1 Linear Algebra

Vector Properties

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

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

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

Unit vector — $\boldsymbol{u} = \frac{\tilde{\boldsymbol{u}}}{||\tilde{\boldsymbol{u}}||}$, 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.
$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_W = \boldsymbol{u}^\top \boldsymbol{W} \boldsymbol{v} = \sum_i u_i v_i w_i$ where $\boldsymbol{W}$ 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.
$||\boldsymbol{u}||_W^2 = \boldsymbol{u}^\top \boldsymbol{W} \boldsymbol{u} = \sum_i u_i^2 w_i$ where $\boldsymbol{W}$ 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 $\boldsymbol{v} \in V$ onto $\boldsymbol{s} \in S$ given by:
$\boldsymbol{v}_S = \frac{\boldsymbol{v} \cdot \boldsymbol{s}}{||\boldsymbol{s}||^2}\boldsymbol{s} = (\boldsymbol{v} \cdot \boldsymbol{s})\boldsymbol{s}$ 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 $\boldsymbol{S}$ if $\boldsymbol{S}\boldsymbol{h} \in H$ for all $\boldsymbol{h} \in H$ — Properties:

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

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

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

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

Solutions:

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

General Matrix Properties

Matrices —

- $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ with elements A_{ij}, rows $i = 1, ..., n$, columns $j = 1, ..., m$
- Transpose $\boldsymbol{A}^\top$
- Identity matrix $\boldsymbol{I}$ with 1 on diagonal, 0 elsewhere
- Scalar matrix $\boldsymbol{K}$ 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:
  - $\boldsymbol{A}^{n \times p}\boldsymbol{B}^{p \times m} = \boldsymbol{C}^{n \times m}$
    - $r_v \times c_v = s$
    - $c_v \times r_v = M$
    - $M \times c_v = c_v$
    - $r_v \times M = r_v$
    - $M \times M = M$
  - Element in $\boldsymbol{C}$ is sum-product of row in $\boldsymbol{A}$ and column in $\boldsymbol{B}$: $C_{ij} = \boldsymbol{A}^{(i)} \cdot \boldsymbol{B}^{(j)}$
  - Column vector in $\boldsymbol{C}$ is a linear combination of the columns in $\boldsymbol{A}$: $\boldsymbol{C}^{(j)} = \boldsymbol{A}\boldsymbol{B}^{(j)} = \sum_p \boldsymbol{A}^{(j=p)} b_p^{(j)}$

  - 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
  - $u \cdot v = u^\top v = \sum_i u_i v_i = c$
  - $uv^\top = C$ with $u_i v_j = C_{ij}$
  - $Au = \sum_{j=i} A^{(j)} u_i = c$ with $A^{(i)} \cdot u = A^{(i)\top} u = c_i$
  - $u^\top A = \sum_{j=i} A^{(i)\top} u_j = c^\top$ with $u \cdot A^{(j)} = u^\top A^{(j)} = c_j$
  - $AB = \sum_{j=i} A^{(j)} B^{(i)\top} = C$ with $A^{(i)} \cdot B^{(j)} = A^{(i)\top} B^{(j)} = C_{ij}$
  - $u^\top A u = \sum_i \sum_j x_i A_{ij} x_j$, 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 $i \leq j$
    - If A is diagonal: $= \sum_i x_i A_{ii} x_i$ since all off-diagonal terms are 0
  - Indexing a vector u on i returns the i^{th} element: $u_i = u_i$
  - Indexing (Au) on i returns the i^{th} element for the vector, generated via the i^{th} row for the matrix: $(Au)_i = A^{(i)} u = \sum_j A_{ij} u_j$ (! here the row is not written as a column vector, but as a true row vector)
- Moving between instance-level → data-level:
  - $x^{(i)} y = a \rightarrow X^\top y = a$ where X consists of rows $x^{(i)}$
  - $x^{(i)} x^{(i)\top} = A \rightarrow X^\top X = A$ where X consists of rows $x^{(i)}$
  - $x^{(i)} \cdot \beta = y_i \rightarrow X\beta = y$ where X consists of rows $x^{(i)}$

Properties —

- $(A+B)^\top = A^\top + B^\top$
- $(\alpha A)^\top = \alpha A^\top$
- $(AB)^\top = B^\top A^\top$
- $(A+B)+C = A+(B+C)$
- $A+B = B+A$
- $\alpha(A+B) = \alpha A + \alpha B$
- $(\alpha+\beta)A = \alpha A + \beta A$
- $(\alpha\beta)A = \alpha(\beta A)$
- $(A+B)x = Ax + Bx = Cx$
- $(AB)x = A(Bx) = Cx$
- $A = 0.5(A + A^\top) + 0.5(A - A^\top) = B + C$ where B is symmetric, but not C
- $A = AI = IA$
- $Ak = AK = KA$
- $\text{rank}(AB) = \min(\text{rank}(A), \text{rank}(B))$
- $A^\top A$ satisfies:
  - Symmetric
  - Psd
  - Has rank m iff it is pd
  - Invertible iff it has rank m and it is pd
  - $\text{rank}(A^\top A) = \text{rank}(A) = \text{rank}(A^\top)$
  - $\text{rank}(A^\top A) = \text{rank}([A^\top A | A^\top x])$

Matrix terminology —

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

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

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

Projection matrices —
Generally:

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

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

- Projection of x onto S is given by: $u = \sum_i (x \cdot b_i) b_i = \sum_i b_i b_i{}^\top x = BB^\top x = Cx$
- Projection of x onto $S^\perp$ is given by: $x - u = Ix - Cx$

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

- Projection of x onto S is given by: $s = AA^\# x$ since $AA^\#$ is a projection matrix due to $AA^\# = (AA^\#)^2$
- $s = U_+ U_+^\top x$
- *SVD Projection Energy*: $\sum_{l=1}^m |u_k \cdot y_l|^2 = \sigma_k^2$
  Proof: $\sum_{l=1}^m |u_k \cdot y_l| = \|u_k^\top A\| = u_k^\top AA^\top u_k = u_k^\top USV^\top VS^\top U^\top u_k = e_k^\top SS^\top e_k = \sigma_k^2$

Square Matrix Properties

Square matrix terminology —

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

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

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

Inversely, if A is not invertible, the columns and rows are not linearly independent, etc.

Matrix inversion lemma —

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

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

Eigenvectors and eigenvalues —

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

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

- $A \succ 0$ iff $x^\top Ax > 0$
- $A \succeq 0$ iff $x^\top Ax \geq 0$

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 diagonalizable (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 diagonalizable (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: $S_n = \sum_{i=1}^{n} a_i r^{i-1} = a_1(\frac{1-r^n}{1-r})$
- 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$, returns vector containing first-order partial derivatives:

  $\nabla_{\boldsymbol{x}} f : [\frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_n}]^\top$
- 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$, returns matrix containing second-order partial derivatives:

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

- $\mathcal{H}$ is symmetric

Jacobian —

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

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

Matrix calculus rules —

- $\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}$
- $\frac{\partial \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}}{\partial \boldsymbol{x}} = (\boldsymbol{A} + \boldsymbol{A}^\top)\boldsymbol{x}$
- $\frac{\partial \boldsymbol{a}^\top \boldsymbol{A} \boldsymbol{b}}{\partial \boldsymbol{A}} = \boldsymbol{a}\boldsymbol{b}^\top$
- For symmetric $\boldsymbol{A}$:

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

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 $\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$ 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 $f(\boldsymbol{x})$ subject to $g(\boldsymbol{x}) = 0$

- Gradient of $f(\boldsymbol{x})$ must be orthogonal to constraint surface, otherwise (if it points into any direction along the constraint surface) $f(\boldsymbol{x})$ could still decrease for movements along the constraint surface
- On the constraint surface, $g(\boldsymbol{x})$ is a constant, so moving along any direction on the constraint surface has a directional derivative of 0. Since the gradient of $g(\boldsymbol{x})$ 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) $g(\boldsymbol{x})$ 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 $\boldsymbol{x}^*$ and λ^*:
  - $\nabla_{\boldsymbol{x}} \mathcal{L} = 0$, expresses parallelity condition at minimum $\boldsymbol{x}^*$
  - $\nabla_{\lambda} \mathcal{L} = 0$, expresses constraint
  - This is an unconstrained optimization problem
- Optimum $\boldsymbol{x}^*$ and λ^* represents a saddle point of $\mathcal{L}$

For inequality constraints: Minimize $f(\boldsymbol{x})$ subject to $g(\boldsymbol{x}) \leq 0$

- If $\boldsymbol{x}^*$ lies in $g(\boldsymbol{x}) < 0$, constraint is inactive
- Otherwise, if $\boldsymbol{x}^*$ lies in $g(\boldsymbol{x}) = 0$, constraint is active:
  - Gradient of $f(\boldsymbol{x})$ must point towards $g(\boldsymbol{x}) < 0$ region, otherwise (if it would point away from $g(\boldsymbol{x}) < 0$ 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 $\boldsymbol{x}^*$ 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 $\boldsymbol{x}^*$ and λ^* represents a saddle point of $\mathcal{L}$

For multiple constraints: Minimize $f(\boldsymbol{x})$ 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 $\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ 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 $f(\boldsymbol{x}^*)$ 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 $\theta(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ 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, $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}}[\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \mathcal{L}] = \min_{\boldsymbol{x}}(f(\boldsymbol{x}))$ (provided barrier function) $\geq \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}}[\min_{\boldsymbol{x}} \mathcal{L}] = \theta(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$
- $\min_{\boldsymbol{x}} \mathcal{L}$ 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, $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}}[\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \mathcal{L}] = \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}}[\min_{\boldsymbol{x}} \mathcal{L}] = \theta(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$
- $\min_{\boldsymbol{x}} \mathcal{L}$ can be solved for general solution $\boldsymbol{x}^*$ in terms of $\boldsymbol{\lambda}, \boldsymbol{\mu}$
- Plug $\boldsymbol{x}^*$ back into $\mathcal{L}$ and maximize to find solutions $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$
- Specify $\boldsymbol{x}^*$ based on $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$

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 —

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

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: $f_{X_1,X_2,...,X_n}(x_1,x_2,\dots,x_n) = \frac{\partial^n}{\partial x_1 \partial x_2 ... \partial x_n} F_{X_1,X_2,...,X_n}(x_1,x_2,\dots,x_n)$ $= \frac{\partial^n}{\partial x_1 \partial x_2 ... \partial x_n} F_{X_1}(x_1) F_{X_2}(x_2) \dots F_{X_n}(x_n)$ $= \frac{\partial}{\partial x_1} F_{X_1}(x_1) \frac{\partial}{\partial x_2} F_{X_2}(x_2) \dots \frac{\partial}{\partial x_n} F_{X_n}(x_n)$ $= f_{X_1}(x_1) f_{X_2}(x_2) \dots f_{X_n}(x_n)$
    - ⁎ Direction 2: $F_{X_1,X_2,...,X_n}(x_1,x_2,\dots,x_n) = \int_{-\infty}^{x_1}\int_{-\infty}^{x_2}\dots\int_{-\infty}^{x_n} f_{X_1,X_2,...,X_n}(\tilde{x}_1,\tilde{x}_2,\dots,\tilde{x}_n)\,d\tilde{x}_1\,d\tilde{x}_2\dots d\tilde{x}_n$ $= \int_{-\infty}^{x_1}\int_{-\infty}^{x_2}\dots\int_{-\infty}^{x_n} f_{X_1}(\tilde{x}_1) f_{X_2}(\tilde{x}_2)\dots f_{X_n}(\tilde{x}_n)\,d\tilde{x}_1\,d\tilde{x}_2\dots d\tilde{x}_n$ $= \int_{-\infty}^{x_1} f_{X_1}(\tilde{x}_1)\,d\tilde{x}_1 \int_{-\infty}^{x_2} f_{X_2}(\tilde{x}_2)\,d\tilde{x}_2 \dots \int_{-\infty}^{x_n} f_{X_n}(\tilde{x}_n)\,d\tilde{x}_n$ $= F_{X_1}(x_1) F_{X_2}(x_2) \dots F_{X_n}(x_n)$
  - 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: $p(x) = \sum_y p(x,y) = \sum_y p(x|y)p(y)$
  - 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: $f(x) = \sum_y f(x,y) = \sum_y f(x|y)p(y)$ and $F(r) = \sum_y p(y)p(X \leq r|y) = \sum_y p(y)F(r|y)$
- *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: $\mathbb{E}[X] = \sum_y \mathbb{E}[X|Y = y]p(y)$
- 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: $\mathbb{E}(\boldsymbol{x}) = \sum_{x_1} ... \sum_{x_n} \boldsymbol{x}^\top p(x_1, ..., x_n) = \boldsymbol{\mu}$
- If X is continuous: $\mathbb{E}(\boldsymbol{x}) = \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} \boldsymbol{x}^\top f_{X_1,...,X_n}(x_1, ..., x_n)dx_1...dx_n = \boldsymbol{\mu}$

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 $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$: $\mathbb{E}(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}\mathbb{E}(\boldsymbol{x})$

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

  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:
  - $\Sigma = \text{Cov}(\boldsymbol{x}) = \mathbb{E}((\boldsymbol{x} - \mathbb{E}(\boldsymbol{x}))(\boldsymbol{x} - \mathbb{E}(\boldsymbol{x}))^\top) = \mathbb{E}(\boldsymbol{x}\boldsymbol{x}^\top) - \mathbb{E}(\boldsymbol{x})\mathbb{E}(\boldsymbol{x})^\top = \boldsymbol{R} - \boldsymbol{\mu}_X\boldsymbol{\mu}_X^\top = \begin{bmatrix} \mathbb{E}(x_1^2) & ... & \mathbb{E}(x_1x_m) \\ ... & ... & ... \\ \mathbb{E}(x_mx_1) & ... & \mathbb{E}(x_m^2) \end{bmatrix} - \begin{bmatrix} \mathbb{E}(x_1)^2 & ... & \mathbb{E}(x_1)\mathbb{E}(x_m) \\ ... & ... & ... \\ \mathbb{E}(x_m)\mathbb{E}(x_1) & ... & \mathbb{E}(x_m)^2 \end{bmatrix} = \begin{bmatrix} \mathbb{V}(x_1) & ... & \text{Cov}(x_1,x_m) \\ ... & ... & ... \\ \text{Cov}(x_m,x_1) & ... & \mathbb{V}(x_m) \end{bmatrix}$ where $\boldsymbol{R} = \mathbb{E}(\boldsymbol{x}\boldsymbol{x}^\top)$ 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 $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$: $\mathbb{V}_y = \boldsymbol{A}\mathbb{V}_X\boldsymbol{A}^\top$
- 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 $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$:
  - $\Sigma_y = \boldsymbol{A}\Sigma_X\boldsymbol{A}^\top$
  - $\boldsymbol{R}_y = \boldsymbol{A}\boldsymbol{R}_X\boldsymbol{A}^\top$
- 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: $\frac{1}{2\pi^{n/2}}\frac{1}{|\Sigma|^{1/2}}exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu}))$ where the term in the exponent is a quadratic form

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

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 $B(\boldsymbol{\alpha})$ is the multivariate generalization of the Beta function: $B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^n \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^n \alpha_k)}$

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 $X_1, \dots, X_n$: $\mathbb{E}[e^{s(S_n - \mathbb{E}[S_n])}] = \prod_{i=1}^n \mathbb{E}[e^{s(X_i - \mathbb{E}[X_i])}]$
- 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 $\text{Var}(S_n)$: $P(|S_n - m_X| \geq \varepsilon) \leq \frac{\text{Var}(S_n)}{\varepsilon^2}$
- $\text{Var}(S_n) = \text{Var}\left(\frac{1}{n} \sum_{k=1}^n X_k\right) = \frac{1}{n^2} n \text{Var}(X_k) = \frac{\sigma_X^2}{n}$
- 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:
  - $\alpha \geq p(\text{type I error}) = p(\overline{x} \geq c \mid H_0)$ 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 $H_0 : \theta = 0$: $\alpha = p(\overline{x} \geq c \mid H_0) = p(\sqrt{n}\overline{x} \geq \sqrt{n}c \mid H_0) = p(z_n \geq \sqrt{n}c \mid H_0) = 1 - \Phi(\sqrt{n}c)$ where
    - * Φ is the CDF of the normal distribution
    - * $z_n \mid H_0 = \frac{\overline{x} - 0}{1/\sqrt{n}} = \sqrt{n}\overline{x}$
- *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*:
  - $1 - \beta = p(\overline{x} \geq c \mid H_1)$
  - Probability, given H_A, that we reject H_0
  - If $\mathcal{X} \sim \mathcal{N}(\theta, 1)$ and $H_0 : \theta = 0$: $1 - \beta = p(\overline{x} \geq c \mid H_1) = p(\sqrt{n}(\overline{x} - 1) \geq \sqrt{n}(c - 1) \mid H_1) = p(z_n \geq \sqrt{n}(c - 1) \mid H_1) = p(z_n \geq \sqrt{n}(c - 1) \mid H_0) = 1 - \Phi(\sqrt{n}(c - 1))$ where
    - * Φ is the CDF of the normal distribution
    - * $z_n \mid H_1 = \frac{\overline{x} - 1}{1/\sqrt{n}} = \sqrt{n}(\overline{x} - 1)$
    - * 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*:
  - $z_n \mid H_0 = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}}$

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 $\alpha* = \alpha/K$

Kernelized Hypothesis Tests — Aim:

- Let (X, Σ, P) be a probability space
- Given $x_1, \dots, x_m \sim p^*$ and $y_1, \dots, y_n \sim q^*$, decide if $p = q$

Idea: If $p \neq q$, then:

- $\exists A \in \Sigma : p(A) \neq q(A)$
- $\mathbb{E}_{x \sim p}[\mathbb{I}_A(x)] \neq \mathbb{E}_{y \sim q}[\mathbb{I}_A(y)]$
- $\exists f : \mathbb{E}_{x \sim p}[f(x)] \neq \mathbb{E}_{y \sim q}[f(y)]$ since identity can be approximated by a continuous function (see below)

Maximum Mean Discrepancy (MMD):

- $p \neq q \iff \exists f : \mathbb{E}_{x \sim p}[f(x)] \neq \mathbb{E}_{y \sim q}[f(y)]$
- $\text{MMD}[F, p, q] = \sup_{f \in F}\left(\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]\right)$
- Compares expectations of functions in a function space F, which typically is a *Reproducing Kernel Hilbert Space (RKHS)*, since the RKHS kernel is universal and can approximate any function, but alternatively F can also be the set of all polynomials (see below)

According to *Riesz Representation Theorem*, supremum in MMD can be reduced to a kernel expression:

- The theorem states that for any bounded linear functional L on a Hilbert space $\mathcal{H}$, there exists a unique $g \in \mathcal{H}$ such that for all $f \in \mathcal{H}$: $L(f) = \langle f, g \rangle_{\mathcal{H}}$, i.e. L can be represented as inner product
- This is relevant because we can now express any $f(x)$ as:$f(x) = \langle f, K(x, \cdot) \rangle_{\mathcal{H}}$
- $\text{MMD}^2[F, p, q] = \mathbb{E}_{p,p}[k(x, x')] - 2\mathbb{E}_{p,q}[k(x, y)] + \mathbb{E}_{q,q}[k(y, y')]$

  Proof:
  - Taking the MMD: $\sup_{f \in F}\left(\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]\right)$
  - $= \sup_{f \in F} \langle \beta_f, \mu_p \rangle - \langle \beta_f, \mu_q \rangle$
  - $= \sup_{\|f\| \leq 1} \langle \beta_f, \mu_p - \mu_q \rangle_F$
  - where $\mu_p = \mathbb{E}_p[k(x, \cdot)]$ and $\mu_q = \mathbb{E}_q[k(x, \cdot)]$
  - Supremum is achieved when f aligns perfectly with $\mu_p - \mu_q$
  - Maximum value is given by norm $\|\mu_p - \mu_q\|_F$
  - This maximum value equation can be expanded
- Note that: $\mathbb{E}_{p,q}[k(x, y)] = \mathbb{E}_p[\langle \phi(x), \phi(y) \rangle] = \mathbb{E}_p[\mathbb{E}_q[\Psi_x(y)]] = \mathbb{E}_p[\langle \phi(x), \mu_q \rangle] = \mathbb{E}_p[\Psi_{\mu,q}(x)] = \langle \mu_q, \mu_p \rangle$

Outcome:

- $p = q$ iff $\text{MMD}^2[F, p, q] = 0$.
- $\text{MMD}^2[F, p, q] = \mathbb{E}_{p,p}[k(x, x')] - 2\mathbb{E}_{p,q}[k(x, y)] + \mathbb{E}_{q,q}[k(y, y')]$ where $-2\mathbb{E}_{p,q}[k(x, y)] \approx -\frac{2}{mn} \sum_{i \leq m, j \leq n} k(x_i, y_j) + \dots$

Aside 1) Approximating identity function with continuous function:

- Let $\mathbb{I}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$
- Approximating sets for A:
  - *Shrunken interior* $A^{0,\epsilon}$: Subset of A where every point is at least ϵ away from complement of A resp. boundary of A: $A^{0,\epsilon} = \{x \in A : d(x, A^c) > \epsilon\}$

- *Expanded complement* $A^{1,\epsilon}$: Subset of the complement of A where every point is at least ϵ away from A: $A^{1,\epsilon} = \{x \in A^c : d(x,A) > \epsilon\}$

- $f(x)$ is defined piecewise to approximate $\mathbb{I}_A(x)$:
  - $f(x) =$ $$\begin{cases} 1 & \text{if } x \in A^{0,\epsilon} \\ 0 & \text{if } x \in A^{1,\epsilon} \\ \frac{1}{2} + \frac{1}{2}\inf\{q \in Q : x \in A^{1,\epsilon,q}\} & \text{if } x \in A \setminus A^{0,\epsilon} \text{ (near boundary of } A) \\ \frac{1}{2}\inf\{q \in Q : x \in A^{0,\epsilon,1-q}\epsilon\} & \text{if } x \in A^c \setminus A^{1,\epsilon} \text{ (near boundary of } A^c) \end{cases}$$ where q parametrizes smoothness, with larger q creating a more gradual transition
  - Then, near boundary of A, we have smooth transition from $0.5 \rightarrow 0$
  - Near boundary of A^c, we have smooth transition from $0 \rightarrow 0.5$

Aside 2) Approximating functions with polynomials

- We know that:
- $1 = 1^n = (x+1-x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k}$
- To approximate a function $f(x)$, we reweight this polynomial expression: $f(x) \approx \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}$

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
  - $f_\theta : \mathbb{R}^m \rightarrow \mathcal{Y}$
  - 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: $\hat{\theta} = \arg\max_\theta(L) = \arg\max_\theta(p(y_1, ..., y_n|\boldsymbol{x_i}, \boldsymbol{\theta})) = \arg\max_\theta(\prod_{i=1}^{n} p(y_i|\boldsymbol{x_i}, \boldsymbol{\theta})) = \arg\max_\theta(\sum_{i=1}^{n} log(p(y_i|\boldsymbol{x_i}, \boldsymbol{\theta})))$
- In discrete case:
  - $\hat{\theta} = \arg\max_\theta(L) = \arg\max_\theta(\prod_{i=1}^{n} p(y_i|\boldsymbol{x_i}, \boldsymbol{\theta})) = \arg\max_\theta \prod_{j=1}^{k} p_j^{N_j} = \arg\max_\theta \sum_{i=1}^{n} N_j log(p_j)$ where
    - $j = 1, ..., k$ is the number of classes
    - N_j county how often the outcome class j appears in $\boldsymbol{y}$
    - $p_j = p(y_i = j|\boldsymbol{x_i}, \boldsymbol{\theta})$
  - 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: $\Lambda = \frac{\partial}{\partial \boldsymbol{\theta}} log(p(y|\boldsymbol{x}, \boldsymbol{\theta})) = \frac{\frac{\partial}{\partial \boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta})}{p(y|\boldsymbol{x}, \boldsymbol{\theta})}$
  - The expected score is given by: $\mathbb{E}(\Lambda) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) \frac{\frac{\partial}{\partial \boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta})}{p(y|\boldsymbol{x}, \boldsymbol{\theta})} dx = \frac{\partial}{\partial \boldsymbol{\theta}} \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) dx = \frac{\partial}{\partial \boldsymbol{\theta}} \times 1 = 0$
- Advantages:
  - *Consistent*: $\hat{\theta} \rightarrow \theta$ as $n \rightarrow \infty$
  - *Asymptotically normal*: $\frac{1}{\sqrt{n}}(\hat{\theta} - \theta)$ converges to $\mathcal{N}(0, \boldsymbol{J}^{-1}(\boldsymbol{\theta}) \boldsymbol{I}(\boldsymbol{\theta}) \boldsymbol{J}^{-1}(\boldsymbol{\theta}))$ where $\boldsymbol{J} = -\mathbb{E}[\frac{\partial^2 log(p(y|\boldsymbol{x}, \boldsymbol{\theta}))}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}]$ and where $\boldsymbol{I}$ is the Fisher information
  - *Asymptotically efficient*: $\hat{\theta}$ minimizes $\mathbb{E}[(\hat{\theta} - \theta)^2] \rightarrow \frac{1}{\boldsymbol{I}_n(\boldsymbol{\theta})}$ as $n \rightarrow \infty$ where $\boldsymbol{I}$ 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: $\Lambda = \frac{\partial}{\partial \boldsymbol{\theta}} log(p(y|\boldsymbol{x}, \boldsymbol{\theta})) = 0$
    - With a Taylor expansion, we can show that $(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})\sqrt{n} = \frac{1}{\sqrt{n}} \Lambda [-\frac{1}{n} \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \sum_{i=1}^{n} p(y_i|\boldsymbol{x_i}, \boldsymbol{\theta}))]^{-1}$ where Λ is the score
    - We set $\boldsymbol{J} = [-\frac{1}{n} \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \sum_{i=1}^{n} p(y_i|\boldsymbol{x_i}, \boldsymbol{\theta}))]$
    - $\frac{1}{\sqrt{n}} \Lambda$ is a random vector with covariance matrix $\boldsymbol{I}$ and converges to the normal distribution $\sim \mathcal{N}(0, \boldsymbol{I})$
    - Then, $(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})\sqrt{n} = \boldsymbol{J}^{-1} \frac{1}{\sqrt{n}} \Lambda \sim \boldsymbol{J}^{-1} \mathcal{N}(0, \boldsymbol{I})$
    - $\mathbb{V}(\boldsymbol{J}^{-1} \frac{1}{\sqrt{n}} \Lambda) = \mathbb{E}[\boldsymbol{J}^{-1} \boldsymbol{I} \boldsymbol{J}^{-1}]$
    - 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 $(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})\sqrt{n} = \boldsymbol{J}^{-1} \frac{1}{\sqrt{n}} \Lambda \sim \mathcal{N}(0, \boldsymbol{J}^{-1} \boldsymbol{I} \boldsymbol{J}^{-1})$
  - Equivariant:
    - Let $t = g(\theta)$ and $h = g^{-1}$
    - Then, $\boldsymbol{\theta} = h(t) = h(g(\boldsymbol{\theta}))$
    - For all t we have: $L(t) = \prod_i p(y_i|\boldsymbol{x_i}, h(t))) = p(y_i|\boldsymbol{x_i}, \boldsymbol{\theta}) = L(\boldsymbol{\theta})$
    - Hence, for all t we can say: $L(t) = L(\boldsymbol{\theta})$ and $L(\hat{t}) = L(\hat{\boldsymbol{\theta}})$
- 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: $\mathrm{BCE}(y,p_1) = -[y\log(p_1) + (1-y)\log(1-p_1)]$
    * When $y = 1$: $\mathrm{BCE}(1,p_1) = -\log(p_1)$
    * When $y = 0$: $\mathrm{BCE}(0,p_1) = -\log(1-p_1)$
  - Log loss:$\mathrm{LL}(y,p) = -\log(p_y)$
    * When $y = 1$: $\mathrm{LL}(y,p) = -\log(p_1)$
    * When $y = 0$: $\mathrm{LL}(y,p) = -\log(p_0) = -\log(1-p_1)$

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 $\mathcal{Z} = \{(x_1,y_1),\dots,(x_n,y_n)\}$ 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 $n > \mathrm{poly}\Big(1/\epsilon, 1/\delta, \mathrm{size}(c)\Big)$, it outputs $\hat{c}$ such that: $P(\mathcal{R}(\hat{c}) \le \epsilon) \ge 1 - \delta$ 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
  - $\mathrm{size}(c)$: 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

Scenario 1: If $\mathcal{C}$ is finite and $\mathcal{H} = \mathcal{C}$:

- We aim to bound the risk of $\hat{c}$ as follows: $P(\mathcal{R}(\hat{c}) > \epsilon) \le \delta$
- What is the required sample size for this?
- According to *Hoeffding's inequality* for a single hypothesis: $P(|\mathcal{R}(c) - \mathcal{R}(\hat{c})| > \epsilon) = P(\mathcal{R}(c) > \epsilon) \le \exp(-n\epsilon)$
- To account for all hypotheses in $\mathcal{C}$, we apply a union bound: $P(\exists c \in \mathcal{C} : \mathcal{R}(c) > \epsilon) \le |\mathcal{C}| \times \exp(-n\epsilon)$
- We then get: $|\mathcal{C}| \times \exp(-n\epsilon) \le \delta$
- From this, we can derive: $\log(|\mathcal{C}|) - n\epsilon \le \log(\delta)$

  $\log(|\mathcal{C}|) - \log(\delta) \le n\epsilon$

  $\frac{1}{\epsilon}\Big[\log(|\mathcal{C}|) + \log\Big(\frac{1}{\delta}\Big)\Big] \le n$

Scenario 2: If $\mathcal{C}$ is finite and $\mathcal{H} \neq \mathcal{C}$:

- *Bayes optimal classifier:*
  - Classifier that achieves minimum possible error
  - Outputs labels based on true probabilities of labels given input features: $\hat{y} = \arg\max_y p(y \mid \boldsymbol{x})$
- If the Bayes optimal classifier is not in $\mathcal{C}$, there will always be a gap between the error of the best hypothesis $\hat{c}$ and Bayes optimal error
- In this case, we aim to bound the risk of $\hat{c}$ as follows: $P\Big(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) \le \epsilon\Big) \ge 1 - \delta$
- How can we specify the bound?
- According to *Hoeffding's inequality* for a single hypothesis: $P\Big(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon\Big) \le 2\exp(-n\epsilon^2)$
- To account for all hypotheses in $\mathcal{C}$, we apply a union bound: $P\Big(\exists c \in \mathcal{C} : \mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon\Big) \le 2|\mathcal{C}|\exp(-n\epsilon^2)$

Scenario 3: If $\mathcal{C}$ has finite VC-dimension, but is infinite:

Solving scenario 3 generically:

- The *VC-dimension* of a concept class $\mathcal{C}$ is a measure of its complexity: VC-dimension V_c is the size of the largest set A that can be shattered by $\mathcal{C}$
  - A set of instances A is *shattered* by the concept class $\mathcal{C}$ if, for every subset $S \subseteq A$, there is a concept $c_S \in \mathcal{C}$ such that $S = c_S \cap A$
  - This means the concept class can realize or perfectly label every possible labeling of A
  - E.g. for 2^n points, there are 2^n possible ways to assign binary labels to the points. If $\mathcal{C}$ can express all 2^n labelings, it shatters A
- How can we specify the bound?
- If $V_c > 2$: $P\Big(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon\Big) \le 9n^{V_c}\exp\Big(-\frac{n\epsilon^2}{32}\Big)$
- How does the bound behave in the limit?
- If the VC-dimension is finite: $P\Big(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon\Big) \le 9n^{V_c}\exp\Big(-\frac{n\epsilon^2}{32}\Big) \to 0$ as $n \to \infty$

Solving scenario 3 for estimators with uniform convergence:

- *Uniform convergence*: Empirical risk converges uniformly to the expected risk: $\sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| \to 0$ as $n \to \infty$
- How can we specify the bound?
- $\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) = \mathcal{R}(\hat{c}^*) - \hat{\mathcal{R}}(\hat{c}^*) + \hat{\mathcal{R}}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c)$ where
  - $\mathcal{R}$ is the expected risk resp. generalization error vs. $\hat{\mathcal{R}}$ is the empirical risk resp. training error
  - c is a generic, not further specified classifier, $\hat{c}$ is the trained classifier, $\hat{c}^*$ is the trained, optimal classifier
- $\le \mathcal{R}(\hat{c}^*) - \hat{\mathcal{R}}(\hat{c}^*) + \hat{\mathcal{R}}(c^*) - \mathcal{R}(c^*)$
- $\le \sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| + \sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)|$
- $\le 2\sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)|$
- Then, $P(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon) \le P(\sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| > \frac{\epsilon}{2})$
- By the union bound: $P(\sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| > \epsilon) \le \sum_{c\in\mathcal{C}}P(|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| > \epsilon)$
- By Hoeffding's inequality, and the fact that $\hat{\mathcal{R}}(c) \in [0,1]$, whereby $b = 1, a = 0$, and assuming the cardinality of C is bounded by N: $\le 2N\exp(-2n\epsilon^2)$
- How does the bound behave in the limit?
- We have $P(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon) \le P(\sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| > \frac{\epsilon}{2})$ and $P(\sup_{c\in\mathcal{C}}|\hat{\mathcal{R}}(c) - \mathcal{R}(c)| > \epsilon) \le 2N\exp(-2n\epsilon^2)$
- This gives $P(\mathcal{R}(\hat{c}^*) - \inf_{c\in\mathcal{C}}\mathcal{R}(c) > \epsilon) \le 2N\exp(-2n\frac{\epsilon^2}{2}) \to 0$ as $n \to \infty$
- If we define confidence $\delta = 2N\exp(-2n\epsilon^2)$, then error tolerance is $\epsilon = \sqrt{\frac{\ln(N) - \ln(\delta/2)}{2n}}$
- Given that the inequality $\mathcal{R}(c) - \hat{\mathcal{R}}(c) \le \epsilon$ holds with probability $1 - \delta$, we get: $\mathcal{R}(c) \le \hat{\mathcal{R}}(c) + \epsilon = \hat{\mathcal{R}}(c) + \sqrt{\frac{\ln(N) - \ln(\delta/2)}{2n}}$

Solving scenario 3 for hyperplanes:

- *Quantization of infinite hypothesis classes resp. fingering argument*: An infinite hypothesis class can be represented by a finite subset of hypotheses
- Hyperplanes in $\mathbb{R}^d$ are represented mathematically as $\sum_{i=1}^{d} a_i x_i + a_0 = 0$
- Each hyperplane defined by d points creates two classifiers: $c_\alpha(x) = \begin{cases} 1 & \text{if } a^T x + a_0 > 0 \\ 0 & \text{otherwise} \end{cases}$
- From n samples, there are $\binom{n}{d}$ possible sets of d points, yielding: $2 \times \binom{n}{d}$ total classifiers
- The best empirical classifier is given by: $\hat{c} = \arg\min_{i=1,\dots,2\times\binom{n}{d}}\hat{\mathcal{R}}(c_i)$
- How can we specify the bound?
- The empirical error of any classifier c satisfies: $\hat{R}(c) \ge \hat{R}(\hat{c}) - \frac{d}{n}$
- Using Hoeffding's inequality and the union bound, if $n \ge d$ and $2d/n \le \epsilon \le 1$: The probability of a large deviation between empirical and expected risk is: $P\left(R(\hat{c}) > \inf_{c\in\mathcal{C}}R(c) + \epsilon\right) \le \exp\left(2d\epsilon\right)\Big(2\binom{n}{d} + 1\Big)\exp\Big(-\frac{n\epsilon^2}{2}\Big)$

  The expected deviation is: $\mathbb{E}[\mathcal{R}(\hat{c}) - \mathcal{R}] \le \sqrt{\frac{2}{n}(d+1)(\log n + 2)}$

Solving scenario 3 for special case of hyperplanes: Zero error classifiers:

- How can we specify the bound?
- If the optimal classifier has zero expected error ($R(c^*) = 0$), and $n > d$, and $\epsilon \le 1$, convergence improves: $P\left(R(\hat{c}) > \epsilon\right) \le 2\binom{n}{d}\exp\left(-\epsilon(n-d)\right)$

  Proof:
  - By fingering argument: $P(\mathcal{R}(\hat{c}_n) > \epsilon) \le P\Big(\max_{i=1,\dots,2\binom{n}{d}}\mathcal{R}(c_i) > \epsilon\Big)$
  - By union bound: $\le \sum_{i=1}^{2\binom{n}{d}} P\Big(\hat{\mathcal{R}}_n(c_i) \le \frac{d}{n} \wedge \mathcal{R}(c_i) > \epsilon\Big)$
  - By symmetry of classifiers: $= 2\binom{n}{d}\mathbb{E}\Big[P\Big(\hat{\mathcal{R}}_n(c_1) \le \frac{d}{n} \wedge \mathcal{R}(c_1) > \epsilon \mid X_1,\dots,X_d\Big)\Big]$
  - $\le 2\binom{n}{d}(1-\epsilon)^{n-d} \le 2\binom{n}{d}\exp(-\epsilon(n-d))$

E.g. for scenario 1:

- The hypothesis class consists of indicator functions over intervals $[\ell,\infty)$: $\mathcal{C} = \{I_{[\ell,\infty)}$
- Precisely, $I_{[\ell,\infty)}(x)$ is defined as $I_{[\ell,\infty)}(x) = \begin{cases} 0 & \text{if } x < \ell \\ 1 & \text{if } x \ge \ell \end{cases}$
- Let c^* be a classifier $I_{[\ell^*,\infty)}$, let $X_1, X_2, \dots$ be iid random variables, and let $Y_i = c^*(X_i)$
- Let $X^n_{\min}$ be the minimum value among all X_i that are classified as 1: $\min\{X_i \mid Y_i = 1\}$
- Let there be a threshold ℓ^+_ϵ such that the probability

$P(\ell^* \le X_i < \ell_\epsilon^+) = \epsilon$, i.e. the difference between ℓ^* and ℓ_ϵ^+ represents a normal range of error

- We can show that $P(\ell_\epsilon^+ \le X_{\text{min}}^n) = (1-\epsilon)^n$
  Proof:
  - The event $\ell_\epsilon^+ \le X_{\text{min}}^n$ means that none of the X_i fall into the interval $[\ell^*, \ell_\epsilon^+)$
  - This can be expressed as $P(\ell_\epsilon^+ \le X_{\text{min}}^n) = P(X_i \notin [\ell^*, \ell_\epsilon^+) \text{ for all } i = 1, 2, \dots, n)$
  - Since $X_1, X_2, \dots, X_n$ are iid, we get: $P(\ell_\epsilon^+ \le X_{\text{min}}^n) = \prod_{i=1}^n P(X_i \notin [\ell^*, \ell_\epsilon^+)$
  - Since $P(\ell_\epsilon^+ \le X_i < \ell^*) = \epsilon$, we get: $\prod_{i=1}^n P(X_i \notin [\ell^*, \ell_\epsilon^+) = \prod_{i=1}^n 1 - \epsilon = (1-\epsilon)^n$
- We can show that if $n \ge \frac{1}{\epsilon}\log(\frac{1}{\delta})$, then $P(\ell_\epsilon^+ \le X_{\text{min}}^n) = (1-\epsilon)^n \le \delta$
  Proof:
  - We can reform $n \ge \frac{1}{\epsilon}\log(\frac{1}{\delta})$ to $\exp(n\epsilon) \ge \frac{1}{\delta}$ to $\exp(-n\epsilon) \le \delta$
  - Since generally $\exp(-z) \ge 1 - z$, we can say that $\exp(-n\epsilon) = \exp(-\epsilon)^n \ge (1-\epsilon)^n$
  - Then, $P(\ell_\epsilon^+ \le X_{\text{min}}^n) = (1-\epsilon)^n \le \exp(-\epsilon)^n \le \delta$
- $\mathcal{C}$ is efficiently PAC learnable
  Proof:
  - If algorithm $\mathcal{A}$ runs in time polynomial in $1/\epsilon$ and $1/\delta$
  - $P(R(\hat{c}) > \epsilon) = P(\ell_\epsilon^+ \le X_{\text{min}}^n) \le \delta$
  - This proves that $\mathcal{C}$ is PAC learnable
  - When $n \ge \frac{1}{\epsilon}\log(\frac{1}{\delta})$, $\mathcal{A}$ runs in $\mathcal{O}(n) = \mathcal{O}(\frac{1}{\epsilon}\frac{1}{\delta})$ time

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: $p(\theta|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\theta)p(\boldsymbol{y}|\boldsymbol{X},\theta)}{p(\boldsymbol{y}|\boldsymbol{X})} = \frac{p(\theta)p(\boldsymbol{y}|\boldsymbol{X},\theta)}{\int p(\theta)p(\boldsymbol{y}|\boldsymbol{X},\theta)d\theta} \propto p(\theta)p(\boldsymbol{y}|\boldsymbol{X},\theta) = p(\theta,\boldsymbol{y}|\boldsymbol{X})$
- Focuses on minimizing cost function $\mathbb{E}[k(\theta', \Theta)|\boldsymbol{X},\boldsymbol{y}] = \int_\theta p(\theta|\boldsymbol{X},\boldsymbol{y}) \times k(\theta', \theta) d\theta \propto \int_\theta p(\theta,\boldsymbol{y}|\boldsymbol{X}) \times k(\theta',\theta) d\theta$ resp. $\sum p(\theta|\boldsymbol{X},\boldsymbol{y}) \times k(\theta', \theta)$
- 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: $\hat{\theta} = \mathbb{E}[\theta|\boldsymbol{X},\boldsymbol{y}]$
  Proof:
  - $\mathbb{E}[|\hat{\theta}-\theta|^2|\boldsymbol{y}] = \int (\hat{\theta}-\theta)^2 p(\theta \mid \boldsymbol{y})\, d\theta = \hat{\theta}^2 - 2\hat{\theta}\int \theta p(\theta \mid \boldsymbol{y})\, d\theta + \int \theta^2 p(\theta \mid \boldsymbol{y})\, d\theta$
  - Taking the derivative with respect to $\hat{\theta}$: $\frac{\partial \mathbb{E}[|\hat{\theta}-\theta|^2|\boldsymbol{y}]}{\partial \hat{\theta}} = 2\hat{\theta} - 2\int \theta p(\theta \mid \boldsymbol{y})\, d\theta$
  - Setting the derivative to zero: $\hat{\theta} = \int \theta p(\theta \mid \boldsymbol{y})\, d\theta = \mathbb{E}[\theta \mid \boldsymbol{y}]$
- 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: $\mathbb{E}[|\hat{\theta}-\theta||\boldsymbol{y}] = \int |\hat{\theta}-\theta| p(\theta \mid \boldsymbol{y})\, d\theta$
  - The integral splits into two parts: $= \int_{-\infty}^{\hat{\theta}} (\hat{\theta}-\theta)p(\theta \mid \boldsymbol{y})\, d\theta + \int_{\hat{\theta}}^{\infty} (\theta-\hat{\theta})p(\theta \mid \boldsymbol{y})\, d\theta$
  - Taking the derivative with respect to $\hat{\theta}$: $\frac{\partial \mathbb{E}[|\hat{\theta}-\theta||\boldsymbol{y}]}{\partial \hat{\theta}}$ for each term separately
  - $\frac{\partial}{\partial \hat{\theta}} \int_{-\infty}^{\hat{\theta}} (\hat{\theta}-\theta)p(\theta \mid \boldsymbol{y})\, d\theta = \int_{-\infty}^{\hat{\theta}} p(\theta \mid \boldsymbol{y})\, d\theta - \hat{\theta}p(\hat{\theta} \mid \boldsymbol{y})$
  - $\frac{\partial}{\partial \hat{\theta}} \int_{\hat{\theta}}^{\infty} (\theta-\hat{\theta})p(\theta \mid \boldsymbol{y})\, d\theta = -\int_{\hat{\theta}}^{\infty} p(\theta \mid \boldsymbol{y})\, d\theta - \hat{\theta}p(\hat{\theta} \mid \boldsymbol{y})$
  - Combining the two derivatives, we get: $\int_{-\infty}^{\hat{\theta}} p(\theta \mid \boldsymbol{y})\, d\theta - \int_{\hat{\theta}}^{\infty} p(\theta \mid \boldsymbol{y})\, d\theta$
  - Setting the derivative to zero: $\int_{-\infty}^{\hat{\theta}} p(\theta \mid \boldsymbol{y})\, d\theta = \int_{\hat{\theta}}^{\infty} p(\theta \mid \boldsymbol{y})\, d\theta$
  - Since the total probability is 1, this implies: $\int_{-\infty}^{\hat{\theta}} p(\theta \mid \boldsymbol{y})\, d\theta = 0.5$
- Returns single point estimate

MAP estimator

- Maximizes posterior: $\hat{\theta} = \arg\max_\theta(p(\theta|\boldsymbol{X})) \propto \arg\max_\theta p(\theta|\boldsymbol{X})p(\boldsymbol{X})$
- 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 $0 < v \le 1$
  - $\hat{\theta} = \arg\max_\theta(p(\theta|\boldsymbol{y})) = \arg\max_\theta(p(\boldsymbol{y}|\theta)p(\theta)) = \arg\max_\theta \prod_{j=1}^k p_j^{N_j} \prod_{j=1}^k p_j^v = \arg\max_\theta \prod_{j=1}^k p_j^{N_j+v} = \arg\max_\theta \sum_{j=1}^k (N_j+v)\log(p_j) = \arg\max_\theta \sum_{j=1}^k \frac{N_j+v}{n+kv}\log(\frac{p_j}{(N_j+v)/(n+kv)}) = \arg\min_\theta \sum_{j=1}^k \frac{N_j+v}{n+kv}\log(\frac{(N_j+v)/(n+kv)}{p_j}) = \arg\min_\theta \sum_{j=1}^k \tilde{p}_j \log(\frac{\tilde{p}_j}{p_j})$
  - 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 $\mathcal{R}(f) = \mathbb{E}_{X,Y}[1 f(X) \neq Y]$, 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 Taxonomy

Active Learning

Active learning —

- Assume:
  - Domain space $\mathcal{X}$
  - Sample space $S \subseteq \mathcal{X}$
  - Labeled data $D_{n-1}(x_i, y_i)_{i<n}$
  - Target space $\mathcal{A} \subseteq \mathcal{X}$
  - We estimate $y_x = f_x + \epsilon_x$
- We aim to find the next x_n that gives us the most information about f in $\mathcal{A}$
- Information gain can be quantified as maximizing the conditional mutual information between y_x and f: $IG[f_x; y_x|D_{n-1}] = H(y_x|D_{n-1}) - H(y_x|f_x, D_{n-1})$ where $H(y_x|D_{n-1})$ is the uncertainty about y_x before labeling x_n and $H(y_x|f_x, D_{n-1})$ is the uncertainty about y_x after labeling x_n. We want to minimize the latter, i.e. we want to maximize the delta between the former and the latter
- We pick $x_n = \arg\max_{x \in S} IG[f_x; y_x|D_{n-1}]$
- To find a closed-form solution, we assume that f is a Gaussian process with a known mean and kernel function:
  - $f \sim \mathcal{GP}(\mu, k)$
  - $f = (f_{x_1}, f_{x_2}, \ldots) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ where elements in mean vector are $\mu_i = \mu(x_i)$ and elements in covariance matrix are $\Sigma_{ij} = k(x_i, x_j)$
- Under this assumption, we can show that $IG[f_x; y_x|D_{n-1}] = \frac{1}{2}log(\frac{\mathbb{V}(y_x|D_{n-1})}{\mathbb{V}(y_x|f_x, D_{n-1})})$
  Proof:
  - Gaussian entropy of $a|B$ given by: $H(a|B) = -\int p(a|B)log(p(a|B))da = -\mathbb{E}[log\mathcal{N}(\mu, \sigma)] = -\mathbb{E}[log((2\pi\sigma^2)^{-1/2}exp(-\frac{(x-\mu)^2}{2\sigma^2}))] = \frac{1}{2}log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\mathbb{E}[(x-\mu)^2] = \frac{1}{2}log(2\pi\sigma^2) + \frac{1}{2} \times 1 = \frac{1}{2}log(2\pi\sigma^2) + \frac{1}{2}log(e) = \frac{1}{2}log(2\pi e\sigma^2)$
  - Plugging this in, we get: $H(y_x|D_{n-1}) - H(y_x|f_x, D_{n-1}) = \frac{1}{2}log(2\pi e\mathbb{V}(y_x|D_{n-1})) - \frac{1}{2}log(2\pi e\mathbb{V}(y_x|f_x, D_{n-1})) = \frac{1}{2}log(\frac{\mathbb{V}(y_x|D_{n-1})}{\mathbb{V}(y_x|f_x, D_{n-1})})$

Batch active learning —

- Variant of active learning
- Assume:
  - Domain space $\mathcal{X}$ and distribution P over $\mathcal{X}$
  - Oracle to unknown function $f: \mathcal{X} \to \mathcal{Y}$
  - Population set $\mathcal{X} = \{x_1, \ldots, x_m\} \subseteq \mathcal{X}$
  - Budget $b \le m$
- We aim to find next batch of data points $L \subseteq \mathcal{X}$ subject to $|L| = b$ that gives us the most information
- Suppose we know $Z = \{(x, f(x)) : x \in L\}$
- 1-nearest-neighbor classifier $\hat{f}$ is fitted to Z
- Let $B_\delta(x) = \{x' \in \mathcal{X} : \|x - x'\| \le \delta\}$ be the set of sufficiently close points to x
  - We consider $B_\delta(x)$ pure if f yields same results for all of $B_\delta(x)$
- Impurity of δ is given by $\hat{\pi}(\delta) = P(\{x \in \mathcal{X} : B_\delta(x) \text{ is not pure}\})$
- Let $C(L, S) = \bigcup_{x \in L} B_\delta(x)$ be the union of all sets B
  - $C = C_r \cup C_w = \{x \in C : \hat{f}(x) = f(x)\} \cup \{x \in C : \hat{f}(x) \neq f(x)\}$

- We have $C_w \subseteq \{x \in \mathcal{X} : B_\delta(x) \text{ is not pure}\}$. Then, $P(C_w) \leq \hat{\pi}(\delta)$
- $\{x : \hat{f}(x) \neq f(x)\} \subseteq C_w \cup C_r^C \subseteq \{x \in \mathcal{X} : B_\delta(x) \text{ is not pure}\} \cup C^C$
- Then, we have $\mathcal{R}(\hat{f}) = P(\hat{f}(x) \neq f(x)) \leq \hat{\pi}(\delta) + 1 - P(C)$ due to union bound on RHS
- We need to choose L and δ such that $\mathcal{R}(\hat{f})$ is minimized
- We approach this by minimizing the upper bound, by picking δ and choosing C that maximizes $P(C)$: $\arg\max_{L \subseteq \mathcal{X}, |L|=b} P(\bigcup_{x \in L} B_\delta(x))$
- Two challenges:
  1. We don't know the distribution
  2. Problem is NP-hard
- We address 1) by using the empirical distribution induced by X. Then, we have: $\arg\max_{L \subseteq \mathcal{X}, |L|=b} \frac{1}{|X|} |\{x' : \|x' - x\| \leq \delta, \text{ for some } x \in L\}|$
- We address 2) with greedy algorithm:
  - Input: $x \subseteq \mathcal{X}, b \in \mathbb{N}$
  - Output: $L \subseteq X$ of size b
  1. $G = (x, E)$ where $E = \{(x, x') : \|x - x'\| \leq \delta\}$
  2. $L = \emptyset$
  3. For $i = 1, ..., b$:
     (a) $\hat{x} \leftarrow \arg\max_{x \in \mathcal{X}} |\{x' : (x, x') \in E, x \in \mathcal{X}\}|$
     (b) $L \leftarrow L \cup \hat{x}$
     (c) $E \leftarrow E - (\{\hat{x}\} \times (B_\delta(\hat{x}) \cap x))$
  4. Return L

Safe Bayesian learning —

- Bayesian approach to active learning
- Assume:
  - We have stochastic process f^*
  - We can iteratively choose points $x_1, ..., x_{n-1} \in \mathcal{X}$ and observe $y_i = f^*(x_1), ..., y_{n-1} = f^*(x_{n-1})$
  - Points should lie in safe area S^* which is the set of $x \in \mathcal{X}$ such that another stochastic process $g^*(x) \geq 0$
  - For chosen points, we can also observe $z_i = g^*(x_1), ..., z_{n-1} = g^*(x_{n-1})$ which are measurements of confidence, indicating high confidence when above 0
- We aim to find estimates of sample space S and target space $\mathcal{A}$
- To do so, we fit a Gaussian process on observed $\{(x_i, y_i)\}_{i<n}$ and $\{(x_i, z_i)\}_{i<n}$. Gaussian process over f and g induces two bounds respectively, which provide the 95% confidence interval of $\mathbb{E}[f(x)]$ resp. $\mathbb{E}[g(x)]$:
  - Upper bound function $u_n^f(x)$ resp. $u_n^g(x)$
  - Lower bound function $l_n^f(x)$ resp. $l_n^g(x)$
- Gaussian process over g allows to derive pessimistic and optimistic estimate of safe area:
  - Pessimistic: $S_n = \{x : l_n^g(x) \geq 0\}$
  - Optimistic: $\hat{S}_n = \{x : u_n^g(x) \geq 0\}$
- We then gather estimates, where upper bound of f lies above baseline set by maximum value of lower bound of f: $\mathcal{A}_n = \{x \in \hat{S}_n : u_n^f(x) \geq max_{x' \in S_n} l_n^f(x')\}$
- We can then perform active learning with sample space $S = S_n$ and target space $\mathcal{A} = \mathcal{A}_n$

Ensemble Methods

Motivation —

- Let $\hat{f}_1(x), ..., \hat{f}_B(x)$ be estimators
- When averaging estimators...
  - Average remains unbiased, if all estimators are unbiased
    Proof:
    Bias
    $= \mathbb{E}[\hat{f}(x)] - \mathbb{E}[y|x] = \frac{1}{B}\sum_{i=1}^B \mathbb{E}[\hat{f}_i(x)] - \mathbb{E}[y|x] = \frac{1}{B}\sum_{i=1}^B \text{ bias}(\hat{f}_i(x))$
  - Variance is reduced by a factor of $\frac{1}{B}$, if the estimators have similar variance and no covariance
    Proof:
    - Variance
      $= \mathbb{E}[\hat{f}(x) - \mathbb{E}[\hat{f}(x)]]^2 = \mathbb{E}[\frac{1}{B}\sum_{i=1}^B \hat{f}_i(x) - \frac{1}{B}\sum_{i=1}^B \mathbb{E}[\hat{f}_i(x)]]^2 = \mathbb{E}[\frac{1}{B}\sum_{i=1}^B (\hat{f}_i(x) - \mathbb{E}[\hat{f}_i(x)])]^2 = \frac{1}{B^2}\sum_{i=1}^B \mathbb{V}[\hat{f}_i(x)] + \frac{1}{B^2}\sum\sum_{i \neq j=1}^B \text{ Cov}[\hat{f}_i(x), \hat{f}_j(x)]$
    - Assuming variance are similar ($\approx \sigma^2$) and covariances are small (≈ 0), we get: Variance $= \frac{1}{B^2}\sum_{i=1}^B \sigma^2 = \frac{1}{B^2} B\sigma^2 = \frac{\sigma^2}{B}$

Requirements — Diversity of estimators, to reduce covariance. Achieved by:

- Different subsets of data for each estimator, e.g. via bootstrapping
- Different features for each estimators
- Decorrelating estimators during training

Variants —

- Regression: Average output of all estimators: $\hat{r}_B(x) = \frac{1}{B}\sum_{b=1}^B r_b(x))$
- Classification: Majority or weighted voting: $\hat{c}_B(x) = sgn(\sum_{b=1}^B \alpha_b c_b(x))$ with majority voting if $\alpha = 1$

Bootstrap Aggregating (Bagging) —

- Algorithm
  1. For $b = 1, ..., B$:
     (a) Hold out $\frac{1}{3}$ of sample
     (b) Construct $Z_b^* = b^{th}$ bootstrap sample from remaining $\frac{2}{3}$ of sample
     (c) Construct estimator f_b based on Z_b^*
  2. Return $\hat{f}_B(x) =$ weighted average of $f_1(x), ..., f_B(x)$
  3. Calculate out-of-bag error on held out sample

If desired: Estimators can be constructed in multiple function classes $f', f'', ...$ and set of estimators in the function class, which generates the lowest empirical error, is returned

Random Forest —

- Algorithm:
  1. Generate multiple training sets via bootstrapping
  2. Construct multiple decision trees based on the generated training sets, where each tree selects a random set of features at each split via bootstrapping
     - Classification: Choose $m = \sqrt{k}$ predictors at each split
     - Regression: Choose $m = \frac{k}{3}$ predictors at each split
  3. Generate prediction by averaging or voting on trees
  4. Calculate out-of-bag error
- 2 sources of randomness:
  - Each tree trained on bootstrap sample of instances
  - At each split, bootstrap sample of features is considered

AdaBoost — Algorithm:

1. Each instance weight is initially $w_i = \frac{1}{n}$
2. Train first classifier $\hat{c}^{(1)}$ generating output $\hat{y}^{(1)}$
3. Calculate weighted error rate of j^{th} classifier:
   $$r^{(j)} = \frac{\sum_{i=1}^n w_i \mathbb{I}_{\{\hat{y}_i^{(j)} \neq y_i\}}}{\sum_{i=1}^n w_i}$$
4. Calculate classifier weight, which is higher, if the classifier has a lower error rate: $\alpha^{(j)} = \eta log(\frac{1 - r^{(j)}}{r^{(j)}})$
5. Update instance weights: For $i = 1, ..., n$:
   $$w_i = \begin{cases} w_i & \hat{y}_i^{(j)} = y_i \\ w_i e^{\alpha^{(j)}} & \hat{y}_i^{(j)} \neq y_i \end{cases}$$
6. Normalize instance weights: $w_i = \frac{w_i}{\sum_{i=1}^n w_i}$
7. Continue by training next classifier
8. ...
9. Generate prediction: $\hat{y}(x) = \arg\max_k \sum_{j=1}^B \alpha_j \mathbb{I}_{\{\hat{c}^{(j)}(x)=k\}}$

AdaBoost as an additive logistic model that minimizes the exponential loss function:

- The combined ensemble classifier in AdaBoost is given by: $\sum_{j=1}^B \alpha_j \hat{c}^{(j)}(x) = F(x)$
- Exponential loss function: $\mathbb{E}[e^{-yF(x)}] = p(y = 1|x)e^{-F(x)} + p(y = -1|x)e^{F(x)}$
- Minimizer of exponential loss function is the log-odds: $\mathbb{E}[e^{-yF(x)}]$ is minimized at $\frac{\partial \mathbb{E}[e^{-yF(x)}]}{\partial F(x)} = -p(y = 1|x)e^{-F(x)} + p(y = -1|x)e^{F(x)} = 0$
- $\Rightarrow \frac{1}{2} log(\frac{p(y=1|x)}{p(y=-1|x)}) = F(x)$
- We can re-arrange to $p(y = 1|x) = \frac{e^{F(x)}}{e^{-F(x)} + e^{F(x)}}$
  Proof:
  - We first develop what we found for optimal $F(x)$:
    - $\frac{1}{2} log(\frac{p(y=1|x)}{p(y=-1|x)}) = F(x)$
    - $\Rightarrow \frac{p(y=1|x)}{p(y=-1|x)} = e^{2F(x)}$
    - $\Rightarrow p(y = 1|x) = e^{2F(x)} p(y = -1|x)$
  - We now further develop $p(y = -1|x)$:
    - $p(y = -1|x) + p(y = 1|x) = 1$
    - Plugging in what we found above: $\Rightarrow p(y = -1|x) + e^{2F(x)} p(y = -1|x) = 1$
    - $\Rightarrow 1 + e^{2F(x)} = \frac{1}{p(y=-1|x)}$
    - $\Rightarrow p(y = -1|x) = \frac{1}{1 + e^{2F(x)}}$
  - We can plug this back in: $\Rightarrow p(y = 1|x) = e^{2F(x)} \frac{1}{1 + e^{2F(x)}}$
  - $\Rightarrow p(y = 1|x) = \frac{e^{F(x)}}{e^{-F(x)} + e^{F(x)}}$
- Exponential loss minimizer corresponds to the AdaBoost minimizer, since AdaBoost models the final output as a sum of classifiers: $log(\frac{p(y=1|x)}{p(y=-1|x)}) = \sum_{j=1}^B \alpha_j \hat{c}^{(j)}(x) = F(x)$

Discrete AdaBoost as gradient descent on an additive logistic model:

- The combined ensemble classifier in AdaBoost is given by: $\sum_{j=1}^B \alpha_j \hat{c}^{(j)}(x) = F(x)$
- The cost function AdaBoost minimizes is the exponential loss function given by: $J(F) = \mathbb{E}[e^{-yF(x)}]$
- Suppose we consider an improved function $F'(x)$: $F'(x) = F(x) + \alpha c(x)$ where $c(x)$ is a new weak classifier
- To approximate $J(F')$ we can perform a Taylor expansion around 0: $J(F') = \mathbb{E}[e^{-y(F(x) + \alpha c(x))}] = \mathbb{E}[e^{-yF(x)}(1 - y\alpha c(x) + \frac{1}{2}\alpha^2)] + O(\alpha^3)$
- Aim: Select classifier that minimizes a weighted error, by minimizing $J(F')$ wrt $c(x)$:
  - We first in generic terms define weighted expectation:
    - Weights: $w(x, y) = e^{-yF(x)}$
    - Weighted expectation of function $g(x, y)$:

$\mathbb{E}[g(x,y)|x] = \frac{\mathbb{E}[w(x,y)g(x,y)|x]}{\mathbb{E}[w(x,y)|x]}$

- For AdaBoost, we aim to choose $c(x)$ that maximizes $yc(x)$, i.e. the alignment between classifier c and true label $y \in \{-1,1\}$. This is equivalent to minimizing exponential loss
- In $O(\alpha^2)$ this yields: $c(x)^* =$
  - $\arg\min_c \mathbb{E}[1 - y\alpha c(x) + \frac{1}{2}\alpha^2|x]$, based on Taylor expansion for $J(F')$
  - $\arg\max_c \mathbb{E}[yc(x)|x]$, based on above considerations
  - $= \begin{cases} 1 & \text{if } \mathbb{E}[y|x] = 1 \times p(y=1|x) + (-1) \times p(y=-1|x) > 0 \\ -1 & \text{otherwise} \end{cases}$
  - This ensures that the new classifier aligns with the weighted majority vote
- We can approximate the exponential via the quadratic loss. Then, we have: $\frac{1}{2}\mathbb{E}[(y-c(x))^2] - 1 = \mathbb{E}[y^2 - 2yc(x) + c(x)^2]\frac{1}{2} - 1 = \mathbb{E}[1 - 2yc(x) + 1]\frac{1}{2} - 1 = -\mathbb{E}[yc(x)]$
- Then, we have: $c(x)^* =$
  - $\arg\min_c -\mathbb{E}[yc(x)|x]$
  - $\arg\max_c \mathbb{E}[yc(x)|x]$
- Minimizing the quadratic approximation leads to a Newton-like step for choosing $c(x)$, making it a weighted least squares choice of $c(x)$

- Aim: After selecting $c(x)^*$, optimize weight for new classifier, by minimizing $J(F')$ wrt α:
  - $\alpha^* = \arg\min_\alpha \mathbb{E}[e^{-y\alpha c(x)}] = e^{\alpha}\mathbb{E}[\mathbb{I}_{\{y \neq c(x)\}}] + e^{-\alpha}\mathbb{E}[\mathbb{I}_{\{y = c(x)\}}] = \frac{1}{2}log(\frac{1-err}{err})$ where $err = \mathbb{E}[\mathbb{I}_{\{y \neq c(x)\}}]$
- Combination yields AdaBoost update:
  - $F'(x) \leftarrow F(x) + \alpha^* c(x) = F(x) + \frac{1}{2}log(\frac{1-err}{err})$
  - $w(x,y) \leftarrow w(x,y) \times e^{-\alpha^* yc(x)} = w(x,y) \times e^{\alpha^*(2\mathbb{I}_{\{y \neq c(x)\}} - 1)} = w(x,y) \times e^{log(\frac{1-err}{err})\mathbb{I}_{\{y \neq c(x)\}}} \times e^{-\alpha^*}$ where $e^{-\alpha^*}$ is a constant

Gradient Boosting — Algorithm:

1. Train first classifier $\hat{c}^{(0)}$ generating output $\hat{y}^{(0)}$
2. For $t = 1, ..., M$:
3. Compute the negative gradient $g_t(x_i) = [\frac{\partial L(y_i, \hat{c}_t(x_i))}{\partial \hat{c}_t(x_i)}]$
4. Find function h_t that minimizes the negative gradient: $h_t = \arg\min_h \sum_{i=1}^n (-g_t(x_i) - h(x_i))^2$
5. Find parameter β_t that minimizes the loss: $\beta_t = \arg\min_\beta \sum_{i=1}^n L(y_i, \hat{c}_{t-1}(x_i) + \beta h_t(x_i))$
6. Update $\hat{c}$: $\hat{c}(x_i) = \hat{c}_{t-1}(x_i) + \beta_t h_t(x_i)$
7. Continue by training next classifier
8. ...
9. Generate prediction: $\hat{y}(x) = \text{sign}(\hat{c}_M(x_i))$

8 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 $\boldsymbol{\beta}_{(t=0)}$
  3. $\boldsymbol{\beta}_{(t+1)} \leftarrow \boldsymbol{\beta}_{(t)} - \eta\nabla_{\beta} LO|_{\beta=\boldsymbol{\beta}_{(t)}}$
  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 $\boldsymbol{\beta}_{(t=0)}$
  3. Shuffle training data and initialize $i \leftarrow 1$
  4. $\boldsymbol{\beta}_{(t+1)} \leftarrow \boldsymbol{\beta}_{(t)} - \eta\nabla_{\beta} LO \text{ for observation i } |_{\beta=\boldsymbol{\beta}_{(t)}}$
  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 $\eta_1, \eta_2, ...$, typically decreasing over time
    2. Randomly initialize $\boldsymbol{\beta}_{(t=0)}$
    3. $\boldsymbol{\beta}_{(t+1)} \leftarrow \boldsymbol{\beta}_{(t)} - \eta\nabla_{\beta} LO|_{\beta=\boldsymbol{\beta}_{(t)}}$ 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: $\boldsymbol{\beta}_{(t+1)} \leftarrow \boldsymbol{\beta}_{(t)} - \eta\nabla_{\beta} LO|_{\beta=\boldsymbol{\beta}_{(t)}} + \alpha(\boldsymbol{\beta}_{(t)} - \boldsymbol{\beta}_{(t-1)})$
- For stochastic gradient descent, a smoothing step can be added because stochastic gradient descent hovers around desired solution: $\hat{\beta}_{(t+1)} \leftarrow \frac{1}{L+1}\sum_{j=t-L}^{t} \boldsymbol{\beta}_{(t)}$

9 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

Fisher information —

- $\boldsymbol{I} = \mathbb{E}[(\Lambda)^2] = \mathbb{E}[(\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}|\boldsymbol{\theta})))^2] = \mathbb{V}(\frac{\partial log(p(\boldsymbol{x}|\boldsymbol{\theta}))}{\partial\boldsymbol{\theta}})$ where Λ is the score
  Proof:
  $\mathbb{V}(x) = \mathbb{E}[(x - \mathbb{E}(x))^2] = \mathbb{E}[x^2]$ if $\mathbb{E}(x) = 0$, which is the case here, given that the expected score is 0
- For n iid samples, we have $\boldsymbol{I}_n = n \times \boldsymbol{I}$
  Proof:
  - $\mathbb{E}[(\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}|\boldsymbol{\theta})))^2] = \mathbb{E}[(\frac{\partial}{\partial\boldsymbol{\theta}} log(\prod_{i=1}^n p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2] = \mathbb{E}[(\sum_{i=1}^n \frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2] = \mathbb{E}[\sum_{i=1}^n (\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2 + \sum_{i\neq j}(\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})) \times \frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta})))]$
  - $= \mathbb{E}[\sum_{i=1}^n (\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2] + \mathbb{E}[\sum_{i\neq j}(\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta}))] \times \mathbb{E}[\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta})))]$ since cross-terms are independent
  - $= \mathbb{E}[\sum_{i=1}^n (\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2]$ since expected value of score is 0
  - $= \sum_{i=1}^n \mathbb{E}[(\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2] = n \times \mathbb{E}[(\frac{\partial}{\partial\boldsymbol{\theta}} log(p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta})))^2]$
- The smaller the variance of a distribution, the larger the Fisher information, the lower the Rao-Cramer bound, and the lower the MSE

Rao-Cramer bound —

- Shows that there does not exist an asymptotically unbiased parameter estimator
- For each unbiased estimator, $\mathbb{E}[(\hat{\theta} - \theta)^2] \geq \frac{1}{I}$ where I is the Fisher information
- For estimators in general, $\frac{(\frac{\partial}{\partial\theta}\text{bias}+1)^2}{\boldsymbol{I}} + \text{bias}^2 \leq \mathbb{E}[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^2]$, so there is a trade off if the bias derivative is negative and the squared bias is positive, whereby a biased estimator may produce better results than an unbiased estimator

Proof:

- Given Cauchy Schwarz inequality, we can say: $\mathbb{E}[(\Lambda - \mathbb{E}((\Lambda))(\hat{\theta} - \mathbb{E}(\hat{\theta}))]^2 \leq \mathbb{E}[(\Lambda - \mathbb{E}((\Lambda))^2]\mathbb{E}[(\hat{\theta} - \mathbb{E}(\hat{\theta}))^2]$ where Λ is the score
- We know that $\mathbb{E}(\Lambda) = 0$
- Let's look at the LHS of the equation:
  - Since $\mathbb{E}(\Lambda) = 0$, we can simplify to $\mathbb{E}[\Lambda(\hat{\theta} - \mathbb{E}(\hat{\theta}))] = \mathbb{E}[\Lambda\hat{\theta}] - \mathbb{E}[\Lambda]\mathbb{E}[\hat{\theta}] = \mathbb{E}[\Lambda\hat{\theta}] - 0$
  - This can be developed to: $\mathbb{E}[\Lambda\hat{\theta}] = \int p(\boldsymbol{x}|\theta)\frac{\frac{\partial}{\partial\theta}p(\boldsymbol{x}|\theta)}{p(\boldsymbol{x}|\theta)}\hat{\theta}dx = \frac{\partial}{\partial\theta}(\int p(\boldsymbol{x}|\theta)\hat{\theta}dx - \theta) + 1$ where the last part $(-\theta) + 1$) can be added, because $\frac{\partial}{\partial\theta} - \theta = -1$ and we compensate this with $+1$
  - This is equal to the derivative of the bias + 1: $\frac{\partial}{\partial\theta}(\int p(\boldsymbol{x}|\theta)\hat{\theta}dx - \theta) + 1 = \frac{\partial}{\partial\theta}(\mathbb{E}[\hat{\theta}] - \theta) + 1 = \frac{\partial}{\partial\theta}\text{bias} + 1$
- Let's look at the RHS of the equation: Since $\mathbb{E}(\Lambda) = 0$, first term is $\mathbb{E}(\Lambda^2) = \boldsymbol{I}$ where $\boldsymbol{I}$ is the Fisher information
- Then, we have: $(\frac{\partial}{\partial\theta}\text{bias} + 1)^2 \leq \boldsymbol{I} \times \mathbb{E}[(\hat{\boldsymbol{\theta}} - \mathbb{E}(\hat{\boldsymbol{\theta}}))^2] = \boldsymbol{I} \times \mathbb{E}[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta} - \mathbb{E}(\hat{\boldsymbol{\theta}}) + \boldsymbol{\theta})^2] = ... = \boldsymbol{I} \times \mathbb{E}[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^2] - \text{bias}^2$
- Then, we have $\frac{(\frac{\partial}{\partial\theta}\text{bias}+1)^2}{\boldsymbol{I}} + \text{bias}^2 \leq \mathbb{E}[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^2]$

Bias Variance Tradeoff

- Mean squared error $\mathbb{E}[(\hat{f}(\boldsymbol{X}) - y)^2]$ can be decomposed into: $(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}))^2 + \mathbb{V}(\hat{f}(\boldsymbol{X})) + \mathbb{E}[\epsilon^2] = \text{bias}^2 + \text{variance} + \text{irreducible error}$
  Proof:
  - $y = f(\boldsymbol{X}) + \epsilon$
  - $\mathbb{E}[(\hat{f}(\boldsymbol{X}) - y)^2] = \mathbb{E}[(\hat{f}(\boldsymbol{X}) - \mathbb{E}[\hat{f}(\boldsymbol{X})] + \mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}) + \epsilon)^2] = \mathbb{E}[(\hat{f}(\boldsymbol{X}) - \mathbb{E}[\hat{f}(\boldsymbol{X})])^2] + \mathbb{E}[(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}))^2] + \mathbb{E}[\epsilon^2] - 2\mathbb{E}[(\hat{f}(\boldsymbol{X}) - \mathbb{E}[\hat{f}(\boldsymbol{X})])(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}) + \epsilon)]$
  - 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 $2\mathbb{E}[(\hat{f}(\boldsymbol{X}) - \mathbb{E}[\hat{f}(\boldsymbol{X})])(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}) + \epsilon)]$ equals 0:
    - $2\mathbb{E}[(\hat{f}(\boldsymbol{X}) - \mathbb{E}[\hat{f}(\boldsymbol{X})])(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}) + \epsilon)] = 2(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}) + \epsilon)\mathbb{E}[(\hat{f}(\boldsymbol{X}) - \mathbb{E}[\hat{f}(\boldsymbol{X})])]$ because $(\mathbb{E}[\hat{f}(\boldsymbol{X})] - f(\boldsymbol{X}) + \epsilon)$ is deterministic
    - In last equation, second term equals 0, so whole equation is 0
  - Then, we are left with: $\text{variance} + \text{bias}^2 + \text{irreducible error}$
- *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 $\boldsymbol{X}$, 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 $n \rightarrow \infty \approx \frac{1}{3}$
  - Probability that (x_i, y_i) is in sample $\mathcal{Z}^{*b}$ of size n is given by $1 - \frac{1}{e}$ as $n \rightarrow \infty \approx \frac{2}{3}$
  - We then define: $\hat{\mathcal{R}}^{(0.632)} = 0.368\hat{\mathcal{R}}(\mathcal{A}) + 0.632\hat{\mathcal{R}}^{bs}$

10 Estimating Common Distributions

Gaussian

Frequentism (MLE) —

- Likelihood (excl. constants):
  $L = (\frac{1}{\sigma})^n \prod_{i=1}^{n} exp(-\frac{1}{2\sigma^2}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2) = \frac{1}{\sigma^n} exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2) = \frac{1}{\sigma^n} exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}} + \overline{\boldsymbol{x}} - \boldsymbol{\mu})^2) = \frac{1}{\sigma^n} exp(-\frac{\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})^2}{2\sigma^2}) exp(-\frac{n(\overline{\boldsymbol{x}} - \boldsymbol{\mu})^2}{2\sigma^2}) = \frac{1}{\sigma^n} exp(-\frac{nS^2}{2\sigma^2}) exp(-\frac{n(\overline{\boldsymbol{x}} - \boldsymbol{\mu})^2}{2\sigma^2})$
  where $S = \frac{1}{n}(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})^2$ is the covariance matrix
- Log-likelihood:
  $LL = -nlog(\sigma) - \sum_{i=1}^{n}(\frac{1}{2\sigma^2}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2) = -nlog(\sigma) - \frac{nS^2}{2\sigma^2} - \frac{n(\overline{\boldsymbol{x}} - \boldsymbol{\mu})^2}{2\sigma^2}$
- $\boldsymbol{\mu}_{MLE}$ is sample mean: $\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)}$:
  - Derivative of log-likelihood wrt μ:
    $\nabla_{\boldsymbol{\mu}} LL = \nabla_{\boldsymbol{\mu}}(-\sum_{i=1}^{n}(\frac{\boldsymbol{x}^{(i)^2} - 2\boldsymbol{x}^{(i)}\boldsymbol{\mu} + \boldsymbol{\mu}^2}{2\sigma^2})) = \nabla_{\boldsymbol{\mu}}(-\sum_{i=1}^{n}(-\frac{\boldsymbol{x}^{(i)}\boldsymbol{\mu}}{\sigma^2} + \frac{\boldsymbol{\mu}^2}{2\sigma^2})) = -\sum_{i=1}^{n}(-\frac{\boldsymbol{x}^{(i)}}{\sigma^2} + \frac{2\boldsymbol{\mu}}{2\sigma^2}) = \sum_{i=1}^{n}(\frac{\boldsymbol{x}^{(i)} - \boldsymbol{\mu}}{\sigma^2}) = \sum_{i=1}^{n} \boldsymbol{x}^{(i)} - n\boldsymbol{\mu} = 0$
- $\boldsymbol{\mu}_{MLE}$ is an unbiased estimator:
  - $\mathbb{E}[\boldsymbol{\mu}_{MLE}] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[\boldsymbol{x}_i] = \boldsymbol{\mu}$
- $\sigma^2{}_{MLE}$ is sample variance: $\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2$:
  - Derivative of log-likelihood wrt σ:
    $\nabla_{\boldsymbol{\sigma}} LL = -n\nabla_{\boldsymbol{\sigma}} log(\boldsymbol{\sigma}) - \nabla_{\boldsymbol{\sigma}}(\sum_{i=1}^{n}(\frac{(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2}{2\sigma^2})) = \frac{-n}{\boldsymbol{\sigma}} - \nabla_{\boldsymbol{\sigma}}(\sum_{i=1}^{n} \frac{1}{2}\boldsymbol{\sigma}^{-2}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2) = \frac{-n}{\boldsymbol{\sigma}} - (\sum_{i=1}^{n} -1\boldsymbol{\sigma}^{-3}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2) = -n + \sum_{i=1}^{n}(\frac{(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^2}{\sigma^2}) = 0$
- $\sigma^2{}_{MLE}$ is a biased estimator:
  - $\mathbb{E}[\Sigma_{MLE}] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\left(\boldsymbol{x}_i - \boldsymbol{\mu}_{MLE}\right)\left(\boldsymbol{x}_i - \boldsymbol{\mu}_{MLE}\right)^\top\right]$
  - $= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_i^\top\right] - \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{\mu}_{MLE}^\top\right] - \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\boldsymbol{\mu}_{MLE}\boldsymbol{x}_i^\top\right] + \mathbb{E}\left[\boldsymbol{\mu}_{MLE}\boldsymbol{\mu}_{MLE}^\top\right]$
  - $= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_i^\top\right] - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_j^\top\right] - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_j^\top\right] + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_j^\top\right]$
  - $= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_i^\top\right] - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_j^\top\right]$
  - $= \frac{1}{n}n(\Sigma + \boldsymbol{\mu}\boldsymbol{\mu}^\top) - \frac{1}{n^2}\left(n^2\boldsymbol{\mu}\boldsymbol{\mu}^\top + n\Sigma\right)$
    Proof:
    - $\mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_j^\top\right] = \delta_{ij}\Sigma + \boldsymbol{\mu}\boldsymbol{\mu}^\top$ where $\delta = 1$ if $i = j$
      Proof:
      - $\Sigma = \mathbb{E}[(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top] = \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_i^\top - 2\boldsymbol{x}_i\boldsymbol{\mu}^\top + \boldsymbol{\mu}\boldsymbol{\mu}^\top\right] = \mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_i^\top\right] - \boldsymbol{\mu}\boldsymbol{\mu}^\top$
      - For $i \neq j$, covariance is 0: $\mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_j^\top\right] = \mathbb{E}[\boldsymbol{x}_i]\mathbb{E}[\boldsymbol{x}_j] = \boldsymbol{\mu}\boldsymbol{\mu}^\top$
    - $... + n\Sigma$ since in n cases $\delta = 1$
  - $= \Sigma - \frac{1}{n}\Sigma$
  - $\mathbb{E}[\Sigma_{MLE}] = \Sigma - \frac{1}{n}\Sigma$

Bayesianism — Univariate:

- Assume σ^2 is known and $\mu \sim \mathcal{N}(\mu_0, \sigma_0^2)$ is the outcome of a random variable
- $p(\mu|\boldsymbol{x}, \mu_0, \sigma_0^2) \propto p(\boldsymbol{x}|\mu, \sigma^2)p(\mu|\mu_0, \sigma_0^2)$
- The likelihood is $p(\boldsymbol{x}|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$
- 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:
  $p(\mu|\boldsymbol{x}, \mu_0, \sigma_0^2) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right)$
- 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: $p(\mu|\boldsymbol{x}, \mu_0, \sigma_0^2) \propto \exp\left(-\frac{1}{2}\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 + \left(\frac{\sum_{i=1}^{n} x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right)\mu + (\text{constant terms})\right)$ 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:
  - $\mu_n = \frac{\frac{\sum_{i=1}^{n} x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{\frac{n\overline{x}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{\frac{n\overline{x}\sigma_0^2 + \mu_0\sigma^2}{\sigma^2\sigma_0^2}}{\frac{n\sigma_0^2 + \sigma^2}{\sigma^2\sigma_0^2}} = \frac{n\overline{x}\sigma_0^2 + \mu_0\sigma^2}{n\sigma_0^2 + \sigma^2}$
  - $\sigma_n = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{1}{\frac{n\sigma_0^2 + \sigma^2}{\sigma^2\sigma_0^2}} = \frac{\sigma^2\sigma_0^2}{n\sigma_0^2 + \sigma^2}$
- Thus, the posterior is $p(\mu|\boldsymbol{x}, \mu_0, \sigma_0^2) \sim \mathcal{N}(\mu_n, \sigma_n^2)$

Multivariate:

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

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 $\alpha_1 =$ number of successes, $\alpha_2 =$ number of failures
- $\hat{p}_{\mathrm{MLE}} = \arg\max(p^{\alpha_1}(1-p)^{\alpha_2}) = \alpha_1/(\alpha_1 + \alpha_2)$ 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 $\alpha_1 =$ number of successes, $\alpha_2 =$ 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:
- $P(p|\delta) \sim \mathrm{Beta}(\alpha_1 + \beta_1, \alpha_2 + \beta_2)$
- $P(p|\delta) = \frac{\Gamma(\alpha_1+\beta_1+\alpha_2+\beta_2)}{\Gamma(\alpha_1+\beta_1)\Gamma(\alpha_2+\beta_2)} p^{\alpha_1+\beta_1-1}(1-p)^{\alpha_2+\beta_2-1}$

MAP:

- Posterior $P(p|\delta)$ has a beta distribution: $P(p|\delta) = \frac{\Gamma(\alpha_1+\beta_1+\alpha_2+\beta_2)}{\Gamma(\alpha_1+\beta_1)\Gamma(\alpha_2+\beta_2)} p^{\alpha_1+\beta_1-1}(1-p)^{\alpha_2+\beta_2-1}$
- $\hat{p}_{\mathrm{MAP}} = \frac{\alpha_1+\beta_1-1}{\alpha_1+\beta_1+\alpha_2+\beta_2-2}$ after logarithmizing and finding local minimum
- Thus, if our prior belief $P(p) \sim \mathrm{Beta}(\beta_1, \beta_2)$ 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 $p_{\mathrm{MLE}} = p_{\mathrm{MAP}}$

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 $\frac{\partial \log(P(x|\lambda))}{\partial \lambda} = \frac{\sum_{i=1}^{n} x_i}{\lambda} - n = 0$
- $\Rightarrow \hat{\lambda}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Log likelihood is concave, so $\hat{\lambda}_{\mathrm{MLE}}$ 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: $P(\lambda|x) \sim \mathrm{Beta}(\alpha', \beta')$
- 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 $\frac{\partial \log(P(\lambda|x))}{\partial \lambda} = \frac{\alpha + \sum_{i=1}^{n} x_i - 1}{\lambda} - (\beta + n) = 0$
- $\Rightarrow \hat{\lambda}_{\mathrm{MAP}} = \frac{\alpha + \sum_{i=1}^{n} x_i - 1}{\beta + n}$
- This corresponds to the posterior mean, since the mean of a Gamma distribution is α/β

11 Linear Regression

Description

Task — Regression

Description —

- Supervised
- Parametric

Formulation

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

Optimization

Parameters — Find parameters $\boldsymbol{\beta}$

Objective function — *Ordinary least squares estimator (OLSE)*:

- Minimize mean squared error: $LO = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)})^2$ resp. $LO = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$

Optimization —

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

Alternatives to OLSE — MLE:

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

Orthogonality principle:

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

Pseudo Inverse:

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

PCA:

- Instances $y^{(i)}, \boldsymbol{x}^{(i)} = \xi^{(i)}$ can be projected onto hyperplane given by $\boldsymbol{X}\boldsymbol{\beta}$
- 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 $\boldsymbol{y}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{y}, \sigma^2\boldsymbol{I}) = \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I})$
- Let $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{X}^+\boldsymbol{y}$ be the OLSE where $\boldsymbol{X}^+$ is a scalar
- Then, $\hat{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{X}^+\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{X}^{+\top}\sigma^2\boldsymbol{X}^+) = \mathcal{N}(\boldsymbol{\beta}, (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\sigma^2)$
  Proof:
  - $\mathcal{N}(\boldsymbol{X}^+\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{X}^{+\top}\sigma^2\boldsymbol{X}^+) = \mathcal{N}(\boldsymbol{I}\boldsymbol{\beta}, \sigma^2\boldsymbol{X}^+\boldsymbol{X}^{+\top})$ since $\boldsymbol{X}^+$ is a scalar
  - Further, we have $\mathcal{N}(\boldsymbol{I}\boldsymbol{\beta}, \sigma^2\boldsymbol{X}^+((\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top)^\top) = \mathcal{N}(\boldsymbol{\beta}, \sigma^2\boldsymbol{X}^+\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1\top}) = \mathcal{N}(\boldsymbol{\beta}, \sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1})$ since $(\boldsymbol{X}^\top\boldsymbol{X})$ is symmetric
- We can estimate σ^2 unbiasedly as: $\hat{\sigma}^2 = \frac{1}{n-m}\sum_{i \leq n}(\boldsymbol{X}\hat{\boldsymbol{\beta}} - \boldsymbol{y})^2$
- 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 $\sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$
- 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 $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \epsilon$, we can substitute $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\beta} + \epsilon) = \boldsymbol{\beta} + (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\epsilon$
  - Taking the expected value on both sides, we have: $\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta} + (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\mathbb{E}(\epsilon)$
  - Then, $\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$ 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 $\hat{\boldsymbol{\beta}} = \boldsymbol{A}^\top\boldsymbol{y} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$ be the OLSE
  - Let $\boldsymbol{C}^\top\boldsymbol{y}$ be another unbiased estimator
  - $\mathbb{V}(\hat{\boldsymbol{\beta}}) = \mathbb{V}(\boldsymbol{A}^\top\boldsymbol{y}) = \boldsymbol{A}^\top\mathbb{V}(\boldsymbol{y})\boldsymbol{A}$ since $\boldsymbol{A}$ 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

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 $x^\top A x + 2x^\top b = (x + A^{-1}b)^\top A(x + A^{-1}b) - b^\top A^{-1} b$, 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 $p(\beta|X, y)$:
  - $p(\beta|X, y) \propto p(y|X, \beta) \times p(\beta) \propto \log(p(y|X, \beta) \times p(\beta)) \propto \log(exp(-\frac{1}{2\sigma^2}(y - X\beta)^\top (y - X\beta)) \times exp(-\frac{1}{2T^2}\beta^\top \beta)) \propto -\frac{\|y - X\beta\|^2}{\sigma^2} - \frac{\|\beta\|^2}{\tau^2}$
- 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 $\frac{\partial(y - X\beta)^\top (y - X\beta)}{\partial\beta} + 2\lambda\beta = 0$
  - Then, $\beta^* = -\frac{1}{2\lambda}\frac{\partial(y - X\beta)^\top (y - X\beta)}{\partial\beta}$
  - 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 $X = USV^\top$
  - We can show that $X\beta = US\left(S^2 + \lambda I\right)^{-1} SU^\top Y$
    Proof:
    - $X\beta^r = X(X^\top X + \lambda I)^{-1} X^\top Y$
    - $= UDV^\top ((UDV^\top)^\top UDV^\top + \lambda I)^{-1} (UDV^\top)^\top Y$
    - $= UDV^\top (VDU^\top UDV^\top + \lambda I)^{-1} VDU^\top Y$
    - $= UDV^\top \left(VD^2V^\top + \lambda VV^\top\right)^{-1} VDU^\top Y$
    - $= UDV^\top \left(V(D^2 + \lambda I)V^\top\right)^{-1} VDU^\top Y$
    - $= UDV^\top V\left(D^2 + \lambda I\right)^{-1} V^\top VDU^\top Y$
    - $= UD\left(D^2 + \lambda I\right)^{-1} DU^\top Y$
  - Similarly, we can show that $\|\beta\|^2 = Y^\top US^2\left(S^2 + \lambda I\right)^{-2} U^\top Y = \frac{W^\top S^2 W}{(S^2 + \lambda I)^2}$ where $W = U^\top Y$
  - 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 $(X^\top X + \lambda I)$ has linearly independent columns
- Can be solved analytically, as $(X^\top X + \lambda I)$ 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: $LO = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \beta \cdot \boldsymbol{x}^{(i)})^2 + \lambda(|\beta| - t)$ resp. $LO = (\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) + \lambda(|\beta| - t)$

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 $p(\boldsymbol{y}|\boldsymbol{X}, \beta)$:
  - Conditional on $\beta, \boldsymbol{y} \propto \mathcal{N}(\boldsymbol{X}\beta, \sigma^2\boldsymbol{I}_n)$
  - $p(\boldsymbol{y}|\boldsymbol{X}, \beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta)\right)$
- Posterior $p(\beta|\boldsymbol{X}, \boldsymbol{y})$:
  - $p(\beta|\boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{X}, \beta) \times p(\beta)$
  - $\log p(\beta|\boldsymbol{X}, \boldsymbol{y}) \propto$ $\log\left(\exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta)\right) \times \prod_{j=1}^{m} \exp\left(-\frac{|\beta_j|}{b}\right)\right)$
  - $\log p(\beta|\boldsymbol{X}, \boldsymbol{y}) \propto -\frac{\|\boldsymbol{y}-\boldsymbol{X}\beta\|^2}{\sigma^2} - \frac{\|\beta\|_1}{b}$
- If we maximize log posterior (MAP estimate), we have: $\arg\min_\beta \frac{\|\boldsymbol{y}-\boldsymbol{X}\beta\|^2}{\sigma^2} + \frac{\|\beta\|_1}{b} = \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \frac{\sigma^2}{b}\|\beta\|_1$
- 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 $\frac{\partial(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)}{\partial\beta} + \frac{\partial\lambda|\beta|}{\partial\beta} = 0$
  - $\frac{\partial\lambda|\beta|}{\partial\beta}$ 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 $-\lambda < \frac{\partial(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)}{\partial\beta} < \lambda$ 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

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

Optimization

Parameters — Find parameters β

Objective function —

- Ordinary least squares estimator
- Minimize mean squared error

Optimization —

- $\nabla_\beta LO = 0$
- $\Rightarrow \beta = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y}$

Characteristics —

- Convex with psd Hessian
- Has global minimum
- Has unique or infinitely many solutions
- Can be solved analytically, if $(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})$ 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 $\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}$ and a feature map $\varphi : \mathbb{R}^m \rightarrow \mathbb{R}^k$ we can define an inner product on $\mathbb{R}^k$ via the kernel function: $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(\boldsymbol{x}^{(j)})$
- 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 $span(\varphi(\boldsymbol{x}^{(i)}), ..., \varphi(\boldsymbol{x}^{(N)})) = \mathbb{R}^k$ 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 $\varphi(\boldsymbol{z})$ 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: $\boldsymbol{x}^\top\boldsymbol{K}\boldsymbol{x} \geq 0$ where $\boldsymbol{K}$ 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 $\varphi : \mathbb{R}^m \rightarrow \mathbb{R}^k$
- 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: $\beta \cdot \varphi(\boldsymbol{z}) = \sum_{i=1}^{n} \alpha^{(i)}\varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(\boldsymbol{z}) = \sum_{i=1}^{n} \alpha^{(i)}k(\boldsymbol{x}^{(i)}, \boldsymbol{z})$
- Act of prediction becomes act of measuring similarity to training instances in feature map space

Kernel Types

Polynomial kernel —

- $\varphi(\boldsymbol{x}) = [x^\alpha]_{\alpha\in\mathbb{N}^m}$ 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 —

- Gives access to infinite feature space
- $\varphi(\boldsymbol{x}) = exp(-\frac{1}{2}\|\boldsymbol{x}\|^2)[\frac{\boldsymbol{x}^\alpha}{\sqrt{\alpha!}}]_{\alpha\in\mathbb{N}^m}$
- $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sigma^2 exp(-\frac{\|\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(j)}\|^2}{2l^2})$
  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, $\sum_\alpha[\frac{\boldsymbol{x}^{(i)\alpha}\boldsymbol{x}^{(j)\alpha}}{\alpha!}] = exp(\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)})$
  - $exp(-\frac{1}{2}\|\boldsymbol{x}^{(i)}\|^2 - \frac{1}{2}\|\boldsymbol{x}^{(j)}\|^2 + \boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}) = exp(-\frac{\|\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(j)}\|^2}{2\sigma^2})$
- 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 $r \rightarrow \infty$ 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: $\beta = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}$
  - Prediction for novel instance: $\beta \cdot \varphi(\boldsymbol{z}) = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y} \cdot \varphi(\boldsymbol{z}) = \boldsymbol{y}^\top \boldsymbol{\Phi} (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \varphi(\boldsymbol{z})$
- Let us define $K = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ 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: $\beta = \boldsymbol{\Phi}^\top \boldsymbol{K}^{-1} \boldsymbol{y}$
    Proof:
    - $(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{I}_D)\beta = \boldsymbol{\Phi}^\top \boldsymbol{y}$
    - $\Rightarrow \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \beta + \lambda \boldsymbol{I}_D \beta = \boldsymbol{\Phi}^\top \boldsymbol{y}$
    - $\Rightarrow \boldsymbol{I}_D \beta = \boldsymbol{\Phi}^\top \lambda^{-1} (\boldsymbol{y} - \boldsymbol{\Phi}\beta)$
    - Since we know from the representer theorem that $\beta = \boldsymbol{\Phi}^\top \alpha$, we can say: $\alpha = \lambda^{-1}(\boldsymbol{y} - \boldsymbol{\Phi}\beta)$
    - We can further develop this to: $\lambda \alpha = (\boldsymbol{y} - \boldsymbol{\Phi}\beta)$
    - Replacing β by $\boldsymbol{\Phi}^\top \alpha$ yields: $\lambda \alpha = (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \alpha)$
    - $\Rightarrow \alpha = (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{y} = \boldsymbol{K}^{-1} \boldsymbol{y}$
    - With this, we can calculate the parameters: $\beta = \boldsymbol{\Phi}^\top \alpha = \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{y} = \boldsymbol{\Phi}^\top \boldsymbol{K}^{-1} \boldsymbol{y}$

    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 $E = \boldsymbol{I}_D, F = \Phi^\top, G = \boldsymbol{\Phi}, H = \boldsymbol{I}_N$, the formula simplifies to $(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{I}_D)^{-1} \boldsymbol{\Phi}^\top = \boldsymbol{I}_D^{-1} \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \boldsymbol{I}_N)^{-1}$
    - Since $\boldsymbol{I}_D^{-1} = \boldsymbol{I}_D$, we have: $(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{I}_D)^{-1} \boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \boldsymbol{I}_N)^{-1}$
    - $\Rightarrow (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{I}_D)^{-1} \boldsymbol{\Phi}^\top y = \hat{\beta} = \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \boldsymbol{I}_N)^{-1} y$
  - Prediction for novel instance: $\beta \cdot \varphi(\boldsymbol{z}) = \boldsymbol{y}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1} \boldsymbol{\Phi} \varphi(\boldsymbol{z}) = \boldsymbol{y}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1} \boldsymbol{k}$ where $\boldsymbol{k} = \boldsymbol{\Phi}\varphi(\boldsymbol{z}) = [k(\boldsymbol{x}^{(1)}, \boldsymbol{z}), ..., k(\boldsymbol{x}^{(n)}, \boldsymbol{z})]^\top = [\varphi(\boldsymbol{x}^{(1)}) \cdot \varphi(\boldsymbol{z}), ..., \varphi(\boldsymbol{x}^{(n)}) \cdot \varphi(\boldsymbol{z})]^\top$ 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 $\alpha = \boldsymbol{K}^{-1} \boldsymbol{y}$ 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 $\boldsymbol{k}$
   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 Gaussian Processes

Description

Task — Models a distribution over functions

Description —

- Supervised
- Non-parametric

Formulation

- $y^{(i)} = \beta \cdot \boldsymbol{x}^{(i)} + \epsilon$ resp. $\boldsymbol{y} = \boldsymbol{X}\beta + \epsilon$
- $\beta \sim \mathcal{N}(0, \boldsymbol{\Lambda}^{-1})$
- $\epsilon \sim \mathcal{N}(0, \sigma \boldsymbol{I}_m)$

Optimization

Optimization —

- If we compute the moment of the Gaussian:
  - $\mathbb{E}[\boldsymbol{y}] = \boldsymbol{X}\mathbb{E}(\beta) = \boldsymbol{X}0 = 0$
  - $\text{Cov}(\boldsymbol{y}) = \mathbb{E}[(\boldsymbol{X}\beta + \epsilon)(\boldsymbol{X}\beta + \epsilon)^\top] = \boldsymbol{X}\mathbb{E}(\beta\beta^\top)\boldsymbol{X}^\top + \boldsymbol{X}\mathbb{E}(\beta)\mathbb{E}(\epsilon^\top) + \mathbb{E}(\epsilon)\mathbb{E}(\beta^\top)\boldsymbol{X} + \mathbb{E}(\epsilon\epsilon^\top)$ where
    - $\mathbb{E}(\beta\beta^\top) = \mathbb{V}(\beta)$ and $\mathbb{E}(\epsilon\epsilon^\top) = \mathbb{V}(\epsilon)$ because $\mathbb{V}(x) = \mathbb{E}[(x - \mathbb{E}(x))^2] = \mathbb{E}[x^2]$ if $\mathbb{E}(x) = 0$, which is the case here due to the defined distributions
    - $\mathbb{E}(\epsilon) = 0$
  - Plugging in the variance for β and ϵ, we get $\text{Cov}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\Lambda}^{-1}\boldsymbol{X}^\top + \sigma^2 \boldsymbol{I}_m$
  - This can be written as a Kernel matrix $\boldsymbol{K}$: $\begin{bmatrix} K_{1,1} + \sigma^2 & ... & ... & K_{1,n} \\ ... & K_{2,2} + \sigma^2 & ... & ... \\ ... & ... & ... & ... \\ K_{n,1} & ... & ... & K_{n,n} + \sigma^2 \end{bmatrix}$ with $K_{ij} = \boldsymbol{x}^{(i)\top} \boldsymbol{\Lambda}^{-1} \boldsymbol{x}^{(j)}$
  - In this kernel matrix, the kernel function can take any shape
- On this basis, Gaussian process is defined as collection of random variables such that every finite subset of variables is jointly Gaussian: $f \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{K})$
- A new instance follows the distribution $p(y_{n+1}) = \mathcal{N}(\boldsymbol{k}^\top \boldsymbol{C}_n^{-1} \boldsymbol{y}, c - \boldsymbol{k}^\top \boldsymbol{C}_n^{-1} \boldsymbol{k})$ where
  - $\boldsymbol{k} = k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n+1)}), ..., k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n+1)})]^\top = [\varphi(\boldsymbol{x}^{(1)}) \cdot \varphi(\boldsymbol{x}^{(n+1)}), ..., \varphi(\boldsymbol{x}^{(n)}) \cdot \varphi(\boldsymbol{x}^{(n+1)})]^\top$ is the kernel vector
  - $\boldsymbol{C}_n = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) + \sigma^2 \boldsymbol{I}_m$
  - $c = k(\boldsymbol{x}^{(n+1)}, \boldsymbol{x}^{(n+1)}) + \sigma^2 \boldsymbol{I}_m$

  Proof:
  - We derive the joint distribution $p(\begin{bmatrix} \boldsymbol{y} \\ y_{n+1} \end{bmatrix}) \sim \mathcal{N}[0, \begin{bmatrix} \boldsymbol{C}_n & \boldsymbol{k} \\ \boldsymbol{k}^\top & c \end{bmatrix}]$
  - To obtain a closed-form solution for this, we can make use of the following theorem:
    - Given a joint Gaussian distribution: $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}) \sim \mathcal{N}[\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}]$
    - The conditional Gaussian distribution is given by: $p(a_2 | a_1 = z) = \mathcal{N}(u_2 + \Sigma_{21}\Sigma_{11}^{-1}(z - u_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$
  - Then, we get $p(y_{n+1}) = \mathcal{N}(\boldsymbol{k}^\top \boldsymbol{C}_n^{-1} \boldsymbol{y}, c - \boldsymbol{k}^\top \boldsymbol{C}_n^{-1} \boldsymbol{k})$
- A new instance vector follows the distribution $p(\boldsymbol{y}_*) = \mathcal{N}(\boldsymbol{K}_{*n} \boldsymbol{K}_{nn}^{-1} \boldsymbol{y}_n, \boldsymbol{K}_{**} - \boldsymbol{K}_{*n} \boldsymbol{K}_{nn}^{-1} \boldsymbol{K}_{n*})$ where
  - Subscript $*$ indicates new instances, subscript n indicates old instances
  - $\boldsymbol{K}$is the kernel matrix, e.g. $\boldsymbol{K}_{*n}$ has new instances on rows and old instances on columns

Algorithm —

1. Compute kernel matrix based on observed data
2. Compute kernel vector based on observed data and new instance
3. Calculate mean and variance of predicted distribution
4. Return predicted distribution

Further proofs

Combining Gaussian Processes —

- Consider a Gaussian Process regression setting where $f \sim \text{GP}(m, k_1)$, with the covariance given by some valid kernel k_1
- If we place a prior on the mean $m \sim \text{GP}(0, k_2)$, f follows a GP with mean 0 and covariance $k = k_1 + k_2$:
  - $\mathbb{E}[f] = \mathbb{E}[m] = 0$
  - We can sum the variances of two independent normally distributed random variables
- If we restrict m to $m(x) = a$, where $a \sim \mathcal{N}(0, \sigma_a^2)$, f follows a GP with mean 0 and covariance $k = k_1 + k_2 = k_1 + \sigma_a^2$:
  - $\mathbb{E}[f] = \mathbb{E}[m] = \mathbb{E}[a] = 0$
  - Covariance: $k_2(m(x), m(x')) = \text{Cov}[m(x), m(x')] = \mathbb{E}[m(x)m(x')] - \mathbb{E}[m(x)]\mathbb{E}[m(x')] = \sigma_a^2 - 0$
- If we restrict m to $m(x) = a^\top x + b$, where $a \sim \mathcal{N}(0, \sigma_a^2)$ and $b \sim \mathcal{N}(0, \sigma_b^2)$, f follows a GP with mean 0 and covariance $k = k_1 + k_2 = k_1 + \sigma_a^2 x^\top x' + \sigma_b^2$:
  - $\mathbb{E}[m(x)] = \mathbb{E}[a] = \mathbb{E}[b] = 0$
  - Covariance: $k_2(m(x), m(x')) = \text{Cov}[m(x), m(x')] = \mathbb{E}[m(x)m(x')] - \mathbb{E}[m(x)]\mathbb{E}[m(x')] = \mathbb{E}[(a^\top x + b)(a^\top x' + b)] - 0 = \mathbb{E}[(a^\top x)(a^\top x')] + \mathbb{E}[b^2] + \mathbb{E}[ba^\top(x + x')] = \sigma_a^2 x^\top x' + \sigma_b^2 + 0$ since b and a are independent

19 Kernel SVM

Description

Task — Classification

Description —

- Supervised
- Parametric

Optimization

- Cost function $\arg\min_\beta \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \xi^{(i)}$ resp. dual Lagrangian objective function

$\arg\max_\alpha \sum_{i=1}^n \alpha^{(i)} - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(\boldsymbol{x}^{(j)})$ of soft-margin SVM is eligible for kernel trick

- Thus, $\beta^* = \sum_{i=1}^n \tilde{\alpha}^{(i)}\varphi(\boldsymbol{x}^{(i)})$ where $\tilde{\alpha}$ refers to Kernel theorem
- Given general Lagrangian solution, $\beta^* = \sum_{i=1}^n \alpha^{(i)}y^{(i)}\varphi(\boldsymbol{x}^{(i)})$ where α refers to Lagrange multiplier, we know that: $\tilde{\alpha}^{(i)} = \alpha^{(i)}y^{(i)}$
- Outcome of novel instance can be predicted as: $\beta^* \cdot \varphi(\boldsymbol{z}) = \sum_{i=1}^n \alpha^{(i)}y^{(i)}\varphi(\boldsymbol{x}^{(i)}) \cdot \varphi(\boldsymbol{z}) = \sum_{i=1}^n \alpha^{(i)}y^{(i)}k(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{z})$

Further proofs

SVMs are 1-nearest-neighbor classifiers, if we choose an RBF kernel, i.e. SVM prediction corresponds to label of nearest neighbor of a given instance:

- Consider an instance $\boldsymbol{x}$ with the nearest neighbor $\boldsymbol{x}_p$ and the second-nearest neighbor $\boldsymbol{x}_q$
- Prediction given by $f(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^n \alpha^{(i)}y^{(i)}k(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{x}))$
- Assuming $\alpha^{(i)} = 1$ and plugging in RBF kernel, we get $f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^n y^{(i)} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}^{(i)}\|^2}{h^2}\right)\right)$ $= \text{sign}\left(y_p \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_p\|^2}{h^2}\right) + \sum_{j=1, j\neq p}^n y^{(j)} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}^{(j)}\|^2}{h^2}\right)\right)$
- If we can find conditions on h which guarantee that the first term in this equation is greater than the second, then we have that $f(\boldsymbol{x}) = \text{sign}(y_p)$ and hence the predicted label will be the same as that of the nearest neighbor
- Looking at the second term: $\sum_{j=1, j\neq p}^n y^{(j)} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}^{(j)}\|^2}{h^2}\right) \leq (n-1)\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_q\|^2}{h^2}\right)$ since $\boldsymbol{x}_q$ is the second-nearest neighbor
- Looking at the first term: $\left|y_p \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_p\|^2}{h^2}\right)\right| = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_p\|^2}{h^2}\right)$ since output label $y \in \{-1, 1\}$
- Then, if we want to guarantee that the first term in this equation is greater than the second, we can set: $\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_p\|^2}{h^2}\right) > (n-1)\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_q\|^2}{h^2}\right)$ $\Rightarrow \exp\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_q\|^2-\|\boldsymbol{x}-\boldsymbol{x}_p\|^2}{h^2}\right) > (n-1) \Rightarrow \frac{\|\boldsymbol{x}-\boldsymbol{x}_q\|^2-\|\boldsymbol{x}-\boldsymbol{x}_p\|^2}{\log(n-1)} > h \Rightarrow h_0 > h$
- Hence, there exists a h_0 such that for all RBF parameter values $h < h_0$, we have that $f(\boldsymbol{x}) = \text{sign}(y_p)$ and hence the predicted label will be the same as that of the nearest neighbor

20 Kernel K-Means Clustering

Formulation

- We specify that we want $j = 1, \ldots, k$ clusters in total
- Clusters defined by centroid $\boldsymbol{\mu}^{[j]}$
- Instances $i = 1, \ldots, n$ 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: $\Theta = \sum_{j=1}^k \sum_{\boldsymbol{x}^{(i)} \in C_j} \left[K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) - \frac{2}{|C_j|}\sum_{\boldsymbol{x}^{(\mu)} \in C_j} K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(\mu)}) + \frac{1}{|C_j|^2}\sum_{\boldsymbol{x}^{(\mu)}, \boldsymbol{x}^{(\rho)} \in C_j} K(\boldsymbol{x}^{(\mu)}, \boldsymbol{x}^{(\rho)})\right]$

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

21 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: $\beta \cdot \boldsymbol{x} + \eta$
- 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 $\boldsymbol{x}_1, \boldsymbol{x}_2$ on the decision boundary $z = 0$, i.e. $\beta \cdot \boldsymbol{x}_1 = 0, \quad \beta \cdot \boldsymbol{x}_2 = 0$ Since $\boldsymbol{x}_1, \boldsymbol{x}_2$ are on the decision boundary, vector $\boldsymbol{z}$ can be considered a linear combination of $\boldsymbol{x}_1 - \boldsymbol{x}_2$
    - ⋆ Combining these equations, we get $\beta \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_2) = \beta \cdot u\boldsymbol{z} = 0$

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 —

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

  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:
  - $\mathcal{H}(\beta) = \nabla_\beta^2\left(-y^{(i)}\beta \cdot \boldsymbol{x}^{(i)}\right) = 0$
  - $\mathcal{H} \succcurlyeq 0$
- For first term:
  - $\mathbb{H}(\beta) = \nabla_\beta^2 \ln(1 + e^{\beta \cdot \boldsymbol{x}^{(i)}})$
  - $= v(x) \times u'(x) - v'(x) \times u(x) = \frac{1}{1+e^{\beta \cdot \boldsymbol{x}}} \times e^{\beta \cdot \boldsymbol{x}} \times \boldsymbol{x}\boldsymbol{x}^\top - \frac{e^{\beta \cdot \boldsymbol{x}} \times \boldsymbol{x}}{(1+e^{\beta \cdot \boldsymbol{x}})^2} \times \boldsymbol{x}^\top \times e^{\beta \cdot \boldsymbol{x}}$
  - $= \frac{e^{\beta \cdot \boldsymbol{x}}\boldsymbol{x}\boldsymbol{x}^\top(1+e^{\beta \cdot \boldsymbol{x}}) - e^{\beta \cdot \boldsymbol{x}}\boldsymbol{x}\boldsymbol{x}^\top e^{\beta \cdot \boldsymbol{x}}}{(1+e^{\beta \cdot \boldsymbol{x}})^2}$
  - $= \frac{e^{\beta \cdot \boldsymbol{x}}\boldsymbol{x}\boldsymbol{x}^T}{(1+e^{\beta \cdot \boldsymbol{x}})^2}$
  - $\mathcal{H} \succcurlyeq 0$

Proof: $\boldsymbol{a}^T\mathcal{H}\boldsymbol{a} = \frac{e^{\beta\cdot x}}{(1+e^{\beta\cdot x})^2}\boldsymbol{a}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{a} = \frac{e^{\beta\cdot x}}{(1+e^{\beta\cdot x})^2}||\boldsymbol{a}^T\boldsymbol{x}||^2 \geq 0$

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 $= \sum_{i=1}^n \ln\left(1 + e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}\right) - y^{(i)}c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}$
  - ∇_c log loss $= \sum_{i=1}^n \frac{1}{1+e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}} \times e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}} \times \tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)} - y^{(i)}\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}$
  - $= \sum_{i=1}^n \tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}\left(\frac{e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}}{1+e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}} - y^{(i)}\right)$
  - Given perfect separation:
    * If $y^{(i)} = 1$, $\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)} > 0$, $\frac{e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}}{1+e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}} - y^{(i)} < 0$
    * If $y^{(i)} = 0$, $\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)} < 0$, $\frac{e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}}{1+e^{c\tilde{\boldsymbol{\beta}}\cdot\boldsymbol{x}^{(i)}}} - y^{(i)} > 0$
  - Thus, for all i, ∇_c log loss < 0
  - Thus gradient descent will cause c to grow without bound

22 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):
  $P(y = k|\boldsymbol{x}) = \frac{e^{f_k(\boldsymbol{x})/T}}{\sum_{j=1}^k e^{f_j(\boldsymbol{x})/T}} = \frac{e^{\boldsymbol{\beta}_k\cdot\boldsymbol{x}/T}}{\sum_{j=1}^k e^{\boldsymbol{\beta}_j\cdot\boldsymbol{x}/T}}$ 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 → 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

23 SVM Classifier

Description

Task — Classification

Description —

- Supervised
- Parametric

24 Hard-Margin SVM Classifier

Formulation

- Assume $y \in \{-1, 1\}$
- Discriminant:
  - $f = sgn(\boldsymbol{\beta}\cdot\boldsymbol{x} + b)$
  - If sign is positive, f outputs 1, else -1
  - Separating hyperplane given by $z = \boldsymbol{\beta}\cdot\boldsymbol{x} + b = 0$
  - Is a linear discriminant
  - $z \perp \boldsymbol{\beta}$
- For some point $\tilde{x}$ closest to the origin:
  - The perpendicular distance to the origin is given by: $\boldsymbol{\beta}\cdot\tilde{\boldsymbol{x}} + b = 0$ since $\tilde{x}$ lies on the separating hyperplane z
  - Then, $||\boldsymbol{\beta}||\,||\tilde{\boldsymbol{x}}||cos(\varphi) + b = ||\boldsymbol{\beta}||\,||\tilde{\boldsymbol{x}}||(-1) + b = 0$ because $\varphi = 180$ degrees
  - Then, $||\tilde{\boldsymbol{x}}|| = \frac{b}{||\boldsymbol{\beta}||}$
- For some point $\boldsymbol{x}^{(i)}$ above z:
  - Projection of instance onto direction of $\boldsymbol{\beta}$: $\boldsymbol{x}^{(i)\prime} = \frac{\boldsymbol{x}^{(i)}\cdot\boldsymbol{\beta}}{||\boldsymbol{\beta}||^2}\boldsymbol{\beta}$
  - Distance of projection to the origin is given by $||\boldsymbol{x}^{(i)\prime}|| = cos(\varphi^{(i)})||\boldsymbol{x}^{(i)}|| = \frac{cos(\varphi^{(i)})||\boldsymbol{x}^{(i)}||\,||\boldsymbol{\beta}||}{||\boldsymbol{\beta}||} = \frac{\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)}}{||\boldsymbol{\beta}||}$
  - Margin $\gamma^{(i)}$ of instance given by: $\gamma^{(i)} = ||\boldsymbol{x}^{(i)\prime}|| + ||\tilde{\boldsymbol{x}}|| = \frac{\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)}+b}{||\boldsymbol{\beta}||}$
- For some point $\boldsymbol{x}^{(i)}$ below z:
  - Margin $\gamma^{(i)}$ of instance given by: $\gamma^{(i)} = -\frac{\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)}+b}{||\boldsymbol{\beta}||}$
- For well-classified points, $\gamma^{(i)} > 0$, for mis-classified points, $\gamma^{(i)} < 0$
- Given that $y \in \{-1, 1\}$ and thus $y^{(i)}(\boldsymbol{\beta}\cdot\boldsymbol{x} + b) > 0$: $\gamma^{(i)} = \frac{y^{(i)}(\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)}+b)}{||\boldsymbol{\beta}||}$
- Margin of system defined by smallest margin for instance: $\gamma = min_i\gamma^{(i)} = \frac{1}{||\boldsymbol{\beta}||}min_i y^{(i)}(\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)} + b)$

- Margin is invariate to scaling of β and b
- Thus, we can write:
  - $min_i \gamma^{(i)} = min_i y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) = 1$
  - Then, $y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \geq 1$ (resp. $\geq m$ without scaling) for all $\boldsymbol{x}^{(i)}$
  - Moreover, since margin for system is defined by smallest margin for instance, $\gamma = \frac{1}{\|\boldsymbol{\beta}\|}$
  - Since margin is defined in both directions (below and above the separating hyperplane), $\gamma = \frac{2}{\|\boldsymbol{\beta}\|}$

Optimization

Parameters — Find parameters $\boldsymbol{\beta}$ and b

Objective function —

- Objective function: $\gamma = \frac{2}{\|\boldsymbol{\beta}\|}$ (resp. $2m$) subject to $y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \geq 1$ (resp. $\geq m$)
- Equivalent cost function: $\gamma = \frac{1}{2}\|\boldsymbol{\beta}\|^2$ subject to $1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \leq 0$ item Cost function in Lagrangian formulation: $\mathcal{L} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \sum_{i=1}^n \alpha^{(i)}(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b))$

Optimization —

- General solution:
  - $\nabla_{\boldsymbol{\beta}}\mathcal{L} = \boldsymbol{\beta} - \sum_{i=1}^n \alpha^{(i)} y^{(i)} \boldsymbol{x}^{(i)} = 0$
  - $\Rightarrow \boldsymbol{\beta}^* = \sum_{i=1}^n \alpha^{(i)} y^{(i)} \boldsymbol{x}^{(i)}$
  - $\nabla_b \mathcal{L} = -\sum_{i=1}^n \alpha^{(i)} y^{(i)} = 0$
  - Subject to:
    - $1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \leq 0$
    - $\alpha^{(i)} \geq 0$
    - $\alpha^{(i)}(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) = 0$
- By Slater's condition (linear separability), strong duality holds
- Objective function in dual Lagrangian, after plugging in found $\boldsymbol{\beta}$: $\mathcal{D} = \frac{1}{2}\|\boldsymbol{\beta}\|^2$ (provided barrier function) $= \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} + \sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)} - \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} = \sum_{i=1}^n \alpha^{(i)} - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$
- Dual optimization: Maximize α subject to
  - $\alpha^{(i)} \geq 0$
  - $\sum_{i=1}^n \alpha^{(i)} y^{(i)} = 0$ due to $\nabla_b \mathcal{L}$
- Note that only *support vectors* ($\alpha^{(i)} > 0$, sit on the hyperplane $1 = y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) = 1$) matter in establishing $\boldsymbol{\beta}^*$ and b^*:
  - Based on complementary slackness condition $\alpha^{(i)}(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) = 0$: We either have
    - $\alpha^{(i)} = 0$ and $(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) > 0$ resp. $y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) > 1$ or
    - $\alpha^{(i)} > 0$ and $(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) = 0$ resp. $y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) = 1$
  - Intercept is calculated as average between optimal intercept for support vector on positive and negative side of margin $b = \frac{1}{2}(\boldsymbol{\beta} \cdot \boldsymbol{x}^+ + \boldsymbol{\beta} \cdot \boldsymbol{x}^-)$ since:
    - $\boldsymbol{\beta} \cdot \boldsymbol{x}^+ + b = 1$
    - $\boldsymbol{\beta} \cdot \boldsymbol{x}^- + b = -1$

Characteristics —

- Strictly convex with psd Hessian
- Has global minimum
- Has unique solution

25 Soft-Margin SVM Classifier

Optimization

Objective function —

- Cost function: Hinge loss: $max(0, 1 - \gamma^{(i)})$
- Mis-classified instances incur a loss
- Well-classified instances incur a loss, if their margin $\gamma^{(i)} < 1$
- Always is equal to or dominates the plain misclassification error
- To translate hinge loss into inequality constraint, we introduce slack variables $\xi^{(i)}$:
  - $y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \geq 1 - \xi^{(i)}$
  - By setting $\xi^{(i)} = 0$, we get pulled down towards hinge loss
  - For
    - Well-classified points outside of margin $\xi^{(i)} < 0$
    - Well-classified points within of margin $0 < \xi^{(i)} < 1$
    - Points on decision boundary $\xi^{(i)} = 1$
    - Mis-classified points $\xi^{(i)} > 1$
- We can then write cost function as slack variables penalized by ℓ_1 norm: $\frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^n \xi^{(i)}$ where
  - $\|\boldsymbol{\beta}\|^2$ maximizes margin and $C\sum_{i=1}^n \xi^{(i)}$ minimizes hinge loss
  - C is a hyperparameter that determines how tolerant we are of margin errors: If C is large, we are less tolerant, margin will decrease, and the soft-margin will become a hard-margin

  subject to:
  - $1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \leq \xi^{(i)}$ resp. $1 - \xi^{(i)} - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) \leq 0$
  - $\xi^{(i)} \geq 0$ resp. $-\xi^{(i)} \leq 0$
- Cost function in Lagrangian formulation: $\mathcal{L} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^n \xi^{(i)} + \sum_{i=1}^n \alpha^{(i)}(1 - \xi^{(i)} - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) - \sum_{i=1}^n \zeta^{(i)} \xi^{(i)}$

Optimization —

- General solution:
  - $\nabla_{\boldsymbol{\beta}}\mathcal{L} = \boldsymbol{\beta} - \sum_{i=1}^n \alpha^{(i)} y^{(i)} \boldsymbol{x}^{(i)} = 0$
  - $\Rightarrow \boldsymbol{\beta}^* = \sum_{i=1}^n \alpha^{(i)} y^{(i)} \boldsymbol{x}^{(i)}$
  - $\nabla_b \mathcal{L} = -\sum_{i=1}^n \alpha^{(i)} y^{(i)} = 0$
  - $\nabla_{\xi^{(i)}} \mathcal{L} = C - \alpha^{(i)} - \zeta^{(i)} = 0$
  - Subject to:
    - $-\xi^{(i)} - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) + 1 \leq 0$
    - $\xi^{(i)} \leq 0$
    - $\alpha^{(i)}, \zeta^{(i)} \geq 0$
    - $\alpha^{(i)}(-\xi^{(i)} - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) + 1) = 0$
    - $\zeta^{(i)}(-\xi^{(i)}) = 0$
- By Slater's condition (linear separability), strong duality holds
- Objective function in dual Lagrangian, after plugging in found $\boldsymbol{\beta}$: $\mathcal{D} = \frac{1}{2}\|\boldsymbol{\beta}\|^2$ (provided barrier function) $= \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} + \sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)} - \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} = \sum_{i=1}^n \alpha^{(i)} - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha^{(i)}\alpha^{(j)} y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$
- Dual optimization: Maximize α subject to
  - $\sum_{i=1}^n \alpha^{(i)} y^{(i)} = 0$ due to $\nabla_b \mathcal{L}$
  - $0 \leq \alpha^{(i)} \leq C$ due to $\nabla_{\xi^{(i)}} \mathcal{L}$
- Note that only *support vectors* ($\alpha^{(i)} > 0$, sit in or on the hyperplane $1 \geq y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) = 1$) matter in establishing $\boldsymbol{\beta}^*$ and b^*:
  - Based on complementary slackness condition $\alpha^{(i)}(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) = 0$: We either have
    - $\alpha^{(i)} = 0$ and $(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) > 0$ resp.$y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) > 1$ or
    - $\alpha^{(i)} > 0$ and $(1 - y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)) = 0$ resp.$y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b) = 1$
  - Similarly, we either have
    - $\zeta^{(i)} = 0$ and $-\xi^{(i)} < 0$ resp.$\xi^{(i)} > 0$ or
    - $\zeta^{(i)} > 0$ and $-\xi^{(i)} = 0$ resp.$\xi^{(i)} = 0$
  - Then, each instance lies in one of three areas:
    - Beyond γ:
      - $\alpha^{(i)} = 0$
      - $C = \zeta^{(i)}$ due to $\nabla_{\xi^{(i)}} \mathcal{L}$
      - $\xi^{(i)} = 0$
      - $1 < y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)$
    - On γ:
      - $\alpha^{(i)}, \zeta^{(i)} > 0$
      - $0 < \alpha^{(i)} < C$ due to $\nabla_{\xi^{(i)}} \mathcal{L}$
      - $\xi^{(i)} = 0$
      - $1 = y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)$
    - Within γ:
      - $\alpha^{(i)} > 0$
      - $\zeta^{(i)} = 0$
      - $\alpha^{(i)} = C$ due to $\nabla_{\xi^{(i)}} \mathcal{L}$
      - $\xi^{(i)} > 0$
      - $1 > y^{(i)}(\boldsymbol{\beta} \cdot \boldsymbol{x}^{(i)} + b)$

Characteristics —

- Strictly convex with psd Hessian
- Has global minimum
- Has unique solution

26 Extensions to the SVM

Multiclass SVMs

Description —

- Instead of binary classification of $y \in \{-1, 1\}$, we have multiclass classification, where each y is assigned to one of k classes $y \in \{1, ..., k\}$

Formulation —

- Let $z^{(i)}$ define the class associated with $\boldsymbol{x}^{(i)}$
- We define a weight vector for each class
- Then, we have: $(\boldsymbol{\beta}_{z^{(i)}} \cdot \boldsymbol{x}^{(i)} + b_{z^{(i)}}) - \max_{z \neq z^{(i)}}(\boldsymbol{\beta}_z \cdot \boldsymbol{x}^{(i)} + b_z) \geq 1$ (resp. $\geq m$ without scaling) for all $\boldsymbol{x}^{(i)}$

Optimization — Hard margin:

- Cost function: $\frac{1}{2}\|\boldsymbol{B}\|^2 = \frac{1}{2}\sum_{z=1}^k \|\boldsymbol{\beta}_z\|^2$ subject to $(\boldsymbol{\beta}_{z^{(i)}} \cdot \boldsymbol{x}^{(i)} + b_{z^{(i)}}) - \max_{z \neq z^{(i)}}(\boldsymbol{\beta}_z \cdot \boldsymbol{x}^{(i)} + b_z) \geq 1$ (resp. $\geq m$)

Soft margin:

- Cost function: $\frac{1}{2}\|\boldsymbol{B}\|^2 + C\sum_{i=1}^n \xi^{(i)} = \frac{1}{2}\sum_{z=1}^k \|\boldsymbol{\beta}_z\|^2 + C\sum_{i=1}^n \xi^{(i)}$ subject to:
  - $1 - \xi^{(i)} - (\boldsymbol{\beta}_{z^{(i)}} \cdot \boldsymbol{x}^{(i)} + b_{z^{(i)}}) + \max_{z \neq z^{(i)}}(\boldsymbol{\beta}_z \cdot \boldsymbol{x}^{(i)} + b_z) \leq 0$
  - $-\xi^{(i)} \leq 0$

Structured SVMs

Description —

- Instead of binary classification of $y \in \{-1, 1\}$, we have structured prediction, where each y is assigned to one structured output (e.g. tree, partition, etc.)

Formulation —

- Challenge: Could be formulated as multi-class SVMs, but that would blow up the number of parameters, since we have one weight for each class

- Solution:
  - Formulate a feature function $\Psi(\boldsymbol{y},\boldsymbol{x})$
  - Define a scoring function $f(\boldsymbol{y},\boldsymbol{x}) = \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y},\boldsymbol{x})$, where the number of parameters depends on the dimensionality of the feature function, but is independent of the number of classes
  - Perform classification via $\hat{y} = \arg\max_y f(\boldsymbol{y},\boldsymbol{x})$
- Then, we have: $f(\boldsymbol{y},\boldsymbol{x}) - \max_{y'} f(\boldsymbol{y}',\boldsymbol{x}) = \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y},\boldsymbol{x}) - \max_{y'}\boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}',\boldsymbol{x}) \geq 1$ (resp. $\geq m$ without scaling) for all $\boldsymbol{x}^{(i)}$

Optimization — Hard margin:

- Cost function: $\frac{1}{2}\|\boldsymbol{B}\|^2 = \frac{1}{2}\sum_{z=1}^{k}\|\boldsymbol{\beta}_z\|^2$ subject to $\boldsymbol{\beta}\cdot\Psi(\boldsymbol{y},\boldsymbol{x}) - \max_{y'}\boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}',\boldsymbol{x}) \geq 1$ (resp. $\geq m$)

Soft margin:

- Cost function: $\frac{1}{2}\|\boldsymbol{B}\|^2 + C\sum_{i=1}^{n}\xi^{(i)} = \frac{1}{2}\sum_{z=1}^{k}\|\boldsymbol{\beta}_z\|^2 + C\sum_{i=1}^{n}\xi^{(i)}$ subject to:
  - $-\xi^{(i)} - \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y},\boldsymbol{x}) + \max_{y'}[\Delta(\boldsymbol{y},\boldsymbol{y}') + \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}',\boldsymbol{x})] \leq 0$ resp. $\Delta(\boldsymbol{y},\boldsymbol{y}') - \xi^{(i)} - \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y},\boldsymbol{x}) + \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}',\boldsymbol{x}) \leq 0$
  - Here, Δ is a loss function, quantifying the loss of predicting y' when the correct output is y
  - This means, we're performing *margin rescaling* to account for varying levels of misclassification severity, based on how far off the prediction is from the correct output
  - $-\xi^{(i)} \leq 0$
- Cost function in Lagrangian formulation: $\mathcal{L} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi^{(i)} + \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}(\Delta(\boldsymbol{y}^{(i)},\boldsymbol{y}^{(j)}) - \xi^{(i)} - \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}^{(i)},\boldsymbol{x}^{(i)}) + \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}^{(j)},\boldsymbol{x}^{(i)})) - \sum_{i=1}^{n}\zeta^{(i)}\xi^{(i)}$
- We can abbreviate $\Psi_i(\boldsymbol{y}^{(j)}) = (-\Psi(\boldsymbol{y}^{(i)},\boldsymbol{x}^{(i)}) + \Psi(\boldsymbol{y}^{(j)},\boldsymbol{x}^{(i)}))$ and $\Delta_i(\boldsymbol{y}^{(j)}) = \Delta(\boldsymbol{y}^{(i)},\boldsymbol{y}^{(j)})$
- General solution:
  - $\nabla_{\boldsymbol{\beta}}\mathcal{L} = \boldsymbol{\beta} - \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Psi_i(\boldsymbol{y}^{(j)}) = 0$
  - $\Rightarrow \boldsymbol{\beta}^* = \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Psi_i(\boldsymbol{y}^{(j)})$
  - $\nabla_b\mathcal{L} = -\sum_{i=1}^{n}\alpha^{(i)}y^{(i)} = 0$
  - $\nabla_{\xi^{(i)}}\mathcal{L} = \sum_{i=1}^{n}C - \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)} - \sum_{i=1}^{n}\zeta^{(i)} = 0$
- By Slater's condition (linear separability), strong duality holds
- Objective function in dual Lagrangian, after plugging in found $\boldsymbol{\beta}$: $\mathcal{D} = -\frac{1}{2}\|\sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Psi_i(\boldsymbol{y}^{(j)})\|^2 + \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Delta_i(\boldsymbol{y}^{(j)})$

  Proof:
  - $\mathcal{D} = \frac{1}{2}\|\boldsymbol{\beta}^*\|^2 + C\sum_{i=1}^{n}\xi^{(i)} + \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Delta(\boldsymbol{y}^{(i)},\boldsymbol{y}^{(j)}) - \alpha^{(ij)}\xi^{(i)} - \alpha^{(ij)}\boldsymbol{\beta}\cdot\Psi_i(\boldsymbol{y}^{(j)}) - \sum_{i=1}^{n}\zeta^{(i)}\xi^{(i)}$
  - $= \frac{1}{2}\|\boldsymbol{\beta}^*\|^2 + \sum_{i=1}^{n}\xi^{(i)}(C - \sum_{y^{(j)}}\alpha^{(ij)} - \zeta^{(i)}) + \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Delta(\boldsymbol{y}^{(i)},\boldsymbol{y}^{(j)}) - \alpha^{(ij)}\boldsymbol{\beta}\cdot\Psi_i(\boldsymbol{y}^{(j)})$
  - $C - \sum_{y^{(j)}}\alpha^{(ij)} - \zeta^{(i)}) = 0$ due to $\nabla_{\xi^{(i)}}\mathcal{L}$
  - After plugging in found $\boldsymbol{\beta}$ and contracting the two first terms, we have $= -\frac{1}{2}\|\boldsymbol{\beta}^*\|^2 + \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Delta(\boldsymbol{y}^{(i)},\boldsymbol{y}^{(j)}) = -\frac{1}{2}\|\sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Psi_i(\boldsymbol{y}^{(j)})\|^2 + \sum_{i=1}^{n}\sum_{y^{(j)}}\alpha^{(ij)}\Delta_i(\boldsymbol{y}^{(j)})$
- Dual optimization: Maximize α subject to
  - $0 \leq \sum_{y^{(j)}}\alpha^{(ij)} \leq C$ due to $\nabla_{\xi^{(i)}}\mathcal{L}$

SVMs in Practice — For a structured SVM, we need to define 4 functions:

- Feature function $\Psi(\boldsymbol{y},\boldsymbol{x})$
- Loss function $\Delta(\boldsymbol{y}',\boldsymbol{y})$
- Prediction rule $\arg\max_y \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y},\boldsymbol{x})$
- Loss-augmented inference $\arg\max_{y'}\left(\Delta(\boldsymbol{y}',\boldsymbol{y}^{(i)}) + \boldsymbol{\beta}\cdot\Psi(\boldsymbol{y}',\boldsymbol{x}^{(i)})\right)$

For a structured SVM, we need to overcome 4 problems:

- Number of parameters needs to be sub-linear with respect to the number of classes
- Enumerating all possible classes may be infeasible, hence, making a single prediction might require a problem-specific algorithm
- A problem-specific loss function must be defined that provides a rank-ordering of solutions with regard to their correctness
- Efficient training algorithms with a run-time complexity sub-linear in the number of classes are needed

SVM Regressor

Description —

- Instead of classification, we do regression

Formulation —

- Let $y^{(i)}$ define the output associated with $\boldsymbol{x}^{(i)}$
- Let ϵ be the width of the region around the regression line in which points can lie at no extra cost
- Let $\hat{\xi}^{(i)}$ and $\xi^{(i)}$ be slack variables
- Then, we have: $(\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)} + b) - y^{(i)} \leq \epsilon + \hat{\xi}^{(i)}$ and $y^{(i)} - (\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)} + b) \leq \epsilon + \xi^{(i)}$ for all $\boldsymbol{x}^{(i)}$

Optimization —

- Cost function: $\frac{1}{2}\|\boldsymbol{B}\|^2 + C\sum_{i=1}^{n}(\xi^{(i)} + \hat{\xi}^{(i)})$ subject to:
  - $(\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)} + b) - y^{(i)} - \epsilon - \hat{\xi}^{(i)} \leq 0$
  - $y^{(i)} - (\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)} + b) - \epsilon - \xi^{(i)} \leq 0$
  - $-\hat{\xi}^{(i)} \leq 0$
  - $-\xi^{(i)} \leq 0$
- Cost function in Lagrangian formulation: $\mathcal{L} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\xi^{(i)} + \hat{\xi}^{(i)}) - \sum_{i=1}^{n}\hat{\mu}^{(i)}\hat{\xi}^{(i)} - \sum_{i=1}^{n}\mu^{(i)}\xi^{(i)} + \sum_{i=1}^{n}\hat{\alpha}^{(i)}((\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)}+b) - y^{(i)} - \epsilon - \hat{\xi}^{(i)}) + \sum_{i=1}^{n}\alpha^{(i)}(y^{(i)} - (\boldsymbol{\beta}\cdot\boldsymbol{x}^{(i)}+b) - \epsilon - \xi^{(i)})$
- General solution:
  - $\nabla_{\boldsymbol{\beta}}\mathcal{L} = \boldsymbol{\beta} + \sum_{i=1}^{n}\hat{\alpha}^{(i)}\boldsymbol{x}^{(i)} - \sum_{i=1}^{n}\alpha^{(i)}\boldsymbol{x}^{(i)} = \boldsymbol{\beta} + \sum_{i=1}^{n}(\hat{\alpha}^{(i)} - \alpha^{(i)})\boldsymbol{x}^{(i)} = 0$
  - $\Rightarrow \boldsymbol{\beta}^* = \sum_{i=1}^{n}\sum_{i=1}^{n}(\alpha^{(i)} - \hat{\alpha}^{(i)})\boldsymbol{x}^{(i)}$
  - $\nabla_b\mathcal{L} = \sum_{i=1}^{n}\hat{\alpha}^{(i)} - \sum_{i=1}^{n}\alpha^{(i)} = \sum_{i=1}^{n}(\hat{\alpha}^{(i)} - \alpha^{(i)}) = 0$
  - $\nabla_{\xi^{(i)}}\mathcal{L} = \sum_{i=1}^{n}C - \sum_{i=1}^{n}\mu^{(i)} - \sum_{i=1}^{n}\alpha^{(i)} = 0$, analog for $\hat{\xi}^{(i)}$
- By Slater's condition (linear separability), strong duality holds
- Objective function in dual Lagrangian, after plugging in found $\boldsymbol{\beta}$: $\mathcal{D} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha^{(i)} - \hat{\alpha}^{(i)})(\alpha^{(j)} - \hat{\alpha}^{(j)})\boldsymbol{x}^{(j)\top}\boldsymbol{x}^{(i)} + \sum_{i=1}^{n}(\alpha^{(i)} - \hat{\alpha}^{(i)})y^{(i)} - \epsilon\sum_{i=1}^{n}(\alpha^{(i)} + \hat{\alpha}^{(i)})$

  Proof:
  - 1) $\mathcal{D} = C\sum_{i=1}^{N}(\xi^{(i)} + \hat{\xi}^{(i)})$ 2) $+\frac{1}{2}\sum_{i=1}^{N}\left((\alpha^{(i)} - \hat{\alpha}^{(i)})\boldsymbol{x}^{(i)}\right)^T\left(\sum_{i=1}^{N}(\alpha^{(i)} - \hat{\alpha}^{(i)})\boldsymbol{x}^{(i)}\right)$ 3) $-\sum_{i=1}^{N}\mu^{(i)}\xi^{(i)} - \hat{\mu}^{(i)}\hat{\xi}^{(i)}$ 4) $+\sum_{i=1}^{N}(\alpha^{(i)} - \hat{\alpha}^{(i)})y^{(i)}$ 5) $+\sum_{i=1}^{N}(\hat{\alpha}^{(i)} - \alpha^{(i)})b$ 6) $+\sum_{i=1}^{N}(-\hat{\alpha}^{(i)} - \alpha^{(i)})\epsilon$ 7) $-\sum_{i=1}^{N}\hat{\alpha}^{(i)}\hat{\xi}^{(i)} - \alpha^{(i)}\xi^{(i)}$ 8) $-\sum_{i=1}^{N}\alpha^{(i)}\sum_{j=1}^{N}((\alpha^{(j)} - \hat{\alpha}^{(j)})\boldsymbol{x}^{(j)})^T\boldsymbol{x}^{(i)} + \sum_{i=1}^{N}\hat{\alpha}^{(i)}\sum_{j=1}^{N}((\alpha^{(j)} - \hat{\alpha}^{(j)})\boldsymbol{x}^{(j)})^T\boldsymbol{x}^{(i)}$
  - We can simplify: 1, 3, 7) $\mathcal{D} = \sum_{i=1}^{N}\left[C - \mu^{(i)} - \alpha^{(i)}\right]\xi^{(i)} + \sum_{i=1}^{N}\left[C - \hat{\mu}^{(i)} - \hat{\alpha}^{(i)}\right]\hat{\xi}^{(i)}$ 8) $-\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha^{(i)} - \hat{\alpha}^{(i)})(\alpha^{(j)} - \hat{\alpha}^{(j)})\boldsymbol{x}^{(j)\top}\boldsymbol{x}^{(i)}$ 2) $+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha^{(i)} - \hat{\alpha}^{(i)})(\alpha^{(j)} - \hat{\alpha}^{(j)})\boldsymbol{x}^{(j)\top}\boldsymbol{x}^{(i)}$ 4) $+\sum_{i=1}^{N}(\alpha^{(i)} - \hat{\alpha}^{(i)})y^{(i)}$ 5) $+\sum_{i=1}^{N}(\hat{\alpha}^{(i)} - \alpha^{(i)})b$ 6) $+\sum_{i=1}^{N}(-\hat{\alpha}^{(i)} - \alpha^{(i)})\epsilon$
  - Given the derivatives and that certain terms $((\sum_{i=1}^{n}\alpha^{(i)} - \hat{\alpha}^{(i)}), (C - \mu^{(i)} - \alpha^{(i)}), (C - \hat{\mu}^{(i)} - \hat{\alpha}^{(i)})$ in 1,3,5,7,8) simplify to 0, we end up with the Dual formulation

27 K-Means Clustering

Description

Task — Clustering

Description —

- Unsupervised
- Non-parametric

Formulation

- We specify that we want $j = 1,\ldots,k$ clusters in total
- Clusters defined by centroid $\boldsymbol{\mu}^{[j]} \in \mathbb{R}^m$
- Instances $i = 1,\ldots,n$ given by $\boldsymbol{x}^{(i)}$
- 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 $\boldsymbol{\mu}^{[j]}$, 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: $j^* = \arg\min_j \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2$
- Distortion function given by: $\Theta = \sum_{i=1}^{n}\sum_{j=1}^{k}p^{i[j]}\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2 = \sum_{\boldsymbol{x}^{(i)}\in C_j}\sum_{j=1}^{k}\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2$ with $p^{i[j]} = \begin{cases}1 & \text{if } j = j^* \\ 0 & \text{otherwise}\end{cases}$ and $C_j = \left\{\boldsymbol{x}^{(i)} \mid p^{i[j]} = 1\right\}$
- If we choose Euclidian distance: $\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\| = \sqrt{\sum_{l=1}^{m}\left(x_l^{(i)} - \mu_l^{[j]}\right)^2}$

Optimization — Lloyd's algorithm:

1. Randomly initialize each $\boldsymbol{\mu}^{[j]}$
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 $\boldsymbol{\mu}^{[j]}$
   - $\nabla_{\boldsymbol{\mu}^{[j]}}\Theta = 2\sum_{i=1}^{n}p^{i[j]}(\boldsymbol{\mu}^{[j]} - \boldsymbol{x}^{(i)}) = 0$
   - Then, $\boldsymbol{\mu}^{[j]} = \frac{\sum_{i=1}^{n}p^{i[j]}\boldsymbol{x}^{(i)}}{\sum_{i=1}^{n}p^{i[j]}} = \frac{1}{|C_j|}\sum_{\boldsymbol{x}^{(i)}\in C_j}\boldsymbol{x}^{(i)}$
   - 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:

- $\Theta_t = \sum_{i=1}^{n}\sum_{j=1}^{k}p_t^{i[j]}\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_t^{[j]}\|^2$
- In M-Step: $\Theta_t \leq \sum_{i=1}^{n}\sum_{j=1}^{k}p_t^{i[j]}\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{t-1}^{[j]}\|^2$
- In E-Step: $\sum_{i=1}^{n}\sum_{j=1}^{k}p_t^{i[j]}\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{t-1}^{[j]}\|^2 \leq \sum_{i=1}^{n}\sum_{j=1}^{k}p_{t-1}^{i[j]}\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{t-1}^{[j]}\|^2 = \Theta_{t-1}$
- Thus, $\Theta_t \leq \Theta_{t-1}$

Proof of convergence 2:

- Let Assignment function $c(i) \to j$ be defined as: $c(i) = \arg\min_{j \in \{1,\dots,k\}} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2$
- Distortion can be rewritten as $\Theta = \sum_i \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{c(i)}\|^2$
- 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 $\boldsymbol{\mu}^{[j]}$ to minimize: $\sum_{\boldsymbol{x}^{(i)} \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2$
- $\boldsymbol{\mu}^{[j]} = \frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \boldsymbol{x}^{(i)}$ minimizes this cost:
  - For any vector $\boldsymbol{\mu}^{[j]\prime}$, we can write the cost as: $\frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]\prime}\|^2 = \frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \|(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}) + (\boldsymbol{\mu}^{[j]} - \boldsymbol{\mu}^{[j]\prime})\|^2 = \frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2 + \|\boldsymbol{\mu}^{[j]} - \boldsymbol{\mu}^{[j]\prime}\|^2 + 2(\boldsymbol{\mu}^{[j]} - \boldsymbol{\mu}^{[j]\prime})^\top \frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})$
  - The third term vanishes since $\frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \boldsymbol{x}^{(i)} = \boldsymbol{\mu}^{[j]}$
  - Then, we have: $= \frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2 + \|\boldsymbol{\mu}^{[j]} - \boldsymbol{\mu}^{[j]\prime}\|^2$
  - We can see that: $\frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2 + \|\boldsymbol{\mu}^{[j]} - \boldsymbol{\mu}^{[j]\prime}\|^2 \geq \frac{1}{|C_j|} \sum_{\boldsymbol{x}^{(i)} \in C_j} \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}\|^2$ with equality if $\boldsymbol{\mu}^{[j]\prime} = \boldsymbol{\mu}^{[j]}$

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

28 Non-Parametric Bayesian Clustering

Motivation

- In GMM, $\pi^{[i]} \sim \mathrm{Dir}(\alpha_1, \dots, \alpha_k)$
- Dir is finite, i.e., all k clusters will be realized with probability 1
- Challenge: selecting k in advance
- Potential solution: Select $k \to \infty$, then only when $n \to \infty$ all clusters will be realized with probability 1
- Better solution: select $k = \infty$, this leads to non-parametric Bayesian models
- Question: How do we get infinite probabilities that sum to 1? We need:
  1. A suitable distribution
  2. A way to sample from it

Dirichlet process (DP)

- Distribution over probability distributions of a space Θ
- $\mathrm{DP}(\alpha, H)$ where α is the concentration parameter and H is the base measure on Θ
- Sample $G \sim \mathrm{DP}(\alpha, H)$ is a probability distribution

Stick Breaking Process

- Observation: Sampling $\pi^{[i]} \sim \mathrm{Dir}(\alpha_1, \dots, \alpha_k)$ is equivalent to sampling: $\pi^{[1]} \sim \mathrm{Beta}(\alpha_1, \dots, \alpha_k)$ and $(\pi^{[2]}, \dots, \pi^{[k]}) \sim \mathrm{Dir}(\alpha_2, \dots, \alpha_k)$
- For finite length k, we have:
  - $\pi^{[1]} = \beta_1$ with $\beta_1 \sim \mathrm{Beta}(\alpha_1, \dots, \alpha_k)$
  - $\pi^{[2]} = (1 - \beta_1)\beta_2$ with $\beta_2 \sim \mathrm{Beta}(\alpha_2, \dots, \alpha_k)$
  - $\pi^{[3]} = (1 - \beta_1)(1 - \beta_2)\beta_3$ with $\beta_3 \sim \mathrm{Beta}(\alpha_3, \dots, \alpha_k)$
- If we fix α, we have a revised form:
  - $\pi^{[1]} = \beta_1$ with $\beta_1 \sim \mathrm{Beta}(\alpha)$
  - $\pi^{[2]} = (1 - \beta_1)\beta_2$ with $\beta_2 \sim \mathrm{Beta}(\alpha)$
  - $\pi^{[k]} = (1 - \sum_{j=1}^{k-1} \pi^{[j]})\beta_k$ with $\beta_k \sim \mathrm{Beta}(\alpha)$
- This is the GEM distribution: $\pi \sim \mathrm{GEM}(\alpha)$ with $\pi = \{\pi^{[k]}\}_{k=1}^{\infty}$
- Connection to DP:
  - If $\pi \sim \mathrm{GEM}(\alpha)$ and $\theta_k \sim H$, then $G(\theta) \sim \sum_{k=1}^{\infty} \pi^{[k]} \delta_{\theta_k}(\theta)$ is a sample from $\mathrm{DP}(\alpha, H)$ and is a distribution over Θ
  - If we repeatedly sample $\theta^{[1]}, \theta^{[2]}, \dots$ from G, we have $\theta^{[i]} = \theta_{k_i}$ where value sometimes has not been observed before ($k_i \neq k_j$ for all $j < i$), sometimes value has been observed before ($k_i = k_j$ for some $j < i$)
  - $\theta^{[i]}, \theta^{[j]}$ with $k_i = k_j$ belong to the same cluster

Chinese restaurant process (CRP)

Connection to clustering:

- Customers are observations $\theta^{[i]}$
- Tables are clusters θ_k
- When a new customer arrives, he either:
  - Joins an existing table with probability proportional to the number of customers already sitting there
  - Starts a new table with probability proportional to α
- Let $\mathcal{P}$ be the current seating arrangement
- The probability that customer $n+1$ joins table T is: $P(\text{customer } n+1 \text{ joins table } T \,|\, \mathcal{P}) = \begin{cases} \frac{|T|}{\alpha+n} & \text{if table } T \text{ is in } \mathcal{P}, \\ \frac{\alpha}{\alpha+n} & \text{otherwise} \end{cases}$
- The joint probability for the seating arrangement $\mathcal{P}$ is: $P(\mathcal{P}) = P(\text{customer 1 joins table T}) \times P(\text{what customer 2 does (join table or start new table)}), \dots = \frac{\alpha^{|\mathcal{P}|}}{\alpha^{(n)}} \prod_{T \in \mathcal{P}} (|T| - 1)!$
- The expected number of tables is: $\mathbb{E}(|\mathcal{P}|) = \mathcal{O}(\alpha \log(N))$

Connection to DP:

- Note: CRP is order and labeling independent
- A sequence x is exchangeable if random vectors $(x_1, x_2, \dots)$ and $(x_{\tau(1)}, x_{\tau(2)}, \dots)$ have the same distribution, where $\tau(\cdot)$ is a random permutation
- According to *De Finetti's Theorem*: For an exchangeable sequence: $P(x_1, x_2, \dots) = \int \prod_i P(x_i|G) dP(G)$ where $G \sim \mathrm{DP}(\alpha, H)$
- We can apply this theorem to CRP

CRP for continuous distributions:

- For continuous distributions, the probability of drawing an x_k that matches exactly one of the previous samples drawn is 0
- Thus: $p(x_k) = \frac{\alpha}{\alpha+k-1}$ and $\sum_{k=1}^{n} p(x_k) = \sum_{k=1}^{n} \frac{\alpha}{\alpha+k-1} = S(n)$
- We can approximate the sum $S(n)$ with the integral $I(n)$: $I(n) = \int_1^{n+1} \frac{\alpha}{\alpha+x-1} dx$
- Integral is bounded above by the sum: $I(n) \leq S(n)$
- Integral is bounded below by: $I(n) \geq \sum_{k=2}^{n+1} \frac{\alpha}{\alpha+k-1}$ since the sum grows smaller as k grows larger. Therefore: $I(n) \geq S(n) - \frac{\alpha}{\alpha} + \frac{\alpha}{\alpha+n} = S'(n)$
- So we have: $S(n) - 1 + \frac{\alpha}{\alpha+n} \leq I(n) \leq S(n)$ which we can manipulate to: $I(n) + 1 - \frac{\alpha}{\alpha+n} \geq S(n) \geq I(n)$
- Computing the integral $I(n)$: $I(n) = \int_1^{n+1} \frac{\alpha}{\alpha+x-1} dx = F(n+1) - F(1) = \alpha(\ln(\alpha+n) - \ln(\alpha))$
- As $n \to \infty$, $I(n) \to \alpha(\ln(n))$
  Thus, as $n \to \infty$, $S(n) \to \alpha(\ln(n))$

DP mixture model

- Let:
  - $\Theta = \mathbb{R}$, space of parameters that defines family of probability distributions
  - $H = \mathcal{N}(\mu_0, \sigma_0)$, base measure on Θ, representing prior belief over parameters
- On that basis, we define the DP mixture model:
  - Probabilities of clusters: $\pi = (\pi^{[1]}, \dots) \sim \mathrm{GEM}(\alpha)$
  - Cluster centers: $\mu_k \sim \mathcal{N}(\mu_0, \sigma_0)$
  - Assignments of datapoints to clusters: $z^{(i)} \sim \mathrm{Categorical}(\pi)$
  - Coordinates of datapoints: $x^{(i)} \sim \mathcal{N}(\mu^{[z^{(i)}]}, \sigma^{[z^{(i)}]})$
- Fitting the model:
  - Prior: Probability of cluster assignment, based on cluster size
  - Likelihood: Probability of datapoint, given cluster center
  - *Gibbs sampling*:
    - ⋆ Idea: Sample each variable in turn, conditioned on values of all other variables in the distribution
    - ⋆ $P(z^{(i)} = k \,|\, \boldsymbol{z}^{(-i)}, \boldsymbol{x}, \alpha, \boldsymbol{\mu}) \propto P(z^{(i)} = k \,|\, \boldsymbol{z}^{(-i)}) P(\boldsymbol{x} \,|\, z^{(i)} = k, \boldsymbol{z}^{(-i)})$ due to Baye's rule
    - ⋆ $\propto P(z^{(i)} = k \,|\, \boldsymbol{z}^{(-i)}) P(x^{(i)} \,|\, z^{(i)} = k, \boldsymbol{z}^{(-i)}, \boldsymbol{x}^{(-i)}) P(\boldsymbol{x}^{(-i)} \,|\, z^{(i)} = k, \boldsymbol{z}^{(-i)})$ due to product rule
    - ⋆ $\propto P(z^{(i)} = k \,|\, \boldsymbol{z}^{(-i)}) P(x^{(i)} \,|\, z^{(i)} = k, \boldsymbol{z}^{(-i)}, \boldsymbol{x}^{(-i)})$ because in last term $\boldsymbol{x}^{(-i)} \perp z^{(i)} \,|\, \boldsymbol{z}^{(-i)}$ by d-separation, so this term is constant with respect to k
    - ⋆ We then have: $P(z^{(i)} = k \,|\, \boldsymbol{z}^{(-i)}, \alpha) P(x^{(i)} \,|\, z^{(i)} = k, \boldsymbol{z}^{(-i)}, \boldsymbol{x}^{(-i)}, \boldsymbol{\mu}) =$ prior × likelihood, with interchangeable $z^{(1)}, \dots, z^{(k)}$
    - ⋆ Prior: Comes from CRP, given by: $P(z^{(i)}) = k \,|\, \boldsymbol{z}^{(-i)}, \alpha) = \begin{cases} \frac{N_{k,-i}}{\alpha+N-1} & \text{if } k \text{ is an existing cluster} \\ \frac{\alpha}{\alpha+N-1} & \text{otherwise} \end{cases}$
    - ⋆ Likelihood: We only have to consider points in $\boldsymbol{x}$ that are assigned to cluster k, given by: $P(z^{(i)} \,|\, z^{(i)} = k, \boldsymbol{z}^{(-i)}, \boldsymbol{x}^{(-i)}, \boldsymbol{\mu}) = \begin{cases} P(x^{(i)} \,|\, \boldsymbol{x}^{(-i,k)}, \boldsymbol{\mu}) = \frac{P(x^{(i)}, \boldsymbol{x}^{(-i,k)} | \boldsymbol{\mu})}{P(\boldsymbol{x}^{(-i,k)} | \boldsymbol{\mu})} & \text{for existing } k \\ P(x^{(i)} \,|\, \boldsymbol{\mu}) & \text{otherwise} \end{cases}$
    - ⋆ Thus, Gibbs sampler: $P(z^{(i)} = k \,|\, \boldsymbol{z}^{(-i)}, \boldsymbol{x}, \alpha, \boldsymbol{\mu}) = \begin{cases} \frac{N_{k,-i}}{\alpha+N-1} \times P(x^{(i)} \,|\, \boldsymbol{x}^{(-i,k)}, \boldsymbol{\mu}) & \text{for existing } k \\ \frac{\alpha}{\alpha+N-1} \times P(x^{(i)} \,|\, \boldsymbol{\mu}) & \text{otherwise} \end{cases}$

Latent dirichlet allocation (LDA)

- Extension of DP Mixture Model where each $x^{(i)}$ can be assigned multiple clusters (multivariate), but we don't know in advance how many (non-parametric, as in DP Mixture Model)
- Distribution of topics in document d: $\boldsymbol{\theta}_d \sim \mathrm{Dir}(\alpha)$
- What topic word w belongs to in document d: $z^{(d,w)} \sim \mathrm{Categorical}(\boldsymbol{\theta}_d)$
- Distribution of words in topic k: $\varphi_k \sim \mathrm{Dir}(\beta)$
- What is word w in document d: $w^{(d,w)} \sim \mathrm{Categorical}(\varphi_{z^{(d,w)}})$
- α controls prior weights of topics in documents
- β controls prior weights of words in topics

29 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 $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n} \in \mathbb{R}^m$ onto space $\mathbb{R}^d$ spanned by orthonormal basis $\{\boldsymbol{u}^{[j]}\}_{j=1}^{d} \in \mathbb{R}^m$ where $d << m$
- Each instance $\boldsymbol{x}^{(i)}$ is projected onto each basis vectors $\boldsymbol{u}^{[j]} \cdot \boldsymbol{x}^{(i)}$: $\boldsymbol{x}^{(i)} \rightarrow [\boldsymbol{u}^{[1]} \cdot \boldsymbol{x}^{(i)}, ..., \boldsymbol{u}^{[d]} \cdot \boldsymbol{x}^{(i)}]^\top$
- Each basis vector $\boldsymbol{u}^{[j]}$ 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: $\boldsymbol{u}^{[j]} \cdot \overline{\boldsymbol{x}} = \boldsymbol{u}^{[j]} \cdot \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)}$
- Variance of projected data for a given basis vector: $\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{u}^{[j]} \cdot \boldsymbol{x}^{(i)} - \boldsymbol{u}^{[j]} \cdot \overline{\boldsymbol{x}})^2 = \boldsymbol{u}^{[j]\top} S \boldsymbol{u}^{[j]}$ where $S = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})^\top = \frac{1}{n}X^\top X$ is the covariance matrix of the data in the orthogonal complement to the subspace spanned by the first $j-1$ principal components, i.e. $X_{j-1} = \{\boldsymbol{x}^{(i)} - \text{projection}_{u_{\leq j-1}}\} = \{\boldsymbol{x}^{(i)} - \sum_{l=1}^{j-1}(\boldsymbol{u}^{[j]} \cdot \boldsymbol{x}^{(i)}) \cdot \boldsymbol{u}^{[j]}\}$

Optimization

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

Objective function —

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

Optimization — For $\boldsymbol{u}^{[1]}$, based on $X = X_0$:

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

For $\boldsymbol{u}^{[2]}$, based on $X = X_1$:

- $\nabla_{\boldsymbol{u}^{[2]}}\mathcal{L} = 2S\boldsymbol{u}^{[2]} - 2\lambda^{[2]}\boldsymbol{u}^{[2]} - \lambda^{[1][2]}\boldsymbol{u}^{[1]} = 0$
- $\Rightarrow S\boldsymbol{u}^{[2]} = \lambda^{[2]}\boldsymbol{u}^{[2]}$
  Proof:
  - Multiplying with $\boldsymbol{u}^{[1]\top}$: $2\boldsymbol{u}^{[1]\top} S \boldsymbol{u}^{[2]} - 2\lambda^{[2]}\boldsymbol{u}^{[1]\top}\boldsymbol{u}^{[2]} - \lambda^{[1][2]}\boldsymbol{u}^{[1]\top}\boldsymbol{u}^{[1]} = 0$
  - $= 2\boldsymbol{u}^{[1]\top} S \boldsymbol{u}^{[2]} - 0 - \lambda^{[1][2]} \times 1 = 0$ because of orthogonality resp. orthonormality
  - $= 2\boldsymbol{u}^{[2]\top} S \boldsymbol{u}^{[1]} - \lambda^{[1][2]} = 0$ because the variance is a scalar and can be transposed and because the covariance matrix is symmetric
  - $= 2\boldsymbol{u}^{[2]\top} \lambda^{[1]} \boldsymbol{u}^{[1]} - \lambda^{[1][2]} = 0$ 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 $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n} \in \mathbb{R}^m$ onto space $\mathbb{R}^d$ spanned by orthonormal basis $\{\boldsymbol{u}^{[j]}\}_{j=1}^{d} \in \mathbb{R}^m$ (subset of $\{\boldsymbol{u}^{[j]}\}_{j=1}^{m} \in \mathbb{R}^m$) where $d \ll m$
- Given basis, we can represent original datapoint as: $\boldsymbol{x}^{(i)} = \sum_{j=1}^{m}(\boldsymbol{x}^{(i)} \cdot \boldsymbol{u}^{[j]})\boldsymbol{u}^{[j]}$
- We can represent projected datapoint as: $\tilde{\boldsymbol{x}}^{(i)} = \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{x}^{(i)} = \boldsymbol{B}^T\boldsymbol{x}^{(i)}$ since $\boldsymbol{B}$ is orthogonal or, equivalently $\tilde{\boldsymbol{x}}^{(i)} = \sum_{j=1}^{d} \alpha_{ij}\boldsymbol{u}^{[j]} + \sum_{j=d+1}^{m} \gamma_j \boldsymbol{u}^{[j]}$ where α_{ij} is specific to the instance and γ_j is generic and maps up to the subspace

Optimization

Objective function —

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

30 Gaussian Mixture Models (GMM)

Description

Task — Clustering

Description —

- Unsupervised
- Non-parametric

Formulation

- Instances $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$
- Each instance has a latent cluster assignment given by: $z^{(i)} \in \{1, ..., k\}$
- Probability that cluster assigned to instance i is cluster j is given by: $\pi^{[j]} = p(z^{(i)} = j)$
- $z^{(i)} \sim \text{Categorical}(\pi^{[1]}, ..., \pi^{[k]})$
- $\pi^{[j]} \sim \text{Dir}(\alpha_1, ..., \alpha_k)$
- Contingent on cluster assignment, each instance is the outcome of a random variable associated with a given cluster: $\boldsymbol{x}^{(i)}|z^{(i)} = j \sim \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}^{[j]}, \Sigma^{[j]})$ where $\mu^{[j]}$ is the mean and $\Sigma^{[j]}$ is the covariance associated with cluster j
- Then, marginal distribution of each instance is given by: $p(\boldsymbol{x}^{(i)}) = \sum_{j=1}^{k} p(\boldsymbol{x}^{(i)}|z^{(i)})p(z^{(i)}) = \sum_{j=1}^{k}\mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}^{[j]}, \Sigma^{[j]}) \times \pi^{[j]}$
- This is the GMM, characterized by parameters $\theta = \{\boldsymbol{\mu}^{[j]}, \Sigma^{[j]}, \pi^{[j]}\}_{j=1}^{k}$

Optimization

Parameters — Find parameters $\theta = \{\boldsymbol{\mu}^{[j]}, \Sigma^{[j]}, \pi^{[j]}\}_{j=1}^{k}$

Objective function — In multi-class case:

- Maximize log likelihood $L = \sum_{i=1}^{n}\log p(\boldsymbol{x}^{(i)}) = \sum_{i=1}^{n}\log(\sum_{j=1}^{k} p(\boldsymbol{x}^{(i)}|z^{(i)})p(z^{(i)})) = \sum_{i=1}^{n}\log(\sum_{j=1}^{k}\mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}^{[j]}, \Sigma^{[j]}) \times \pi^{[j]})$ subject to $\sum_{j=1}^{k}\pi^{[j]} = 1$ and $\Sigma^{[j]} \succ 0$
- Challenge: This is a constrained, not concave, not analytically solvable optimization problem, due to the sum within the log
- Solution: Expectation maximization algorithm, which we now motivate
- Let's temporarily assume we know which cluster each instance is associated with, i.e. we know $z^{(i)}$
- For each instance, let $q_{z^{(i)}}$ be some probability distribution over $z^{(i)}$
- Then, we can further expand log likelihood: $L = \sum_{i=1}^{n}\log\sum_{z^{(i)}} q(z^{(i)})\frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{q(z^{(i)})}$
- According to Jensen's inequality: $L = \sum_{i=1}^{n}\log\sum_{z^{(i)}} q(z^{(i)})\frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{q(z^{(i)})} \geq \sum_{i=1}^{n}\sum_{z^{(i)}} q(z^{(i)})\log\frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{q(z^{(i)})}$
- We have thus derived a lower bound for L

- We can tighten this bound and achieve equality by selecting $q_{z^{(i)}}$ accordingly: $q_{z^{(i)}} = p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})$
  Proof 1:
  $\sum_{i=1}^n \sum_{z^{(i)}} q(z^{(i)}) \log \frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{q(z^{(i)})} =$
  $\sum_{i=1}^n \sum_{z^{(i)}} p_\theta(z^{(i)}|\boldsymbol{x}^{(i)}) \log \frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})} =$
  $\sum_{i=1}^n \sum_{z^{(i)}} p_\theta(z^{(i)}|\boldsymbol{x}^{(i)}) \log \frac{p_\theta(\boldsymbol{x}^{(i)}) p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})}{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})} =$
  $\sum_{i=1}^n \sum_{z^{(i)}} p_\theta(z^{(i)}|\boldsymbol{x}^{(i)}) \log p_\theta(\boldsymbol{x}^{(i)}) = \sum_{i=1}^n \times 1 \times \log p_\theta(\boldsymbol{x}^{(i)}) = L$
  Proof 2:
  $L = \mathbb{E}_q[log(p_\theta(\boldsymbol{x}^{(i)}))] = \mathbb{E}_q[log(\frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})} \frac{q(z^{(i)})}{q(z^{(i)})})] =$
  $\mathbb{E}_q[log(\frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{q(z^{(i)})})] + \mathbb{E}_q[log(\frac{q(z^{(i)})}{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})})] = M + E$
- We can further specify $q_{z^{(i)}} = p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})$ to: $q_{z^{(i)}} = p_\theta(z^{(i)}|\boldsymbol{x}^{(i)}) = \frac{p_\theta(\boldsymbol{x}^{(i)}|z^{(i)}) p(z^{(i)})}{\sum_{j=1}^k p_\theta(\boldsymbol{x}^{(i)}|z^{(i)}) p(z^{(i)})} = \frac{\mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}^{[j]}, \Sigma^{[j]}) \times \pi^{[j]}}{\sum_{j=1}^k \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}^{[j]}, \Sigma^{[j]}) \times \pi^{[j]}} = \gamma^{i[j]}$

In binary case:

- Let's temporarily assume we know which cluster each instance is associated with, i.e. we know $z^{(i)}$
- For each instance, let
  $M_{z^{(i)}} = \begin{cases} 1 & \text{if cluster } z^{(i)} = j \text{ has generated } \boldsymbol{x}^{(i)} \\ 0 & \text{otherwise} \end{cases}$
- Then, we can write joint likelihood as:
  $\prod_{i=1}^n \prod_{z^{(i)}} (p(\boldsymbol{x}^{(i)}|z^{(i)}) p(z^{(i)}))^{M_{z^{(i)}}} = \prod_{i=1}^n \prod_{z^{(i)}} (p(\boldsymbol{x}^{(i)}|z^{(i)}) \pi^{[j]})^{M_{z^{(i)}}}$
- Log likelihood is given by: $\sum_{i=1}^n \sum_{z^{(i)}} M_{z^{(i)}} \log(p(\boldsymbol{x}^{(i)}|z^{(i)}) \pi^{[j]})$
- Expectation over latent variables M, given θ:
  $\mathbb{E}[\sum_{i=1}^n \sum_{z^{(i)}} M_{z^{(i)}} \log(p(\boldsymbol{x}^{(i)}|z^{(i)}) \pi^{[j]})] =$
  $\sum_{i=1}^n \sum_{z^{(i)}} \mathbb{E}[M_{z^{(i)}}|\mathcal{X}] \log(p(\boldsymbol{x}^{(i)}|z^{(i)}) \pi^{[j]})$
- We can develop: $\mathbb{E}[M_{z^{(i)}}|\mathcal{X}] = P(M_{z^{(i)}} = 1|\boldsymbol{x}^{(i)}) \times 1 + P(M_{z^{(i)}} = 0|\boldsymbol{x}^{(i)}) \times 0 = P(M_{z^{(i)}} = 1|\boldsymbol{x}^{(i)}) = P(z^{(i)} = j|\boldsymbol{x}^{(i)}) = \frac{P(\boldsymbol{x}^{(i)}|z^{(i)}=j) P(z^{(i)}=j)}{P(\boldsymbol{x}^{(i)})}$

Optimization — Expectation maximization algorithm

1. Randomly initialize $\theta^{(t)} = \{\boldsymbol{\mu}^{[j](t)}, \Sigma^{[j](t)}, \pi^{[j](t)}\}_{j=1}^k$
2. *E-step*: Minimize E, by computing $q(z^{(i)})$ given $\boldsymbol{x}^{(i)}$ and $\theta^{(t)}$
3. *M-step*: Maximize M, by updating $\theta^{(t)}$ based on MLE for Gaussians, while keeping $q(z^{(i)})$ fixed:
   - $\boldsymbol{\mu}^{[j](t+1)} = \frac{\sum_{i=1}^n q(z^{(i})\boldsymbol{x}^{(i)}}{\sum_{i=1}^n q(z^{(i})}$
   - $\Sigma^{[j](t+1)} = \frac{\sum_{i=1}^n q(z^{(i})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j](t+1)})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j](t+1)})^\intercal}{\sum_{i=1}^n q(z^{(i})}$
   - $\pi^{[j](t+1)} = \frac{1}{n} \sum_{i=1}^n q(z^{(i})$
4. Repeat 2 and 3 until convergence

Further proofs

M Step — Recall: Likelihood is:

$L = \sum_{i=1}^n \log\left(\sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)\right)$

Proof of optimal mean in M-step:

- $\nabla_{\boldsymbol{\mu}^{[j]}} L = \sum_{i=1}^n \frac{\delta L}{\delta \mathcal{N}(...)} \times \frac{\delta \mathcal{N}(...)}{\delta \boldsymbol{\mu}^{[j]}}$
  $= \sum_{i=1}^n \frac{\pi^{[j]}}{\sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)} \times \frac{\delta \mathcal{N}(...)}{\delta \boldsymbol{\mu}^{[j]}}$
  $= \sum_{i=1}^n \frac{\pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)}{\sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)} \times \frac{\delta \log \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)}{\delta \boldsymbol{\mu}^{[j]}}$
  because $\frac{\delta \log \mathcal{N}(...)}{\delta \boldsymbol{\mu}^{[j]}} = \frac{1}{\mathcal{N}(...)} \times \frac{\delta \mathcal{N}(...)}{\delta \boldsymbol{\mu}^{[j]}}$
  $\Rightarrow \frac{\delta \mathcal{N}(...)}{\delta \boldsymbol{\mu}^{[j]}} = \mathcal{N}(\ldots) \times \frac{\delta \log \mathcal{N}(...)}{\delta \boldsymbol{\mu}^{[j]}}$
  $= \sum_{i=1}^n \gamma^{i[j]} \times \frac{\delta \log\left(\text{constant} \times \frac{1}{|\Sigma|^{1|2}} \times \exp\left(-\frac{1}{2}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal \Sigma^{-1} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})\right)\right)}{\delta \boldsymbol{\mu}^{[j]}}$
  $= \sum_{i=1}^n \gamma^{i[j]} \times \frac{\delta}{\delta \boldsymbol{\mu}^{[j]}} \left(\log(1) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal \Sigma^{-1} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})\right)$
  $= \sum_{i=1}^n \gamma^{i[j]} \times \left(-\frac{1}{2} \times \frac{\delta}{\delta \boldsymbol{\mu}^{[j]}} \left[\log(|\Sigma|) + (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal \Sigma^{-1} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})\right]\right)$
  $= \sum_{i=1}^n \gamma^{i[j]} \times \left(-\frac{1}{2} \times 2\Sigma^{-1}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}) \times -1\right)$ due to matrix calculus
  $= \sum_{i=1}^n \gamma^{i[j]} \Sigma^{-1}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]}) = 0$
- $= \sum_{i=1}^n \gamma^{i[j]} \Sigma^{-1} \boldsymbol{x}^{(i)} = \sum_{i=1}^n \gamma^{i[j]} \Sigma^{-1} \boldsymbol{\mu}^{[j]}$
- $\boldsymbol{\mu}^{[j]*} = \frac{\sum_{i=1}^n \gamma^{i[j]} \boldsymbol{x}^{(i)}}{\sum_{i=1}^n \gamma^{i[j]}}$

Proof of optimal variance in M-step:

- $\nabla_{\Sigma^{[j]}} L = ...\text{as above}$
  $= \sum_{i=1}^n \gamma^{i[j]} \times \left(-\frac{1}{2} \times \frac{\delta}{\delta \Sigma^{[j]}} \left[\log(|\Sigma|) + (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal \Sigma^{-1} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})\right]\right)$
  $= \sum_{i=1}^n \gamma^{i[j]} \times \left(-\frac{1}{2} \times (\Sigma^{-1} - \Sigma^{-1}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal \Sigma^{-1})\right) = 0$
  due to matrix calculus
- $\sum_{i=1}^n \gamma^{i[j]} \times \frac{1}{2} \times \Sigma^{-1} = \sum_{i=1}^n \gamma^{i[j]} \times \frac{1}{2} \times \Sigma^{-2}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal$
  $\Rightarrow \sum_{i=1}^n \gamma^{i[j]} \Sigma = \sum_{i=1}^n \gamma^{i[j]} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal$
- $\Sigma^{[j]*} = \frac{\sum_{i=1}^n \gamma^{i[j]} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{[j]})^\intercal}{\sum_{i=1}^n \gamma^{i[j]}}$

Proof of optimal π in M-step:

- Lagrangian optimization: $\arg\min_{\pi^{[j]}} L$ subject to $\sum_{j=1}^k \pi^{[j]} = 1$, i.e. $\sum_{j=1}^k \pi^{[j]} - 1 = 0$
- Lagrangian:
  $\mathcal{L}(\pi^{[j]}) = \sum_{i=1}^n \log\left(\sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)\right) + \lambda\left(\sum_{j=1}^k \pi^{[j]} - 1\right)$
- Derivative: $\nabla_{\pi^{[j]}} \mathcal{L}(\pi^{[j]}) = \sum_{i=1}^n \frac{\sum_{j=1}^k \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)}{\sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)} + \lambda$
  $= \sum_{i=1}^n \sum_{j=1}^k \frac{\gamma^{i[j]}}{\pi^{[j]}} + \lambda = 0$
- $\sum_{i=1}^n \sum_{j=1}^k \gamma^{i[j]} = -\lambda \sum_{j=1}^k \pi^{[j]}$
- Solving for λ:
  $\frac{\sum_{i=1}^n \sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)}{\sum_{i=1}^n \sum_{j=1}^k \pi^{[j]} \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{[j]}, \Sigma^{[j]}\right)} = -\lambda \times 1$ due to constraint
  $n = -\lambda$
- Solving for $\pi^{[j]}$: $\pi^{[j]} = \frac{\sum_{i=1}^n \gamma^{i[j]}}{-\lambda}$
  Plugging in λ: $\pi^{[j]*} = \frac{1}{n} \sum_{i=1}^n \gamma^{i[j]}$

Proof that M-step objective is concave:

- We aim top optimize $\log(p(\boldsymbol{x}^{(i)}, z^{(i)}) = \log(p(\boldsymbol{x}^{(i)}|z^{(i)}) p(z^{(i)}) = \log(\mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}^{[j]}, \Sigma^{[j]})) + \log(\pi^{[j]})$
- This is a sum of concave functions

E Step — E corresponds to the KL divergence between $q(z^{(i)})$ and $p(z^{(i)}|\boldsymbol{x}^{(i)})$

Relationship L, M, E — Proof that $L \geq M \Leftrightarrow E \geq 0$:

- $E = \mathbb{E}_q[log(\frac{q(z^{(i)})}{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})})] = \mathbb{E}_q[-log(\frac{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})}{q(z^{(i)})})]$
- According to Jensen's inequality:
  $E \geq -log(\mathbb{E}_q[\frac{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})}{q(z^{(i)})}]) = -log(\sum_{i=1}^k q(z^{(i)}) \frac{p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})}{q(z^{(i)})}) =$
  $-log(\sum_{i=1}^k p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})) = -log(1) = 0$

$L = M \Leftrightarrow E = 0$, i.e. when $q(z^{(i)}) = p_\theta(z^{(i)}|\boldsymbol{x}^{(i)})$

Proof that EM-algorithm converges —

- According to Jensen's inequality:
  $L_t \geq \sum_{i=1}^n \sum_{z^{(i)}} q_t(z^{(i)}) \log \frac{p_{\theta_t}(\boldsymbol{x}^{(i)}, z^{(i)})}{q_t(z^{(i)})}$
- In previous M-step, L was maximized by setting θ_t:
  $\theta_t = \arg\max_\theta(q_t(z^{(i)}) \log \frac{p_\theta(\boldsymbol{x}^{(i)}, z^{(i)})}{q_t(z^{(i)})})$
- Thus, $\sum_{i=1}^n \sum_{z^{(i)}} q_t(z^{(i)}) \log \frac{p_{\theta_t}(\boldsymbol{x}^{(i)}, z^{(i)})}{q_t(z^{(i)})} \geq$
  $\sum_{i=1}^n \sum_{z^{(i)}} q_t(z^{(i)}) \log \frac{p_{\theta_{t-1}}(\boldsymbol{x}^{(i)}, z^{(i)})}{q_t(z^{(i)})}$
- RHS of this equation is output of previous E-step, where q_t was set such that: $\sum_{i=1}^n \sum_{z^{(i)}} q_t(z^{(i)}) \log \frac{p_{\theta_{t-1}}(\boldsymbol{x}^{(i)}, z^{(i)})}{q_t(z^{(i)})} = L_{t-1}$
- Then $L_t \geq L_{t-1}$

Characteristics —

- Not convex
- May converge to local minimum
- Not analytically solvable
- Always converges, since $L \geq M$ and $M^{(t+1)} \geq M^{(t)}$ due to maximizing over M at each step

31 Neural Networks

Formulation

Formulation —

- Model architecture:
  - Input features $(\boldsymbol{X})$ > weights $(\boldsymbol{B})$ > weighted sums $(\boldsymbol{S})$ > activation functions (φ) > hidden states $(\boldsymbol{H})$ > weights > ... > hidden states > activation function > outputs
  - E.g.
    - * Outputs: Probability $p(y \mid \boldsymbol{x})$ for each class y
    - * Activation: Softmax ensures all P add to 1:
      $p(y \mid \boldsymbol{x}) = \frac{\exp\left(\boldsymbol{h}_y^{(K)}\right)}{\sum_{y'} \exp\left(\boldsymbol{h}_{y'}^{(K)}\right)}$
    - * Hidden layer: $\boldsymbol{h}^{(K)} = \sigma(\boldsymbol{w}^{(K)} \boldsymbol{h}^{(K-1)})$
      ...
      $\boldsymbol{h}^{(1)} = \sigma(\boldsymbol{w}^{(1)} \boldsymbol{e}(\boldsymbol{x}))$
    - * Concatenated vector of word embeddings: $\boldsymbol{e}(\boldsymbol{x}) = \frac{1}{n} \sum_{w_i} \boldsymbol{e}(w_i)$
    - * Transformation: word embedding $\boldsymbol{e}(w_i)$
    - * 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 $\boldsymbol{x}^{(i)[0]}$ resp. instance from previous layer $\boldsymbol{h}^{(i)[k-1]}$ given by:
  $\boldsymbol{h}^{(j)[1]} = \varphi(\boldsymbol{x}^{(i)[0]} \cdot \boldsymbol{\beta}^{(j)[1]})$ resp. $\boldsymbol{h}^{(j)[k]} = \varphi(\boldsymbol{h}^{(i)[k-1]} \cdot \boldsymbol{\beta}^{(j)[k]})$
- Outputs for neurons $1, ..., j$ in fixed layer (notation for layer omitted below) given by:

$\boldsymbol{H} = \varphi(\boldsymbol{XB}) = \varphi(\boldsymbol{S})$ where

- $\boldsymbol{X} \in \mathbb{R}^{n \times m+1}$ (incl. bias term)
- $\boldsymbol{B} \in \mathbb{R}^{m+1 \times j}$ with a weight vector for each neuron in each column (incl. bias term)
- $\boldsymbol{S} \in \mathbb{R}^{n \times j}$ 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 $\theta = \boldsymbol{B}$

Objective function —

- Minimize standard objectives, e.g. MSE

Optimization —

- Perform *forward pass* with randomly initialized parameters, to calculate loss
- Perform *backpropagation*, to calculate gradient:
  - $\frac{\partial L}{\partial \theta} = [\frac{\partial L}{\partial \boldsymbol{B}^{[0]}}, \ldots, \frac{\partial L}{\partial \boldsymbol{B}^{[output]}}]$
  - $\frac{\partial L}{\partial \boldsymbol{B}^{[k]}} = \frac{\partial L}{\partial \boldsymbol{H}^{[l]}} \frac{\partial \boldsymbol{H}^{[l]}}{\partial \boldsymbol{B}^{[k]}} = C$
    - When $l > k+1$, i.e. when going several layers back:
      $\frac{\partial L}{\partial \boldsymbol{B}^{[k]}} = \frac{\partial L}{\partial \boldsymbol{H}^{[l]}} \frac{\partial \boldsymbol{H}^{[l]}}{\partial \boldsymbol{S}^{[l-1]}} \frac{\partial \boldsymbol{S}^{[l-1]}}{\partial \boldsymbol{H}^{[l-1]}} \frac{\partial \boldsymbol{H}^{[l-1]}}{\partial \boldsymbol{B}^{[k]}}$
    - When $l = k+1$, i.e. when going one layer back:
      $\frac{\partial L}{\partial \boldsymbol{B}^{[k]}} = \frac{\partial L}{\partial \boldsymbol{H}^{[k+1]}} \frac{\partial \boldsymbol{H}^{[k+1]}}{\partial \boldsymbol{S}^{[k]}} \frac{\partial \boldsymbol{S}^{[k]}}{\partial \boldsymbol{B}^{[k]}}$
- 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:
    - $h_m = \sigma(\theta h_{m-1} + x_m)$
    - $\frac{\partial h_{m+k}}{\partial h_m} = \prod_{i=0}^{k-1} \frac{\partial h_{m+k-i}}{\partial h_{m+k-i-1}} = \prod_{i=1}^{k} \theta \times \sigma'(\theta h_{m+k-i-1} + x_{m+k-i})$
    - $\leq \prod_{i=1}^{k} \theta \times 0.25 = \theta^k \times 0.25^k$ 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 $b^{[l]} \leq 0$
    - 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

32 Backpropagation

Big Picture

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

Point of Departure

Composite function — Ordered series of equations (*primitives*), where each equation is a function only of the preceding equations

For example: $f(x,z) = x^2 + 3z$ $a = x^2, \quad b = 3z, \quad c = a + b, \quad f(x,z) = c$

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 $j \leq i$
  - 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:
   $z_i = \begin{cases} x_i & \text{if } i \leq n \quad \text{(n input nodes initialized to value)} \\ 0 & \text{if } i > n \quad \text{(non-input nodes initialized to 0)} \end{cases}$ For $i = n+1, \ldots, M$ 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 $\{z_1, z_2, \ldots, z_M\}$

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: $\boldsymbol{z} \leftarrow \text{forward propagate}(f, \boldsymbol{x})$
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 $i = M-1, \ldots, 1$:
   $\frac{\partial f}{\partial z_i} = \sum_{j: i \in \text{Pa}(i)} \frac{\partial f}{\partial z_j} \frac{\partial z_j}{\partial z_i} = \sum_{j: i \in \text{Pa}(i)} \frac{\partial f}{\partial z_j} \frac{g_j\left(\langle z_{\text{pa}(j)} \rangle\right)}{\partial z_i}$
4. Return $\left[\frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2}, \ldots, \frac{\partial f}{\partial z_M}\right]$

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 $k^t h$ order derivatives:

- Backpropagation on a graph with $|E|$ edges, for inputs $\boldsymbol{x} = (x_1, \ldots, x_n)^\intercal$, for the k^{th} order derivative has runtime

$\mathcal{O}(|E|n^{k-1})$
Proof:

- For second-order derivative: $\nabla_2 f(\boldsymbol{x})(\text{i.e. Hessian}) = \begin{bmatrix} \nabla(\boldsymbol{e}_1^\top \nabla f(\boldsymbol{x})(\text{i.e. Jabobian})) \\ ... \\ \nabla(\boldsymbol{e}_n^\top \nabla f(\boldsymbol{x})) \end{bmatrix}$ 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

33 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 $q(w|\theta) \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$ with $\boldsymbol{\theta} = (\boldsymbol{\mu}, \sigma)$

Optimization

Parameters — Find parameters θ

Objective function —

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

Optimization —

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

34 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 $n-1$: $\boldsymbol{z}_k = b_k + \sum_{f_n} \sum_{f_w} \sum_{f_n'} \boldsymbol{W}_{k',k} \boldsymbol{X}_{k'}$
- 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

35 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 $\boldsymbol{h}_t$, cell output $\boldsymbol{y}_t$
  - Output layer
- Output of neuron in layer n:
  $\boldsymbol{Y}_t = \phi\left(\boldsymbol{X}_t \boldsymbol{W}_x + \boldsymbol{H}_{t-1} \boldsymbol{W}_y + \boldsymbol{b}\right)\boldsymbol{V} = \phi\left(\begin{bmatrix} \boldsymbol{X}_t \\ \boldsymbol{H}_{t-1} \end{bmatrix} \boldsymbol{W} + \boldsymbol{b}\right)\boldsymbol{V}$
  - $\boldsymbol{V}$ is optional: If $\boldsymbol{V} = \boldsymbol{I}$ (identity matrix), $\boldsymbol{H}_t = \boldsymbol{Y}_t$, this is assumed in the following
  - $\boldsymbol{Y}_t$ is an $n_{\text{instances}} \times n_{\text{neurons}}$ matrix containing layer outputs for instances in the mini-batch at time t
  - $\boldsymbol{X}_t$ is an $n_{\text{instances}} \times m$ matrix containing encoded inputs for all instances in the mini-batch
  - $\boldsymbol{H}_{t-1}$ is an $n_{\text{instances}} \times n_{\text{neurons}}$ matrix containing cell state outputs for instances in the mini-batch at time $t-1$
  - $\boldsymbol{W}_x$ is an $m \times n_{\text{neurons}}$ matrix containing connection weights for $\boldsymbol{X}_t$
  - $\boldsymbol{W}_y$ is an $n_{\text{neurons}} \times n_{\text{neurons}}$ matrix containing connection weights for $\boldsymbol{Y}_{t-1}$ (resp. $\boldsymbol{H}_{t-1}$)
  - $\boldsymbol{b}$ is a vector of length n_{neurons} containing the bias term
  - $\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_x \\ \boldsymbol{W}_y \end{bmatrix}$
  - ϕ is a non-linear activation function

Optimization

Parameters — Find parameters $\theta = \boldsymbol{W}_x, \boldsymbol{W}_y, \boldsymbol{b}, \boldsymbol{X}_t$, 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:
  - $\boldsymbol{y} = \varphi(\boldsymbol{X}_h \boldsymbol{W}_h)$
  - $\nabla_{\boldsymbol{W}_h} L \propto \sum_{k=1}^{t} (\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}) \frac{\partial h_k}{\partial \boldsymbol{W}_k}$
  - $\frac{\partial h_{i+k}}{\partial h_i} = \prod_{j=0}^{k-1} \frac{\partial h_{i+k-j}}{\partial h_{i+k-j-1}}$
- Perform gradient descent to find best weights

36 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

37 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
  - $\text{Multi Head Attention} Z = \text{Concat}(Z_{\text{head}_1}, \dots, Z_{\text{head}_h}) W_O + b_O$ where
    - $\text{Concat}(...)$ 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: $\text{Attention} \Pi Z = \Pi \ \text{Attention} Z$
- The self-attention is given by: $A = Z W_q W_k^\top Z^\top$
- After permutation, self-attention is given by:
  $A' = (\Pi Z) W_q W_k^\top (\Pi Z)^\top = \Pi Z W_q W_k^\top Z^\top \Pi^\top = \Pi (Z W_q W_k^\top Z^\top) \Pi^\top = \Pi A \Pi^\top$
- Applying softmax:
  $\text{softmax}(A') = \text{softmax}(\Pi A \Pi^\top) = \Pi \ \text{softmax}(A) \ \Pi^\top$ 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:
  $Z' = \text{softmax}(A')(\Pi Z) W_v = \Pi \ \text{softmax}(A) \ \Pi^\top (\Pi Z) W_v = \Pi \ \text{softmax}(A)(\Pi^{-1} \Pi) Z W_v = \Pi \ \text{softmax}(A) Z W_v$ 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: $\text{Attention} \Pi Z = \text{Attention} Z$
- 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

38 Positional Embeddings

- Can be absolute or relative
- Attention with absolute positional encodings:
  $A_{q,k}^{\text{absolute}} = \left(Z_q + P_q\right) W_q W_k^\top (Z_k + P_k)^\top = Z_q W_q W_k^\top Z_k^\top + Z_q W_q W_k^\top P_k^\top + P_q W_q W_k^\top Z_k^\top + P_q W_q W_k^\top P_k^\top$
- Attention with relative positional encodings, where relative difference $\delta = q - k$:
  $A_{q,k}^{\text{relative}} := Z_q W_q W_k^\top Z_k^\top + Z_q W_q \widetilde{W}_k r_\delta + u^\top W_k Z_k + v^\top \widetilde{W}_k r_\delta$
  where *Gaussian encodings* are given by parameters
  - $W_q = W_k = 0$

- $\widetilde{\boldsymbol{W}}_k = \boldsymbol{I}$
- $\boldsymbol{r}_\delta = \begin{pmatrix} \|\delta\|^2 \\ \delta_1 \\ \delta_2 \end{pmatrix}$
- $\boldsymbol{v} = -\alpha \begin{pmatrix} 1 \\ -2\Delta_1 \\ -2\Delta_2 \end{pmatrix}$
- $\boldsymbol{v}$ and $\boldsymbol{r}_\delta$ 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 $\boldsymbol{Z}$ for relative encodings has the same complexity as for absolute encodings, thus diminishing the benefit

39 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 $\boldsymbol{h}_n^{(e)} = f(\boldsymbol{W}_1^{(e)} \boldsymbol{h}_{n-1}^{(e)} + \boldsymbol{W}_2 \boldsymbol{w}_n)$ where
      - f is activation function
      - $\boldsymbol{w}_n$ is input token embedding at time step n in input sequence
      - $\boldsymbol{h}_{n-1}^{(e)}$ is encoder hidden state from previous time step
  - Outputs from encoder to decoder are weighted by attention weights:
    - Context vector $\boldsymbol{z}_m = \sum_{n=1}^{N} \alpha_{m,n} \boldsymbol{h}_n^{(e)}$ where
      - $\boldsymbol{h}_n^{(e)}$ 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
      - $\alpha_{m,n} = \text{softmax}(\boldsymbol{h}_{m-1}^{(d)} \times [\boldsymbol{h}_1^{(e)}, ..., \boldsymbol{h}_N^{(e)}])$ where
      - $\boldsymbol{h}_{m-1}^{(d)}$ is previous decoder hidden state (= Q)
      - $[\boldsymbol{h}_1^{(e)}, ..., \boldsymbol{h}_N^{(e)}]^\top$ 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 $\boldsymbol{h}_m^{(d)} = f(\boldsymbol{W}_3^{(d)} \boldsymbol{h}_{m-1}^{(d)} + \boldsymbol{W}_2^{(d)} \boldsymbol{w'}_{m-1} + \boldsymbol{W}_1^{(d)} \boldsymbol{z}_m)$ where
      - f is activation function
      - $\boldsymbol{w'}_{m-1}$ is target token embedding at time step $m-1$ in target sequence
      - $\boldsymbol{h}_{m-1}^{(d)}$ is decoder hidden state from previous time step with $\boldsymbol{h}_0^{(d)} = \boldsymbol{h}_N^{(e)}$, i.e. last encoder output is first decoder input
      - $\boldsymbol{z}_m$ is cross-attention
- Runtime analysis:
  - Let $\boldsymbol{W}^{(e)}\boldsymbol{h}$ be in $((N \times d) \times (d \times d))$ resp. $\boldsymbol{W}^{(d)}\boldsymbol{h}$ in $((M \times d) \times (d \times d))$
  - Let number of encoder resp. decoder layers be l_e, l_d
  - We perform $\boldsymbol{z}_m = \sum_{n=1}^{N} \alpha_{m,n} \boldsymbol{h}_n^{(e)}$ for M decoder time steps, summing over N encoder outputs $\boldsymbol{h}_n^{(e)}$ 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 $\theta = \boldsymbol{W}_1^{(e)}, \boldsymbol{W}_2^{(e)}, \boldsymbol{W}_1^{(d)}, \boldsymbol{W}_2^{(d)}, \boldsymbol{W}_3^{(d)}$

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:
  - $\nabla_{\boldsymbol{h}_1^{(e)}} L = \frac{\partial L}{\partial \boldsymbol{h}_m^{(d)}}) \frac{\partial \boldsymbol{h}_m^{(d)}}{\partial \boldsymbol{h}_n^{(e)}}$
  - $\frac{\partial \boldsymbol{h}_m^{(d)}}{\partial \boldsymbol{h}_n^{(e)}} = \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \boldsymbol{W}_3^{(d)} \boldsymbol{h}_{m-1}^{(d)} \times \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \boldsymbol{W}_1^{(d)} \boldsymbol{z}_m \times \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} f(\boldsymbol{W}_3^{(d)} \boldsymbol{h}_{m-1}^{(d)} + \boldsymbol{W}_2^{(d)} \boldsymbol{w'}_{m-1} + \boldsymbol{W}_1^{(d)} \boldsymbol{z}_m)$
  - We can further decompose $\frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \boldsymbol{W}_1^{(d)} \boldsymbol{z}_m$:
    - $= \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \alpha_{m,n} \boldsymbol{h}_n^{(e)} + \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \sum_{i \neq n}^{N} \alpha_{m,i} \boldsymbol{h}_i^{(e)}$
      $= \alpha_{m,n} + \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \alpha_{m,n} \boldsymbol{h}_n^{(e)} + \frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \sum_{i \neq n}^{N} \alpha_{m,i} \boldsymbol{h}_i^{(e)}$ due to product rule for $\alpha_{m,n} \boldsymbol{h}_n^{(e)}$
      $= \Phi_{m,n} + \Phi'_{m,n} \boldsymbol{h}_n^{(e)} + \sum_{i \neq n}^{N} \left[\Phi_{m,i} \frac{\partial \boldsymbol{h}_i^{(e)}}{\partial \boldsymbol{h}_n^{(e)}} + \Phi'_{m,i} \boldsymbol{h}_i^{(e)} \right]$ due to product rule for $\alpha_{m,i} \boldsymbol{h}_i^{(e)}$ and by replacing $\alpha_{m,n}$ with $\Phi_{m,n}$
  - Runtime analysis:
    - $\frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \boldsymbol{W}_3^{(d)} \boldsymbol{h}_{m-1}^{(d)}$ is in $O(m \times N + N - n)$:
      - Taking derivatives of m decoder steps, due to attention $\boldsymbol{z}_m$ which is applied over N encoder outputs, takes $O(m \times N)$
      - Taking derivatives of $N - n$ encoder steps takes $O(N - n)$
    - $\frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} \boldsymbol{W}_1^{(d)} \boldsymbol{z}_m$ is in $O(N)$:
    - $\frac{\partial}{\partial \boldsymbol{h}_n^{(e)}} f(...)$ is in $O(N)$:
      - $\Phi_{m,n}$ and $\Phi'_{m,n}$ are in $O(1)$, since they don't contain $\boldsymbol{h}_n^{(e)}$
      - $\sum_{i \neq n}^{N} \Phi_{m,i} \frac{\partial \boldsymbol{h}_i^{(e)}}{\partial \boldsymbol{h}_n^{(e)}}$ 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
      - $\sum_{i \neq n}^{N} \Phi'_{m,i} \boldsymbol{h}_i^{(e)}$ is in $O(N)$, since here we're summing over all N encoder steps
      - $\boldsymbol{W}_3^{(d)} \boldsymbol{h}_{m-1}^{(d)}$ and $\boldsymbol{W}_2^{(d)} \boldsymbol{w'}_{m-1}$ are in $O(1)$, since they don't contain $\boldsymbol{h}_n^{(e)}$
      - $\boldsymbol{W}_1^{(d)} \boldsymbol{z}_m$ 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

40 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, $\boldsymbol{X}$ in $(N \times d_{model})$) and positional embeddings (sinusoidal pointer vector for word position, given that model is not sequential, $\boldsymbol{P}$ in $(N \times d_{model})$) and adds them: $\boldsymbol{H}_0^{(e)} = \boldsymbol{X} + \boldsymbol{P}$
    - Multi-head self-attention, applied to all tokens jointly, where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are tokens in input sequence:
      - $\boldsymbol{Q} = \boldsymbol{H}_{(l-1)}^{(e)} \boldsymbol{W}_q$
      - $\boldsymbol{K} = \boldsymbol{H}_{(l-1)}^{(e)} \boldsymbol{W}_k$
      - $\boldsymbol{V} = \boldsymbol{H}_{(l-1)}^{(e)} \boldsymbol{W}_v$
      - Attention$\boldsymbol{Z}$ = softmax$\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}} \right) \boldsymbol{V}$
      - Multi Head Attention$\boldsymbol{Z}$ = Concat$(\boldsymbol{Z}_{\text{head}_1}, \ldots, \boldsymbol{Z}_{\text{head}_h}) \boldsymbol{W}_O + \boldsymbol{b}_O$
    - Addition and normalization: Skip connections (from token + positional embeddings) added back and normalized: $\boldsymbol{H}_l^{(e)}$ = Layer Norm(Multi Head Attention$(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) + \boldsymbol{H}_{(l-1)}^{(e)}$)
    - Feed-forward network, parallel for each token: FFN$(\boldsymbol{H}_l^{(e)})$ = ReLU$(\boldsymbol{H}_l^{(e)} \boldsymbol{W}_1 + \boldsymbol{b}_1) \boldsymbol{W}_2 + \boldsymbol{b}_2$ where
      - $\boldsymbol{W}_1 \in \mathbb{R}^{(d_v \times r)}$
      - $\boldsymbol{b}_1 \in \mathbb{R}^{(1 \times r)}$
      - $\boldsymbol{W}_2 \in \mathbb{R}^{(r \times d_v)}$
      - $\boldsymbol{b}_2 \in \mathbb{R}^{(1 \times d_v)}$
    - Addition and normalization: Skip connections (from first addition and normalization) added back and normalized: $\boldsymbol{H}_l^{(e)}$ = Layer Norm(FFN$(\boldsymbol{H}_l^{(e)}) + \boldsymbol{H}_l^{(e)}$)
    - Generates hidden states $\boldsymbol{h}_n^{(e)}$
  - 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, $\boldsymbol{Y}$ in $(M \times d_{model})$) and positional embeddings (sinusoidal pointer vector for word position, given that model is not sequential, $\boldsymbol{P}$ in $(M \times d_{model})$) and adds them: $\boldsymbol{H}_0^{(d)} = \boldsymbol{Y} + \boldsymbol{P}$
      - Masked self-attention, applied to all tokens jointly, where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are tokens in target sequence:
        - $\boldsymbol{Q} = \boldsymbol{H}_{(l-1)}^{(d)} \boldsymbol{W}_q$

      - $K = H_{(l-1)}^{(d)} W_k$
      - $V = H_{(l-1)}^{(d)} W_v$
      - $\text{Masked Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \text{mask}\right) V$ 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: $x, \langle y_t, ..., y_{t-1} \rangle$)
    - 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

41 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 $\delta \neq \Delta$

42 Variational Autoencoders (VAE)

Description

Task — Autoencoders as representation learning (learn to copy inputs to outputs)

Description — Variational autoencoders as probabilistic autoencoders

Formulation

Formulation —

- Model architecture:
  - Encoder / recognition network: Converts input to latent representation
  - Decoder / generative network: Converts latent representation to output
- Assumes data x is generated by latent variable h: $p(x, h) = p_\theta(x|h) \times p_{\theta'}(h)$
- Prior $p_{\theta'}(h) \sim \mathcal{N}(0, \sigma^2 I)$
- Encoder produces a probability distribution $q_\Phi(h|x)$ which approximates the true posterior $p_{\theta,\theta'}(h|x)$ since the true posterior is intractable:
  - Produces $\mu_{h|x}$ and $\Sigma_{h|x}$ according to which h is approximately distributed in latent space
- Decoder reconstructs input $p_\theta(x|h)$ by sampling from latent space $h \sim q_\Phi(h|x)$:
  - Produces $\mu_{x|h}$ and $\Sigma_{x|h}$ according to which $\hat{x}$ is approximately distributed in output space

Optimization

Parameters — Find parameters θ' (prior), θ (likelihood), Φ (approximate posterior)

Objective function —

- Maximize log likelihood $\sum_{i=1}^n \log p_{\theta,\theta'}(x^{(i)})$
- Challenge: This does not involve the encoder
- Solution: $\arg\max_{\theta,\theta',\Phi} \sum_{i=1}^n \mathbb{E}_{q(h|x)} \log[\frac{p_{\theta,\theta'}(x^{(i)}, h)}{p_{\theta,\theta'}(h|x^{(i)})} \times \frac{q_\Phi(h|x^{(i)})}{q_\Phi(h|x^{(i)})}] = \sum_{i=1}^n \mathbb{E} \log[\frac{p_{\theta,\theta'}(x^{(i)}, h)}{q_\Phi(h|x^{(i)})}] + \mathbb{E} \log[\frac{q_\Phi(h|x^{(i)})}{p_{\theta,\theta'}(h|x^{(i)})}] = \sum_{i=1}^n \text{ELBO}_{\theta,\theta',\Phi}(x^{(i)}) + \text{KL divergence}(q_\Phi(\cdot|x^{(i)}) | p_{\theta,\theta'}(\cdot|x^{(i)}))$
- ELBO provides lower bound of log likelihood: $\log p_{\theta,\theta'}(x^{(i)}) \geq \text{ELBO}_{\theta,\theta',\Phi}(x^{(i)}) = \mathbb{E} \log[\frac{p_{\theta,\theta'}(x^{(i)}, h)}{q_\Phi(h|x^{(i)})}] = \mathbb{E} \log[\frac{p_{\theta'}(h) \times p_\theta(x^{(i)}|h)}{q_\Phi(h|x^{(i)})}] = \mathbb{E} \log[p_\theta(x^{(i)}|h)] + \mathbb{E} \log[\frac{p_{\theta'}(h)}{q_\Phi(h|x^{(i)})}] =$ mutual information resp. cross-entropy from decoder − KL divergence $(q_\Phi(\cdot|x^{(i)}) | p_{\theta'}(\cdot))$ from encoder
- ELBO can also be formulated as $\text{ELBO}_{\theta,\theta',\Phi}(x^{(i)}) = \mathbb{E} \log[\frac{p_{\theta,\theta'}(x^{(i)}, h)}{q_\Phi(h|x^{(i)})}] = \mathbb{E} \log[p_{\theta,\theta'}(x^{(i)}, h)] - \mathbb{E} \log[q_\Phi(h|x^{(i)})] = \mathbb{E} \log[p_{\theta,\theta'}(h|x^{(i)}) p_{\theta,\theta'}(x^{(i)})] - \mathbb{E} \log[q_\Phi(h|x^{(i)})] = \mathbb{E} \log[p_{\theta,\theta'}(h|x^{(i)})] + \mathbb{E} \log[p_{\theta,\theta'}(x^{(i)})] - \mathbb{E} \log[q_\Phi(h|x^{(i)})] = -\mathbb{E} \log[\frac{q_\Phi(h|x^{(i)})}{p_{\theta,\theta'}(h|x^{(i)})}] + \mathbb{E} \log[p_{\theta,\theta'}(x^{(i)})]$
- Mutual information enforces infomax principle of ANNs
- KL divergence acts as regularization

43 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

44 Diffusion Models

Description

Task — Generate new samples

Formulation

Formulation —

- Model architecture:
  - VAE for compression: Compress original data into latent space
  - Forward diffusion: Gradually add Gaussian noise to latent representation
  - Reverse diffusion: Gradually remove noise from latent representation, potentially guided by condition (e.g. text input), until latent representation is restored
  - Decoder: Decodes latent representation back to original data
- Generally implemented as a U-Net

Components

Forward diffusion —

- Real data: $x_0 \sim p(x)$
- Gaussian noise: $x_T \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$
- $x_0 \to x_1 \to ... \to x_T$
- Stochastic differential equation: $dx = f(x,t)dt + g(t)dW$
  - x : Variable being diffused
  - $f(x,t)$: Deterministic *drift term*: Represents the mean rate of change of x at time t
  - $g(t)$: *Diffusion coefficient*: Function that scales the magnitude of the noise
  - dW : Random *Wiener process resp. Brownian motion*: Represents random fluctuations
  - Function shows how x changes over time due to deterministic and random influences
- Two transitions:
  - $q(x_t|x_{t-1}) \sim \mathcal{N}(x_t|\sqrt{1-\beta} x_{t-1}, \beta_t \boldsymbol{I})$
    - β_t : Added noise
  - $q(x_t|x_0) \sim \mathcal{N}(x_t|\sqrt{\overline{\alpha}_t} x_0, (1-\overline{\alpha}_t)\boldsymbol{I})$
    - $\alpha_t = 1 - \beta_t$: Retained signal
    - $\overline{\alpha}_t = \prod_{s \leq t} \alpha_t$: Cumulative retained signal until t
    - $1 - \overline{\alpha}_t$: Cumulative added noise until t

Reverse diffusion —

- $x_T \to x_{T-1} \to ... \to x_0$
- Stochastic differential equation: $dx = [f(x,t) - g(t)^2 \nabla_x \log p(x,t)]dt + g(t)dW$
  - x : Variable being diffused
  - $f(x,t)$: Deterministic *drift term*: Represents the mean rate of change of x at time t
  - $\nabla_x \log p(x,t)$: Score function: Captures gradient of log probability of system state at time t. Guides the system to reverse the noise by pulling the state x to higher-probability regions.
  - $g(t)^2$: Function that scales the impact of the score function
  - $g(t)$: Function that scales the magnitude of the noise
  - dW : Random *Wiener process resp. Brownian motion*: Represents random fluctuations
  - Function shows how x changes over time due to deterministic and random influences
- To approximate the score function resp. predict the noise, we train a model $f_\theta(x,t) \approx \nabla_x \log p(x,t)$:
  - $f_\theta(x,t) : \mathcal{X} \times [0,1] \to \mathcal{X}$, where $t \in [0,1]$ and $x \sim p(x)$
  - Proof that $f_\theta(x,t)$ approximates the score function and predicts the noise:
    - Gradient of log probability of Gaussian is: $\frac{-(x_t - \mu)}{\sigma^2}$
    - Therefore and based on $q(x_t|x_0)$ from forward diffusion, score is given by: $\nabla_x \log p(x,t) = \frac{-(x_t - \sqrt{\overline{\alpha}_t} x_0)}{(1-\overline{\alpha}_t)}$
    - x_0 is unknown and must be estimated as $\hat{x}_0$
    - Then, we have $\nabla_x \log p(x,t) = \frac{-(x_t - \sqrt{\overline{\alpha}_t} \hat{x}_0)}{(1-\overline{\alpha}_t)}$, where $f_\theta(x,t) = (x_t - \sqrt{\overline{\alpha}_t} \hat{x}_0)$, which is the predicted noise
    - From this we see, that $f_\theta(x,t)$ is equal to $\nabla_x \log p(x,t)$ up to a scaling factor $-\frac{1}{(1-\overline{\alpha}_t)}$
  - $f_\theta(x,t)$ is trained to minimize the difference between the added and predicted noise: $min_\theta \mathbb{E}_{x,\epsilon,t} \|\epsilon - f_\theta(x + \sigma_t \epsilon, t)\|^2$
- Reconstruction formula: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}} \epsilon_\theta(x_t, t)) + \sigma_t z$
  - $\frac{1}{\sqrt{\alpha_t}}$: Scaling factor
  - $\frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}} \epsilon_\theta(x_t, t)$: Predicted noise
  - In formula, predicted noise is subtracted from x_t, after which a small amount of random noise $\sigma_t z$ is re-introduced to maintain stochasticity

Compression —

- Motivation: Images often have high dimensionality, which can be easily compressed, since there are many redundant dimensions that do not contain unique information
- VAE first encodes x into latent representation z and then decodes z into original representation $\hat{x}$

Conditional denoising —

- Extension of diffusion model to include additional condition y, e.g. a text prompt, and guiding the denoising process such that the generated sample $\hat{x}$ aligns with y
- VAE first encodes x into latent representation z and then decodes z into original representation $\hat{x}$
- To approximate the score function resp. predict the noise, we train a model $f_\theta(x,y,t) \approx \nabla_x \log p(x,t)$:
  - $f_\theta(x,y,t) : \mathcal{X} \times \mathcal{Y} \times [0,1] \to \mathcal{X}$, where $t \in [0,1]$ and $x \sim p(x)$
  - $f_\theta(x,y,t)$ is trained to minimize the difference between the added and predicted noise: $min_\theta \mathbb{E}_{x,y,\epsilon,t} \|\epsilon - f_\theta(x + \sigma_t \epsilon, y, t)\|^2$
- Condition y is injected via:
  - Concatenating y with latent feature maps
  - Cross-attention:
    - Dynamically weighs importance of different input features by paying attention to specific regions of the feature map based on guidance
    - Queries from latent feature map z
    - Keys and values from condition y
    - Dimensionality analysis:
      - Latent feature map z: $B \times C \times h \times w$
      - Text embedding y: $B \times M \times D_E$
      - W_q: $F \times C \times D_K \times f_h \times f_w$
      - Q: $B \times F \times h \times w \times D_K$
      - W_k: $F \times D_E \times D_K$
      - K: $B \times F \times M \times D_K$
      - W_v: $F \times D_E \times D_V$
      - V: $B \times F \times M \times D_V$
      - P and S: $B \times F \times h \times w \times M$
      - SV: $B \times F \times h \times w \times D_V$
    - When y is text embedding, but z is image embedding, we need *CLIP resp. contrastive learning* to match text and image:
      - Let true caption embeddings be $[t_1, ..., t_N]$ and corresponding image embeddings be $[i_1, ..., i_N]$
      - Matrix representation given by $\begin{bmatrix} t_1 \cdot i_1 & ... & t_N \cdot i_1 \\ ... & ... & ... \\ t_1 \cdot i_N & ... & t_N \cdot i_N \end{bmatrix}$
      - Aim is to maximize similarity between image and its true caption on diagonal, but minimize other pairs
  - Spatial transformer: Dynamically transforms spatial properties of an input (e.g. scale, rotation, location) to focus on most relevant parts

U-Net —

- Architecture:
  - CNN (encoder): Downsamples image to create compressed, low-dimensional representation, uses pooling and ReLu
  - Inverted CNN (decoder): Upsamples compressed, low-dimensional representation to generate image, uses upsampling and softmax
- Key operations:
  - Skip connections going directly from encoder to decoder
  - Convolution operation: Apply spatial filters to extract spatial features (e.g. edges, textures, patterns)
  - Down- and upsampling: Reduce spatial dimensions of input (e.g. via max-pooling, strided convolutions) resp. reconstruct spatial resolution (e.g. via transposed convolution, interpolation), low-level details (e.g. edges) captured at shallow layers, high-level patterns (e.g. objects) captured at deeper layers

45 Other

Causal Models

Basics —

- Correlation / prediction does not necessarily imply causality, but causality necessarily implies correlation / predictability
- If A and O are correlated, there are 4 possibilities:
  - *Chain*:
    - A causes O
    - O causes A
  - *Fork*:
    - A and O have a common cause C
  - A and O do not have a causal relationship and are correlated by chance
- If upon intervention A, O happens, there are 3 possibilities:
  - A directly causes O, i.e., O happens upon intervention A if all other variables are kept constant
  - A indirectly causes O, i.e., O happens upon intervention A if all prior and simultaneous variables are kept constant
  - A does not cause O, i.e., O does not happen upon intervention A if all prior and simultaneous variables are kept constant

- Causality can be inferred only when controlling for all prior and simultaneous variables
- If we want to model the causal relationship between A and O, we need to consider the following:
  - A might be imperfectly correlated with B, which also (in)directly affects O:
    - Regression coefficient for A can carry unwanted proxy effects of B if used alone in a model. Instead, it is desirable to single out direct effects of each factor if all other prior and simultaneous variables are kept constant
    - Options:
      - Chain:
      - A causes B
      - B causes A
      - Fork:
      - A and B have a common cause C
      - A and B do not have a causal relationship and are correlated by chance
    - *Confounding variables* (i.e., prior variables that A is caused by and simultaneous variables that A is correlated with by chance, which also affect O) should be included in the model to avoid *omitted variable bias*:
      - Variable B, if it causes A
      - Common cause C of B and A
      - Variable B, if it has no causal relationship to A
      - If variable B only indirectly affects O, either variable B or an intermediate between B and O can be included in the model
    - *Intermediate* and *collider variables* (i.e., variables caused by A and O, creating a *selection bias*) should be excluded from the model to avoid *overcontrol* and *endogenous selection bias*:
      - Variable B, if it is caused by A
      - Variable D, if it is caused by A and O
  - A might be perfectly correlated with B, which also (in)directly affects O: Causal effects of each variable cannot be separated
  - A might be perfectly uncorrelated with B, which also (in)directly affects O:
    - A and B do not have a causal relationship
    - B does not impact the regression coefficient for A and can be excluded from the model
  - *Shortcut learning*: A might be spuriously correlated with B, which does not affect O
    - Important to only encode A in the features (i.e., generate an invariant representation), not B
    - Aim is to represent counterfactually invariant relationships between A and O, i.e., invariant to different counterfactuals represented by different states of B

Causal scenarios —

- *Causal scenario without selection bias*: $\mathcal{X}$ affects $\mathcal{Y}$ and there is no selection bias. Features $\mathcal{X}$ can be grouped into:
  - Features $\mathcal{X}_{\perp Y}$ do not causally affect $\mathcal{Y}$, but are affected by $\mathcal{W}$. These features have no causal relationship
  - Features $\mathcal{X}_{\perp W}$ causally affect $\mathcal{Y}$, but are not affected by $\mathcal{W}$. These features have a direct causal relationship
  - Features $\mathcal{X}_{W\&Y}$ causally affect $\mathcal{Y}$ and are affected by $\mathcal{W}$ (as well as $\mathcal{X}_{\perp Y}$ and $\mathcal{X}_{\perp W}$). These features link $\mathcal{W}$ and $\mathcal{Y}$ in a causal network
- *Anti causal scenario*: We assume $\mathcal{Y}$ affects $\mathcal{X}$, rather than the other way around
- *Causal scenario with selection bias*: $\mathcal{X}$ affects $\mathcal{Y}$ and there is a selection bias

Counterfactual invariance —

- Results of estimator remain consistent across different counterfactual scenarios, i.e. if $\mathcal{X} \to \mathcal{Y}$ and $\mathcal{W} \to \mathcal{X}$ but $\mathcal{W} \not\to \mathcal{Y}$ where $\to$ is a causal relationship, our estimator should be invariant to states of $\mathcal{W}$, i.e. $f(\mathcal{X}(\mathcal{W}_1)) = f(\mathcal{X}(\mathcal{W}_2))$
- For counterfactual invariance, the following must hold:
  - Anti causal scenario: $(f(\mathcal{X}) \perp \mathcal{W}) | \mathcal{Y}$, i.e. estimate f only depends on $\mathcal{X}_{\perp W}$, provided $\mathcal{Y}$ is known
  - Causal scenario without selection bias, potentially with confoundedness: $f(\mathcal{X}) \perp \mathcal{W}$, i.e. estimate f only depends on $\mathcal{X}_{\perp W}$
  - Causal scenario without confoundedness, potentially with selection bias: $(f(\mathcal{X}) \perp \mathcal{W}) | \mathcal{Y}$ as long as $\mathcal{X}_{\perp Y}$ and $(\mathcal{Y} \perp \mathcal{X}) | \mathcal{X}_{\perp W}, \mathcal{W}$
- We need to show:
  - For causal scenario without selection bias we need to show: $\mathcal{X}_{\perp W} \perp \mathcal{W}$
  - For anti causal scenario we need to show: $(\mathcal{X}_{\perp W} \perp \mathcal{W}) | \mathcal{Y}$
  - This can be shown via *d-separation*

D separation —

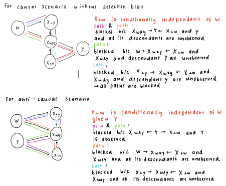
- Undirected path of n nodes is *d-separated*, if it contains 3 nodes following any of the following forms and if this form is *blocked*:
  - *Chain structure*: $X \to Z \to Y$ or $Y \to Z \to X$ – is blocked, if we condition on Z, i.e. Z is known
  - *Fork structure*: $X \leftarrow Z \to Y$ – is blocked, if we condition on Z, i.e. Z is known
  - *Collider structure*: $X \to Z \leftarrow Y$ – is blocked, if we don't condition on Z or any of its descendants
- Random variables X and Y are conditionally independent if each path between them is d-separated
  $\to$ as soon as we have one blocked triple on path, entire path is blocked
  $\to$ as soon as one path is active, we cannot guarantee conditional independence
- For causal scenario without selection bias we can show $\mathcal{X}_{\perp W} \perp \mathcal{W}$ since all paths are blocked
- For anti causal scenario we can show $(\mathcal{X}_{\perp W} \perp \mathcal{W}) | \mathcal{Y}$ since all paths are blocked, conditioned on $\mathcal{Y}$, i.e. if $\mathcal{Y}$ is known

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 $n = k + 1$

1) D separation



For causal scenario without selection bias

X⊥w is conditionally independent of W
path & path :
blocked b/c Xw&y → Y ← X⊥w and y and all its descendants are unobserved
path :
blocked b/c W → Xw&y ← X⊥w and Xw&y and descendant Y are unobserved
path :
blocked b/c X⊥y → Xw&y ← X⊥w and Xw&y and descendant y are unobserved
→ all paths are blocked

For anti-causal Scenario

X⊥w is conditionally independent of W given Y
path & path :
blocked b/c Xw&y ← Y → X⊥w and Y is observed
path :
blocked b/c W → Xw&y ← X⊥w and Xw&y and all its descendants are unobserved
path :
blocked b/c X⊥y → Xw&y ← X⊥w and Xw&y and all its descendants are unobserved