{(j)\top}\right] = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \mathbb{E}[x^{(i)} x^{(j)\top}]$
  - For $i = j$ in n cases, $\mathbb{E}[x^{(i)} x^{(j)\top}] = \mathbb{E}[xx^\top]$
  - For $i \neq j$ in $n(n-1)$ combinations, $\mathbb{E}[x^{(i)} x^{(j)\top}] = \mathbb{E}[x]\mathbb{E}[x]^\top = \mu\mu^\top$
  - Then, we have: $\mathbb{E}[\mu\mu^\top] = \frac{1}{n^2}(n\mathbb{E}[xx^\top] + n(n-1)\mu\mu^\top) = \frac{1}{n}\mathbb{E}[xx^\top] + \frac{n-1}{n}\mu\mu^\top$
  - Substituting into $\mathbb{E}[xx^\top] - \mu\mu^\top$, we get: $\mathbb{E}[xx^\top] - \mu\mu^\top = \mathbb{E}[xx^\top] - \frac{1}{n}\mathbb{E}[xx^\top] - \frac{n-1}{n}\mu\mu^\top = \frac{n-1}{n}(\mathbb{E}[xx^\top] - \mu\mu^\top) = \frac{n-1}{n}\Sigma$

Bayesianism — Univariate:

- Assume σ^2 is known and $\mu \sim \mathcal{N}(\mu_0, \sigma_0^2)$ is the outcome of a random variable
- $p(\mu|x, \mu_0, \sigma_0^2) \propto p(x|\mu, \sigma^2)p(\mu|\mu_0, \sigma_0^2)$
- The likelihood is $p(x|\mu, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$
- The prior is $p(\mu|\mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(\mu - \mu_0)^2}{2\sigma_0^2}\right)$
- The, the posterior is given by: $p(\mu|x, \mu_0, \sigma_0^2) \propto \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^n (x_i - \mu)^2 - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right)$
- Expanding the likelihood term: $\sum_{i=1}^n (x_i - \mu)^2 = \sum_{i=1}^n (x_i^2 - 2\mu x_i + \mu^2) = \sum_{i=1}^n x_i^2 - 2\mu \sum_{i=1}^n x_i + n\mu^2$
- Expanding the prior term: $(\mu - \mu_0)^2 = (\mu^2 - 2\mu\mu_0 + \mu_0^2)$
- This yields: $p(\mu|x, \mu_0, \sigma_0^2) \propto \exp\left(-\frac{1}{2}\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 + \left(\frac{\sum_{i=1}^n x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right)\mu + (\text{constant terms})\right)$ where the constant terms include terms that do not depend on μ, such as $\sum_{i=1}^n x_i^2$ and μ_0^2
- Based on the parametric form of the Gaussian distribution, this yields $\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 = \frac{1}{2\sigma_n{}^2}\mu^2$ and $\left(\frac{\sum_{i=1}^n x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right)\mu = \frac{\mu_n}{\sigma_n{}^2}\mu$
- Solving for σ_n and μ_n:
  - $\mu_n = \frac{\frac{\sum_{i=1}^n x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{\frac{n\bar{x}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{\frac{n\bar{x}\sigma_0^2 + \mu_0\sigma^2}{\sigma^2\sigma_0^2}}{\frac{n\sigma_0^2 + \sigma^2}{\sigma^2\sigma_0^2}} = \frac{n\bar{x}\sigma_0^2 + \mu_0\sigma^2}{n\sigma_0^2 + \sigma^2}$
  - $\sigma_n = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{1}{\frac{n\sigma_0^2 + \sigma^2}{\sigma^2\sigma_0^2}} = \frac{\sigma^2\sigma_0^2}{n\sigma_0^2 + \sigma^2}$
- Thus, the posterior is $p(\mu|x, \mu_0, \sigma_0^2) \sim \mathcal{N}(\mu_n, \sigma_n^2)$

Multivariate:

- Assume Σ is known and $\mu \sim \mathcal{N}(\mu_0, \Sigma_0)$ is the outcome of a random variable
- $p(\mu|X, \mu_0, \Sigma_0) \propto p(X|\mu, \Sigma)p(\mu|\mu_0, \Sigma_0)$
- $p(X|\mu, \Sigma) = \frac{1}{2\pi^{mn/2}} \frac{1}{|\Sigma|^{n/2}} exp(\frac{1}{2} \sum_{i=1}^n (x^{(i)} - \mu)^\top \Sigma^{-1} (x^{(i)} - \mu))$
- $p(\mu|\mu_0, \Sigma_0) = \frac{1}{2\pi^{m/2}} \frac{1}{|\Sigma_0|^{n/2}} exp(\frac{1}{2} \sum_{i=1}^n (\mu - \mu_0)^\top {\Sigma_0}^{-1} (\mu - \mu_0))$
- $p(\mu|X, \mu_0, \Sigma_0) \propto exp(-\frac{1}{2}(\mu^\top {\Sigma_0}^{-1}\mu + n\mu^\top \Sigma^{-1}\mu - 2\mu_0^\top {\Sigma_0}^{-1}\mu - 2n\bar{x}^\top \Sigma^{-1}\mu))$ after combining exponents of the prior and likelihood, expanding, absorbing terms unrelated to μ into a constant, and replacing $\sum_{i=1}^n x^{(i)\top}$ by $n\bar{x}^\top$
- We now apply a symmetric matrix property

$x^\top Ax + 2x^\top b = (x + A^{-1}b)^\top A(x + A^{-1}b) - b^\top A^{-1}b$, with $\mu = x$, $-(\Sigma_0{}^{-1} + n\Sigma^{-1})^{-1} = A^{-1}$ and $(\Sigma^{-1} n\bar{x} + \Sigma_0{}^{-1}\mu_0) = b$
- Through this, we get $p(\mu|X, \mu_0, \Sigma_0) \propto \exp(\frac{1}{2}(\mu(\Sigma_0{}^{-1} + n\Sigma^{-1})^{-1}(\Sigma^{-1} n\bar{x} + \Sigma_0{}^{-1}\mu_0))^\top(\Sigma_0{}^{-1} + n\Sigma^{-1})(\mu - (\Sigma_0{}^{-1} + n\Sigma^{-1})^{-1}(\Sigma^{-1} n\bar{x} + \Sigma_0{}^{-1}\mu_0))) = exp(\frac{1}{2}(\mu - \mu_n)^\top \Sigma_n^{-1}(\mu - \mu_n))$
- Thus, $p(\mu|X, \mu_0, \Sigma_0) \sim \mathcal{N}(\mu_n, \Sigma_n)$ with
  - $\mu_n = (\Sigma_0{}^{-1} + n\Sigma^{-1})^{-1}(\Sigma^{-1} n\bar{x} + \Sigma_0{}^{-1}\mu_0) =$ (if Σ equals 1) $\frac{n\bar{x}\Sigma_0 + \mu_0}{n\Sigma_0 + 1}$
  - $\Sigma_n = (\Sigma_0{}^{-1} + n\Sigma^{-1})^{-1} =$ (if Σ equals 1) $\frac{\Sigma_0}{n\Sigma_0 + 1}$

Implications:
- The Gaussian distribution is a conjugate prior for the mean of a Gaussian distribution, if Σ is known
- For Bayesian parameter μ_n:
  - μ_n is a compromise between MLE and prior, approximating prior for small n and MLE for large n
  - If prior variance is small (i.e. if we are certain of our prior), prior mean weighs more strongly
- For Bayesian parameter Σ_n:
  - Σ_n approximates prior for small n and MLE for large n
  - If prior variance is small (i.e. if we are certain of our prior), posterior variance is also small

Bayesianism: Absolute Error —
- Conditional median of y given $X = x$ is the Bayesian estimation of y from $X = x$, when we take the absolute error $|\hat{y} - y|$ as the cost function: $\mathbb{E}[|\hat{y} - y||X = x]$

Binomial

Frequentism — MLE:
- Likelihood $P(\delta|p)$ has a binomial distribution: $P(\delta|p) \sim p^{\alpha_1}(1-p)^{\alpha_2}$ where α_1 = number of successes, α_2 = number of failures
- $\hat{p}_{\text{MLE}} = \arg\max(p^{\alpha_1}(1-p)^{\alpha_2}) = \alpha_1/(\alpha_1 + \alpha_2)$ after logarithmizing and finding local minimum

Bayesianism —
- Likelihood $P(\delta|p)$ has a binomial distribution: $P(\delta|p) \sim p^{\alpha_1}(1-p)^{\alpha_2}$ where α_1 = number of successes, α_2 = number of failures
- Prior $P(p)$ has a beta distribution: $P(p) \sim \frac{\Gamma(\beta_1+\beta_2)}{\Gamma(\beta_1)\Gamma(\beta_2)} p^{\beta_1-1}(1-p)^{\beta_2-1}$
- Posterior is:
- $P(p|\delta) = \frac{P(\delta|p)P(p)}{P(\delta)} \propto p^{\alpha_1}(1-p)^{\alpha_2} p^{\beta_1-1}(1-p)^{\beta_2-1} = p^{\alpha_1+\beta_1-1}(1-p)^{\alpha_2+\beta_2-1}$
- For a binomial likelihood, the conjugate prior is the beta distribution, which guarantees that the posterior is also a beta distribution:
- $P(p|\delta) \sim \text{Beta}(\alpha_1 + \beta_1, \alpha_2 + \beta_2)$
- $P(p|\delta) = \frac{\Gamma(\alpha_1+\beta_1+\alpha_2+\beta_2)}{\Gamma(\alpha_1+\beta_1)\Gamma(\alpha_2+\beta_2)} p^{\alpha_1+\beta_1-1}(1-p)^{\alpha_2+\beta_2-1}$

MAP:
- Posterior $P(p|\delta)$ has a beta distribution:
  $P(p|\delta) = \frac{\Gamma(\alpha_1+\beta_1+\alpha_2+\beta_2)}{\Gamma(\alpha_1+\beta_1)\Gamma(\alpha_2+\beta_2)} p^{\alpha_1+\beta_1-1}(1-p)^{\alpha_2+\beta_2-1}$
- $\hat{p}_{\text{MAP}} = \frac{\alpha_1+\beta_1-1}{\alpha_1+\beta_1+\alpha_2+\beta_2-2}$ after logarithmizing and finding local minimum
- Thus, if our prior belief $P(p) \sim \text{Beta}(\beta_1, \beta_2)$ is strong, β_1 and β_2 will be large and the prior dominates the posterior
- If we gather more data, α_1 and α_2 will be large and the posterior will begin to dominate the prior
- If we have no strong prior belief, we can select $\beta_1 = \beta_2 = 1$, and thus $p_{\text{MLE}} = p_{\text{MAP}}$

Poisson

Frequentism — MLE:
- Likelihood $P(x|\lambda)$ has a Poisson distribution:
  $P(x|\lambda) = \prod_{i=1}^n \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} = \frac{\lambda^{\sum_{i=1}^n x_i} e^{-\lambda n}}{\prod_{i=1}^n x_i!}$
- Log-likelihood is given by
  $\log(P(x|\lambda)) = \sum_{i=1}^n x_i \log(\lambda) - \lambda n - \sum_{i=1}^n \log(x_i!)$
- Derivative of log-likelihood is given by $\frac{\partial \log(P(x|\lambda))}{\partial \lambda} = \frac{\sum_{i=1}^n x_i}{\lambda} - n = 0$
- $\Rightarrow \hat{\lambda}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^n x_i$
- Log likelihood is concave, so $\hat{\lambda}_{\text{MLE}}$ is the maximizer

Bayesianism —
- Likelihood $P(x|\lambda)$ has a Poisson distribution:
  $P(x|\lambda) = \prod_{i=1}^n \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} = \frac{\lambda^{\sum_{i=1}^n x_i} e^{-\lambda n}}{\prod_{i=1}^n x_i!}$
- Prior $P(\lambda)$ has a Gamma distribution: $P(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}$
- Posterior is given by
  $P(\lambda|x) \propto P(x|\lambda)P(\lambda) = \lambda^{\sum_{i=1}^n x_i} e^{-\lambda n} \times \lambda^{\alpha-1} e^{-\beta\lambda} = \lambda^{\sum_{i=1}^n x_i + \alpha - 1} \times e^{-\lambda n} \times e^{-\beta\lambda} = \lambda^{\sum_{i=1}^n x_i + \alpha - 1} \times e^{-(n+\beta)\lambda}$
- For a Poisson likelihood, the conjugate prior is the Gamma distribution, which guarantees that the posterior is also a Gamma distribution: $P(\lambda|x) \sim \text{Beta}(\alpha', \beta')$
- where $\alpha' = \alpha + \sum_{i=1}^n x_i$ and $\beta' = \beta + n$

MAP:
- Log-posterior is given by
  $\log(P(\lambda|x)) = (\alpha + \sum_{i=1}^n x_i - 1)\log(\lambda) - (\beta + n)\lambda + \text{constant}$
- Derivative of log-posterior is given by
  $\frac{\partial \log(P(\lambda|x))}{\partial \lambda} = \frac{\alpha + \sum_{i=1}^n x_i - 1}{\lambda} - (\beta + n) = 0$
- $\Rightarrow \hat{\lambda}_{\text{MAP}} = \frac{\alpha + \sum_{i=1}^n x_i - 1}{\beta + n}$
- This corresponds to the posterior mean, since the mean of a Gamma distribution is α/β

10 Linear Regression

Description

Task — Regression

Description —
- Supervised
- Parametric

Formulation

- $y^{(i)} = \beta \cdot x^{(i)}$ resp. $y = X\beta$ where X contains n rows, each of which represents an instance, and m columns, each of which represents a feature
- To incorporate offset, first column of X (i.e. first feature) is set to 1 and first element of β is set to β_0
- $\hat{y} = X\beta$ is a projection of y to the columnspace of X
- β lies in the rowspace of X resp. columnspace of $X^\top$

Optimization

Parameters — Find parameters β

Objective function — Ordinary least squares estimator (OLSE):
- Minimize mean squared error: $LO = \frac{1}{n}\sum_{i=1}^n (y^{(i)} - \beta \cdot x^{(i)})^2$ resp. $LO = (y - X\beta)^\top (y - X\beta)$

Optimization —
- $\nabla_\beta LO = \frac{1}{2}\nabla_\beta((y - \beta \cdot x)^2 = (y - \beta \cdot x)x = 0$ resp. $\nabla_\beta LO = \frac{1}{2}\nabla_\beta((y - X\beta)^\top (y - X\beta)) = \frac{1}{2}\nabla_\beta(\beta^\top X^\top X\beta - 2y^\top X\beta) = X^\top X\beta - X^\top y = X^\top(X\beta - y) = 0$
- $\Rightarrow \beta = (X^\top X)^{-1} X^\top y$

Alternatives to OLSE — MLE:
- Yields same result as OLSE
- The likelihood is: $p(y \mid \beta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|y - X\beta\|^2\right)$
- The log likelihood is: $\mathcal{L} = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|y - X\beta\|^2$
- We minimize: $\|y - X\beta\|^2$
- This is equivlent to OLSE

Orthogonality principle:
- $\hat{y} = X\beta$ is a projection of y to the columnspace of X
- We wish to minimize $\|\hat{y} - y\| = \|X\beta - y\|$ by selecting β appropriately
- By the orthogonality principle, $x^{[j]} \cdot (\hat{y} - y) = x^{[j]} \cdot (X\beta - y) = 0$ where $x^{[j]}$ is the j^{th} column of X resp. $X^\top(\hat{y} - y) = X^\top(X\beta - y) = 0$
- $\Rightarrow \beta = (X^\top X)^{-1} X^\top y$
- Alternatively, β lies in the columnspace of $X^\top$
- Then, we can express β as $X^\top[\alpha_1, ..., \alpha_n]^\top$
- This yields an equation system $y = XX^\top[\alpha_1, ..., \alpha_n]^\top$ which can be solved for α_i
- On that basis, β can be calculated

Pseudo Inverse:
- Yields same result as OLSE
- Minimum-norm solution
- β minimizes MSE if $\hat{y} = X\beta$ is a projection of y to the columnspace of X
- Given matrix projection via SVD, $XX^\# y$ is that projection
- $\Rightarrow \beta = X^\# y = (X^\top X)^{-1} X^\top y$
- Shows that β is largely determined by $X^\#$ and, thus, singular values of X based on SVD

PCA:
- Instances $y^{(i)}, x^{(i)} = \xi^{(i)}$ can be projected onto hyperplane given by $X\beta$
- Projections are given by $\hat{\xi}^{(i)}$
- Residuals are given by $e^{(i)} = \xi^{(i)} - \hat{\xi}^{(i)}$
- Since $e^{(i)}$ is orthogonal to $\hat{\xi}^{(i)}$, we can write using Pythagorean theorem: $\|e^{(i)}\|^2 = \|\xi^{(i)}\|^2 - \|\hat{\xi}^{(i)}\|^2$
- This is a PCA via SVD problem

Gradient descent:
- Minimum-norm solution
- Yields same result as OLSE

Hypothesis Testing of Found Parameters —
- Let $y|X \sim \mathcal{N}(y, \sigma^2 I) = \mathcal{N}(X\beta, \sigma^2 I)$
- Let $\hat{\beta} = (X^\top X)^{-1} X^\top y = X^+ y$ be the OLSE where X^+ is a scalar
- Then, $\hat{\beta} \sim \mathcal{N}(X^+ X\beta, X^{+\top}\sigma^2 X^+) = \mathcal{N}(\beta, (X^\top X)^{-1}\sigma^2)$
  Proof:
  - $\mathcal{N}(X^+ X\beta, X^{+\top}\sigma^2 X^+) = \mathcal{N}(I\beta, \sigma^2 X^+ X^{+\top})$ since X^+ is a scalar
  - Further, we have $\mathcal{N}(I\beta, \sigma^2 X^+((X^\top X)^{-1} X^\top)^\top) = \mathcal{N}(\beta, \sigma^2 X^+ X(X^\top X)^{-1\top}) = \mathcal{N}(\beta, \sigma^2(X^\top X)^{-1})$ since $(X^\top X)$ is symmetric
- We can estimate σ^2 unbiasedly as: $\hat{\sigma}^2 = \frac{1}{n-m}\sum_{i \le n}(X\hat{\beta} - y)^2$
- Then, confidence interval for $\hat{\beta}_j$ given by: $\hat{\beta}_j \pm z_{\alpha/2}\hat{se}(\hat{\beta}_j)$ where
  - $z_{\alpha/2} = \Phi^{-1}(\alpha/2)$ is Gaussian CDF
  - $\hat{se}(\hat{\beta}_j)$ is the j^{th} diagonal element of the covariance matrix $\sigma^2(X^\top X)^{-1}$
- We can perform a hypothesis test on $\hat{\beta}$ with the *Wald test*:
  - $H_0 : \beta = \beta_0$ (typically 0)

$H_1 : \beta \neq \beta_0$

- Wald statistic: $W = \frac{\hat{\beta}-\beta_0}{\hat{se}}$
- If p-value associated with W is smaller than α resp. if $|W|$ is greater than or equal to the critical value $z_{\alpha/2}$, we reject H_0

Evaluation —

- OLSE is unbiased if noise ϵ has zero mean:
  - Given $y = X\beta + \epsilon$, we can substitute $\hat{\beta} = (X^\top X)^{-1}X^\top(X\beta+\epsilon) = \beta + (X^\top X)^{-1}X^\top\epsilon$
  - Taking the expected value on both sides, we have: $\mathbb{E}(\hat{\beta}) = \beta + (X^\top X)^{-1}X^\top\mathbb{E}(\epsilon)$
  - Then, $\mathbb{E}(\hat{\beta}) = \beta$ if the noise has zero mean
- *Gauss Markov theorem*: OLSE is best (lowest variance, lowest MSE) unbiased estimator, if assumptions (X is full rank and there is no multicollinearity, heteroskedasticity, and exogeneity) are met
  Proof:
  - Let $\hat{\beta} = A^\top y = (X^\top X)^{-1}X^\top y$ be the OLSE
  - Let $C^\top y$ be another unbiased estimator
  - $\mathbb{V}(\hat{\beta}) = \mathbb{V}(A^\top y) = A^\top\mathbb{V}(y)A$ since A is constant
  - We can further develop to: $A^\top\sigma^2 I_m A = \sigma^2 A^\top A$ since variance is given by error term
  - Similarly, $\mathbb{V}(C^\top y) = \sigma^2 C^\top C$
  - For the OLSE, we can plug in $(X^\top X)^{-1}X^\top$ for A which yields: $\mathbb{V}(A^\top y) = \sigma^2(X^\top X)^{-1}X^\top X(X^\top X)^{-1} = \sigma^2(X^\top X)^{-1}$
  - Then, we have shown that $\mathbb{V}(A^\top y) \leq \mathbb{V}(C^\top y)$
- Nonetheless, there may be biased estimators that generate a lower variance and MSE

Characteristics —

- Convex with psd Hessian
- Has global minimum
- Has unique or infinitely many solutions
- Can be solved analytically, if $X^\top X$ is invertible
- In case of *multicollinearity*:
  - The rank of X is less than full, i.e. there are multiple columns (predictor variables) that are linearly dependent
  - $X^\top X$ is singular, i.e. non-invertible
  - There are multiple solutions for β
- If it has infinitely many solutions, the preferred solution is the *minimum-norm solution*, which minimizes $\|\beta\|$ and lies in the column space of $X^\top$ resp. is a solution to Xu

11 Linear Minimum Mean Squared Error Estimation (LMMSE)

Description

- Minimizes mean squared error of two random variables, leveraging information about their mean and covariance
- Linear regression with large samples is a data-based proxy for LMMSE, since $\frac{1}{n}\mathbb{E}[X^\top X] = \mathbb{E}[yy^\top]$ and $\frac{1}{n}\mathbb{E}[X^\top y] = \mathbb{E}[xy]$ and as $n \to \infty$ the strong law of large numbers applies

Formulation

- $y = ax + z$ is a vector of random variables and is observed
- a is a known vector
- z a zero-mean noise vector
- x is a random variable and quantity of interest
- We estimate x as $\hat{x} = h^\top y = \sum_i h_i y_i$
- This can be considered as a projection of x to the space spanned by y

Optimization

Parameters — Find parameters h

Objective function —

- Minimize expected squared error: $LO = \mathbb{E}[|\hat{x} - x|]$

Optimization —

- By the orthogonality principle, $\mathbb{E}[(\hat{x}-x)y_i] = \mathbb{E}[(h^\top y - x)y_i] = \mathbb{E}[(\sum_{l=1}^n h_l y_l - x)y_i] = 0$ for $i = 1,...,n$
- Then, $h^\top\mathbb{E}[yy_i] = \sum_{l=1}^n \mathbb{E}[y_l y_i]h_l = \mathbb{E}[xy_i]$ for $i = 1,...,n$ which in matrix notation corresponds to
  $\begin{bmatrix}\mathbb{E}[y_1y_1] & ... & \mathbb{E}[y_1y_n]\\ ... & ... & ...\\ \mathbb{E}[y_ny_1] & ... & \mathbb{E}[y_ny_n]\end{bmatrix}\begin{bmatrix}h_1\\ ...\\ h_n\end{bmatrix} = \begin{bmatrix}\mathbb{E}[xy_1]\\ ...\\ \mathbb{E}[xy_n]\end{bmatrix}$ resp. concisely
  $\mathbb{E}[yy^\top]h = \mathbb{E}[xy]$ where output is column vector (similar to linear regression), or $h^\top\mathbb{E}[yy^\top] = \mathbb{E}[xy^\top]$ where output is row vector (peculiar to LMMSE)
- Then, $h^\top = \frac{\mathbb{E}[xy^\top]}{\mathbb{E}[yy^\top]}$
- Note for exam: First set up matrix notation, then (if applicable) replace y_i by its observation $h_0 + h_1 x + z$, then evaluate elements in matrix, based on information given, then set up equation system, then solve it for h_i

Characteristics —

- $\mathbb{E}[yy^\top]$ is not invertible if there exists an $a \neq 0$ such that $y^\top a = 0$ with probability 1
  Proof:
  - Both y and a are in $\mathbb{R}^n$
  - Direction 1:
    - If $y^\top a = 0$, these two vectors are orthogonal. Then, y cannot span the entire space $\mathbb{R}^n$
    - This means that some rows or columns in the matrix $yy^\top$ are linearly dependent
    - By the invertible matrix theorem, such matrices are not invertible
  - Direction 2:
    - If the matrix $yy^\top$ is not invertible, by the invertible matrix theorem, it must have linearly dependent rows or columns
    - This means that one column or row can be expressed in terms of the sum over all other columns and rows
    - Then, the equation system $yy^\top a = 0$ is underdetermined
    - Then, it is possible to formulate $y^\top a = 0$ such that $a \neq 0$

12 Bayesian Linear Regression

Description

Task — Regression

Description —

- Supervised
- Parametric

Formulation

- $y^{(i)} = \beta \cdot x^{(i)} + \epsilon$ resp. $y = X\beta + \epsilon$
- $\beta \sim \mathcal{N}(0, T^2 I_m)$
- $p(\beta) \propto -\frac{1}{2T^2}\beta^\top\beta$
- $\epsilon \sim \mathcal{N}(0, \sigma^2)$

Optimization

Parameters — Find distribution of parameters β

Optimization —

- Prior $p(\beta)$:
  - $\beta \sim \mathcal{N}(0, T^2 I_m)$
  - $p(\beta) = \frac{1}{(2\pi T^2)^{m/2}} exp(-\frac{1}{2T^2}\beta^\top\beta)$
- Likelihood $p(y|X,\beta)$:
  - Conditional on β, $y \sim \mathcal{N}(X\beta, \sigma^2 I_n)$
  - $p(y|X,\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} exp(-\frac{1}{2\sigma^2}(y-X\beta)^\top(y-X\beta))$
- Posterior $p(\beta|X,y)$:
  - $p(\beta|X,y) \propto p(y|X,\beta) \times p(\beta) \propto exp(-\frac{1}{2\sigma^2}(y-X\beta)^\top(y-X\beta)) \times exp(-\frac{1}{2T^2}\beta^\top\beta) = exp(-\frac{1}{2}(\frac{1}{\sigma^2}y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X\beta) + \frac{1}{2T^2}\beta^\top\beta) \propto exp(-\frac{1}{2}(\beta^\top(\frac{1}{\sigma^2}X^\top X + \frac{1}{2T^2}I_m)\beta - \frac{2}{\sigma^2}\beta^\top X^\top y)$
  - We now apply a symmetric matrix property $x^\top Ax + 2x^\top b = (x + A^{-1}b)^\top A(x + A^{-1}b) - b^\top A^{-1}b$, with $\beta = x$, $(\frac{1}{\sigma^2}X^\top X + \frac{1}{2T^2}I_m) = A$ and $(\frac{1}{\sigma^2}X^\top y) = b$
  - Through this, we get $p(\beta|X,y) \propto \exp(\frac{1}{2}(\beta + (\frac{1}{\sigma^2}X^\top X + \frac{1}{T^2}I_m)^{-1}(\frac{1}{\sigma^2}X^\top y))^\top(\frac{1}{\sigma^2}X^\top X + \frac{1}{T^2}I_m)(\beta + (\frac{1}{\sigma^2}X^\top X + \frac{1}{T^2}I_m)^{-1}(\frac{1}{\sigma^2}X^\top y)))$
  - Thus, $p(\beta|X,y) \sim \mathcal{N}(\mu, \Sigma)$ with
    - $\mu = \Sigma \times \frac{1}{\sigma^2}X^\top y$
    - $\Sigma = (\frac{1}{\sigma^2}X^\top X + \frac{1}{T^2}I_m)^{-1}$
  - If we set an infinitely broad prior T^2 then the Bayesian estimate converges to the MLE estimate – if we have n = 0 training instances, the Bayesian estimate reverts to the prior

Characteristics —

- Convex with psd Hessian
- Has global minimum
- Can be solved analytically

13 Ridge (ℓ_2) Regression

Description

Task — Regression

Description —

- Supervised
- Parametric

Formulation

- $y^{(i)} = \beta \cdot x^{(i)}$ resp. $y = X\beta$

Optimization

Parameters — Find parameters β subject to $\|\beta\|^2 \leq t$ resp. $\|\beta\|^2 - t \leq 0$

Objective function —

- Minimize mean squared error subject to constraint
- Lagrangian formulation: $LO = \frac{1}{n}\sum_{i=1}^n (y^{(i)} - \beta \cdot x^{(i)})^2 + \lambda(\|\beta\|^2 - t)$ resp. $LO = (y - X\beta)^\top(y - X\beta) + \lambda(\|\beta\|^2 - t)$

Optimization —

- $\nabla_\beta LO = 0$
- $\Rightarrow \beta = (X^\top X + \lambda I)^{-1}X^\top y$

Alternative formulations — Still a OLSE problem:

- We can rewrite the objective to minimize
  $\|X\beta - y\|^2 + \lambda\|\beta\|^2 = \|\begin{bmatrix}X\beta - y\\ \lambda\beta\end{bmatrix}\|^2$ as the objective to minimize
  $\|X'\beta - y'\|^2$ with $X' = \begin{bmatrix}X\\ \lambda I\end{bmatrix}$ and $y' = \begin{bmatrix}y\\ 0\end{bmatrix}$

Bayesian regression (MAP estimation), where β is modeled as a zero-mean Gaussian variable, corresponds to ridge regression if λ is chosen as $\frac{\sigma^2}{\tau^2}$, where σ is standard deviation of y and τ is standard deviation of β, where high τ implies low confidence in prior:

- Prior $p(\beta)$:
  - $\beta \propto \mathcal{N}(0, \tau^2 I_m)$
  - $p(\beta) = \frac{1}{(2\pi\tau^2)^{m/2}} exp(-\frac{1}{2\tau^2}\beta^\top\beta)$
- Likelihood $p(y|X,\beta)$:
  - Conditional on β, $y \propto \mathcal{N}(X\beta, \sigma^2 I_n)$
  - $p(y|X,\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} exp(-\frac{1}{2\sigma^2}(y-X\beta)^\top(y-X\beta))$

- Posterior $p(\beta|X,y)$:
  - $p(\beta|X,y) \propto p(y|X,\beta) \times p(\beta) \propto \log(p(y|X,\beta) \times p(\beta)) \propto \log(exp(-\frac{1}{2\sigma^2}(y-X\beta)^\intercal(y-X\beta)) \times exp(-\frac{1}{2T^2}\beta^\intercal\beta)) \propto -\frac{\|y-X\beta\|^2}{\sigma^2} - \frac{\|\beta\|^2}{\tau^2}$
- If we maximize log posterior (MAP estimate), we have:
  $\arg\min_\beta \frac{\|y-X\beta\|^2}{\sigma^2} + \frac{\|\beta\|^2}{\tau^2} = \|y-X\beta\|^2 + \frac{\sigma^2}{\tau^2}\|\beta\|^2$
- This mirrors log loss of ridge regression with $\lambda = \frac{\sigma^2}{\tau^2}$

Orthogonality principle:
- We wish to minimize $\|\hat{y}-y\| = \|X\beta - y\|$ by selecting β subject to the condition that $C\beta = d$
- Let β and $\tilde{\beta}$ be solutions of this condition
- Then, we can rewrite condition as: $C\beta - C\tilde{\beta} = d - d = C(\beta - \tilde{\beta}) = 0$
- Then, $(\beta - \tilde{\beta})$ is in the nullspace of C, which is spanned by the columns of B
- Then, $(\beta - \tilde{\beta})$ can be represented as a linear combination of the basis of the nullspace: $(\beta - \tilde{\beta}) = B\beta'$
- From this, we get $\beta = B\beta' + \tilde{\beta}$
- Then, the cost function amounts to $\|X(B\beta' + \tilde{\beta}) - y\| = \|XB\beta' + X\tilde{\beta} - y\| = \|X'\beta' - y'\|$ where $X' = XB$ and $y' = y - X\tilde{\beta}$

Effect —
- Shrinks certain elements of β to near 0
  Proof:
  - Gradient at optimality given by $\frac{\partial(y-X\beta)^\intercal(y-X\beta)}{\partial\beta} + 2\lambda\beta = 0$
  - Then, $\beta^* = -\frac{1}{2\lambda}\frac{\partial(y-X\beta)^\intercal(y-X\beta)}{\partial\beta}$
  - This means that each parameter is shrunk by a factor determined by size of λ - the larger λ, the more the parameters are shrunk
  - Larger parameters experience a larger shrinkage
- Addresses multicollinearity:
  - SVD for $X = USV^\intercal$
  - We can show that $X\beta = US\left(S^2 + \lambda I\right)^{-1} SU^\intercal Y$
    Proof:
    - $X\beta^r = X(X^\intercal X + \lambda I)^{-1} X^\intercal Y$
    - $= UDV^\intercal((UDV^\intercal)^\intercal UDV^\intercal + \lambda I)^{-1}(UDV^\intercal)^\intercal Y$
    - $= UDV^\intercal(VDU^\intercal UDV^\intercal + \lambda I)^{-1} VDU^\intercal Y$
    - $= UDV^\intercal\left(VD^2V^\intercal + \lambda VV^\intercal\right)^{-1} VDU^\intercal Y$
    - $= UDV^\intercal\left(V(D^2 + \lambda I)V^\intercal\right)^{-1} VDU^\intercal Y$
    - $= UDV^\intercal V\left(D^2 + \lambda I\right)^{-1} V^\intercal VDU^\intercal Y$
    - $= UD\left(D^2 + \lambda I\right)^{-1} DU^\intercal Y$
  - Similarly, we can show that $\|\beta\|^2 = Y^\intercal US^2\left(S^2 + \lambda I\right)^{-2} U^\intercal Y = \frac{W^\intercal S^2 W}{(S^2+\lambda I)^2}$ where $W = U^\intercal Y$
  - In case of multicollinearity, the rank of X is less than full, and S^2 cannot be inverted. By adding λI to S, ridge regression ensures that the equation remains solvable even if S is not invertible on its own

Characteristics —
- Strictly with pd Hessian, since Lagrangian term is strictly convex and the sum of a strictly convex function with a convex function is strictly convex
- Has global minimum
- Has unique solution, as $(X^\intercal X + \lambda I)$ has linearly independent columns
- Can be solved analytically, as $(X^\intercal X + \lambda I)$ is always invertible

14 Lasso (ℓ_1) Regression

Description

Task — Regression

Description —
- Supervised
- Parametric

Formulation

- $y^{(i)} = \beta \cdot x^{(i)}$ resp. $y = X\beta$

Optimization

Parameters — Find parameters β subject to $|\beta| \leq t$ resp. $|\beta| - t \leq 0$

Objective function —
- Minimize mean squared error subject to constraint
- Lagrangian formulation: $LO = \frac{1}{n}\sum_{i=1}^n (y^{(i)} - \beta \cdot x^{(i)})^2 + \lambda(|\beta| - t)$ resp. $LO = (y - X\beta)^\intercal(y - X\beta) + \lambda(|\beta| - t)$

Alternative formulations — Bayesian regression (MAP estimation), where β is modeled as a zero-mean Laplacian variable, corresponds to LASSO regression if λ is chosen as $\frac{\sigma^2}{b}$, where σ is standard deviation of y and b is the scale parameter of the Laplacian prior, where high b implies low confidence in prior:
- Prior $p(\beta)$:
  - $\beta \propto \text{Laplacian}(0, b)$
  - $p(\beta) = \prod_{j=1}^m \frac{1}{2b}\exp\left(-\frac{|\beta_j|}{b}\right)$
- Likelihood $p(y|X,\beta)$:
  - Conditional on β, $y \propto \mathcal{N}(X\beta, \sigma^2 I_n)$
  - $p(y|X,\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^\intercal(y - X\beta)\right)$
- Posterior $p(\beta|X,y)$:
  - $p(\beta|X,y) \propto p(y|X,\beta) \times p(\beta)$
  - $\log p(\beta|X,y) \propto \log\left(\exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^\intercal(y - X\beta)\right) \times \prod_{j=1}^m \exp\left(-\frac{|\beta_j|}{b}\right)\right)$
  - $\log p(\beta|X,y) \propto -\frac{\|y-X\beta\|^2}{\sigma^2} - \frac{\|\beta\|_1}{b}$
- If we maximize log posterior (MAP estimate), we have:
  $\arg\min_\beta \frac{\|y-X\beta\|^2}{\sigma^2} + \frac{\|\beta\|_1}{b} = \|y - X\beta\|^2 + \frac{\sigma^2}{b}\|\beta\|_1$
- This mirrors the log loss of LASSO regressionwith $\lambda = \frac{\sigma^2}{b}$

Effect —
- Shrinks certain elements of β to 0
  Proof:
  - Gradient at optimality given by $\frac{\partial(y-X\beta)^\intercal(y-X\beta)}{\partial\beta} + \frac{\partial\lambda|\beta|}{\partial\beta} = 0$
  - $\frac{\partial\lambda|\beta|}{\partial\beta}$ non-differentiable because there is a sharp edge at $\beta = 0$, but we can work with subgradients for $\beta \neq 0$:
    $\frac{\partial}{\partial\beta}|\beta| = sgn(\beta) = \begin{cases} -1 & \beta < 0 \\ 0 & \beta = 0 \\ 1 & \beta > 0 \end{cases}$
  - If we have $-\lambda < \frac{\partial(y-X\beta)^\intercal(y-X\beta)}{\partial\beta} < \lambda$ the optimum is given by $\beta = 0$
  - This means that some parameters are set to 0
  - The larger λ, the more parameters are set to 0
  - Small parameter values (i.e. unimportant features) are more likely to be set to 0
  - For parameters that are not set to 0, LASSO regression has a similar effect as ridge regression and shrinks these parameters towards 0

Characteristics —
- Convex, but not strictly convex
- Has global minimum
- Has unique or infinitely many solutions
- Cannot be solved analytically, since $|\beta|$ is not differentiable at $\beta_i = 0$

15 Polynomial Regression

Description

Task — Regression

Description —
- Supervised
- Parametric

Formulation

- $y^{(i)} = \beta \cdot \phi(x^{(i)})$ resp. $y = \Phi\beta$ where Φ is the transformed design matrix with rows $\phi(x^{(i)})^\intercal$

Optimization

Parameters — Find parameters β

Objective function —
- Ordinary least squares estimator
- Minimize mean squared error

Optimization —
- $\nabla_\beta LO = 0$
- $\Rightarrow \beta = (\Phi^\intercal\Phi)^{-1}\Phi^\intercal y$

Characteristics —
- Convex with psd Hessian
- Has global minimum
- Has unique or infinitely many solutions
- Can be solved analytically, if $(\Phi^\intercal\Phi)$ is invertible

16 Kernel Methods

Background on Kernel Methods

Description —
- Mechanism for tractably resp. implicitly mapping data into higher-dimensional feature space so that linear models can be used in this feature space
- To do so, we can employ the *kernel trick* and the *representer theorem*
- The requirements are that the kernel function fulfills *Mercer's theorem*, i.e. the kernel is a Mercer kernel

Kernel trick —
- Allows to operate in higher-dimensional feature space, without explicitly calculating this transformation, but instead implicitly computing the inner product in this feature space via a kernel function
- Given two inputs $x^{(i)}, x^{(j)}$ and a feature map $\varphi : \mathbb{R}^m \to \mathbb{R}^k$ we can define an inner product on $\mathbb{R}^k$ via the kernel function: $k(x^{(i)}, x^{(j)}) = \varphi(x^{(i)}) \cdot \varphi(x^{(j)})$
- If a prediction function is described solely in terms of inner products in the input space, it can be lifted into the feature space by replacing the inner product with the kernel function
- Kernel trick requires that span of training instances $span(\varphi(x^{(i)}), ..., \varphi(x^{(N)})) = \mathbb{R}^k$ and, thus, that $N \geq k$
  Proof: $dim(span(...)) = \begin{cases} N & \text{if } N < k \\ k & \text{if } N \geq k \end{cases}$
- Kernel trick cannot be used in conjunction with feature selection resp. sparsity inducing regularize (e.g. ℓ_1), as this does not satisfy the representer theorem

Representer theorem —

- Allows to avoid directly seeking the k parameters, but only the n parameters that characterize α
- Allows to avoid calculating $\varphi(z)$ when evaluating novel instance, but only sum over weighted set of n kernel function outputs

Mercer's theorem —
- Kernel function is psd and symmetric iff $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = \varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(\boldsymbol{x}^{(j)})$
  - Psd: $x^\top K x \geq 0$ where K is the kernel matrix
  - Symmetric: $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = k(\boldsymbol{x}^{(j)},\boldsymbol{x}^{(i)})$
- Kernel that satisfies Mercer's theorem is a Mercer kernel, i.e. we can prove a kernel is a Mercer kernel either if it is psd and symmetric or by finding a feature map such that the kernel function corresponds to an inner product

Formulation

- Feature map $\varphi : \mathbb{R}^m \to \mathbb{R}^k$
- Linear prediction function: $\beta \cdot \varphi(\boldsymbol{x}^{(i)})$
- Regularized loss function: $LO = \sum_{i=1}^{n} LO(y^{(i)}, \beta \cdot \varphi(\boldsymbol{x}^{(i)}) + \Omega(\beta))$
- Iff $\Omega(\beta))$ is a non-decreasing function, then the parameters β that minimize the loss function can be rewritten as: $\beta = \sum_{i=1}^{n} \alpha^{(i)} \varphi(\boldsymbol{x}^{(i)})$
- Outcome of novel instance can be predicted as: $\beta \cdot \varphi(z) = \sum_{i=1}^{n} \alpha^{(i)} \varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(z) = \sum_{i=1}^{n} \alpha^{(i)} k(\boldsymbol{x}^{(i)}, z)$
- Act of prediction becomes act of measuring similarity to training instances in feature map space

Kernel Types

Polynomial kernel —
- $\varphi(x) = [x^\alpha]_{\alpha \in \mathbb{N}^m}$ where $\alpha = (\alpha_1, ..., \alpha_m)$ is the multi-index representing the power and $x^\alpha = x_1^{\alpha_1} \times ... \times x_m^{\alpha_m}$ is the mononomial term corresponding to the multi-index α
- E.g. if degree = 2, then $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = 1 + 2x_1^{(i)}x_1^{(j)} + 2x_2^{(i)}x_2^{(j)} + (x_1^{(i)}x_1^{(j)})^2 + (x_2^{(i)}x_2^{(j)})^2 + 2x_1^{(i)}x_1^{(j)}x_2^{(i)}x_2^{(j)}$
- Inner product diverges to infinity
- To address this, we often use RBF kernel instead

RBF kernel —
- $\varphi(\boldsymbol{x}) = exp(-\frac{1}{2}\|\boldsymbol{x}\|^2)[\frac{\boldsymbol{x}^\alpha}{\sqrt{\alpha!}}]_{\alpha \in \mathbb{N}^m}$
- $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = \sigma^2 exp(-\frac{\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2}{2l^2})$

  Proof:
  - $exp(-\frac{1}{2}\|\boldsymbol{x}^{(i)}\|^2) exp(-\frac{1}{2}\|\boldsymbol{x}^{(j)}\|^2) \sum_\alpha [\frac{\boldsymbol{x}^{(i)\alpha}\boldsymbol{x}^{(j)\alpha}}{\alpha!}]$
  - Given multinomial series expansion, $\sum_\alpha [\frac{\boldsymbol{x}^{(i)\alpha}\boldsymbol{x}^{(j)\alpha}}{\alpha!}] = exp(\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)})$
  - Then, we get $exp(-\frac{1}{2}\|\boldsymbol{x}^{(i)}\|^2 - \frac{1}{2}\|\boldsymbol{x}^{(j)}\|^2 + \boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}) = exp(-\frac{\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2}{2})$
- Gives access to infinite feature space

  Proof:
  - By expanding the square, the kernel can be rewritten as: $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = \sigma^2 \exp(-\frac{\|\boldsymbol{x}^{(i)}\|^2}{2l^2}) \exp(-\frac{\|\boldsymbol{x}^{(j)}\|^2}{2l^2}) \exp(\frac{\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}}{l^2})$
  - We can expand the last term: $\exp(\frac{\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}}{l^2}) = \sum_{k=0}^{\infty} \frac{(\frac{\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}}{l^2})^k}{k!}$.
  - Then, we get: $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = \sigma^2 \exp(-\frac{\|\boldsymbol{x}^{(i)}\|^2}{2l^2}) \exp(-\frac{\|\boldsymbol{x}^{(j)}\|^2}{2l^2}) \sum_{k=0}^{\infty} \frac{(\frac{\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}}{l^2})^k}{k!}$.
  - Assume a feature map of the form $\phi_k(x) = P_k(x)\exp(-\frac{\|x\|^2}{2l^2})$, where $P_k(x)$ is a polynomial
  - Then, we can equate: $\sum_{k=0}^{\infty} P_k(\boldsymbol{x}^{(i)}) P_k(\boldsymbol{x}^{(j)}) e^{-\frac{\|\boldsymbol{x}^{(i)}\|^2}{2l^2}} e^{-\frac{\|\boldsymbol{x}^{(j)}\|^2}{2l^2}} = \sum_{k=0}^{\infty} \frac{(\frac{\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}}{l^2})^k}{k!} e^{-\frac{\|\boldsymbol{x}^{(i)}\|^2}{2l^2}} e^{-\frac{\|\boldsymbol{x}^{(j)}\|^2}{2l^2}}$
  - We can see that the RBF kernel corresponds to an infinite-dimensional feature space, since the series expansion of the exponential represents an infinite sum of polynomials
  - Based on the equation, we can specify the polynomials to $P_k(x) = \frac{x^k}{\sqrt{k!}}$
  - This results in a feature map of the form: $\phi_k(x) = \frac{x^k}{\sqrt{k!}} \exp(-\frac{\|x\|^2}{2l^2})$
- Length scale parameter l controls how quickly the similarity decays with distance: If l is large, points with high distance still have high covariance
- Variance parameter σ controls the vertical scale of the function
- RBF kernel is *stationary*, meaning that only the relative distance between two points determines the value output by the kernel function
- Challenge: Cannot ignore irrelevant dimensions (whereas e.g. a neural network can do this by setting the associated weights to 0)

Periodic kernel —
- Suitable for capturing periodic patterns
- $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = \sigma^2 exp(-\frac{2\sin^2(\frac{\pi\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|}{p})}{l^2})$
- Period of oscillation p controls the length of the cycle

Laplace kernel —
- Suitable for modeling sharper edges than the RBF kernel
- $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = \sigma^2 exp(-\frac{|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}|}{l})$

Kernel compositions —
- New valid kernels can be composed via:
  - Addition: $k_1 + k_2$
  - Multiplication: $k_1 \times k_2$
  - Scaling: $c \times k_1$ for $c > 0$
  - Composition: $f(k_1)$ where f is a polynomial with positive coefficients or the exponential function
- Valid kernels:
  - $k'(x_1, x_2) = ck(x_1, x_2)$, since $\varphi'(x) = \sqrt{c}\varphi(x)$
  - $k'(x_1, x_2) = f(x_1)k(x_1, x_2)f(x_2)$, since $\varphi'(x) = f(x)\varphi(x)$
  - $k'(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$, since the requirements for a valid kernel are that its psd and symmetric, which is retained when two psd and symmetric matrices are added resp. since $\varphi'(x) = \begin{bmatrix}\varphi_1(x)\\ \varphi_2(x)\end{bmatrix}$
  - $k'(x_1, x_2) = k_1(x_1, x_2)k_2(x_1, x_2)$, since new kernel is given by the i^{th} feature value under feature map φ_1 multiplied by the j^{th} feature value under feature map φ_2
  - $k'(x_1, x_2) = exp(k(x_1, x_2))$, since we can apply Taylor series expansion $\sum_{n=1}^{r} \frac{k(x_1,x_2)^r}{r!} = exp(k(x_1, x_2)) = k'(x_1, x_2)$ as $r \to \infty$ and we know that exponentiation, addition, and scaling produces valid new kernels from above
  - $k'(x_1, x_2) = x_1^\top A x_2$ for psd and symmetric A

17 Polynomial Kernel Regression

Description

Task — Regression

Description —
- Supervised
- Parametric

Formulation

- $y = \beta \cdot \varphi(\boldsymbol{x}^{(i)})$

Optimization

Parameters — Find parameters β

Objective function —
- Ordinary least squares estimator (OLSE)
- Minimize mean squared error: $LO = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \beta \cdot \varphi(\boldsymbol{x}^{(i)}))^2$

Optimization —
- Primal solution:
  - Parameters can be estimated as: $\beta = (\Phi^\top\Phi)^{-1}\Phi^\top y$
  - Prediction for novel instance: $\beta \cdot \varphi(z) = (\Phi^\top\Phi)^{-1}\Phi^\top y \cdot \varphi(z) = y^\top\Phi(\Phi^\top\Phi)^{-1}\varphi(z)$
- Let us define $K = \Phi\Phi^\top$ as the kernel matrix of the training data with $K_{ij} = \varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(\boldsymbol{x}^{(j)})$
- Dual solution α if we have no regularization, i.e. $\lambda = 0$:
  - Parameters can be estimated as: $\beta = \Phi^\top K^{-1} y$

    Proof:
    - $(\Phi^\top\Phi + \lambda I_D)\beta = \Phi^\top y$
    - $\Rightarrow \Phi^\top\Phi\beta + \lambda I_D\beta = \Phi^\top y$
    - $\Rightarrow I_D\beta = \Phi^\top\lambda^{-1}(y - \Phi\beta)$
    - Since we know from the representer theorem that $\beta = \Phi^\top\alpha$, we can say: $\alpha = \lambda^{-1}(y - \Phi\beta)$
    - We can further develop this to: $\lambda\alpha = (y - \Phi\beta)$
    - Replacing β by $\Phi^\top\alpha$ yields: $\lambda\alpha = (y - \Phi\Phi^\top\alpha)$
    - $\Rightarrow \alpha = (\Phi\Phi^\top + \lambda I_N)^{-1}y = K^{-1}y$
    - With this, we can calculate the parameters: $\beta = \Phi^\top\alpha = \Phi^\top(\Phi\Phi^\top + \lambda I_N)^{-1}y = \Phi^\top K^{-1}y$

    Proof 2:
    - According to *Sherman-Morrison-Woodbury Formula*, $(FH^{-1}G + E)^{-1}FH^{-1} = E^{-1}F(GE^{-1}F + H)^{-1}$
    - If we assume $E = I_D, F = \Phi^\top, G = \Phi, H = I_N$, the formula simplifies to $(\Phi^\top\Phi + I_D)^{-1}\Phi^\top = I_D^{-1}\Phi^\top(\Phi\Phi^\top + I_N)^{-1}$
    - Since $I_D^{-1} = I_D$, we have: $(\Phi^\top\Phi + I_D)^{-1}\Phi^\top = \Phi^\top(\Phi\Phi^\top + I_N)^{-1}$
    - $\Rightarrow (\Phi^\top\Phi + I_D)^{-1}\Phi^\top y = \hat{\beta} = \Phi^\top(\Phi\Phi^\top + I_N)^{-1}y$
  - Prediction for novel instance: $\beta \cdot \varphi(z) = y^\top(\Phi\Phi^\top)^{-1}\Phi\varphi(z) = y^\top(\Phi\Phi^\top)^{-1}k$ where $k = \Phi\varphi(z) = [k(\boldsymbol{x}^{(1)}, z), ..., k(\boldsymbol{x}^{(n)}, z)]^\top = [\varphi(\boldsymbol{x}^{(1)}) \cdot \varphi(z), ..., \varphi(\boldsymbol{x}^{(n)}) \cdot \varphi(z)]^\top$ is a kernel vector, consisting of kernel values between training instances and new instance

Algorithm — Training:
1. Compute kernel matrix given RBF kernel
   Time complexity: $\mathcal{O}(n^2 \times m)$ for n^2 kernel matrix values and m number of features in each instance vector
2. Perform training by solving $\alpha = K^{-1}y$ for α
   Time complexity: $\mathcal{O}(n^3)$
3. Store α
   Space complexity: $\mathcal{O}(n^2)$

Prediction:
1. Compute kernel vector
   Time complexity: $\mathcal{O}(n \times m \times d)$ for d new instances, given n instances in training data and m features in each instance vector
2. Store k
   Space complexity: $\mathcal{O}(n \times d)$ for d new instances, given n as length of kernel vector
3. Predict response using stored kernel vector
   Time complexity: $\mathcal{O}(n \times d)$ for d new instances, given n as length of α

Value:
- Primal solution training is of time complexity $\mathcal{O}(k^3)$ and prediction is of time complexity $\mathcal{O}(k)$
- Dual solution speeds this up as seen above in the algorithm

Characteristics —
- Convex with psd Hessian
- Has global minimum
- Has unique or infinitely many solutions
- Can be solved analytically

18 Kernel K-Means Clustering

Formulation

- We specify that we want $j = 1,...,k$ clusters in total
- Clusters defined by centroid $\boldsymbol{\mu}^{[j]}$
- Instances $i = 1,...,n$ given by $\boldsymbol{x}^{(i)}$
- Clusters and instances can be lifted from input to feature space via mapping $\phi(\cdot)$
- Each instance i has k indicator variables, which describe whether instance i is assigned to cluster j, given by $\{p^{i[j]}\}_{j=1}^{k} \in [0,1]$

Optimization

Parameters — Find centroids $\boldsymbol{\mu}^{[j]}$, i.e., find cluster assignments (instance always assigned to closest cluster in feature space)

Objective function —
- Minimize the distance between each instance and the centroid of its closest cluster in feature space: $j^* = \arg\min_j \|\phi(\boldsymbol{x}^{(i)}) - \phi(\boldsymbol{\mu}^{[j]})\|^2$
- Distortion function given by:

  $\Theta = \sum_{i=1}^{n}\sum_{j=1}^{k} p^{i[j]}\|\phi(\boldsymbol{x}^{(i)}) - \phi(\boldsymbol{\mu}^{[j]})\|^2 =$

  $\sum_{\boldsymbol{x}^{(i)}\in C_j}\sum_{j=1}^{k}\|\phi(\boldsymbol{x}^{(i)}) - \phi(\boldsymbol{\mu}^{[j]})\|^2$ with $p^{i[j]} = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases}$

  and $C_j = \left\{\boldsymbol{x}^{(i)} \mid p^{i[j]} = 1\right\}$
- $\phi(\boldsymbol{\mu}^{[j]})$ is defined by mean of points in cluster C_j in feature space: $= \frac{1}{|C_j|}\sum_{\boldsymbol{x}^{(\mu)}\in C_j}\phi(\boldsymbol{x}^{(\mu)})$
- Then, we can rewrite the distance explicitly in terms of the kernel function: $\Theta = \sum_{j=1}^{k}\sum_{\boldsymbol{x}^{(i)}\in C_j}\Big[K(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(i)}) -$

  $\frac{2}{|C_j|}\sum_{\boldsymbol{x}^{(\mu)}\in C_j}K(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(\mu)}) + \frac{1}{|C_j|^2}\sum_{\boldsymbol{x}^{(\mu)},\boldsymbol{x}^{(\rho)}\in C_j}K(\boldsymbol{x}^{(\mu)},\boldsymbol{x}^{(\rho)})\Big]$

Optimization — Lloyd's algorithm:
1. Randomly initialize each $\boldsymbol{\mu}^{[j]}$ in feature space implicitly, using the kernel function
2. E-Step:
   - Re-assign instances while keeping all centroids fixed, i.e., minimize Θ with respect to $p^{i[j]}$
3. M-Step:
   - Re-compute centroids implicitly in feature space while keeping all instance assignments fixed, i.e., minimize Θ with respect to $\boldsymbol{\mu}^{[j]}$: $\phi(\boldsymbol{\mu}^{[j]}) = \frac{\sum_{i=1}^{n} p^{i[j]}\phi(\boldsymbol{x}^{(i)})}{\sum_{i=1}^{n} p^{i[j]}} = \frac{1}{|C_j|}\sum_{\boldsymbol{x}^{(i)}\in C_j}\phi(\boldsymbol{x}^{(i)})$
4. Repeat E- and M-Step until convergence

19 Logistic Regression

Description

Task — Binary classification

Description —
- Supervised
- Parametric

Formulation

- Probability of each class is estimated via sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$
- $P(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-\beta\cdot\boldsymbol{x}}} = \frac{e^{\beta\cdot\boldsymbol{x}}}{1+e^{\beta\cdot\boldsymbol{x}}}$
- $P(y=0|\boldsymbol{x}) = \frac{1}{1+e^{\beta\cdot\boldsymbol{x}}} = \frac{e^{-\beta\cdot\boldsymbol{x}}}{1+e^{-\beta\cdot\boldsymbol{x}}}$
- Odds: $\frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})} = e^{\beta\cdot\boldsymbol{x}}$
- Log-odds: $ln(\frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})}) = \beta\cdot\boldsymbol{x} = ln(\frac{P(y=1)}{P(y=0)}) + ln(\frac{P(\boldsymbol{x}|y=1)}{P(\boldsymbol{x}|y=0)})$, i.e. can be decomposed into prior and likelihood
- We can influence the prior by introducing an offset to the log-odds: $\beta\cdot\boldsymbol{x} + t$:
  - In multinomial case: $p_k = \frac{e^{z_k+t_k}}{\sum_{\ell=1}^{k} e^{z_\ell+t_\ell}}$ where $t_k = \ln(\frac{q(x_k)}{p(x_k)})$
  - Then, the prior on x ($p(x)$) is replaced by $q(x)$
- Geometrically, $z = \beta\cdot\boldsymbol{x}$ defines a linear separating hyperplane:
  - When $z > 0$ resp. log-odds > 0, then odds > 1 resp. $P(y=1|\boldsymbol{x}) > P(y=0|\boldsymbol{x})$ resp. $\sigma(z) > 0.5$, then predict $y = 1$
  - When $z < 0$ resp. log-odds < 0, then odds < 1 resp. $P(y=1|\boldsymbol{x}) < P(y=0|\boldsymbol{x})$ resp. $\sigma(z) < 0.5$, predict $y = 0$
  - When $z = 0$ resp. log-odds $= 0$, then odds $= 1$ resp. $P(y=1|\boldsymbol{x}) = P(y=0|\boldsymbol{x})$ resp. $\sigma(z) = 0.5$, then we are on the decision boundary
  - Decision boundary z is orthogonal to β
    Proof:
    * For any 2 points x_1, x_2 on the decision boundary $z = 0$, i.e. $\beta\cdot x_1 = 0, \quad \beta\cdot x_2 = 0$ Since x_1, x_2 are on the decision boundary, vector z can be considered a linear combination of $x_1 - x_2$
    * Combining these equations, we get $\beta\cdot(x_1 - x_2) = \beta\cdot uz = 0$

Optimization

Parameters — Find parameters β

Objective function —
- Likelihood: $L(\beta) = \prod_{i=1}^{n} P(y^{(i)}|\boldsymbol{x}^{(i)};\beta) = \prod_{i=1}^{n}\sigma(z^{(i)})^{y^{(i)}}(1-\sigma(z^{(i)}))^{1-y^{(i)}}$
- Maximize log-likelihood: $\log L(\beta) = \sum_{i=1}^{n}\left[y^{(i)}\log\sigma(z^{(i)}) + (1-y^{(i)})\log(1-\sigma(z^{(i)}))\right] =$

  $\sum_{i=1}^{n}\left[y^{(i)}\log\frac{1}{1+e^{-z^{(i)}}} + (1-y^{(i)})\log\frac{e^{-z^{(i)}}}{1+e^{-z^{(i)}}}\right] =$

  $\sum_{i=1}^{n}\left[y^{(i)}z^{(i)} - \log(1+e^{z^{(i)}})\right]$
- Minimize log-loss: $-\log L(\beta)$

Optimization —
- $\frac{\partial -\log L(\beta)}{\partial\beta} = -\sum_{i=1}^{n}\frac{\partial}{\partial\beta}[y^{(i)}\log\sigma(z^{(i)}) + (1-y^{(i)})$

$\log(1-\sigma(z^{(i)}))] = \sum_{i=1}^{n}[\sigma(z^{(i)}) - y^{(i)}]\frac{\partial z^{(i)}}{\partial\beta} = \sum_{i=1}^{n}[\sigma(z^{(i)}) - y^{(i)}]\boldsymbol{x}^{(i)}$

Proof:
- Derivative of the sigmoid: $\frac{\partial\sigma(z^{(i)})}{\partial z^{(i)}} = \sigma(z^{(i)})(1-\sigma(z^{(i)}))$
- Derivative of the first term: $\frac{\partial}{\partial\beta}\left[y^{(i)}\log\sigma(z^{(i)})\right] = y^{(i)}\frac{1}{\sigma(z^{(i)})}\frac{\partial\sigma(z^{(i)})}{\partial z^{(i)}}\frac{\partial z^{(i)}}{\partial\beta} = y^{(i)}\frac{1}{\sigma(z^{(i)})}\sigma(z^{(i)})(1-\sigma(z^{(i)}))\boldsymbol{x}^{(i)} = y^{(i)}(1-\sigma(z^{(i)}))\boldsymbol{x}^{(i)} = y^{(i)}\boldsymbol{x}^{(i)} - y^{(i)}\sigma(z^{(i)})\boldsymbol{x}^{(i)}$
- Derivative of the second term: $\frac{\partial}{\partial\beta}\left[(1-y^{(i)})\log(1-\sigma(z^{(i)}))\right] = (1-y^{(i)})\frac{1}{1-\sigma(z^{(i)})}(-1)\frac{\partial\sigma(z^{(i)})}{\partial z^{(i)}}\frac{\partial z^{(i)}}{\partial\beta} =$

  $(1-y^{(i)})\frac{1}{1-\sigma(z^{(i)})}(-1)\sigma(z^{(i)})(1-\sigma(z^{(i)}))\boldsymbol{x}^{(i)} =$

  $-(1-y^{(i)})\sigma(z^{(i)})\boldsymbol{x}^{(i)} = y^{(i)}\sigma(z^{(i)})\boldsymbol{x}^{(i)} - \sigma(z^{(i)})\boldsymbol{x}^{(i)}$
- If we set gradient to 0, we have expectation matching: $\sum_{i=1}^{n}\sigma(z^{(i)})\boldsymbol{x}^{(i)} = \sum_{i=1}^{n} y^{(i)}\boldsymbol{x}^{(i)}$ resp. optimum is where expected feature counts, weighted by predicted probability = observed feature counts, weighted by true labels

Characteristics —
- Convex (since sum of convex functions in log loss is convex) with psd Hessian
- Has global minimum
- Has unique or infinitely many solutions
- Can be solved numerically

Proof that log loss is convex:
- Sum of convex functions is convex
- Thus, we need to prove convexity of $\ln(1+e^{\beta\cdot\boldsymbol{x}^{(i)}})$ and $-y^{(i)}\beta\cdot\boldsymbol{x}^{(i)}$
- For second term:
  - $\mathcal{H}(\beta) = \nabla_\beta^2\left(-y^{(i)}\beta\cdot\boldsymbol{x}^{(i)}\right) = 0$
  - $\mathcal{H} \geqslant 0$
- For first term:
  - $\mathbb{H}(\beta) = \nabla_\beta^2\ln(1+e^{\beta\cdot\boldsymbol{x}^{(i)}})$
  - $= v(x)\times u'(x) - v'(x)\times u(x) = \frac{1}{1+e^{\beta\cdot\boldsymbol{x}}}\times e^{\beta\cdot\boldsymbol{x}}\times\boldsymbol{x}\boldsymbol{x}^\top - \frac{e^{\beta\cdot\boldsymbol{x}}\times\boldsymbol{x}}{(1+e^{\beta\cdot\boldsymbol{x}})^2}\times\boldsymbol{x}^\top\times e^{\beta\cdot\boldsymbol{x}}$
  - $= \frac{e^{\beta\cdot\boldsymbol{x}}\boldsymbol{x}\boldsymbol{x}^\top(1+e^{\beta\cdot\boldsymbol{x}}) - e^{\beta\cdot\boldsymbol{x}}\boldsymbol{x}\boldsymbol{x}^\top e^{\beta\cdot\boldsymbol{x}}}{(1+e^{\beta\cdot\boldsymbol{x}})^2}$
  - $= \frac{e^{\beta\cdot\boldsymbol{x}}\boldsymbol{x}\boldsymbol{x}^T}{(1+e^{\beta\cdot\boldsymbol{x}})^2}$
  - $\mathcal{H} \geqslant 0$
    Proof: $\boldsymbol{a}^T\mathcal{H}\boldsymbol{a} = \frac{e^{\beta\cdot\boldsymbol{x}}}{(1+e^{\beta\cdot\boldsymbol{x}})^2}\boldsymbol{a}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{a} = \frac{e^{\beta\cdot\boldsymbol{x}}}{(1+e^{\beta\cdot\boldsymbol{x}})^2}\|\boldsymbol{a}^T\boldsymbol{x}\|^2 \geq 0$

Regularization —
- Perfectly separable data requires regularization
- Let weights for each class k be scaled by c as $c\tilde{\beta}_k$
- Gradient of log-loss with respect to c is always negative, causing gradient descent to grow c without bound
  Proof:
  - Log loss $= \sum_{i=1}^{n}\ln\left(1+e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}\right) - y^{(i)}c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}$
  - ∇_c log loss $= \sum_{i=1}^{n}\frac{1}{1+e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}}\times e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}\times\tilde{\beta}\cdot\boldsymbol{x}^{(i)} - y^{(i)}\tilde{\beta}\cdot\boldsymbol{x}^{(i)}$
  - $= \sum_{i=1}^{n}\tilde{\beta}\cdot\boldsymbol{x}^{(i)}\left(\frac{e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}}{1+e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}} - y^{(i)}\right)$
  - Given perfect separation:
    * If $y^{(i)} = 1$, $\tilde{\beta}\cdot\boldsymbol{x}^{(i)} > 0$, $\frac{e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}}{1+e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}} - y^{(i)} < 0$
    * If $y^{(i)} = 0$, $\tilde{\beta}\cdot\boldsymbol{x}^{(i)} < 0$, $\frac{e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}}{1+e^{c\tilde{\beta}\cdot\boldsymbol{x}^{(i)}}} - y^{(i)} > 0$
  - Thus, for all i, ∇_c log loss < 0
  - Thus gradient descent will cause c to grow without bound

20 Multinomial Logistic Regression

Description

Task — Multiclass classification

Description —
- Supervised
- Parametric

Formulation

- Probability of each class is estimated via the softmax function (generalizes the sigmoid function to multiple classes): $P(y=k|\boldsymbol{x}) = \frac{e^{f_i(x)/T}}{\sum_{j=1}^{k}e^{f_j(x)/T}} = \frac{e^{\beta_k\cdot x/T}}{\sum_{j=1}^{k}e^{\beta_j\cdot x/T}}$ where T is the temperature

and allows to smoothly interpolate between the differentiable softmax ($T = 1$) and the non-differentiable argmax ($T = 0$)
- The predicted class is the one with the highest probability: $\hat{y} = \arg\max_k P(y = k|\boldsymbol{x})$
- Geometrically, the softmax defines $k-1$ linear separating hyperplanes

Proof of softmax function:
- Take class k as reference class
- We start with log-odds:
  - $\log\left(\frac{P_y(y=i|\boldsymbol{x})}{P_y(y=k|\boldsymbol{x})}\right) = f_i(\boldsymbol{x}) - f_k(\boldsymbol{x}) = (\boldsymbol{\beta}_i - \boldsymbol{\beta}_k)\cdot\boldsymbol{x} + (\beta_{i0} - \beta_{K0})$
  - $\frac{P_y(y=i|\boldsymbol{x})}{P_y(y=k|\boldsymbol{x})} = \exp(f_i(\boldsymbol{x}) - f_k(\boldsymbol{x})) = \exp((\boldsymbol{\beta}_i - \boldsymbol{\beta}_k)\cdot\boldsymbol{x} + (\beta_{i0} - \beta_{K0}))$
  - $\sum_{i=1}^{k-1}(\frac{P_y(y=i|\boldsymbol{x})}{P_y(y=k|\boldsymbol{x})}) = \frac{1-P_y(y=k|\boldsymbol{x})}{P_y(y=k|\boldsymbol{x})} = \sum_{i=1}^{k-1}\exp(f_i(\boldsymbol{x}) - f_k(\boldsymbol{x}))$
- We can re-form last equation to posterior: $P_y(y = k \mid \boldsymbol{x}) = \frac{1}{1+\sum_{i=1}^{k-1}\exp(f_i(\boldsymbol{x})-f_k(\boldsymbol{x}))}$
- We can re-form secondlast equation to posterior:
  - $P_y(y = i \mid \boldsymbol{x}) = 1 - \frac{1}{1+\sum_{i=1}^{k-1}\exp(f_i(\boldsymbol{x})-f_k(\boldsymbol{x}))}$
  - $= \frac{\exp(f_i(\boldsymbol{x})-f_k(\boldsymbol{x}))}{1+\sum_{i=1}^{k-1}\exp(f_i(\boldsymbol{x})-f_k(\boldsymbol{x}))}$
  - $= \frac{\frac{\exp(f_i(\boldsymbol{x}))}{\exp(f_k(\boldsymbol{x}))}}{1+\sum_{i=1}^{k-1}\frac{\exp(f_i(\boldsymbol{x}))}{\exp(f_k(\boldsymbol{x}))}}$
  - $= \frac{\frac{\exp(f_i(\boldsymbol{x}))}{\exp(f_k(\boldsymbol{x}))}}{\sum_{i=1}^{k-1}\frac{\exp(f_k(\boldsymbol{x}))+\exp(f_i(\boldsymbol{x}))}{\exp(f_k(\boldsymbol{x}))}}$
  - $= \frac{\exp(f_i(\boldsymbol{x}))\times\exp(f_k(\boldsymbol{x}))}{\exp(f_k(\boldsymbol{x}))\times\sum_{i=1}^{k-1}(\exp(f_k(\boldsymbol{x}))+\exp(f_i(\boldsymbol{x})))}$
  - $= \frac{\exp(f_i(\boldsymbol{x}))}{\sum_{j=1}^{k}\exp(f_j(\boldsymbol{x}))}$

Proof that softmax $\to$ argmax if $T \to 0$:
- Assume $\boldsymbol{x} = [x_1, x_2]^\intercal$
- $\lim_{T\to 0} p(\boldsymbol{x}) = \lim_{T\to 0}\frac{e^{x_1/T}}{e^{x_1/T}+e^{x_2/T}}$
- $= \lim_{T\to 0}\frac{e^{x_1/T}e^{-x_1/T}}{(e^{x_1/T}+e^{x_2/T})e^{-x_1/T}}$
- $= \lim_{T\to 0}\frac{1}{1+e^{(x_2-x_1)/T}}$
- $\lim_{T\to 0} e^{(x_2-x_1)/T} = \begin{cases} 0 & \text{if } x_1 > x_2 \\ 1 & \text{if } x_1 = x_2 \\ \infty & \text{otherwise} \end{cases}$
- Plugging back into $\lim_{T\to 0} p(\boldsymbol{x}) = \frac{1}{1+e^{(x_2-x_1)/T}}$:

$$p(\boldsymbol{x}) = \begin{cases} [1,0]^\intercal & \text{if } x_1 > x_2 \\ [0.5,0.5]^\intercal & \text{if } x_1 = x_2 \\ [0,1]^\intercal & \text{otherwise} \end{cases}$$

Optimization

Parameters — Find parameters $\beta_1, \dots, \beta_k$

Objective function —
- Likelihood: $L(\boldsymbol{\beta}) = \prod_{i=1}^n P(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\beta}) = \prod_{i=1}^n\prod_{\ell=1}^k(\frac{e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}^{(i)}}}{\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}}})^{\delta\{y^{(i)}=\ell\}}$
- Maximize log-likelihood: $\log L(\boldsymbol{\beta}) = \sum_{i=1}^n\sum_{\ell=1}^k \delta\{y^{(i)} = \ell\}[\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}^{(i)} - \log(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}})]$
- Minimize log-loss: $-\log L(\boldsymbol{\beta})$

Optimization —
- $\frac{\partial -\log L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_k} = -\sum_{i=1}^n\sum_{\ell=1}^k\frac{\partial}{\partial\boldsymbol{\beta}_k}[\delta\{y^{(i)} = \ell\}[\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}^{(i)} - \log(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}})]] = -\sum_{i=1}^n\delta\{y^{(i)} = k\}\boldsymbol{x}^{(i)} - P(y = k|\boldsymbol{x}^{(i)})\boldsymbol{x}^{(i)}$

  Proof:
  - Derivative of the first term: $\frac{\partial}{\partial\boldsymbol{\beta}_k}(\sum_{\ell=1}^k\delta\{y^{(i)} = \ell\}\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}^{(i)}) = \delta\{y^{(i)} = k\}\boldsymbol{x}^{(i)}$
  - Derivative of the second term: $\frac{\partial}{\partial\boldsymbol{\beta}_k}(-\log(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}})) = -\frac{1}{\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}}}\frac{\partial}{\partial\boldsymbol{\beta}_k}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}}) = -\frac{e^{\boldsymbol{\beta}_k\cdot\boldsymbol{x}^{(i)}}}{\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}^{(i)}}}\boldsymbol{x}^{(i)} = -P(y = k|\boldsymbol{x}^{(i)})\boldsymbol{x}^{(i)}$
  - For reference: Softmax derivative if $\ell = k$: $\frac{\partial P(y=\ell|\boldsymbol{x})}{\partial\boldsymbol{\beta}_k}$:
    - Using the quotient rule: $\frac{\partial P(y=\ell|\boldsymbol{x})}{\partial\boldsymbol{\beta}_\ell} = \frac{\frac{\partial}{\partial\boldsymbol{\beta}_\ell}e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}}) - e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}\frac{\partial}{\partial\boldsymbol{\beta}_\ell}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}})}{(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}})^2}$
    - $\frac{\partial}{\partial\boldsymbol{\beta}_\ell}e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}} = \frac{\partial}{\partial\boldsymbol{\beta}_\ell}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}}) = e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}\boldsymbol{x}$
    - After plugging this in, we get: $\frac{\partial P_\ell}{\partial\boldsymbol{\beta}_\ell} = \frac{e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}}) - e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}\boldsymbol{x}}{(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}})^2} = P(y = \ell|\boldsymbol{x})(1 - P(y = \ell|\boldsymbol{x}))\boldsymbol{x}$
  - For reference: Softmax derivative if $\ell \neq k$: $\frac{\partial P(y=\ell|\boldsymbol{x})}{\partial\boldsymbol{\beta}_k}$:
    - Using the quotient rule: $\frac{\partial P(y=\ell|\boldsymbol{x})}{\partial\boldsymbol{\beta}_k} = \frac{\frac{\partial}{\partial\boldsymbol{\beta}_k}e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}}) - e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}\frac{\partial}{\partial\boldsymbol{\beta}_k}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}})}{(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}})^2}$
    - First term vanishes, since it does not depend on $\boldsymbol{\beta}_k$
    - $\frac{\partial}{\partial\boldsymbol{\beta}_k}(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}}) = e^{\boldsymbol{\beta}_k\cdot\boldsymbol{x}}\boldsymbol{x}$
    - After plugging this in, we get: $\frac{\partial P_\ell}{\partial\boldsymbol{\beta}_k} = \frac{-e^{\boldsymbol{\beta}_\ell\cdot\boldsymbol{x}}e^{\boldsymbol{\beta}_k\cdot\boldsymbol{x}}\boldsymbol{x}}{(\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}})^2} = -P(y = \ell|\boldsymbol{x})P(y = k|\boldsymbol{x})\boldsymbol{x}$
- If we set gradient to 0, we have expectation matching:
  - $\sum_{i=1}^n\delta\{y^{(i)} = k\}\boldsymbol{x}^{(i)} = \sum_{i=1}^n P(y = k|\boldsymbol{x}^{(i)})\boldsymbol{x}^{(i)}$
  - $\boldsymbol{x}^{(l)} = \mathbb{E}_{j\sim\text{softmax}}[\boldsymbol{x}^{(l)}]$
  - Optimum is where observed features = expected features

Characteristics —
- Convex (sum of convex functions in log-loss is convex) with psd Hessian
- Has global minimum
- Has unique or infinitely many solutions (depending on feature linear independence)
- Can be solved numerically

21 Decision Tree for Regression

Description

Task — Regression

Description —
- Supervised
- Non-parametric

Formulation
- Predictor space is split along $J-1$ internal nodes
- Thus, we get J distinct regions resp. terminal nodes $R_1, \dots, R_J$

Algorithm
- Splitting approach: Recursive binary splitting (CART):
  - Top-down: Successively splits the predictor space
  - Greedy: At each step, the best split is made
  - Steps:
    1. Select predictor x_j and threshold s such that splitting the predictor space into the regions $\{x_j \leq s\}$ and $\{x_j > s\}$ leads to the smallest MSE
    2. Repeat the process for one of the previously identified regions
    3. The process ends when the maximum depth or minimum number of observations is reached
- Prediction approach:
  - For each observation falling into R_j, the prediction is the mean or median of the response values for the training observations in R_j, denoted as $\hat{y}_{R_j}$
  - At each leaf node, 2 numbers are stored: The sum of outputs and the number of samples

Optimization

Parameters — Find regions $R_1, \dots, R_J$

Objective function —
- Minimize $\sum_{j=1}^J\sum_{i\in R_j}(y_i - \hat{y}R_j)^2$

22 Decision Tree for Classification

Description

Task — Classification

Description —
- Supervised
- Non-parametric

Formulation
- Predictor space is split along $J-1$ internal nodes
- Thus, we get J distinct regions resp. terminal nodes $R_1, \dots, R_J$

Algorithm
- Splitting approach: Recursive binary splitting (CART):
  - Top-down: Successively splits the predictor space
  - Greedy: At each step, the best split is made
  - Steps:
    1. Select predictor x_j and threshold s such that splitting the predictor space into the regions $\{x_j \leq s\}$ and $\{x_j > s\}$ leads to the smallest Gini-index or cross-entropy
    2. Repeat the process for one of the previously identified regions
    3. The process ends when the maximum depth or minimum number of observations is reached
- Prediction approach:
  - For each observation falling into R_j, the prediction is the most common response class for the training observations in R_j, denoted as $\hat{y}_{R_j}$
  - At each leaf node, k numbers are stored: Number of samples in each of the k classes

Optimization

Parameters — Find regions $R_1, \dots, R_J$

Objective function —
- Gini-index:
  - $G_j = \sum_{k=1}^K\hat{p}_{jk}(1 - \hat{p}_{jk})$ where $\hat{p}_{jk}$ is the proportion of training observations in R_j that are from class k
  - G_j takes a small value when nodes are pure, i.e., $\hat{p}_{jk}$ is 0 or 1
- Cross-entropy:
  - $D_j = -\sum_{k=1}^K\hat{p}_{jk}\log(\hat{p}_{jk})$ where $\hat{p}_{jk}$ is the proportion of training observations in R_j

- Alternatively $J(B) = -\frac{1}{h}\sum_{i=1}^{h}\sum_{k=1}^{K} y_i^{(k)} \log(\hat{p}_i^{(k)})$ where $y_i^{(k)}$ indicates whether the i^{th} training observation belongs to class k
- D_j (or $J(B)$) takes a small value when nodes are pure, i.e., $\hat{p}_{jk}$ (or $\hat{p}_i^{(k)}$ is 0 or 1
- Information gain: Reduction in cross-entropy

23 K-Means Clustering

Description

Task — Clustering

Description —
- Unsupervised
- Non-parametric

Formulation

- We specify that we want $j = 1,...,k$ clusters in total
- Clusters defined by centroid $\mu^{[j]} \in \mathbb{R}^m$
- Instances $i = 1,...,n$ given by $x^{(i)}$
- Each instance i has k indicator variables, which describe whether instance i is assigned to cluster j, given by $\{p^{i[j]}\}_{j=1}^{k} \in [0,1]$

Optimization

Parameters — Find centroids $\mu^{[j]}$, i.e. find cluster assignments (instance always assigned to closest cluster)

Objective function —
- Minimize distance between each instance and the centroid of its closest cluster: $j^* = \arg\min_j \|x^{(i)} - \mu^{[j]}\|^2$
- Distortion function given by:
  $\Theta = \sum_{i=1}^{n}\sum_{j=1}^{k} p^{i[j]}\|x^{(i)} - \mu^{[j]}\|^2 = \sum_{x^{(i)}\in C_j}\sum_{j=1}^{k}\|x^{(i)} - \mu^{[j]}\|^2$ with
  $p^{i[j]} = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases}$ and $C_j = \left\{x^{(i)} \mid p^{i[j]} = 1\right\}$
- If we choose Euclidian distance: $\|x^{(i)} - \mu^{[j]}\| = \sqrt{\sum_{l=1}^{m}\left(x_l^{(i)} - \mu_l^{[j]}\right)^2}$

Optimization — Lloyd's algorithm:

1. Randomly initialize each $\mu^{[j]}$
2. E-Step:
   - Re-assign instances while keeping all centroids fixed, i.e. minimize Θ with respect to $p^{i[j]}$
3. M-Step:
   - Re-compute centroids while keeping all instance assignments fixed, i.e. minimize Θ with respect to $\mu^{[j]}$
   - $\nabla_{\mu^{[j]}}\Theta = 2\sum_{i=1}^{n} p^{i[j]}(\mu^{[j]} - x^{(i)}) = 0$
   - Then, $\mu^{[j]} = \frac{\sum_{i=1}^{n} p^{i[j]} x^{(i)}}{\sum_{i=1}^{n} p^{i[j]}} = \frac{1}{|C_j|}\sum_{x^{(i)}\in C_j} x^{(i)}$
   - Θ is strictly convex, thus, we find a global minimum and a unique solution here
4. Repeat E- and M-Step until convergence

Proof of convergence 1:
- $\Theta_t = \sum_{i=1}^{n}\sum_{j=1}^{k} p_t^{i[j]}\|x^{(i)} - \mu_t^{[j]}\|^2$
- In M-Step: $\Theta_t \le \sum_{i=1}^{n}\sum_{j=1}^{k} p_t^{i[j]}\|x^{(i)} - \mu_{t-1}^{[j]}\|^2$
- In E-Step:
  $\sum_{i=1}^{n}\sum_{j=1}^{k} p_t^{i[j]}\|x^{(i)} - \mu_{t-1}^{[j]}\|^2 \le \sum_{i=1}^{n}\sum_{j=1}^{k} p_{t-1}^{i[j]}\|x^{(i)} - \mu_{t-1}^{[j]}\|^2 = \Theta_{t-1}$
- Thus, $\Theta_t \le \Theta_{t-1}$

Proof of convergence 2:
- Let Assignment function $c(i) \to j$ be defined as:
  $c(i) = \arg\min_{j\in\{1,...,k\}}\|x^{(i)} - \mu^{[j]}\|^2$
- Distortion can be rewritten as $\Theta = \sum_i \|x^{(i)} - \mu^{c(i)}\|^2$
- In E-Step, the centroids are fixed and we recalculate the assignment function $c(i)$. Thus, in E-Step Θ decreases unless all assignments $c(i)$ remain unchanged
- In M-Step, the clusters C_j are fixed and we recalculate $\mu^{[j]}$ to minimize: $\sum_{x^{(i)}\in C_j}\|x^{(i)} - \mu^{[j]}\|^2$
- $\mu^{[j]} = \frac{1}{|C_j|}\sum_{x^{(i)}\in C_j} x^{(i)}$ minimizes this cost:
  - For any vector $\mu^{[j]'}$, we can write the cost as:
    $\frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}\|x^{(i)} - \mu^{[j]'}\|^2 =$
    $\frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}\|(x^{(i)} - \mu^{[j]}) + (\mu^{[j]} - \mu^{[j]'})\|^2 = \frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}\|x^{(i)} - \mu^{[j]}\|^2 + \|\mu^{[j]} - \mu^{[j]'}\|^2 + 2(\mu^{[j]} - \mu^{[j]'})^\top\frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}(x^{(i)} - \mu^{[j]})$
  - The third term vanishes since $\frac{1}{|C_j|}\sum_{x^{(i)}\in C_j} x^{(i)} = \mu^{[j]}$
  - Then, we have: $= \frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}\|x^{(i)} - \mu^{[j]}\|^2 + \|\mu^{[j]} - \mu^{[j]'}\|^2$
  - We can see that:
    $\frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}\|x^{(i)} - \mu^{[j]}\|^2 + \|\mu^{[j]} - \mu^{[j]'}\|^2 \ge \frac{1}{|C_j|}\sum_{x^{(i)}\in C_j}\|x^{(i)} - \mu^{[j]}\|^2$
    with equality if $\mu^{[j]'} = \mu^{[j]}$

Characteristics —
- Challenges:
  - Has no way representing size of shape of a cluster, all that matters is distance
  - Hard cluster assignment, rather than soft cluster assignment, which implies that all instances in a cluster have an equal vote within that cluster and no votes in any other cluster with regards to the assignment of a new point
- Not convex
- Has local minimum
- Has unique or infinitely many solutions
- Can be solved numerically

24 Principal Component Analysis (PCA)

Maximize Variance Approach

Description

Task — Dimensionality reduction via projection, create uncorrelated features

Description —
- Unsupervised
- Non-parametric

Overview — Identifies lower-dimensional subspace and projects data onto it such that the maximum amount of variance in the data is preserved. In lower-dimensional subspace:
- Axes are called *principal components*, where the first principal component is the axis accounting for the largest variance
- Each axis is given by an eigenvector with *loadings*, indicating how much each variable in the original data contributes to this eigenvector
- Variance captured along each axis is given by the corresponding eigenvalue

Formulation

- Project data $\{x^{(i)}\}_{i=1}^{n} \in \mathbb{R}^m$ onto space $\mathbb{R}^d$ spanned by orthonormal basis $\{u^{[j]}\}_{j=1}^{d} \in \mathbb{R}^m$ where $d << m$
- Each instance $x^{(i)}$ is projected onto each basis vectors $u^{[j]} \cdot x^{(i)}$:
  $x^{(i)} \to [u^{[1]} \cdot x^{(i)},..., u^{[d]} \cdot x^{(i)}]^\top$
- Each basis vector $u^{[j]}$ contains m loadings $[u_i^{[j]},..., u_m^{[j]}]$, whose value indicates how important each feature m is for the j^{th} principal component
- Mean of projected data for a given basis vector:
  $u^{[j]} \cdot \bar{x} = u^{[j]} \cdot \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$
- Variance of projected data for a given basis vector:
  $\frac{1}{n}\sum_{i=1}^{n}(u^{[j]} \cdot x^{(i)} - u^{[j]} \cdot \bar{x})^2 = u^{[j]\top} S u^{[j]}$ where
  $S = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^\top = \frac{1}{n}X^\top X$ is the covariance matrix of the data in the orthogonal complement to the subspace spanned by the first $j-1$ principal components, i.e.
  $X_{j-1} = \{x^{(i)} - \text{projection}_{u_{\le j-1}}\} = \{x^{(i)} - \sum_{l=1}^{j-1}(u^{[j]} \cdot x^{(i)}) \cdot u^{[j]}\}$

Optimization

Parameters — Find principal components $\{u^{[j]}\}_{j=1}^{d}$

Objective function —
- For the j^{th} principal component, maximize variance $\sum_{j=1}^{d} u^{[j]\top} S u^{[j]}$ subject to orthonormal $\{u^{[j]}\}_{j=1}^{d}$
- Gives rise to Lagrangian formulation
- Lagrangian formulation for $u^{[1]}$ capturing the most variance:
  $\mathcal{L} = u^{[1]\top} S u^{[1]} - \lambda^{[1]}(u^{[1]} \cdot u^{[1]} - 1)$ where $\lambda^{[1]}$ captures the orthonormality constraint that $u^{[1]} \cdot u^{[1]} = 1$
- Lagrangian formulation for $u^{[2]}$ capturing the secondmost variance:
  $\mathcal{L} = u^{[2]\top} S u^{[2]} - \lambda^{[2]}(u^{[2]} \cdot u^{[2]} - 1) - \lambda^{[1][2]}(u^{[1]} \cdot u^{[2]} - 0)$ where $\lambda^{[1][2]}$ captures the orthogonality constraint that $u^{[1]} \cdot u^{[2]} = 0$

Optimization — For $u^{[1]}$, based on $X = X_0$:
- $\nabla_{u^{[1]}}\mathcal{L} = 2Su^{[1]} - 2\lambda^{[1]}u^{[1]} = 0$
- $\Rightarrow Su^{[1]} = \lambda^{[1]}u^{[1]}$
- This is the eigenvector/eigenvalue equation, so $u^{[1]}$ is the eigenvector of S and $\lambda^{[1]}$ is the associated eigenvalue
- We see that the variance of the projected data is equal to $\lambda^{[1]}$:
  $u^{[1]\top} S u^{[1]} = u^{[1]\top}\lambda^{[1]}u^{[1]} = \lambda^{[1]}u^{[1]\top}u^{[1]} = \lambda^{[1]} \times 1$

For $u^{[2]}$, based on $X = X_1$:
- $\nabla_{u^{[2]}}\mathcal{L} = 2Su^{[2]} - 2\lambda^{[2]}u^{[2]} - \lambda^{[1][2]}u^{[1]} = 0$
- $\Rightarrow Su^{[2]} = \lambda^{[2]}u^{[2]}$

  Proof:
  - Multiplying with $u^{[1]\top}$:
    $2u^{[1]\top}Su^{[2]} - 2\lambda^{[2]}u^{[1]\top}u^{[2]} - \lambda^{[1][2]}u^{[1]\top}u^{[1]} = 0$
  - $= 2u^{[1]\top}Su^{[2]} - 0 - \lambda^{[1][2]} \times 1 = 0$ because of orthogonality resp. orthonormality
  - $= 2u^{[2]\top}Su^{[1]} - \lambda^{[1][2]} = 0$ because the variance is a scalar and can be transposed and because the covariance matrix is symmetric
  - $= 2u^{[2]\top}\lambda^{[1]}u^{[1]} - \lambda^{[1][2]} = 0$ after plugging in the first found basis vector
  - $= 2\lambda^{[1]} \times 0 - \lambda^{[1][2]} = 0$
  - $= \lambda^{[1][2]} = 0$

... continue as for previous vector

In the end, we have a total projected variance of $\sum_{j=1}^{d}\lambda^{[j]}$

Characteristics —
- Convex
- Has global minimum
- Has unique or infinitely many solutions
- Can be solved analytically

SVD Approach

Formulation

- Project data $\{x^{(i)}\}_{i=1}^n \in \mathbb{R}^m$ onto space $\mathbb{R}^d$ spanned by orthonormal basis $\{u^{[j]}\}_{j=1}^d \in \mathbb{R}^m$ (subset of $\{u^{[j]}\}_{j=1}^m \in \mathbb{R}^m$) where $d \ll m$
- Given basis, we can represent original datapoint as: $x^{(i)} = \sum_{j=1}^m (x^{(i)} \cdot u^{[j]}) u^{[j]}$
- We can represent projected datapoint as: $\tilde{x}^{(i)} = BB^T x^{(i)} = B^T x^{(i)}$ since B is orthogonal or, equivalently $\tilde{x}^{(i)} = \sum_{j=1}^d \alpha_{ij} u^{[j]} + \sum_{j=d+1}^m \gamma_j u^{[j]}$ where α_{ij} is specific to the instance and γ_j is generic and maps up to the subspace

Optimization

Objective function —

- Reconstruction error: $J = \frac{1}{n}\sum_{i=1}^n \|x^{(i)} - \tilde{x}^{(i)}\|^2 = \frac{1}{n}\sum_{i=1}^n \|x^{(i)}\|^2 - \|\tilde{x}^{(i)}\|^2 = \frac{1}{n}\sum_{i=1}^n \|x^{(i)}\|^2 - \|B^T x^{(i)}\|^2$
- We can show that $\alpha_{ij} = x^{(i)} \cdot u^{[j]}$ and $\gamma_j = \overline{x} \cdot u^{[j]}$
  Proof:
  - Take derivative of reconstruction error J and set it to 0
- If $m = d-1$, $J = u^{[d]\intercal} S u^{[d]}$
  Proof:
  - For $d = m-1$, $\tilde{x}^{(i)} = \sum_{j=1}^d \alpha_{ij} u^{[j]}$
  - Substituting $\tilde{x}^{(i)}$ and $x^{(i)}$ in J, we get: $J = \frac{1}{n}\sum_{i=1}^n \|x^{(i)} - \tilde{x}^{(i)}\|^2 = \frac{1}{n}\sum_{i=1}^n \|\sum_{j=1}^m (x^{(i)} \cdot u^{[j]}) u^{[j]} - \sum_{j=1}^{m-1} \alpha_{ij} u^{[j]}\|^2 = \frac{1}{n}\sum_{i=1}^n \|\sum_{j=1}^m (x^{(i)} \cdot u^{[j]}) u^{[j]} - \sum_{j=1}^{m-1} (x^{(i)} \cdot u^{[j]}) u^{[j]}\|^2 = \frac{1}{n}\sum_{i=1}^n \|(x^{(i)} \cdot u^{[D]}) u^{[D]}\|^2$
  - Using the orthonormality of $u^{[D]}$, we simplify: $\|(x^{(i)} \cdot u^{[D]}) u^{[D]}\|^2 = (x^{(i)} \cdot u^{[D]})^2$
  - Thus: $J = \frac{1}{n}\sum_{i=1}^n (x^{(i)} \cdot u^{[D]})^2$
  - To relate this to the covariance matrix $S = \frac{1}{n}\sum_{i=1}^n \left(x^{(i)} - \overline{x}\right)\left(x^{(i)} - \overline{x}\right)^T$,, we transform the projection to use centered data: $J = \frac{1}{n}\sum_{i=1}^n ((x^{(i)} - \overline{x}) \cdot u^{[D]})^2$
  - Expanding the squared projection: $J = \frac{1}{n}\sum_{i=1}^n ((x^{(i)} - \overline{x}) \cdot u^{[D]})^\intercal ((x^{(i)} - \overline{x}) \cdot u^{[D]}) = u^{[D]\intercal} (x^{(i)} - \overline{x})(x^{(i)} - \overline{x})^\intercal u^{[D]} = u^{[D]\intercal} S u^{[D]}$
  - Here, $u^{[D]}$ is the eigenvector of S with the smallest eigenvalue
- $x^{(i)}$ is a column of $X^\intercal$
- If $A = X^\intercal$ and SVD of A is $USV^\intercal$, then B is given by $U^{(j \le d)}$, i.e. the first d columns of U
- Then, reconstruction error is given by: $J = \frac{1}{n}\sum_{i=1}^n \|x^{(i)}\|^2 - \|B^T x^{(i)}\|^2 = \sum_{i=1}^n \|x^{(i)}\|^2 - \frac{1}{d}\sum_{i=1}^d \sigma_d^2$ given SVD Projection Energy

25 Neural Networks

Formulation

Formulation —

- Model architecture:
  - Input features (X) > weights (B) > weighted sums (S) > activation functions (φ) > hidden states (H) > weights > ... > hidden states > activation function > outputs
  - E.g.
    - Outputs: Probability $p(y \mid x)$ for each class y
    - Activation: Softmax ensures all P add to 1: $p(y \mid x) = \frac{\exp\left(h_y^{(K)}\right)}{\sum_{y'} \exp\left(h_{y'}^{(K)}\right)}$
    - Hidden layer: $h^{(K)} = \sigma(w^{(K)} h^{(K-1)})$ ... $h^{(1)} = \sigma(w^{(1)} e(x))$
    - Concatenated vector of word embeddings: $e(x) = \frac{1}{n}\sum_{w_i} e(w_i)$
    - Transformation: word embedding $e(w_i)$
    - Inputs: n words
  - Can be seen as a composition of:
    - Encoders that construct disentangled and robust latent representation from input, which maximizes mutual information between representation and input, as according to infomax principle
    - Linear estimators
- Neuron (j) in layer $[k]$ given training instance $x^{(i)[0]}$ resp. instance from previous layer $h^{(i)[k-1]}$ given by: $h^{(j)[1]} = \varphi(x^{(i)[0]} \cdot \beta^{(j)[1]})$ resp. $h^{(j)[k]} = \varphi(h^{(i)[k-1]} \cdot \beta^{(j)[k]})$
- Outputs for neurons $1, ..., j$ in fixed layer (notation for layer omitted below) given by: $H = \varphi(XB) = \varphi(S)$ where
  - $X \in \mathbb{R}^{n \times m+1}$ (incl. bias term)
  - $B \in \mathbb{R}^{m+1 \times j}$ with a weight vector for each neuron in each column (incl. bias term)
  - $S \in \mathbb{R}^{n \times j}$ with the weighted sum (prior to activation) for instance i in neuron j is on the i^{th} row and j^{th} column
  - The activation function differs by neuron

Activation functions —

- Introduce non-linearities
- Can differ by neuron
- Sigmoid:
  - $[0, 1]$
  - $\varphi(z) = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$
  - $\varphi'(z) = \frac{e^{-z}}{(1+e^{-z})^2}$ with maximum at 0.25
- Hyperbolic Tangent:
  - $[-1, 1]$
  - $\varphi(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{1-e^{-2z}}{1+e^{-2z}}$
  - $\varphi'(z) = 1 - tanh(z)^2$
- ReLu:
  - $\varphi(z) = max(0, z)$
  - $\varphi'(z) = 1$ if $z > 0$; 0 otherwise

Optimization

Parameters — Find parameters $\theta = B$

Objective function —

- Minimize standard objectives, e.g. MSE

Optimization —

- Perform *forward pass* with randomly initialized parameters, to calculate loss
- Perform *backpropagation*, to calculate gradient:
  - $\frac{\partial L}{\partial \theta} = [\frac{\partial L}{\partial B^{[0]}}, ..., \frac{\partial L}{\partial B^{[output]}}]$
  - $\frac{\partial L}{\partial B^{[k]}} = \frac{\partial L}{\partial H^{[l]}} \frac{\partial H^{[l]}}{\partial B^{[k]}} = C$
    - When $l > k+1$, i.e. when going several layers back: $\frac{\partial L}{\partial B^{[k]}} = \frac{\partial L}{\partial H^{[l]}} \frac{\partial H^{[l]}}{\partial S^{[l-1]}} \frac{\partial S^{[l-1]}}{\partial H^{[l-1]}} \frac{\partial H^{[l-1]}}{\partial B^{[k]}}$
    - When $l = k+1$, i.e. when going one layer back: $\frac{\partial L}{\partial B^{[k]}} = \frac{\partial L}{\partial H^{[k+1]}} \frac{\partial H^{[k+1]}}{\partial S^{[k]}} \frac{\partial S^{[k]}}{\partial B^{[k]}}$
- Perform gradient descent to find best weights

Challenges —

- Unstable gradients:
  - Can happen since backpropagation computes gradients using the chain rule, meaning many gradients are multiplied across many layers
  - Caused by poor choice of activation, typically sigmoid or tanh with high absolute input values
  - Caused when weights are shared across many layers, especially in RNNs
  - Exploding gradients: If gradients are > 1, gradients grow bigger and bigger during backpropagation, algorithm diverges
  - Vanishing gradients: If gradients are < 1 (resp. parameter θ is < 1), gradients approach 0 during backpropagation, algorithm fails to converge
    Proof:
    - $h_m = \sigma(\theta h_{m-1} + x_m)$
    - $\frac{\partial h_{m+k}}{\partial h_m} = \prod_{i=0}^{k-1} \frac{\partial h_{m+k-i}}{\partial h_{m+k-i-1}} = \prod_{i=1}^{k} \theta \times \sigma'(\theta h_{m+k-i-1} + x_{m+k-i})$
    - $\le \prod_{i=1}^{k} \theta \times 0.25 = \theta^k \times 0.25^k$ since derivative of sigmoid has 0.25 as maximum value
    - $\to 0$ as $k \to \infty$, since $\theta < 1$
  - Solution:
    - Use fewer layers
    - Use ReLU activation function
    - Use residual networks (ResNet)
    - Use LSTM or GRU units
    - Glorot or He initialization: Connection weights of each layer are initialized randomly
    - Batch normalization
    - Gradient clipping: Set maximum threshold for gradients during backpropagation
- Dying ReLUs:
  - Caused when weights are tweaked such that a neuron becomes negative, causing ReLU activation to output 0
  - Can happen if β_0 in $x^T w + \beta_0$ is large and negative
  - Dead neuron cannot be brought back:
    - Let $z = x^T w + b^{[l]}$.
    - $\text{ReLU}\left(b^{[l]}\right) = 0$ if $b^{[l]} \le 0$
    - Then $\frac{\partial l}{\partial z} = 0$, $\frac{\partial z}{\partial h} = 0$, $\frac{\partial h}{\partial b^{[l]}} = 0$
    - $h_{b^{[l]}}$ is guaranteed to be zero for all inputs if h is dead
    - Then, parameters cannot change and h will remain dead
  - Solution: Leaky ReLU, ELU, scaled ELU

26 Backpropagation

Big Picture

- In ANNs, we learn both f and θ, meaning that the objective function is not convex
- ANN objective: $\arg\min_\theta L(\theta) = \arg\min_\theta \sum_{(x,y)\in\mathcal{D}} \ell(f(x;\theta), y)$
- Objective is optimized via gradient descent
- Gradient descent update rule: $\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta L(\theta)\big|_{\theta=\theta_t}$
- Backpropagation uses the chain rule in combination with dynamic programming techniques to compute the gradient $\nabla_\theta L(\theta)$

Point of Departure

Composite function — Ordered series of equations (*primitives*), where each equation is a function only of the preceding equations
For example: $f(x,z) = x^2 + 3z$ $a = x^2$, $b = 3z$, $c = a + b$, $f(x,z) = c$

Computation graph $\mathcal{G}$ — Graphical representation of a composite function

- Directed acyclic hypergraph (E, V) where V is the set of vertices

(*nodes*) representing variables and E is the set of edges representing functions
  - Directed: One direction
  - Acyclic: No cycles
  - An edge can connect any number of vertices, not just two
- Edge from $v' \to v$: labeled with a differentiable function f_v
- Vertex v with outgoing edges: argument to f_v
- Vertex v with incoming edges: result of f_v
- If we have n input nodes and $|E| = M$ edges, we have $|V| = M + n$ total nodes

Bauer's Formula — Extension of partial derivative

- For two nodes i and j in $\mathcal{G}$, let the Bauer path $P(i,j)$ define the set of all directed paths starting at j and ending at i
- Let z_i and z_j represent these nodes as variables
- The partial derivative of z_i with respect to z_j is: $\frac{\partial z_i}{\partial z_j} = \sum_{p \in P(j,i)} \prod_{(k,l) \in p} \frac{\partial z_l}{\partial z_k}$ where we sum over all paths and take the product over all nodes on a given path
  Proof:
  - Base case: $i = j$, i.e., $z_i = z_j$: $\frac{\partial z_i}{\partial z_j} = 1$
  - Inductive hypothesis: Bauer's formula holds for all $j \le i$
  - Inductive step:
    * Let $m < j$ (i.e., z_m comes before z_j in topological order, and $j \in \text{out}(m)$)
    * $\frac{\partial z_i}{\partial z_m} = \sum_{j \in \text{out}(m)} \frac{\partial z_i}{\partial z_j} \frac{\partial z_j}{\partial z_m}$ by the chain rule
    * $\frac{\partial z_i}{\partial z_m} = \sum_{j \in \text{out}(m)} (\sum_{p \in P(j,i)} \prod_{(k,l) \in p} \frac{\partial z_l}{\partial z_k}) \frac{\partial z_j}{\partial z_m}$ due to inductive hypothesis
    * $\frac{\partial z_i}{\partial z_m} = \sum_{j \in \text{out}(m)} (\sum_{p \in P(j,i)} \frac{\partial z_j}{\partial z_m} \prod_{(k,l) \in p} \frac{\partial z_l}{\partial z_k})$ due to distributivity
    * $\frac{\partial z_i}{\partial z_m} = \sum_{p \in P(m,i)} \prod_{(k,l) \in p} \frac{\partial z_l}{\partial z_k}$ by concatenating paths
- Challenge of naively calculating this: With $\sum_{p \in P(j,i)}$, we are summing over an exponential number of paths, leading to a runtime complexity of $O(|P(j,i)|) = O(2^{|E|})$

Forward Propagation

Initialize values of input nodes, and on that basis, calculate values of all descendant nodes
Algorithm $(f, \boldsymbol{x})$:

1. Initialize node values: $z_i = \begin{cases} x_i & \text{if } i \le n \quad \text{(n input nodes initialized to value)} \\ 0 & \text{if } i > n \quad \text{(non-input nodes initialized to 0)} \end{cases}$ For $i = n+1, \dots, M$ non-input nodes: $z_i = g_i\left(\langle z_{\text{pa}(i)} \rangle\right)$ where g_i denotes the primitive at edge i and $\langle z_{\text{pa}(i)} \rangle$ denotes the ordered set of parent nodes of z_i
2. Return $\{z_1, z_2, \dots, z_M\}$

Properties:

- Runtime complexity: $O(|E|)$, i.e., linear in the number of edges
- Space complexity: $O(|V|)$, i.e., linear in the number of vertices

Backpropagation

After forward propagation, compute the gradient of f with respect to input nodes
Algorithm:

1. Perform forward propagation: $z \leftarrow \text{forward propagate}(f, \boldsymbol{x})$
2. Initialize: $\frac{\partial f}{\partial z_i} = \begin{cases} 1 & \text{if } i = M \\ & \text{(initialize gradient of } f \text{ with respect to} \\ & \text{output node as 1, since } \partial f / \partial f = 1) \\ 0 & \text{otherwise} \end{cases}$
3. For $i = M-1, \dots, 1$:
   $\frac{\partial f}{\partial z_i} = \sum_{j : i \in \text{Pa}(i)} \frac{\partial f}{\partial z_j} \frac{\partial z_j}{\partial z_i} = \sum_{j : i \in \text{Pa}(i)} \frac{\partial f}{\partial z_j} \frac{g_j\left(\langle z_{\text{pa}(j)} \rangle\right)}{\partial z_i}$
4. Return $\left[\frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2}, \dots, \frac{\partial f}{\partial z_M}\right]$

Properties:

- This is a dynamic program
- Presents improvement over naive Bauer's formula: partial derivatives that appear on multiple paths are memoized
- Runtime complexity: $O(|E|)$, i.e., the same as forward propagation (which computes f)
- Space complexity: $O(|V|)$, i.e., the same as forward propagation (which computes f)
- *Cheap gradient principle*: Calculating the gradient has the same complexity as evaluating the function

Extension of backpropagation to k^th order derivatives:

- Backpropagation on a graph with $|E|$ edges, for inputs $\boldsymbol{x} = (x_1, \dots, x_n)^\top$, for the k^{th} order derivative has runtime $\mathcal{O}(|E| n^{k-1})$
  Proof:
  - For second-order derivative:
    $\nabla_2 f(\boldsymbol{x})\text{(i.e. Hessian)} = \begin{bmatrix} \nabla(\boldsymbol{e}_1^\top \nabla f(\boldsymbol{x})\text{(i.e. Jabobian))} \\ \dots \\ \nabla(\boldsymbol{e}_n^\top \nabla f(\boldsymbol{x})) \end{bmatrix}$ because $\boldsymbol{e}_k^\top \boldsymbol{A}$ returns k^{th} row of $\boldsymbol{A}$
  - For third-order derivative, similar principle
  - We first differentiate in M edges (∇_1), which means complexity is $1 \times M$
  - We then differentiate by n variables in M edges (∇_2), which means complexity is $n \times M$
  - We then differentiate these n variables by another n variables in M edges (∇_3), which means complexity is $n^2 \times M$
  - ...

Requirements for backpropagation:

- Weights need to be initialized to different values: If they are initialized to the same constant, each neuron produces the same output (since the constant weight is applied to the input), and then during backpropagation, all neurons receive the same gradient updates
  - Weights initialized to 0:
    * $\boldsymbol{h} = \varphi\left(\boldsymbol{x}\boldsymbol{B}^{[1]}\right) = 0$ where $\boldsymbol{h}$ corresponds to z, φ is the primitive, and $\boldsymbol{x}$ corresponds to the input nodes
    * $\boldsymbol{y} = \boldsymbol{h}\boldsymbol{B}^{[2]} = 0$ where $\boldsymbol{y}$ corresponds to z'
    * $\frac{\partial L}{\partial \boldsymbol{B}^{[2]}} = \frac{\partial L}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{B}^{[2]}} = \frac{\partial L}{\partial \boldsymbol{y}} \boldsymbol{h} = 0$, i.e. $\boldsymbol{B}^{[2]}$ is not updated
    * $\frac{\partial L}{\partial \boldsymbol{B}^{[1]}} = \frac{\partial L}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{B}^{[1]}} = \frac{\partial L}{\partial \boldsymbol{y}} \boldsymbol{B}^{[2]} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{B}^{[1]}} = 0$, i.e. $\boldsymbol{B}^{[1]}$ is not updated
    * Thus, weights will always remain 0 and network will not learn
  - Weights initialized to same constant:
    * $\boldsymbol{B}^{[1]} = \boldsymbol{B}^{[2]}$
    * $\boldsymbol{h}_1 = \boldsymbol{h}_2$
    * $\frac{\partial L}{\partial \boldsymbol{B}^{[1]}} = \frac{\partial L}{\partial \boldsymbol{B}^{[2]}}$
    * Thus, weights will always receive same updates, will remain equal, and network will not learn
- At least one activation must be non-linear so that there is a non-zero gradient

27 Bayesian Neural Networks

Setting

- In Bayesian setting, normalization constant is computationally intractable

Formulation

Since original setting is computationally intractable, we can turn to *variational inference*:

- Variational inference approximates true posterior $p(w|D)$ by simpler, parametrized distribution $q(w|\theta)$
- We assume $q(w|\theta) \sim \mathcal{N}(\mu, \sigma^2 I)$ with $\theta = (\mu, \sigma)$

Optimization

Parameters — Find parameters θ
Objective function —

- Minimize KL divergence: $\theta^* = \arg\min_\theta KL[q(w|\theta) \| p(w|D)] = \arg\min_\theta \mathbb{E}_{w \sim q} log(q(w|\theta)) - \mathbb{E}_{w \sim q} log(p(D|w)) - \mathbb{E}_{w \sim q} log(p(w))$
  Proof:
  - $\arg\min_\theta KL[q(w|\theta) | p(w|D)] = \arg\min_\theta \mathbb{E}_{w \sim q}[log(\frac{q(w|\theta)}{p(w|D)})] =$
    $\arg\min_\theta \mathbb{E}_{w \sim q}[log(q(w|\theta))] - \mathbb{E}_{w \sim q}[log(p(w|D))] =$
    $\arg\min_\theta \mathbb{E}_{w \sim q}[log(q(w|\theta))] - \mathbb{E}_{w \sim q}[\frac{p(D|w) \times p(w)}{p(D)}] =$
    $\arg\min_\theta \mathbb{E}_{w \sim q} log(q(w|\theta)) - \mathbb{E}_{w \sim q} log(p(D|w)) - \mathbb{E}_{w \sim q} log(p(w)) + \mathbb{E}_{w \sim q} log(p(D)) = \arg\min_\theta \mathbb{E}_{w \sim q} log(q(w|\theta)) - \mathbb{E}_{w \sim q} log(p(D|w)) - \mathbb{E}_{w \sim q} log(p(w)) + \text{const.}$

Optimization —

- To calculate gradient, we can leverage the *reparametrization trick*
- $\frac{\partial}{\partial \theta} \mathbb{E}_{w \sim q}[log(q(w|\theta)) - log(p(D|w)) - log(p(w))] = \frac{\partial}{\partial \theta} \mathbb{E}_{w \sim q}[F(w, \theta)]$ can be reparametrized to:
  - $\frac{\partial}{\partial \mu} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[\frac{\partial}{\partial w} F(w, \theta) + \frac{\partial}{\partial \mu} F(w, \theta)]$
  - $\frac{\partial}{\partial \sigma} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[\epsilon^\top \frac{\partial}{\partial w} F(w, \theta) + \frac{\partial}{\partial \sigma} F(w, \theta)]$
- To optimize this, we can use gradient descent with the following algorithm:
  1. Initialize μ and σ
  2. For $t = 1, 2, \dots$
     (a) Sample $\epsilon \sim \mathcal{N}(0, I)$
     (b) Compute $F(w, \theta)$
     (c) $\mu_{t+1} \leftarrow \mu_t - \eta_t [\frac{\partial}{\partial w} F(w, \theta) + \frac{\partial}{\partial \mu} F(w, \theta)]|_{\mu = \mu_t}$
     (d) $\sigma_{t+1} \leftarrow \sigma_t - \eta_t [\epsilon^\top \frac{\partial}{\partial w} F(w, \theta) + \frac{\partial}{\partial \sigma} F(w, \theta)]|_{\sigma = \sigma_t}$

28 Convolutional Neural Networks (CNNs)

Formulation

Formulation —

- Model architecture:
  - Input: Composed of channels (e.g. R with 3 channels)
  - Convolutional layer: Composed of feature maps
  - Channel: Sublayer in input and output, composed of pixels
  - Feature map: Sublayer in convolutional layer, composed of neurons, each neuron is generated by applying filter to all receptive fields across all sublayers in lower layer, weights and biases shared across all neurons in feature map
  - Receptive field: Group of neurons in lower layer, that single neuron in higher layer is connected to, size $f_h \times f_w$
  - Filter resp. convolutional kernel: Weights applied to all receptive fields across all sublayers in lower layer, size $K \times K$
  - Zero padding: Padding applied to retain same dimensions in each layer, size $\frac{f_h - 1}{2}$ resp. $\frac{f_w - 1}{2}$

  - Stride: By how many neurons receptive field shifts, size $s_h \times s_w$, if stride > 1, spatial dimensions in subsequent layer decrease (convolution), if stride < 1, spatial dimensions increase (deconvolution)
- Output of neuron in layer n, given previous layer $n-1$: $z_{i,j,k} = b_k + \sum_{f_n} \sum_{f_w} \sum_{f_n'} x_{i',j',k'} \cdot w_{u,v,k',k}$, i.e. sum of element-wise matrix product over all receptive fields and all feature maps, where
  - $z_{i,j,k}$ is the output of neuron in row i and column j on feature map k in layer n
  - f_n and f_w are dimensions of the receptive field in layer $n-1$
  - f_n' is the number of feature maps in layer $n-1$
  - $x_{i',j',k'}$ is the output of neuron in row i' and column j' on feature map k' in layer $n-1$
  - $i' = i \times \text{stride}_h + u - \text{padding}_h$ and $j' = j \times \text{stride}_w + v - \text{padding}_w$
  - $w_{u,v,k',k}$ is the connection weight between any neuron on feature map k in layer n and its input at u,v on feature map k'
  - $u, v \in \Delta_K$ are possible shifts allowed by kernel
- Output of neurons in layer n, given previous layer $n-1$: $z_k = b_k + \sum_{f_n} \sum_{f_w} \sum_{f_n'} W_{k',k} X_{k'}$
- Output size in layer n, given previous layer $n-1$: $H' = \frac{H+2p-K}{\text{stride}_h} + 1$ and $W' = \frac{W+2p-K}{\text{stride}_w} + 1$

Optimization

Parameters — Find parameters $\theta = W$

Objective function —

- Minimize standard objectives, e.g. MSE

Optimization —

- Perform *forward pass* with randomly initialized parameters, to calculate loss
- Perform *backpropagation*, to calculate gradient
- Perform gradient descent to find best weights

29 Recurrent Neural Networks (RNN)

Description

Description —

- Can deal with sequential data and the persistence of information over time
- Can be seq-to-seq, seq-to-vec, or vec-to-seq
- Can be unidirectional or bidirectional
- Challenge: Cannot preserve long-term dependencies well. Solution: LSTM

Formulation

Formulation —

- Model architecture:
  - Input layer
  - Hidden layer resp. memory cell: Cell state h_t, cell output y_t
  - Output layer
- Output of neuron in layer n:

$$Y_t = \phi\left(X_t W_x + H_{t-1} W_y + b\right) V = \phi\left(\begin{bmatrix} X_t \\ H_{t-1} \end{bmatrix} W + b\right) V$$

  - V is optional: If $V = I$ (identity matrix), $H_t = Y_t$, this is assumed in the following
  - Y_t is an $n_{\text{instances}} \times n_{\text{neurons}}$ matrix containing layer outputs for instances in the mini-batch at time t
  - X_t is an $n_{\text{instances}} \times m$ matrix containing encoded inputs for all instances in the mini-batch
  - H_{t-1} is an $n_{\text{instances}} \times n_{\text{neurons}}$ matrix containing cell state outputs for instances in the mini-batch at time $t-1$
  - W_x is an $m \times n_{\text{neurons}}$ matrix containing connection weights for X_t
  - W_y is an $n_{\text{neurons}} \times n_{\text{neurons}}$ matrix containing connection weights for Y_{t-1} (resp. H_{t-1})
  - b is a vector of length n_{neurons} containing the bias term
  - $W = \begin{bmatrix} W_x \\ W_y \end{bmatrix}$
  - ϕ is a non-linear activation function

Optimization

Parameters — Find parameters $\theta = W_x, W_y, b, X_t$, which are shared across time steps

Objective function —

- Maximize log likelihood

Optimization —

- Perform *forward pass* with randomly initialized parameters, to calculate loss
- Perform *backpropagation through time*, to calculate gradient:
  - $y = \varphi(X_h W_h)$
  - $\nabla_{W_h} L \propto \sum_{k=1}^{t} (\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}) \frac{\partial h_k}{\partial W_k}$
  - $\frac{\partial h_{i+k}}{\partial h_i} = \prod_{j=0}^{k-1} \frac{\partial h_{i+k-j}}{\partial h_{i+k-j-1}}$
- Perform gradient descent to find best weights

30 Long-Short-Term Memory (LSTM)

Description

Description —

- Can deal with sequential data and the persistence of information over time
- Can be seq-to-seq, seq-to-vec, or vec-to-seq
- Can be unidirectional or bidirectional

Formulation

Formulation —

- Model architecture:
  - Input layer
  - Hidden layer resp. memory cell:
    - Short-term state h_t, long-term state c_t
    - Cell output y_t
  - Output layer
- Forget gate:
  - Sigmoid layer
  - Serves to decide which information to keep from previous cell state, 1 : retain, 0: forget
  - $f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f$ where

$$w_f = \begin{bmatrix} w_{xf} & = \text{connection weight for } x_t \\ w_{hf} & = \text{connection weight for } h_{t-1} \end{bmatrix}$$

- Input gate:
  - Two stages:
    - Sigmoid layer, determines if values are updated, 1 : update values, 0: do not update values
    - Tanh layer, creates vector of new candidate values that could be added to the cell state
  - Serves to decide which information will be stored in the cell state
  - $i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i$ where

$$w_i = \begin{bmatrix} w_{xi} & = \text{connection weight for } x_t \\ w_{hi} & = \text{connection weight for } h_{t-1} \end{bmatrix}$$

  - $\tilde{c}_t = tanh(w_{\tilde{c}} \cdot [h_{t-1}, x_t] + b_c$ where

$$w_{\tilde{c}} = \begin{bmatrix} w_{x\tilde{c}} & = \text{connection weight for } x_t \\ w_{h\tilde{c}} & = \text{connection weight for } h_{t-1} \end{bmatrix}$$

- Cell state:
  - Two stages:
    - Calculate what is left from previous cell state after forget care
    - Calculate what we need to add to cell state after input gate
  - Serves to update old cell state into new cell state
  - $c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t$ where $\otimes$ is element-wise multiplication
- Output gate:
  - Three stages:
    - Sigmoid layer, determines which part of cell state to output
    - Tanh layer, activates cell state values
    - Multiply activated cell state and output gate values to get h_t
  - Serves to output h_t for next time step, which is a filtered version of cell state c_t
  - $o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o$ where

$$w_o = \begin{bmatrix} w_{xo} & = \text{connection weight for } x_t \\ w_{ho} & = \text{connection weight for } h_{t-1} \end{bmatrix}$$

  - $h_t = o_t \otimes tanh(c_t)$

Optimization

Parameters — Find parameters θ, which are shared across time steps

Objective function —

- Maximize log likelihood

Optimization —

- Perform *forward pass* with randomly initialized parameters, to calculate loss
- Perform *backpropagation through time*, to calculate gradient
- Perform gradient descent to find best weights

31 Attention

Description

- Attention helps specify which inputs we need to pay attention to when producing a given output
- Can be used:
  - As cross-attention: Between encoder and decoder
  - As self-attention: Within a single hidden layer resp. within the encoder or decoder
- Usually applied after embedding and before applying an activation function

Naive method

- Let E be the embedding matrix $\mathbb{R}^{s \times d}$ where s = number of tokens, d = embedding dimensionality
- Steps:
  - Re-weight embeddings: $\tilde{E} = Ew$
  - Compute similarity matrix: $S = \sigma(\tilde{E}\tilde{E}^\top)$ where σ is softmax, normalizing the rows of S to sum to 1, $\tilde{E}\tilde{E}^\top$ is similarity between two tokens in $\tilde{E}$
  - Compute attention-weighted embedding matrix: $A = S\tilde{E}$
- In this context, E and w are trainable parameters
- Challenge: Similarity matrix is symmetric, given that dot product $\tilde{E}\tilde{E}^\top$ is symmetric, i.e. attention paid by token i to token j (S_{ij}) is same as attention paid by token j to token i (S_{ji})

Adjusted method

- Steps:
  - Generate three sets of re-weighted embeddings:
    - $Q = EW^q$ resp. $q_i = e_i W^q$
      - E $(m \times h)$
      - W_q $(h \times d_k)$
      - Q $(m \times d_k)$
      - q_i is row vector $(1 \times d_k)$
      - e_i is row vector $(1 \times h)$
    - $K = EW^k$ resp. $k_i = e_i W^k$
      - E $(n \times h)$
      - W_k $(h \times d_k)$
      - K $(n \times d_k)$
      - k_i is row vector $(1 \times d_k)$
      - e_i is row vector $(1 \times h)$

  - $V = EW^v$ resp. $v_i = e_i W^v$ where
    - E $(n \times h)$
    - W_v $(h \times d_v)$
    - V $(n \times d_v)$
    - v_i is row vector $(1 \times d_v)$
    - e_i is row vector $(1 \times h)$
  - (Note: if Q, K, V contain multiple heads, they are expanded in this step)
- Compute similarity matrix: $A = \sigma(\frac{QK^\top}{\sqrt{d_k}})$ in $(m \times n)$ resp.

  $\alpha_t = \sigma(\frac{q_t K^\top}{\sqrt{d_k}})$ resp. $\alpha_{ti} = \frac{exp(q_t \cdot k_i)}{\sum_{i'} exp(q_t \cdot k_{i'})}$
  - $\sum_i \alpha_{ti} = 1$
  - $\alpha_{ti} \geq 0$
- Compute attention-weighted embedding matrix: $Z = AV$ in $(m \times d_v)$ in $(m \times d_v)$ resp. $z_t = \alpha_t V = \sum_i \alpha_{ti} v_i$
- (Note: if A contains multiple heads, they are flattened in this step by multiplying with d_v)

- In cross-attention:
  - Attention for decoder-encoder alignment
  - Q is generated with decoder input during training with m
  - V, K are generated with encoder outputs during training with n
- In self-attention:
  - Attention for encoder resp. decoder inputs
  - Q, V, K are generated with input during training with all either n or m
  - In masked self-attention:
    - We first calculate $P = QK^\top$ where masked elements (e.g. states with time $\geq m$ in decoder) are set to $-\infty$
    - $S = \sigma(\frac{P}{\sqrt{d_k}})$
- In multi-head attention:
  - Creates multiple sets of Q, K, V and calculates attention correspondingly
  - Concatenates generated matrices Z
  - Multi Head Attention$Z$ = Concat$(Z_{\text{head}_1}, \ldots, Z_{\text{head}_h})W_O + b_O$ where
    - Concat(...) in $(m \times (n \times n_{heads}))$
    - W_O in $((n_{heads} \times n) \times d_v)$
    - b_O in $1 \times d_v)$

Further proofs

Self-attention without positional encodings is permutation equivariant —

- Permutation equivariance: AttentionΠZ = Π Attention$Z$
- The self-attention is given by: $A = ZW_q W_k^\top Z^\top$
- After permutation, self-attention is given by: $A' = (\Pi Z)W_q W_k^\top (\Pi Z)^\top = \Pi Z W_q W_k^\top Z^\top \Pi^\top = \Pi(ZW_q W_k^\top Z^\top)\Pi^\top = \Pi A \Pi^\top$
- Applying softmax: softmax(A') = softmax$(\Pi A \Pi^\top) = \Pi$ softmax(A) $\Pi^\top$ since permutation matrix simply swaps rows and columns. The softmax operates on a matrix row-wise, i.e. the normalization for each row only depends on entries in that row. For this reason, it does not matter whether the permutation happens before or after applying the softmax
- Final output: $Z' =$ softmax$(A')(\Pi Z)W_v = \Pi$ softmax(A) $\Pi^\top (\Pi Z)W_v = \Pi$ softmax$(A)(\Pi^{-1}\Pi)ZW_v = \Pi$ softmax$(A)ZW_v$ because Π, as a permutation matrix, has exactly one 1 in each row and each column and 0 everywhere else. It is an orthogonal matrix, thus $\Pi^\top = \Pi^{-1}$ and $\Pi\,\Pi^{-1} = I$

Self-attention with learned Q and without positional encodings is permutation invariant —

- Permutation invariance: AttentionΠZ = Attention$Z$
- See proof above, but do not decompose Q

Self-attention with positional encodings is not permutation equi- or invariant —

- P encodes absolute positions that are altered when permuted
- This disrupts the symmetry introduced by the permutation matrix
- Therefore, the positional encoding P introduces information that is not equi- or invariant to permutations

32 Positional Embeddings

- Can be absolute or relative
- Attention with absolute positional encodings: $A^{\text{absolute}}_{q,k} = (Z_q + P_q)W_q W_k^\top (Z_k + P_k)^\top = Z_q W_q W_k^\top Z_k^\top + Z_q W_q W_k^\top P_k^\top + P_q W_q W_k^\top Z_k^\top + P_q W_q W_k^\top P_k^\top$
- Attention with relative positional encodings, where relative difference $\delta = q - k$: $A^{\text{relative}}_{q,k} := Z_q W_q W_k^\top Z_k^\top + Z_q W_q \widetilde{W}_k r_\delta + u^\top W_k Z_k + v^\top \widetilde{W}_k r_\delta$ where *Gaussian encodings* are given by parameters
  - $W_q = W_k = 0$
  - $\widetilde{W}_k = I$
  - $r_\delta = \begin{pmatrix} \|\delta\|^2 \\ \delta_1 \\ \delta_2 \end{pmatrix}$
  - $v = -\alpha \begin{pmatrix} 1 \\ -2\Delta_1 \\ -2\Delta_2 \end{pmatrix}$
  - v and r_δ are in $(1 \times d_p)$
  - If these parameters are plugged into formula for attention with relative positional encodings, we recover formula for attention with absolute positional encodings
- Relative encodings speed up the calculation of the attention vs. absolute encodings (since d_p is very small), but applying softmax and calculating Z for relative encodings has the same complexity as for absolute encodings, thus diminishing the benefit

33 Encoder Decoder RNNs

Description

Task — Generate embeddings, perform sequence-to-sequence tasks

Formulation

Formulation —

- Model architecture:
  - Inputs fed into encoder in reverse order
  - Encoder:
    - Sequence-to-vector
    - Hidden states $h_n^{(e)} = f(W_1^{(e)} h_{n-1}^{(e)} + W_2 w_n)$ where
      - f is activation function
      - w_n is input token embedding at time step n in input sequence
      - $h_{n-1}^{(e)}$ is encoder hidden state from previous time step
  - Outputs from encoder to decoder are weighted by attention weights:
    - Context vector $z_m = \sum_{n=1}^N \alpha_{m,n} h_n^{(e)}$ where
      - $h_n^{(e)}$ is the encoder hidden state (= V)
      - $\alpha_{m,n}$ is attention weight at decoder time step m for encoder hidden state at time step n
      - $\alpha_{m,n} =$ softmax$(h_{m-1}^{(d)} \times [h_1^{(e)}, \ldots, h_N^{(e)}])$ where
      - $h_{m-1}^{(d)}$ is previous decoder hidden state (= Q)
      - $[h_1^{(e)}, \ldots, h_N^{(e)}]^\top$ are the final encoder hidden states at each time step (= K)
  - Alongside context vectors, target sequence inputs are fed into decoder with one time step lag during training
  - Decoder:
    - Vector-to-sequence
    - Hidden states $h_m^{(d)} = f(W_3^{(d)} h_{m-1}^{(d)} + W_2^{(d)} w'_{m-1} + W_1^{(d)} z_m)$ where
      - f is activation function
      - w'_{m-1} is target token embedding at time step $m-1$ in target sequence
      - $h_{m-1}^{(d)}$ is decoder hidden state from previous time step with $h_0^{(d)} = h_N^{(e)}$, i.e. last encoder output is first decoder input
      - z_m is cross-attention
- Runtime analysis:
  - Let $W^{(e)}h$ be in $((N \times d) \times (d \times d))$ resp. $W^{(d)}h$ in $((M \times d) \times (d \times d))$
  - Let number of encoder resp. decoder layers be l_e, l_d
  - We perform $z_m = \sum_{n=1}^N \alpha_{m,n} h_n^{(e)}$ for M decoder time steps, summing over N encoder outputs $h_n^{(e)}$ of dimensionality d
  - Encoder: $O(l_e N d^2)$ from hidden states
  - Decoder: $O(l_d M d^2 + l_d d N M)$
    - $O(l_d M d^2)$ from hidden states
    - $O(l_d d N M)$ from cross-attention
- Challenge: Sequential, cannot be parallelized. Solution: Transformers

Optimization

Parameters — Find parameters $\theta = W_1^{(e)}, W_2^{(e)}, W_1^{(d)}, W_2^{(d)}, W_3^{(d)}$

Objective function —

- Maximize log likelihood

Optimization —

- Perform *forward pass* with randomly initialized parameters, to calculate loss
- Perform backpropagation, to calculate gradient
- Gradient with regard to encoder output:
  - $\nabla_{h_1^{(e)}} L = \frac{\partial L}{\partial h_m^{(d)}}) \frac{\partial h_m^{(d)}}{\partial h_n^{(e)}}$
  - $\frac{\partial h_m^{(d)}}{\partial h_n^{(e)}} = \frac{\partial}{\partial h_n^{(e)}} W_3^{(d)} h_{m-1}^{(d)} \times \frac{\partial}{\partial h_n^{(e)}} W_1^{(d)} z_m \times \frac{\partial}{\partial h_n^{(e)}} f(W_3^{(d)} h_{m-1}^{(d)} + W_2^{(d)} w'_{m-1} + W_1^{(d)} z_m)$
  - We can further decompose $\frac{\partial}{\partial h_n^{(e)}} W_1^{(d)} z_m$:
    - $= \frac{\partial}{\partial h_n^{(e)}} \alpha_{m,n} h_n^{(e)} + \frac{\partial}{\partial h_n^{(e)}} \sum_{i \neq n}^N \alpha_{m,i} h_i^{(e)}$ $= \alpha_{m,n} + \frac{\partial}{\partial h_n^{(e)}} \alpha_{m,n} h_n^{(e)} + \frac{\partial}{\partial h_n^{(e)}} \sum_{i \neq n}^N \alpha_{m,i} h_i^{(e)}$ due to product rule for $\alpha_{m,n} h_n^{(e)}$
    - $= \Phi_{m,n} + \Phi'_{m,n} h_n^{(e)} + \sum_{i \neq n}^N \left[\Phi_{m,i} \frac{\partial h_i^{(e)}}{\partial h_n^{(e)}} + \Phi'_{m,i} h_i^{(e)} \right]$ due to product rule for $\alpha_{m,i} h_i^{(e)}$ and by replacing $\alpha_{m,n}$ with $\Phi_{m,n}$

- Runtime analysis:
  - $\frac{\partial}{\partial h_n^{(e)}} W_3^{(d)} h_{m-1}^{(d)}$ is in $O(m \times N + N - n)$:
    - Taking derivatives of m decoder steps, due to attention z_m which is applied over N encoder outputs, takes $O(m \times N)$
    - Taking derivatives of $N - n$ encoder steps takes $O(N - n)$
  - $\frac{\partial}{\partial h_n^{(e)}} W_1^{(d)} z_m$ is in $O(N)$:
  - $\frac{\partial}{\partial h_n^{(e)}} f(...)$ is in $O(N)$:
    - $\Phi_{m,n}$ and $\Phi'_{m,n}$ are in $O(1)$, since they don't contain $h_n^{(e)}$
    - $\sum_{i \neq n}^{N} \Phi_{m,i} \frac{\partial h_i^{(e)}}{\partial h_n^{(e)}}$ is in $O(N - n)$ if we reuse terms in chain rule by factorizing, since the derivative is only non-null for $N - n$ encoder steps
    - $\sum_{i \neq n}^{N} \Phi'_{m,i} h_i^{(e)}$ is in $O(N)$, since here we're summing over all N encoder steps
    - $W_3^{(d)} h_{m-1}^{(d)}$ and $W_2^{(d)} w'_{m-1}$ are in $O(1)$, since they don't contain $h_n^{(e)}$
    - $W_1^{(d)} z_m$ is in $O(N)$, since it represents the attention applied over N encoder outputs
- Perform gradient descent to find best weights

Variants

Variants —
- ELMO: Bi-directional LSTM

34 Encoder Decoder Transformers

Description

Task — Generate embeddings, perform sequence-to-sequence tasks

Formulation

Formulation —
- Model architecture:
  - Inputs fed into encoder in reverse order
  - Encoder:
    - Sequence-to-vector
    - Takes in input sequence token embeddings (semantic vector, X in $(N \times d_{model})$) and positional embeddings (sinusoidal pointer vector for word position, given that model is not sequential, P in $(N \times d_{model})$) and adds them: $H_0^{(e)} = X + P$
    - Multi-head self-attention, applied to all tokens jointly, where Q, K, V are tokens in input sequence:
      - $Q = H_{(l-1)}^{(e)} W_q$
      - $K = H_{(l-1)}^{(e)} W_k$
      - $V = H_{(l-1)}^{(e)} W_v$
      - $\text{Attention} Z = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$
      - $\text{Multi Head Attention} Z = \text{Concat}(Z_{\text{head}_1}, \ldots, Z_{\text{head}_h}) W_O + b_O$
    - Addition and normalization: Skip connections (from token + positional embeddings) added back and normalized: $H_l^{(e)} = \text{Layer Norm}(\text{Multi Head Attention}(Q, K, V) + H_{(l-1)}^{(e)})$
    - Feed-forward network, parallel for each token: $\text{FFN}(H_l^{(e)}) = \text{ReLU}(H_l^{(e)} W_1 + b_1) W_2 + b_2$ where
      - $W_1 \in \mathbb{R}^{(d_v \times r)}$
      - $b_1 \in \mathbb{R}^{(1 \times r)}$
      - $W_2 \in \mathbb{R}^{(r \times d_v)}$
      - $b_2 \in \mathbb{R}^{(1 \times d_v)}$
    - Addition and normalization: Skip connections (from first addition and normalization) added back and normalized: $H_l^{(e)} = \text{Layer Norm}(\text{FFN}(H_l^{(e)}) + H_l^{(e)})$
    - Generates hidden states $h_n^{(e)}$
  - Decoder:
    - Vector-to-sequence
    - Target sequence inputs are fed into decoder with one time step lag (masked self-attention):
      - Takes in target sequence token embeddings (semantic vector, Y in $(M \times d_{model})$) and positional embeddings (sinusoidal pointer vector for word position, given that model is not sequential, P in $(M \times d_{model})$) and adds them: $H_0^{(d)} = Y + P$
      - Masked self-attention, applied to all tokens jointly, where Q, K, V are tokens in target sequence:
      - $Q = H_{(l-1)}^{(d)} W_q$
      - $K = H_{(l-1)}^{(d)} W_k$
      - $V = H_{(l-1)}^{(d)} W_v$
      - $\text{Masked Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \text{mask}\right) V$ where mask covers tokens in positions $m \geq t$
      - Addition and normalization: Skip connections (from token + positional embeddings) added back and normalized: $H_l^{(d)} = \text{Layer Norm}(\text{Masked Attention}(Q, K, V) + H_{(l-1)}^{(d)})$
    - Encoder outputs are fed into decoder with cross-attention:
      - Cross-attention:
      - $Q = H_l^{(d)} W_q$
      - $K = H_{(N)}^{(e)} W_k$
      - $V = H_{(N)}^{(e)} W_v$
      - $\text{Cross Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$
      - Addition and normalization: Skip connections (from first addition and normalization) added back and normalized: $H_l^{(d)} = \text{Layer Norm}(\text{Cross Attention}(Q, K, V) + H_l^{(d)})$
    - Feed-forward network, parallel for each token: $\text{FFN}(H_l^{(d)}) = \text{ReLU}(H_l^{(d)} W_1 + b_1) W_2 + b_2$ where
      - $W_1 \in \mathbb{R}^{(d_v \times r)}$
      - $b_1 \in \mathbb{R}^{(1 \times r)}$
      - $W_2 \in \mathbb{R}^{(r \times d_v)}$
      - $b_2 \in \mathbb{R}^{(1 \times d_v)}$
    - Addition and normalization: Skip connections (from second addition and normalization) added back and normalized: $H_l^{(d)} = \text{Layer Norm}(\text{FFN}(H_l^{(d)}) + H_l^{(d)})$
    - Generates hidden states $h_m^{(d)}$
  - Linear layer applied to $h_M^{(d)}$
  - Softmax layer applied to select token with highest probability:
    - Neural networks make no independence assumption, i.e. output y_t is conditioned on entire history (non-Markovian structure: $x, y_{<t}$) rather than window of size n (Markovian structure: $x, \langle y_t, \ldots, y_{t-1} \rangle$)
    - This results in runtime of $O(|\Sigma|^n)$ rather than $O(|\Sigma| \times n)$
    - Since it is intractable to search for best sequence overall (explore), we turn to deterministic or stochastic variants (exploit):
      - Greedy decoding: Select highest-probability token at each step
      - Beam search: Keep n-highest-probability tokens in memory (beam size) and return k-most-likely sequences (top beams)
      - Nucleus sampling: Sample tokens from items that cover $p\%$ of PMF
- Runtime analysis: Cross-attention:
  - Computing Q with $O(m \times h \times d_k)$, K with $O(n \times h \times d_k)$, V with $O(n \times h \times d_v)$
  - Assume $m = n$ and $d_k = d_v = d$
  - Computing A with $O(m \times d_k \times n)$, computing Z with $O(m \times d_k \times n \times d_v)$
  - Assume $m = 1$ (for one specific query) and $d_v = d$
  - Total runtime for single layer: $O(n \times h \times d + h \times n \times d)$
- Advantage:
  - Relies on attention to obtain a fixed-size representation of a sequence (in contrast to RNN)
  - Allows to learn longer-range dependencies than RNNs
  - Allows for parallelization, whereas RNNs must run sequentially

Variants

Variants —
- GPT: Uni-directional, decoder-only transformer, predicts next word in sequence
- BERT: Bi-directional, encoder-only transformer, predicts masked word from context

35 Connection CNN and Multi Head Self Attention

Theorem: A multi-head self-attention layer operating on K^2 heads of dimension n and output dimension d_v, employing a relative positional encoding of dimension $d_p \geq 3$, can express any convolutional layer of kernel size $K \times K$ and d_v output channels

Theorem part 1:
- Given a multi-head self-attention layer with $n_{heads} = K^2$ and $n \geq d_v$
- Given a convolutional layer with a $K \times K$ kernel and d_v output channels
- Let $f : [n_{heads}] \to \Delta_K$ be a bijective map between heads and shifts
- Assume $\text{softmax}(A) = \begin{cases} 1 & \text{if } f(h) = q - k = \delta \\ 0 & \text{otherwise} \end{cases}$
- Then, for any convolutional layer, there exists a corresponding weight per head W_v such that the multi-head self-attention equals the convolution
- Proof:
  - Contribution of each head in multi-head self-attention is given by: $W = W_v W_{\text{out}}^{(h)}$ where $W_{\text{out}}^{(h)}$ is the portion of W_{out} associated with head h
  - This means, we can rewrite $\text{Multi Head Attention} Z = \sum_{h \in n_{heads}} \text{softmax}(A^{(h)}) Z W^{(h)} + b_O$
  - This matches $\text{Convolution } Z = \sum_{(u,v) \in \Delta_K} X_{i',j'} W_{u,v} + b$

Theorem part 2:

- It is possible to construct a relative encoding scheme r_δ using parameters W_q, W_k, $\widetilde{W}_k$, and u so that, for every shift $\in \Delta_K$, there exists a vector v that yields the mapping $f : [n_{heads}] \to \Delta_K$
- Assume $A = -\alpha\left(\|\delta - \Delta\|^2 + c\right)$
- Behavior for $\delta = \Delta$ resp. $\delta \neq \Delta$:
  - Softmax is given by: $\text{softmax}(A) = \frac{\exp(-\alpha(\|\delta - \Delta\|^2 + c))}{\sum_{k'} \exp(-\alpha(\|\delta' - \Delta\|^2 + c))}$
  - In numerator:
    - If $\delta = \Delta$, $\exp(A) = \exp(-\alpha c)$
    - If $\delta \neq \Delta$, $\exp(A) \to 0$ as $\alpha \to \infty$, since entire term inside exponent grows very negative
  - In denominator: $\exp(A) \to \exp(-\alpha c)$ as $\alpha \to \infty$, since only the term corresponding to $\delta = \Delta$ contributes significantly
  - Then,
    - If $\delta = \Delta$, softmax $\to 1$
    - If $\delta \neq \Delta$, softmax $\to 0$
  - This proves assumption in part 1 of theorem
- Constant c is given by $c = \max_{\delta \neq \Delta} \|\delta - \Delta\|^2$:
  - $A = -\alpha\left(\|\delta - \Delta\|^2 + c\right)$
  - To ensure proper softmax behavior $-\alpha c$ must dominate over $-\alpha\|\delta - \Delta\|^2$
  - Then, we require $\|\delta - \Delta\|^2 + c \gg 0$ for $\delta \neq \Delta$

36 Generative Adversarial Nets (GAN)

Description

Task — Generate new samples

Formulation

Formulation —

- Model architecture:
  - Generator: Encoder decoder network, aims to produce samples that pass as real in the discriminator network
  - Discriminator: Fake detection network, aims to tell fake from real samples

Optimization

Steps —

- Encoder decoder network is trained in autoencoder mode to reproduce its input
- Decoder is fed with Gaussian noise input vectors $z^{(i)}$ and produces corresponding samples $\tilde{y}^{(i)}$
- Fake detection network produces $\hat{\xi} = 0$ for real samples $y^{(i)}$ and $\hat{\xi} = 1$ for $\tilde{y}^{(i)}$

Objective function —

- Encoder is trained to produce $\hat{z}^{(i)} = z^{(i)}$
- Decoder is trained to produce $\hat{\xi} = 0$
- Fake detection network is trained to correctly produce $\hat{\xi}$
- Trained in interleaving manner

37 Other

Proofs

Proofs —

- To prove $p \to q$:
  - Prove $p \to \neg q$ is impossible
  - Prove $\neg q \to \neg p$
- To prove $p \leftrightarrow q$: Prove $p \to q$ and $q \to p$
- To prove statement by induction:
  - Prove base case for $n = 0$ or $n = 1$
  - Assume inductive hypothesis: Assume statement holds for $n = k$
  - Prove inductive step: Prove statement holds for $n = k + 1$